European Journal of Operational Research 271 (2018) 179–192

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

Decomposition methods for dynamic room allocation in hotel revenue management

N. Aydin[a,*], S. I. Birbil[b]

[a] Warwick Business School, University of Warwick, Coventry CV4 7AL, United Kingdom
[b] Econometric Institute, Erasmus School of Economics, Erasmus University Rotterdam, 3000 DR, Rotterdam PO Box 1738, The Netherlands

ARTICLE INFO

Article history:
Received 19 June 2017
Accepted 15 May 2018
Available online 22 May 2018

Keywords:
Revenue management
Hotel
Capacity control
Decomposition methods

ABSTRACT

Long-term stays are quite common in the hotel business. Consequently, it is crucial for the hotel managements to consider the allocation of available rooms to a stream of customers requesting to stay multiple days. This requirement leads to the solving of dynamic network revenue management problems that are computationally challenging. A remedy is to apply decomposition approaches so that an approximate solution can be obtained by solving many simpler problems. In this study, we investigate several room allocation policies in hotel revenue management. We work on various decomposition methods to find reservation policies for advance bookings and stay-over customers. We also devise solution algorithms to solve the resulting problems efficiently.

1. Introduction

Historically, the airline industry played the steering role in revenue management (RM). Today, however, there is a wide range of applications in different industries with volatile demand, requesting fixed and perishable capacity (Kimes, 1989). Although the hotel industry is one of the typical application areas of revenue management, the research in this particular area lags behind the work produced for other service industries. In their recent work, Ivanov and Zhechev (2012) and Ivanov (2014) present a review of the methods proposed in the hotel RM literature and point out the gaps.

In general, well-known airline RM techniques, such as booking control and pricing, can be applied to hotel RM problems. However, it is important to consider several constraints that are unique to hotel reservation systems. First, multi-day stays in hotels are quite common. While a flight itinerary includes, on average fewer than three legs, the number of nights a typical customer spends in a hotel can be a week or even more (Zhang & Weatherford, 2017). Second, the demand process is different. Hotel customers may decide to stay longer and extend their reservation while they are staying in the hotel (Kimes, 1989). Third, airline customers generally make advance bookings but a number of hotel customers consist of walk-ins. Moreover, the early reservations in the booking interval are even allowed to cancel their bookings at no extra cost.

A reservation for a room spans several days, and hence, the number of available rooms in the hotel for any day is determined only by the reservations including that particular day. Therefore, the room allocation problem becomes the control of total room capacity when customer demand is characterized by the length of stay and the type of rooms. Although this setting leads to a special linear network structure, the dynamics of the problem are still quite challenging for analysis and optimization. The state space of the problem is the Cartesian product of daily capacities in the network. This is a well-known difficulty in network RM problems. As a remedy, approximation methods based on problem decomposition are frequently used. The main idea behind these methods is to partition the network problem into independent subproblems. This idea has been widely adopted for network airline RM problems (Talluri & van Ryzin, 2004). However, it is important to note that these decomposition methods risk undermining the network effects of the shared daily capacities among different products (reservation types). To improve the performances of the decomposition methods, recent studies focus on protecting the network information by defining time and capacity dependent bid-prices and consider their effects on the network (Aslani, Modarres, & Sibdari, 2013; Kunnumkal & Topaloglu, 2010; Zhang, 2011). Bid price control is a widely adopted strategy in the airline revenue management literature. It is used to assess the value of

* Corresponding author.
E-mail addresses: nursen.aydin@wbs.ac.uk (N. Aydin), birbil@ese.eur.nl (S.I. Birbil).

https://doi.org/10.1016/j.ejor.2018.05.027
0377-2217/© 2018 Elsevier B.V. All rights reserved.

allocating one capacity to a request. Basically, the bid price control sets a threshold price for each resource (e.g. flight leg, hotel day capacity). Then, a reservation request is accepted, only if its revenue exceeds the sum of the bid prices associated with the allocated resources. Vinod (2005) discusses that the bid price controls can be used as a price mechanism to determine the value of the product.

In this paper, we focus on the room allocation decisions for a hotel. The optimal policy to accept or reject an arriving customer can be obtained by analyzing the stochastic nature of the customer arrival process. In hotel reservation systems, the customers are classified as the *advance bookings*, the *stay-overs* and the *walk-ins*. While the advance bookings make room reservations before they arrive at the hotel, the walk-ins show up without any reservation. The stay-overs are the customers who ask for an extension for their reservations during their stay in the hotel. Recently, hotel reservation systems have started offering extended stay as an option due to high customer demand (Tepper, 2015). For instance, Priceline (2017) and Hotwire (2017) present “add-a-night” and “add to your stay” options to their existing customers. The arrival process of the advance bookings and walk-ins are similar. The only difference is that the walk-in customers arrive after the reservation period ends. However, the stay-over requests depend on the accepted advance bookings. To simplify our notation, we ignore the walk-in customers and formulate our problem by considering the advance bookings and the stay-overs. Then, we explain how one can easily incorporate the walk-in customers to our proposed models. To the best of our knowledge, the dynamic model of stay-over customers in a network setting has not been previously studied in the literature.

The research contributions in this paper come from the application and the analysis of two decomposition approaches. These are the day-based and the period-based decompositions. Our day-based decomposition is similar to the one proposed by Kunnumkal and Topaloglu (2010). We simplify their decomposition method and show that our proposed model provides a lower bound to their model. We set forth a dynamic model for the advance bookings and formulate a linear program for the problem. The resulting model is then solved with the constraint generation method. We also propose alternate approximate models, which provide upper and lower bounds on the optimal expected revenue of the original model. To manage the stay-over requests, one needs to keep track of the number of reservations in each booking type. A day-based method, however, decomposes the network problem into independent days, and this decomposition approach causes loss of information on the number of customers in each booking type. Our solution to this hindrance is a period-based decomposition method, which is an extension of another approach recently proposed by Birbil, Frenk, Gromicho, and Zhang (2014). First, we focus on the single-day stay-over problem, as the request for an additional night is the most frequently realized stay-over case in real-life (Talya, 2016). Though our model is different than the one set forth by Birbil et al. (2014), we successfully build on their decomposition idea. Second, we consider the multi-day stay-over problem and present a two-period approximation, which combines the pair-based decomposition with the deterministic linear programming approach. In period one, we observe the reservation activity of the advance booking customers. In period two, we take into account the stay-over requests of the customers whose bookings have been accepted. To test the performances of the proposed decomposition approaches, we conduct simulation experiments and compare our results with those obtained by several well-known models from the literature. Our computational study indicates that the proposed decomposition approaches are apt to effective room allocation in hotel RM.

2. Review of related literature

We begin by reviewing the related work on hotel RM. Then, we summarize the decomposition approaches frequently applied to the network RM problems.

Ladany (1976) works on a single-day stay model for a hotel with two types of resources. The aim of the model is to find an allocation policy to maximize the daily expected revenue. He develops a dynamic programming formulation and obtains the decision policy for each resource. Williams (1977) works on the single-day stay model during the peak demand period. In this model, he assumes that demand arrives from three different sources: the stay-overs, the reservations and the walk-ins. He computes the reservation policy for each customer type by comparing the costs of underbooking and overbooking. Bitran and Leong (1989) focus on the multi-day problem by considering the walk-in and stay-over requests. They model the multi-day reservations as a series of independent, single-day reservations. Bitran and Mondschein (1995) develop a dynamic programming model for a single-day stay problem with multiple products. Since the resulting model is computationally intractable for the real size problems, they utilize several heuristics when searching for the optimal allocation policy. Weatherford (1995) focuses on the effect of the length of stay. He proposes a heuristic method based on a static model and compares this method with the other booking policies developed for the single-day stay problems. Bitran and Gilbert (1996) work on a single-day stay and single-room problem. They assume that during the service day, three types of customers show-up: the customers with guaranteed reservations, the customers with reservations and the walk-ins. They develop a dynamic model and propose a heuristic method to obtain the room allocation policy. Baker and Collier (1999) extend the study of Weatherford (1995) as well as the work of Bitran and Mondschein (1995) by allowing cancellations, overbooking and stay-overs. They develop two heuristics that integrate overbooking with the capacity allocation decisions. They compare the performances of these heuristics against the other booking control policies in the literature. Through this comparison, Baker and Collier (1999) discuss the advantages of each policy under different operating environments.

Later studies focus on multi-product and multi-day stay problems. Chen (1998) presents a general formulation for a deterministic problem and discusses that it can be transferred to a network flow problem. Moreover, he shows that the optimal solution of the linear program is always integral. Goldman, Freling, Pak, and Piersma (2002) propose deterministic and stochastic linear programming models to find the nested booking limits and the bid prices for the multi-day stay problem. They follow the work of Weatherford (1995) to develop the deterministic model. For the stochastic model, they extend the work of De Boer, Freling, and Piersma (2002) on the airline revenue management problem. However, unlike the models proposed by Weatherford (1995) and De Boer et al. (2002), they use the booking control policies over a rolling horizon of decision periods. Lai and Ng (2005) work on a stochastic programming formulation for a multi-day stay problem. They apply robust optimization techniques to solve the problem on a scenario basis. They also consider the risk aversion of the decision maker and use the mean absolute value to measure the revenue deviation risk. Koide and Ishii (2005) work on the optimal room allocation policies for a single-day stay by considering the early discounts, the cancellations and the overbookings. They examine the properties of the expected revenue function and show that it is unimodal on the number of allocated rooms for early discount and overbooking. As with Lai and Ng (2005), Liu, Lai, and Wang (2008) present revenue optimization models for a multi-day stay problem by considering the revenue risk. They propose a stochastic programming model with semi-absolute deviations to measure the risk. Guadix, Cortes, Onieva, and Munuzuri (2010) present a decision support system for forecasting and room allocation decisions. They work on the deterministic and stochastic programming models by considering group arrivals. The proposed decision support system integrates these models for room allocation and pricing decisions. Nadarajah, Lim, and Ding (2015) study dynamic pricing policy for a single type of room by considering the multiple day stays. Since the resulting model is computationally intractable, they propose pricing heuristics based on fluid approximation and approximate linear programming. They analyze the properties of the pricing policy under the peak demand.

The solution approaches considered in this study build on the literature on decomposition methods in network revenue management. The output of a decomposition method is used to construct various capacity controls, such as bid-prices and nested booking limits. Adelman (2007) develops an approximation method to compute the dynamic bid prices. He first formulates the network problem as a dynamic model, which suffers from the curse of dimensionality. Thus, he derives a standard linear program by approximating the dynamic programming value functions. This approach provides an upper bound on the optimal expected revenue. Zhang (2011) proposes a nonlinear, non-separable approximation to the dynamic programming model that leads to a tighter upper bound. Topaloglu (2009) focuses on a Lagrangian relaxation method to decompose the network problem into many single capacity problems. Erdelyi and Topaloglu (2009) work on the overbooking problem in an airline network and develop separable approximations to decompose the problem by individual flights (legs). This approach constructs capacity dependent bid prices. However, it becomes quite difficult to compute the value functions for each leg as the size of the problem increases. To reduce the computational burden, Kunnumkal and Topaloglu (2011) develop a stochastic approximation algorithm that provides a set of capacity independent bid prices. In this approach, they formulate the total expected profit as a function of the bid prices and use stochastic gradients to obtain a good bid price policy. Recently, Kunnumkal and Topaloglu (2010) propose a new leg-based decomposition method for the airline revenue management problems that involve the customer choices. In this method, they first allocate the revenue of each itinerary among the legs covered by the itinerary. Then, they define a penalty term to incorporate the network effect. They view the revenue allocations and the penalty terms as decision variables, and use subgradient search to find the optimal solution. Although this solution approach is manageable in small size networks, it can be impractical for the problems of substantial size networks. Hotel network revenue management problems are also tackled with the decomposition methods. Zhang and Weatherford (2017) work on a dynamic pricing problem. They generalize the approximation method of Zhang (2011) and decompose the problem into independent single-day problems by approximating the value functions with nonlinear non-separable functions. They test the proposed approach by using the data from a hotel. Aslani et al. (2013) also propose a decomposition method for a pricing problem in hotel revenue management. They develop an approach to estimate the effective arrival rate for each day by considering the stock-outs and the customer losses due to high price levels. They decompose the network problem into single-day subproblems by using these daily arrival rates. Our study has several distinguishing features compared with the earlier work. To begin with, we focus on the multiple day problem and propose several decomposition methods to attack the problem. In particular, our aim is to find a dynamic capacity allocation policy that takes into account the advance bookings and the stay-over customers. We first study advance bookings and propose day-based decomposition methods. We work on a fare-allocation strategy where the reservation fares are allocated on day basis depending on the time of the booking. Our method is based on dynamic programming formulations for the single-day revenue management problems, hence it can capture the temporal dynamics of the reservation requests more accurately compared with the static models. We also present alternate solution methods to improve the computational time for the large-scale problems. Later, we study stay-over requests in hotel RM and propose a pair-based dynamic programming method. To the best of our knowledge, the dynamic model of stay-over customers in a network setting has not previously been studied in the literature. We also discuss the applicability of the proposed models to several cases, such as late checkout and overbooking. Finally, our computational experiments demonstrate that the proposed methods can generate significantly higher profits than the well-known benchmarks in the literature. The performance gaps are especially significant when the daily hotel capacity is tight and the stay-over probability is high. In addition, day-based decomposition methods perform significantly better when the hotel controls the fares on a per-day basis and does not offer discount for long-term stays.

3. Problem definition

We consider the reservation process for a hotel. Customers may book for a single day or for up to n consecutive days. We call the combination of a check-in and a check-out day a *pair*. As there are different rooms in a hotel, the customers pay different prices. Therefore, a combination of a price and a pair becomes a *product*.

In our formulation, the planning horizon consists of two intervals: the reservation interval and the service interval. The state of the system changes with two types of events. The first type is a booking request for a product. While the advance booking requests arrive in the reservation interval, the walk-in customers show up without any reservation in the service interval. The second type of event is a booking extension request that also occurs in the service interval. To simplify our notation, we ignore the walk-in customers and formulate our problem by considering the advance bookings and stay-overs. Later, we explain how we can incorporate the walk-in customers into our proposed models.

Fig. 1 shows the structure of a hotel network with multiple-day stays. Each node in the service interval corresponds to a day. We index the service days by i, the products by k and the pairs by s. The bold line in the figure shows product k associated with pair $s = (2, n)$. The dashed lines show the service days and their capacities. The capacity of a hotel may vary from day to day and, hence, we represent the capacity on day i by C_i. The set of service days in the hotel network is denoted by $\mathcal{L}$. Again using capital letters for sets, we denote the set of products and the set of pairs by $\mathcal{K}$ and $\mathcal{S}$, respectively. We use the operator $|.|$ when we refer to the cardinality of a set.

We divide the reservation interval into τ periods and define the set $\mathcal{T} = \{1, 2, \ldots, \tau\}$. At each period $t \in \mathcal{T}$, a reservation request for product k arrives with probability p_{kt}, and we need to decide whether to accept or reject this booking. If we accept a reservation request for product k, then we generate a revenue of f_k. Each accepted product k request consumes one unit of capacity from those days spanned by the product. We assume that there are no reservations in the system at time $t = 0$ and *at most one* reservation request arrives at each time period. Moreover, in the service interval we again observe the reservation requests for the pairs. However, this time the pairs correspond to the various days of the stay-over requests. We assume that each accepted booking of pair s, independently of other bookings, may request an extension corresponding to pair j at the beginning of its check-out day with probability q_{sj}. If the stay-over request for extension j of pair s is met, then a revenue of θ_{sj} is generated.

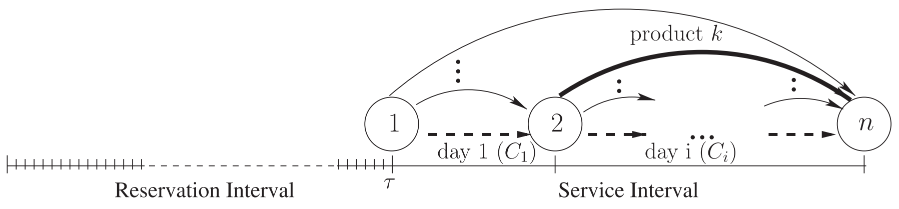


Fig. 1. The hotel network with multiple time intervals.

This model is a network capacity allocation problem, and hence, solving the complete model is, in general, not possible (Talluri & van Ryzin, 2004). In the subsequent part, we discuss two decomposition approaches to approximate the problem. First, we ignore the stay-over requests and work on the approximation methods for the dynamic model only with the advance bookings. Then, we discuss how to incorporate the stay-over requests and develop an approximate model for the resulting problem.

4. Mathematical models without stay-overs

In this section, we study the hotel capacity allocation problem without the stay-over requests and consider only the advance bookings. To give a dynamic programming formulation for this problem, we need to store the number of reservations received each day. We use z_{it} to denote the total number of reservations for day i in period t. Then, the vector $\mathbf{z}_t = (z_{it})_{i\in\mathcal{L}}$ shows the state in our dynamic programming model. We define a_{ik} to indicate whether product k customer occupies one unit from C_i. In other words, if day i is used by product k, then $a_{ik} = 1$; otherwise, $a_{ik} = 0$. We also define $\mathbf{e}_i$ as an $|\mathcal{L}|$-dimensional unit vector with a one in the element corresponding to day i.

Let $J_t(\mathbf{z}_t)$ denote the expected optimal revenue from t up to τ given that the state of the reservations at the beginning of time period t is $\mathbf{z}_t$. For every $1 \le t \le \tau$, we can find the optimal policy by computing the value functions through the optimality equation

$$J_t(\mathbf{z}_t) = \sum_{k\in\mathcal{K}} p_{kt} \max\left\{ f_k + J_{t+1}(\mathbf{z}_t + \sum_{i\in\mathcal{L}} a_{ik}\mathbf{e}_i), J_{t+1}(\mathbf{z}_t) \right\} + \left(1 - \sum_{k\in\mathcal{K}} p_{kt}\right) J_{t+1}(\mathbf{z}_t), \tag{1}$$

where the boundary condition is $J_{\tau+1}(\mathbf{z}_t) = 0$ for all $\mathbf{z}_t$. This model is still intractable even for the small-sized networks. To overcome this difficulty, we shall consider several approximation methods based on decomposition.

4.1. Day-based decomposition

We focus on splitting the service interval into n days. The fundamental idea behind this approach is to decompose the hotel network into independent single days and solve each subproblem to obtain the decision policy for the overall network problem. To decompose the network, we distribute the revenue of each product to the days that it uses. First, we define a day-based fare α_{ikt}, $i \in \mathcal{L}, k \in \mathcal{K}, t \in \mathcal{T}$ for each product. Then, we impose the constraints

$$\sum_{i\in\mathcal{L}_k} \alpha_{ikt} = f_k, \quad k \in \mathcal{K}, t \in \mathcal{T}, \tag{2}$$

where $\mathcal{L}_k$ denotes the set of days spanned by product k. This condition on the product revenues guarantees that if we make the same decision for a product at each subproblem, then the overall revenue obtained from that product is equal to the actual fare. We explain how to obtain revenue allocations, α_{ikt}, later in this section.

When we accept a booking request for product k at time period t on day i, we generate a revenue of α_{ikt} and use one unit from the capacity of day i. If we define $\alpha = \{\alpha_{ikt} : k \in \mathcal{K}, i \in \mathcal{L}_k, t \in \mathcal{T}\}$, then we can formulate the single-day problem as a dynamic program with the recursive relation

$$V_{it}(z_{it}|\alpha) = \sum_{k\in\mathcal{K}} p_{kt} \max\{\alpha_{ikt} + V_{i,t+1}(z_{it} + a_{ik}|\alpha), V_{i,t+1}(z_{it}|\alpha)\} + \left(1 - \sum_{k\in\mathcal{K}} p_{kt}\right) V_{i,t+1}(z_{it}|\alpha), \tag{3}$$

for every $1 \le t \le \tau$. The boundary conditions simply become $V_{it}(C_i|\alpha) = 0$ and $V_{i\tau+1}(z_{it}|\alpha) = 0$ for all $z_{it} = 1, \dots, C_i - 1$. Kunnumkal and Topaloglu (2010) show that this type of approximation provides an upper bound on the optimal expected revenue for the customer choice model. The next result follows similar steps that are given by Kunnumkal and Topaloglu (2010) to establish an upper bound for the original dynamic program in (1). The proof of the proposition is presented in the supplementary document.

Proposition 4.1. *For all $z_{it} \le C_i$ and $1 \le t \le \tau$, we have $J_t(\mathbf{z}_t) \le \sum_{i\in\mathcal{L}} V_{it}(z_{it}|\alpha)$ such that α satisfies condition (2).*

Since $\sum_{i\in\mathcal{L}} V_{i1}(0|\alpha)$ gives an upper bound on $J_1(\mathbf{0})$, we can obtain the tightest bound by solving the following problem

$$\text{minimize} \sum_{i\in\mathcal{L}} V_{i1}(0|\alpha) \tag{4}$$

$$\text{subject to} \sum_{i\in\mathcal{L}_k} \alpha_{ikt} = f_k, \quad k \in \mathcal{K}, t \in \mathcal{T}, \tag{5}$$

where $\{\alpha_{ikt} : i \in \mathcal{L}_k, k \in \mathcal{K}, t \in \mathcal{T}\}$ are the decision variables. As a direct consequence of Proposition 4.1, the optimal objective function value of problem (4)–(5) provides an upper bound on the maximum expected revenue over the whole planning horizon. Moreover, it gives the optimal values of the revenue distribution $\{\alpha^*_{ikt},\ i \in \mathcal{L}, k \in \mathcal{K}, t \in \mathcal{T}\}$ that we can use to construct a control policy for each single-day dynamic programming model (3). The reservation policy for the network problem can be summarized as follows: Given the allocations $\{\alpha_{ikt},\ i \in \mathcal{L}, k \in \mathcal{K}, t \in \mathcal{T}\}$ and the state variables $\{z_{it},\ i \in \mathcal{L}\}$, we accept a reservation request for product k at time period t, if we have

$$f_k \ge \sum_{i\in\mathcal{L}_k} (V_{it+1}(z_{it}|\alpha) - V_{it+1}(z_{it} + 1|\alpha)). \tag{6}$$

We next reformulate problem (4)–(5) as a tractable linear program. This derivation also shows simply that (4)–(5) is a convex optimization problem. To rewrite the recursion let us first define

$$x_{itz} = V_{it}(z_{it}|\alpha) - V_{i,t+1}(z_{it}|\alpha). \tag{7}$$

By using the fact that $V_{i\tau+1}(z_{it}|\alpha) = 0$ for all $z_{it} = 1, \dots, C_i - 1$, we have $V_{it}(z_{it}|\alpha) = \sum_{l=t}^{\tau} x_{ilz}$. Then, we can rewrite (3) as

$$x_{itz} = \sum_{k \in \mathcal{K}} p_{kt} \max \left\{ \alpha_{ikt} + \sum_{l=t+1}^{\tau} (x_{i,l,z+1} - x_{ilz}), 0 \right\} \quad (8)$$

for every $1 \le t < \tau$, and

$$x_{k\tau} := \sum_{k \in \mathcal{K}} p_{kt} \max\{\alpha_{ik\tau}, 0\}. \quad (9)$$

x_{itz} can be interpreted as the expected opportunity cost of holding a seat from period t to period $t+1$. Let us define u^i_{ktz} to denote the decisions for product k at time period t on day i. Then, the set of feasible decisions at time period t for day i is given by

$$\mathcal{U}_{it}(z_{it}) = \{u^i_{ktz} \in \{0,1\} : z_{it} + u^i_{ktz} \le C_i, \quad k \in \mathcal{K}\}.$$

Note that the total number of accepted customers on day i at time period t is at most $\min\{C_i, t\}$. Let κ_t denote this bound on the total number of reservations. Together with (8) and (9), we have to introduce, for each $u^i_{ktz} \in \mathcal{U}_{it}(z_{it})$, $i \in \mathcal{L}$, $t = 1, \dots, \tau - 1$ and $z = 1, \dots, \kappa_t - 1$, the following constraint into our linear programming problem:

$$x_{itz} \ge \sum_{k \in \mathcal{K}} p_{kt} \left(\alpha_{ikt} + \sum_{l=t+1}^{\tau} (x_{i,l,z+1} - x_{ilz}) \right) u^i_{ktz}.$$

The linear programming formulation of problem (4)–(5) is then given by

$$\text{minimize} \sum_{i \in \mathcal{L}} \sum_{t \in \mathcal{T}} x_{it0} \quad (10)$$

$$\text{subject to } x_{itz} \ge \sum_{k \in \mathcal{K}} p_{kt} (\alpha_{ikt} + \sum_{l=t+1}^{\tau} (x_{i,l,z+1} - x_{ilz})) u^i_{ktz},$$
$$u^i_{ktz} \in \mathcal{U}_{it}(z_{it}), i \in \mathcal{L}, t = 1, \dots, \tau - 1, z = 0, \dots, \kappa_t - 1, \quad (11)$$

$$x_{i\tau z} \ge \sum_{k \in \mathcal{K}} p_{kt} \alpha_{ik\tau} u^i_{k\tau z}, \quad u^i_{k\tau z} \in \mathcal{U}_{i\tau}(z_{i\tau}), \quad i \in \mathcal{L}, z = 1, \dots, C_i - 1, \quad (12)$$

$$x_{itC_i} = 0, \qquad i \in \mathcal{L}, t \in \mathcal{T}, \quad (13)$$

$$\sum_{i \in \mathcal{L}_k} \alpha_{ikt} = f_k, \qquad k \in \mathcal{K}, t \in \mathcal{T}, \quad (14)$$

$$x_{itz} \ge 0, \qquad i \in \mathcal{L}, t \in \mathcal{T}, z = 0, \dots, \kappa_t. \quad (15)$$

Here, constraints (11)–(13) follow from the dynamic programming recursion given in (3). While constraints (11) and (12) correspond to the recursive relations from $1 \le t \le \tau$, constraints (13) correspond to the boundary condition, $z = C_i$. The LP model given by (10)–(15) can be solved by using the constraint generation method (Ruszczynski, 2006). Note that for given $i \in \mathcal{L}$, $t = 1, \dots, \tau - 1$ and $z = 1, \dots, \kappa_t - 1$, the optimal value of x_{itz} is greater than the right-hand side of constraints (11) and (12) for all values of u^i_{ktz}. Therefore, in the master problem, we relax constraints (11) and (12) for some values of u^i_{ktz}. Then, given the optimal values $\{x^*_{itz} : i \in \mathcal{L}, t \in \mathcal{T}, z = 0, \dots, \kappa_t - 1\}$ and $\{\alpha^*_{ikt} : i \in \mathcal{L}, k \in \mathcal{K}, t \in \mathcal{T}\}$ to the current master problem, we check whether this solution violates any one of the constraints in (11) and (12) by solving

$$\max_{u^i_{ktz} \in \mathcal{U}_{it}(z_{it})} \left\{ \sum_{k \in \mathcal{K}} p_{kt} (\alpha^*_{ikt} + \sum_{l=t+1}^{\tau} (x^*_{i,l,z+1} - x^*_{ilz})) u^i_{ktz} \right\}$$
$$\text{for all } i \in \mathcal{L}, t \in \mathcal{T}, z = 0, \dots, \kappa_t - 1. \quad (16)$$

If the optimal value of this problem exceeds x^*_{itz}, then the corresponding constraint is added to the current master problem. The master problem is resolved with the added constraint. Notice that subproblem (16) can be solved very quickly as it has an analytical solution.

Remark 4.1. As we have mentioned in Section 1, we do not consider the walk-in customers in our problem formulation. However, the arrival process of the advance bookings and the walk-ins are similar. The only difference is that the walk-ins arrive during the service period. By considering the arrival of the walk-in customers as new product requests with higher prices, we can easily extend the dynamic model (1) to include the walk-ins. In this new problem, the number of products shall be doubled. By updating the product set, the fares and the probabilities, we can use the day-based methods to obtain the decision policy for the room allocation problem with the walk-ins and the advance bookings.

Some readers would notice that Kunnumkal and Topaloglu (2010) propose a similar revenue allocation model for the customer choice problem in airline revenue management. However, in addition to the fare allocation constraint (2), they also introduce the set of constraints

$$\sum_{i \in \mathcal{L}_k} \beta_{ikt} = 0 \quad k \in \mathcal{K}, t \in \mathcal{T}, \quad (17)$$

where the penalty terms $\{\beta_{ikt} : i \in \mathcal{L}_k, k \in \mathcal{K}, t \in \mathcal{T}\}$ for each product are introduced to coordinate the network decisions. These conditions on penalties guarantee that if we make the same decision for a product at each subproblem, then the overall penalty for that product is equal to zero. In other words, if we accept a reservation request of product k for each day it uses, then the collected revenue is $\sum_{i \in \mathcal{L}_k} \alpha_{ikt} = f_k$ and the collected penalty is $\sum_{i \in \mathcal{L}_k} \beta_{ikt} = 0$. Kunnumkal and Topaloglu (2010) discuss how using these penalties with allocated revenues provides a tighter upper bound on the optimal expected revenue. If we define penalty variables in our problem, then an accepted product k request generates a net revenue $\alpha_{ikt} + \beta_{ikt}$ on day i at time period t. In this case, we replace α_{ikt} with $\alpha_{ikt} + \beta_{ikt}$ in constraints (11) and (12) and we add the equality (17) into model (10)–(15) as a new constraint. For ease of expression, we define this new model as $LP_{(\bar{\alpha},\bar{\beta})}$ and model (10)–(15) as LP_α. Next, we discuss that introducing the penalty terms along with constraints (17) do not improve the bound on the optimal expected revenue. This observation leads to our current model that is simpler than the one given by Kunnumkal and Topaloglu (2010).

Lemma 4.1. *Let $LP^*_{(\bar{\alpha},\bar{\beta})}$ and LP^*_α be the optimal values of models $LP_{(\bar{\alpha},\bar{\beta})}$ and LP_α, respectively. Then, we have $LP^*_\alpha \le LP^*_{(\bar{\alpha},\bar{\beta})}$.*

The proof of Lemma 4.1 is given in the supplementary document. We conclude this subsection with two additional approximation approaches. We shall also test the performances of these approaches in our computational study (Section 6).

4.1.0.1. Time-independent fare allocation. To decrease the problem size, we propose an alternate model by assuming that the fare allocations are time independent. In other words, we replace the decision variables α_{ikt} with α_{ik} for each time period t in model (10)–(15). Clearly, the optimal values of the fare allocations for the time independent problem also provide a feasible solution for model (10)–(15). Therefore, it also gives an upper bound on the original dynamic programming model (1). More importantly, the problem size becomes drastically smaller.

4.1.0.2. Deterministic fare allocation. Deterministic linear programming is a well-known approximation method used to compute the optimal capacity control policy for the network revenue management problems. We discuss the deterministic model proposed for the hotel capacity allocation problem in the supplementary document. By using the deterministic model (41)–(44) given in the supplementary document, we can obtain a fare allocation policy. In order to do this, we decompose model (41)–(44) by days in the service interval. Topaloglu (2012) proposes a method to decompose the network of airline alliances by the airlines. Inspired by this approach, we define a fictitious day ζ with infinite capacity and the set of service days becomes $\mathcal{L}\cup\{\zeta\}$. To rewrite the model, we also define ν_{ik} denoting the reserved capacity for product k on day i. Then, problem (41)–(44) becomes

$$\text{maximize} \sum_{k\in\mathcal{K}} f_k \nu_{\zeta k} \tag{18}$$

$$\text{subject to} \sum_{k\in\mathcal{K}} a_{ik}\nu_{ik} \le C_i, \qquad i\in\mathcal{L}, \tag{19}$$

$$\nu_{ik} \le \sum_{t\in\mathcal{T}} p_{kt}, \qquad i\in\mathcal{L}, k\in\mathcal{K}, \tag{20}$$

$$\nu_{\zeta k} - \nu_{ik} = 0, \qquad i\in\mathcal{L}, k\in\mathcal{K}, \tag{21}$$

$$\nu_{ik} \ge 0, \qquad i\in\mathcal{L}, k\in\mathcal{K}. \tag{22}$$

It is easy to see the equivalence between problems (41)–(44) and (18)–(22). Let $\{\hat{\alpha}_{ik} : i\in\mathcal{L}, k\in\mathcal{K}\}$ be the optimal values of the dual variables associated with constraints (21). In the dual of problem (18)–(22), we have the constraint $\sum_{i\in\mathcal{L}_k}\hat{\alpha}_{ik} = f_k$ associated with the decision variable $\nu_{\zeta k}$. Therefore, we can use the dual variables $\hat{\alpha}_{ik}$ to obtain a fare allocation for product k.

4.2. Pair-based decomposition

Recently, Birbil et al. (2014) have proposed an origin-destination (OD-pair) based decomposition method for network airline revenue management problems. The proposed approach consists of two stages. In the first stage, the network capacities are allocated to each OD-pair, and in the second stage, a seat allocation policy is determined within each OD-pair. In this way, one only needs to solve a single-leg problem for each pair.

In this section, we present a similar method that decomposes the dynamic program (1) by the pairs. Recall that the combination of a check-in and a check-out day produces a pair in our hotel network. Our objective with this decomposition is to find the optimal room allocation policy for each check-in and check-out pair. Let y_s be the allocated capacity for pair s. If we accept the arriving request, then we generate a revenue of f_k and use one unit of the allocated capacity on pair s. With this notation, we are ready to formulate the problem. Let $\varrho_s^t(y_s)$ denote the expected optimal revenue of pair s from t up to τ given that the remaining capacity is y_s at time period t. Then, the overall pair-based decomposition model becomes

$$\begin{array}{lll} \text{maximize} & \sum_{s\in\mathcal{S}} \varrho_s^1(y_s) & \\ \text{subject to} & \sum_{s\in\mathcal{S}} a_{is} y_s \le C_i, & i\in\mathcal{L}, \\ & y_s\in\mathbb{Z}_+, & s\in\mathcal{S}, \end{array} \tag{23}$$

where $a_{is}=1$, if pair s involves day i; otherwise, $a_{is}=0$. In this problem, the objective function $\varrho_s^t(y_s)$ for a given y_s is an optimization problem itself, and its dynamic programming recursion is given by

$$\varrho_s^t(y_s) = \sum_{k\in\mathcal{K}_s} p_{kt}\max\{f_k + \varrho_s^{t+1}(y_s-1), \varrho_s^{t+1}(y_s)\} + \left(1-\sum_{k\in\mathcal{K}_s} p_{kt}\right)\varrho_s^{t+1}(y_s), \tag{24}$$

where $\mathcal{K}_s$ denotes the set of products in pair s. The boundary conditions are $\varrho_s^t(0)=0$ for all t and $\varrho_s^{\tau+1}(y_s)=0$ for all $y_s\ge 0$. It computes the optimal seat allocation policy for each pair $s\in\mathcal{S}$ with the objective of maximizing revenue. Note that the dynamic nature of the overall problem is not exactly preserved, since the capacity is allocated to the pairs prior to obtaining the optimal booking policy for each pair. However, after the capacity is set for a pair, the responses to customer requests for that pair are dynamic in the sense that the decisions change with the remaining time and the capacity. Therefore, the resulting model is partially dynamic (Birbil et al., 2014).

Lippman and Stidham (1977) show that the objective function $y_s \mapsto \varrho_s^t(y_s)$ is discrete concave. Therefore, we can always replace it by a piece-wise linear concave function and relax the integrality constraints. Then, the overall problem can be written as a linear programming problem. The optimal solution of this linear programming model may not be integer. Simply by rounding down the non-integer allocation, we can obtain a feasible solution. We refer the reader to Birbil et al. (2014) for the details. Next, we present that problem (23) provides a *lower bound* on the maximum expected revenue over the whole planning horizon. The proof of the proposition is given in the supplementary document.

Proposition 4.2. *The optimal objective value of problem (23) gives a lower bound on the optimal expected revenue of the dynamic programming model given in (1). That is, we have* $\sum_{s\in\mathcal{S}}\varrho_s^1(y_s) \le J_1(\mathbf{0})$.

As we have expressed in Proposition 4.1, day-based approximation given in (3) provides an upper bound on the dynamic model (1). Therefore, we obtain an upper bound on the optimality gap by using the Propositions 4.1 and 4.2 together. This bound is useful to evaluate the error committed by solving the approximate models instead of the complete model.

5. Mathematical models with stay-overs

To formulate the stay-over customers, we need to keep track of the accepted reservations in each pair. However, the day-based decomposition discussed in Section 4.1 only uses the total number of reservations in each day. For instance, a customer staying for two days is considered as two different reservations in the state space. Therefore, when we compute the number of stay-over requests, we may evaluate a customer as two different requests. Moreover, since the stay-over probability is reservation dependent, we have to approximate the probability to apply the day-based decomposition. As a result, the day-based methods may perform poorly for the stay-over problem. Since the pair-based decomposition breaks down the problem into pairs, it is more applicable for the problem with stay-overs.

5.1. Single-day stay-over

We first consider the case where only one day stay-over extension is possible. Our motivation for having a separate model for the single-day stay-over problem comes from the hotel industry. The majority of the requests for a stay-over is for an additional night (Talya, 2016). For instance, the popular hotel reservation application HotelTonight has launched a specific service called HotelTonight+1 to “a large portion of customers who switch hotels between single-night stays (Tepper, 2015).”

Moving forward to a stay-over model, we introduce x_s as the total number of reservations accepted for pair s. Then, the vector $\mathbf{x}=[x_s]_{s\in\mathcal{S}}$ forms the state of reservations. We let $a_{ks}=1$, if product k is in pair s; otherwise, $a_{ks}=0$. We define $\mathbf{e}_s$ as an $|\mathcal{S}|$-dimensional unit vector with a one in the element corresponding to pair s. Let $v_t(\mathbf{x})$ denote the expected optimal revenue from t up to τ given that the state of the reservations at the beginning of time period t is $\mathbf{x}$. For every $1 \le t \le \tau$, we can find the optimal policy by computing the value functions through the optimality equation

$$v_t(\mathbf{x}) = \sum_{k\in\mathcal{K}} p_{kt} \max\left\{ f_k + v_{t+1}(\mathbf{x} + \sum_{s\in\mathcal{S}} a_{ks}\mathbf{e}_s), v_{t+1}(\mathbf{x}) \right\} + \left(1 - \sum_{k\in\mathcal{K}} p_{kt}\right) v_{t+1}(\mathbf{x}). \tag{25}$$

The boundary condition comes from the revenue obtained by accepting stay-over requests. We introduce j_s to represent the day requested by pair s for the stay-over. Assuming that the number of reservations for pair s is x_s, $\mathbf{B}(x_s)$ denotes the number of requests by pair s for stay-over on day j_s. Given the assumption that the stay-over requests of different bookings are independent, $\mathbf{B}(x_s)$ is a binomial random variable with success probability q_{sj_s} and the number of trials x_s. We use ϑ_{j_s} to denote the number of accepted stay-overs for day j_s requested by pair s. We can compute the random revenue associated with the accepted stay-over requests by solving the problem.

$$\Pi(\mathbf{B}(\mathbf{x})) = \text{maximize} \sum_{s\in\mathcal{S}} \theta_{sj_s}\vartheta_{j_s} \tag{26}$$

$$\text{subject to} \sum_{s\in\mathcal{S}} (a_{is}x_s + a_{ij_s}\vartheta_{j_s}) \le C_i, \quad i \in \mathcal{L}, \tag{27}$$

$$\vartheta_{j_s} \le \mathbf{B}(x_s), \quad s \in \mathcal{S}, \tag{28}$$

$$\vartheta_{j_s} \in \mathbb{Z}_+, \quad s \in \mathcal{S}. \tag{29}$$

Constraints (27) ensure that the advance bookings and stay-over requests that we accept do not exceed the hotel capacity. Constraints (28) guarantee that the number of accepted stay-over requests for day j_s do not exceed admitted reservations in pair s that requested a one-day extension. Then, the boundary condition is given by $v_{\tau+1}(\mathbf{x}) = \mathbb{E}[\Pi(\mathbf{B}(\mathbf{x})]$. Unfortunately, this model is intractable because the state variables may involve many dimensions in practical applications and, hence, solving the complete dynamic model becomes computationally intractable. Moreover, evaluating the boundary condition is rather impractical due to the involved distribution of the random revenue $\Pi(\mathbf{B}(\mathbf{x}))$. Next, we describe an approximate method that can be used to jointly make the capacity control and stay-over decisions.

In our approximate model, we again use y_s as the allocated capacity for pair s. We also define w_{j_s} as the reserved capacity for stay-over requests of pair s. Let $\phi_s^t(x_s; \eta_s)$ be the optimal expected revenue for pair s with x_s guests in the system and the total reserved capacity from period t to τ is $\eta_s := y_s + w_{j_s}$. Then, the optimization problem becomes

$$\text{maximize} \sum_{s\in\mathcal{S}} \phi_s^1(0; \eta_s) \tag{30}$$

$$\text{subject to} \sum_{s\in\mathcal{S}} (a_{is}y_s + a_{ij_s}w_{j_s}) \le C_i, \quad i \in \mathcal{L}, \tag{31}$$

$$w_{j_s} \le y_s, \quad s \in \mathcal{S}, \tag{32}$$

$$\eta_s = w_{j_s} + y_s, \quad s \in \mathcal{S}, \tag{33}$$

$$y_s, w_{j_s} \in \mathbb{Z}_+, \quad s \in \mathcal{S}. \tag{34}$$

The first set of constraints ensures that the total number of reservations that we accept as the advance bookings and the stay-overs do not exceed the hotel capacity on each day. The second set of constraints ensures that only the reservations that have been admitted can request stay-overs.

Formally, the objective function (30) is given for $t = 1, \dots, \tau$ with the dynamic programming recursion

$$\phi_s^t(x_s; \eta_s) = \sum_{k\in\mathcal{K}_s} p_{kt} \max\{f_k + \phi_s^{t+1}(x_s + 1; \eta_s), \phi_s^{t+1}(x_s; \eta_s)\} + \left(1 - \sum_{k\in\mathcal{K}_s} p_{kt}\right)\phi_s^{t+1}(x_s; \eta_s), \tag{35}$$

under the conditions $x \le y_s$ and $\eta_s = w_{j_s} + y_s$. The boundary condition is given by $\phi_s^{\tau+1}(x_s; \eta_s) = \theta_{sj_s}\mathbb{E}[\min(w_{j_s}, \mathbf{B}(x_s))]$. Here, we assume without loss of generality that $y_s \le \tau$. Next, we present that problem (30)–(34) provides a *lower bound* on the optimal total expected revenue. The proof of the proposition is given in the supplementary document.

Proposition 5.1. *The optimal objective value of problem (30)–(34) gives a lower bound on the optimal expected revenue of the dynamic programming model given in (25). That is, when $\eta_s^* = y_s^* + w_{j_s}^*$ is the optimal capacity reserved for pair s for advance bookings and stay-over reservations, we have $\sum_{s\in\mathcal{S}} \phi_s^1(0; \eta_s^*) \le v_1(\mathbf{0})$.*

We note that the approach of Birbil et al. (2014) is applicable when the objective function (30) is discrete concave. The next lemma shows that this condition is satisfied thanks to the dynamic programming recursion in (35). The proof of the lemma is given in the supplementary document.

Lemma 5.1. *The function $\eta_s \mapsto \phi_s^t(0; \eta_s)$ is discrete concave. That is, the differences given by $\eta_s \mapsto \phi_s^t(0; \eta_s + 1) - \phi_s^t(0; \eta_s)$ are non-increasing.*

In the following lemma, we show that the dynamic programming recursion (35), which constitutes the objective function of our optimization problem, is supermodular in terms of the total number of guests and the reserved capacity for stay-overs. That is, the incremental change in reserved stay-over capacity increases with the number of guests in the system.

Lemma 5.2. *Given that $\eta_s = w_{j_s} + y_s$, the function $\phi_s^t(x_s; \eta_s)$ is supermodular in x_s and w_{j_s} for any given y_s.*

The proof of Lemma 5.2 is given in the supplementary document. Note that the same result can be shown for the capacity y_s reserved for pair s. This would then imply that the incremental change in the total number of guests (x_s) for larger values of y_s is greater than the smaller values of y_s (Powell, 2010). Given that, the difference $\phi^t(x_s; \eta_s) - \phi^t(x_s + 1; \eta_s)$ provides the opportunity cost of a room in pair s when we have x_s reservations at time period t. The supermodularity means that the lowest acceptable product fare can decrease as the allocated capacity to the stay-overs increases.

Remark 5.1. Although the walk-in customers are not included in model (30)–(34), they can be easily incorporated. In our original problem formulation, we assume that customers can only arrive during the reservation interval. On the other hand, the walk-in requests for pair s will arrive in the service interval. Again, we can update the model by considering the walk-in customers as new products with higher prices. By updating the product set, the fares and the probabilities, we can use the pair-based decomposition method proposed in this section to obtain the bid prices for the hotel problem with the advance bookings, the walk-ins and the stay-overs.

5.2. Multi-day stay-over

To model the multi-day stay-over requests, we divide the planning horizon into two periods. In the first period, the customers request rooms for the days in the second period. In the second period, the customers in the hotel ask to stay over for a few additional days. As it is the core idea behind the pair-based decomposition, in the first period we reserve rooms for each pair and then the customers of different product classes are accepted according to the policy obtained by the single-resource dynamic programming model. Likewise, in the second period we again reserve rooms for pairs. However, this time the pairs correspond to the various days of the stay-over requests.

To incorporate stay-over requests in the second period, we use a deterministic model. For ease of notation, we define $\mathcal{N}_s$ as the set of pairs that can be booked in the second period by pair s. We assume that each guest of pair s, independently of other reservations, can extend her stay and request a reservation for pair j with probability q_{sj}, $j \in \mathcal{N}_s$. Given that there are y_s accepted customers in pair s, the random number of pair j stay-over requests, $M_j(y_s)$, $j \in \mathcal{N}_s$ follow collectively the multinomial distribution with probabilities q_{s0} and q_{sj}, $j \in \mathcal{N}_s$ and the number of trials y_s. We use μ_{sj} to denote the expected demand for extension j coming from pair s. Let w_{sj} be the number of requests for extension j coming from pair s that we may accept over the second period. Then, the multi-day stay-over problem becomes

$$\text{maximize} \sum_{s\in\mathcal{S}} \varrho_s^1(y_s) + \sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{N}_s} \theta_{sj} w_{sj} \tag{36}$$

$$\text{subject to} \sum_{s\in\mathcal{S}} a_{is} y_s + \sum_{s\in\mathcal{S}}\sum_{j\in\mathcal{N}_s} a_{ij} w_{sj} \leq C_i, \qquad i \in \mathcal{L}, \tag{37}$$

$$\sum_{j\in\mathcal{N}_s} w_{sj} \leq y_s, \qquad s \in \mathcal{S}, \tag{38}$$

$$w_{sj} \leq \mu_{sj}, \qquad s \in \mathcal{S}, j \in \mathcal{N}_s, \tag{39}$$

$$y_s, w_{sj} \in \mathbb{Z}_+, \qquad s \in \mathcal{S}, j \in \mathcal{N}_s. \tag{40}$$

Here, $\varrho_s^1(y_s)$ is a single-resource dynamic programming recursion defined by (24). As we have discussed in Section 4.2, $y_s \mapsto \varrho_s^t(y_s)$ is discrete concave, and hence, the objective function in (36) is also discrete concave. Consequently, we can directly apply the pair-based decomposition approach proposed by Birbil et al. (2014).

Remark 5.2. Our stay-over models can be extended to several other options in hotel RM. For instance, many hotels offer a late check-out option to their customers. This option allows the customers to stay a few more hours in the hotel. While some of the hotels offer this option upon customer request without any additional charge, others present this option in return of a small fee. Since late check-outs occupy the room until the allowed time, they generally consume the day capacity. We can slightly modify our multi-day stay-over model (36)–(40) to incorporate this option. Late check-out requests can be considered as a special case of stay-over request. To incorporate this option, again we divide the planning horizon into two periods. In the first period, the customers request rooms for the days in the second period. In the second period, the customers can ask for late check-out or to stay over for a few additional days. We define new pairs to represent the late check-out requests in the second period. For each day, the pair (i, l_i) is used to denote a late check-out option on day i. For instance, a reservation with check-in and check-out pair (b, i) can request pair (i, l_i) for late check-out and pair (i, j) to extend her stay for ($j-i$) days. By defining new pairs for late check-out option, we can update the stay-over model (36)–(40) to find reservation policies for late checkouts. Another important application in hotel revenue management is overbooking. There are several overbooking models proposed for single-leg revenue management that are discrete concave in terms of the capacities (see Aydin, Birbil, Frenk, and Noyan (2013) and Talluri and van Ryzin (2004)). The single-leg capacity corresponds to each pair in our network model. Again, following our construction in multi-day stay-over model, the overbooking problem for hotel RM can also be solved in two stages. However, it is important to note that preallocating the hotel capacity to even more pairs and determining the individual overbooking limit for each pair may poorly affect the proper control of hotel capacity network-wide. This poor performance of the pair-based decomposition due to further partitioning has also been raised by Birbil et al. (2014).

6. Computational experiments

In this section, we present our computational results that illustrate the performances of the proposed solution methods. The benchmarking study here follows a similar outline as given by Topaloglu (2009). We start with our simulation setup.

We assume that the product requests arrive over the discrete time periods $\mathcal{T} = \{1, \ldots, \tau\}$. At each time period, we first generate an arrival request and then apply the corresponding policy. The probability that there is a request for product k at time period t is p_{kt}. To test the performances of the booking policies against varying arrival intensities, we use the load factor parameter ρ given by

$$\rho = \frac{\rho_1 + \rho_2}{\sum_{i\in\mathcal{L}} C_i},$$

where

$$\rho_1 = \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{L}}\sum_{k\in\mathcal{K}} a_{ik} p_{kt} \quad \text{and} \quad \rho_2 = \sum_{s\in\mathcal{S}}\sum_{k\in\mathcal{K}_s}\sum_{j\in\mathcal{N}_s}\sum_{t\in\mathcal{T}} a_{ij} p_{kt} q_{sj}.$$

Here, ρ_1 and ρ_2 are the total expected loads for the advanced bookings and the stay-overs, respectively.

In all our numerical experiments, we set the capacity of the hotel and the length of the planning horizon to 50 and 200, respectively. To compute the product fares, we define the day-based prices, σ_i. If a product covers only a single day, its fare is set equal to the corresponding day-based price. In our simulation set-up, we assume that the hotel makes a discount (d), if a customer requests to stay more than one day. We compute the fares of these type of products by

$$f_k = d \sum_{i\in\mathcal{L}} a_{ik}\sigma_i.$$

6.1. Models without stay-overs

In this section, we compare the performances of the decomposition methods that we have proposed for the dynamic model without stay-overs. We begin by describing the solution methods that we use for benchmarking.

- *Linear Programming Formulation (LPF):* This strategy is related to our linear model given in (10)–(15). We solve this model to obtain the optimal fare allocations $\{\alpha_{ikt} : i \in \mathcal{L}, k \in \mathcal{K}, t = 1, \ldots, \tau\}$. These fare allocations are then used to compute the capacity allocation policy of each day. As given in relation (6), the differences in the objective function values are used as the threshold values for accepting or rejecting the reservation requests.
- *Time-independent Fare Allocation (TIF):* LPF computes the fare allocations for each day, each product and each time period. In

this strategy, we redefine the fare allocations by relaxing the time dependency. In other words, we replace the decision variables $\{\alpha_{ikt} : i \in \mathcal{L}, k \in \mathcal{K}, t = 1, \dots, \tau\}$ with $\{\alpha_{ik} : i \in \mathcal{L}, k \in \mathcal{K}\}$ in problem (10)–(15). The optimal values of these time independent allocations are then used to compute the accept-reject policy as in the LPF strategy.

- *Deterministic Linear Program (DLP):* This model is presented in the supplementary document. Model (41)–(44) is solved to obtain the optimal dual variables associated with the capacity constraints (42). When a reservation request arrives, the summation of the optimal dual variables corresponding to those days covered by the reservation is used as the bid-price for the decision; see Talluri and van Ryzin (2004).
- *Deterministic Fare Allocation (DFA):* This strategy solves models (18)–(22) to find the fare allocation. We use the dual variables associated with constraints (21) in model (18)–(22) as fare allocations of a product. Then, we use these day-based fare allocations to compute the capacity allocation policy of each day as in the LPF method.
- *Randomized Linear Program (RLP):* Different from DLP, the randomized linear program uses the actual demand samples. In this strategy, we generate Z independent sample paths for the demand and for each sample path, we replace the right-hand side of constraints (43) by the numbers of arrivals in the sample path. After solving the modified problem, we store the optimal dual variables associated with constrains (42). The averages of these optimal dual variables are then used for computing the bid prices in accept-reject decisions; see Talluri and van Ryzin (1999). We use $Z = 50$ in our computational experiments.
- *Finite Differences on Deterministic Linear Program (FDD):* To compute the bid prices in accept-reject decisions, this strategy uses the total opportunity cost of the day-capacities consumed by a product request. We first compute the optimal expected revenue with the current levels of the capacities by solving models (41)–(44). Then, for each product we set up a new linear program by decreasing the capacities of those days covered in the product by one. The difference between the optimal expected revenues of model (41)–(44) with the original capacities and the new problem becomes the threshold value for products. If the fare price of a product request is greater than its threshold value, we accept the request; see Bertsimas and Popescu (2003). This approach requires the solving of a separate linear program for each product.
- *Finite Differences on Randomized Linear Program (RFD):* This strategy is an extension of RLP and FDD. As in RLP strategy, we first generate Z independent sample paths for the demand. Then, we compute the threshold value for each product and for each sample path. We accept an arriving product request, if its fare price is greater than the average threshold value (Topaloglu, 2009). We use $Z = 50$ in our computational experiments.
- *Pair-Based Decomposition (PBD):* This strategy focuses on finding a decision policy for each pair. We solve model (23) to obtain the bid prices associated with the capacity constraints of the service days. As with DLP, these bid prices are used to compute the decisions.

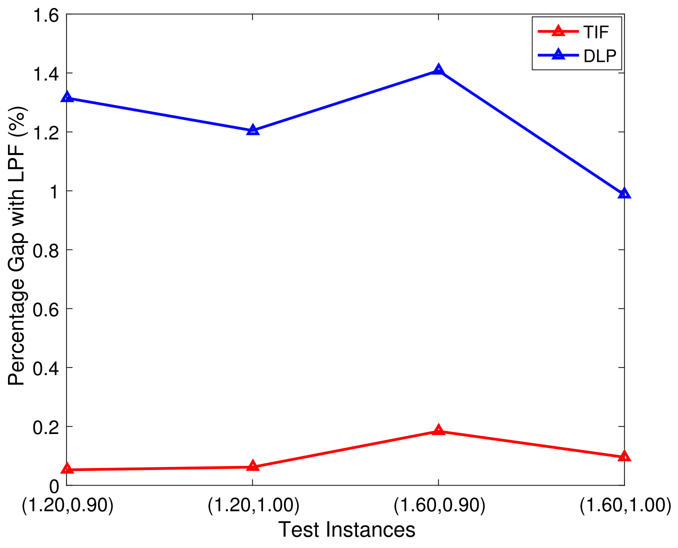
To refine the bid prices and threshold values in our implementation, we divide the planning horizon into five equal segments and revise the strategies at the beginning of each segment with the updated daily hotel capacities and the updated demand. Our experimental design is based on various factors of the network size (n), the load factor (ρ) and the discount multiplier (d). We test our models in two networks with $n \in \{5, 7\}$. We use the load factor values $\rho \in \{1.2, 1.6\}$ corresponding to the low and the high loads, respectively. Likewise, the discount factor values $d \in \{0.90, 1.00\}$ are used to represent the low and the high fares, respectively. We label our test problems by using various combinations of these parameters.

As we have shown by Proposition 4.1 in Section 4.1, the optimal objective values of the day-based decomposition methods provide upper bounds on the total expected revenue obtained by the dynamic model (1). We first compare these bounds. Our computational results show that LPF consistently provides the tightest upper bound. Therefore, the upper bound obtained by LPF is used as a base approach to report the relative performances of the upper bounds provided by the remaining approaches. Fig. 2(a) presents the percentage gaps between the optimal objective value of LPF and the optimal objective values of TIF and DLP. In these figures, the tuple (ρ, d) in the horizontal axis shows a problem instance. The results depict that TIF also provides relatively tight upper bounds. For all of the test problems, the percentage gap between LPF and TIF is below 1%. TIF uses the time independent fare allocations and, hence, it is computationally more efficient than LPF. These results indicate that we can use the time independent fare allocations to obtain a good estimate of the optimal expected revenue. An interesting result we have is that the upper bound provided by DLP is also tight. This may be due to the linear structure of the network. Although the deterministic model approximates the stochastic arrival process, it considers the whole network problem without decomposing it. On the other hand, LPF decomposes the network problem by days. This approximation deteriorates the optimal objective value of LPF.

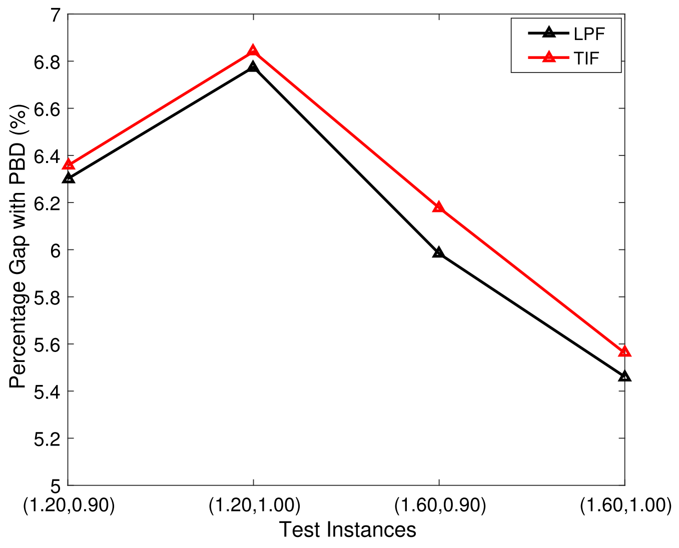
In Section 4.2, we have shown that the optimal objective function value of the pair-based decomposition problem (23) gives a lower bound for the complete dynamic model (Proposition 4.2). Fig. 2(b) presents the percentage gaps between the optimal objective values of LPF and PBD as well as between those of TIF and PBD. We report the gaps only for LPF and TIF as they give the tightest upper bounds in Fig. 2(a). We observe that the percentage gaps in Fig. 2(b) are on average less than 7%, and these gaps are mainly affected by the load factor. Note that each one of these relative differences provides an upper bound on the optimality gap. Therefore, these figures could be used for estimating the loss in revenue due to approximating the complete dynamic model.

Our simulation results are summarized in Figs. 3 and 4. In these figures, we present the average revenues obtained by the decomposition methods with respect to the low and the high load factor values (ρ) and the discount factor (d). Fig. 3 shows the average revenues for the test problems with four days. The results indicate that LPF and TIF outperform all solution methods. We observe that the discount factor d significantly affects the performances of the benchmark methods. When d is 1.0, LPF outperforms the randomized methods by 3% and the static methods by 8% on average. When d is less than 1.0, accepting the long-stays is not advantageous due to the low daily revenue contribution. Day-based decomposition methods perform much better when the daily contribution of the long-stays is not lower than the single-day stays. As depicted in Fig. 3, the performance differences are particularly noticeable when the load factor is high. When we compare RLP and RFD with DLP and FDD, we see that introducing randomness to the deterministic model substantially improves the performances of the bid price policies. However, the performances of RLP and DFA become relatively close. This is due to the fact that the dynamic booking policy of DFA captures the stochastic nature of the hotel network problem better than the randomized linear programming. On the other hand, PBD performs worse than the remaining solution methods.

Next, we investigate the effect of network size on the performances of the solution methods. Fig. 4 shows these performances for the test problems with six days. Due to its



(a) Upper bound percentage gaps



(b) Upper bounds on optimality gap

Fig. 2. Percentage gaps on the maximum total expected revenue ($n = 5$).

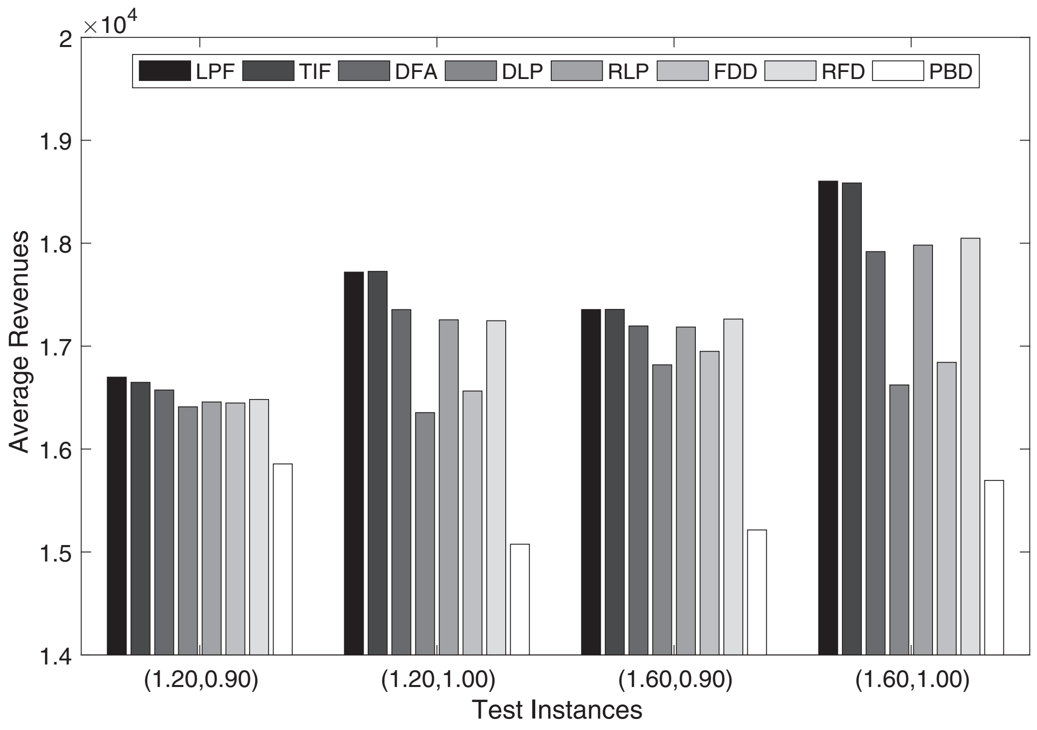


Fig. 3. The average revenues for varying load-factor (ρ) and discount factor (d) parameters ($n = 5$).

computational burden, we do not test the performance of LPF for these test instances. The first observation we have is that the performances of DLP, FDD and PBD worsen under this set-up. Comparing the performances of TIF, DFA, RLP and RFD in Fig. 3 with those in Fig. 4, we note that day-based decomposition models coordinate the network booking decisions well when the load-factor is high and there is no discount for the long-stays. For all test instances, TIF and DFA perform better than the remaining strategies. When we compare RLP and RFD, we observe that their performances are quite close. On the other hand, PBD performs worse than the other solution methods. This behavior can be attributed to the effect of capacity sharing. Since SBP partitions the network problem into independent pairs, its performance is altered by the cross-effects in the network.

6.2. Models with stay-overs

In this part of the computational study, we test the performances of the models in the presence of stay-over customers. We use the following benchmark strategies.

Single-Day Stay-Over Model (SSM): This is the solution method that we have presented in Section 5 for the single-day stay-over problem. We solve the model in (30)–(34) by relaxing the integrality constraints. The optimal values of the dual

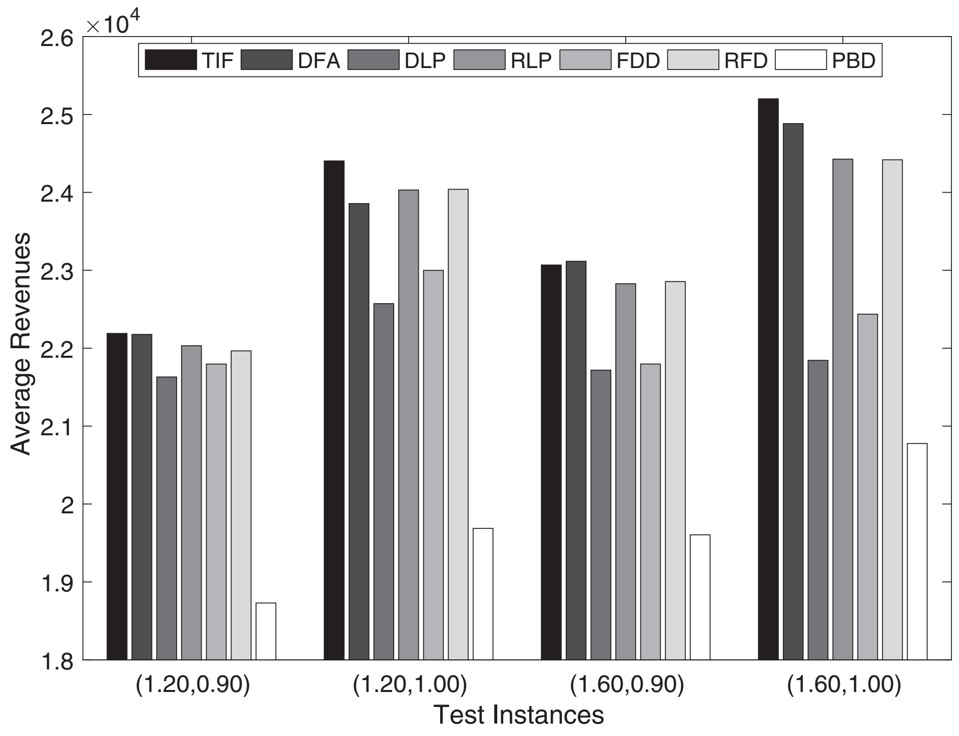


Fig. 4. The average revenues for varying load-factor (ρ) and discount factor (d) parameters ($n = 7$).

variables associated with the capacity constraints are then used as the bid-prices for the reservation policy.

Multi-Day Stay-Over Model (MSM): This is the solution method described in Section 5 for the multi-day stay-over problem. We solve the model in (36)–(40) to obtain the optimal dual variables associated with the capacity constraints. Similar to SSM, these dual variables are used as the bid-prices for the reservation policy.

Deterministic Stay-Over Model (DSM): The deterministic model considering stay-over customers is given in the supplementary document. This strategy solves the model in (45)–(49) to obtain the optimal dual variables associated with the capacity constraints. We use these dual variables as the bid prices for the decision policy.

Randomized Stay-Over Model (RSM): This solution method is similar to the RLP strategy, which has been used for the advance booking problem in the previous subsection. In this strategy, we generate Z independent sample paths for the demand and for each demand sample, we generate W samples for the stay-over requests. Again, for each demand sample, we replace the right-hand-side of constraints (47) by the numbers of arrivals in the sample path. Moreover, instead of using the expected stay-over requests, we use the W samples of the stay-over requests. After solving the modified problem for each demand sample, we store the optimal values of the dual variables associated with constraints (46). The averages of these optimal dual variables are then used for computing the bid prices in the accept-reject decisions; *cf.* Kunnumkal, Talluri, and Topaloglu (2012). We use $Z = 25$ and $W = 50$ in our computational experiments.

Pair-Based Decomposition (PBD): This strategy focuses on the advance bookings and ignores the stay-over customer requests. We solve model (23) to obtain the bid prices for the decision policy. We use this myopic strategy to measure the effects of having the stay-over customers in the system.

We test the performance of our solution methods on two types of problems. In the first type, we only consider the single-day stay-over requests. In the second type, we allow multiple day stay-over reservations.

We begin by presenting our results on the single-day stay-over case. In these problems, we set the network size (n) to five. We test the performances of the benchmark solutions with respect to the load factor (ρ), the stay-over probability ($q_{s.}$) and the stay-over fare coefficient (r). We use the load factor values $\rho \in \{1.2, 1.6\}$ corresponding to the low and the high loads. We give two sets of stay-over probabilities to represent the low and the high stay-over rates. These are $q_s^L = 0.05$ and $q_s^H = 0.10$ for the single-day stay-over; $q_s^L \in \{0.08, 0.06, 0.02\}$ and $q_s^H \in \{0.15, 0.12, 0.08\}$ for the one day to three days stay-over problems. We compute the price of stay-over reservation on day i by using the daily prices. For example, the price of the stay-over request on day i is $(1 + r)\sigma_i$. To represent the low and the high stay-over fares, we use the stay-over fare coefficient $r \in \{0.10, 0.20\}$. We label our test problems by using all combinations of these parameters. Under this setup, we have evaluated the reservation policies of all solution methods. To update the bid prices in our implementation, we divide the planning horizon into five equal segments and reoptimize the bid price policies at the beginning of each segment.

Our main results are summarized in Figs. 5 and 6. Comparing the average revenues in Fig. 5, we observe that SSM generally outperforms the remaining solution methods. The performance differences are most notable when ρ is high. As the load factor increases and the capacity on each day gets tighter, the performance gaps between the benchmark solutions increase. The performance of MSM booking policy can be poor. As it is depicted in Fig. 5, in some cases the average revenues obtained by MSM lag behind RSM. This demonstrates the importance of considering the uncertainty due to the stay-over customers. DSM also performs poorly as it disregards the variance in the arrival processes. When we compare the performances of PBD and SSM, we see that taking the

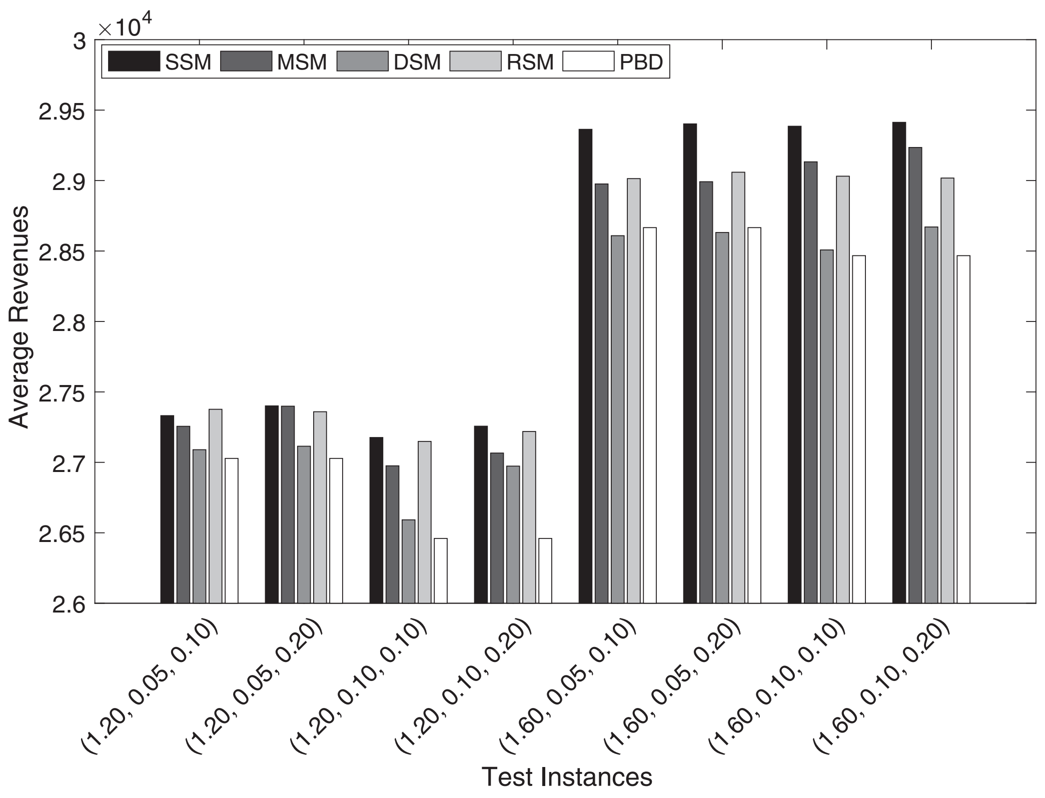


Fig. 5. The average revenues for varying load-factor (ρ), stay-over probability and stay-over fare (r) parameters for single-day stay-over problem.

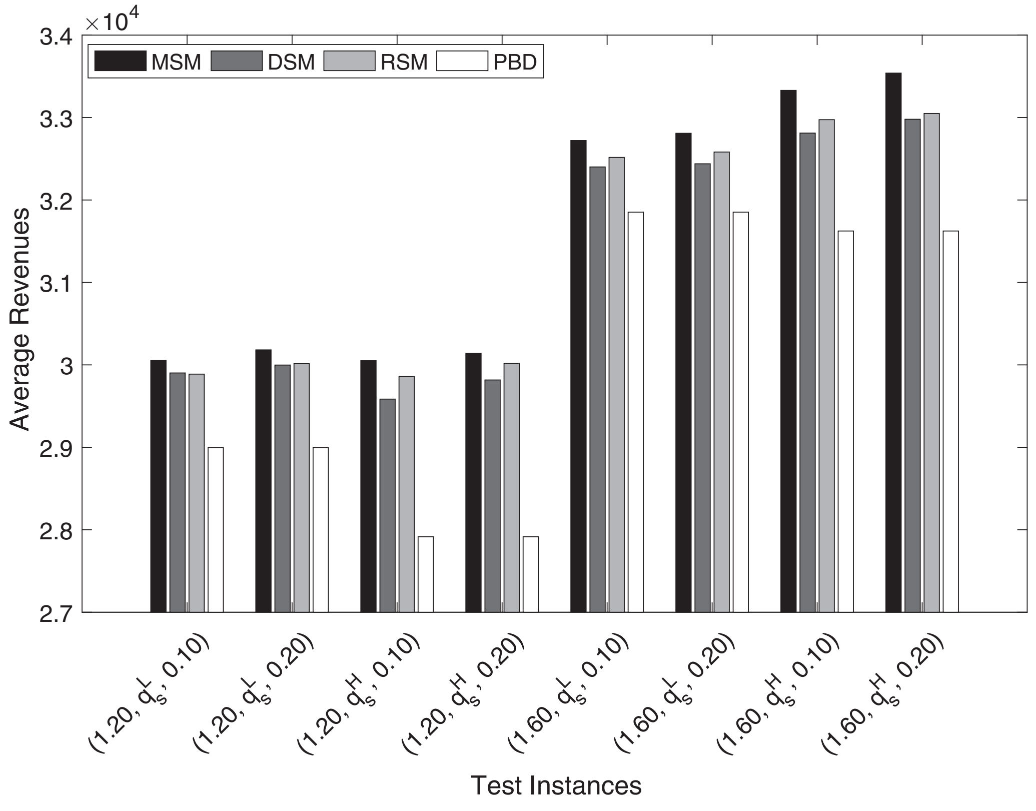


Fig. 6. The average revenues for varying load-factor (ρ), stay-over probability and stay-over fare (r) parameters for multi-day stay-over problem.

stay-over requests into account brings a significant advantage. As expected, the relative gap between these two models decreases as the load factor increases. However, even in this case, reserving a part of capacity to the stay-overs brings additional revenue.

Fig. 6 illustrates the average revenues for the multiple day stay-over problem. The first observation we have is that the performance of MSM method is better than those of RSM and DSM in the multi-day case. Comparing the plots for MSM, RSM and DSM in Fig. 5 with those in Fig. 6, we note that the performances of RSM and DSM deteriorate when the multi-day stay-overs are possible. The difference is more striking when both the load factor and the stay-over probability are high. This result shows that MSM strategy manages the multi-day stay-over reservations better as the hotel capacity gets tighter. When we compare RSM and DSM, we observe that the performance gap between these two methods increases with the stay-over probability. This behavior implicitly demonstrates the importance of considering the inherent stochasticity in the problem.

7. Conclusion

In this study, we work on the dynamic room allocation problem in hotel revenue management. Due to the complexity of this problem, we concentrate on several approximation methods. We analyze the structural properties of the problem and present day- and pair-based decomposition approaches that can handle the walk-in and the stay-over customers. First, we work on the day-based decomposition methods. Day-based decomposition generates independent subproblems for each day and, hence, it cannot store the number of reserved rooms for each product. Therefore, incorporating the stay-over customers becomes a challenge. In the second part, we work on the stay-over extension. To the best of our knowledge, the dynamic programming model that includes the stay-over customers has not been proposed in the literature before. We first focus on the single-day stay-over problem. By extending the work of Birbil et al. (2014), we propose a solution method. Second, we consider the multi-day stay-over problem and present a two-period approximation, which combines the pair-based decomposition with the deterministic linear programming. We conduct a thorough computational study and investigate the performances of our proposed models along with some well-known approaches used in the literature. Our computational experiments indicate that the proposed policies perform well. The performance gaps are especially significant when the hotel's daily capacity is tight and the stay-over probability is high.

As we mentioned in Section 5.2, our stay-over models can be extended to several other applications in hotel RM. Recently, hotel reservation systems have started to offer late checkout option to their customers. Late checkout requests can be considered as a special case of stay-over problem where the customers can extend their stay until the allowed time specified by the hotel. Following the same construction as for the stay-over model, we can obtain the reservation policies for late checkouts. Another important issue in hotel revenue management is overbooking. Similarly, the overbooking option can be incorporated in the multi-day stay-over model and it can also be solved in two stages. However, it is important to note that preallocating the hotel capacity to even more pairs and determining the individual overbooking limit for each pair may poorly affect the control of hotel capacity network-wide. Incorporation of the overbooking option is a potential topic for future research.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2018.05.027.

References

Adelman, D. (2007). Dynamic bid prices in revenue management. *Operations Research, 56*, 647–661.

Aslani, S., Modarres, M., & Sibdari, S. (2013). A decomposition approach in network revenue management: Special case of hotel. *Journal of Revenue and Pricing Management, 12*, 451–463.

Aydin, N., Birbil, S. I., Frenk, J. B. G., & Noyan, N. (2013). Single-leg airline revenue management with overbooking. *Transportation Science, 47*, 560–583.

Baker, T. K., & Collier, D. A. (1999). A comparative revenue analysis of hotel yield management heuristics. *Decision Sciences, 30*, 239–263.

Bertsimas, D., & Popescu, I. (2003). Revenue management in dynamic network environment. *Transportation Science, 37*, 257–277.

Birbil, S. I., Frenk, J. B. G., Gromicho, J. A. S., & Zhang, S. (2014). A network airline revenue management framework based on decomposition by origins and destinations. *Transportation Science, 48*, 313–333.

Bitran, G. R., & Gilbert, S. M. (1996). Managing hotel reservations with uncertain arrivals. *Operations Research, 44*, 35–49.

Bitran, G. R., & Leong, T. Y. (1989). Hotel sales and reservations planning. *Technical Report*. Sloan School of Management, Massachusetts Institute of Technology.

Bitran, G. R., & Mondschein, S. V. (1995). An application of yield management to the hotel industry considering multiple day stays. *Operations Research, 43*, 427–443.

Chen, D. (1998). Network flows in hotel yield management. *Technical Report*. New York, USA: Cornell University.

De Boer, S. V., Freling, R., & Piersma, N. (2002). Stochastic programming for multiple-leg network revenue management. *Europen Journal of Operational Research, 137*, 72–92.

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.

Goldman, P., Freling, R., Pak, K., & Piersma, N. (2002). Models and techniques for hotel revenue management using a rolling horizon. *Journal of Revenue and Pricing Management, 3*, 207–219.

Guadix, J., Cortes, P., Onieva, L., & Munuzuri, K. (2010). Technology revenue management system for customer groups in hotels. *Journal of Business Research, 63*, 519–527.

Hotwire (2017). https://www.hotwire.com/hotels/. (Last accessed: 31 May 2017).

Ivanov, S. (2014). *Hotel revenue management-from theory to practice*. Varna, Bulgaria: Zangador.

Ivanov, S., & Zhechev, V. (2012). Hotel revenue management-a critical literature review. *Tourism, 60*, 175–198.

Kimes, S. E. (1989). Yield management: a tool for capacity-constrained service firms. *Journal of Operations Management, 8*, 348–363.

Koide, T., & Ishii, H. (2005). The hotel yield management with two types of room prices, overbooking and cancellations. *International Journal of Production Economics, 93*, 417–428.

Kunnumkal, S., Talluri, K., & Topaloglu, H. (2012). A randomized linear programming method for network revenue management with product-specific no-shows. *Transportation Science, 46*, 90–108.

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.

Kunnumkal, S., & Topaloglu, H. (2011). A stochastic approximation algorithm to compute bid prices for joint capacity allocation and overbooking over an airline network. *Naval Research Logistics, 54*, 323–343.

Ladany, S. P. (1976). Dynamic operating rules for motel reservations. *Decision Sciences, 7*, 829–840.

Lai, K. K., & Ng, W. L. (2005). A stochastic approach to hotel revenue optimization. *Computers and Operations Research, 32*, 1059–1072.

Lippman, S. A., & Stidham, S. (1977). Individual versus social optimization in exponential congestion systems. *Operations Research, 25*, 233–247.

Liu, S., Lai, K. K., & Wang, S. Y. (2008). Booking models for hotel revenue management considering multiple-day stays. *International Journal of Revenue Management, 2*, 78–91.

Nadarajah, S., Lim, Y. F., & Ding, Q. (2015). Dynamic pricing for hotel rooms when customers request multiple-day stays. *Research Collection Lee Kong Chian School of Business*, 1–42.

Powell, W. B. (2010). *Approximate dynamic programming: solving the curses of dimensionality* (2nd). New Jersey, USA: John Wiley and Sons.

Priceline (2017). https://www.priceline.com/hotels/. (Last accessed: 31 May 2017).

Ruszczynski, A. (2006). *Nonlinear optimization*. Princeton, NJ: Princeton University Press.

Talluri, K., & van Ryzin, G. (1999). A randomized linear programming method for computing network bid prices. *Transportation Science, 33*, 207–216.

Talluri, K. T., & van Ryzin, G. J. (2004). *The theory and practice of revenue management*. New York, NY: Springer.

Talya (2016). *Elektra hotel management system*. Talya Computer Systems. Personal communication.

Tepper, F. (2015). HotelTonight launches Tonight +1 to entice guests to stay an extra night. https://techcrunch.com/2015/11/19/hoteltonight-launches-tonight-1-to-tempt-guests-to-stay-an-extra-night/?guccounter=1/ Accessed 27 May 2018.

Topaloglu, H. (2009). Using Lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. *Operations Research, 57*, 637–649.

Topaloglu, H. (2012). A duality based approach for network revenue management in airline alliances. *Journal of Revenue and Pricing Management, 11*, 500–517.

Vinod, B. (2005). Alliance revenue management. *Journal of Revenue and Pricing Management, 4*, 66–82.

Weatherford, L. R. (1995). Length of stay heuristics: Do they really make a difference? *Cornell Hotel and Restaurant Administration Quarterly, 36*, 70–79.

Williams, F. E. (1977). Decision theory and the innkeeper: An approach for setting hotel reservation policy. *Interfaces, 7*, 18–30.

Zhang, D. (2011). An improved dynamic programming decomposition approach for network revenue management. *Manufacturing and Service Operations Management, 13*, 35–52.

Zhang, D., & Weatherford, L. (2017). Dynamic pricing for network revenue management: A new approach and application in the hotel industry. *Informs Journal on Computing, 29*, 18–35.