Dynamic Control Mechanisms for Revenue Management with Flexible Products

24.09.2009

Anita Petrick, Jochen Gönsch, Claudius Steinhardt, Robert Klein

Anita Petrick

Chair of Operations Research, Department of Business Administration, Technische Universität Darmstadt, Hochschulstr. 1, 64289 Darmstadt

e-mail: petrick@bwl.tu-darmstadt.de

Jochen Gönsch

Chair of Mathematical Methods, Department of Statistics and Economic Theory, Universität Augsburg, Universitätsstraße 16, 86159 Augsburg

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

Claudius Steinhardt

Chair of Mathematical Methods, Department of Statistics and Economic Theory, Universität Augsburg, Universitätsstraße 16, 86159 Augsburg

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

Robert Klein (✉)

Chair of Mathematical Methods, Department of Statistics and Economic Theory, Universität Augsburg, Universitätsstraße 16, 86159 Augsburg

Phone: +49 821 598 4150

Fax: +49 821 598 4226

e-mail: robert.klein@wiwi.uni-augsburg.de

Abstract

Revenue management with flexible products has experienced a growing interest in the academic literature within the last few years. Flexible products allow supply-side substitution between resources and can therefore help to maximize overall revenue as well as capacity utilization in markets with highly uncertain demand. This paper addresses the question of how the mathematical models which have been developed for capacity control with flexible products should be used over time to exploit the substitution opportunities, while keeping practical applicability in mind. Several dynamic control mechanisms are proposed, each of which makes use of the flexibility to a different extent. A comprehensive computational study shows the potential of the different approaches by revealing their strengths and weaknesses.

Keywords: Flexible Products, Revenue Management, Capacity Control

1 Introduction

During the last three decades, revenue management has evolved into one of the most important fields of application of Operations Research methods. Basically, it is concerned with the task of optimally selling a fixed capacity of perishable resources within a given selling horizon, thereby maximizing the overall revenue or contribution margin. This is essentially achieved by the application of two instruments (see, e.g., [1]): In a first step, on a rather tactical planning level, *price differentiation* is performed, leading to a variety of differently priced products defined on the same set of resources. In a second step, on the operational level, the availability of the products is permanently adjusted by means of *capacity control*, according to the current forecast regarding future demand within the selling horizon.

Thus far, many mathematical models have been proposed to solve the capacity control problem. The most accurate approach is the formulation of a stochastic dynamic program (see, e.g., [2, 3]), which explicitly takes demand's stochastic nature and temporal distribution into consideration. However, as this approach is computationally intractable even for small resource networks, some kind of static approximation is typically used. In practical applications, one common approximation is the use of a deterministic linear program (DLP), which calculates an optimal allocation of the available resources' capacity to the products given the expected values of future demand (see, e.g., [4–6]). Within the selling period, the output of the DLP is then either used directly to define *booking limits* by specifying the maximum amount of units that should be offered for each product, or is used indirectly via its dual program as *bid prices*, which denote the minimum amount of money to be gained for each resource. Owing to demand's stochastic nature, the DLP has to be periodically reoptimized within the selling horizon to adapt to the current demand realization and forecast. For the sake of completeness, it should be mentioned that many other static approximations have been discussed in the literature. One group of papers aims at better incorporating the parameters of the demand distribution (see, e.g., [7, 8]), whereas others try to minimize the need for reoptimization by analytically calculating a whole set of time and/or resource-dependent bid prices in advance (see, e.g., [9, 10]). Furthermore, approaches using simulation-based optimization have been proposed to calculate booking limits (see, e.g., [11, 12]) and bid prices (see, e.g., [13, 14]).

A relatively new aspect of revenue management research is the incorporation of *flexible products* into capacity control (see [15, 16]). Following the initial definition of Gallego and Phillips [16], a flexible product is a menu of several alternatives offered by a supplier, who reserves the right to assign customers to one of the alternatives at a predefined point in time after purchase. The supply-side flexibility arising from this delay in resource allocation can help to mitigate the negative impact of forecast errors which often occur in an early phase of the selling horizon (see [17]). From an application point of view, revenue management with flexible products has been discussed in the context of air cargo (see, e.g., [18, 19]) or make-to-order environments (see, e.g., [20, 21]), where the flexibility arises from given time windows that allow the supplier to autonomously arrange accepted requests with respect to time and resources. Other service industry applications explicitly confront the customer with the flexibility. Examples include applications in passenger air transport, the hotel industry, cruise lines, or the scheduling of commercials in the broadcasting industry, where customers have to be notified of their specific itinerary, room, cabin, or commercial break, respectively (see, e.g., [22, 23]).

In the literature mentioned above, several traditional DLP-based model formulations have been extended to problem-specific flexible product settings. However, from a more general perspective, it has not yet been intensively discussed how mathematical models for revenue management with flexible products should be used over time within a capacity control mechanism, thereby exploiting the supply-side substitution opportunities after purchase and, simultaneously, guaranteeing the provision of all sold flexible products. In this paper, we concentrate on this issue and propose several dynamic control mechanisms that differ in the extent to which they exploit the flexibility and in the effort related to their practical application.

The paper is organized as follows: In Section 2, we formulate a basic static model for capacity control incorporating flexible products, which is an extension of the traditional DLP. On this basis, we develop several dynamic control mechanisms and judge them according to their flexibility utilization as well as their practical applicability (Section 3). In Section 4, we perform a computational study that compares the different control mechanisms and their impact on the overall revenue in realistic revenue management scenarios. In Section 5, the results are summarized and overall conclusions are drawn.

2 Basic model formulation

In the following, we briefly describe the basic deterministic linear programming model for revenue management with flexible products (see, e.g., [15, 23] for similar model formulations). For this purpose, let $\mathcal{H} = \{1,\ldots,l\}$ be a firm's resource network at a point in time t, consisting of l resources with available capacity $\mathbf{c} = (c_1,\ldots,c_l)$. We consider a set of *specific products* $\mathcal{I} = \{1,\ldots,n^s\}$ defined on $\mathcal{H}$. The capacity consumption of a single unit of product $i \in \mathcal{I}$ with respect to a resource $h \in \mathcal{H}$ is expressed by a_{hi}. For the sake of simplicity, we assume that $a_{hi} \in \{0,1\}$ for all $h \in \mathcal{H}$, $i \in \mathcal{I}$. Furthermore, we consider a set of *flexible products* $\mathcal{J} = \{1,\ldots,n^f\}$. For each $j \in \mathcal{J}$, there is a set $\mathcal{M}_j \subseteq \mathcal{I}$ describing its possible execution modes, which we assume to be a subset of the existing specific products without loss of generality. r_i and f_j denote the revenues obtained when selling one unit of product i and j, respectively. The expected aggregated demands-to-come at the point in time t are expressed by $\bar{D}^s_{it}$ and $\bar{D}^f_{jt}$.

We use decision variables x_i, denoting the number of expected future requests to be accepted for each product i. As $a_{hi} \in \{0,1\}$, these contingents correspond directly to allocations of capacity with regard to the resources required by i. Furthermore, the decision variables y_{jm} denote allocations to the expected requests for flexible products for each of the possible execution modes $m \in \mathcal{M}_j$. The optimization problem can then be stated as follows (**DLP-flex**):

$$V_t(\mathbf{c}) = max \sum_{i \in \mathcal{I}} r_i \cdot x_i + \sum_{j \in \mathcal{J}} f_j \cdot \sum_{m \in \mathcal{M}_j} y_{jm} \tag{1}$$

subject to

$$\sum_{i \in \mathcal{I}} a_{hi} \cdot x_i + \sum_{j \in \mathcal{J}} \sum_{m \in \mathcal{M}_j} a_{hm} \cdot y_{jm} \leq c_h \qquad \text{for all } h \in \mathcal{H} \tag{2}$$

$$\sum_{m \in \mathcal{M}_j} y_{jm} \leq \bar{D}^f_{jt} \qquad \text{for all } j \in \mathcal{J} \tag{3}$$

$$x_i \leq \bar{D}^s_{it} \qquad \text{for all } i \in \mathcal{I} \tag{4}$$

$$y_{jm} \geq 0 \qquad \text{for all } j \in \mathcal{J},\ m \in \mathcal{M}_j \tag{5}$$

$$x_i \geq 0 \qquad \text{for all } i \in \mathcal{I} \tag{6}$$

The objective function (1) maximizes the total revenue-to-go. Constraints (2) guarantee that the remaining capacity $\mathbf{c}$ is sufficient for the allocations x_i and y_{jm}. Furthermore, x_i and y_{jm} should not exceed the expected demands (constraints (3); (4)) and all allocations must be nonnegative (constraints (5); (6)). Like in the standard DLP for revenue management (see, e.g., [24]), integrality constraints for the decision variables are omitted; consequently, the resulting linear program can be efficiently solved and allows information contained in the optimal dual solution to be used in order to obtain an operational control policy.

In practical applications, the derivation of *bid prices* is a widely used approach to obtain such a control policy. For each resource $h \in \mathcal{H}$, the related bid price π_h represents a threshold for the amount of money to be gained when giving away one unit of h. In DLP-flex, bid prices can be derived by using the shadow prices of the corresponding capacity constraints (2) obtained from the optimal dual solution. Given that there is enough capacity, an incoming request for a specific product $i \in \mathcal{I}$ should be accepted if (and only if) the following condition holds:

$$r_i \geq \sum_{h \in \mathcal{H}} a_{hi} \cdot \pi_h \tag{7}$$

This result is well-known for the traditional DLP formulation. The term on the right-hand side can be regarded as an approximation of the opportunity cost resulting from the acceptance of a request for product i. A similar result holds for the extended setting with flexible products (see [25]), namely that an incoming request for a flexible product $j \in \mathcal{J}$ should be accepted if (and only if) there is at least one available execution mode $m \in \mathcal{M}_j$ with

$$f_j \geq \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h \tag{8}$$

The important questions arising in this context are: How should the availability checks be performed and how does the acceptance of a request affect the remaining capacity for future requests? If a request for a specific product has been accepted, it is obvious that each of the required resources' capacity should immediately be reduced by one. With respect to flexible products' requests, however, this is not straightforward because an immediate allocation and reduction of a specific execution mode's capacity would limit the flexibility within the remaining selling horizon. Therefore, it may be quite difficult to check whether there is at least one available execution mode. We examine these issues in the next section.

3 Dynamic control mechanisms

The complexity related to the incorporation of flexible products arises from the fact that all accepted requests for flexible products must ultimately be served. In other words, the feasibility of the underlying allocation problem – considering the remaining resources' capacity – has to be guaranteed at all times during the control process. In this section, we present several dynamic control mechanisms that serve this purpose and can be classified according to the following two dimensions (see Fig. 1):

- In one way or another, all of the mechanisms (temporarily) allocate resources in order to ensure feasibility. However, they differ in the extent to which they allow for the *reallocation* of capacity with regard to accepted requests for flexible products within the selling horizon. In this context, we distinguish between mechanisms with *time-based* reallocation, meaning that a rearrangement of requests for flexible products is only permitted at certain predefined points in time, and *request-based* reallocation, which allows rearrangements with each incoming request.
- The mechanisms also vary in terms of the frequency with which a *reoptimization* of the underlying static optimization model is performed in order to obtain updated bid prices. As before, a differentiation can be made between *time-based* reoptimization, with the control parameters, especially bid prices, being only updated in predefined intervals, and *request-based* reoptimization, which performs reoptimizations with each incoming request.

The degree of flexibility, which denotes the extent to which the mechanisms exploit the substitution opportunities inherent in flexible products, increases with the reallocation frequency and that of the reoptimization. We will, however, see that, simultaneously, the control effort also increases, which can be crucial for practical applications.

In the following subsections, we describe the five dynamic control mechanisms from Fig. 1 in detail. To support understanding, a consistent example is used throughout the paper to illustrate how each mechanism processes the same given set of ten requests. Additionally, Table 1 at the end of this section summarizes all mechanisms' acceptance/rejection decisions. Moreover, a semi-formal procedure using both math notation and verbal descriptions is formulated for each mechanism.

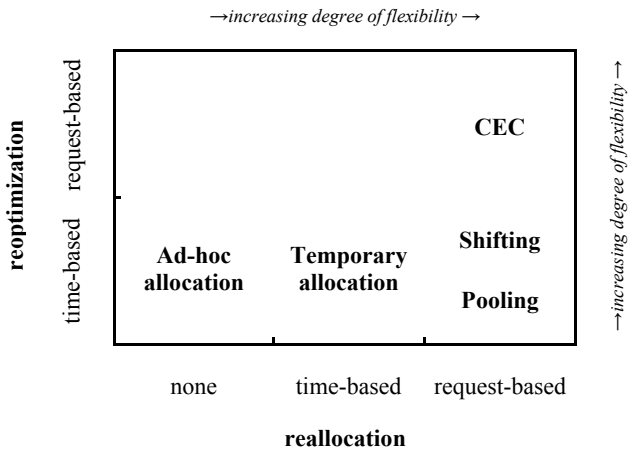


Fig. 1. Overview of dynamic control mechanisms

3.1 Ad-hoc allocation

The simplest way to guarantee feasibility is the ad-hoc allocation of resources to incoming requests for flexible products, meaning that the final execution mode is irrevocably determined immediately after acceptance. The aim of this approach is to treat flexible requests exactly like specific requests once the execution mode is fixed, which greatly simplifies the control process. Hence, DLP-flex can be directly applied in the following way: If the bid price condition (8) is fulfilled for at least one execution mode m with $a_{hm} \le c_h$ for all $h \in \mathcal{H}$, the request is accepted. Intuitively, it is assigned to the mode that causes the least opportunity cost; that is, to the mode $m' \in \mathcal{M}_j$ with $m' = \underset{m \in \mathcal{M}_j}{arg\,min} \left\{ \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h \mid a_{hm} \le c_h \ \forall h \in \mathcal{H} \right\}$. Since this assignment is final, the capacity of the needed resources can now simply be reduced as for specific products ($c_h := c_h - a_{hm'}$), so that the availability check for later incoming specific or flexible product requests can be uniformly undertaken against the remaining capacity.

Example 1. *We consider a small resource network consisting of three resources A, B, and C with capacity $c_A = c_B = 2$ and $c_C = 1$. Four products are defined using these resources. Three specific products P_A, P_B, and P_C need one unit of the resources A, B, and C, respectively, and yield revenues of $r_A = \$100$, $r_B = \$120$, and $r_C = \$110$. Additionally, a flexible product P_{flex} with three execution modes using either resource A, B, or C, respectively, is offered at $f_{flex} = \$40$. We consider the following ten requests arriving successively: P_{flex}, P_{flex}, P_A, P_A, P_A, P_{flex}, P_{flex}, P_B, P_{flex}, P_C.*

To process these requests with the ad-hoc mechanism, DLP-flex is solved first, using a demand forecast because exact demand is, of course, not known in advance. We assume that the

bid prices $\pi_A = \pi_B = \pi_C = \$40$ *are obtained. As \$40 is below or equal to every product's price, all requests are accepted as long as the capacity is sufficient. Thus, the first two requests for* P_{flex} *are accepted and, one after another, randomly assigned to the first execution mode, reducing the capacity of resource A to* $c_A = 0$ *(see also Table 1). As this assignment is irrevocable, the following two requests for* P_A *have to be declined due to the lack of capacity on A, even though the bid price* π_A *is well below the revenue* r_A*. A reoptimization scheduled after the fourth request yields the new bid prices* $\pi_A = \$100$*,* $\pi_B = \$40$*, and* $\pi_C = \$110$*, using an updated demand forecast predicting strong demand for* A *and* C *and no demand for* B*. The following request for* P_A *is declined again, but the two subsequent flexible ones are accepted and assigned to* B *because the bid price* π_B *equals* f_{flex} *and there is enough free capacity (*$c_B = 2$*). With respect to the last three requests, only* P_C *is accepted, leading to a total revenue of \$270.*

The complete control mechanism can be summarized as follows:

Procedure 1. Ad-hoc allocation

Precondition: an incoming request for $i \in \mathcal{I}$ or $j \in \mathcal{J}$, bid prices $\pi_h \ \forall h \in \mathcal{H}$, remaining capacity $c_h \ \forall h \in \mathcal{H}$, expected demands-to-come $\bar{D}^s_{nt} \ \forall n \in \mathcal{I}$ and $\bar{D}^f_{kt} \ \forall k \in \mathcal{J}$.

(1) if a *reoptimization* of the underlying model is scheduled:

(1a) solve DLP-flex with $c_h \ \forall h \in \mathcal{H}$, $\bar{D}^s_{nt} \ \forall n \in \mathcal{I}$ and $\bar{D}^f_{kt} \ \forall k \in \mathcal{J}$ as input parameters;

(1b) update the bid prices $\pi_h \ \forall h \in \mathcal{H}$ according to the model's output;

(2) if the incoming request is for *specific product* $i \in \mathcal{I}$:

(2a) if $r_i \geq \sum_{h \in \mathcal{H}} a_{hi} \cdot \pi_h$ and $a_{hi} \leq c_h \ \forall h \in \mathcal{H}$:

accept request;

set $c_h := c_h - a_{hi} \ \forall h \in \mathcal{H}$;

(2b) else: reject request;

(3) if the incoming request is for *flexible product* $j \in \mathcal{J}$:

(3a) if $\exists m \in \mathcal{M}_j$ with $f_j \geq \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h$ and $a_{hm} \leq c_h \ \forall h \in \mathcal{H}$:

accept request;

set $c_h := c_h - a_{hm'} \ \forall h \in \mathcal{H}$ with

$$m' = \underset{m \in \mathcal{M}_j}{arg\,min} \left\{ \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h \;\middle|\; a_{hm} \leq c_h \ \forall h \in \mathcal{H} \right\};$$

(3b) else: reject request;

Note that we define the $arg\,min$-operator to randomly return one of the minimizing arguments if there are several. Ad-hoc allocations of this type have gained some popularity due to the simplicity of their practical implementation. The resulting products are called *opaque products*, which guarantee one of several fully specified products but hide the identity of the product that the consumer will actually obtain until after the purchase is completed (see, e.g., [16], [26]). Opaque products are used, for example, in the hotel and tourism industry as well as in passenger air transport. In the latter, they are offered either directly (e.g., "blind booking" by Lufthansa's low cost subsidiary Germanwings) or by intermediaries (like Priceline, Hotwire, and Travelocity) who combine seats on similar flights by different airlines as alternative execution modes of a single product. Immediately after the purchase of this opaque product, the execution mode is fixed and the customer is informed about the airline and flight details like the exact itinerary or arrival and departure times. Hence, the opaque product effectively becomes a specific one after the sale; the relevant airline is informed about the sale as if it were a specific product, without having to consider that the booking originally stemmed from an opaque product. Thus, such products are often not intended to improve any kind of capacity control but simply serve as an instrument of price discrimination, allowing the supplier to induce additional low-value demand. Since the exact product specification is unknown at the time of purchase, cannibalization of high-value customers' demand is prevented.

3.2 Temporary allocation

As pointed out in the introduction, bid prices are regularly updated by reoptimization of the underlying static model. Likewise, we propose that the accepted requests for flexible products should be rearranged periodically. Since updating the bid prices anyway requires solving a mathematical optimization problem, it is intuitive to simultaneously allow the rearrangement of the accepted requests for flexible products according to the current demand forecast and capacity utilization. This will improve the previously described ad-hoc allocation mechanism. Therefore, some extensions of DLP-flex are required: First, we define additional model parameters Y_j^a for all $j \in \mathcal{J}$, which denote the number of requests that have already been accepted for a flexible product j. Furthermore, we introduce decision variables y_{jm}^a for all $j \in \mathcal{J}$, $m \in \mathcal{M}_j$, which represent capacity allocations to the accepted requests in the different

execution modes. In order to ensure that an adequate amount of capacity remains available for all accepted flexible requests $Y_1^a, \ldots, Y_{n^f}^a$, we add the constraints

$$\sum_{m \in \mathcal{M}_j} y_{jm}^a = Y_j^a \qquad \text{for all } j \in \mathcal{J} \tag{9}$$

and replace y_{jm} with $\left(y_{jm}^a + y_{jm}\right)$ in constraint (2).

With these modifications at hand, the control process is basically unchanged. However, checking against capacity now requires explicitly considering the allocations of requests for flexible products within the selling process, that is, the check has to be performed against reduced capacity $c_h - \sum_{j \in \mathcal{J}} \sum_{m \in \mathcal{M}_j} a_{hm} \cdot y_{jm}^a$ instead of c_h. While the required resources are still immediately reduced if a request for a specific product i is accepted ($c_h := c_h - a_{hi} \quad \forall h \in \mathcal{H}$), there is no immediate reduction of capacity for any accepted flexible request j, but the correspondent variable that stores the number of accepted requests in mode m' must be adjusted accordingly, namely $y_{jm'}^a := y_{jm'}^a + 1$. However, these allocations are only temporary: Just before the extended model is resolved to obtain updated bid prices, all accepted requests for each flexible product $j \in \mathcal{J}$ are consolidated in the corresponding model parameter Y_j^a (that is, $Y_j^a := \sum_{m \in \mathcal{M}_j} y_{jm}^a \quad \forall k \in \mathcal{J}$) and can thus be rearranged, leading to new temporary allocations y_{jm}^a.

Example 2. *To illustrate temporary allocation, we again use the setting described in Example 1. After handling the first four requests as the ad-hoc mechanism did (see also Table 1), the reoptimization not only yields the bid prices $\pi_A = \$100$, $\pi_B = \$40$, and $\pi_C = \$110$, but also rearranges the two accepted flexible requests from resource A to B, leading to $c_A = 2$, $c_B = 0$ and $c_C = 1$. Compared to ad-hoc, P_A can now be accepted. The following four requests for P_{flex} and P_B are rejected. Finally, P_C is accepted, obtaining a total revenue of \$290.*

The complete control mechanism is as follows:

Procedure 2. Temporary allocation

Precondition: an incoming request for $i \in \mathcal{I}$ or $j \in \mathcal{J}$, bid prices $\pi_h \quad \forall h \in \mathcal{H}$, capacity c_h $\forall h \in \mathcal{H}$, temporary allocations $y_{km}^a \quad \forall k \in \mathcal{J}, m \in \mathcal{M}_k$, expected demands-to-come $\bar{D}_{nt}^s \quad \forall n \in \mathcal{I}$ and $\bar{D}_{kt}^f \quad \forall k \in \mathcal{J}$.

(1) if a *reoptimization* of the underlying model is scheduled:

(1a) set $Y_k^a := \sum_{m \in \mathcal{M}_k} y_{km}^a \quad \forall k \in \mathcal{J}$;

(1b) solve DLP-flex with the extensions described in Section 3.2 with $c_h \quad \forall h \in \mathcal{H}$, $Y_k^a \quad \forall k \in \mathcal{J}$, $\bar{D}_{nt}^s \quad \forall n \in \mathcal{I}$ and $\bar{D}_{kt}^f \quad \forall k \in \mathcal{J}$ as input parameters;

(1c) update the bid prices $\pi_h \quad \forall h \in \mathcal{H}$ and the temporary allocations $y_{km}^a \quad \forall k \in \mathcal{J}, m \in \mathcal{M}_k$ according to the model's output;

(2) if the incoming request is for *specific product* $i \in \mathcal{I}$:

(2a) if $r_i \geq \sum_{h \in \mathcal{H}} a_{hi} \cdot \pi_h$ and $a_{hi} \leq c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y_{km}^a \quad \forall h \in \mathcal{H}$:

accept request;

set $c_h := c_h - a_{hi} \quad \forall h \in \mathcal{H}$;

(2b) else: reject request;

(3) if the incoming request is for *flexible product* $j \in \mathcal{J}$:

(3a) if $\exists m \in \mathcal{M}_j$ with $f_j \geq \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h$ and $a_{hm} \leq c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y_{km}^a \quad \forall h \in \mathcal{H}$:

accept request;

set $y_{jm'}^a := y_{jm'}^a + 1$ with

$$m' = \underset{m \in \mathcal{M}_j}{arg\,min} \left\{ \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h \;\middle|\; a_{hm} \leq c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y_{km}^a \quad \forall h \in \mathcal{H} \right\};$$

(3b) else: reject request;

3.3 Pooling

According to our classification scheme (see Fig. 1), the temporary allocation mechanism described before allows for *time-based reallocations* of requests for flexible products. However, quite obviously, temporary allocation of resources may lead to suboptimal control decisions. In particular, incoming high-value requests for specific products might not be accepted because some or all of the resources needed are temporarily occupied by requests for flexible products, although there is enough capacity left in other possible execution modes. Hence, it is desirable to find some kind of *request-based* mechanism that can take the capacity available for other execution modes into account when having to decide whether to accept a request.

One way to achieve this flexibility is to construct a common resource pool for accepted requests for flexible products by deriving an appropriate surrogate constraint from the previously presented model. The intuition of the approach is to pool the capacity still available for flexible products, instead of considering each execution mode separately, as done by the aforementioned mechanisms. More precisely, multiple potential execution modes' capacity is

aggregated in a pool whose "virtual" capacity then directly delivers the maximum number of requests for flexible products that can be accepted in the remaining booking horizon, given that the bid price condition is satisfied. In the following, we facilitate the presentation by explaining the resulting mechanism for a network with a *single flexible product* j that uses only *one resource* in each execution mode. The mechanism is simple to handle and can be adapted to more complex networks, but is not applicable to arbitrary network structures.

The pooling mechanism is based on the extended static model formulation presented in Section 3.2. After having solved the model, the temporary allocations y_{jm}^{a} for all $m \in \mathcal{M}_j$ are discarded and a common resource pool $\tilde{\mathcal{H}}$ is constructed. However, it is not mandatory to include all potential resources in the flexible product pool, as it could be better to protect some of them for higher valued requests. Thus, only resources with bid prices below or equal to the revenue f_j of product j are aggregated in $\tilde{\mathcal{H}}$. Furthermore, the following special case has to be considered: After a phase of low bid prices leading to the acceptance of requests for the flexible product, a subsequent reoptimization could yield higher bid prices, preventing the acceptance of further requests for the flexible product. Even in this case, it is desirable to maintain flexibility with respect to the *already accepted* requests for the flexible product when deciding on incoming requests for specific products. Thus, the resources with the lowest bid price are included in $\tilde{\mathcal{H}}$.

Let $\bar{\pi}_m$ denote the bid price of the resource used in execution mode m (that is $\bar{\pi}_m = \sum_{l \in \mathcal{H}} a_{lm} \cdot \pi_l$). Then the two aforementioned cases are combined by including all execution modes with bid prices below or equal to $max\left\{f_j, \min_{m \in \mathcal{M}_j} \bar{\pi}_m\right\}$ in the pool, where f_j comes into effect when the flexible product is still sold and $\min_{m \in \mathcal{M}_j} \bar{\pi}_m$ when no further requests for the flexible product are accepted because the bid prices $\bar{\pi}_m$ of all execution modes exceed f_j ($f_j < \min_{m \in \mathcal{M}_j} \bar{\pi}_m$). Thus, the set $\tilde{\mathcal{H}}$ can be defined as follows:

$$\tilde{\mathcal{H}} := \left\{ h \in \mathcal{H} \,\middle|\, \exists m \in \mathcal{M}_j \text{ with } a_{hm} = 1 \text{ and } \pi_h \leq max\left\{f_j, \min_{m \in \mathcal{M}_j} \bar{\pi}_m\right\} \right\} \tag{10}$$

The aggregated, virtual capacity c^p of the pool then simply equals the sum of the capacity c_h of the resources included in $\tilde{\mathcal{H}}$, decreased by the number of requests for the flexible product already accepted:

$$c^p := \sum_{h \in \tilde{\mathcal{H}}} c_h - \sum_{m \in \mathcal{M}_j} y_{jm}^{a} \tag{11}$$

Note that, by construction, the calculation of c^p by (11) always leads to nonnegative values. This results from all resources with capacity temporarily allocated to accepted requests for the flexible product also being included in $\tilde{\mathcal{H}}$, due to the latter's definition based on bid prices (see (10)).

Compared to the previously described temporary allocation mechanism, the control process differs only slightly. While the bid price conditions (7) and (8) remain as before, the check against capacity is now accomplished by means of the resource pool instead of using the adjusted values $c_h - \sum_{j \in \mathcal{J}} \sum_{m \in \mathcal{M}_j} a_{hm} \cdot y^a_{jm}$. Hence, a request for the flexible product j is simply acknowledged if there is enough capacity left in the pool ($c^p \geq 1$). If the request is accepted, the pool capacity is reduced by one. When a request for a specific product $i \in \mathcal{I}$ comes in, the required resources' capacity must be sufficient ($a_{hi} \leq c_h \quad \forall h \in \mathcal{H}$). Furthermore, if the resources included in the pool are required for product i, the virtual capacity c^p of the pool has to be sufficient as well ($\sum_{h \in \tilde{\mathcal{H}}} a_{hi} \leq c^p$). If i is accepted, the resources' specific capacity values must be reduced accordingly. Moreover, the virtual pool capacity also has to be reduced to reflect that the capacity needed by specific product i is no longer available to the flexible product.

Example 3. *The resource pool $\tilde{\mathcal{H}}$ is constructed using the initial bid prices from the previous examples. As all the bid prices do not exceed f_{flex}, it contains A, B, and C with $c^p = 5$ (see Table 1). The first two requests for P_{flex} are accepted because the bid price condition has been fulfilled and c^p is sufficient. Consequently, the pool capacity is reduced to $c^p = 3$. After successfully performing a bid price check on π_A to decide on the acceptance of the following request for P_A, c_A and – since A is included in the pool – c^p are checked to ensure sufficient capacity. Both requests for P_A are accepted, leading to a remaining capacity of $c_A = 0$ and $c^p = 1$, respectively. Afterwards, the reoptimization is performed and the new pool encompasses only resource B with $c^p = 0$, according to the new bid prices. The following five requests are rejected due to a lack of capacity with respect to resource A and the pool. P_C is accepted because the bid price π_C equals r_C and the capacity $c_C = 1$ is sufficient. c^p is not considered here, because C is not included in the pool. The resulting revenue is \$390.*

The following procedure summarizes the complete algorithm:

Procedure 3. Pooling

Precondition: an incoming request for $i \in \mathcal{I}$ or j, bid prices $\pi_h \ \forall h \in \mathcal{H}$, remaining capacity $c_h \ \forall h \in \mathcal{H}$, remaining capacity c^p of the resource pool of the sole flexible product j with $\sum_{h \in \mathcal{H}} a_{hm} = 1 \forall m \in \mathcal{M}_j$, the set of resources $\tilde{\mathcal{H}}$ included in the pool; expected demands-to-come $\bar{D}^s_{nt} \ \forall n \in \mathcal{I}$ and $\bar{D}^f_{jt}$ for the flexible product j.

(1) if a *reoptimization* of the underlying model is scheduled:

(1a) set $Y^a_j := \sum_{h \in \tilde{\mathcal{H}}} c_h - c^p$;

(1b) solve DLP-flex with the extensions described in Section 3.2 with $c_h \ \forall h \in \mathcal{H}$, Y^a_j, $\bar{D}^s_{nt} \ \forall n \in \mathcal{I}$ and $\bar{D}^f_{jt}$ as input parameters;

(1c) update the bid prices $\pi_h \ \forall h \in \mathcal{H}$ according to the model's output;

(1d) set $\tilde{\mathcal{H}} := \left\{ h \in \mathcal{H} \,\middle|\, \exists m \in \mathcal{M}_j \text{ with } a_{hm} = 1 \text{ and } \pi_h \le max\left\{ f_j, \min_{m \in \mathcal{M}_j} \bar{\pi}_m \right\} \right\}$;

(1e) set $c^p := \sum_{h \in \tilde{\mathcal{H}}} c_h - \sum_{m \in \mathcal{M}_j} y^a_{jm}$;

(2) if the incoming request is for *specific product* $i \in \mathcal{I}$:

(2a) if $r_i \ge \sum_{h \in \mathcal{H}} a_{hi} \cdot \pi_h$ and $a_{hi} \le c_h \ \forall h \in \mathcal{H}$ and $\sum_{h \in \tilde{\mathcal{H}}} a_{hi} \le c^p$:

accept request;

set $c_h := c_h - a_{hi}$ for all $h \in \mathcal{H}$;

set $c^p := c^p - \sum_{h \in \tilde{\mathcal{H}}} a_{hi}$;

(2b) else: reject request;

(3) if the incoming request is for *flexible product* j:

(3a) if $\exists m \in \mathcal{M}_j$ with $f_j \ge \bar{\pi}_m$ and $c^p \ge 1$:

accept request;

set $c^p := c^p - 1$;

(3b) else: reject request;

The intuition behind step (1) is as follows: Whenever a reoptimization of the underlying static model has to be performed, the number of accepted requests for flexible products Y^a_j is determined by calculating the difference between the sum of the pooled resources' capacity $\sum_{h \in \tilde{\mathcal{H}}} c_h$ and the capacity c^p of the pool, that is $Y^a_j := \sum_{h \in \tilde{\mathcal{H}}} c_h - c^p$. Knowing Y^a_j, DLP-flex with the extensions from Section 3.2 can now be used to update the bid prices π_h. The resource pool is then again constructed according to (10), based on the new set of bid prices. To complete the reoptimization step, the capacity c^p of the resource pool is calculated as given by

(11). Steps (2) and (3) represent the control process for specific requests and flexible ones, respectively, which are handled as described above.

3.4 Shifting

Although pooling circumvents the temporary allocation of requests for flexible products, the static *ex ante* construction of the resource pool $\tilde{\mathcal{H}}$ may lead to suboptimal results when deciding whether to accept incoming requests for specific products. More precisely, the decision whether a resource's capacity should be made available for the pool is actually strongly dependent on the value of the incoming request requiring capacity from the resource pool, which is not known in advance. Consequently, a somewhat more sophisticated algorithm may be helpful. From a practical perspective, however, it is important that such a mechanism remains intuitive for it to be realized with little effort: Similar to the aforementioned approaches, it should exclusively rely on the current set of bid prices having been calculated during the last optimization.

In order to ensure practical applicability, in the following, we therefore suggest a simple and straightforward greedy heuristic extending the temporary allocation mechanism in a different way by partially allowing the rearrangement of temporary assigned requests for flexible products between two scheduled optimizations. For restricted settings to which pooling is applicable, it can even be shown that the procedure always leads to optimal decisions given the current set of bid prices. The basic idea of the proposed heuristic – after which the resulting mechanism is named – is to make room on the resources that are scarce with respect to the current request for a specific product by *shifting* accepted requests for flexible products to alternative execution modes requiring resources that are not scarce. Note that in this context, scarcity of a particular resource means that its remaining capacity is not sufficient to fulfill the request under consideration. Hence, as a first step, the resources that are scarce with respect to the current request for a specific product are successively processed, trying for each h of them to find a flexible product k with current execution mode m that can be transferred to an alternative execution mode m' such that capacity on the current resource h is freed up. If there are several possibilities, a rearrangement that leads to the lowest opportunity cost is selected. This cost can be calculated by simply taking the sum of the current bid price values for the resource requirements in the potential execution mode m' obtained from the last optimiza-

tion, minus the corresponding sum for the resources currently required in mode m, which would be freed up. That is, in a "greedy" manner, a triple (k^*,m^*,m'^*) with a minimal $\sum_{h\in\mathcal{H}} a_{hm'^*}\cdot\pi_h - \sum_{h\in\mathcal{H}} a_{hm^*}\cdot\pi_h$ is chosen as a candidate for shifting.

After having processed all the scarce resources and, thus, having identified a potential set $\mathcal{U}$ of rearrangements, which would, in total, allow the current request for specific product i to be accepted, the final decision has to be made. Therefore, the revenue obtained from i must be compared to the total opportunity cost resulting from the acceptance, similar to the classical bid price condition. In this case, however, the opportunity cost does not only consist of the direct cost resulting from the acceptance, but also incorporates the indirect costs implied by the rearrangements defined by $\mathcal{U}$ as explained above. That is, the request is accepted and the proposed shifts of requests for flexible products are finally performed if (and only if) the following condition holds:

$$r_i \geq \sum_{h\in\mathcal{H}} a_{hi}\cdot\pi_h + \sum_{(k,m,m')\in\mathcal{U}} \left(\sum_{h\in\mathcal{H}} a_{hm'}\cdot\pi_h - \sum_{h\in\mathcal{H}} a_{hm}\cdot\pi_h \right) \tag{12}$$

Example 4. *Applied to the setting outlined in Example 1, the shifting mechanism accepts the two requests for P_{flex} and, in the same way as temporary allocation, assigns them to A (see Table 1). However, contrary to temporary allocation, the following two requests for P_A can be accepted by shifting the flexible requests to B. The request for P_A after the reoptimization as well as the two for P_{flex} are declined, whereas P_B is accepted. This is because the bid price condition (12) is fulfilled and capacity on B is freed up by shifting an accepted flexible request from B to C. Although π_C prevents the acceptance of flexible requests using C, this rearrangement can be performed because the revenue r_B is not lower than the total opportunity cost resulting from accepting P_B and performing the rearrangement ($r_B \geq \pi_B + (\pi_C - \pi_B)$). Thus, all capacity is occupied, leading to a revenue of \$400.*

More formally, the entire shifting mechanism can be stated as follows:

Procedure 4. Shifting

Precondition: an incoming request for $i \in \mathcal{I}$ or $j \in \mathcal{J}$, bid prices $\pi_h \ \forall h \in \mathcal{H}$, capacity c_h $\forall h \in \mathcal{H}$, temporary allocations $y^a_{km} \ \forall k \in \mathcal{J}, m \in \mathcal{M}_k$, expected demands-to-come $\bar{D}^s_{nt} \ \forall n \in \mathcal{I}$ and $\bar{D}^f_{kt} \ \forall k \in \mathcal{J}$.

(1) if a *reoptimization* of the underlying model is scheduled:
……*see Procedure 2*…….

(2) if the incoming request is for *specific product* $i \in \mathcal{I}$:

(2a) if $r_i \geq \sum_{h \in \mathcal{H}} a_{hi} \cdot \pi_h$ and $a_{hi} \leq c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y^a_{km} \quad \forall h \in \mathcal{H}$:

accept request;

set $c_h := c_h - a_{hi} \quad \forall h \in \mathcal{H}$;

(2b) elseif $r_i \geq \sum_{h \in \mathcal{H}} a_{hi} \cdot \pi_h$ and $\exists h \in \mathcal{H}$ with $a_{hi} > c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y^a_{km}$

and $a_{hi} \leq c_h \quad \forall h \in \mathcal{H}$:

set $\mathcal{U} := \varnothing$;

set $c_h' := c_h - min\left\{ a_{hi}, c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y^a_{km} \right\} \quad \forall h \in \mathcal{H}$;

set $y^a_{km}' := y^a_{km} \quad \forall k \in \mathcal{J}, m \in \mathcal{M}_k$;

set $\hat{\mathcal{H}} := \left\{ h \in \mathcal{H} \middle| a_{hi} > c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y^a_{km} \right\}$;

foreach $h \in \hat{\mathcal{H}}$:

if $c_h' = c_h$:

set $\mathcal{P} := \left\{ k \in \mathcal{J} \middle| \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y^a_{km}' > 0 \right\}$;

set $\mathcal{N}_k := \left\{ m \in \mathcal{M}_k \middle| a_{hm} \cdot y^a_{km}' > 0 \right\} \quad \forall k \in \mathcal{P}$;

set $\mathcal{S} := \left\{ (k, m, m') \middle| k \in \mathcal{P},\ m \in \mathcal{N}_k, m' \in \mathcal{M}_k, a_{hm'} = 0, \right.$

$\left. c_g' - \sum_{q \in \mathcal{J}} \sum_{o \in \mathcal{M}_q} a_{go} \cdot y^a_{qo}' > 0 \ \forall g \in \{ l \in \mathcal{H} \mid a_{lm} = 0, a_{lm'} > 0 \} \right\}$;

if $\mathcal{S} = \varnothing$: break;

set $(k^*, m^*, m'^*) := \underset{(k,m,m') \in \mathcal{S}}{arg\,min} \left\{ \sum_{h \in \mathcal{H}} a_{hm'} \cdot \pi_h - \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h \right\}$;

set $\mathcal{U} := \mathcal{U} \cup (k^*, m^*, m'^*)$;

set $y^a_{k^* m^*}' := y^a_{k^* m^*}' - 1$;

set $y^a_{k^* m'^*}' := y^a_{k^* m'^*}' + 1$;

set $c_g' := c_g - min\left\{ a_{gi}, c_g - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{gm} \cdot y^a_{km}' \right\} \quad \forall g \in \mathcal{H}$;

if $a_{hi} \leq c_h - \sum_{k \in \mathcal{J}} \sum_{m \in \mathcal{M}_k} a_{hm} \cdot y^a_{km}' \ \forall h \in \hat{\mathcal{H}}$ and

$r_i \geq \sum_{h \in \mathcal{H}} a_{hi} \cdot \pi_h + \sum_{(k,m,m') \in \mathcal{U}} \left(\sum_{h \in \mathcal{H}} a_{hm'} \cdot \pi_h - \sum_{h \in \mathcal{H}} a_{hm} \cdot \pi_h \right)$:

accept request;

set $y^a_{km} := y^a_{km}' \quad \forall k \in \mathcal{J}, m \in \mathcal{M}_k$;

set $c_h := c_h - a_{hi} \quad \forall h \in \mathcal{H}$;

(2c) else: reject request;

(3) if the incoming request is for *flexible product* $j \in \mathcal{J}$:

……*see Procedure 2*…….

The formulation of step (2b) can be explained as follows: While temporary allocation simply declines an incoming request for specific product i if the bid price condition has been satisfied but the current temporary allocations prevent acceptance, shifting now comes into play. We first create duplicates c_h' and y^a_{km}' of the resources' current capacity and the temporary allocations, respectively. This allows us to roll matters back if it turns out that there is no suitable shifting strategy that permits the acceptance of the incoming specific request. In particular, c_h' is required to implement the general idea of the heuristic's chosen design, which is to always reserve space for the incoming specific request as soon as capacity is made available. Consequently, given the temporary allocations of flexible products, we begin by allocating capacity on the duplicated resources needed by the requested product i by reducing the values c_h' wherever possible ($c_h' := c_h - min\left\{a_{hi}, c_h - \sum_{k\in\mathcal{J}} \sum_{m\in\mathcal{M}_k} a_{hm} \cdot y^a_{km}'\right\} \forall h \in \mathcal{H}$). Through this kind of reservation, we avoid reoccupying resources through a later shifted request for a flexible product. All resources that are scarce, in the sense that they are needed to accept the request but for which reduction is not possible at the moment, are included in the set $\hat{\mathcal{H}}$. To try to make room for the incoming request on these resources, $\hat{\mathcal{H}}$ is now processed, as described before, by iteratively considering each $h \in \hat{\mathcal{H}}$. To find a candidate for shifting, we define the set $\mathcal{S}$ of potential triples (k, m, m'), freeing up the current resource h. In this context, the sets $\mathcal{P}$ and $\mathcal{N}_k$ are used to ensure the existence of at least one request of product k in execution mode m that uses resource h. If there is no direct shifting possibility that frees up h, meaning that $\mathcal{S}$ is empty, the current iteration can be skipped. Otherwise, the "best" shifting possibility (k^*, m^*, m'^*) is identified as described before and added to set $\mathcal{U}$ of potential rearrangements. At the end of each iteration, the temporary allocations' values are adjusted according to the identified candidate for shifting ($y^a_{k^*m^*}' := y^a_{k^*m^*}' - 1$ and $y^a_{k^*m'^*}' := y^a_{k^*m'^*}' + 1$). Furthermore, following the general intuition of the design mentioned before, the values c_h' are again updated for all resources, so that *all* freed-up resources are immediately reserved if they are required to fulfill the request for product i. Note that this is why each iteration is only performed if $c_h' = c_h$; otherwise ($c_h' < c_h$), the current resource has already been freed up by

an earlier iteration and reconsideration is not necessary. If, after having completely processed set $\hat{\mathcal{H}}$, the rearrangements contained in $\mathcal{U}$ are sufficient to permit acceptance and if, according to (12), the overall decision is to perform them in order to accept the request, acceptance is finalized by simply overwriting the temporary allocations' original values with their duplicates ($y^a_{km} := y^a_{km}{}' \quad \forall k \in \mathcal{J}, m \in \mathcal{M}_k$) and adjusting capacity with respect to the incoming request ($c_h := c_h - a_{hi} \quad \forall h \in \mathcal{H}$). The procedure can be extended in a straightforward manner so that shifting is also performed when a request for a flexible product comes in, which becomes relevant if there are several flexible products that are not disjoint with respect to the resources they can use. Furthermore, in this case, one might be tempted to further extend the procedure to allow arbitrary recursive rearrangements of requests for flexible products. Remember, however, that extensions of this dimension are contrary to the heuristic's primary objective to arrive at a comprehensible, intuitive-to-use mechanism suitable for practical applications.

3.5 Certainty equivalent control (CEC)

The most accurate way of exploiting a static model's output for capacity control is the application of certainty equivalent control (CEC), which has originally been proposed for traditional DLP formulations by Bertsimas and Popescu [3]. The approach is a straightforward result of considering the capacity control problem's original stochastic dynamic program formulation and replacing the stochastic demand influences with their deterministic equivalents. In the fundamental literature on approximate dynamic programming, this technique is usually referred to as CEC (see, e.g., [27]); the term is also common in revenue management (see, e.g. [3]). However, to avoid misunderstanding, remember that all other approaches presented in this paper also assume that demand is deterministic, as they basically rely on the same underlying mathematical model formulation.

Instead of using bid prices, CEC directly calculates the "true" opportunity cost of each incoming request. For this purpose, two instances of DLP-flex with the extensions described in Section 3.2 are solved for each request. While the first one calculates an approximation to the optimal revenue-to-go under the assumption that the current request has been rejected, the second one calculates the approximated optimal revenue-to-go as if it has been accepted. If the current request is for a specific product, the latter model is obtained by additionally reserving the capacity for the requested product i through the appropriate adjustment of the

right-hand sides of the constraints (2). If there is an incoming request for a flexible product, the right-hand side of the related constraint (9) is simply incremented by one. The difference between the two models' objective function values then obviously corresponds to the opportunity cost caused by accepting the current request. Consequently, the request should be accepted if (and only if) this cost is not larger than the proceeds of the sale. If the second model has no feasible solution due to the resource restrictions, the incoming request can obviously not be accepted. For simplicity's sake, we define the value of the objective function to be minus infinity in this case, so that the opportunity cost is infinite and the request will not be accepted by the aforementioned criteria.

Example 5. *In order to decide on the first request of the example, CEC first calculates a model with the initial capacities. Then, as the request is a flexible one, a second model with the number of accepted flexible request incremented by one ($Y^a = 1$, as if the request had been accepted) is calculated. We assume that the opportunity cost – as calculated by the difference between the two objective values – does not exceed \$40, allowing the request to be accepted. A bit later, the second request comes in. Again, two models (one with $Y^a = 1$ and the second with $Y^a = 2$) are calculated, using the updated demand forecast available at this point in time. This leads to an opportunity cost of more than \$40 and the request is rejected. The remaining eight requests are processed likewise, leading to the decisions given in Table 1 and a total revenue of \$470.*

The whole CEC mechanism is given as follows:

Procedure 5. CEC

Precondition: an incoming request for $i \in \mathcal{I}$ or $j \in \mathcal{J}$, capacity $c_h \ \forall h \in \mathcal{H}$, accepted requests for flexible products $Y_k^a \ \forall k \in \mathcal{J}$, expected demands-to-come $\bar{D}_{nt}^s$ $\forall n \in \mathcal{I}$ and $\bar{D}_{kt}^f \ \forall k \in \mathcal{J}$.

(1) if the incoming request is for *specific product* $i \in \mathcal{I}$:

(1a) solve DLP-flex with the extensions described in Section 3.2 with $c_h \ \forall h \in \mathcal{H}$ and $Y_k^a \ \forall k \in \mathcal{J}$ as input parameters; set V_1 to the optimal value;

(1b) solve DLP-flex with the extensions described in Section 3.2 with $c_h - a_{hi}$ $\forall h \in \mathcal{H}$ and $Y_k^a \ \forall k \in \mathcal{J}$ as input parameters; set V_2 to the optimal value (set $V_2 := -\infty$ if the model has no feasible solution);

(1c) if $r_i \geq V_1 - V_2$:

accept request;

set $c_h := c_h - a_{hi} \quad \forall h \in \mathcal{H}$;

(1d) else: reject request;

(2) if the incoming request is for *flexible product* $j \in \mathcal{J}$:

(2a) solve DLP-flex with the extensions described in Section 3.2 with $c_h \quad \forall h \in \mathcal{H}$ and $Y_k^a \quad \forall k \in \mathcal{J}$ as input parameters; set V_1 to the optimal value;

(2b) solve DLP-flex with the extensions described in Section 3.2 with $c_h \quad \forall h \in \mathcal{H}$, $Y_k^a \quad \forall k \in \mathcal{J} \setminus \{j\}$ and $Y_j^a + 1$ as input parameters; set V_2 to the optimal value (set $V_2 := -\infty$ if the model has no feasible solution);

(2c) if $f_j \geq V_1 - V_2$:

accept request;

set $Y_j^a := Y_j^a + 1$;

(2d) else: reject request;

Note that according to our classification scheme (see Fig. 1), CEC is the approach that maximally exploits the flexibility arising from flexible products, as it simultaneously allows for *request-based reallocation* as well as for *request-based reoptimization*. Nevertheless, it has to be noted that CEC is often only of theoretical interest, as most practical applications need rather simpler mechanisms, especially without the need to permanently resolve mathematical optimization problems. This is also crucial for many fields of application where the selling process requires quick responses and even solving linear problems is too time-consuming due to the practical problem size.

Table 1
Overview of Examples 1–5

request	ad-hoc			temporary allocation			pooling					shifting			CEC				
no type	✓/×	(c_A,c_B,c_C)	rev.	✓/×	(c_A,c_B,c_C)	rev.	✓/×	pool $\tilde{\mathcal{H}}$	c_p	(c_A,c_B,c_C)	rev.	✓/×	(c_A,c_B,c_C)	rev.	✓/×	Y^a	(c_A,c_B,c_C)	rev.	
	starting bid prices: 40/40/40			starting bid prices: 40/40/40			starting bid prices: 40/40/40					starting bid prices: 40/40/40							
		(2,2,1)			(2,2,1)			{A,B,C}	5	(2,2,1)			(2,2,1)			0	(2,2,1)		
1 P_{Flex}	✓	(1,2,1)	$40	✓	(1,2,1)	$40	✓	{A,B,C}	4	(2,2,1)	$40	✓	(1,2,1)	$40	✓	1	(2,2,1)	$40	
2 P_{Flex}	✓	(0,2,1)	$40	✓	(0,2,1)	$40	✓	{A,B,C}	3	(2,2,1)	$40	✓	(0,2,1)	$40	×	1	(2,2,1)		
3 P_A	×	(0,2,1)		×	(0,2,1)		✓	{A,B,C}	2	(1,2,1)	$100	✓	(0,1,1)	$100	✓	1	(1,2,1)	$100	
4 P_A	×	(0,2,1)		×	(0,2,1)		✓	{A,B,C}	1	(0,2,1)	$100	✓	(0,0,1)	$100	✓	1	(0,2,1)	$100	
	reoptim. bid prices: 100/40/110			reoptim. bid prices: 100/40/110			reoptim. bid prices: 100/40/110					reoptim. bid prices: 100/40/110							
		(0,2,1)			(2,0,1)			{B}	0	(0,2,1)			(0,0,1)						
5 P_A	×	(0,2,1)		✓	(1,0,1)	$100	×	{B}	0	(0,2,1)		×	(0,0,1)		×	1	(0,2,1)		
6 P_{Flex}	✓	(0,1,1)	$40	×	(1,0,1)		×	{B}	0	(0,2,1)		×	(0,0,1)		×	1	(0,2,1)		
7 P_{Flex}	✓	(0,0,1)	$40	×	(1,0,1)		×	{B}	0	(0,2,1)		×	(0,0,1)		×	1	(0,2,1)		
8 P_B	×	(0,0,1)		×	(1,0,1)		×	{B}	0	(0,2,1)		✓	(0,0,0)	$120	✓	1	(0,1,1)	$120	
9 P_{Flex}	×	(0,0,1)		×	(1,0,1)		×	{B}	0	(0,2,1)		×	(0,0,0)		×	1	(0,1,1)		
10 P_C	✓	(0,0,0)	$110	✓	(1,0,0)	$110	✓	{B}	0	(0,2,0)	$110	×	(0,0,0)		✓	1	(0,1,0)	$110	
remaining cap.:		(0,0,0)			(1,0,0)					(0,0,0)			(0,0,0)				(0,0,0)		
total revenue:			$270			$290					$390			$400					$470

4 Computational experiments

4.1 Simulation experiment design

To perform the computational experiments, a simulation framework was implemented which allows for the definition of several problem classes with different instances automatically being generated. The problem classes are based on two underlying resource networks which are motivated by passenger airline revenue management, but also occur in a wide range of other areas.

Network 1 consists of four similar and independent flight legs with no connection possibilities and a total capacity of $C_h = 200$ homogeneous seats each, as this is typical of low cost carriers. We define 16 *specific products* for each combination of flight and one of four booking classes which are priced at 550, 400, 300, and 210 on each flight. Furthermore, a *flexible product* is offered that assures transportation on one of the four flights, priced at 168. For each flight leg *h*, a *nominal load factor* (LF_h) is defined as the total expected demand for seats divided by the total number of available seats. LF_h splits up into 85% expected demand for specific products, which in turn is assigned according to the ratios 1:2:3:4 to the four booking classes, and 15% demand for the flexible product. By using different sets of input values LF_h, we define three *problem classes* reflecting high demand (1-H), medium demand (1-M), and low demand (1-L) with an average nominal load factor of 1.4, 1.1, and 0.9, respectively.

Network 2 represents a part of the daily network of a full service carrier and therefore allows for a combination of flight legs to form connection flights. As the network is much more complex, we do not go into too much detail but restrict ourselves to a description of its general properties. Network 2 consists of 15 flight legs, each of which corresponds to a possible itinerary, connecting four cities A, B, C, and D (see Fig. 2): There are three short haul flights from A to B (1–3) and three from C to D (4–6). Seven medium haul flights connect A with C (7–9) and B with D (10–13). D can be reached from A directly with two long haul flights (14, 15). Capacity varies between 250 and 400 homogeneous seats. Furthermore, nine two-leg itineraries, consisting of one short and one medium haul leg, are available. They connect A and D with a transfer in B or C. We consider only one compartment and four booking classes for each of the itineraries, leading to a total of 96 *specific products*. The direct long haul flights from A to D are priced at 1200, 950, 700, and 500, depending on the booking class.

Passengers with a transfer in B or C obtain a discount of 8% on these fares. Single medium flights are charged at 80%, short haul flights at 30% of the corresponding booking class on a long haul flight. Furthermore, three different *flexible products* connecting A to D are offered at 25% below the direct flight's fare in the cheapest booking class. The first product guarantees a direct flight from A to D with flexible departure time either early in the morning (14) or late in the evening (15). The second one offers transportation on a one-stop itinerary from A to D with early departure, consisting of one of the pairs of flights (1,10), (1,11), (7,4), or (7,5). The third product is a one-stop itinerary from A to D with late departure and possible transportation modes (3,12), (3,13), or (9,6). As with network 1, we define three *problem classes* reflecting high (2-H), medium (2-M), and low (2-L) demand with capacity weighted average nominal load factors LF_h of 1.3, 1.1, and 0.9, respectively. On legs being involved by one of the flexible products, LF_h splits up into 80% for specific products, with demand ratios 1:2:3:4 with respect to the booking classes, and 20% for the corresponding flexible product. Note that these assumptions regarding the amount of flexible product demand in relation to the network's total demand situation are rather conservative. More specific details on the different problem classes of networks 1 and 2, like the individual values of load factors and expected demands, can be found in the appendix of [17], from which the simulation setting has been adopted.

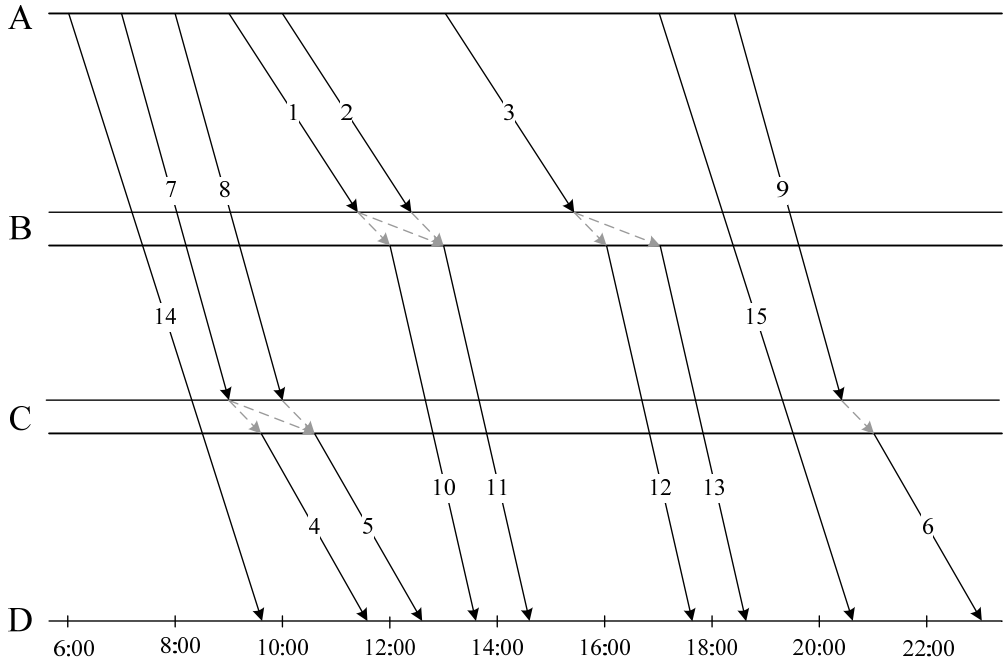


Fig. 2. Network 2 (time space network)

In order to obtain simulation runs for both networks, we generate booking requests for the different products according to independent, nonhomogeneous Poisson processes, a common assumption in revenue management (see [28]). The corresponding arrival rates $\lambda_i(t)$ for all $i \in \mathcal{I}$ and $\lambda_j(t)$ for all $j \in \mathcal{J}$, which are called *booking curves* in the context of airline revenue management, distribute the given total expected demand over time. We assume that these functions have a triangular shape (see [13]). As flight tickets are usually available one year in advance, we set the length of the booking period to 360 days. Identical parameter values are used for the beginning, maximum, and end of the corresponding triangle for all products based on the same booking class. The values have been chosen so that more expensive booking classes are generally requested near departure (see [17]).

To incorporate forecast accuracy in the simulations, we have implemented a stochastic *forecast error*. In order to make simulations with different forecast qualities easily comparable within the simulation study, this error has no influence on the generation of the requests themselves, but simply distorts the expectation of demand $\bar{D}_{it}^{s}$ for each specific product i ($\bar{D}_{jt}^{f}$ for each flexible product j, respectively) used within the optimization models and the control process after the requests have been generated as outlined above. Reflecting typical real-world correlation in demand behavior, we define the forecast error as *itinerary-based*, meaning that the forecasts of all products concerning a particular itinerary are distorted by the same value. This allows some itineraries to have higher demand than expected, while others are less popular than predicted. The size of the error is controlled by the parameter $\delta \in [0,1]$, which serves as the distortion's *upper error bound*. Given this parameter, a random value $\hat{\delta} \sim U(-\delta,\delta)$ is drawn within each simulation run for each itinerary, and the expected demand for each specific product i based on this itinerary is distorted by $\hat{\delta}$. That is, $(1+\hat{\delta}) \cdot \bar{D}_{it}^{s}$ is used as a forecast of the expected aggregated demand-to-come. The expectation $\bar{D}_{jt}^{f}$ is altered accordingly, but with independent values $\hat{\delta}$ drawn for each flexible product j.

A problem class, as defined earlier, combined with a specific value for the upper error bound δ, defines a certain *test case*. For each of the considered test cases, we investigate the five different control mechanisms introduced in Section 3, with the bid prices (re)calculated 13 times during the booking horizon by reoptimizing the underlying model 360, 180, 110, 80, 65, 50, 40, 30, 20, 10, 5, 3, and 1 day(s) before departure, thereby increasing the frequency towards the end of the booking horizon when demand intensity is higher. Unless otherwise

noted, all accepted requests for flexible products are finally assigned to execution modes 15 days before departure. To judge the performance of the control mechanisms, $K = 200$ independent simulation runs are performed for the test cases, each consisting of a complete selling period with requests for all products. All implementations have been undertaken by means of "Java 2 Platform, Standard Edition" (J2SE) version 1.6.0 by Sun Microsystems. The simulations ran on a PC with two 3 GHz Intel Pentium processors, 1 GB RAM and Windows XP.

4.2 Computational results

To evaluate the control mechanisms presented in Section 3, each of them is used to decide on the acceptance of requests in the previously described problem classes. The considered test cases are obtained by varying the upper error bound δ between 0 (very good forecast) and 1 (poor forecast). To facilitate the discussion, the results of selected problem classes are graphically illustrated. Table 2–5 at the end of this section provide detailed figures of all problem classes.

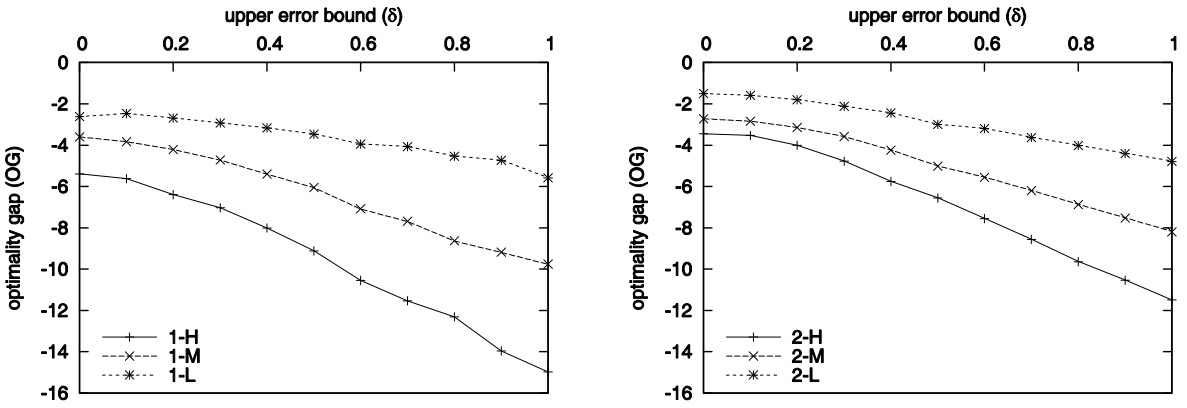
We begin with the first control mechanism presented in Section 3, the ad-hoc allocation. For our investigation, two relative measures are used: *optimality gap* (OG) and *capacity utilization* (CU). OG is the relative difference between the revenue of the ad-hoc control mechanism and the optimal value that could theoretically be realized under complete demand information. For each simulation run κ, this hindsight upper bound V_κ^{Opt} on the revenue of any control mechanism can easily be computed after observing the demand realization. Given the ad-hoc mechanism's revenue by V_κ^{AH}, OG is calculated as follows:

$$OG = \frac{1}{K}\sum_{\kappa=1}^{K}\frac{V_\kappa^{AH} - V_\kappa^{Opt}}{V_\kappa^{Opt}} \cdot 100\% \tag{13}$$

The second measure is the average capacity utilization across all resources in the network, with $R_{h\kappa}$ being the quotient of occupied and total capacity on leg h in simulation run κ:

$$CU = \frac{1}{K \cdot l}\sum_{\kappa=1}^{K}\sum_{h\in\mathcal{H}} R_{h\kappa} \cdot 100\% \tag{14}$$

In practice, this ratio of occupied to total capacity is often reported as an important performance indicator, although a higher capacity utilization obviously does not necessarily lead to a higher revenue.



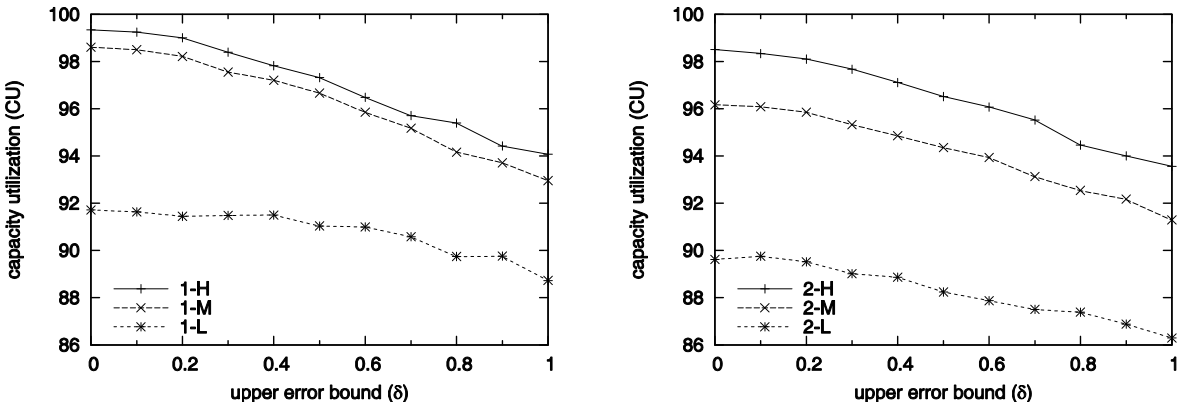
a) Network 1 b) Network 2

Fig. 3. Optimality gap of the ad-hoc mechanism

In Fig. 3a), the optimality gap of each demand situation (high, medium, and low) is plotted against δ for network 1. As OG measures the deviation from the optimum under complete information, values closer to zero indicate better performance. The respective values for network 2 are illustrated in Fig. 3b).

The ad-hoc mechanism obviously yields 2%–15% less revenue than the optimal value under complete information. Thus, the optimality gap is always negative. Two trends can be clearly observed: First, the optimality gap depends on demand. Demand scenarios with high demand relative to capacity – implying higher average nominal load factors – show larger optimality gaps than scenarios with weaker demand. Second, the optimality gap increases with forecast impreciseness. For a good forecast ($\delta = 0$) the optimality gap is between 2.5% and 5.5% in network 1, and 1.5% and 3.5% in network 2. However, the ad-hoc approach is hampered by a bad forecast, while the hindsight upper bound V_κ^{Opt} only depends on the actual demand realization. Thus, for a less accurate forecast (e.g., $\delta = 1$), the revenue is 5.5%–15% (network 1) and 4%–11% (network 2) below V_κ^{Opt}.

Capacity utilization (Fig. 4) also depends on the accuracy of the forecast. A bad forecast increases the risk of declining a request and waiting for a more valuable one which never arrives and, thus, ending up with unsold capacity. This is reflected by a decrease in capacity utilization of about 1% for every 0.25 increase in the upper error bound δ. Note that even for $\delta = 0$, the capacity utilization does not reach 100% because the variance of the Poisson process (see Section 4.1) remains and demand may be unbalanced.



a) Network 1 b) Network 2

Fig. 4. Capacity utilization of the ad-hoc mechanism

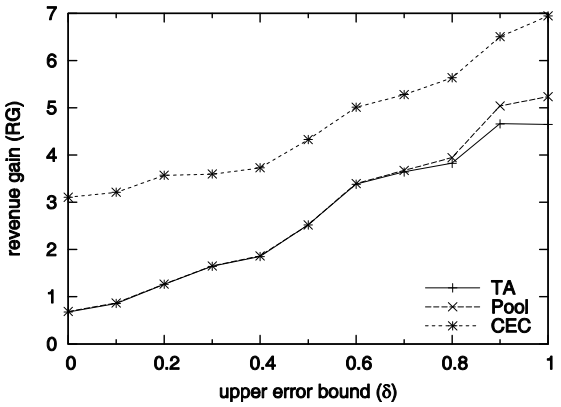
These results show that the ad-hoc mechanism leaves a significant revenue potential unused, especially when confronted with high demand and imprecise forecasts. Therefore, in the following, we analyze the performance of the other mechanisms presented in Section 3: temporary allocation, pooling, shifting and CEC. To facilitate this comparison, we define the measure *revenue gain* (RG), which shows the relative improvement of revenue V_κ^s of mechanism s over the ad-hoc mechanism's revenue V_κ^{AH}:

$$RG^s = \frac{1}{K}\sum_{\kappa=1}^{K}\frac{V_\kappa^s - V_\kappa^{AH}}{V_\kappa^{AH}} \cdot 100\% \tag{15}$$

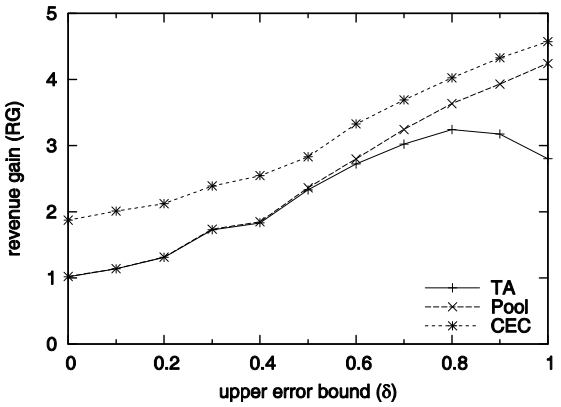
We begin our investigation with network 1. Fig. 5 illustrates temporary allocation (TA), pooling (Pool) and CEC's revenue gain for the three demand situations. Owing to the special structure of network 1, pooling is applicable in this setting. Shifting yields almost exactly the same results as pooling. Thus, it is not explicitly included in Fig. 5.

The temporary allocation mechanism attains up to 4.5% more revenue than the ad-hoc one. The gain initially increases with δ in all demand situations, as it becomes more important to correct inappropriate allocations of resources to requests for flexible products accepted earlier. However, as δ increases further, RG^{TA} remains constant or even decreases. This effect is especially striking in the 1-L problem class: initially, RG^{TA} increases slightly from just under 2% to 2.5% for $\delta = 0.6$, but then falls sharply to around 1% at $\delta = 1$. This is because even though temporary allocation tries to improve the allocation by moving requests for flexible products when reoptimizing, the new allocation remains constant until the next reoptimization. If the forecast is very poor, this new allocation may not be much better than the previous

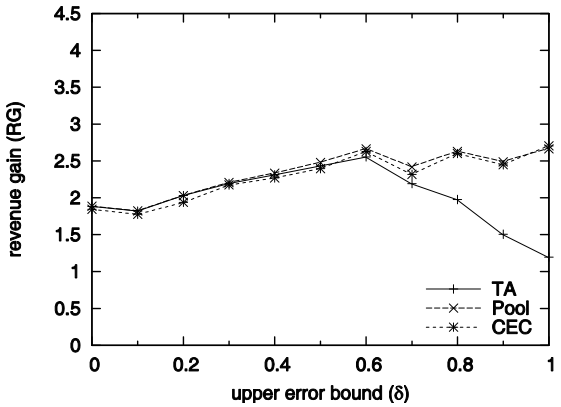
one. Thus, the benefit obtained from moving the requests for flexible products is small, but – in realistic scenarios – remains positive.



a) High demand (1-H)



b) Medium demand (1-M)



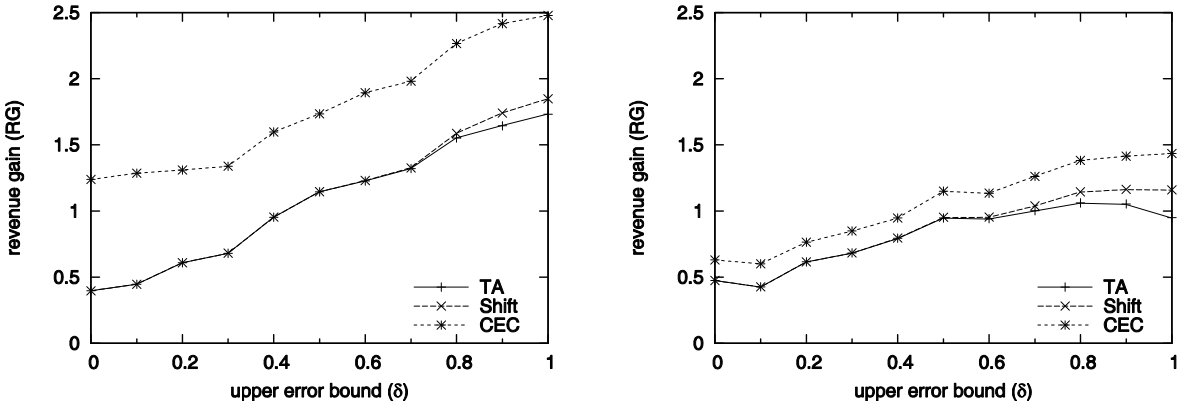
c) Low demand (1-L)

Fig. 5. Revenue gain in network 1

By aggregating the capacity of the flexible product's multiple execution modes, pooling successfully circumvents this temporary allocation shortcoming. While pooling generates almost exactly the same revenue given small upper error bounds, its revenue gain RG^{Pool} continues to rise with increasing δ when RG^{TA} already stagnates or declines. There is a difference of up to 1.5% (e.g. 1-M, $\delta = 1$). When individual demand streams are analyzed in detail, it turns out that temporary allocation and pooling accept exactly the same requests until about 60 days prior to departure. But as soon as the first resources become scarce, temporary allocation has to decline high-value requests for specific products because there is no remaining capacity on the required resources. Any requests for flexible products that occupy space on a scarce resource cannot be moved to other execution modes until the next reoptimization. In contrast, pooling does not need to wait for the reoptimization and can continue to accept requests for specific products as long as there is sufficient capacity and the revenue exceeds the corres-

ponding bid price. Thus, pooling accepts more requests for specific products near the end of the booking horizon. Although the number of accepted requests for flexible products is almost the same (see Table 3), the capacity utilization increases by up to 1%.

As mentioned before, the shifting mechanism nearly yields the same results as pooling. Only negligible improvements in five test cases (see 1-H and 1-M in Table 4) indicate its theoretical advantage. Although shifting allows rearranging accepted requests for flexible products whenever a new request for a specific product comes in, it still uses bid prices from the previous reoptimization to approximate the opportunity cost of accepting a request. Conversely, the CEC mechanism calculates two value functions for each request and thus further improves the revenue by 0.5% to 2.5%. An exception is 1-L, where demand is less than capacity. Here, CEC performs roughly as well as shifting or pooling, but also yields slightly less revenue in some test cases. The well-known trade-off between the maximization of capacity utilization and revenue can be demonstrated by comparing CEC with temporary allocation: For 1-H, for example, CEC yields strictly more revenue, but has lower capacity utilization than temporary allocation.



a) Medium demand (2-M) b) Low demand (2-L)

Fig. 6. Revenue gain in network 2

The analysis of network 2 confirms our results (see Fig. 6 for 2-M and 2-L, and Table 5 for all demand situations). Here, temporary allocation yields 0.25%–2% more revenue than the ad-hoc mechanism. Initially, the revenue gain rises but from a certain threshold onwards, it stays constant or decreases for higher upper error bounds δ. Until this threshold is reached, shifting leads to the same results, but as soon as temporary allocation ceases to improve, shifting – pooling is not applicable here – continues to increase its revenue gain with increasing δ. The maximal difference is 0.2% for 2-L and $\delta_1 = 1$. As before, CEC further improves the revenues,

in this network up to 1.25% over shifting. Overall, the impact of the controls – which mainly improve the handling of accepted requests for flexible products – is less significant in the more conservative network 2 because the ratio of flexible to specific demand is much smaller than in network 1. Only a subset of 9 of the 24 itineraries can be booked with flexible products.

Table 2
Optimality gap (OG), capacity utilization (CU), and number of requests for flexible products accepted ($\#\ flex.$) for the ad-hoc mechanism (network 1)

δ	1-H			1-M			1-L		
	OG	CU	# flex.	OG	CU	# flex.	OG	CU	# flex.
0	-5.39	99.35	48	-3.60	98.60	125	-2.62	91.72	111
0.25	-6.45	98.87	47	-4.44	98.04	126	-2.73	91.42	112
0.5	-9.12	97.32	61	-6.05	96.67	126	-3.47	91.03	112
0.75	-11.84	95.37	74	-7.97	94.82	126	-4.29	89.85	110
1	-14.98	94.08	88	-9.76	92.95	124	-5.59	88.72	112

Table 3
Optimality gap (OG), capacity utilization (CU), and number of requests for flexible products accepted ($\#\ flex.$) for the ad-hoc mechanism (network 2)

δ	2-H			2-M			2-L		
	OG	CU	# flex.	OG	CU	# flex.	OG	CU	# flex.
0	-3.46	98.51	113	-2.73	96.17	280	-1.50	89.62	271
0.25	-4.35	97.81	125	-3.39	95.52	281	-1.89	89.30	271
0.5	-6.56	96.52	148	-5.02	94.36	281	-3.00	88.24	268
0.75	-9.11	95.04	183	-6.53	93.03	278	-3.92	87.35	270
1	-11.49	93.56	206	-8.20	91.29	278	-4.79	86.29	263

Table 4
Revenue gain (RG), capacity utilization (CU), and number of requests for flexible products accepted ($\#\ flex.$) for temporary allocation, pooling, shifting, and CEC (network 1)

Class	δ	temporary allocation			pooling			shifting			CEC		
		RG	CU	# flex.	RG	CU	# flex.	RG	CU	# flex.	RG	CU	# flex.
1-H													
	0	0.67	99.47	45	0.69	99.48	45	0.68	99.47	45	3.11	98.75	19
	0.25	1.05	99.15	42	1.06	99.16	42	1.06	99.15	42	3.14	98.25	21
	0.5	2.52	98.06	48	2.52	98.07	48	2.52	98.06	48	4.33	97.09	30
	0.75	3.80	96.79	60	3.91	96.87	60	3.92	96.85	60	5.48	96.01	42
	1	4.65	95.81	73	5.24	96.10	70	5.27	96.05	71	6.95	95.41	55
1-M													
	0	1.02	99.03	120	1.02	99.03	120	1.02	99.03	120	1.87	99.03	120
	0.25	1.44	98.72	120	1.44	98.72	120	1.45	98.72	120	2.14	98.72	120
	0.5	2.33	97.92	119	2.36	97.94	119	2.36	97.94	119	2.83	97.92	119
	0.75	2.88	96.48	120	3.10	96.63	119	3.12	96.63	119	3.55	96.48	120
	1	2.80	94.52	120	4.24	95.50	120	4.27	95.50	120	4.57	94.52	120
1-L													
	0	1.88	93.23	111	1.88	93.23	111	1.88	93.23	111	1.84	93.14	111
	0.25	2.07	92.94	112	2.07	92.94	112	2.07	92.94	112	2.02	92.83	112
	0.5	2.43	92.68	112	2.48	92.70	112	2.48	92.70	112	2.40	92.58	112
	0.75	2.04	91.16	110	2.41	91.38	110	2.41	91.38	110	2.34	91.25	110
	1	1.20	89.44	112	2.66	90.34	112	2.66	90.34	112	2.70	90.33	111

Table 5
Revenue gain (RG), capacity utilization (CU), and number of requests for flexible products accepted ($\#\ flex.$) for temporary allocation, shifting, and CEC (network 2)

Class	δ	temporary allocation			shifting			CEC		
		RG	CU	# flex.	RG	CU	# flex.	RG	CU	# flex.
2-H										
	0	0.25	98.68	109	0.25	98.68	109	1.59	98.33	65
	0.25	0.53	98.05	118	0.53	98.05	118	1.63	97.72	79
	0.5	1.07	96.87	133	1.07	96.87	133	2.24	96.71	99
	0.75	1.56	95.54	164	1.58	95.55	164	2.96	95.57	130
	1	1.86	94.26	186	2.02	94.31	185	3.48	94.40	158
2-M										
	0	0.40	96.40	273	0.40	96.40	273	1.24	96.14	248
	0.25	0.68	95.90	273	0.68	95.90	273	1.34	95.65	249
	0.5	1.14	94.92	270	1.15	94.92	270	1.73	94.77	254
	0.75	1.54	93.76	269	1.56	93.77	269	2.21	93.75	256
	1	1.73	92.15	269	1.85	92.21	269	2.48	92.35	261
2-L										
	0	0.47	90.05	270	0.47	90.05	270	0.63	90.01	269
	0.25	0.62	89.74	270	0.62	89.74	270	0.77	89.72	268
	0.5	0.95	88.83	265	0.95	88.83	265	1.15	88.89	263
	0.75	1.13	88.00	266	1.18	88.02	266	1.40	88.10	263
	1	0.95	86.87	260	1.16	86.96	260	1.43	87.15	260

5 Summary & Conclusion

In this paper, we proposed and investigated several dynamic capacity control mechanisms for revenue management with flexible products, which allow for supply-side flexibility even after the time of sale. In our computational experiments, it turned out that total revenue can be significantly increased by these mechanisms compared to the "ad-hoc" approach used in the context of opaque products, which allocates resources immediately after the acceptance of requests. The mechanisms can be ordered according to the degree to which they make use of flexibility opportunities. In this context, it could be shown that the revenue gain when using a more sophisticated mechanism – which exploits the flexibility to a greater extent – is strongly dependent on the forecast quality as well as the proportion of flexible products on the network. While the simplest mechanism, the temporary allocation of resources over time between two optimizations, is sufficient for applications with a rather good forecast, advanced mechanisms like shifting and pooling are even more gainful if the forecast quality is rather poor. Further improvements could theoretically be reached by making use of CEC.

However, there is a general trade-off between the level of the mechanism's sophistication with the resulting revenue gain on the one hand and the practical applicability on the other. Exploiting supply-side flexibility makes high demands on the organizational environment, integration into existing software systems, degree of automation, and existing business

processes. In particular, it can be crucial if many requests have to be answered real-time in a semi-automated process during which permanent mathematical optimizations, which are, for example, required for the CEC mechanism, are impossible or at least undesirable. The proposed pooling and shifting mechanisms, which are heuristics formalizing intuitive ideas that can easily be comprehended and implemented, seem to be a reasonable trade-off between the revenue gain obtained by the CEC and the simplicity of a pure ad-hoc mechanism.

References

[1] Belobaba PP. Air travel demand and airline seat inventory management. Ph.D. Thesis, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, 1987.

[2] Talluri KT, Van Ryzin GJ. An analysis of bid-price controls for network revenue management. Management Science 1998;44(11):1577-93.

[3] Bertsimas D, Popescu I. Revenue management in a dynamic network environment. Transportation Science 2003;37(3):257-77.

[4] Smith BC, Penn CW. Analysis of alternative origin-destination control strategies. In: AGIFORS Annual Symposium Proceedings 28, New Seabury, 1988. p. 113-21.

[5] Williamson EL. Airline network seat control. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, 1992.

[6] Simpson RW. Using network flow techniques to find shadow prices for market and seat inventory control. Memorandum M89-1, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, 1989.

[7] Talluri KT, Van Ryzin GJ. A randomized linear programming method for computing network bid prices. Transportation Science 1999;33(2):207-16.

[8] De Boer SV, Freling R, Piersma N. Mathematical programming for network revenue management revisited. European Journal of Operational Research 2002;137(1):72-92.

[9] Adelman D. Dynamic bid-prices in revenue management. Operations Research 2007;55(4):647-61.

[10] Topaloglu H. Using Lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. Operations Research 2009;57(3):637-49.

[11] Bertsimas D, De Boer SV. Simulation-based booking limits for airline revenue management. Operations Research 2005;53(1):90-106.

[12] Van Ryzin GJ, Vulcano G. Simulation-based optimization of virtual nesting controls for network revenue management. Operations Research 2008;56(4):865-80.

[13] Klein R. Network capacity control using self-adjusting bid-prices. OR Spectrum 2007;29(1):39-60.

[14] Topaloglu H. A stochastic approximation method to compute bid prices in network revenue management problems. INFORMS Journal on Computing 2008;20(4):596-610.

[15] Gallego G, Iyengar G, Phillips R, Dubey A. Managing flexible products on a network. CORC Technical Report Tr-2004-01, IEOR Department, University of Columbia, 2004.

[16] Gallego G, Phillips R. Revenue management of flexible products. Manufacturing and Service Operations Management 2004;6(4):321-37.

[17] Petrick A, Steinhardt C, Gönsch J, Klein R. Using flexible products to cope with demand uncertainty in revenue management. Working Paper, Chair of Mathematical Methods, Institute of Statistics and Economic Theory, University of Augsburg, 2009.

[18] Bartodziej P, Derigs U. On an experimental algorithm for revenue management for cargo airlines. In: Ribeiro CC, Martins SL, editors. Proceedings of the 3rd International Workshop on Experimental and Efficient Algorithms (WEA), Lecture Notes in Computer Science, vol. 3059. Berlin: Springer, 2004. p. 57-71.

[19] Bartodziej P, Derigs U, Zils M. O&D revenue management in cargo airlines - a mathematical programming approach. OR Spectrum 2007;29(1):105-21.

[20] Kimms A, Klein R. Revenue Management im Branchenvergleich. Zeitschrift für Betriebswirtschaft 2005;75(Special Issue 1):1-30.

[21] Spengler T, Rehkopf S, Volling T. Revenue management in make-to-order manufacturing - an application to the iron and steel industry. OR Spectrum 2007;29(1):157-71.

[22] Kimms A, Müller-Bungart M. Revenue management for broadcasting commercials: the channel's problem of selecting and scheduling advertisements to be aired. International Journal of Revenue Management 2007;1(1):28-44.

[23] Talluri KT. Airline revenue management with passenger routing control: a new model with solution approaches. International Journal of Services Technology and Management 2001;2(1/2):102-15.

[24] Talluri KT, Van Ryzin GJ. The theory and practice of revenue management. New York: Springer, 2004.

[25] Chen VCP, Günther D, Johnson EL. Routing considerations in airline yield management. In: Ciriani TA, Fasano G, Gliozzi S, Tadei R, editors. Operations Research in Space and Air. Boston: Kluwer, 2003. p. 333-50.

[26] Jiang Y. Price discrimination with opaque products. Journal of Revenue and Pricing Management 2007;6:118-34.

[27] Bertsekas DP. Dynamic programming and optimal control, volume 1. 3rd edn, Belmont: Athena Scientific, 2005.

[28] Kimms A, Müller-Bungart M. Simulation of stochastic demand data streams for network revenue management problems. OR Spectrum 2007;29(1):5-20.

DOI: 10.1016/j.cor.2010.02.003
URN: urn:nbn:de:hbz:465-20220609-141921-3

This is the"Authors Accepted Manuscript"version of: Petrick, A.; Gönsch J; Steinhardt, C.; Klein, R. (2010) Dynamic ControlMechanisms for Revenue Management with Flexible Products.Computers & Operations Research, 37 (11), pp.2027-2039. The final article version is available at: https://doi.org/10.1016/j.cor.2010.02.003