Competitive Strategies for Online Cloud Resource Allocation with Discounts

The 2-Dimensional Parking Permit Problem

Xinhui Hu[1], Arne Ludwig[2], Andrea Richa[1], Stefan Schmid[2,3]
[1] Computer Science, SCIDSE, Arizona State University, Tempe, AZ 85287, USA
[2] TU Berlin, Germany [3] Telekom Innovation Laboratories (T-Labs), Germany

***Abstract*—Cloud computing heralded an era where resources can be scaled up and down elastically and in an online manner. This paper initiates the study of cost-effective cloud resource allocation algorithms under *price discounts*, using a competitive analysis approach. We show that for a single resource, the online resource renting problem can be seen as a 2-dimensional variant of the classic online parking permit problem, and we formally introduce the PPP^2 problem accordingly. Our main contribution is an online algorithm for PPP^2 which achieves a deterministic competitive ratio of k (under a certain set of assumptions), where k is the number of resource bundles. This is almost optimal, as we also prove a lower bound of $k/3$ for any deterministic online algorithm. Our online algorithm makes use of an optimal offline algorithm, which may be of independent interest since it is the first optimal offline algorithm for the 1D and 2D versions of the parking permit problem. Finally, we show that our algorithms and results also generalize to multiple resources (i.e., multi-dimensional parking permit problems).**

I. Introduction

As the Internet becomes increasingly virtualized, resources can be allocated more flexibly and at large scale. Virtualization not only introduces an Internet-wide resource market, but also new business models. For example, a startup company running a webservice no longer needs to invest in its own infrastructure, but can dynamically lease cloud resources to provide the service to its users in a cost-efficient manner. Also a hybrid model is possible where the cloud resources are just used to complement a limited own infrastructure in peak demand times (a.k.a. "cloud bursting"). New business models are also introduced by resource brokering opportunities: a broker may lease a large amount of resources from different providers and resell them to its customers (at a higher price).

This paper studies the problem of a (cloud) *customer* who rents resource bundles from a (cloud) *provider*, in order to offer a certain service to its users (or to resell the resources). The customer is faced with the challenge that its resource demand is not known in advance (e.g., it depends on the popularity of its website). In order to ensure that its resource demand is satisfied at any time, and in order to avoid a costly over-provisioning of the service, additional resources must be bought in an *online* manner. The online resource allocation problem may further be complicated by the fact that the provider offers *discounts* for larger and longer resource contracts.

The goal of the customer is to come up with a smart resource renting strategy to satisfy its dynamic and unpredictable resource demand, while minimizing the overall costs of the bought resource bundles.

Our Contributions. This paper shows that at the heart of efficient cloud resource allocation lies a fundamental algorithmic problem, and makes the following contributions. We first observe that the problem of renting a single resource over time can be seen as a *2-dimensional* variant of the well-known *online Parking Permit Problem* (PPP). While in the classic parking permit problem, only the *contract durations* need to be chosen, in the 2-dimensional variant PPP^2 introduced in this paper, also the *resource rates* are subject to optimization.

Our main contribution is the deterministic online algorithm On2D whose performance is close to the one of a clairvoyant optimal offline algorithm which knows the entire resource demand in advance: given some simplifying assumptions (stated in Section II), On2D provably achieves a competitive ratio of $O(k)$, where k is the total number of available resource contracts; this is asymptotically optimal in the sense that there cannot exist any deterministic online algorithm with competitive ratio $o(k)$.

We also give a constructive proof that the offline variant of the PPP^2 problem can be solved in polynomial time, by presenting a dynamic programming algorithm Off2D accordingly. To the best of our knowledge, Off2D is also the first offline algorithm to efficiently solve PPP and PPP^2 for long enough request sequences.

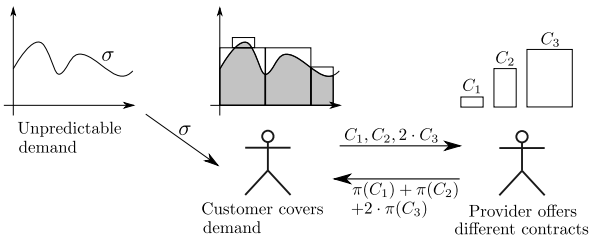

Figure 1. Overview of the model: A customer has to cover its resource demand σ by buying contracts $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ from the provider. Larger contracts C_i come with a price discount (price $\pi(C_i)$).

OFF2D is used as a subroutine in ON2D.

Finally, we show that our algorithms and results also generalize to multi-resource scenarios, i.e., to higher-dimensional parking permit problems.

Paper Organization. The remainder of this paper is organized as follows. Section II formally introduces our model. Section III presents our online algorithm. Section IV discusses an example and provides intuition for the analysis, and Section V presents the general analysis. We show that our algorithm is almost optimal by deriving a lower bound in Section VI. Section VII presents a polynomial-time optimal offline algorithm, and Section VIII shows how to extend our results from 2-dimensions to D-dimensions (for a constant D). We report on simulation results in Section IX, review related literature in Section X, and conclude our work in Section XI.

II. MODEL

We attend to the following setting (for an illustration, see Figure 1). We consider a customer with a dynamically changing resource demand. We model this resource demand as a sequence $\sigma = (\sigma_t)_t$, where σ_t refers to the resource demand at time t. We use $\widehat{\sigma}$ to denote $\max_t \sigma_t$. The customer is faced with the challenge that its future resource requirements are hard to predict, and may change in a worst-case manner over time: We are in the realm of *online algorithms and competitive analysis*.

In order to cover its resource demand, the customer buys resource contracts from a (cloud) provider. For ease of presentation, we will focus on a single resource scenario for most of our paper; however, we will also show that our results can be extended to multi-resource scenarios. Concretely, we assume that the provider offers different resource contracts $C(r, d)$ of *resource rate* r and *duration* d. We will refer to the set of available contracts by $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$. Each contract type has a *price* $\pi(C) = \pi(r, d)$, which depends on its rate $r(C) = r$ and its duration $d(C) = d$. We will assume that resource contracts have a monotonically increasing rate-duration product $r \times d$, and will denote by C_i the i^{th} largest contract.

A specific contract instance of type C_i will be denoted by $C_i^{(j)}$, for some index j. Each instance $C_i^{(j)}$ of contract type $C_i(r, d)$ has a specific start time $t_i^{(j)}$, and we will sometimes refer to a contract instance by $C_i^{(j)}(t_i^{(j)}, r_i, d_i)$. The identifiers are needed since multiple contracts of the same type can be "stacked" in our model, but will be omitted if the contract is clear from the context.

We will make three simplifying assumptions:

- A1 *Monotonic Prices:* Prices are monotonically increasing, i.e., larger contracts are more expensive than shorter contracts: $\pi(C_{i-1}) < \pi(C_i)$ since $r_{i-1} \times d_{i-1} < r_i \times d_i$ for any i.
- A2 *Multiplicity:* The duration and resource rate of a contract of type $C_i(r_i, d_i)$ are perfect multiples of the duration and rate, respectively, of contract $C_{i-1}(r_{i-1}, d_{i-1})$. That is, we assume that $d_i = x \cdot d_{i-1}$ and $r_i = y \cdot r_{i-1}$ for fixed bases $x, y \geq 2$. Moreover, without loss of generality (*w.l.o.g.*), we will assume that the smallest contract has $d_1 = 1$ and $r_1 = 1$.
- A3 *Intervals:* We assume that a contract of duration d can only be bought at time $t_0 + i \cdot d$, where $t_0 = 0$ is the start time.

Assumption A1 is natural: contracts which are dominated by larger, cheaper contracts may simply be ignored. Assumption A2 restricts the variety of available contracts the customer can choose from, and constitutes the main simplification made in this paper. Assumption A3 mainly serves the ease of presentation: as we will prove in this paper (and as it has already been shown for the classic Parking Permit Problem [13]), an offline or online algorithm limited by the interval model is at most a factor of two off the respective optimal solution in the general model.

Now, let ON be some online algorithm, let $\mathcal{C}_t(\text{ON})$ denote the set of contracts bought according to ON at time t and let $\mathcal{C}_{\leq t}(\text{ON})$ denote the set of contracts bought according to ON through time t. We will use the notation $\mathcal{C}_t^*(\text{ON}) \subseteq \mathcal{C}_{\leq t}(\text{ON})$ to describe the set of *active* contracts at time t: i.e., contracts $C_i(t_i, r_i, d_i)$ bought by ON with $t_i \leq t < t_i + d_i$. Likewise we denote the set of contracts bought by an optimal offline algorithm OFF to cover the demand prefix $\sigma_1, ..., \sigma_t$ until time t by $\mathcal{C}_{\leq t}(\text{OFF})$.

Since a correct algorithm must ensure that there are always sufficient resources to cover the current demand, the invariant $\sum_{C(r,d) \in \mathcal{C}_t^*(\text{ON})} r \geq \sigma_t$ must hold at any

moment of time t. We will use the *one-lookahead model* [3] frequently considered in online literature, and allow an online algorithm to buy a contract at time t *before* serving the request σ_t; however, ON does not have any information at all about $\sigma_{t'}$ for $t' > t$.

The goal is to minimize the overall price $\pi_\sigma(\text{ON}) = \sum_{C \in \mathcal{C}_{\leq t}(\text{ON})} \pi(C)$. More specifically, we seek to be competitive against an optimal offline algorithm and want to minimize the *competitive ratio* ρ of ON: We compare the price $\pi_\sigma(\text{ON})$ of the online algorithm ON under the external (online) demand σ, to the price $\pi_\sigma(\text{OFF})$ paid by an optimal offline algorithm OFF, which knows the entire demand σ in advance. Formally, we assume a worst-case perspective and want to minimize the (strict) competitive ratio ρ for any σ: $\rho = \max_\sigma \pi_\sigma(\text{ON})/\pi_\sigma(\text{OFF})$.

We are interested in long demand sequences σ; in particular, we will assume that the length of σ, $|\sigma|$, is at least as large as the largest single demand σ_t.

Our problem is a new variant of the classic Parking Permit Problem PPP [13], which we review quickly in the following. In PPP, a driver has to choose between k parking permits of different durations and costs in order to satisfy all of his/her parking needs while minimizing the total cost paid. More precisely, the driver has a sequence of days when he/she needs a parking space at a parking garage and there are k different parking permits, where each parking permit P_i allows the driver to use one parking space for d_i consecutive days at a cost of c_i. In the online version of the problem, the sequence of days when the driver will need a parking space is not known in advance.

III. COMPETITIVE ONLINE ALGORITHM

This section presents the deterministic online algorithm ON2D for the PPP^2 problem. As a subroutine, in order to determine which contracts to buy at time t, ON2D uses an optimal offline algorithm OFF2D that computes optimal contracts for a prefix $\sigma_{\leq t}$ of the demand through time t. In this section, we will treat OFF2D as a black box, but we will describe a polynomial-time construction later in Section VII of the paper.

In order to formally describe and analyze our algorithm, we propose a scheme that assigns bought contracts to the 2-dimensional *time-demand plane*. Our model requires that each point below the demand curve σ is covered by a contract, i.e., the mapping of contracts to demand points must be surjective.

We pursue the following strategy to assign contracts to the time-demand plane: at any time t, we order the set of active contracts by their duration, and stack the active contracts in such a way that longer contracts are embedded lower in the plane, i.e., the longest running contract $C_i(r_i, d_i)$ covers the demand from 1 to r_i, the next shorter contract $C_j(r_j, d_j)$ then covers the demand $r_i + 1$ to $r_i + r_j$, and so on. This guarantees a unique mapping of a demand point $p(time, demand)$ at time t to a contract C_i for the offline algorithm.

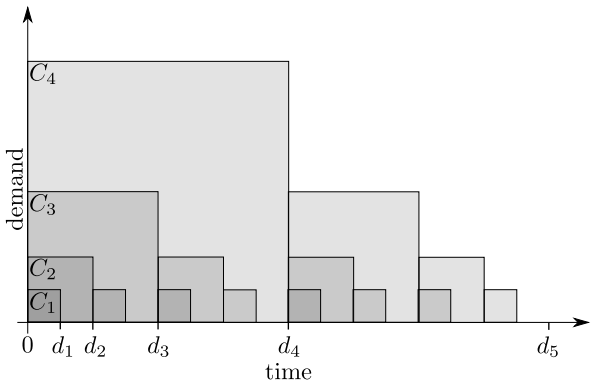

Figure 2. Worst-case example where $\sigma_t = 1 \ \ \forall t$. While OFF2D, at time d_5, buys a single contract C_5, ON2D is forced to buy all the depicted contracts, in addition to C_5. For instance, ON2D buys C_1 in every second time step.

Algorithm 1 Online Algorithm ON2D

Input: Demand prefix $\sigma_{\leq t} = \sigma_1, \sigma_2, ..., \sigma_t$; set of contracts $\mathcal{C}_{\leq t-1}(\text{ON2D})$ bought by ON2D through time $t-1$

Output: Contracts to be bought at time t: $\mathcal{C}_t(\text{ON2D})$

1: $\mathcal{C}_{\leq t}(\text{OFF2D}) \leftarrow \text{OFF2D}(\sigma_1, \sigma_2, ..., \sigma_t)$
2: **for** $C \in \mathcal{C}_{\leq t}(\text{OFF2D})$ **do**
3: **if** $\exists$ demand point p covered by C such that p is not covered by $\mathcal{C}_{\leq t-1}(\text{ON2D})$ **then**
4: $\mathcal{C}_t(\text{ON2D}).add(C)$
5: **return** $\mathcal{C}_t(\text{ON2D})$

Our online algorithm ON2D (see Algorithm 1) is based on an oracle OFF2D computing optimal offline solutions for the demand *so far*. ON2D uses these solutions to purchase contracts at time t. Concretely, ON2D mimics the offline algorithm in an efficient way, in the sense that it only buys the optimal offline contracts covering time t if the corresponding demand is not already covered by contracts bought previously by ON2D: At each time t, ON2D compares the set of previously bought contracts $\mathcal{C}_{\leq t-1}(\text{ON2D})$ with the set of contracts $\mathcal{C}_{\leq t}(\text{OFF2D})$ that OFF2D would buy for an offline demand sequence $\sigma_1, ..., \sigma_t$; ON2D then only buys the contracts $C \in \mathcal{C}_{\leq t}(\text{OFF2D})$ for the demand at time t that is not covered by $\mathcal{C}_{\leq t-1}(\text{ON2D})$.

IV. EXAMPLE

In order to provide some intuition of the behavior of ON2D, as a case study, we consider the special scenario where contracts are perfect squares, i.e., $C_i = (2^{i-1}, 2^{i-1})$, and where the contract prices have a specific discount structure, namely $\pi(C_i) = 2 \cdot C_{i-1}$, with $\pi(C_1) = 1$. This price function ensures that OFF2D will buy at most one C_j contract before it is worthwhile to buy the next larger contract C_{j+1} for the given time interval.

Let us now study the maximal cumulative price $\Pi(C_i)$. It is easy to see that under the price function above, the demand sequence σ with a constant demand of one unit per time, maximizes $\Pi(C_i)$ for $C_i = (2^{i-1}, 2^{i-1})$ and $\pi(C_i) = 2 \cdot C_{i-1}$: higher demands imply missed opportunities to charge ON2D for smaller contracts, as already a demand of two at given time t renders it worthwhile to buy C_2, and a demand of four renders it worthwhile to buy C_3, etc.

With the given demand σ, OFF2D will end up buying each of the smaller contracts once before it buys the next larger contract. The cumulative price derived from σ according to this behavior is $\Pi(C_i) = \sum_{j=1}^{i-1} \Pi(C_j) + \pi(C_i)$. We prove this claim by induction over the contract types i. For the base case $i = 1$, $\Pi(C_i) = \pi(C_i)$ holds trivially. Assuming the induction hypothesis for i we have:

$$\begin{aligned}\Pi(C_{i+1}) &= \sum_{j=1}^{i} \Pi(C_j) + \pi(C_{i+1}) \\ &= \sum_{j=1}^{i-1} \Pi(C_j) + \Pi(C_i) + \pi(C_{i+1}) \\ &\overset{Hyp.}{=} \sum_{j=1}^{i-1} \Pi(C_j) + \sum_{j=1}^{i-1} \Pi(C_j) + \pi(C_i) + \pi(C_{i+1})\end{aligned}$$

Due to the induction hypothesis, the cost of a quarter of $2^{i+1} \times 2^{i+1}$ is maximized for $\sum_{j=1}^{i-1} \Pi(C_j) + \pi(C_i)$. In order to maximize the cost in the second quarter (at the bottom of the time-demand plane) OFF2D would need to buy $\sum_{j=1}^{i-1} \Pi(C_j)$ again, and instead of buying a second contract C_i, the pricing scheme requires the purchase of contract C_{i+1}. Therefore, buying the same contracts again (despite C_i) must lead to $\Pi(C_{i+1}) = \sum_{j=1}^{i} \Pi(C_j) + \pi(C_{i+1})$.

In summary, we have derived a worst-case sequence σ for the considered price function, for which ON2D is k-competitive.

Theorem 1. *For the special setting considered in our case study,* ON2D *is k-competitive.*

Proof: Consider the discussed worst-case sequence σ, where ON2D has to buy every contract (total cost $\Pi(C_i)$) while OFF2D can simply buy C_i at price $\pi(C_i)$. We can show that $\Pi(C_i) \leq i \cdot \pi(C_i)$ and hence $\Pi(C_i) \leq k \cdot \pi(C_i)$. According to the observed behavior of OFF2D, every second contract bought by ON2D is C_1 (2^{i-2} times), every fourth is C_2 (2^{i-3} times), etc., and finally ON2D also buys C_i. See Figure 2 for an example. Thus,

$$\begin{aligned}\Pi(C_i) &= 2^{i-2} \cdot \pi(C_1) + 2^{i-3} \cdot \pi(C_2) + \cdots + 1 \cdot \pi(C_i) \\ &= 2^{i-2} \cdot 2^0 + 2^{i-3} \cdot 2^1 + \cdots + 2^{i-i} \cdot 2^{i-2} + 1 \cdot 2^{i-1} \\ &\leq 2^{i-1} + 2^{i-1} + 2^{i-1} + \cdots + 2^{i-1} + 2^{i-1} \\ &= i \cdot 2^{i-1} = i \cdot \pi(C_i)\end{aligned}$$

■

V. ANALYSIS: UPPER BOUND

With these intuitions in mind, we now present a general analysis of ON2D. First, we derive some simple properties of the contracts bought by the optimal offline algorithm OFF2D over time. Let us fix an arbitrary *demand point* p, i.e., a point below the σ-curve in the time-demand plane. We make the following claim: if p is covered by a certain contract C in $\mathcal{C}_{\leq t}(\text{OFF2D})$, p will never be covered by a smaller contract C' in $\mathcal{C}_{t'}(\text{OFF2D})$ for any $t' > t$. In other words, when considering a longer offline demand sequence $\sigma_1, \ldots, \sigma_{t'}$, OFF2D will never buy a smaller contract than C to cover the demand point p. This property of "growing contracts" together with the assumption of disjoint intervals motivates the notion of *contract independence*, which we formalize in the lemma below:

Lemma 1 (Contract Independence)**.** *Consider a demand point p_i covered by contract $C_i \in \mathcal{C}_{\leq t}(\text{OFF2D})$ and a demand point p_j covered by a distinct contract $C_j \in \mathcal{C}_{\leq t}(\text{OFF2D})$. Then there does not exist a contract $C \in \mathcal{C}_{t'}(\text{OFF2D})$ for any $t' < t$ such that p_i, p_j are covered by C. We say that the two contracts C_i and C_j are* independent*.*

Independence between contracts is trivially ensured in our model. This allows us to introduce a simple characterization of the scenarios maximizing the competitive ratio.

Lemma 2. *The competitive ratio is maximized in a scenario where* OFF2D *buys only one contract to satisfy the entire demand σ.*

Proof: By contradiction. Assume OFF2D buys more than one contract, say C_i and C_j. Now assume that over time, ON2D buys a set of (possibly smaller) contracts

$C_{i'}, C_{i''}, \ldots$ to cover the demand points of C_i and $C_{j'}, C_{j''}, \ldots$ to cover the demand points of C_j. Thus, ON2D pays $\pi(C_i)+\pi(C_{i'})+\ldots$ and $\pi(C_j)+\pi(C_{j'})+\ldots$ whereas OFF2D pays $\pi(C_i)$ and $\pi(C_j)$; the resulting competitive ratio is $\rho_{C_i} = (\pi(C_i)+\pi(C_{i'})+\ldots)/\pi(C_i)$ for the C_i part and $\rho_{C_j} = (\pi(C_j)+\pi(C_{j'})+\ldots)/\pi(C_j)$ respectively. Since all contracts in OFF2D are independent, the competitive ratio ρ of OFF2D will be $\max\{\rho_{C_i}, \rho_{C_j}\}$, which would also be the case if the larger contract was the only one bought by OFF2D. ■

We hence want to show that ON2D will never buy too many small contracts to cover a demand for which OFF2D would later only buy one contract. Concretely, let us fix any contract $C_i \in \mathcal{C}_{\leq t}(\text{OFF2D})$, and let us study the set of contracts S bought by ON2D during the time interval $[0, t)$ which overlap with C_i in the time-demand plane. Recall that S will only contain distinct instances of the contracts (since ON2D does not buy "repeated" contracts) and it will be contained in $\cup_{t'<t}\mathcal{C}_{t'}(\text{OFF2D})$. By the interval and independence property, we know that contracts in S are all "inside" C_i, i.e., do not exceed its boundary in the plane. Accordingly, we can compute an upper bound on the maximum cumulative price spent on contracts in S by ON2D while OFF2D at time t only bought a single contract C_i at price π. In the following, let us refer to this cumulative price paid by ON2D by $\Pi(C_i) = \sum_{C\in S}\pi(C)$.

Lemma 3. *The maximum cumulative price paid by* ON2D *to cover a contract C_i, $\Pi(C_i)$, is less than or equal to $i \cdot \pi(C_i)$, for any $i \geq 0$.*

Proof: Consider a contract $C_i \in \mathcal{C}_{\leq t}(\text{OFF2D})$ and S as defined above. Let ℓ be such that ON2D has bought ℓ contracts C_{i-1} to cover the area of C_i during time $[0, t)$, where $0 \leq \ell \leq \frac{\pi(C_i)}{\pi(C_{i-1})}$. For all other $C \in S$, we must have $C \in \{C_1, \ldots, C_{i-2}\}$. Let $S' = \{C \in S,\ s.t.\ C \in \{C_1, \ldots, C_{i-2}\}\}$. Hence we have $\sum_{C\in S'}\pi(C) \leq \pi(C_i) - \ell\cdot\pi(C_{i-1})$, since the area covered by all contracts in S is at most equal to the area covered by C_i, and given Assumption A1. We argue by induction on i.

Base case $i = 1$: If there is just one type of contract C_1, the online algorithm will buy the same contracts as the offline algorithm, and the claim holds trivially.

Inductive step $i > 1$: Assuming the induction hypothesis holds for all $j < i$, we have:

$$
\begin{aligned}
\Pi(C_i) &= \ell\cdot\Pi(C_{i-1}) + \pi(C_i) + \sum_{C_j^{(p)}\in S'}\Pi(C_j^{(p)}) \\
&\leq \ell\cdot(i-1)\cdot\pi(C_{i-1}) + \pi(C_i) + \sum_{C_j^{(p)}\in S'} j\cdot\pi(C_j) \\
&\leq \ell\cdot(i-1)\cdot\pi(C_{i-1}) + \pi(C_i) + (i-2)\sum_{C_j^{(p)}\in S'}\pi(C_j) \\
&\leq \ell\cdot(i-1)\cdot\pi(C_{i-1}) + \pi(C_i) + \\
&\quad + (i-2)\left[\pi(C_i) - \ell\cdot\pi(C_{i-1})\right] \\
&= \ell\cdot\pi(C_{i-1}) + (i-1)\cdot\pi(C_i) \\
&\leq \frac{\pi(C_i)}{\pi(C_{i-1})}\cdot\pi(C_{i-1}) + (i-1)\cdot\pi(C_i) \\
&= \pi(C_i) + (i-1)\cdot\pi(C_i) = i\cdot\pi(C_i)
\end{aligned}
$$

■

With these results, we can derive the competitive ratio. According to Lemma 3, for each contract $C_i^{(j)} \in \mathcal{C}_{\leq t}(\text{OFF2D})$, the accumulated cost $\Pi(C_i^{(j)})$ is bounded by $i\cdot\pi(C_i)$. Therefore, summing up all the accumulated costs of each contract in $\mathcal{C}_{\leq t}(\text{OFF2D})$, we get the total cost of ON2D at time t. Note that every contract bought by ON2D must be totally covered by contracts in $\mathcal{C}_t(\text{OFF2D})$, since $\mathcal{C}_t(\text{OFF2D})$ is an optimal solution for the entire demand sequence $\sigma_{\leq t}$ and the contract independence property holds. Since we have k different contracts and for each contract C_i in $\mathcal{C}_t(\text{OFF2D})$, we have $\Pi(C_i) \leq i\cdot\pi(C_i) \leq k\cdot\pi(C_i)$, and:

Theorem 2. ON2D *is k-competitive, where k is the total number of contracts.*

As we will show in Section VI, this is almost optimal.

Finally, observe that restricting ON2D to Assumption A3 does not come at a large cost.

Theorem 3. *Let* ALG_1 *be an optimal offline algorithm for* PPP^2*, and let* ALG_2 *be an optimal offline algorithm for* PPP^2 *where we relax Assumption A3. Then* $\pi(\text{ALG}_2) \leq \pi(\text{ALG}_1) \leq 2\cdot\pi(\text{ALG}_2)$.

Proof: Consider any contract $C_i^{(m)}(r_i, d_i)$ bought by an optimal offline algorithm for PPP^2 without Assumption A3. When time is divided into intervals of length d_i, $C_i^{(m)}$ will overlap in time with at most two contracts $C_i^{(j)}$ and $C_i^{(l)}$ of duration d_i. Therefore, we can modify the optimal solution output by ALG_2 by purchasing those two contracts instead of $C_i^{(m)}$, eventually transforming the optimal solution output by ALG_2 into a feasible solution for PPP^2 (under Assumption A3). Hence, we can guarantee that $\pi(\text{ALG}_2) \leq \pi(\text{ALG}_1) \leq 2\cdot\pi(\text{ALG}_2)$. ■

Hence, since ON2D is k-competitive under Assumption A3 (Theorem 2), and since the optimal offline cost is

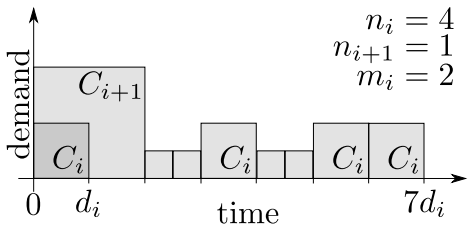

Figure 3. ON buys $n_i = 4$ contracts C_i and $n_{i+1} = 1$ contract C_{i+1} over seven intervals of length d_i. In two of these seven intervals ON buys several contracts smaller than C_i to cover the demand.

at most a factor of two lower without the interval model (Theorem V), we have:

Corollary 1. ON2D *is* $2k$*-competitive for the general* PPP^2 *problem without Assumption A3, where* k *is the number of contracts.*

VI. LOWER BOUND

Theorem 2 is essentially the best we can hope for:

Theorem 4. *No deterministic online algorithm can achieve a competitive ratio less than* $k/3$*.*

The proof is the 2-dimensional analogon of the proof in [13]. We consider a scenario where the next larger available contract doubles in cost. With k being the number of different contracts, each contract is $2k$ times longer and has $2k$ times more rate, i.e., in our plane representation contracts are squares covering an area $(2k)^{2i}$.

$$\pi(C_i) = 2^{i-1}$$
$$r_1 = 1; r_i = 2k \cdot r_{i-1} = (2k)^{i-1}$$
$$d_1 = 1; d_i = 2k \cdot d_{i-1} = (2k)^{i-1}$$

In the following, let us focus on a simple demand which only assumes rates $\sigma_t \in \{0, 1\}$ for all t. We let the adversary schedule demand only when ON has no valid contract. For each interval $(2k)^i$ where the adversary asks for a 1-demand, ON can choose between three options (see also Figure 3):

1 Eventual purchase of contract C_i. Assume that this happens n_i times.
2 Eventual purchase of larger contracts $C_j, j > i$. Assume that this happens $\sum_{j>i}^{k} n_j$ times.
3 Never purchase contract C_i or any larger contracts. Assume this happens m_i times.

Therefore the sum of all contracts bought by ON is given by $\pi(\text{ON}) = \sum_{i=1}^{k} n_i \cdot \pi(C_i)$. Given an interval of length ℓ, we estimate the cost of OFF by less than buying multiples of only one kind of contract over the full interval, i.e., ℓ/d_i contracts *for any* i: $\pi(\text{OFF}) \leq \pi(C_i)(m_i + \sum_{j\geq i}^{k} n_j)$. In order to derive the lower bound we first prove a minimum cost of any algorithm ON on intervals that start with a demand rate of 1.

Lemma 4. *Any* ON *must pay at least* $\pi(C_i)$ *on each interval of length* d_i *that starts with a demand rate of* 1.

Proof: By induction on the different intervals 2^{i-1}. For $i = 1$, each algorithm must at least buy a contract of type C_1 in order to cover that demand. Assume that for $i - 1$, it holds and now let us argue for i. If ON does not buy a contract of type C_i, we can divide the volume into $(2k)^2$ squares with side length d_{i-1} each, where $2k \cdot d_{i-1} = d_i$. We let each of these $2k$ intervals (at the bottom row) start with a demand of 1 which then cost at least $\pi(C_{i-1})$ due to the induction hypothesis. The total cost is at least $2k \cdot \pi(C_{i-1}) = k \cdot \pi(C_i)$ for every interval where ON does not buy a contract i and at least $\pi(C_i)$ otherwise. ■

Consider now an interval of length $(2k)^{i-1}$ where no contract of type i or higher was bought. We know from the induction that $\pi(\text{ON}) \geq m_i \cdot k \cdot \pi(C_i)$. We can derive the following lower bound:

$$k \cdot \pi(\text{OFF}) \leq \sum_{i=0}^{k} \left[\pi(C_i)(m_i + \sum_{j\geq i}^{k} n_j) \right] \quad (1)$$
$$\leq \sum_{i=0}^{k} \left[n_i \sum_{j=1}^{i} \pi(C_i) + m_i \cdot \pi(C_i) \right] \quad (2)$$
$$\leq \sum_{i=0}^{k} [2n_i \cdot \pi(C_i) + m_i \cdot \pi(C_i)] \quad (3)$$
$$\leq 3 \cdot \pi(\text{ON}) \quad (4)$$

Inequality (1) is given by the cost estimation of OFF against any ON buying only one kind of contracts. Inequality (2) is a reorganization of the sum since $\pi(C_i)$ is multiplied by every $n_j, j \geq i$ which is also given after the reordering. Afterwards, we use the geometric sum on the cost of the contracts to derive Inequality (3). This leads to a lower bound of $k/3$ since $\pi(\text{ON}) = \sum_{i=1}^{k} n_i \cdot \pi(C_i)$ and $\pi(\text{ON}) \geq m_i \cdot k \cdot \pi(C_i)$.

VII. OPTIMAL OFFLINE ALGORITHM

So far, we have treated the optimal offline algorithm on which ON2D relies as a black box. In the following, we show that offline solutions can indeed be computed in polynomial time, and present a corresponding dynamic programming algorithm OFF2D.

The basic idea behind the offline algorithm OFF2D is that the optimal cost for any contract over a certain

Algorithm 2 Pre-computation of matrix M for d_k-length

Input: Demand sequence $\sigma_t, \ldots, \sigma_{t+d_k}$ (over interval $[t, t+d_k]$).
Output: Matrix M.

1: **for** $i = 1$ to d_k **do**
2: $\quad M[i,i] \leftarrow \sigma_{t+i}$
3: **for** $i = 1$ to $d_k - 1$ **do**
4: $\quad$ **for** $j = i+1$ to d_k **do**
5: $\quad\quad M[i,j] \leftarrow \max\{M[i,j-1], \sigma_{t+j}\}$
6: **return** M

interval is obtained either by splitting the cost at some time, or by buying a long contract with a certain rate r. In the following, recall that d_k is the duration of the largest contract C_k.

OFF2D proceeds as follows: It splits time into intervals of length d_k and solves each of these interval separately using Algorithm 3. OFF2D relies on the following data structures: For each d_k-length time interval I, we precompute the maximum demand within any subinterval $[i,j]$ of I, and store this information in position $M[i,j]$ of a $d_k \times d_k$ matrix M (Algorithm 2). In particular the maximum requested demand $\hat{\sigma}$ in interval I is stored in $M[1,d_k]$. A $d_k \times d_k \times \hat{\sigma}$ matrix OPT is used to compute the optimal cost. The entry $\texttt{OPT}[i,j,\lambda]$ indicates the optimal cost of covering a demand rate of $M[i,j]-\lambda$ over the interval $[i,j]$ — i.e. λ indicates the amount of *covered demand* for $[i,j]$. Initially, all entries are set to 0.

Algorithm 2 pre-computes the matrix M over the d_k-length interval $[t, t+d_k]$, where $t = b \cdot d_k$, for integer $b \geq 0$. Lines 1-2 initialize the matrix and store the demand σ_{t+i} in entry $M[i,i]$. Lines 3-5 compute the maximum demand within any time interval $[t+i, t+j]$, $0 \leq i \leq j \leq d_k$. The demand can be obtained by comparing the demand at time $t+j$ (i.e., σ_{t+j}) with the maximum demand between time $t+i$ and $t+j-1$, which has already been computed by our algorithm.

After obtaining the matrix M over interval $[t, t+d_k]$, we can compute the optimal solution for the PPP^2 problem over the same interval using Algorithm 3, as we show in Theorem 5:

Theorem 5. *Algorithm 3 computes an optimal offline solution for any given interval of length d_k in time $O(d_k^3 \cdot \hat{\sigma})$, where $\hat{\sigma}$ is the maximum demand over the interval.*

Proof: We assume, for the sake of simplicity and without loss of generality, that $t = 0$ and the d_k-length interval we consider is $[0, d_k]$.

Correctness: By induction over the length of the subintervals $\ell = j - i + 1$ and the respective uncovered

Algorithm 3 Offline Algorithm for d_k-length interval

Input: Precomputed matrix M over interval $[t, t+d_k]$.
Output: Optimal total costs $\texttt{OPT}[i,j,\cdot]$ for all intervals within $[t, t+d_k]$.

1: Initialize all entries in OPT to be 0.
2: Let $\hat{\sigma} = M[i,j]$.
3: **for** $i = 1$ to d_k **do**
4: $\quad$ **for** $\lambda = M[i,i] - 1$ to 0 **do**
5: $\quad\quad \texttt{OPT}[i,i,\lambda] \leftarrow \min_{C(r,d)\in\mathcal{C}}\{\texttt{OPT}[i,i,\min\{\hat{\sigma}, \lambda+r\}] + \pi(r,d)\}$
6: **for** $\ell = 2$ to d_k **do**
7: $\quad$ **for** $i = 1$ to $d_k - \ell + 1$ **do**
8: $\quad\quad j = i + \ell - 1$
9: $\quad\quad$ **for** $\lambda = M[i,j] - 1$ to 0 **do**
10: $\quad\quad\quad \texttt{OPT}[i,j,\lambda] \leftarrow \min_{i \leq z < j}\{\texttt{OPT}[i,z,\min\{M[i,z],\lambda\}] + \texttt{OPT}[z+1,j,\min\{M[z+1,j],\lambda\}]\}$
11: $\quad\quad\quad \mathcal{C}' \leftarrow \{C^{(x)}(t^{(x)}, r, d) \in \mathcal{C} : t^{(x)} = b \cdot d$ for some positive integer b and $t^{(x)} \leq i < j \leq t^{(x)} + d\}$
12: $\quad\quad\quad$ **if** $\mathcal{C}'$ is not empty **then**
13: $\quad\quad\quad\quad \texttt{OPT}[i,j,\lambda] \leftarrow \min\{\texttt{OPT}[i,j,\lambda]; \min_{C(r,d)\in\mathcal{C}'} \texttt{OPT}[i,j,\min\{\hat{\sigma}, \lambda + r\}] + \pi(r,d)\}$
14: **return** $\texttt{OPT}[1, d_k, 0]$

demand λ. Clearly, the claim is true for intervals $[i,i]$ ($\ell = 1$) (Lines 3-5): If $\lambda > 0$ we need at least one contract $C(r,d)$ to finish covering the demand at time i; the remaining demand at time i not covered by C must be covered optimally by other contracts, as previously computed in $\texttt{OPT}[i,i,\lambda+r]$. Now consider a subinterval $[i,j]$ of length $\ell = j - i + 1 \geq 2$, where $1 \leq i \leq j \leq d_k$. This interval is either split into two non-overlapping subintervals of smaller length (Case I), or a long contract of length equal to or greater than ℓ that completely covers $[i,j]$ is bought, at a certain demand rate r, where $0 \leq r \leq M[i,j]$ (Case II). Given Assumption A2 and A3, for any instances of contracts $C_x^{(y)}$ and $C_p^{(q)}$, *either the duration of one contract is fully contained in the other*, or the *two contracts never overlap in time*: Hence, given that we consider all intervals $[i,j]$, including the ones that may correspond to actual instances of contracts, it is enough to consider only these two cases.

In Case I, we split the interval at time z such that the solution $\texttt{OPT}[i,z,\min\{M[i,z],\lambda\}] + \texttt{OPT}[z+1, j, \min\{M[z+1,j],\lambda\}]$ is minimized over all z between i and j (Line 10). Since the lengths of the two subintervals $z - i + 1$ and $j - z$ are both smaller than ℓ,

$\texttt{OPT}[i, z, \lambda]$ and $\texttt{OPT}[z+1, j, \lambda]$ already store the cost of optimal solutions for these subproblems, respectively, by the induction hypothesis. Hence $\texttt{OPT}[i, z, \lambda] + \texttt{OPT}[z + 1, j, \lambda]$ will yield the optimal solution for $\texttt{OPT}[i, j, \lambda]$ if Case I applies.

In Case II, we buy a long contract with rate r. First, we need to check which contracts with longer durations can cover $[i, j]$ fully, and store the candidate contracts in $\mathcal{C}'$. A candidate contract $C^{(x)}(t^{(x)}, r, d)$, where $t^{(x)} = b \cdot d$ according to Assumption A3, satisfies $t^{(x)} \leq i < j < t^{(x)} + d$. The algorithm picks the valid candidate contract that minimizes $\pi(r, d)$ plus the optimal cost of covering the largest remaining demand $M[i, j] - (\lambda + r)$ over $[i, j]$, which has been previously computed and stored in $\texttt{OPT}[i, j, \lambda + r]$ (Line 11).

By choosing the smaller value of Cases I and II, we obtain the optimal cost for subproblem $[i, j, \lambda]$ (Line 13).

Time Complexity: The total time complexity of OFF2D for the pre-computation part in Algorithm 2 is $O(d_k^2)$. The first part of Algorithm 3 in (Lines 3-5) takes $O(d_k \cdot k \cdot \widehat{\sigma})$ time, where $\widehat{\sigma}$ is the maximum demand for the whole time interval. The first two loops of the second part (Lines 6-7) take $O(d_k^2)$ time and the for-loop in Line 9 takes $O(\widehat{\sigma})$ time. The statement in Line 10 requires $O(d_k)$ time and Lines 11 and 13 take time $O(k)$ each. Therefore, the total time complexity is $O(d_k^3 \cdot \widehat{\sigma})$ for a subinterval with length d_k. ∎

Taking Theorem 5 into account for all intervals of length d_k in σ, and for a long enough demand sequence σ (i.e., such that $|\sigma| = \Omega(\widehat{\sigma})$, where $\widehat{\sigma}$ is the maximum demand over σ), we get the following corollary, which expresses the total running time of the offline algorithm:

Corollary 2. *Algorithm* OFF2D *runs in time* $O(|\sigma|^2 d_k^2)$.

Proof: By summing up the computation time of $\lceil |\sigma|/d_k \rceil$ subintervals of length d_k, we have an overall complexity of $O(|\sigma| \cdot d_k^2 \cdot \widehat{\sigma}) = O(|\sigma|^2 d_k^2)$, since $|\sigma| = \Omega(\widehat{\sigma})$. ∎

VIII. Higher Dimensions

OFF2D and ON2D are designed for the two-dimensional version of the PPP problem but they can also be extended towards a D-dimensional version of the problem, where each additional dimension (other than the time duration dimension) would indicate the rate at which you would buy a certain resource. Regarding OFF2D we need to do the following changes: For each additional dimension we need to extend the dimension of the optimal cost matrix $\texttt{OPT}$ by one and add two additional loops in OFF2D's Algorithm 3. Furthermore we only need to add one additional dimension for the M matrix in Algorithm 2 which indicates the current demand dimension β, $M[i, j, \beta]$ and run the algorithm β times for the pre-computation. We illustrate those changes below in $3D$.

Assume a scenario where a third dimension is added, e.g. computational and network resources over time. The contracts $C(r, r', d)$ then cover $r \times r' \times d$ cuboids. In order to adjust Algorithm 3, we add another loop after Line 4 which goes through the maximum values λ' of the additional demand (**for** $\lambda' = M[i, i, 2] - 1$ to 0 **do**) and change the statement in Line 5 to: $\texttt{OPT}[i, i, \lambda, \lambda'] \leftarrow \min_{C(r,r',d) \in \mathcal{C}'} \texttt{OPT}[i, i, \lambda + r, \lambda' + r'] + \pi(r, r', d)$. The same loop must also be added after Line 9 and the updates of the $\texttt{OPT}$ matrix must be changed accordingly in Lines 10 and 13.

The runtime of the pre-computation in Algorithm 2 would be increased by a factor of D (i.e., by the dimension of the problem) and still be negligible regarding the overall runtime (assuming D is a constant). For Algorithm 3 the runtime would increase by a factor of $\Pi_{i \geq 2} \widehat{\sigma}^i$, where σ^i is the maximum demand for resource i, for $i \geq 1$, leading to an overall runtime of $O(d_k^3 \cdot \Pi_{i \geq 1} \widehat{\sigma}^i)$ for each interval d_k.

No changes are needed regarding ON2D. It still mimics OFF2D's behavior and given the Assumptions A2 and A3, the contract independence still holds for higher dimensions. Hence, the proof for the competitive ratio of k still applies.

IX. Simulations

We have conducted a small simulation study to complement our formal analysis. In this simulation, we consider k square contracts where $C_i(r_i, d_i)$ has rate and duration $r_i = d_i = 2^{i-1}$, for $1 \leq i \leq k$. The price π of a contract is a function of the rate-duration product $r_i \cdot d_i$, and we study a parameter x to vary the discount. Concretely, we consider a scenario where a twice as large time-rate product is by factor $(1 + x)$ more expensive, i.e., $\pi(2 \cdot d \cdot r) = (1 + x) \cdot \pi(d \cdot r)$; we set $\pi(1) = 1$.

To generate the demand σ, we use a randomized scheme: non-zero demand requests arrive according to a Poisson distribution with parameter λ, i.e., the time between non-zero σ_t is exponentially distributed. For each non-zero request, we sample a demand value uniformly at random from the interval $[1, y]$.

Each simulation run represents 1000 time steps, and is repeated 10 times.

Impact of the request model. We first study how the competitive ratio depends on the demand arrival pattern. Figure 4 plots the competitive ratio ρ as a function of the Poisson distribution parameter λ. The price model with $x = 0.5$ is used, and there are $k = 8$ contract types. First,

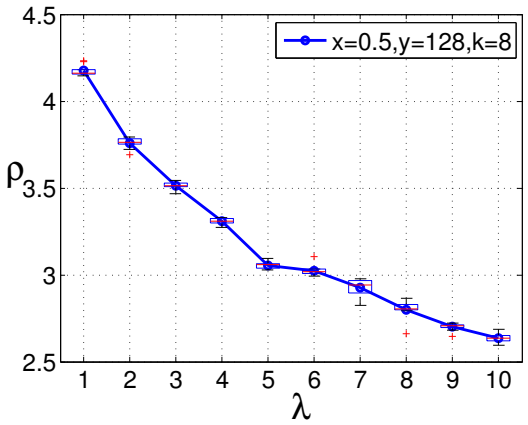

Figure 4. Effect of request distribution (Poisson λ).

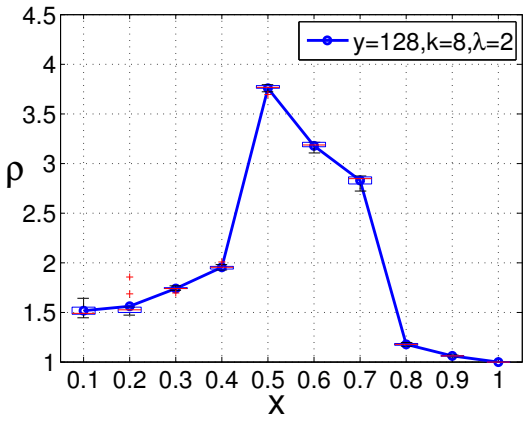

Figure 5. Effect of discount x.

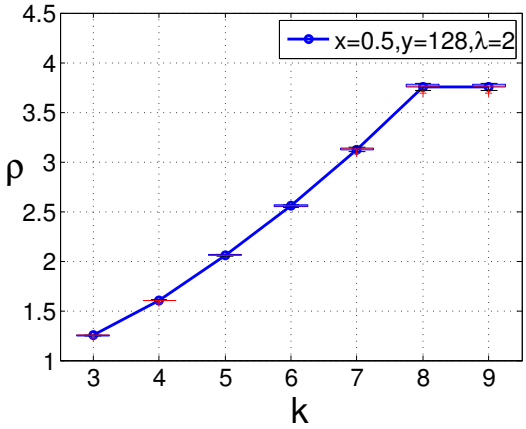

Figure 6. Effect of the number of contracts k

we observe that the competitive ratio ρ is bounded by approx. 5, which is slightly lower than what we expect in the worst-case (cf Theorem 2). Another observation is that the competitive ratio decreases as λ increases. This can be explained by the fact that demand rates become sparser for increasing λ, and hence less contracts will be bought. Meanwhile, when the demand rates are sparse, the offline algorithm will have less chance to buy a larger contract. Put differently, the online algorithm will pay relatively more compared to the offline algorithm for small λ, as it purchases more small contracts.

Impact of the price model. Different price models also affect the purchasing behavior of our online algorithm. Figure 5 shows the competitive ratio ρ for different values x. (For this scenario, we set $y = 128$, $k = 8$ and $\lambda = 2$.) We see a tradeoff: for small x, until $x = 0.5$, the competitive ratio increases and then begins to decrease again. The general trend can be explained by the fact that for small x, it is worthwhile to buy larger contracts earlier, and it is hence impossible to charge ON2D much; for larger x, also an offline solution cannot profit from buying a large contract.

Impact of the number of contracts. Finally, Figure 6 shows the competitive ratio as a function of the number of contracts k. (We fix $x = 0.5$, $y = 128$ and $\lambda = 2$ in this simulation.) The competitive ratio first increases as k increases but then stabilizes. This stabilization is due to the fact that when we have eight or more contracts ($k \geq 8$), the largest contract can cover the maximum rate. In the beginning, the ratio increases since the offline algorithm buys larger and larger contracts, and the online algorithm pays for many small contracts along the way.

X. Related Work

Cost reductions (due to economy-of-scale effects) are one of the main motivations behind today's trend to outsource infrastructure and software to the cloud. A large body of literature in the field focuses on resource allocation and scheduling problems. For a good overview, we refer the reader to the surveys [2], [5].

Compared to the algorithmic problems of the resource allocation and scheduling, the economical aspects are less well-understood. Different economical cloud models have been proposed and compared by various authors, e.g., by Armbrust et al. [12], Pal et al. [14], or Dash et al. [6]. Some of the studied pricing models have their origins in the context of ISP-customer relationships [17] and are also related to classic economic problems [8]. An interesting tradeoff between time and price has been studied in [10], from a scheduling complexity perspective. More generally, there are several interesting proposals for novel adaptive resource and spot market

pricing schemes, e.g., [1]. Our model is motivated by architectures such as [4], [16] which allow for subrenting and recursion.

This paper assumed an online algorithm and competitive analysis perspective. Many online models, such as ski-rental problems [3], facility location problems [9], or buy-at-bulk [15] and rent-or-buy [11] problems, assume that once an item has been purchased, it remains indefinitely for no extra charge. The Parking Permit Problem [13] and the Bahncard Problem [7] are the archetype for online problems where purchases have time durations which expire regardless of whether the purchase is used or not.

The paper closest to ours is the Parking Permit Problem (PPP) paper by Meyerson [13]. Formally, PPP specifies a set of k different types of parking permits of a certain price π_i and duration d_i. In [13], Meyerson presents an asymptotically optimal deterministic online algorithm with competitive ratio $O(k)$ (together with a lower bound of $\Omega(k)$). The paper also discusses randomized algorithms and an application to Steiner forests. While we can build upon some of the techniques in [13], the rate dimension renders the problem different in nature, both from an online and an offline algorithm perspective.

XI. Summary and Conclusion

This paper has shown that at the heart of efficient cloud resource allocation lies a fundamental algorithmic problem, and we introduced the PPP^2 problem, a *2-dimensional* variant of the *online Parking Permit Problem* PPP. We presented a deterministic online algorithm On2D that provably achieves a competitive ratio of k, where k is the total number of available contracts; if we relax Assumption A3, the competitive ratio of our algorithm is $2k$. We also showed that On2D is almost optimal in the sense that no deterministic online algorithm for PPP^2 can achieve a competitive ratio lower than $k/3$. Finally, we proved that the offline version of PPP^2 can be solved in polynomial time.

We believe that our work opens interesting directions for future research.

1) *Optimality:* The obvious open question regards the gap between the upper bound k and the lower bound $k/3$ derived in this paper.
2) *Relaxing the assumptions:* While the interval model only comes at the cost of a small additional approximation factor (a constant), it seems hard to remove the assumption entirely but still being able to compute optimal solutions in polynomial time: we conjecture that the problem is NP-hard. We believe that relaxing the multiplicity assumption (which is needed for the concept of contract independence in the upper bound proof of Section V) is a more promising direction for future research.
3) *Randomized algorithms:* It will be interesting to study whether the randomized algorithms known from the classic parking permit problem can also be generalized to multiple dimensions.

Acknowledgments. This research was supported by the BMBF (01IS12056).

References

[1] V. Abhishek, I. A. Kash, and P. Key. Fixed and market pricing for cloud services. In *Proc. NetEcon Workshop*, 2012.
[2] M. Armbrust, A. Fox, R. Grifith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. In *UC Berkeley Technical Report EECS-2009-28*, 2009.
[3] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Elsevier FGCS*, 25(6), 2009.
[5] N. M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, 54:862–876, 2010.
[6] D. Dash, V. Kantere, and A. Ailamaki. An economic model for self-tuned cloud caching. In *Proc. IEEE International Conference on Data Engineering*, pages 1687–1693, 2009.
[7] R. Fleischer. On the bahncard problem. *Theor. Comput. Sci.*, 268(1):161–174, 2001.
[8] S. Goyal and B. Giri. Recent trends in modeling of deteriorating inventory. *Elsevier EJOR*, 134(1), 2001.
[9] S. Guha and K. Munagala. Improved algorithms for the data placement problem. In *Proc.13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 106–107, 2002.
[10] T. A. Henzinger, A. V. Singh, V. Singh, T. Wies, and D. Zufferey. A marketplace for cloud resources. In *Proc. 10th ACM EMSOFT*, 2010.
[11] A. Kumar, A. Gupta, and T. Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In *Proc. 43rd Symposium on Foundations of Computer Science (FOCS)*, 2002.
[12] M. Armbrust et al. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.
[13] A. M. Meyerson. The parking permit problem. In *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 274–284, 2005.
[14] R. Pal and P. Hui. Economic models for cloud service markets. In *Proc. ICDCN*, 2012.
[15] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian. Buy-at-bulk network design: Approximating the single-sink edge installation problem. In *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 619–628, 1997.
[16] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy. Network virtualization architecture: Proposal and initial prototype. In *Proc. ACM SIGCOMM VISA*, 2009.
[17] S. Shakkottai and R. Srikant. Economics of network pricing with multiple isps. *IEEE/ACM TON*, 14(6), 2006.