Integrated revenue management approaches for capacity control with planned upgrades

May 25, 2012

Claudius Steinhardt, Jochen Gönsch

Claudius Steinhardt (Corresponding Author)

Department of Analytics & Optimization,
University of Augsburg, Universitätsstraße 16, 86159 Augsburg, Germany

Phone: +49 821 598 4159

Fax: +49 821 598 4226

e-mail: claudius.steinhardt@wiwi.uni-augsburg.de

Jochen Gönsch

Department of Analytics & Optimization,
University of Augsburg, Universitätsstraße 16, 86159 Augsburg, Germany

e-mail: jochen.goensch@wiwi.uni-augsburg.de

Abstract

In many service industries, firms offer a portfolio of similar products based on different types of resources. Mismatches between demand and capacity can therefore often be managed by using product upgrades. Clearly, it is desirable to consider this possibility in the revenue management systems that are used to decide on the acceptance of requests. To incorporate upgrades, we build upon different dynamic programming formulations from the literature and gain several new structural insights that facilitate the control process under certain conditions. We then propose two dynamic programming decomposition approaches that extend the traditional decomposition for capacity control by simultaneously considering upgrades as well as capacity control decisions. While the first approach is specifically suited for the multi-day capacity control problem faced, for example, by hotels and car rental companies, the second one is more general and can be applied in arbitrary network revenue management settings that allow upgrading. Both approaches are formally derived and analytically related to each other. It is shown that they give tighter upper bounds on the optimal solution of the original dynamic program than the well-known deterministic linear program. Using data from a major car rental company, we perform computational experiments that show that the proposed approaches are tractable for real-world problem sizes and outperform those disaggregated, successive planning approaches that are used in revenue management practice today.

Keywords: Revenue Management, Upgrades, Capacity Control, Dynamic Programming, Decomposition, Car Rental

appears in: European Journal of Operational Research 223 (2012), 380–391

Integrated revenue management approaches for capacity control with planned upgrades

Claudius Steinhardt, Jochen Gönsch

1. Introduction

Capacity control, which is considered to be the key component of modern revenue management, is concerned with the task of optimally selling a fixed perishable capacity within a given time horizon by controlling the availability of the products that make use of this capacity (see, e.g., Talluri & van Ryzin, 2004). However, traditional capacity control often ignores the fact that many firms in the service industry offer several products that are substitutable in the sense that the seller could fulfill a certain product request with a more desirable substitute from a prespecified set of alternative products. Examples include airlines selling economy, business, and first class seats and car rental companies offering many types of cars differing in size and features.

In this paper, we consider the situation when this substitution is provided at the original product's price, which is called an *upgrade*. It is assumed that customers always accept upgrades to superior products if they are offered these at no extra cost. Conversely, if customers are urged to voluntarily buy a higher value product – in practice, this usually occurs at the time of fulfillment – this is termed an *upsell*. Upgrading and upselling can be beneficial if the selling firm faces a mismatch between supply and demand. This mismatch often occurs when capacity decisions have been determined for the long term, which is usual in traditional revenue management scenarios, while firms experience stochastic and seasonal demand. We assume that the capacity mismatch is relatively transient or not very pronounced; customers will therefore not strategically adapt to upgrades.

Two important aspects of upgrading are *fairness* and *scope* (see, e.g., Gallego & Stefanescu, 2009). Informally, upgrades are defined as fair if upgrade priority is given to customers who purchase higher quality products. Specifically, no customer should receive a higher quality upgrade than another who had originally bought a higher quality product. Regarding the scope of upgrades, two basic models can be distinguished: *Full cascading*, which allows upgrades to any higher quality product, and *limited cascading*, which is less flexible as upgrades are allowed only to the next higher quality product.

The research presented in this paper is motivated by an industry project conducted with a major car rental company. In fact, the car rental industry is one of the most important users of upgrades. As Geraghty and Johnson (1997) note, it is essential for car rental companies to undertake market segmentation by not only offering different products using the same resources but also by offering many different car types. This is especially important as some typical fencing conditions known from the airline industry, for example, advanced purchase restrictions, do not seem to work well in the car rental business (see, e.g., Carroll & Grimes, 1995). However, by offering many different car types, car rental companies cannot directly substitute one type for another but must consider a certain predefined upgrade hierarchy. The resulting supply-side substitution opportunities are inevitable as, although desirable, there still is no tight integration of fleet management and revenue management decisions in the car rental industry (see, e.g., Lieberman, 2007). Furthermore, as the differences in the cost are usually quite small, car rental firms have a common policy to acquire fewer economy cars than required and more compact or full-size cars. Through upgrading, the latter cars offer more flexibility if the demand situation changes.

Although the above examples illustrate an obvious need to explicitly incorporate upgrading opportunities in revenue management's capacity control process, there is not much theoretical work on developing appropriate approaches. Practical implementations usually resort to rather simple heuristics, such as successive planning, which means that upgrade contingents on higher-valued resources are determined first and that, subsequently, the new virtual capacity is considered fixed for standard capacity control. Therefore, one of the contributions of this paper is the proposition of integrated dynamic programming decomposition approaches for capacity control with planned upgrades. Our approaches extend the well-known dynamic programming decomposition from traditional capacity control to additionally exploit the opportunities of planned upgrades. Furthermore, we show that the approaches are applicable to real-world scenarios in car rental revenue management and that they outperform common practice procedures, such as the successive planning of upgrade contingents and capacity control decisions.

The paper is structured as follows: Section 2 provides a brief review of related literature. In Section 3, we present two dynamic programming formulations for capacity control with planned upgrades that are adopted from the literature. Based on these formulations, as our first contribution, we theoretically derive new structural insights that facilitate the control process under certain conditions that often occur in practice, such as single-leg airline capacity control and daily car rental capacity

control. Based upon these models and insights, we then propose two different dynamic programming decomposition approaches for capacity control with planned upgrades, which is the second contribution of the paper (Section 4). While the first approach is specifically suited for multi-day revenue management problems that occur, for example, in the hotel and car rental businesses, the second approach is more general and can be applied in arbitrary network revenue management settings. In Section 5, we discuss the computational results from a simulation study based on real-world data obtained from our car rental industry partner. These results demonstrate the decomposition approaches' practical applicability and their relative performance compared to other common control procedures, such as successive planning. We conclude with a summary of the key results (Section 6). The proofs of all propositions as well as complementary tables with additional information on the simulation study are provided in the Online Appendix.

2. Literature Review

There is an extensive literature on revenue management in general. For surveys, see, for example, Belobaba (1987), Weatherford and Bodily (1992), McGill and van Ryzin (1999), and Chiang, Chen and Xu (2007), as well as the textbooks by Talluri and van Ryzin (2004) and Phillips (2005).

While most revenue management publications consider only one type of resource and thus do not mention any supply-side substitution, a few authors have proposed approaches that integrate planned upgrades with capacity control. Alstrup, Boas, Madsen and Vidal (1986) present a dynamic programming formulation for an overbooking problem, with two types of resources incorporating upgrades as well as downgrades. Karaesmen and van Ryzin (2004) also incorporate overbooking but propose a two-stage model: In the first stage, optimal booking limits are determined and then used for capacity control. In the second stage, just before the service provision, the accepted customers are assigned to inventory classes to maximize the net benefit. The latter step is performed by a transportation problem's solution that allows for upgrades as well as downgrades.

In a more recent publication, Shumsky and Zhang (2009) present a dynamic programming approach for capacity control with planned upgrades as well. They consider a non-network setting with a number of resources in a hierarchy with a one-to-one relationship between resources and products. Furthermore, they assume that in each time period, multiple requests arrive. The total demand of the period is observed and then the number of accepted as well as upgraded requests can be determined ex post. Their analysis is restricted to limited-cascading planned upgrades: In addition to revenues,

they define different types of costs from which it follows that only one step-upgrading is profitable at all. Based on these assumptions, Shumsky and Zhang (2009) are able to prove the optimality of an upgrade policy that relies on protection levels. In particular, in their setting, at the end of each period, it is optimal to first accept as much of each product's demand as can be served with the corresponding resource and then to accept existing additional demand by assigning it to the next-higher resource, up to a calculated protection limit. The authors also derive upper and lower bounds on the optimal protection limits by restricting the capacity of an arbitrary higher value resource to infinity and zero, respectively. In contrast to the approach of Shumsky and Zhang (2009), we consider a more traditional revenue management setting that is also considered, e.g., by Gallego and Stefanescu (2009). In particular, we are faced with a successive arrival process in which an acceptance decision must be made immediately for every single incoming request. Furthermore, we generally consider a multi-day setting, that is, a network structure. In addition, as we do not impose a specific cost structure, cascading upgrades are potentially beneficial so that we do not restrict ourselves to limited-cascading planned upgrades.

The paper by Gallego and Stefanescu (2009) is most closely related to our work and therefore necessities a detailed discussion. The authors introduce two dynamic programming formulations for capacity control with planned upgrades in an independent demand setting. The formulations differ in terms of the point in time when the final upgrade decision must be made. In what follows, the authors concentrate exclusively on a static deterministic linear programming approximation for both programs. Basically, this is an extension to the well-known DLP-approximation for standard capacity control in revenue management. Gallego and Stefanescu show their approximation to give upper bounds on the original dynamic programs' values. For a specific, non-network case of this model, the authors show that a fair upgrading solution always exists. Furthermore, they derive conditions for the existence of a revenue-optimal non-cascading solution. Our work directly builds upon the two dynamic programming formulations presented early in the paper by Gallego and Stefanescu (2009). However, differing from their paper, we do not concentrate on the DLP-approximation of these formulations but instead work directly with the dynamic programming formulations. In particular, we directly derive new structural results for the dynamic programming formulations, additionally assuming a common non-network structure similar to the structure that Gallego and Stefanescu assume when analyzing fairness and limited cascading issues in their DLP. Furthermore,

we develop new decomposition approaches directly for a dynamic program that includes upgrades and is adapted to the multi-day setting in terms of notation.

Additionally, the literature on revenue management with flexible products is somewhat related to our work, because flexible products can, in fact, be considered to be a generalization of upgradeable products. A flexible product consists of several alternatives offered by a supplier, who reserves the right to assign one of the alternatives to customers at a predefined point in time after purchase. In the scientific literature, flexible products were introduced and first investigated by Gallego and Phillips (2004) and Gallego, Iyengar, Phillips and Dubey (2004). Petrick, Steinhardt, Gönsch and Klein (2012) undertake comprehensive computational studies that reveal the revenue potential when flexible products are used in settings with uncertain demand forecasts (see also Petrick, Gönsch, Steinhardt & Klein, 2010).

It should be noted that a vast body of literature investigates supply-side substitution from a cost perspective in the context of production and inventory management. See, for example, Lang (2009) for an overview.

In the literature on capacity control in the traditional network revenue management setting, a number of approaches that are used to approximate the problem's original dynamic programming formulation, which itself is generally computationally intractable, exist. In this context, dynamic programming decomposition techniques are specifically related to our research. The standard decomposition approach, which is used to decompose the traditional network capacity control problem to obtain a number of single-leg problems, is well-established in standard software implementations and described, for example, in detail by Talluri and van Ryzin (2004). Liu and van Ryzin (2008) are the first to adapt the dynamic programming decomposition technique to the choice-based network revenue management setting. Their approach has been adopted and discussed in a number of follow-up papers (see, e.g., Miranda Bront, Méndez-Díaz & Vulcano, 2009; Kunnumkal & Topaloglu, 2010; Meissner & Strauss, 2012). Zhang and Adelman (2009) derive an upper bound that relates the single-leg problem's value to the network value in the choice-based setting. Decomposition heuristics have also been applied in settings that incorporate overbooking (see, e.g., Erdelyi & Topaloglu, 2009; Erdelyi & Topaloglu, 2010). Further decomposition techniques such as prorating (see, e.g., Kemmer, Strauss & Winter, 2011), displacement adjusted virtual nesting (see, e.g., Belobaba, 1987; Smith & Penn, 1988), and iterative nesting (see, e.g., Talluri & van Ryzin, 2004, chapter 3.4.5) have been described in the literature. Another group of approximations are not based on decomposition

but on easier-to-solve network models, such as the DLP model (see, e.g., Glover, 1982), the PNLP model (see, e.g., Wollmer, 1986), and the RLP model (see, e.g., Smith & Penn, 1988). Furthermore, there are approaches that are based on reinforcement learning and that usually result in simulation-based heuristics for solving single-leg problems and thus could be used within a decomposition scheme (see, e.g., Gosavi, Bandla & Das, 2002; Gosavi, 2004). In addition, there are a number of model-based or model-free simulation-based optimization methods that use stochastic gradient methods (see, e.g., Bertsimas & de Boer, 2005; van Ryzin & Vulcano, 2008) or meta-heuristics such as simulated annealing (see Gosavi, Ozkaya & Kahraman, 2007) and scatter search (see Klein, 2007).

Finally, there are also a few articles that specifically discuss the application of revenue management approaches in the car rental industry (see, e.g., Edelstein & Melnyk, 1977; Carroll & Grimes, 1995; Geraghty & Johnson, 1997; Lieberman, 2007; Haensel, Mederer & Schmidt, 2012). However, most of these publications focus on successful implementations of car rental revenue management systems and the arising challenges, without considering the details of the applied mathematical models and techniques. For example, Geraghty and Johnson (1997) underline the relevance of planned upgrades in the car rental business but only mention that they use a modification of the well-known EMSR heuristic without providing any further insight. In a very recent publication, Haensel, Mederer and Schmidt (2012) present a stochastic programming approach suited for revenue management in the car rental industry but without considering multiple car types.

3. Model Formulations and Structural Properties

3.1. Model Formulations

In this subsection, we restate two general dynamic programming formulations of the revenue management problem with planned upgrades that have been proposed by Gallego and Stefanescu (2009). We consider a firm offering n products. For each product $k \in \{1,\ldots,n\}$, p_k denotes its predefined price that is valid throughout the sales horizon. The sales horizon can be sufficiently discretized into T time periods so that no more than one buying request arrives in each period $t \in \{1,\ldots,T\}$. Each request asks for one single unit of one of the products, and the common ID assumption holds regarding demand. Note that, in addition to many other fields of relevance, this assumption is particularly valid for the car rental revenue management setting with corporate customers, which has motivated this research. The probability of a request for product $k \in \{1,\ldots,n\}$ is giv-

en by λ_k, and consequently, with probability $1-\sum_{k=1}^{n}\lambda_k$, there is no request in a time period. To provide the products, there exist m resources. The total capacity of each resource $q \in \{1,\ldots,m\}$ is given by c_q, which is a non-negative integer value. The resources' capacity is grouped in the $m\times 1$ vector $\mathbf{c}=(c_1,\ldots,c_m)^T$. The products' capacity consumption is defined by the $m\times n$ incidence matrix $\mathbf{A}$. Let a_{qk} refer to the element at the *q*-th row and the *k*-th column of $\mathbf{A}$. Then, $a_{qk}=1$ if resource q is used by product k, and $a_{qk}=0$ otherwise. Furthermore, $\mathbf{a}_k$ denotes the *k*-th column of $\mathbf{A}$, that is, the vector of resources used by one unit of product k. As is common in revenue management, capacity is assumed to be fixed, and capacity that remains unused after the sales horizon is worthless. We assume that the products can be ordered with respect to a given upgrade hierarchy. The firm guarantees that a customer purchasing an upgradeable product $k \in \{1,...,n\}$ at price p_k will obtain an alternative belonging to a set $\mathcal{V}_k=\{k\}\cup\mathcal{W}_k$, which consists of the product itself and the upgrade possibilities $\mathcal{W}_k \subseteq \{k+1,...,n\}$ that can arbitrarily be defined. The buyer of a non-upgradeable product obtains exactly the product he purchased, that is $\mathcal{V}_k=\{k\}$. To ease notation, we omit the index sets for the symbols k and j referring to products, q referring to resources, and t referring to time periods. For example, the notation $\forall k$ means $k \in \{1,\ldots,n\}$, and $\sum_k$ means $\sum_{k\in\{1,\ldots,n\}}$. Moreover, throughout the paper, we use $\mathbf{e}_h$ to denote the h-th standard basis vector of the H-dimensional vector-space of integer values, without stating the dimension H wherever it is obvious from the context. The symbol **0** refers to the zero vector.

In this setting, two different upgrade mechanisms can be considered. Making use of the ad-hoc mechanism, the firm must immediately decide to upgrade or not when an upgradeable product is sold, whereas the second mechanism postpones this decision until the end of the sales horizon.

We first consider the case that upgrades are decided ad-hoc at the time of sale. Each possible state within the capacity control process can be described by $(\mathbf{x},t)$, where *t* denotes the considered time period, and the $m\times 1$ vector $\mathbf{x}=(x_1,...,x_m)^T$ denotes the corresponding free resources' capacity with $\mathbf{x}\le\mathbf{c}$. Let $V(\mathbf{x},t)$ denote the expected revenue-to-go when the current state is $(\mathbf{x},t)$. Defining the opportunity cost $\Delta_j V(\mathbf{x},t)$ of assigning one unit of product $j \in \mathcal{V}_k$ to a customer requesting k as $\Delta_j V(\mathbf{x},t)=V(\mathbf{x},t)-V(\mathbf{x}-\mathbf{a}_j,t)$, the expected revenue-to-go can be computed recursively via the Bellman equation

$$V(\mathbf{x},t)=\sum_k \lambda_k \left(p_k-\min_{j\in\mathcal{V}_k}\Delta_j V(\mathbf{x},t-1)\right)^+ + V(\mathbf{x},t-1) \qquad (1)$$

with the boundary conditions $V(\mathbf{x},0)=0$ for $\mathbf{x}\geq\mathbf{0}$, and $V(\mathbf{x},0)=-\infty$ otherwise. Note that the operator $(\cdot)^+$ is the maximum of zero and the value in brackets.

Postponing the upgrade decision to the end of the sales horizon necessitates a slightly more complex formulation. The state space must be expanded by a vector of commitments to track sales, as it is no longer possible to immediately reduce capacity after the sale of an upgradeable product. Let y_k be the non-negative number of requests that have been accepted for product *k* until the current state within the capacity control process, and let $\mathbf{y}=(y_1,\ldots,y_n)^T$ be the corresponding vector including the current commitments for all products. Then, each possible state within the capacity control process can be described by $(\mathbf{x},\mathbf{y},t)$, and the expected revenue-to-go is denoted by $V(\mathbf{x},\mathbf{y},t)$. The opportunity cost of selling one unit of product k is now defined as $\Delta_k V(\mathbf{x},\mathbf{y},t)=V(\mathbf{x},\mathbf{y},t)-V(\mathbf{x},\mathbf{y}+\mathbf{e}_k,t)$. At the end of the sales horizon, all accepted upgradeable products $\mathbf{y}$ must be provided with the remaining capacity $\mathbf{x}$. Furthermore, let $\mathcal{A}$ denote the set of feasible pairs of $\mathbf{x}$ and $\mathbf{y}$ as defined by Gallego and Stefanescu (2009); that is, $(\mathbf{x},\mathbf{y})\in\mathcal{A}$ holds if and only if $\mathbf{x}\geq\mathbf{0}$ and there exists a feasible allocation of upgrades. Then, the expected revenue-to-go can be calculated by

$$V(\mathbf{x},\mathbf{y},t)=\sum_k \lambda_k\left(p_k-\Delta_k V(\mathbf{x},\mathbf{y},t-1)\right)^+ + V(\mathbf{x},\mathbf{y},t-1) \qquad (2)$$

with the boundary conditions $V(\mathbf{x},\mathbf{y},0)=0$ if $(\mathbf{x},\mathbf{y})\in\mathcal{A}$ and $V(\mathbf{x},\mathbf{y},0)=-\infty$ otherwise.

3.2. Structural Properties

Based upon the formulations presented in the previous subsection, we now derive some new structural results that have an important impact on the process of capacity control. The analysis focusses the case that the consumption matrix $\mathbf{A}$ is equal to the identity matrix **I**; that is, each unit of product k consumes exactly one unit of resource k and requires no other resources. Furthermore, we assume a full cascading upgrading model where $\mathcal{V}_k=\{k,\ldots,n\}$. All of these additional assumptions are quite common to many fields of application, such as single-leg airline capacity control and daily car rental capacity control. Note that Gallego and Stefanescu (2009) make very similar assumptions when analyzing the problem's static linear programming approximation with respect to fairness and revenue-optimal limited-cascading solutions.

We first state the following important monotonicity result that holds in respect of ad-hoc upgrading as defined in model (1):

Proposition 1. *If* $\mathbf{A}=\mathbf{I}$ *and* $\mathcal{V}_k=\{k,\ldots,n\}$ *for all products, then*

$$\Delta_q V(\mathbf{x},t)\geq\Delta_{q'}V(\mathbf{x},t) \qquad \text{for all } t,\ \mathbf{x} \text{ with } x_q,x_{q'}>0 \text{ and } q\geq q'.$$

Proof. In the Online Appendix A.1, we prove this inequality by induction over t. □

Proposition 1 means that the resources' opportunity cost is always non-decreasing when moving up the upgrade hierarchy. Note that this result is completely independent of the products' prices. More precisely, prices do not need to increase with the product's position in the upgrade hierarchy.

Proposition 1 has a straightforward implication for the capacity control process, that is, the process of accepting and rejecting incoming requests within the sales horizon. Owing to the opportunity cost monotonicity, it is no longer necessary to consider all the potential upgrade possibilities for an incoming request to identify the minimum opportunity cost, but only the smallest index available in the upgrade hierarchy to which the current request can be upgraded must be identified. If the opportunity cost of the corresponding resource is less than revenue, the request is accepted and assigned to this resource, and if not, it can be rejected, as all larger indices will result in even higher opportunity cost, as implied by Proposition 1. The following algorithm outlines the resulting optimal capacity control step when using the ad-hoc upgrading dynamic program in the investigated setting ($\mathbf{A}=\mathbf{I}$ and $\mathcal{V}_k=\{k,\ldots,n\}$), given that there is an incoming request for product k, the current point of time within the sales horizon is t and the vector of remaining capacity is $\mathbf{x}$:

Algorithm 1.

1. *For an incoming request* k*, find product* $j^*=\min\{j \mid j\geq k\wedge x_j>0\}$.
2. *If* j^* *exists and* $p_k\geq\Delta_{j^*}V(\mathbf{x},t-1)$*:*
   - *accept incoming request* k
   - *upgrade the request to product* j^* ($\mathbf{x}:=\mathbf{x}-\mathbf{e}_{j^*}$)

   Otherwise: reject request k

We next relate the values of models (1) and (2), using the following definition:

Definition 1. *Given* $\mathbf{A}=\mathbf{I}$ *and* $\mathcal{V}_k=\{k,\ldots,n\}$*, a vector of capacity* $\mathbf{x}$ *and a vector of commitments* $\mathbf{y}$ *with* $(\mathbf{x},\mathbf{y})\in\mathcal{A}$*, let* $\mathcal{S}_{\mathbf{xy}}$ *be the set containing all possible feasible allocations of requests* $\mathbf{y}$ *on capacity* $\mathbf{x}$*. For each* $s\in\mathcal{S}_{\mathbf{xy}}$*, let* $\mathbf{h}^{(s)}$ *be the* $m\times 1$ *vector of remaining capacity after allocation.*

Let $\mathcal{H}_{\mathbf{xy}} = \left\{ \mathbf{h}^{(s)} \mid s \in \mathcal{S}_{\mathbf{xy}} \right\}$ *be the set of all possible remaining capacity vectors. Then, we define the function* $\psi : \mathcal{A} \to \mathbb{N}_0^m$ *as follows (*$\mathbb{N}_0^m$ *is the set of all* $m \times 1$ *vectors of non-negative integer values):*

$$\psi(\mathbf{x},\mathbf{y}) = \underset{\mathbf{h}^{(s)} \in \mathcal{H}_{\mathbf{xy}}}{arg\,max} \left\{ \sum_j j \cdot h_j^{(s)} \right\}. \tag{3}$$

The intuition of $\psi(\mathbf{x},\mathbf{y})$ is as follows: The function returns a vector of free capacity that remains after the allocation of the given commitments **y** on the given capacity **x**. The vector is chosen such that resources with a larger index in the upgrade hierarchy are preferably kept free whenever possible. Within the maximization, this is technically achieved by weighting each resource's remaining capacity with its index in the upgrade hierarchy. Algorithmically, a feasible allocation leading to $\psi(\mathbf{x},\mathbf{y})$, given $(\mathbf{x},\mathbf{y})$, can be obtained by successively moving up the product hierarchy from $k = 1,\ldots,n$, allocating capacity for the commitments y_k as low in the hierarchy as possible.

Based on Definition 1, the relationship between models (1) and (2) can be stated as follows:

Proposition 2. *If* $\mathbf{A} = \mathbf{I}$ *and* $\mathcal{V}_k = \{k,\ldots,n\}$ *for all products, then*

$$V(\mathbf{x},\mathbf{y},t) = V(\psi(\mathbf{x},\mathbf{y}),t) \qquad \text{for all } t,(\mathbf{x},\mathbf{y}) \in \mathcal{A}$$

Proof. Without loss of generality, we assume that the control process upgrades requests ad-hoc at the time of sale to the available resource with the smallest index, as described in Algorithm 1. The idea behind the proof is that if requests that would have led to commitments **y** in the postponement variant have been accepted, the free capacity is $\psi(\mathbf{x},\mathbf{y})$ – independent of the order in which the requests have arrived. The proof can be outlined as follows. First, we show that the postponement mechanism and the ad-hoc variant with free capacity $\psi(\mathbf{x},\mathbf{y})$ have an identical set of feasible policies. By induction over t, we then show the equivalence of the value functions as stated in Proposition 2. The complete proof is given in the Online Appendix A.2. □

From Proposition 2, it follows that $V(\mathbf{c},\mathbf{0},T) = V(\mathbf{c},T)$ because, obviously, $\mathbf{c} = \psi(\mathbf{c},\mathbf{0})$. Thus, Proposition 2 implies the equivalence of models (1) and (2) in the setting under consideration. Hence, it is not necessary to postpone any upgrade decisions, but the final resource allocation can immediately be performed after acceptance without any loss in the overall revenue.

Note that both propositions can also be generalized to other resource network structures. For example, it is possible to show that they also hold in the airline network capacity control problem under the condition that upgrade decisions can be made separately for each leg of connecting flights.

4. Dynamic Programming Decomposition Approaches for the Multi-Day Capacity Control Problem

In this section, we focus on multi-day capacity control with planned upgrades, as faced, for example, by hotels and car rental companies. We first slightly change the notation used in Section 3 to make the modeling approach more comprehensible for this application field (Subsection 4.1). Note that we restrict ourselves to considering the ad-hoc variant in which the upgrade decision is made at the time of sale. Because the dynamic programming (DP) formulation is intractable for multi-day problems of real-world sizes, we subsequently develop dynamic programming decomposition approaches in Subsections 4.2 and 4.3, which are based on the traditional DP network decomposition but allow for the simultaneous consideration of upgrades and capacity control decisions.

4.1. Multi-Day Upgrade Setting

A car rental company disposes of the same car types every day, although its capacity may vary. The product portfolio encompasses renting various car types for different lengths of time, and pick-up is usually possible every day. Likewise, hotels are built with a fixed number of rooms of different types and offer accommodation for one or more days. These two examples show a peculiarity found in many service industries where the product portfolio and the resource network have a special structure. In principle, the same products are offered for provision at different points in time, and similarly, constant types of resources are used over time. Providing a product therefore necessitates a certain amount of (physical) resources for a specified time interval. To reflect this multi-day setting, which is obviously a special case of the setting described in Section 3, we slightly change the notation as follows.

We now consider a planning horizon of Γ days and a total of *m* available resource types. For each day $\tau \in \{1,\ldots,\Gamma\}$ of the planning horizon, $c_{r\tau}$ units of resource type $r \in \{1,\ldots,m\}$ are available. $\mathbf{C}$ is the $m \times \Gamma$ matrix containing these initial capacity values for the whole planning horizon. Each product $k \in \{1,\ldots,n\}$ is sold at its price p_k and corresponds to a resource type $r_k \in \{1,\ldots,m\}$. To provide a product, one unit of a resource of type r_k is necessary from the starting day $s_k \in \{1,\ldots,\Gamma\}$ for $l_k \in \{0,\ldots,\Gamma - s_k + 1\}$ days until $s_k + l_k - 1$. All products are upgradeable, so instead of using resources of type r_k, the firm can decide to upgrade the request to another type $r \in \{r_k,\ldots,m\}$, which must be the same for all days. Furthermore, let $\mathbf{A}^{(kr)}$ denote the $m \times \Gamma$ resource consumption matrix of product k assigned to resource type r being defined as $\mathbf{A}^{(kr)} = \left[a_{i\tau}^{(kr)} \right]$ with $a_{i\tau}^{(kr)} = 1$ for

$i = r$ and $s_k \le \tau \le s_k + l_k - 1$, $a_{i\tau}^{(kr)} = 0$ otherwise. In addition, $\mathbf{a}_\tau^{(kr)}$ denotes the τ-th column of $\mathbf{A}^{(kr)}$. As before, the sales horizon is defined by T time periods, and the probability that a request for product k will arrive in period $t \in \{1,\ldots,T\}$ is given by $\lambda_k(t)$. Similar to in Section 3, to ease notation, we omit writing the index sets for the symbols k referring to products, r and i referring to resource types, τ and η referring to days, and t to time periods. For example, the notation $\forall k$ means $k \in \{1,\ldots,n\}$, and $\sum_k$ means $\sum_{k\in\{1,\ldots,n\}}$.

We consider the case that upgrades are decided at the time of sale. Each possible state within the capacity control process can be described by $(\mathbf{X},t)$, where t denotes the considered time period, and the $m \times \Gamma$ matrix $\mathbf{X} = [x_{r\tau}]$ denotes the corresponding free resources' capacity with $\mathbf{X} \le \mathbf{C}$. Let $V(\mathbf{X},t)$ denote the expected revenue-to-go when the current state is $(\mathbf{X},t)$. Furthermore, $\Delta_{kr}V(\mathbf{X},t) = V(\mathbf{X},t) - V(\mathbf{X} - \mathbf{A}^{(kr)},t)$ is the opportunity cost of accepting a request for product k, depending on the assigned resource type $r \ge r_k$. Then, the expected revenue-to-go can be computed recursively through the Bellman equation

$$V(\mathbf{X},t) = \sum_k \lambda_k(t)\left(p_k - \min_{r\ge r_k} \Delta_{kr}V(\mathbf{X},t-1)\right)^+ + V(\mathbf{X},t-1) \qquad (4)$$

with the boundary conditions $V(\mathbf{X},0) = 0$ for $\mathbf{X} \ge \mathbf{0}$, and $V(\mathbf{X},0) = -\infty$ otherwise.

A special case occurs if only one day is considered. Obviously, the upgrade model is a full cascading one, and with respect to the general model notation introduced in Section 3, $\mathbf{A} = \mathbf{I}$ either holds directly for such a single day setting, or an equivalent formulation can be easily constructed where $\mathbf{A} = \mathbf{I}$ holds. Thus, Propositions 1 and 2 derived in Section 3 apply; that is, a simplified control process can be used (see Algorithm 1), and model (4) is completely equivalent to the corresponding postponement formulation.

4.2. Daily DP Decomposition

Because considering multiple days renders the DP formulation (4) intractable, we first propose a decomposition by days, which necessitates only the calculation of less complex single day DPs to obtain an approximation of the opportunity costs necessary to perform a price-based control policy.

The starting point of our considerations is the deterministic linear programming (DLP) formulation corresponding to the multi-day DP (4) (see, e.g., Gallego & Stefanescu, 2009):

$$V^{DLP}(\mathbf{X},t) = max \sum_{k} \sum_{r \geq r_k} p_k z_{kr} \tag{5}$$

subject to

$$\sum_{k} \sum_{r \geq r_k} a_{i\tau}^{(kr)} z_{kr} \leq x_{i\tau} \qquad \text{for all } i \text{ and } \tau \tag{6}$$

$$\sum_{r \geq r_k} z_{kr} \leq D_{kt} \qquad \text{for all } k \tag{7}$$

$$z_{kr} \geq 0 \qquad \text{for all } k \text{ and } r \in \{r_k,\ldots,m\} \tag{8}$$

The decision variables in this model are z_{kr} for all k and $r \in \{r_k,\ldots,m\}$, denoting, with respect to each product k, the number of requests planned for acceptance and upgrading to r. Similar to the traditional DLP without upgrades, constraints (6) and (7) reflect the limitations in capacity and demand, respectively, with D_{kt} denoting the expected aggregated demand-to-come over the remaining periods $t,\ldots,1$.

Let $\pi_{r\tau}$ be the optimal value of the dual variable associated with the capacity constraint (6) for resource type r on day τ. We now choose one day η and relax constraints (6) for all other days using the associated values of $\pi_{r\tau}$ as Lagrange multipliers. Rearranging the objective function, the following formulation is obtained:

$$V^{DLP}(\mathbf{X},t) = max \sum_{k} \sum_{r \geq r_k} \left(p_k - \sum_{\tau \neq \eta} a_{r\tau}^{(kr)} \pi_{r\tau} \right) z_{kr} + \sum_{r} \sum_{\tau \neq \eta} \pi_{r\tau} x_{r\tau} \tag{9}$$

subject to

$$\sum_{k} \sum_{r \geq r_k} a_{i\eta}^{(kr)} z_{k\eta} \leq x_{i\eta} \qquad \text{for all } i \tag{10}$$

$$\sum_{r \geq r_k} z_{kr} \leq D_{kt} \qquad \text{for all } k \tag{11}$$

$$z_{kr} \geq 0 \qquad \text{for all } k \text{ and } r \in \{r_k,\ldots,m\} \tag{12}$$

As implied by linear programming duality, (9)-(12) has the same optimal objective value as (5)-(8). Furthermore, both models have a very similar structure: Ignoring the term $\sum_{r} \sum_{\tau \neq \eta} \pi_{r\tau} x_{r\tau}$ in the objective function (9), model (9)-(12) is the DLP formulation for the capacity control problem if capacity is only constrained on day η and each product k yields a revenue of $p_k - \sum_{\tau \neq \eta} a_{r\tau}^{(kr)} \pi_{r\tau}$, depending on the resource type $r \geq r_k$ used to provide it. That is, if a resource that is considered unrestricted because it belongs to another day is to be used by a product k, its bid price is subtracted from the product's revenue p_k.

On the other hand, if we consider capacity only on day η and define revenues as described above, the optimal total expected revenue can be calculated by the Bellman equation

$$V_\eta\left(\mathbf{x}_\eta,t\right)=\sum_k \lambda_k\left(t\right)\left(\max_{r\geq r_k}\left(p_k-\sum_{\tau\neq\eta}a_{r\tau}^{(kr)}\pi_{r\tau}-\Delta_{kr}V_\eta\left(\mathbf{x}_\eta,t-1\right)\right)\right)^+ +V_\eta\left(\mathbf{x}_\eta,t-1\right) \tag{13}$$

with $\Delta_{kr}V_\eta\left(\mathbf{x}_\eta,t\right)=V_\eta\left(\mathbf{x}_\eta,t\right)-V_\eta\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(kr)},t\right)$ and the boundary conditions $V_\eta\left(\mathbf{x}_\eta,0\right)=0$ if $\mathbf{x}_\eta\geq\mathbf{0}$ and $V_\eta\left(\mathbf{x}_\eta,0\right)=-\infty$ otherwise. Thus, as the DLP formulation of the day η problem is an upper bound for the optimal expected revenue $V_\eta\left(\mathbf{x}_\eta,t\right)$, we have $V^{DLP}\left(\mathbf{X},t\right)\geq V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}$ (see Kunnumkal and Topaloglu (2010) for a similar argumentation regarding a different problem). Furthermore, the optimal value of the day-η-DP $V_\eta\left(\mathbf{x}_\eta,t\right)$ can also be related to the optimal value of the original DP $V\left(\mathbf{X},t\right)$, which is expressed by the following proposition.

Proposition 3. *For each day* η, $V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}$ *is a tighter upper bound on the optimal expected revenue* $V\left(\mathbf{X},t\right)$ *than* $V^{DLP}\left(\mathbf{X},t\right)$*:*

$$V\left(\mathbf{X},t\right)\leq V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\leq V^{DLP}\left(\mathbf{X},t\right) \qquad \text{for all } \eta,\ \mathbf{X},\ t\ .$$

Proof. The second inequality follows from the discussion above. The proof of the first inequality is based on a reformulation of the DP (4) in terms of an equivalent linear program. The inequality can then be shown by induction over t. The complete proof of the first inequality is given in the Online Appendix A.3. □

We now define a control mechanism based on the derived dynamic programming decomposition by days as follows:

Algorithm 2.

1. *Solve the multi-day DLP* (5)-(8) *to derive a set of bid prices* $\left\{\pi_{r\tau}\ \middle|\forall r,\tau\right\}$.

2. *For each day* η*, compute displacement-adjusted revenues* $\bar{p}_{\eta kr}$ *for every product* $k\in\mathcal{K}_\eta=\left\{k\,|\,s_k\leq\eta\leq s_k+l_k-1\right\}$*, that is, for every product that uses capacity on day* η*, depending on the assigned resource type* $r\geq r_k$*:* $\bar{p}_{\eta kr}=p_k-\sum_{\tau\neq\eta}a_{r\tau}^{(kr)}\pi_{r\tau}$.

3. *Solve a daily DP* $V_\eta^{daily}\left(\mathbf{x}_\eta,t\right)$ *for each day* η*, considering only the subset* $\mathcal{K}_\eta$ *of products consuming capacity on that day. With* $\Delta_{kr}V_\eta^{daily}\left(\mathbf{x}_\eta,t\right)=V_\eta^{daily}\left(\mathbf{x}_\eta,t\right)-V_\eta^{daily}\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(kr)},t\right)$*, this*

daily DP is given by

$$V_\eta^{daily}\left(\mathbf{x}_\eta,t\right)=\sum_{k\in\mathcal{K}_\eta}\lambda_k\left(t\right)\left(\max_{r\geq r_k}\left(\bar{p}_{\eta kr}-\Delta_{kr}V_\eta^{daily}\left(\mathbf{x}_\eta,t-1\right)\right)\right)^+ +V_\eta^{daily}\left(\mathbf{x}_\eta,t-1\right).$$

4. *Accept an incoming request for product k if there exists a resource type $r\geq r_k$ with $p_k\geq\sum_{\eta=s_k}^{s_k+l_k-1}\Delta_{kr}V_\eta^{daily}\left(\mathbf{x}_\eta,t-1\right)$.*
   In the case of acceptance, allocate capacity on a resource type $r^*\geq r_k$ with the highest net revenue; that is $p_k-\sum_{\eta=s_k}^{s_k+l_k-1}\Delta_{kr^*}V_\eta^{daily}\left(\mathbf{x}_\eta,t-1\right)=max\left\{p_k-\sum_{\eta=s_k}^{s_k+l_k-1}\Delta_{kr}V_\eta^{daily}\left(\mathbf{x}_\eta,t-1\right)\mid r\geq r_k\right\}$.

The algorithm is a straightforward generalization of the standard decomposition heuristic for revenue management without upgradable products (see, e.g., Talluri & van Ryzin, 2004, chapter 3.4.4) and can be explained as follows: In Steps 1 to 3, we apply the decomposition technique and calculate a separate daily DP for each day of the planning horizon. Note that by defining $V_\eta^{daily}\left(\mathbf{x}_\eta,t\right)$, we slightly modify the Bellman equation as given by (13). In particular, we eliminate all products that do not consume capacity on the respective day η. This modification makes the control process more intuitive and, at the same time, does not change the result, as by construction, the opportunity cost $\Delta_{kr}V_\eta^{daily}\left(\mathbf{x}_\eta,t\right)$ is always identical to $\Delta_{kr}V_\eta\left(\mathbf{x}_\eta,t\right)$. In Step 4, the control step is specified. A request is accepted if and only if its revenue is not less than the total opportunity cost it will cause. Following the general idea of common DP decomposition heuristics (see, e.g., Talluri & van Ryzin, 2004, page 107), we use the sum $\sum_\tau V_\tau^{daily}\left(\mathbf{x}_\tau,t\right)$ to approximate the value function $V\left(\mathbf{X},t\right)$. Therefore, the total opportunity cost is approximated by the sum of the opportunity costs being calculated by the separate daily DPs for the days that are associated with the requested products. Due to the upgrade option, several possibilities to assign the request may exist, so we consider all possibilities and in the case of acceptance, allocate capacity on the resource type with highest net revenue.

4.3. Single Resource DP Decomposition

Although the decomposition presented in the previous subsection drastically reduces the size of the state space, the problem remains computationally intensive for large instances. To further reduce the state space to only one dimension, the approach can be slightly altered to obtain DPs that consider only a single resource's capacity.

The starting point is again the relaxation of constraints (6) in (5)-(8) for all days except the one chosen day η. However, we now also choose a resource type i and further relax the constraints (6) of

all the resource types except i. Thus, only the capacity constraint of resource type i on day η remains. If the term $\sum_r \sum_\tau \pi_{r\tau} x_{r\tau} - \pi_{i\eta} x_{i\eta}$ in the objective function is ignored, the resulting LP is again the DLP formulation of a capacity control problem. In this problem, capacity is restricted only on resource type i on day η. The demand remains exactly as before; that is, all the products and upgrade possibilities are considered, even if they do not consume the capacity of the chosen resource type i on the chosen day η. Furthermore, the revenue of product k is slightly changed to $p_k - \sum_\tau a_{r\tau}^{(kr)} \pi_{r\tau} + a_{i\eta}^{(kr)} \pi_{i\eta}$, depending on the resource type $r \geq r_k$ used. As before, the bid price of every resource used to provide the product but considered unrestricted is subtracted, but this is now also done with resource types $r \neq i$ on day η. The corresponding Bellman equation for this problem is

$$V_{i\eta}\left(x_{i\eta},t\right) = \sum_k \lambda_k(t) \left(\max_{r \geq r_k} \left(p_k - \sum_\tau a_{r\tau}^{(kr)} \pi_{r\tau} + a_{i\eta}^{(kr)} \pi_{i\eta} - \Delta_{kr} V_{i\eta}\left(x_{i\eta},t-1\right) \right) \right)^+ + V_{i\eta}\left(x_{i\eta},t-1\right) \tag{14}$$

with $\Delta_{kr} V_{i\eta}\left(x_{i\eta},t\right) = V_{i\eta}\left(x_{i\eta},t\right) - V_\eta\left(x_{i\eta} - a_{i\eta}^{(kr)},t\right)$ and the boundary conditions $V_{i\eta}\left(x_{i\eta},0\right) = 0$ if $x_{i\eta} \geq 0$ and $V_{i\eta}\left(x_{i\eta},0\right) = -\infty$ otherwise.

The logic behind the equation is quite intuitive: While in (13), all resources on the considered day η have been modeled explicitly, there is now only one single resource – that is, resource type i on day η – which defines the DP's state space. Nevertheless, the upgrade decision itself is again explicitly modeled in the DP; the difference is that the upgrade possibilities requiring a different resource type $r \neq i$ on day η are no longer limited in capacity, but instead come at the additional cost of the resource's bid price $\pi_{r\eta}$.

It is straightforward that a proposition similar to Proposition 3 holds for $V_{i\eta}\left(x_{i\eta},t\right)$ as well.

Proposition 4. *For each day η and resource type i, $V_{i\eta}\left(x_{i\eta},t\right) + \sum_r \sum_\tau \pi_{r\tau} x_{r\tau} - \pi_{i\eta} x_{i\eta}$ is a tighter upper bound on the optimal expected revenue $V\left(\mathbf{X},t\right)$ than $V^{DLP}\left(\mathbf{X},t\right)$:*

$$V\left(\mathbf{X},t\right) \leq V_{i\eta}\left(x_{i\eta},t\right) + \sum_r \sum_\tau \pi_{r\tau} x_{r\tau} - \pi_{i\eta} x_{i\eta} \leq V^{DLP}\left(\mathbf{X},t\right) \qquad \text{for all } \eta,\ i,\ \mathbf{X},\ t.$$

Proof. The proof is very similar to the proof for Proposition 3 and is omitted.

The next proposition relates the two upper bounds on $V\left(\mathbf{X},t\right)$ obtained by considering all the resource types on one day and the bound constructed by using only one resource type on that day.

Proposition 5. $\quad V_\eta\left(\mathbf{x}_\eta,t\right) - \sum_{r \neq i} \pi_{r\eta} x_{r\eta} \leq V_{i\eta}\left(x_{i\eta},t\right) \qquad \text{for all } \eta,\ i,\ \mathbf{X},\ t.$

Proof. In the Online Appendix A.4, we prove the inequality by induction over t, making use of the fact that all policies that are feasible in the problem $V_\eta\left(\mathbf{x}_\eta,t\right)$, which considers all resource types on day η, are also feasible if only resource type i is considered in $V_{i\eta}\left(x_{i\eta},t\right)$. □

By combining Propositions 3, 4, and 5, it follows that the daily decomposition provides a tighter upper bound on the optimal expected revenue of the multi-day problem than the decomposition to single resources:

$$V\left(\mathbf{X},t\right)\le V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{i\tau}x_{i\tau}\le V_{i\eta}\left(x_{i\eta},t\right)+\sum_r\sum_\tau\pi_{r\tau}x_{r\tau}-\pi_{i\eta}x_{i\eta}\le V^{DLP}\left(\mathbf{X},t\right). \quad (15)$$

However, this comes at the cost of the state space remaining multi-dimensional in the daily decomposition.

The definition of a control mechanism for the single resource decomposition is very similar to the one that was defined for the daily decomposition in the previous subsection:

Algorithm 3.

1. *Solve the multi-day DLP* (5)-(8) *to derive a set of bid prices* $\left\{\pi_{r\tau}\ \middle|\forall r,\tau\right\}$.
2. *For each day* η *and resource type* i*, compute displacement-adjusted revenues for every product* $k\in\mathcal{K}_{i\eta}=\left\{k\,|\,s_k\le\eta\le s_k+l_k-1 \text{ and } r_k\le i\right\}$*, that is, for every product that can potentially use capacity on day* η *and resource type i, depending on the assigned resource type* $r\ge r_k$*:* $\bar{p}_{\eta ikr}=p_k-\sum_\tau a_{r\tau}^{(kr)}\pi_{r\tau}+a_{i\eta}^{(kr)}\pi_{i\eta}$.
3. *Solve a single resource DP* $V_{i\eta}^{single}\left(\mathbf{x}_\eta,t\right)$ *for each day* η *and resource type* i*, considering only the subset* $\mathcal{K}_{i\eta}$ *of products. With* $\Delta_{kr}V_{i\eta}^{single}\left(x_{i\eta},t\right)=V_{i\eta}^{single}\left(x_{i\eta},t\right)-V_{i\eta}^{single}\left(x_{i\eta}-a_{i\eta}^{(kr)},t\right)$*, this single resource DP is given by*

$$V_{i\eta}^{single}\left(x_{i\eta},t\right)=\sum_{k\in\mathcal{K}_{i\eta}}\lambda_k\left(t\right)\left(\max_{r\ge r_k}\left(p_k-\sum_\tau a_{r\tau}^{(kr)}\pi_{r\tau}+a_{i\eta}^{(kr)}\pi_{i\eta}-\Delta_{kr}V_{i\eta}^{single}\left(x_{i\eta},t-1\right)\right)\right)^+ + V_{i\eta}^{single}\left(x_{i\eta},t-1\right).$$

4. *Accept an incoming request for product* k *if there exists a resource type* $r\ge r_k$ *with* $p_k\ge\sum_{\eta=s_k}^{s_k+l_k-1}\Delta_{kr}V_{r\eta}^{single}\left(x_{r\eta},t-1\right)$.
   In the case of acceptance, allocate capacity on a resource type $r^*\ge r_k$ *with the highest net revenue; that is* $p_k-\sum_{\eta=s_k}^{s_k+l_k-1}\Delta_{kr^*}V_{r\eta}^{single}\left(x_{r\eta},t-1\right)=\max\left\{p_k-\sum_{\eta=s_k}^{s_k+l_k-1}\Delta_{kr}V_{r\eta}^{single}\left(x_{r\eta},t-1\right)\,|\,r\ge r_k\right\}$.

While Step 1 remains unchanged, the displacement-adjusted revenues calculated in Step 2 are now also resource type specific. In Step 3, independent DPs are calculated for each day and resource type. In Step 4, the opportunity costs of the single resource DPs $\Delta_{r\eta}V_{r\eta}^{single}\left(x_{r\eta},t-1\right)$ are used, everything else being unchanged. Within set $\mathcal{K}_{i\eta}$, we now only need to consider the products that could potentially use the current resource type *i* on day η, either directly or by upgrading. In Step 4, now the opportunity costs given by the single resource DPs are summed to calculate the approximation of the total opportunity cost.

It should be noted that although single resource decomposition is presented here in the context of a full cascading upgrading model in a multi-day revenue management setting, it can be applied much more broadly. In fact, it is not necessary to assume a specific upgrading model or the structure of a multi-day revenue management problem, as the approach also works with the general network revenue management problem with arbitrary upgrade hierarchies as given by (1). The approach can therefore also be applied to traditional capacity control settings occurring, for example, in the airline industry.

Finally, note that the theoretical results derived in Propositions 3, 4, and 5 already give an indication of the performance of the decomposition approaches when being applied to control either a real or a simulated booking process. As Talluri (2009) points out, "tight bounds are of great interest as it appears from empirical studies and practical experience that models that give tighter bounds also lead to better controls (better in the sense that they lead to more revenue)." Therefore, what we can conclude from our analysis is the following: The fact that our decomposition approaches give tighter upper bounds on the optimal revenue than the DLP suggests that the approaches will perform better when used to control a booking process. In addition, it is likely that the daily decomposition will perform better than the single resource decomposition, because the theoretical bound is tighter. Nevertheless, this is something that must be validated within a simulation study, which we perform in the following section.

5. Computational Results

In this section, we report the results of a simulation study that demonstrates the applicability and relative performance of the dynamic programming decomposition approaches derived above. The simulations are based on real-world demand and capacity data provided by a major car rental company. To protect the interests of this company and to guard against the release of sensitive data, we

have held back (or modified) certain details. The experiments were conducted on an Intel Xeon processor-based server (Xeon 5450 CPU, RAM of 16 GB, operating system Microsoft Windows Server 2008 Enterprise 64-Bit SP2). The algorithms were coded in C# running on the Microsoft .NET Framework 3.5 SP 1 and were linked to the ILOG CPLEX 12.1 64-Bit optimization routines.

In the study, we analyze a decentralized control approach; that is, capacity control is performed separately for each rental station. We limit our investigation to business stations serving corporate customers at pre-negotiated rates. At these stations, the available car capacity is only a bottleneck on peak-days, which are usually Tuesday and/or Wednesday. Hence, modeling capacity only on these days and assuming that capacity is unlimited on the other days of the week can be justified. Furthermore, an analysis of historical demand data from the stations under consideration shows that the fraction of rentals that require more than six consecutive days is very small and can thus be ignored. In particular, this means that there are no requests that extend over several weeks' midweek peaks, so the simulation study can be performed on a weekly basis, considering only requests that require one or both of the midweek peaks.

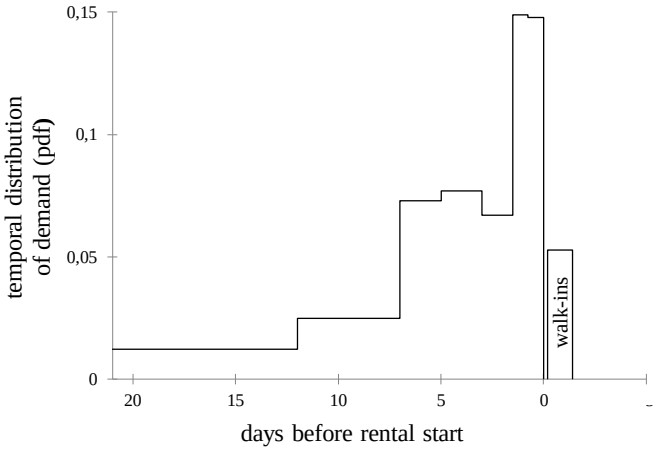

Figure 1: Typical demand pattern

For each station under consideration, we received unconstrained demand data calculated from historical data averaged over many weeks. The demand data are given by the requested car type, the length of rent, snapshot, peaks that would be blocked by the requested rent, the revenue, and the variable cost. The car type is from one of the three groups: economy, compact, or full-size. The length of rent varies between 1 and 6 days, as described above. Data are available for a total of nine snapshots taken at 21, 12, 7, 5, 3, 1.5, 0.75, and 0 days before the rental starts, plus additional walk-in customers (see Figure 1 for a typical demand pattern over time). The total net revenue varies be-

tween 25 and 400 Euros, depending on the requested car type and the length of rent. Given these data, we calculate the arrival parameters as follows (see, e.g., Subramanian, Stidham Jr. & Lautenbacher, 1999):

1. By combining the snapshots of the different products on a common time axis, subintervals with time-homogeneous demand are identified.
2. Each interval s is divided into N_s time periods.
3. The booking probability $\lambda_k(t)$ for product k, defined by car type, length of rent, revenue, and cost in time period t (in interval s) is calculated as

$$\lambda_k(t) = \frac{\#\text{ requests for product } k \text{ in interval } s}{N_s}.$$

The number of time periods for interval *s* is calculated so that

$$\sum_k \lambda_k(t) < 1.$$

To investigate how the ratio of overall capacity and demand influences the performance of the different methods, we use a parameter α to proportionally scale demand. The value $\alpha = 1$ corresponds to the case with the original demand data obtained from our industry partner. A value, for example, of $\alpha = 1.1$, means that we simultaneously scale up the original demand values at all demand snapshots by 10% and then perform the above Steps 1 to 3 to obtain the corresponding arrival parameters for the simulation.

We implement the following methods:

1. *DP*: This method implements the multi-day dynamic programming approach incorporating adhoc upgrades as given by equation (4).
2. *DPD-D*: This is the daily DP decomposition described in Subsection 4.2.
3. *DPD-S*: This is the single resource DP decomposition described in Subsection 4.3.
4. *SUCC*: This method is based on successive planning and mimics an upgrade control that is currently widely used in commercial revenue management software systems. In the first step, virtual capacities are determined for each of the different car types by estimating in advance the number of upgrades that must be performed. For example, if ten upgrades from economy to compact car are required on a certain day, the capacity of compact cars is reduced by ten, and that of economy cars is increased by the same number. To realize this step, we use the primal solution of model (5)-(8)

and adjust the different car types' capacities according to the optimal values of the decision variables. In the second step, the resulting virtual capacity vector is fed into a traditional control method without upgrade capabilities, which then performs capacity control over time. In our study, we use a traditional deterministic linear programming model (see, e.g., Talluri & van Ryzin, 2004) to determine bid prices in this step. Both steps are iterated three times during the control process; that is, the virtual capacities as well as the bid prices are twice reoptimized according to the current demand information and capacity load.

5. *FCFS*: This method is a simple first-come-first-served control. Requests are accepted as long as they can be fulfilled by the remaining capacity. Upgrades are undertaken if necessary, moving up the upgrade hierarchy successively. The revenue obtained by this method is used to judge the relative performance of the methods described above.

6. *EXPOST*: This method calculates the perfect hindsight optimal revenue that can be obtained if full information on the incoming demand is used. For each simulation run, a model of type (5)-(8) is solved, optimally allocating capacity to the current demand stream's requests, which are used as the RHS of constraints (7) instead of the forecasted demand. The obtained revenue serves as an upper bound for all the other methods' output.

We generate 200 streams of demand and process each stream using the methods described above, given their applicability in the setting under consideration.

5.1. Example 1: One Peak-Day

This example analyzes a real-world car rental station of our industry partner disposing of 15 economy cars, 40 compact cars, and 40 full-size cars. There is one peak in the weekly demand, such that there is a bottleneck in capacity only on that day, say, Tuesday. As described above, capacity is modeled only on Tuesday and considered unrestricted on all other days. Depending on the demand factor α, the sales horizon consists of 103-303 time periods, with an average of 79-232 rental requests. To control the booking process, we use all mechanisms except DPD-D, which is not applicable if only one day is considered.

	% FCFS		% DP		% DPD-S		% SUCC	
α	AVG	99% Conf. Int.	AVG	99% Conf. Int.	AVG	99% Conf. Int.	AVG	99% Conf. Int.
0.5	100.00	(98.37; 101.63)	100.00	(98.37; 101.63)	100.00	(98.37; 101.63)	98.42	(96.78; 100.06)
0.6	99.53	(98.20; 100.86)	99.89	(98.52; 101.26)	99.69	(98.35; 101.04)	97.03	(95.65; 98.41)
0.7	93.71	(92.65; 94.78)	99.37	(98.10; 100.65)	97.48	(96.37; 98.60)	95.63	(94.48; 96.78)
0.8	85.43	(84.52; 86.34)	99.32	(98.38; 100.26)	98.64	(97.63; 99.64)	96.34	(95.34; 97.35)
0.9	81.95	(81.13; 82.78)	99.12	(98.20; 100.04)	98.64	(97.75; 99.53)	95.55	(94.69; 96.41)
1.0	77.67	(76.90; 78.44)	99.10	(98.23; 99.96)	98.44	(97.63; 99.25)	93.40	(92.62; 94.19)
1.1	74.72	(73.98; 75.46)	99.09	(98.21; 99.98)	97.65	(96.94; 98.36)	91.94	(91.17; 92.71)
1.2	72.03	(71.30; 72.75)	99.19	(98.34; 100.04)	96.44	(95.82; 97.07)	91.54	(90.74; 92.33)
1.3	69.07	(68.38; 69.76)	99.13	(98.33; 99.93)	94.86	(94.27; 95.46)	91.48	(90.81; 92.14)
1.4	67.83	(67.15; 68.51)	99.07	(98.33; 99.81)	98.36	(97.70; 99.02)	92.85	(92.08; 93.63)
1.5	66.28	(65.61; 66.95)	99.01	(98.30; 99.71)	97.59	(97.02; 98.16)	93.02	(92.36; 93.67)

Table 1: Simulated revenues (Example 1)

Table 1 summarizes the revenues obtained under the different methods relative to EXPOST, along with the corresponding 99% confidence intervals. For example, a value of 99.53 means that the method under consideration attains 99.53% of the revenue obtained by EXPOST. In addition, Table A.1 in the Online Appendix summarizes the load factor, the percentage of requests accepted, and the percentage of accepted customers who are upgraded to a higher car type than requested. We observe that if capacity is not scarce ($\alpha \leq 0.6$), all requests are accepted and the mechanisms yield almost the same revenue as EXPOST. However, when demand exceeds capacity, revenue management becomes relevant, and there are remarkable differences between the mechanisms. For all $\alpha > 0.6$, FCFS clearly yields the worst result. When the proportion of upgrades is examined, one reason for this becomes obvious: by construction, FCFS does not control availability and uses upgrades to superior car types extensively to satisfy demand for smaller cars even though there is enough higher valued, later-arriving demand to utilize the superior car type to the full. The other methods are all more-or-less successful in using capacity for the most valuable requests. While SUCC already attains a considerably higher revenue than FCFS, this can be further improved by DPD-S and DP, which always yields slightly more than 99% of EXPOST's revenue, and is – in terms of expected revenue – the optimal method in real-world applications where perfect hindsight information is not available. The dominance of DP over DPD-S reflects the left inequality of Proposition 4; nevertheless, in most cases, DPD-S performs only slightly worse than DP. Furthermore, it turns out that both DP as well as DPD-S perform significantly better than SUCC at the 99% level for all values of α. Table A.2 in the Online Appendix summarizes the corresponding confidence intervals for the revenue improvements.

Additionally, note that from Proposition 2 it directly follows that postponement and ad-hoc upgrading are equivalent in terms of revenue in this one peak-day setting. The revenue obtained by DP is therefore identical to the result we would obtain if we would have postponed the upgrade decision.

Table 2 shows computational time statistics for DP, DPD-S, and SUCC. With respect to DP, the column V shows the time required for calculating the value function for all states, while the column control is the time necessary to handle all the customer requests of one demand stream. The third and fourth columns present the corresponding values for DPD-S. Here, V also includes the time to calculate the displacement adjusted revenues; that is, the time required to solve the dual of the corresponding network deterministic linear program. The last column is the total time needed by SUCC to process all the requests of one stream. This table mainly clarifies that all methods are computationally feasible for the rental station considered here. Certainly, calculating the DP value function is very time consuming. However, this is a batch process that the rental company would run once. Likewise, the DPD-S value function can be pre-calculated, but this requires only a fraction of the time necessary for DP. The times required to handle a request are rather similar and negligible in all three methods.

	DP		DPD-S		SUCC
α	V [h:mm:ss]	control [s]	V [s]	control [s]	total [s]
0.5	0:33:41	0.187	8.171	0.234	0.171
0.6	0:51:31	0.265	11.750	0.328	0.234
0.7	1:11:05	0.328	21.250	0.375	0.265
0.8	1:29:06	0.421	17.093	0.500	0.296
0.9	1:46:05	0.453	18.359	0.531	0.312
1.0	2:09:41	0.562	21.562	0.625	0.359
1.1	2:31:27	0.625	32.875	0.687	0.375
1.2	2:54:29	0.765	24.671	0.828	0.484
1.3	3:14:49	0.812	27.718	0.843	0.437
1.4	3:27:31	0.953	30.687	1.031	0.546
1.5	3:59:47	0.968	36.546	1.062	0.421

Table 2: CPU times for DP, DPD-S, and SUCC (Example 1)

Overall, from our investigation of this example, we can conclude that the proposed decomposition approach DPD-S performs particularly well in terms of realized revenue at very low computational effort. More precisely, if the computational time becomes an issue for a larger station so that DP is no longer applicable, our results indicate that the rental company could safely resort to the less time-consuming DPD-S and would still obtain high quality results.

5.2. Example 2: Two Peak-Days

The second example refers to a real-world car rental station where capacity forms a bottleneck on two days of the week: Tuesday and Wednesday. Thus, the station's capacity of 10 economy cars, 25 compact cars, and 30 full-size cars is modeled explicitly on these two days. Depending on the demand factor α, the sales horizon consists of 99-291 time periods, with an average of 76-223 rental requests. Compared to the previous example, the state space of the two-day dynamic program is so large that DP is too computationally expensive to be applied here. However, as multiple days are considered, DPD-D is applicable in this setting.

	% FCFS		% DPD-D		% DPD-S		% SUCC	
α	AVG	99% Conf. Int.	AVG	99% Conf. Int.	AVG	99% Conf. Int.	AVG	99% Conf. Int.
0.5	99.73	(98.21; 101.25)	99.86	(98.34; 101.39)	99.79	(98.27; 101.31)	94.68	(93.12; 96.23)
0.6	95.02	(93.77; 96.27)	99.02	(97.63; 100.40)	98.11	(96.81; 99.41)	93.02	(91.60; 94.44)
0.7	88.67	(87.60; 89.75)	98.14	(97.01; 99.27)	95.98	(94.90; 97.05)	93.63	(92.50; 94.76)
0.8	84.61	(83.66; 85.56)	97.72	(96.69; 98.76)	96.84	(95.83; 97.85)	93.16	(92.09; 94.23)
0.9	80.39	(79.50; 81.29)	97.37	(96.39; 98.35)	96.74	(95.77; 97.71)	93.22	(92.19; 94.25)
1.0	77.16	(76.36; 77.97)	97.52	(96.68; 98.36)	95.98	(95.16; 96.80)	92.84	(91.99; 93.70)
1.1	74.39	(73.57; 75.20)	97.46	(96.65; 98.26)	95.23	(94.35; 96.11)	92.68	(91.82; 93.53)
1.2	71.87	(71.04; 72.70)	97.54	(96.69; 98.38)	94.97	(94.04; 95.89)	93.05	(92.25; 93.85)
1.3	70.11	(69.34; 70.88)	97.51	(96.74; 98.27)	97.20	(96.40; 98.01)	93.50	(92.72; 94.28)
1.4	68.78	(68.00; 69.55)	97.41	(96.59; 98.23)	97.08	(96.28; 97.89)	92.85	(92.11; 93.58)
1.5	67.42	(66.69; 68.15)	97.44	(96.64; 98.24)	97.12	(96.31; 97.92)	92.69	(91.89; 93.48)

Table 3: Simulated revenue (Example 2)

Table 3 summarizes the revenues obtained under the different methods with respect to EXPOST. In addition, Table A.3 in the Online Appendix summarizes the load factor, the percentage of requests accepted, and the percentage of accepted customers who are upgraded. Again, we observe that if capacity is not scarce ($\alpha = 0.5$), the mechanisms accept almost all requests and yield similar revenues as EXPOST. When demand is stronger, FCFS quickly becomes far worse than the other mechanisms, as it grants upgrades as long as capacity is available. DPD-D constantly attains very good revenues, with even the lowest just less than 2.6 percentage points from the EXPOST value.

Please note that, because both DP decomposition models are shown to give theoretical upper bounds on the optimal expected revenue calculated by the DP (Propositions 3 and 4 in Section 4), they could be used as reference values as well. For example, for $\alpha = 1.0$, the tightest bound we obtain from the decomposition models is at 99.10% of the average ex post value and therefore it is tighter than the ex post upper bound in this example. The bounds for all values of α are summarized in Table A.4 in the Online Appendix with respect to EXPOST. In this example, the theoretical

bounds obtained by DPD-D are always tighter than the one obtained by EXPOST, while DPD-S gives tighter and weaker bounds than EXPOST, depending on the specific value of α. Additionally, there are a few cases where the theoretical upper bound is even smaller than the average revenue we obtained from our simulation of 200 demand streams (see, e.g., $\alpha = 0.5$).

Furthermore, it turns out that DPD-D, in most cases, yields significantly more revenue than DPD-S (see Table A.5 in the Online Appendix, which shows the average improvement of DPD-D over DPD-S as well as the confidence intervals at three different levels). This result is consistent with the theoretical results from Section 4, which have already indicated that DPD-D performs better than DPD-S in terms of total achieved revenue. Only in the case of $\alpha = 0.5$ – that is, if capacity is not scarce – is the DPD-D gain not significant, even at the 90% level. Similar to example 1, it also turns out that both DPD-S and DPD-D significantly exceed SUCC's revenue. Table A.6 in the Online Appendix summarizes the corresponding confidence intervals for the revenue improvements at the 99% level.

	DPD-D		DPD-S		SUCC
α	V [h:mm:ss]	control [s]	V [m:ss]	control [s]	total [s]
0.5	1:08:48	1.062	0:35.436	1.359	0.593
0.6	1:22:41	1.375	0:39.375	1.625	0.640
0.7	1:50:30	1.828	0:50.921	2.062	0.828
0.8	2:14:39	2.125	1:09.046	2.437	0.765
0.9	2:34:30	2.625	0:57.781	3.000	0.796
1.0	2:52:58	3.078	1:08.171	3.062	0.828
1.1	3:28:01	3.406	1:10.687	3.578	0.906
1.2	3:45:21	4.093	1:38.421	4.734	1.093
1.3	4:11:15	4.937	1:54.562	5.500	1.109
1.4	5:00:52	5.437	1:51.640	5.953	1.218
1.5	4:53:24	6.093	2:16.781	6.343	1.171

Table 4: CPU times for DPD-D, DPD-S and SUCC (Example 2)

Table 4 summarizes DPD-D, DPD-S, and SUCC's computational times. The meaning of the columns has been explained in detail in the previous subsection. Likewise, the main message is that all methods are computationally feasible. Calculating the value function is especially time-consuming for DPD-D, but this is a batch process that does not need to be executed online when customer requests have to be decided.

Overall, the numerical investigation of example 2 shows that both decomposition approaches proposed in this paper are applicable and lead to good results. In particular, they both significantly outperform SUCC. If computational time is scarce or becomes too long for larger stations, DPD-S is

preferable, as it is computationally much less intensive than DPD-D, due to its one-dimensional state space (also see Subsection 5.3). These savings in computation time come at the price of significant revenue losses compared to DPD-D; nonetheless, the revenue is still much higher than the one resulting from the application of SUCC.

5.3. Example 3: More than Two Peak-Days

In further practical applications, there might exist settings where capacity is scarce on more than two consecutive days. Generally, the methods we propose are still applicable in such cases. By construction, both DPD-D as well as DPD-S scale up linearly with the number of scarce resources in terms of runtime, everything else being unchanged. For example, if we apply the DPD-D approach and need to explicitly model an additional peak-day, there is just another daily DP that must be solved separately and that must additionally be considered when the opportunity cost for incoming requests is calculated. However, even though runtime scales up linearly, DPD-D will surely often become too computationally intensive due to the multidimensional state space of each single daily DP.

In what follows, we give an idea of the computational times when capacity is scarce on more than two days. As these types of problems do not exist for our industry partner, we define an artificial example as follows: We consider a station with a capacity of 10 economy cars, 20 compact cars, and 30 full-size cars. The planning horizon consists of 14 consecutive days. The prices of a one-day rental are €50.00 (economy), €62.50 (compact), and €95.00 (full-size). In addition to one-day rentals, there are also two-day and three-day rentals that can begin on each day within the planning horizon and are priced at 1.9 and 2.5 times the one-day rental price, respectively. To derive demand values, we assume an expected demand of 30.5 customers requesting a rental each day. A total of 55% of this demand is for one-day rentals, 27% for two-day rentals, and 18% for three-day rentals. These shares are chosen such that if all demand was accepted, a third of the cars in use at any day would be for each length of rent. Regarding the different car types, we assume that demand splits up into 25% demand for economy cars, 50% for compact cars, and 25% for full-size cars. On the basis of the resulting expected demand values, we assume a time-homogeneous demand arrival process for each product and use the procedure described at the beginning of Section 5 to generate requests and distribute them over time, which leads to a total of 1,107 time periods.

	% FCFS		% DPD-D		% DPD-S		% SUCC	
#sdays	AVG	99% Conf. Int.	AVG	99% Conf. Int.	AVG	99% Conf. Int.	AVG	99% Conf. Int.
1	83.99	(82.96; 85.02)	98.05	(99.33; 96.78)	97.32	(95.96; 98.68)	93.66	(92.41; 94.91)
2	85.91	(85.21; 86.62)	97.73	(96.80; 98.66)	96.12	(95.19; 97.04)	93.23	(92.23; 94.23)
3	86.84	(86.20; 87.48)	97.64	(96.84; 98.43)	96.27	(95.58; 96.95)	93.55	(92.78; 94.33)
4	87.52	(86.99; 88.05)	-	-	96.30	(95.68; 96.92)	93.40	(92.68; 94.12)
5	87.99	(87.50; 88.48)	-	-	96.44	(95.91; 96.97)	93.86	(93.23; 94.50)
6	88.53	(88.10; 88.97)	-	-	96.53	(96.07; 96.99)	94.00	(93.44; 94.56)
7	88.83	(88.46; 89.20)	-	-	96.52	(96.08; 96.95)	94.23	(93.72; 94.75)
8	89.00	(88.63; 89.38)	-	-	96.66	(96.27; 97.06)	94.37	(93.90; 94.84)
9	89.41	(89.05; 89.76)	-	-	96.83	(96.44; 97.22)	94.61	(94.18; 95.04)
10	89.53	(89.20; 89.85)	-	-	96.77	(96.42; 97.11)	94.43	(94.02; 94.84)
11	89.52	(89.20; 89.85)	-	-	96.80	(96.46; 97.14)	94.56	(94.18; 94.95)
12	89.70	(89.41; 89.98)	-	-	96.90	(96.59; 97.21)	94.68	(94.33; 95.03)
13	90.32	(90.05; 90.59)	-	-	97.02	(96.74; 97.30)	94.84	(94.51; 95.16)
14	90.72	(90.43; 91.00)	-	-	96.93	(96.64; 97.22)	94.89	(94.55; 95.24)

Table 5: Simulated revenue (Example 3)

We now apply the different capacity control methods as defined in Subsection 5.1. Table 5 summarizes the revenues obtained under the different methods with respect to EXPOST depending on #sdays, that is, the number of consecutive days starting from day 1 of the planning horizon that are scarce and therefore modeled explicitly in the methods. The experiments that did not complete within 48 hours of computation are marked with "-". As additional information, the load factor, the percentage of requests accepted, and the percentage of accepted customers who are upgraded are given by Table A.7 in the Online Appendix.

	DPD-D		DPD-S		SUCC
	V	control	V	control	total
#sdays	[hh:mm:ss]	[s]	[h:mm:ss]	[s]	[s]
1	11:59:00.391	11.746	0:10:19.464	12.136	17.144
2	24:15:43.337	11.949	0:17:42.086	12.729	17.035
3	36:29:44.978	11.840	0:22:46.911	12.355	17.191
4	-	-	0:28:22.906	12.682	17.160
5	-	-	0:34:52.519	12.402	17.550
6	-	-	0:40:46.111	12.121	17.222
7	-	-	0:47:02.806	12.261	17.472
8	-	-	0:51:53.358	12.168	17.830
9	-	-	0:58:38.914	12.838	17.612
10	-	-	1:05:07.684	12.136	17.378
11	-	-	1:11:17.953	12.589	17.737
12	-	-	1:20:51.599	12.261	17.971
13	-	-	1:29:28.821	12.058	17.222
14	-	-	1:36:22.411	12.854	17.924

Table 6: CPU times for DPD-D, DPD-S and SUCC (Example 3)

Table 6 summarizes DPD-D, DPD-S, and SUCC's computational times. As suggested by the theoretical analysis, the methods indeed scale linearly with the number of scarce days. However, it also

turns out that DPD-D already requires more than 48 hours of computational time when more than 3 days are scarce, so DPD-S is preferable in these cases. For example, it takes more than 7 days to compute DPD-D with 14 scarce days. More generally, the computational example also shows that the methods are basically computationally applicable even when a significantly larger number of time periods are involved. Everything else being equal, it follows from theory that the number of time periods has a linear impact on the computational times of DPD-D and DPD-S.

6. Conclusions

In this paper, we address the problem of integrating revenue management's capacity control with upgrade decision making. We derive new structural properties for an integrated dynamic programming formulation (DP) of this problem that was recently proposed by Gallego and Stefanescu (2009). Depending on the field of application, these new properties simplify the resulting control process under certain conditions, for example, when only one leg or day is considered. In particular, we prove the monotonicity of the resources' opportunity cost, allowing to sequentially search for upgrade opportunities in the product hierarchy, and we show the equivalence of ad-hoc upgrading and upgrade postponement.

As the DP is often not applicable to practical problem sizes due to its high-dimensional state space, we subsequently propose two different dynamic programming decomposition approaches that allow the heuristic solution of real-world, multi-day instances of the problem. The first approach is specifically suited to multi-day revenue management, as found, for example, in the hotel or car rental industry. This approach decomposes the full network capacity control problem to a series of single-day problems (DPD-D). The second approach decomposes the network even more and splits the problem into one-dimensional single resource dynamic programs (DPD-S). Analytically, we show that both decomposition approaches imply tighter upper bounds on the value of the original dynamic program than a static linear approximation with upgrades (DLP). Furthermore, we prove that the bound implied by DPD-D is tighter than the one of DPD-S. These theoretical results make it likely that both decomposition approaches perform better in applications than the DLP, and it can be expected that the daily decomposition will perform better than the single resource decomposition, because the theoretical bound is tighter.

We run an exhaustive series of computational tests based on real-world data obtained from a major car rental company. Our investigations demonstrate that both decomposition approaches perform

well in practice and show a remarkably good revenue performance. With respect to the order of the obtained revenues compared to DP, our theoretical results are confirmed. Furthermore, we show that the new heuristics consistently outperform a standard approach based on successive planning and currently widely used to manage the upgrade issue in revenue management software systems. Our experiments also verify the practical feasibility of the two different approaches with respect to runtime. Clearly, DPD-D is the more time-consuming method. However, the computationally intensive part can be realized in an overnight batch process, which still makes it practically feasible in most cases. Nevertheless, if time becomes an issue due to the size of the resource network and the capacity, resorting to DPD-S is a good idea, as runtime is not an issue due to the one-dimensional state space and the revenue is often not much lower than that of DPD-D.

The results derived in this paper have remarkable implications for today's revenue management practice when faced with upgrades. The methods investigated in this paper can be very easily implemented and may significantly improve the revenues of especially car rental companies, which are usually quite flexible and not committed to specific software systems with given interfaces and parameters defined, for example, by distribution systems. Nevertheless, these approaches also appear to be promising for other applications, as they can be easily adapted to other typical settings such as the standard airline case.

Acknowledgements

The authors thank the editor and three anonymous referees for their detailed and constructive comments that helped improving this paper.

References

Alstrup, J., Boas, S., Madsen, O. B. G., & Vidal, R. V. V. (1986). Booking policy for flights with two types of passengers. *European Journal of Operational Research, 27*, 274-288.

Belobaba, P. P. (1987). Air travel demand and airline seat inventory management (Ph.D. thesis). Flight Transportation Laboratory, Massachusetts Institute of Technology.

Bertsimas, D., & de Boer, S. (2005). Simulation-based booking limits for airline revenue management. *Operations Research, 53*, 90-106.

Carroll, W. J., & Grimes, R. C. (1995). Evolutionary change in product management: Experiences in the car rental industry. *Interfaces, 25*(5), 84-104.

Chiang, W., Chen, J. C. H., & Xu, X. (2007). An overview of research on revenue management: Current issues and future research. *International Journal of Revenue Management, 1*, 97-128.

Edelstein, M., & Melnyk, M. (1977). The pool control system. *Interfaces, 8*(1:2), 21-36.

Erdelyi, A., & Topaloglu, H. (2009). Separable approximations for joint capacity control and overbooking decisions in network revenue management. *Journal of Revenue and Pricing Management, 8*, 3-20.

Erdelyi, A., & Topaloglu, H. (2010). A dynamic programming decomposition method for making overbooking decisions over an airline network. *INFORMS Journal on Computing, 22*, 443-456.

Gallego, G., Iyengar, G., Phillips, R., & Dubey, A. (2004). Managing flexible products on a network (CORC technical report Tr-2004-01). IEOR Department, Columbia University.

Gallego, G., & Phillips, R. (2004). Revenue management of flexible products. *Manufacturing & Service Operations Management, 6*, 321-337.

Gallego, G., & Stefanescu, C. (2009). Upgrades, upsells and pricing in revenue management (Working paper). IEOR Department, Columbia University.

Geraghty, M. K., & Johnson, E. (1997). Revenue management saves national car rental. *Interfaces, 27*(1), 107-127.

Glover, F., Glover, R., Lorenzo, J., & McMillan, C. (1982). The passenger-mix problem in the scheduled airlines. *Interfaces, 12*(3), 73-80.

Gosavi, A. (2004). A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning, 55*, 5-29.

Gosavi, A., Bandla, N., & Das, T.K. (2002). A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking. *IIE Transactions, 34*, 729-742.

Gosavi, A., Ozkaya, E., & Kahraman A. F. (2007). Simulation optimization for revenue management of airlines with cancellations and overbooking. *OR Spectrum, 29*, 21-38.

Haensel, A., Mederer, M., & Schmidt, H. (2012). Revenue management in the car rental industry: A stochastic programming approach. *Journal of Revenue and Pricing Management, 11*, 99-108.

Karaesmen, I., & van Ryzin, G. J. (2004). Overbooking with substitutable inventory classes. *Operations Research, 52*, 83-104.

Kemmer, P., Strauss, A. K., & Winter, T. (2011). Dynamic simultaneous fare proration for large-scale network revenue management. *Journal of the Operational Research Society*, doi:10.1057/jors.2011.143

Klein, R. (2007). Network capacity control using self-adjusting bid-prices. *OR Spectrum, 29*, 39-60.

Kunnumkal, S., & Topaloglu, H. (2010). A new dynamic programming decomposition method for the network revenue management problem with customer choice behavior. *Production and Operations Management, 19*, 575-590.

Lang, J. C. (2009). *Production and inventory management with substitutions.* Springer: Berlin.

Lieberman, W. H. (2007). From the back seat to the driver's seat. *Journal of Revenue and Pricing Management, 6*, 300-303.

Liu, Q., & van Ryzin, G. J. (2008). On the choice-based linear programming model for network revenue management. *Manufacturing & Service Operations Management, 10*, 288-310.

McGill, J. I., & van Ryzin, G. J. (1999). Revenue management: Research overview and prospects. *Transportation Science, 33*, 233-256.

Meissner, J., & Strauss, A. K. (2012). Network revenue management with inventory-sensitive bid prices and customer choice. *European Journal of Operational Research, 216*, 459-468.

Miranda Bront, J. J., Méndez-Díaz, I., & Vulcano, G. (2009). A column generation algorithm for choice-based network revenue management. *Operations Research, 57*, 769-784.

Petrick, A., Gönsch, J., Steinhardt, C., & Klein, R. (2010). Dynamic control mechanisms for revenue management with flexible products. *Computers & Operations Research, 37*, 2027-2039.

Petrick, A., Steinhardt, C., Gönsch, J., & Klein, R. (2012). Using flexible products to cope with demand uncertainty in revenue management. *OR Spectrum, 34*, 215-242.

Phillips, R. L. (2005). *Pricing and revenue optimization.* Stanford University Press: Stanford.

Shumsky, R. A., & Zhang, F. (2009). Dynamic capacity management with substitution. *Operations Research, 57*, 671-684.

Smith, B. C., & Penn, C. W. (1988). Analysis of alternative origin-destination control strategies. *Proceedings of the Twenty Eighth Annual AGIFORS Symposium*, New Seabury.

Subramanian, J., Stidham Jr., S., & Lautenbacher, C. J. (1999). Airline yield management with overbooking, cancellations, and no-shows. *Transportation Science, 33*, 147-167.

Talluri, K. T., & van Ryzin, G. J. (2004). *The theory and practice of revenue management.* Kluwer: Boston.

Talluri, K. T. (2009). On bounds for network revenue management (Working paper). Universitat Pompeu Fabra, Barcelona.

Van Ryzin, G., & Vulcano, G. (2008). Simulation-based optimization of virtual nesting controls for network revenue management. *Operations Research, 56*, 865-880.

Weatherford, L. R., & Bodily, S. E. (1992). A taxonomy and research overview of perishable-asset revenue management: Yield management, overbooking, and pricing. *Operations Research, 40*, 831-844.

Wollmer, R. D. (1986). A hub-and-spoke seat management model (Company report). Douglas Aircraft Company, McDonnell Douglas Corporation.

Zhang, D., & Adelman, D. (2009). An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science, 43*, 381-394.

Online Appendix

Integrated revenue management approaches for capacity control with planned upgrades

May 25, 2012

Claudius Steinhardt, Jochen Gönsch

Proofs of Propositions

In the following Appendices A.1 to A.4, we present the proofs of the Propositions 1, 2, 3, and 5 stated in Sections 3 and 4 of the paper. The proof of Proposition 4 is very similar to the proof of Proposition 3 and is omitted. Unless otherwise mentioned, we use the mathematical notation as defined in Sections 3.1 and 4.1.

A.1. Proof of Proposition 1

Proposition 1. *If* $\mathbf{A}=\mathbf{I}$ *and* $\mathcal{V}_k=\{k,\ldots,n\}$ *for all products, then*

$$\Delta_q V(\mathbf{x},t)\geq\Delta_{q'}V(\mathbf{x},t) \qquad \text{for all } t,\ \mathbf{x} \text{ with } x_q,x_{q'}>0 \text{ and } q\geq q'.$$

Proof. To simplify the presentation, without loss of generality, we assume with respect to the value function (1) that if a request is accepted, the controller chooses the upgrade possibility with the smallest index if $\underset{j\in\mathcal{V}_k}{arg\,min}\,\Delta_j V(\mathbf{x},t-1)$ is not unique; that is, if there are several resources with the same minimal opportunity cost. Furthermore, we reformulate the value function (1) by means of a policy vector $\mathbf{u}\in\{0,1,\ldots,m\}^n$ determined for each stage with the following meaning: if a request for product k arrives in the current time period, it will be denied if $u_k=0$ and accepted and upgraded to resource type u_k if $u_k\neq 0$. To guarantee that the policy $\mathbf{u}$ is feasible with the remaining capacity $\mathbf{x}$, the vector must satisfy

$$\mathbf{u}\in\mathcal{U}(\mathbf{x})=\left\{\mathbf{u}\,|\,\mathbf{u}\in\{0,\ldots,m\}^n\wedge\left((u_k=0)\vee\left(\mathbf{e}_{u_k}\leq\mathbf{x}\wedge u_k\geq k\right)\ \forall k\right)\right\}. \tag{A.1}$$

Then (1) can be rewritten as

$$V(\mathbf{x},t)=\max_{\mathbf{u}\in\mathcal{U}(\mathbf{x})}\left\{\sum_{k|u_k\neq 0}\lambda_k\cdot\left\{p_k+V\left(\mathbf{x}-\mathbf{e}_{u_k},t-1\right)\right\}+\sum_{k|u_k=0}\lambda_k\cdot V(\mathbf{x},t-1)+\left(1-\sum_k\lambda_k\right)\cdot V(\mathbf{x},t-1)\right\} \tag{A.2}$$

with the boundary condition $V(\mathbf{x},0)=0$.

From the definition of the opportunity cost, showing $\Delta_q V(\mathbf{x},t)\geq\Delta_{q'}V(\mathbf{x},t)$ is equivalent to showing $V(\mathbf{x}-\mathbf{e}_q,t)\leq V(\mathbf{x}-\mathbf{e}_{q'},t)$. We prove this inequality by induction over t. It holds for $t=0$, because $V(\mathbf{x}-\mathbf{e}_q,0)=V(\mathbf{x}-\mathbf{e}_{q'},0)=0$. Next, assume that the result holds for $t-1$. We now show that it will then hold for *t* as well. We consider the optimal policies $\mathbf{u}^{(q)*}$ and $\mathbf{u}^{(q')*}$ at stage t for the problems $V(\mathbf{x}-\mathbf{e}_q,t)$ and $V(\mathbf{x}-\mathbf{e}_{q'},t)$, respectively. In the following, we show monotonicity component-wise for each of the terms that add up to the term within the

outer brackets of (A.2). Therefore, we relate the terms product by product. In respect of each product k, the following four exhaustive cases can be distinguished:

1) $u_k^{(q)*}, u_k^{(q')*} \neq 0$:

The relevant terms that need to be related to each other are $p_k + V\left(\mathbf{x} - \mathbf{e}_q - \mathbf{e}_{u_k^{(q)*}}, t-1\right)$ and $p_k + V\left(\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q')*}}, t-1\right)$. Note that, as $\mathbf{u}^{(q)*}$ and $\mathbf{u}^{(q')*}$ are feasible policies, the capacity vectors $\mathbf{x} - \mathbf{e}_q - \mathbf{e}_{u_k^{(q)*}}$ and $\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q')*}}$ are non-negative. Now, if $u_k^{(q)*} \geq u_k^{(q')*}$, by the induction hypothesis it follows that $p_k + V\left(\mathbf{x} - \mathbf{e}_q - \mathbf{e}_{u_k^{(q)*}}, t-1\right) \leq p_k + V\left(\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q)*}}, t-1\right)$ $\leq p_k + V\left(\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q')*}}, t-1\right)$. If $u_k^{(q)*} < u_k^{(q')*}$, it follows by contradiction that $u_k^{(q)*} = q'$ as otherwise $u_k^{(q')*}$ would be equal to the smaller $u_k^{(q)*}$ according to the induction hypothesis. Furthermore, $u_k^{(q')*} \leq q$ according to the induction hypothesis and because of the availability of resource q. In total, it follows that $p_k + V\left(\mathbf{x} - \mathbf{e}_q - \mathbf{e}_{u_k^{(q)*}}, t-1\right) = p_k + V\left(\mathbf{x} - \mathbf{e}_q - \mathbf{e}_{q'}, t-1\right)$ $\leq p_k + V\left(\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q')*}}, t-1\right)$ where the inequality on the right follows from the induction hypothesis.

2) $u_k^{(q)*} = 0,\ u_k^{(q')*} > 0$:

The relevant terms are $V\left(\mathbf{x} - \mathbf{e}_q, t-1\right)$ and $p_k + V\left(\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q')*}}, t-1\right)$. We have $V\left(\mathbf{x} - \mathbf{e}_q, t-1\right) \leq V\left(\mathbf{x} - \mathbf{e}_{q'}, t-1\right) \leq p_k + V\left(\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q')*}}, t-1\right)$, where the left-hand inequality follows from the induction hypothesis and the right-hand inequality follows from the fact that the request k is accepted and upgraded to $u_k^{(q')*}$ in the optimal policy $\mathbf{u}^{(q')*}$.

3) $u_k^{(q)*} > 0,\ u_k^{(q')*} = 0$:

The relevant terms are $p_k + V\left(\mathbf{x} - \mathbf{e}_q - \mathbf{e}_{u_k^{(q)*}}, t-1\right)$ and $V\left(\mathbf{x} - \mathbf{e}_{q'}, t-1\right)$. We have $p_k + V\left(\mathbf{x} - \mathbf{e}_q - \mathbf{e}_{u_k^{(q)*}}, t-1\right) \leq p_k + V\left(\mathbf{x} - \mathbf{e}_{q'} - \mathbf{e}_{u_k^{(q)*}}, t-1\right) \leq V\left(\mathbf{x} - \mathbf{e}_{q'}, t-1\right)$, where the left-hand inequality follows from the induction hypothesis and the right-hand inequality follows from the fact that request k is not accepted in the optimal policy $\mathbf{u}^{(q')*}$.

4) $u_k^{(q)*} = u_k^{(q')*} = 0$:

The relevant terms are $V\left(\mathbf{x} - \mathbf{e}_q, t-1\right)$ and $V\left(\mathbf{x} - \mathbf{e}_{q'}, t-1\right)$ so $V\left(\mathbf{x} - \mathbf{e}_q, t-1\right) \leq V\left(\mathbf{x} - \mathbf{e}_{q'}, t-1\right)$ follows directly following from the induction hypothesis.

Additionally, the last addend within the outer brackets of (A.2) has to be considered. The argumentation is identical to 4).

Overall, monotonicity has been shown component-wise, so that in total, we obtain

$$V\left(\mathbf{x}-\mathbf{e}_q,t\right)=$$

$$\max_{\mathbf{u}\in\mathcal{U}(\mathbf{x})}\left\{\sum_{k|u_k\neq 0}\lambda_k\cdot\left\{p_k+V\left(\mathbf{x}-\mathbf{e}_q-\mathbf{e}_{u_k},t-1\right)\right\}+\sum_{k|u_k=0}\lambda_k\cdot V\left(\mathbf{x}-\mathbf{e}_q,t-1\right)+\left(1-\sum_k\lambda_k\right)\cdot V\left(\mathbf{x}-\mathbf{e}_q,t-1\right)\right\}$$

$$\leq \max_{\mathbf{u}\in\mathcal{U}(\mathbf{x})}\left\{\sum_{k|u_k\neq 0}\lambda_k\cdot\left\{p_k+V\left(\mathbf{x}-\mathbf{e}_{q'}-\mathbf{e}_{u_k},t-1\right)\right\}+\sum_{k|u_k=0}\lambda_k\cdot V\left(\mathbf{x}-\mathbf{e}_{q'},t-1\right)+\left(1-\sum_k\lambda_k\right)\cdot V\left(\mathbf{x}-\mathbf{e}_{q'},t-1\right)\right\}.$$

$$=V\left(\mathbf{x}-\mathbf{e}_{q'},t\right)$$

□

A.2. Proof of Proposition 2

Proposition 2. *If* $\mathbf{A}=\mathbf{I}$ *and* $\mathcal{V}_k=\{k,\ldots,n\}$ *for all products, then*

$$V\left(\mathbf{x},\mathbf{y},t\right)=V\left(\psi\left(\mathbf{x},\mathbf{y}\right),t\right) \qquad \text{for all } t,\left(\mathbf{x},\mathbf{y}\right)\in\mathcal{A}$$

Proof. The proof is based on alternative formulations of the Bellman equations $V\left(\mathbf{x},\mathbf{y},t\right)$ and $V\left(\mathbf{x},t\right)$, making use of a common decision vector $\mathbf{u}\in\{0,1\}^n$. If a request for product k arrives in the current time period, it will be denied if $u_k=0$ and accepted if $u_k=1$. For $V\left(\mathbf{x},\mathbf{y},t\right)$, to be feasible with remaining capacity $\mathbf{x}$ and the vector of commitments $\mathbf{y}$, $\mathbf{u}$ must satisfy

$$\mathbf{u}\in\mathcal{U}\left(\mathbf{x},\mathbf{y}\right)=\left\{\mathbf{u}\,|\,\mathbf{u}\in\{0,1\}^n\wedge\left(\mathbf{x},\mathbf{y}+u_k\cdot\mathbf{e}_k\right)\in\mathcal{A}\ \forall k\right\}. \tag{A.3}$$

Then, (2) can be rewritten as

$$V\left(\mathbf{x},\mathbf{y},t\right)=\max_{\mathbf{u}\in\mathcal{U}(\mathbf{x},\mathbf{y})}\sum_{k|u_k=1}\lambda_k\left(p_k+V\left(\mathbf{x},\mathbf{y}+\mathbf{e}_k,t-1\right)\right)+\left(1-\sum_{k|u_k=1}\lambda_k\right)\cdot V\left(\mathbf{x},\mathbf{y},t-1\right) \tag{A.4}$$

with the boundary condition $V\left(\mathbf{x},\mathbf{y},0\right)=0$.

For $V\left(\mathbf{x},t\right)$ to guarantee that the policy $\mathbf{u}$ is feasible with the remaining capacity $\mathbf{x}$, the vector must satisfy

$$\mathbf{u}\in\mathcal{U}\left(\mathbf{x}\right)=\left\{\mathbf{u}\,|\,\mathbf{u}\in\{0,1\}^n\wedge\left(\exists j\geq k\,|\,u_k\cdot\mathbf{e}_j\leq\mathbf{x}\right)\ \forall k\right\}. \tag{A.5}$$

Using (A.5) as well as Proposition 1, (1) can be rewritten as

$$V\left(\mathbf{x},t\right)=\max_{\mathbf{u}\in\mathcal{U}(\mathbf{x})}\left\{\sum_{k|u_k=1}\lambda_k\cdot\left\{p_k+V\left(\mathbf{x}-\mathbf{e}_{\min_j\{j\geq k|\mathbf{e}_j\leq\mathbf{x}\}},t-1\right)\right\}+\left(1-\sum_{k|u_k=1}\lambda_k\right)\cdot V\left(\mathbf{x},t-1\right)\right\} \tag{A.6}$$

with the boundary condition $V\left(\mathbf{x},0\right)=0$.

We first show $\mathcal{U}(\psi(\mathbf{x},\mathbf{y})) = \mathcal{U}(\mathbf{x},\mathbf{y})$. Let us consider an arbitrary vector $\mathbf{u} \in \mathcal{U}(\psi(\mathbf{x},\mathbf{y}))$. Assume that there exists a vector-component *k* with $u_k = 1$; that is, request k can be accepted in the ad-hoc DP. It then directly follows that k can also be accepted in the postponement DP, as all requests including the new one for k can be assigned to resources in exactly the same way as the ad-hoc allocation does. As this is true for any arbitrary component *k* with $u_k = 1$, $\mathbf{u}$ is also a valid decision vector for the postponement DP and therefore is included in $\mathcal{U}(\mathbf{x},\mathbf{y})$. This shows that $\mathcal{U}(\psi(\mathbf{x},\mathbf{y})) \subseteq \mathcal{U}(\mathbf{x},\mathbf{y})$.

Let us now consider an arbitrary vector $\mathbf{u} \in \mathcal{U}(\mathbf{x},\mathbf{y})$. Assume that there exists a vector-component *k* with $u_k = 1$; that is, request k can be accepted in the postponement DP. Then there are feasible allocations with remaining free capacity $\psi(\mathbf{x},\mathbf{y})$ and $\psi(\mathbf{x},\mathbf{y}+\mathbf{e}_k)$. Owing to the acceptance of k, the overall remaining capacity will decrease by one; that is $\sum_j \psi(\mathbf{c},\mathbf{y})_j = 1 + \sum_j \psi(\mathbf{c},\mathbf{y}+\mathbf{e}_k)_j$.

By contradiction, we show that $\psi(\mathbf{c},\mathbf{y})_j \geq \psi(\mathbf{c},\mathbf{y}+\mathbf{e}_k)_j \ \forall j$, that is, no resource within the feasible allocation after the acceptance of k has more remaining capacity than before. Suppose that a resource j' with $\psi(\mathbf{c},\mathbf{y})_{j'} < \psi(\mathbf{c},\mathbf{y}+\mathbf{e}_k)_{j'}$ exists. Then, clearly, at least one request k' that used capacity on j' prior to the acceptance of k must now be assigned to another resource j''. Regarding these resources j' and j'', two cases can be distinguished. If $j' > j''$, this downgrade of k', leading to an increase in $\sum_j j \cdot h_j^{(s)}$ by $j' - j''$, could also have been performed before accepting k and, thus, $\sum_j j \cdot h_j^{(s)}$ was not maximal. If $j' < j''$, similarly, just this upgrade of k' leads to a decrease in $\sum_j j \cdot h_j^{(s)}$ by $j'' - j'$, compared to if k had been accepted in the same way just without the upgrade of k'. Thus, $\sum_j j \cdot h_j^{(s)}$ is not maximal if k' is upgraded. Together, this shows that no resource's remaining capacity increases due to the acceptance of k and, as overall remaining capacity decreases by one, it follows that exactly one resource's capacity is decreased by one, while the remaining capacity of all other resources is unchanged.

In what follows, let j' denote the resource whose capacity is decreased by one in the feasible allocation. Then, again by contradiction, it follows that $j' \geq k$. Suppose $j' < k$. By definition, a request for k cannot be downgraded to j', but must use capacity on some $j'' \geq k$. Hence, there must be another request for a product $k' \leq j'$ that used capacity on $j'' \geq k$ in the previous allocation, leading to $\psi(\mathbf{x},\mathbf{y})$, which can be downgraded to j'. However, this downgrade

of k', which itself leads to an increase in $\sum_j j \cdot h_j^{(s)}$ by $j'' - j'$, could also have been performed before accepting the request for k. Thus, $\sum_j j \cdot h_j^{(s)}$ was not maximal, which contradicts the definition of $\psi(\mathbf{x},\mathbf{y})$ and therefore proves $j' \geq k$.

From the definition of $\psi(\mathbf{x},\mathbf{y})$, it follows that $j' = min\left\{ j \geq k \,|\, \psi(\mathbf{x},\mathbf{y})_j > 0 \right\}$, so, in total, we have

$$\psi(\mathbf{x},\mathbf{y}+\mathbf{e}_k) = \psi(\mathbf{x},\mathbf{y}) - \mathbf{e}_{j'} \text{ with } j' = min\left\{ j \geq k \,|\, \psi(\mathbf{x},\mathbf{y})_j > 0 \right\}. \tag{A.7}$$

According to Algorithm 1, ad-hoc DP could also accept the request *k* and could assign it to this resource $q = j'$. As this argumentation is true for any arbitrary component *k* with $u_k = 1$, $\mathbf{u}$ is also a valid decision vector for the ad-hoc DP and therefore is included in $\mathcal{U}(\psi(\mathbf{x},\mathbf{y}))$. This shows that $\mathcal{U}(\psi(\mathbf{x},\mathbf{y})) \supseteq \mathcal{U}(\mathbf{x},\mathbf{y})$, which completes the proof of

$$\mathcal{U}(\psi(\mathbf{x},\mathbf{y})) = \mathcal{U}(\mathbf{x},\mathbf{y}). \tag{A.8}$$

We now perform induction over t. From (A.4), (A.6), and (A.8) it follows that the assumption holds for $t = 0$ for all $(\mathbf{x},\mathbf{y}) \in \mathcal{A}$:

$$V(\mathbf{x},\mathbf{y},0) = \max_{\mathbf{u}\in\mathcal{U}(\mathbf{x},\mathbf{y})} \sum_{k|u_k=1} \lambda_k p_k = \max_{\mathbf{u}\in\mathcal{U}(\psi(\mathbf{x},\mathbf{y}))} \sum_{k|u_k=1} \lambda_k p_k = V(\psi(\mathbf{x},\mathbf{y}),0).$$

Next, we assume the result holds for $t-1$ and show that it holds for t. From (A.7), (A.8), and the induction hypothesis we have

$$\begin{aligned}
V(\mathbf{x},\mathbf{y},t) &= \max_{\mathbf{u}\in\mathcal{U}(\mathbf{x},\mathbf{y})} \sum_{k|u_k=1} \lambda_k \left(p_k + V(\mathbf{x},\mathbf{y}+\mathbf{e}_k,t-1) \right) + \left(1 - \sum_{k|u_k=1} \lambda_k \right) V(\mathbf{x},\mathbf{y},t-1) \\
&= \max_{\mathbf{u}\in\mathcal{U}(\psi(\mathbf{x},\mathbf{y}))} \sum_{k|u_k=1} \lambda_k \left(p_k + V(\mathbf{x},\mathbf{y}+\mathbf{e}_k,t-1) \right) + \left(1 - \sum_{k|u_k=1} \lambda_k \right) V(\mathbf{x},\mathbf{y},t-1) \\
&= \max_{\mathbf{u}\in\mathcal{U}(\psi(\mathbf{x},\mathbf{y}))} \left\{ \sum_{k|u_k=1} \lambda_k \cdot \left\{ p_k + V\left(\psi(\mathbf{x},\mathbf{y}) - \mathbf{e}_{\min_{j'}\{ j\geq k|\psi(\mathbf{x},\mathbf{y})_j>0 \}}, t-1 \right) \right\} + \left(1 - \sum_{k|u_k=1} \lambda_k \right) \cdot V(\psi(\mathbf{x},\mathbf{y}),t-1) \right\} \\
&= V(\psi(\mathbf{x},\mathbf{y}),t) \qquad \textit{for all } (\mathbf{x},\mathbf{y}) \in \mathcal{A}.
\end{aligned}$$

□

A.3. Proof of Proposition 3

Proposition 3. *For each day* η, $V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}$ *is a tighter upper bound on the optimal expected revenue* $V\left(\mathbf{X},t\right)$ *than* $V^{DLP}\left(\mathbf{X},t\right)$*:*

$$V\left(\mathbf{X},t\right)\leq V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\leq V^{DLP}\left(\mathbf{X},t\right) \qquad \text{for all } \eta,\ \mathbf{X},\ t\ .$$

Proof. The second inequality follows from the discussion in Section 4.2. The proof of the first inequality is based on ideas presented in Adelman (2007) and Zhang and Adelman (2009). Similar to the proof of Proposition 1, we make use of a decision vector $\mathbf{u}\in\{0,1,\ldots,m\}^n$ for each remaining capacity $\mathbf{X}$. If a request for product k arrives in the current time period, it is denied if $u_k=0$ and accepted and upgraded to resource type u_k if $u_k\neq 0$. To guarantee that the decision $\mathbf{u}$ is feasible with the remaining capacity $\mathbf{X}$, the vector must satisfy

$$\mathbf{u}\in\mathcal{U}\left(\mathbf{X}\right)=\left\{\mathbf{u}\,|\,\mathbf{u}\in\{0,\ldots,m\}^n\wedge\left(\left(u_k=0\right)\vee\left(\mathbf{A}^{(ku_k)}\leq\mathbf{X}\wedge u_k\geq r_k\right)\right)\ \forall k\right\}. \tag{A.9}$$

Using this set of feasible decision vectors $\mathcal{U}\left(\mathbf{X}\right)$, we reformulate the DP (4) in terms of an equivalent linear program (see, e.g., Powell, 2007, p. 63):

$$V\left(\mathbf{X},t\right)=min\ V\left(\mathbf{X},t\right) \tag{A.10}$$

subject to

$$V\left(\mathbf{X},t\right)\geq\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\left(p_k+V\left(\mathbf{X}-\mathbf{A}^{(ku_k)},t-1\right)\right)+\left(1-\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\right)V\left(\mathbf{X},t-1\right)$$
$$\text{for all } t,\mathbf{X},\mathbf{u}\in\mathcal{U}\left(\mathbf{X}\right) \tag{A.11}$$

with decision variables $V\left(\mathbf{X},t\right)\geq 0\ \forall\mathbf{X},t$.

We now substitute $V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}$ for $V\left(\mathbf{X},t\right)$ into (A.10)-(A.11) and obtain

$$V\left(\mathbf{X},t\right)=min\ V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau} \tag{A.12}$$

subject to

$$V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\geq\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\left(p_k+V_\eta\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(ku_k)},t-1\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}\left(x_{r\tau}-a_{r\tau}^{(ku_k)}\right)\right)$$
$$+\left(1-\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\right)\left(V_\eta\left(\mathbf{x}_\eta,t-1\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\right)$$
$$\text{for all } t,\mathbf{X},\mathbf{u}\in\mathcal{U}\left(\mathbf{X}\right) \tag{A.13}$$

with decision variables $V_\eta\left(\mathbf{x}_\eta,t\right)\geq 0\ \forall\mathbf{x}_\eta,t$.

Based on these linear reformulations, we now prove Proposition 3 by induction over t for an arbitrary day η as follows. For $t=1$, we note from (A.13) that

$$V_\eta\left(\mathbf{x}_\eta,1\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\geq\sum_{k|u_k\neq 0}\lambda_k\left(1\right)\left(p_k+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}\left(x_{r\tau}-a_{r\tau}^{(ku_k)}\right)\right)$$
$$+\left(1-\sum_{k|u_k\neq 0}\lambda_k\left(1\right)\right)\left(\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\right) \qquad \text{for all } \mathbf{X},\mathbf{u}\in\mathcal{U}\left(\mathbf{X}\right).$$

Since this holds for all $\mathbf{u}\in\mathcal{U}\left(\mathbf{X}\right)$, we have

$$V_\eta\left(\mathbf{x}_\eta,1\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\geq\max_{\mathbf{u}\in\mathcal{U}(\mathbf{X})}\sum_{k|u_k\neq 0}\lambda_k\left(1\right)p_k=V\left(\mathbf{X},1\right) \qquad \text{for all } \mathbf{X},\mathbf{u}\in\mathcal{U}\left(\mathbf{X}\right),$$

where the right-hand equality follows from (A.10) and (A.11). This shows that the result holds for $t=1$. Next, we assume it holds for $t-1$ and show that it then holds for t. From (A.13) and the induction hypothesis, we have

$$V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\geq\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\left(p_k+V\left(\mathbf{X}-\left(\mathbf{a}_\eta^{(ku_k)}\right),t-1\right)\right)$$
$$+\left(1-\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\right)\left(V\left(\mathbf{X},t-1\right)\right) \qquad \text{for all } \mathbf{X},\mathbf{u}\in\mathcal{U}\left(\mathbf{X}\right).$$

From the reformulation given by (A.10) and (A.11) for period t, we obtain

$$V_\eta\left(\mathbf{x}_\eta,t\right)+\sum_r\sum_{\tau\neq\eta}\pi_{r\tau}x_{r\tau}\geq V\left(\mathbf{X},t\right) \qquad \text{for all } \mathbf{X}.$$

□

A.4. Proof of Proposition 5

Proposition 5. $V_\eta\left(\mathbf{x}_\eta,t\right)-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}\leq V_{i\eta}\left(x_{i\eta},t\right)$ *for all* η, i, $\mathbf{X}$, t.

Proof. We restate $V_\eta\left(\mathbf{x}_\eta,t\right)$ and $V_{i\eta}\left(x_{i\eta},t\right)$ using a decision vector $\mathbf{u}$. We have

$$V_\eta\left(\mathbf{x}_\eta,t\right)=\max_{\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)}\left\{\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\left(p_k-\sum_{\tau\neq\eta}a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}+V_\eta\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(ku_k)},t-1\right)\right)\right.$$
$$\left.+\left(1-\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\right)V_\eta\left(\mathbf{x}_\eta,t-1\right)\right\}$$

with $\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)=\left\{\mathbf{u}\,|\,\mathbf{u}\in\{0,\ldots,m\}^n\wedge\left(\left(u_k=0\right)\vee\left(\mathbf{a}_\eta^{(ku_k)}\le\mathbf{x}_\eta\wedge u_k\ge r_k\right)\right)\forall k\right\}$ and

$$V_{i\eta}\left(x_{i\eta},t\right)=\max_{\mathbf{u}\in\mathcal{U}\left(x_{i\eta}\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(t\right)\left(p_k-\sum_\tau a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}+a_{i\eta}^{(ku_k)}\pi_{i\eta}+V_{i\eta}\left(x_{i\eta}-a_{i\eta}^{(ku_k)},t-1\right)\right)\right.$$
$$\left.+\left(1-\sum_{k|u_k\neq0}\lambda_k\left(t\right)\right)V_{i\eta}\left(x_{i\eta},t-1\right)\right\}$$

with $\mathbf{u}\in\mathcal{U}\left(x_{i\eta}\right)=\left\{\mathbf{u}\,|\,\mathbf{u}\in\{0,\ldots,m\}^n\wedge\left(\left(u_k=0\right)\vee\left(a_{i\eta}^{(ku_k)}\le x_{i\eta}\wedge u_k\ge r_k\right)\right)\forall k\right\}$.

Obviously, $\mathcal{U}\left(\mathbf{x}_\eta\right)\subseteq\mathcal{U}\left(x_{i\eta}\right)$ for all $\mathbf{x}_\eta$. Through induction over t, the proof is now as follows. For $t=1$, we have

$$V_\eta\left(\mathbf{x}_\eta,1\right)-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}=\max_{\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(1\right)\left(p_k-\sum_{\tau\neq\eta}a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}\right)\right\}-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}$$
$$\le\max_{\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(1\right)\left(p_k-\sum_{\tau\neq\eta}a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}\right)\right\}$$
$$\le\max_{\mathbf{u}\in\mathcal{U}\left(x_{i\eta}\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(1\right)\left(p_k-\sum_\tau a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}+a_{i\eta}^{(ku_k)}\pi_{i\eta}\right)\right\}=V_{\eta i}\left(x_{\eta i},1\right)\quad\text{for all }\eta,\ i,\mathbf{X}.$$

This shows that the result holds for $t=1$. Next, we assume it holds for $t-1$ and show that it then holds for t.

$$V_\eta\left(\mathbf{x}_\eta,t\right)-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}$$
$$=\max_{\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(t\right)\left(p_k-\sum_{\tau\neq\eta}a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}+V_\eta\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(ku_k)},t-1\right)\right)+\left(1-\sum_{k|u_k\neq0}\lambda_k\left(t\right)\right)V_\eta\left(\mathbf{x}_\eta,t-1\right)\right\}-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}$$
$$=\max_{\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(t\right)\left(p_k-\sum_{\tau\neq\eta}a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}+V_\eta\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(ku_k)},t-1\right)\right)\right.$$
$$\left.+\left(1-\sum_{k|u_k\neq0}\lambda_k\left(t\right)\right)\left(V_\eta\left(\mathbf{x}_\eta,t-1\right)-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}\right)\right\}$$
$$=\max_{\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(t\right)\left(p_k-\sum_\tau a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}+a_{u_k\eta}^{(ku_k)}\pi_{u_k\eta}-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}+V_\eta\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(ku_k)},t-1\right)\right)\right.$$
$$\left.+\left(1-\sum_{k|u_k\neq0}\lambda_k\left(t\right)\right)\left(V_\eta\left(\mathbf{x}_\eta,t-1\right)-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}\right)\right\}$$
$$=\max_{\mathbf{u}\in\mathcal{U}\left(\mathbf{x}_\eta\right)}\left\{\sum_{k|u_k\neq0}\lambda_k\left(t\right)\left(p_k-\sum_\tau a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}+a_{i\eta}^{(ku_k)}\pi_{u_k\eta}+V_\eta\left(\mathbf{x}_\eta-\mathbf{a}_\eta^{(ku_k)},t-1\right)-\sum_{r\neq i}\pi_{r\eta}\left(x_{r\eta}-a_{r\eta}^{(ku_k)}\right)\right)\right.$$
$$\left.+\left(1-\sum_{k|u_k\neq0}\lambda_k\left(t\right)\right)\left(V_\eta\left(\mathbf{x}_\eta,t-1\right)-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}\right)\right\}$$

Using the induction hypothesis, we get

$$V_\eta\left(\mathbf{x}_\eta,t\right)-\sum_{r\neq i}\pi_{r\eta}x_{r\eta}\leq \max_{\mathbf{u}\in\mathcal{U}\left(x_{i\eta}\right)}\left\{\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\left(p_k-\sum_{\tau}a_{u_k\tau}^{(ku_k)}\pi_{u_k\tau}+a_{i\eta}^{(ku_k)}\pi_{i\eta}+V_{i\eta}\left(x_{i\eta}-a_{i\eta}^{(ku_k)},t-1\right)\right)\right.$$
$$\left.+\left(1-\sum_{k|u_k\neq 0}\lambda_k\left(t\right)\right)V_\eta\left(x_{i\eta},t-1\right)\right\}=V_{i\eta}\left(x_{i\eta},t\right)$$

□

Supplement to Computational Results

	FCFS			DP			DPD-S			SUCC		
α	% LF	% AR	% Upg	% LF	% AR	% Upg	% LF	% AR	% Upg	% LF	% AR	% Upg
0.5	76.09	100.00	8.32	76.09	100.00	8.32	76.09	100.00	8.32	74.54	97.72	6.52
0.6	91.16	99.41	16.93	91.03	99.27	16.58	91.16	99.41	16.87	88.22	95.88	13.97
0.7	97.02	91.02	19.39	95.91	89.95	15.21	96.97	90.97	17.48	95.15	89.01	15.06
0.8	98.44	78.09	19.90	97.91	77.66	12.22	96.75	76.72	13.48	96.94	76.69	9.98
0.9	98.85	72.31	20.30	98.29	71.89	14.85	98.13	71.77	14.63	97.86	71.44	9.26
1.0	99.47	65.42	20.40	98.74	64.94	16.25	98.52	64.79	17.93	98.92	65.03	9.40
1.1	99.84	59.60	20.60	99.06	59.13	13.94	99.22	59.23	16.35	99.33	59.26	7.17
1.2	99.94	54.78	20.80	99.24	54.39	11.21	99.37	54.47	14.86	99.55	54.54	5.63
1.3	100.00	49.46	20.97	99.36	49.15	8.28	99.51	49.22	13.17	99.32	49.08	6.58
1.4	100.00	46.95	20.99	99.55	46.74	6.63	99.42	46.67	7.84	99.35	46.64	1.58
1.5	100.00	43.73	20.61	99.64	43.57	4.47	99.70	43.59	6.68	99.43	43.44	1.34

Table A.1: Load factor, accepted requests, and upgrades (Example 1)

	%DPD-S		%DP	
α	AVG	99% Conf. Int.	AVG	99% Conf. Int.
0.5	1.61	(1.19; 2.02)	1.61	(1.19; 2.02)
0.6	2.74	(2.33; 3.16)	2.95	(2.55; 3.34)
0.7	1.94	(1.54; 2.33)	3.92	(3.47; 4.36)
0.8	2.38	(1.96; 2.79)	3.09	(2.69; 3.49)
0.9	3.23	(2.84; 3.62)	3.73	(3.31; 4.16)
1.0	5.39	(4.93; 5.85)	6.09	(5.62; 6.56)
1.1	6.20	(5.71; 6.69)	7.78	(7.21; 8.34)
1.2	5.36	(4.81; 5.92)	8.36	(7.73; 9.00)
1.3	3.70	(3.27; 4.13)	8.36	(7.76; 8.96)
1.4	5.93	(5.39; 6.47)	6.69	(6.14; 7.25)
1.5	4.92	(4.45; 5.39)	6.44	(5.96; 6.92)

Table A.2: Revenue improvements over SUCC (Example 1)

	FCFS			DPD-D			DPD-S			SUCC		
α	% LF	% AR	% Upg	% LF	% AR	% Upg	% LF	% AR	% Upg	% LF	% AR	% Upg
0.5	80.83	99.67	12.97	80.72	99.50	12.76	80.85	99.66	12.95	74.81	91.94	6.33
0.6	90.80	93.32	17.52	89.86	91.86	15.18	89.12	92.39	18.40	83.54	86.36	8.24
0.7	94.35	82.65	18.77	93.97	81.06	14.45	93.33	82.71	17.40	90.67	79.43	9.40
0.8	96.59	74.84	19.37	95.17	73.33	12.46	95.53	73.16	15.06	93.50	71.62	8.00
0.9	97.85	67.38	19.97	95.99	66.87	10.46	95.62	67.10	11.89	94.85	66.40	6.27
1.0	98.63	61.67	19.95	96.73	60.85	8.16	96.73	61.20	10.50	95.59	61.07	5.17
1.1	99.07	56.61	20.66	97.12	55.42	7.05	96.40	56.36	9.27	95.71	57.02	3.50
1.2	99.35	52.08	20.69	97.60	51.23	6.41	96.87	52.40	8.41	97.04	53.07	3.10
1.3	99.53	48.34	21.15	97.71	47.49	5.89	97.20	47.57	8.36	97.37	48.56	1.72
1.4	99.52	44.96	21.33	97.52	44.45	5.94	97.02	44.43	7.90	97.57	45.45	1.74
1.5	99.65	41.92	21.41	97.92	42.13	6.07	97.61	42.09	7.79	98.00	42.77	1.55

Table A.3: Load factor, accepted requests, and upgrades (Example 2)

α	% DPD-D Bound	% DPD-S Bound
0.5	98.98	99.27
0.6	98.33	99.22
0.7	99.12	100.13
0.8	99.41	100.52
0.9	98.66	99.84
1.0	99.10	100.03
1.1	98.95	99.88
1.2	99.17	99.93
1.3	99.32	100.37
1.4	99.55	100.40
1.5	99.28	99.94

Table A.4: Theoretical upper bounds (Example 2)

α	AVG	99% Conf. Int.	95% Conf. Int.	90% Conf. Int.
0.5	0.07	(-0.05; 0.20)	(-0.02; 0.17)	(0.00; 0.15)
0.6	0.92	(0.61; 1.24)	(0.68; 1.16)	(0.72; 1.12)
0.7	2.26	(1.82; 2.69)	(1.93; 2.59)	(1.98; 2.54)
0.8	0.91	(0.60; 1.22)	(0.68; 1.15)	(0.71; 1.11)
0.9	0.65	(0.34; 0.97)	(0.42; 0.89)	(0.45; 0.85)
1.0	1.60	(1.26; 1.95)	(1.34; 1.87)	(1.39; 1.82)
1.1	2.34	(1.86; 2.83)	(1.97; 2.71)	(2.03; 2.65)
1.2	2.71	(2.24; 3.18)	(2.35; 3.06)	(2.41; 3.01)
1.3	0.31	(0.09; 0.53)	(0.14; 0.48)	(0.17; 0.45)
1.4	0.34	(0.13; 0.54)	(0.18; 0.49)	(0.21; 0.47)
1.5	0.33	(0.13; 0.53)	(0.18; 0.48)	(0.21; 0.46)

Table A.5: Revenue improvement of DPD-D over DPD-S (Example 2)

	%DPD-S		%DPD-D	
α	AVG	99% Conf. Int.	AVG	99% Conf. Int.
0.5	5.40	(4.82; 5.99)	5.48	(4.92; 6.04)
0.6	5.47	(4.92; 6.02)	6.44	(5.93; 6.96)
0.7	2.51	(2.00; 3.01)	4.82	(4.30; 5.35)
0.8	3.95	(3.45; 4.45)	4.90	(4.45; 5.35)
0.9	3.78	(3.26; 4.29)	4.45	(4.04; 4.87)
1.0	3.38	(2.92; 3.83)	5.04	(4.59; 5.48)
1.1	2.75	(2.10; 3.40)	5.16	(4.61; 5.70)
1.2	2.05	(1.46; 2.65)	4.82	(4.32; 5.32)
1.3	3.96	(3.47; 4.45)	4.28	(3.81; 4.76)
1.4	4.56	(4.05; 5.07)	4.91	(4.40; 5.43)
1.5	4.77	(4.25; 5.30)	5.12	(4.61; 5.64)

Table A.6: Revenue improvement over SUCC (Example 2)

	FCFS			DPD-D			DPD-S			SUCC		
#sdays	% LF	% AR	% Upg	% LF	% AR	% Upg	% LF	% AR	% Upg	% LF	% AR	% Upg
1	97.30	70.40	27.35	96.18	69.63	23.70	95.34	69.03	25.58	94.92	67.57	15.95
2	97.40	71.27	26.92	96.11	70.79	15.99	95.27	70.79	17.54	94.36	69.20	15.86
3	96.83	71.20	26.98	95.57	71.90	16.03	95.71	73.94	19.25	93.57	71.05	16.51
4	96.81	71.07	27.22	-	-	-	95.83	73.43	19.79	93.17	71.10	15.92
5	96.54	71.04	27.40	-	-	-	95.48	73.99	19.51	93.32	71.24	17.04
6	96.53	71.20	27.60	-	-	-	95.60	74.07	19.89	93.22	71.60	16.69
7	96.66	71.03	27.44	-	-	-	95.46	74.06	19.32	93.64	71.38	17.25
8	96.53	70.93	27.49	-	-	-	95.46	74.01	19.43	93.42	71.63	16.89
9	96.58	71.10	27.55	-	-	-	95.45	74.39	19.23	93.66	71.82	17.50
10	96.50	71.07	27.71	-	-	-	95.40	74.25	19.38	93.30	71.82	17.07
11	96.41	70.98	27.82	-	-	-	95.23	74.37	19.15	93.42	71.85	17.60
12	96.44	71.14	27.97	-	-	-	95.41	74.39	19.27	93.47	72.03	17.42
13	96.23	71.63	28.23	-	-	-	95.11	75.02	19.41	93.26	72.46	17.95
14	94.77	72.85	28.10	-	-	-	93.92	75.99	20.51	92.04	73.52	18.39

Table A.7: Load factor, accepted requests, and upgrades (Example 3)

References

Adelman, D. (2007). Dynamic bid-prices in revenue management. *Operations Research, 55*, 647-661.

Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality.* Wiley: New Jersey.

Zhang, D., & Adelman, D. (2009). An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science, 43*, 381-394.