Decomposition Methods for Solving Dynamic Programming Problems in Hotel Revenue Management

Nurşen Aydın, Ş. İlker Birbil
Faculty of Engineering and Natural Sciences, Sabancı University, Orhanlı-Tuzla, 34956 Istanbul, Turkey
nursenaydin@sabanciuniv.edu, sibirbil@sabanciuniv.edu

Abstract: Long-term stays are common in the hotel business. Therefore, it is crucial to consider the allocation of the available rooms to a stream of customers requesting to stay multiple days. This requirement leads to solving large-scale dynamic network revenue management problems, which are computationally challenging. A remedy is to apply decomposition approaches so that an approximate solution can be obtained by solving many single-day problems. In this study, we investigate optimal 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 large problems efficiently.

Keywords: Revenue management; hotel; decomposition; solution approaches

1. Introduction. Historically, airline industry played the steering role in revenue management (RM). Today, however, there is a wide range of applications in different industries. In particular, RM techniques are applicable in any industry with volatile demand requesting fixed, perishable capacity (Kimes, 1989). Although 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) present a review of the methods proposed in hotel RM and point out the gaps in the literature.

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 special in hotel reservation systems. First, multi-night stays in hotels are quite common. While a flight itinerary includes on average less than several legs, the number of nights a typical customer spends in a hotel can be as high as twelve (Zhang and Weatherford, 2012). Second, the demand process is different. Hotel customers may decide to stay longer while they are in the hotel (Kimes, 1989). Third, airline customers generally make advance purchases but a portion of the hotel customers consists of walk-ins, and even the early reservations in the booking interval can 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; i.e., the capacity, for any day is determined only by those reservations including that particular day. Therefore, the room allocation problem becomes the control of total room capacity when the customer demand is characterized by the length of stay and the room types. Although this setting leads to a special linear network structure, the dynamics of the problem is still quite challenging for analysis and optimization. Above all, the state space is the Cartesian product of daily capacities in the network. This is a standard difficulty in a network RM problem. As a remedy, approximation methods based on decomposition can be proposed. 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 and van Ryzin, 2004). However, it is important to note that these decomposition methods risk undermining the network effects of shared daily capacities among different products (reservation types).
To improve the performance, recent studies focus on protecting network information by defining time and capacity dependent bid-prices and consider their effects on the network (Kunnumkal and Topaloglu, 2010; Zhang, 2011; Aslani et al., 2013).

In this paper, we focus on the room allocation decisions for a hotel. The optimal policy to admit or reject an arriving customer can be obtained by analyzing the stochastic nature of the customer arrival process. In hotel reservation systems, customers are classified as advance bookings, stay-overs and walk-ins. While advance bookings make room reservations before they arrive at the hotel, walk-ins show up without any reservation. Stay-overs are those customers who ask for an extension for their reservations during their stay in the hotel. The arrival process of advance bookings and walk-ins are similar. The only difference is that the walk-in customers arrive after reservation period ends. However, arrivals of stay-over requests depend on the accepted advance bookings. To simplify our notation, we ignore walk-in customers and formulate our problem by considering advance booking and stay-overs. Then, we explain how one can easily incorporate walk-in customers to our proposed models. To the best of our knowledge, the dynamic model of stay-over customers in network setting have 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 day-based and 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 advance bookings and formulate a linear program for the problem. The resulting model has a block angular structure and can be efficiently solved. However, for some large network problems, it may still require a high computation time to obtain a solution. Therefore, we propose alternate approximate models that can be solved very fast. We show that these alternate models provide upper and lower bounds on the optimal expected revenue of the original model. To manage 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 that 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 et al. (2014). Different than the work of Birbil et al., our model has two dimensional objective function. By utilizing from their idea, we propose a modelling approach to handle stay-over customers. We also discuss how to incorporate walk-ins in our decomposition approaches. To test the performances of the proposed decomposition approaches, we conduct extensive numerical experiments and compare our results against those obtained by several well-known models from the literature. Our computational experiments indicate that the proposed decomposition approaches are apt to effective room allocation in hotel RM problems.

2. Review of Related Literature. We begin by reviewing the related work on hotel RM. Then, we summarize the decomposition approaches frequently applied to 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 in order 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; stayovers, reservations and walk-ins. He computes the reservation policy for each customer type by comparing underbooking and overbooking costs. 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 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: customers with guaranteed reservations, customers with reservations and 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 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 optimal solution of the linear program is always integral. Goldman et al. (2002) propose deterministic and stochastic linear programming models to find nested booking limits and 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 Boer et al. (2002) on airline revenue management problem. However, unlike the models proposed by Weatherford and Boer et al., 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 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 early discounts, cancellations and 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 et al. (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 et al. (2010) present a decision support system for forecasting and room allocation decisions. They work on the deterministic and stochastic programming models by considering the group arrivals. The proposed decision support system integrates these models for room allocation and pricing decisions.

The problem considered in this study builds 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 dynamic bid prices. He first formulates the network problem as a dynamic model, which suffers from 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, nonseparable 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 problem size increases. To reduce the computational burden, Kunnumkal and Topaloglu (2011) develop a stochastic approximation algorithm which provides capacity independent bid prices. In this approach, they formulate the total expected profit as a function of 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 airline revenue management with customer choice behavior. In this method, they first allocate revenues of each itinerary among the covered legs. Then, they define a penalty term to incorporate the network effect. They view the revenue allocations and 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 network problems of substantial size. Hotel network revenue management problems are also tackled with decomposition methods. Zhang and Weatherford (2012) 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 non-linear non-separable functions. They test the proposed approach by using the data from an 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 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. We focus on multiple day problem and propose several decomposition methods to attack the problem. In particular, our aim is to find a dynamic capacity allocation policy considering advance booking and stay-over customers. Moreover, we develop alternate solution methods to improve the computational time for large-scale problems.

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

In our formulation, the planning horizon consists of two sub-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. These requests arrive in the reservation interval and hence, they are called as advance bookings. Each accepted customer is then entitled for a one-day extension, if there is available capacity. These extension requests constitute the second type of event that occur in the service interval. Note that allowing only one day extension is a simplified version of the actual process as customers may request multiple day extensions in real life. However, an extension request spanning multiple days is quite rare in the hotel industry.

Figure 1 shows the structure of a hotel network with multiple-day stays. Each node in the service
interval corresponds to a day. We index service days by \(i\), products by \(k\) and 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 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 \(L\). Again using capital letters for
sets, \(K\) and \(S\) denote the set of products and the set of pairs, respectively. We use the operator \(|.|\) when
we refer to the cardinality of a set.

\[
\begin{align*}
\text{Reservation Interval} & \quad \text{Service Interval} \\
\text{\(1\)} & \quad \text{\(2\)} & \quad \text{\(n\)} \\
\text{day 1 \((C_1)\)} & \quad \text{day 2 \((C_2)\)} & \quad \text{day \(i\) \((C_i)\)} & \ldots & \text{product \(k\)}
\end{align*}
\]

Figure 1: The hotel network with multiple time intervals

We divide the reservation interval into \(\tau\) 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. We define \(a_{ik}\) to denote 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 \(e_i\) as an \(|L|\)-dimensional unit vector with a one in the element corresponding to day \(i\).
Moreover, we assume that each accepted booking of pair \(s\), independently of other bookings, can request
one day extension at the beginning of its check-out day with probability \(\pi_s\). If we accept the stay-over
request for pair \(s\), then we generate a revenue of \(\theta_s\). We let \(\bar{a}_{ks} = 1\), if product \(k\) is in pair \(s\); otherwise,
\(\bar{a}_{ks} = 0\). As above, we define \(e_s\) as an \(|S|\)-dimensional unit vector with a one in the element corresponding
to pair \(s\).

To give a dynamic programming formulation for this problem, we need to store the number of reser-
vations in each day and for each pair. We use \(z_i\) to denote the total number of reservations for day \(i\).
We also use \(x_s\) as the total number of reservations accepted for pair \(s\). Then, the vectors \(z = [z_i]_{i \in L}\)
and \(x = [x_s]_{s \in S}\) together form the state in our dynamic programming model. Let \(J_t(z, x)\) denote the
expected optimal revenue from \(t\) up to \(\tau\) given that the state of the reservations at the beginning of time
period \(t\) is \((z, x)\). For every \(1 \leq t \leq \tau\), we can find the optimal policy by computing the value functions
through the optimality equation

\[
J_t(z, x) = \sum_{k \in K} p_{kt} \max \left\{ f_k + J_{t+1}(z + \sum_{i \in L} a_{ik} e_i, x + \sum_{s \in S} \bar{a}_{ks} e_s), J_{t+1}(z, x) \right\} + \left( 1 - \sum_{k \in K} p_{kt} \right) J_{t+1}(z, x). 
\]

(1)

Then, the boundary condition becomes

\[
J_{\tau+1}(z, x) = \sum_{i=2}^{n-1} \sum_{s \in S_i} \theta_s \mathbb{E}[\min\{B(\pi_s, x_s), C_i - z_i\}],
\]

where \(S_i\) denotes the pairs ending on day \(i\) and \(B(\pi_s, x_s)\) is a Binomial random variable with success
probability \(\pi_s\) and the number of trials \(x_s\).
Unfortunately, this model is intractable because the state variables may involve many dimensions in practical applications and hence, solving the complete dynamic model is, in general, not possible (Talluri and van Ryzin, 2004). In the next section, we discuss two decomposition approaches to approximate the dynamic programming formulation. First, we ignore the stay-over requests and work on approximation methods for the dynamic model only with the advance bookings. Then, we discuss how to incorporate stay-over requests and develop an approximate model for the dynamic model with both advance bookings and stay-overs.

4. Mathematical Models without Stay-overs. In this section, we do not consider stay-over customers. Thus, the state variables \((z, x)\) in the dynamic model (1) are simply reduced to \(z\). Consequently, we can rewrite the dynamic model (1) as

\[
\tilde{J}_t(z) = \sum_{k \in \mathcal{K}} p_{kt} \max\{f_k + \tilde{J}_{t+1}(z + \sum_{i \in \mathcal{L}} a_{ikt} x_i), \tilde{J}_{t+1}(z)\} + (1 - \sum_{k \in \mathcal{K}} p_{kt}) \tilde{J}_{t+1}(z),
\]

where the boundary condition is \(\tilde{J}_{\tau+1}(z) = 0\) for all \(z\). This model is still intractable even for small-sized networks. To overcome this difficulty, we work on decomposition based approximation methods.

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 \(\alpha_{ikt}, i \in \mathcal{L}, k \in \mathcal{K}, t \in \mathcal{T}\) for each product. Then, we impose the constraints

\[
\sum_{i \in \mathcal{L}_k} \alpha_{ikt} = f_k, \quad k \in \mathcal{K}, t \in \mathcal{T}, \tag{3}
\]

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.

When we accept a booking request for product \(k\) at time period \(t\) on day \(i\), we generate a revenue of \(\alpha_{ikt}\) 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

\[
V_{it}(z_i|\alpha) = \sum_{k \in \mathcal{K}} p_{kt} \max\{\alpha_{ikt} + V_{i,t+1}(z_i + a_{ikt}|\alpha), V_{i,t+1}(z_i|\alpha)\} + (1 - \sum_{k \in \mathcal{K}} p_{kt}) V_{i,t+1}(z_i|\alpha), \quad \text{for every } 1 \leq t < \tau.
\]

The boundary conditions simply become \(V_{i\tau}(C_i|\alpha) = 0\) and \(V_{i\tau+1}(z_i|\alpha) = 0\) for all \(z_i = 1, \ldots, C_i - 1\). Kunnumkal and Topaloglu (2010) show that this type of approximation provides an upper bound on the optimal expected revenue for customer choice model. The next result follows similar steps that are given Kunnumkal and Topaloglu to establish an upper bound for the original dynamic program in (2).

**Proposition 4.1** For all \(z_i \leq C_i\) and \(1 \leq t \leq \tau\), we have \(\tilde{J}_t(z) \leq \sum_{i \in \mathcal{L}_t} V_{it}(z_i|\alpha)\) such that \(\alpha\) satisfies condition (3).

**Proof.** The desired result follows from induction. For any \(\alpha\) satisfying condition (3), at time period \(\tau\) we have

\[
\tilde{J}_\tau(z) = \sum_{k \in \mathcal{K}} p_{k\tau} \max\{f_k, 0\} = \sum_{k \in \mathcal{K}} p_{k\tau} \max\{\sum_{i \in \mathcal{L}} \alpha_{ikt}, 0\} \leq \sum_{i \in \mathcal{L}} \sum_{k \in \mathcal{K}} p_{k\tau} \max\{\alpha_{ikt}, 0\} = \sum_{i \in \mathcal{L}} V_{i\tau}(z_i|\alpha).
\]
Suppose now, the proposition holds for time period \( t + 1 \). Our proof is then completed once we show that 
\[
\sum_{i \in L} V_{it}(z_i|\alpha) \quad \text{provides an upper bound.}
\]
We have
\[
\tilde{J}_t(z) = \sum_{k \in K} p_{kt} \max_{i \in L} \{f_k + \tilde{J}_{t+1}(z + \sum_{i \in L} a_{ikt} e_i, \tilde{J}_{t+1}(z)) + (1 - \sum_{k \in K} p_{kt}) \tilde{J}_{t+1}(z)\}
\]
\[
= \sum_{k \in K} p_{kt} \max_{i \in L} \left\{ \sum_{i \in L} \alpha_{ikt} + \tilde{J}_{t+1}(z + \sum_{i \in L} a_{ikt} e_i, \tilde{J}_{t+1}(z)) + (1 - \sum_{k \in K} p_{kt}) \tilde{J}_{t+1}(z) \right\}
\]
\[
\leq \sum_{i \in L} \left[ \sum_{k \in K} p_{kt} \max_{i \in L} \left\{ \sum_{i \in L} \alpha_{ikt} + V_{it+1}(z_i + 1|\alpha), V_{it+1}(z_i|\alpha) \right\} + (1 - \sum_{k \in K} p_{kt}) V_{it+1}(z_i|\alpha) \right]
\]
\[
= \sum_{i \in L} V_{it}(z_i|\alpha).
\]
The first inequality holds due to the induction hypothesis assumption and the second inequality follows from the use of max function.

Since \( \sum_{i \in L} V_{ti}(0|\alpha) \) gives an upper bound on \( \tilde{J}_1(0) \), we can obtain the tightest bound by solving the following problem:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in L} V_{ti}(0|\alpha) \\
\text{subject to} & \quad \sum_{i \in K} \alpha_{ikt} = f_k, \quad k \in K, t \in T.
\end{align*}
\]

As a direct consequence of Proposition 4.1, the optimal objective function value of problem (5)-(6) provides an upper bound on the maximum expected revenue over the whole planning horizon. Moreover, it gives the optimal values of the revenue distribution that we can use to construct a control policy for each single-day dynamic programming model (4). The reservation policy for the network problem can be summarized as follows: Given the allocations \( \{\alpha_{ikt}, i \in L, k \in K, t \in T\} \) and the state variables \( \{z_i, i \in L\} \), we then accept a reservation request for product \( k \) at time period \( t \), if we have

\[
f_k \geq \sum_{i \in L} \left( (V_{it}(z_i|\alpha) - V_{it}(z_i + 1|\alpha)) \right).
\]

We next reformulate problem (5)-(6) as a tractable linear program. This derivation also shows simply that (5)-(6) is a convex optimization problem (see Kunnunkal and Topaloglu (2010) for an alternate proof of convexity). To rewrite the recursion let us first define

\[
x_{ktz}^i := \max \{\alpha_{ikt} + V_{i,t+1}(z_i + 1|\alpha) - V_{i,t+1}(z_i|\alpha), 0\}
\]

and

\[
x_{k\tau}^i := \max \{\alpha_{ikt}, 0\}.
\]

Then, we can rewrite (4) as

\[
V_{it}(z_i|\alpha) = \sum_{k \in K} p_{kt} x_{ktz}^i + V_{i,t+1}(z_i|\alpha)
\]

\[
= \sum_{k \in K} p_{kt} x_{ktz}^i + \sum_{k \in K} p_{k,t+1} x_{k,t+1,z}^i + V_{i,t+2}(z_i|\alpha)
\]

If we continue in the same fashion, we obtain

\[
V_{it}(z_i|\alpha) = \sum_{k \in K} p_{kt} x_{ktz}^i + \cdots + p_{k,t-1} x_{k,t-1,z}^i + p_{k,t} x_{k}\tau^i.
\]
We can then simplify the difference

\[ V_i(z_i + 1|\alpha) - V_i(z_i|\alpha) = \sum_{k \in \mathcal{K}_i} p_{kt}(x_{ikt,z+1}^i - x_{iktz}^i) + \cdots + p_{k,T-1}(x_{ikt,T-1,z+1}^i x_{ikt,T-1,z}) \]

\[ = \sum_{k \in \mathcal{K}_i} \sum_{t=t+1}^{T-1} p_{ks}(x_{ikt,s,z+1}^i - x_{iktz}^i). \]

Note that the total number of accepted customers on day \( i \) at time period \( t \) is at most \( \min\{\theta_t \} \). Let \( \theta_t \) denote this bound on the total number of reservations. Together with (8) and (9), we have to introduce, for each \( k \in \mathcal{K}, i \in \mathcal{L}_k, t = 1, \cdots, T - 1 \) and \( z = 1, \cdots, \theta_t - 1 \), the following constraint into our linear programming problem:

\[ x_{iktz}^i \geq \alpha_{ikt} + \sum_{s \in \mathcal{L}_k} \sum_{t-t+1}^{T-1} p_{ks}(x_{ikt,s,z+1}^i - x_{iktz}^i). \]

The linear programming formulation of problem (5)-(6) is then given by

\begin{align}
\text{minimize} & \quad \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{L}_k} \sum_{t=1}^{T} p_{k1}x_{ikt0}^i \\
\text{subject to} & \quad x_{iktz}^i \geq \alpha_{ikt} + \sum_{l \in \mathcal{K}} \sum_{t-t+1}^{T-1} a_{il}p_{lt+1}(x_{ilsz+1}^i - x_{ilsz}^i), \\
& \quad k \in \mathcal{K}, i \in \mathcal{L}_k, t = 1, \ldots, T - 1, z = 0, \ldots, \theta_t - 1, \\
& \quad x_{iktz}^i \geq \alpha_{ikt}, \quad k \in \mathcal{K}, i \in \mathcal{L}_k, z = 1, \ldots, \theta_t - 1, \\
& \quad x_{iktz}^i \geq 0, \quad i \in \mathcal{L}, k \in \mathcal{K}, t \in \mathcal{T}, \\
& \quad \sum_{i \in \mathcal{L}_k} \alpha_{ikt} = f_k, \quad k \in \mathcal{K}, t \in \mathcal{T}, \\
& \quad \sum_{i \in \mathcal{L}_k} \beta_{ikt} = 0 \quad k \in \mathcal{K}, t \in \mathcal{T}, \\
& \quad \beta_{ikt} = 0, \quad k \in \mathcal{K}, t \in \mathcal{T}, \\
& \quad \alpha_{ikt} \geq 0, \quad i \in \mathcal{L}, k \in \mathcal{K}, t \in \mathcal{T}, z = 0, \ldots, \theta_t - 1.
\end{align}

LP formulation given by (10)-(15) has a block angular structure where independent blocks are linked by coupling constraints. In this problem, the blocks correspond to the different days and coupling constraints correspond to the fare allocation decisions. In other words, constraints (11)-(13) belong to the subproblems and constraints (14) belong to the master problem. Therefore, it is suitable to use the decomposition method of Dantzig and Wolfe (1960).

Some readers would notice that Kunnumkal and Topaloglu (2010) propose a similar revenue allocation model for customer choice problem in airline revenue management. However, in addition to the fare allocation constraint (3), they also introduce the set of constraints

\[ \sum_{i \in \mathcal{L}_k} \beta_{ikt} = 0 \quad k \in \mathcal{K}, t \in \mathcal{T}, \]

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 \( \sum_{i \in \mathcal{L}_k} \beta_{ikt} = 0 \). They discuss that using these penalties with allocated revenues provides a tighter upper bound on the optimal expected revenue. If we define penalty variables in our problem, 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 \( \alpha_{ikt} \) with \( \alpha_{ikt} + \beta_{ikt} \) in constraints (11) and (12) and we add the equality (16) into the model (10)-(15) as a new constraint. For ease of expression, we define this new model as \( LP_{(\tilde{\alpha}, \tilde{\beta})} \) and model (10)-(15) as \( LP_\alpha \). Next, we discuss that introducing penalty terms along with the constraints (16) 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 \( LP_{(\tilde{\alpha}, \tilde{\beta})}^* \) and \( LP_{\alpha}^* \) be the optimal values of models \( LP_{(\tilde{\alpha}, \tilde{\beta})} \) and \( LP_\alpha \), respectively. Then, we have \( LP_{\alpha}^* \leq LP_{(\tilde{\alpha}, \tilde{\beta})}^* \).

**Proof.** Let \((\tilde{\alpha}^*, \tilde{\beta}^*, \tilde{x}^*)\) be the optimal values of the decision variables in model \( LP_{(\tilde{\alpha}, \tilde{\beta})} \). We define \( \gamma_{ikt} = \tilde{\alpha}_{ikt} + \tilde{\beta}_{ikt} \) as the allocated revenue of product \( k \) on day \( i \) at time period \( t \). Then, the set \( \{(\gamma_{ikt}^*, \tilde{x}_{ikt}^*) : i \in L, k \in K, t \in T, z \leq C_i \} \) also presents the optimal solution for model \( LP_{(\tilde{\alpha}, \tilde{\beta})} \). By replacing \((\alpha_{ikt}, \tilde{x}_{ikt})\) with \((\gamma_{ikt}^*, \tilde{x}_{ikt}^*)\), we see that this solution is feasible for model \( LP_\alpha \). Therefore, we have \( LP_{\alpha}^* \leq LP_{(\tilde{\alpha}, \tilde{\beta})}^* \). \(\square\)

Problem (10)-(15) has \(|L||K|(\tau - 1)/2 + \tau|K|\) constraints which may increase the solution time as the planning horizon length increases. In the subsequent part of this section, we discuss several methods to solve such large-scale problems. The performances of these approaches shall be tested in our computational study (Section 6).

**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 \( \alpha_{ikt} \) with \( \alpha_{ik} \) 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 the 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 for the total number of constraints decreases to \(|L||K|(\tau - 1)/2 + |K|\).

**Iterative Heuristic.** This approach provides a heuristic solution method to problem (10)-(15). Instead of solving the problem (10)-(15) from scratch, we solve it for each time period starting from the last time period \( \tau \). We define \( \tilde{V}_t(z_i|\tilde{\alpha}) \) to denote the expected optimal revenue from \( t \) up to \( \tau \) of the alternative solution approach. To compute the optimal value of \( \tilde{V}_t(z_i|\tilde{\alpha}) \) for \( \tilde{\alpha} \) at any time period \( t \), we use the optimal values of the value functions, \( \tilde{V}_{i,t+1}^*(z_i|\tilde{\alpha}^*) \), at time period \( t + 1 \). In other words, at each time period \( t \) (starting from \( \tau \) to 1), we solve

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in L} \sum_{z_i=0}^{C_i-1} \tilde{V}_t(z_i|\tilde{\alpha}) \\
\text{subject to} & \quad \sum_{i \in L} \tilde{\alpha}_{ikt} = f_k, \quad \forall k \in K,
\end{align*}
\]

where

\[
\tilde{V}_t(z_i|\tilde{\alpha}) = \sum_{k \in K} p_{kt} \max\{\tilde{\alpha}_{ikt} + \tilde{V}_{t+1}(z_i + 1|\tilde{\alpha}^*), \tilde{V}_{i,t+1}(z_i|\tilde{\alpha}^*)\} + (1 - \sum_{k \in K} p_{kt}) \tilde{V}_{i,t+1}(z_i|\tilde{\alpha}^*). 
\]

This process continues recursively until we reach first time period. This method divides the problem (10)-(15) into \( \tau \) subproblems each of which can be solved very fast. The following lemma shows that this iterative heuristic also provides an upper bound on the problem (5)-(6).
Lemma 4.2 Let \( \alpha^* \) and \( \bar{\alpha}^* \) denote the optimal solutions to problem (10)-(15) and iterative heuristic method, respectively. Then, for all \( z_i \leq C_i \), we have \( \sum_{i \in L} V_{i1}(z_i|\alpha^*) \leq \sum_{i \in L} \bar{V}_{i1}(z_i|\bar{\alpha}^*) \).

Proof. Let \( \bar{\alpha}_i^* = \{\bar{\alpha}_{ik} : k \in K, i \in L_k\} \) is the set of optimal values of decision variables for IHM for time period \( t \in T \). Notice that \( \bar{\alpha}_i^* \) satisfies condition (3) for all time periods and for all products and hence, it is feasible but not necessarily an optimal to problem (10)-(15). Therefore, we have \( \sum_{i \in L} V_{i1}(z_i|\alpha^*) \leq \sum_{i \in L} \bar{V}_{i1}(z_i|\bar{\alpha}^*) \). \( \square \)

Deterministic Fare Allocation. Deterministic linear programming is a well-known approximation method used to compute optimal capacity policy for the hotel revenue management problem. We discuss deterministic models proposed for hotel capacity allocation in the appendix. By using the deterministic method used to compute optimal capacity policy for the hotel revenue management problem. We discuss Deterministic linear programming is a well-known approximation method used to compute optimal capacity policy for the hotel revenue management problem. We discuss Deterministic F are Allocation.

**Deterministic Fare Allocation.** Deterministic linear programming is a well-known approximation method used to compute optimal capacity policy for the hotel revenue management problem. We discuss deterministic models proposed for hotel capacity allocation in the appendix. By using the deterministic model (30)-(33) given in appendix, we can obtain a fare allocation policy. In order to do this, we decompose the model (30)-(33) by days in the service interval. Recently, Topaloglu (2012) proposes a method to decompose the network of airline alliances by the airlines. Inspired by this approach, we define a fictitious day \( \delta \) with infinite capacity and the set of service days becomes \( L \cup \{\delta\} \). To rewrite the model, we also define \( x_{ik} \) as the number of the reservations that we plan to accept for product \( k \) on day \( i \). Then, the problem (30)-(33) becomes

\[
\begin{align*}
\text{maximize} & \quad \sum_{k \in K} f_k x_{\delta k} \\
\text{subject to} & \quad \sum_{k \in K} a_{ik} x_{ik} \leq C_i, & i \in L, \\
& \quad x_{ik} \leq \sum_{t=1}^{\bar{\tau}} p_{kt}, & i \in L, k \in K, \\
& \quad x_{\delta k} - x_{ik} = 0, & i \in L, k \in K, \\
& \quad x_{ik} \geq 0, & i \in L, k \in K.
\end{align*}
\]

It is easy to see the equivalence between problems (30)-(33) and (17)-(21). Let \( \{\bar{\alpha}_{ik} : i \in L, k \in K\} \) be the optimal values of the dual variables associated with constraints (20). In the dual of problem (17)-(21), we have the constraint \( \sum_{i \in L_k} \bar{\alpha}_{ik} = f_k \) associated with the decision variable \( x_{\delta k} \). Therefore, we can use the dual variables \( \bar{\alpha}_{ik} \) to obtain a fare allocation.

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, 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 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 \( \zeta^s_t(y_s) \) denote the expected optimal revenue of pair \( s \) from \( t \) up to \( \tau \) given that the remaining capacity is \( y_s \) at time period \( t \). Then, the overall period-based decomposition
model becomes

\[
\begin{align*}
\text{maximize} & \quad \sum_{s \in S} \varphi^1_s(y_s) \\
\text{subject to} & \quad \sum_{s \in S} \hat{a}_{is} y_s \leq C_i, \quad i \in L, \\
& \quad y_s \in \mathbb{Z}_+, \quad s \in S,
\end{align*}
\]

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

\[
\varphi^t_s(y_s) = \sum_{k \in K_s} p_{kt} \max \left\{ f_k + \varphi^{t+1}_{s} (y_s - 1), \varphi^{t+1}_{s} (y_s) \right\} + (1 - \sum_{k \in K_s} p_{kt}) \varphi^{t+1}_{s} (y_s),
\]

where \( K_s \) denotes the set of products in pair \( s \). The boundary conditions are \( \varphi^t_s(0) = 0 \) for all \( t \) and \( \varphi^{t+1}_{s} (y_s) = 0 \) for all \( y_s \geq 0 \). It computes the optimal seat allocation policy for each pair \( s \in S \) 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 capacity. Therefore, the resulting model is partially dynamic (Birbil et al., 2014).

Lippman and Stidham (1977) show that the objective function \( y_s \mapsto \varphi^t_s(y_s) \) is discrete concave. Therefore, we can always replace it by a piece-wise linear concave function and then, rewrite the overall problem as a linear programming problem. We refer the reader to Birbil et al. (2014) for the details. Next, we present that problem (22) provides a lower bound on the maximum expected revenue over the whole planning horizon.

**Proposition 4.2** The optimal objective value of the problem (22) gives a lower bound on the optimal expected revenue of dynamic programming model given in (2). That is, we have \( \sum_{s \in S} \varphi^1_s(y_s) \leq \bar{J}_1(0) \).

**Proof.** Suppose that \( Y_{ks}, \forall k, s \) denote the random number of reservations accepted for product \( k \) in pair \( s \) over the planning horizon under the optimal policy of the decomposed dynamic programming model (22). Then we have

\[
\sum_{k \in K_s} Y_{ks} = Y_s, \quad s \in S,
\]

\[
\sum_{s \in S} a_{is} Y_s \leq C_i, \quad i \in L.
\]

Then, the total expected revenue under the optimal policy of the decomposed dynamic programming model is

\[
\sum_{s \in S} \varphi^1_s(Y_s) = \sum_{s \in S} \sum_{k \in K_s} f_k \mathbb{E}(Y_{ks}).
\]

This solution is clearly feasible. However, the exact dynamic model considers all possible combinations of capacity allocations over each product and hence, decomposed dynamic model provides a lower bound on the optimal expected revenue. Therefore, we have

\[
\sum_{s \in S} \sum_{k \in K_s} f_k \mathbb{E}(Y_{ks}) = \sum_{s \in S} \varphi^1_s(y_s) \leq \bar{J}_1(0).
\]

As we expressed in Proposition 4.1, day-based approximation given in (4) provides an upper bound on the dynamic model (2). Therefore, we can present an upper bound on the optimality gap by using
Proposition 4.1 and Proposition 4.2. This bound is useful to evaluate the error committed by solving approximate models instead of the complete model.

As we mentioned in Section 1, we ignore walk-in customers in our problem formulation. However, the arrival process of advance bookings and walk-ins are similar. The only difference is that the walk-ins arrive during the service period. By considering the arrival of walk-in customers as new products with higher prices, we can easily extend dynamic model (2) to include walk-ins. In this new problem, the number of products will be doubled. By updating the product set, fares and probabilities, we can use day-based methods to obtain decision policy for room allocation problem with walk-ins and advance bookings.

5. Mathematical Model with Stay-overs. To formulate stay-over customers, we need to keep track of the accepted reservations in each pair. However, 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 stay-over requests. Moreover, since stay-over probability ($\pi_s$) is pair dependent, we have to approximate the probability to apply day-based decomposition. As a result, day-based methods may perform poor for stay-over problem. Since pair-based decomposition breaks down the problem into the pairs, it is more applicable for the problem with stay-overs.

In our problem formulation, we again define $y_s$ as the allocated capacity for pair $s$. We also define $w_s$ as the reserved capacity for stay-over requests of pair $s$. The check-out date of pairs can be different and hence, we represent the check-out day of pair $s$ by $t_s$. Let $\phi^1_s(y_s, w_s)$ be the optimal expected revenue for pair $s$ with reserved capacities $y_s$ and $w_s$ from period $t$ to $t_s$. Then, the optimization problem becomes

$$\text{maximize } \sum_{s \in S} \phi^1_s(y_s, w_s) \quad (23)$$

subject to

$$\sum_{s \in S} \hat{a}_{is}(y_s + w_s) \leq C_i, \quad \forall i \in L, \quad (24)$$

$$w_s \leq y_s, \quad \forall s \in S, \quad (25)$$

$$y_s, w_s \in \mathbb{Z}^+, \quad \forall s \in S. \quad (26)$$

The objective function (23) is defined as $\phi^1_s(y_s, w_s) = \psi^1_s(0, 0)$, where the associated dynamic programming recursion is given for $t = 1, \ldots, t_s - 1$ by

$$\psi^t_s(x, z) = \sum_{k \in K_s} p_{kt} \max \left\{ f_k + \psi^{t+1}_s(x + 1, z + 1), \psi^{t+1}_s(x, z) \right\} + (1 - \sum_{k \in K_s} p_{kt})\psi^{t+1}_s(x, z) \quad (27)$$

under the conditions $x \leq y_s$ and $z \leq w_s$. The boundary condition is given by $\psi^{t_s}_s(x, z) = \theta_s \mathbb{E}[\min(w_s, B(\pi_s, x))]$. Here, we assume that $y_s \leq t_s$.

We note that the approach of Birbil et al. (2014) is applicable when the objective function (23) is discrete concave. The next lemma shows that this condition is satisfied thanks to the dynamic programming recursion in (27).

**Lemma 5.1** The differences given by $\psi^t_s(x + 1, z + 1) - \psi^t_s(x, z)$ are nonincreasing.
where \( x \) which is clearly nonincreasing in \( x \). If we introduce auxiliary variables \( \hat{\lambda} \) to denote the set of states in our model. Suppose the corresponding objective function values are given \( i \).

Again we can use the result in (Lippman and Stidham, 1977, pg. 237), \( x \mapsto \psi_s^i(x) \) is discrete concave, if \( x \mapsto u_s^+(x) \) is discrete concave. Note that when \( x < w_s \) the boundary condition becomes

\[ \theta_s \mathbb{E}[\min(w_s, B(\pi_s, x))] = \pi_s x, \]

which is clearly discrete concave in \( x \). This shows that \( \psi_s^i(x+1, z+1) - \psi_s^i(x, z) \) is nonincreasing whenever \( x < w_s \).

For the second part, let \( x \geq w_s \), then \( z = w_s \) and hence, we have

\[ \psi_s^i(x, z) = \sum_{k \in K_s} \min\{f_k + u_s^t(x+1), u_s^t(x)\} + (1 - \sum_{k \in K_s} p_{kt}) \psi_s^t(x, w_s) \]

As in the first part, we can define another one-dimensional function \( v_s^i(x) := \psi_s^i(x, z) \) as

\[ v_s^i(x) = \sum_{k \in K_s} \min\{f_k + v_s^t(x+1), v_s^t(x)\} + (1 - \sum_{k \in K_s} p_{kt}) v_s^t(x) \]

Again we can use the result in (Lippman and Stidham, 1977, Lemma 2, pg. 237) to establish that \( x \mapsto v_s^i(x) \) is discrete concave. This result follows, if we can show from the boundary condition that \( x \mapsto \theta_s \mathbb{E}[\min(w_s, B(\pi_s, x))] \) is discrete concave. Let us denote the \( n \)th Bernoulli trial by \( U_n \). Then,

\[ \mathbb{E}[\min(w_s, B(\pi_s, x))] - \mathbb{E}[\min(w_s, B(\pi_s, x+1))] = \pi_s \mathbb{E}[\min(w_s, B(\pi_s, x+1)) - \min(w_s, B(\pi_s, x))] \mathbb{E}[U_{x+1} \leq \pi_s] \]

\[ = \pi_s \mathbb{E}[\min(w_s, B(\pi_s, x) + 1) - \min(w_s, B(\pi_s, x))] \mathbb{E}[U_{x+1} \leq \pi_s] \]

\[ = \pi_s \mathbb{P}(B(\pi_s, x) \leq w_s - 1), \]

which is clearly nonincreasing in \( x \). This shows that \( \psi_s^i(x+1, z+1) - \psi_s^i(x, z) \) is also nonincreasing when \( x \geq w_s \). This completes the proof.

Next, we present the steps to construct the two dimensional piece-wise linear concave objective function (23). Suppose that there are \( m \) states in our dynamic programming model in (27). Then, let \( u_i \in \mathbb{Z}_+^2, \) \( i = 1, ..., m \) be the coordinates corresponding to these states. For ease of notation, we define \( U = \{1, ..., m\} \) to denote the set of states in our model. Suppose the corresponding objective function values are given by \( \lambda_i := \phi^i(u_i), i \in U \). Since the objective function is concave over \( \mathbb{R}_+^2 \), we have for all \( i \) and \( j \) pairs, the subgradient inequality

\[ \lambda_j \leq \lambda_i + \gamma^T(u_j - u_i). \]

If we introduce auxiliary variables \( \hat{\lambda}_i, i \in U \), then we can write an optimization problem as

\[ \text{minimize } \sum_{i \in U} |\hat{\lambda}_i - \lambda_i| \quad (28) \]

subject to

\[ \hat{\lambda}_j \leq \hat{\lambda}_i + \gamma^T(u_j - u_i), \quad i, j \in U, \]

where \( \hat{\lambda}_i \in \mathbb{R} \) and \( \gamma_i \in \mathbb{R}_+^2, i \in U \) are the decision variables. After loosing the absolute value function in a straightforward manner, we obtain a linear programming problem

\[ \text{minimize } \sum_{i \in U} z_i \]
subject to  
\[ \lambda_j \leq \hat{\lambda}_i + \gamma_i^T (u_j - u_i), \quad i, j \in U, \]
\[ -z_i \leq \lambda_i - \hat{\lambda}_i, \quad i \in U, \]
\[ z_i \geq \lambda_i - \hat{\lambda}_i, \quad i \in U, \]
\[ z_i \geq 0 \quad i \in U. \]

This problem can be solved very efficiently. Clearly, its optimal objective function value should be zero with the optimal solution \( \hat{\lambda}_i = \lambda_i \) and \( \gamma_i^* \) for \( i \in U \) if the objective function \( (y_s, w_s) \mapsto \phi_s^t(y_s, w_s; x_s) \) is concave. Then, we can write

\[ \phi_s^t(y_s, w_s; x_s) = \min_{i \in U} \left\{ \lambda_i + \gamma_i^T \left( \begin{array}{c} y_s \\ w_s \end{array} \right) - u_i \right\} \]

Equivalently,

\[ \phi_s^t(y_s, w_s; x_s) = \min_{i \in U} \left\{ \gamma_i^T \left[ \begin{array}{c} y_s \\ w_s \end{array} \right] + b_i \right\} \]

where \( b_i = \lambda_i - \gamma_i^T u_i, \quad i \in U \). By using this result, we can rewrite the model (23) - (26) as a linear programming problem.

As we explained at the end of Section 4.1, walk-in customers can be easily incorporated. In our original problem formulation, we assume that customers can only arrive during the reservation interval. On the other hand, walk-in requests for pair \( s \) will arrive between time periods \( \tau \) and \( t_s \). Again, we can update the model by considering the walk-in customers as new products with higher prices. By updating the product set, fares and probabilities, we can use pair-based decomposition method proposed in this section to obtain bid prices for hotel problem with advance bookings, walk-ins and stay-overs.

6. Computational Experiments. In this section, we present computational results that illustrate the performances of proposed solution methods for the hotel revenue management problem. We start with our simulation setup.

We assume that product requests arrive over discrete time periods \( T = \{1, ..., T\} \). 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 \( \rho \) given by

\[ \rho = \frac{\sum_{t \in T} \sum_{i \in L} \sum_{k \in K} a_{ik} p_{kt}}{\sum_{i \in L} C_i}. \]

In all our numerical experiments, we set the capacity of the hotel and the length of the planning horizon to 50 and 175, respectively. To compute the product fares, we define day-based prices, \( \sigma_i \). 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 hotel makes a discount \( (\kappa) \), if a customer request to stay more than one day. We compute the fares of these type of products by

\[ f_k = \kappa \sum_{i \in L} a_{ik} \sigma_i. \]

6.1 Models without Stay-overs. In this section, we compare the performances of decomposition methods proposed for dynamic model without stay-overs. We begin by describing the benchmark solution methods.
- **Linear Programming Formulation (LPF):** This strategy is related to our linear model given in (10)-(15). We solve this model to obtain the optimal values of \( \{ \alpha_{ikt} : i \in L, k \in K, t = 1, \ldots, \tau \} \). Then, we use these fare allocations to compute the decision policy using relation (7).

- **Time-independent fare allocation (TIF):** LPF computes the fare allocations for each day, each product and each time period. We redefine fare allocations by relaxing the time dependency and replace the decision variables \( \{ \alpha_{ikt} : i \in L, \forall k \in K, t = 1, \ldots, \tau \} \) with \( \{ \alpha_{ik} : i \in L, k \in K \} \) in problem (10)-(15). Then, we use these values to compute our accept-reject policy.

- **Iterative Heuristic Method (IHM):** In particular, we solve the linear model (10)-(15) for each time period starting from the last time period \( \tau \) to obtain the optimal solution of \( \alpha \). Then, we compute the decision policy by using these values. This method is an iterative method since we use the optimal values computed for time period \( t + 1 \) to obtain the optimal solution for time period \( t \).

- **Deterministic Linear Program (DLP):** These deterministic models are given in the appendix. This strategy solves the model (30)-(33) to obtain the optimal values of dual variables associated to capacity constraints (31). We use these dual variables as the bid prices for our accept-reject policy. To refine the bid prices in our implementation, we solve the problem (30)-(33) five times over the decision horizon. At each solution time, we replace the right side of constraints (31) and (32) with the remaining hotel daily capacities and the updated expected demand.

- **Deterministic Fare Allocation (DFA):** This strategy solves the model (17)-(21) to find the fare allocation. We use the dual variables associated to constraint (20) in model (17)-(21) as fare allocations of product \( k \). Then, we use these day-based fares to compute the decision policy of each day as in fare allocation heuristic. As in the DLP policy, we refine bid prices five times during the decision horizon.

- **Pair-Based Decomposition (PBD):** This strategy focuses on finding decision policy for each pair. We solve the model (22) to obtain the bid prices associated with the capacity constraints of the service days. As in the DLP policy, we use these bid prices in our decision policy and refine the bid prices five times during the decision horizon.

Our experimental design is based on various factors of the network size \( (n) \), the load factor \( (\rho) \) and the discount multiplier \( (\kappa) \). We test our models in two networks with 3 and 6 days, respectively. We use load factor values \( \rho \in \{1.2, 1.6\} \) corresponding to low and high loads. The discount factor values \( \kappa \in \{0.50, 0.75\} \) are used to represent low and high fares. We label our test problems by using all combinations of these parameters.

As we mentioned in Section 4.1, optimal objective values of the day-based decomposition methods provide upper bounds on the maximum total expected revenue obtained by the dynamic model in (2) (see Proposition 4.1 and Lemma 4.2). We first compare the upper bounds obtained by day-based solution methods. Computational results show that LPF consistently provides the tightest upper bounds. 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. Figure 2(a) presents the percentage gaps between the optimal objective value of LPF and the remaining methods. In these figures, the tuple \( (\rho, \kappa) \) in the horizontal axes shows a problem instance. The results show that TIF also provides relatively tight
upper bounds. For a majority of the test problems, the percentage gap between LPF and TIF is below 1%. An interesting result we have is that the upper bounds provided by DLP are tighter than those provided by TIF. Although deterministic model approximates the stochastic arrival process, it considers the whole network problem without decomposing it. On the other hand, TIF allocates the revenues of each product to days and uses this fixed allocation for each time period to compute dynamic policy. By comparing the percentage gaps between LPF and DLP with those between LPF and TIF, we conjecture that using fixed revenue allocations for each time period deteriorates the upper bound provided by day-based decomposition methods. On the other hand, the upper bound provided IHM is loose compared with the other dynamic policies. Recall that IHM decomposes the network problem twice; first by days and then, by time periods. This two step approximation deteriorates the optimal objective value of IHM.

In Section 4.2, we have discussed that the optimal objective function value of the pair-based decomposition problem (22) gives a lower bound for the complete dynamic model (see Proposition 4.2). Figure 2(b) presents the percentage gaps between the optimal objective values of LPF and PBD as well as DLP and PBD. We report gaps only for LPF and DLP as they give the tightest upper bounds in Figure 2(a). We observe that the percentage gaps in Figure 2(b) are on average less than 5%, and these gaps are mainly affected by the load factor. Note that these relative differences provide upper bound on the optimality gap. Therefore, these figures could be used for estimating the loss in revenue due to the lack of solving the exact dynamic model.

![Graphs](image)

(a) Upper bound percentage gaps  (b) Upper bounds on optimality gap

Figure 2: Percentage gaps on the maximum total expected revenue (n = 4)

Our simulation results are summarized in Figures 3 and 4. In these figures, we present the average revenues obtained by decomposition methods with respect to low and high load factor values (ρ) and the discount factor, κ. Figure 3 presents the average revenues for the test problems with three days. The results indicate that LPF and TIF perform better than the remaining solution methods. We also observe that the performances of LPF and TIF can become significantly close. Therefore, we conjecture that relaxing time dependency in the fare allocations provides a good solution policy to the problem (10)-(15). Moreover, it improves the solution time considerably. Comparing the total expected revenues of ALP and DFA, we observe that the performances of DFA and TIF are quite close. On the other hand, IHM and DLP compete for the fourth and fifth places. There are three test instances where IHM performs
noticeably worse than DLP. These instances correspond to the cases where arrival intensity and revenues are high. As we mentioned before, IHM decomposes the problem (10)-(15) into time periods and hence, it does not consider all possible combinations of fare allocations. Therefore, late arrival of expensive fare customers deteriorates the performance of IHM. On the other hand, PBD consistently provides the lowest expected revenues.

![Figure 3: The average revenues for varying load-factor ($\rho$) and discount factor ($\kappa$) parameters ($n = 4$)](image)

The number of days in a hotel network significantly affects the performances of decomposition methods. Figure 4 show the performances of different solution methods for the test problems with 6 days. To observe sufficient amount of customer arrival, we increase the number of time periods to 300. Due to the long computational time, we do not test the performances of LPF and TIF for these test instances. The first observation we have is that the performances of deterministic models deteriorate under this set-up. Comparing the performances of IHM, DFA and DLP in Figure 3 with those in Figure 4, we note that deterministic models cannot coordinate the network booking decisions well for large network sizes and high load factor values. We caution the reader to the total expected revenues obtained of DLP and PBD. When load factor is low, PBD performs better than DLP.

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 three benchmark strategies.

- **Stay-Over Bid Price (SBP):** This is the solution method that we present in Section 5. We solve the model (23)-(26) by relaxing the integrality constraints. The optimal values of the dual variables associated to capacity constraints are then used as the bid-prices for reservation policy.

- **Deterministic Stay-Over Model (DSM):** The deterministic model considering stay-over customers is given in the appendix. This strategy solves the model (34)-(38) to obtain the optimal values of dual variables associated with the capacity constraints. We use these dual variables as the bid prices for our decision policy.
- **Period-Based Decomposition (PBD):** This strategy focuses on advance bookings and ignores the stay-over customer requests. We solve the model (22) to obtain the bid prices for the decision policy. We use this strategy to measure the effect of stay-over customers.

Our experimental design is based on various factors of the network size \(n\), the load factor \(\rho\), the stay-over probability \(\pi_s\) and the stay-over fare coefficient \(r\). We scale the network to include \(n \in \{4, 7\}\) nodes. We use load factor values \(\rho \in \{1.2, 1.6\}\) corresponding to low and high loads. The stay-over probabilities for all pairs \(\pi_s \in \{0.10, 0.20\}\) are used to represent low and high stay-over demand. We compute price of stay-over reservation on day \(i\) by using the daily prices. For example, the price of stay-over request on day \(i\) is \((1 + r)\sigma_i\). To represent low and high stay-over fares, we use stay-over fare coefficient \(r \in \{0.15, 0.25\}\). 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 reoptimize the bid price policies five times over the decision horizon.

Our main results are summarized in Figure 5 and Figure 6. Comparing the average revenues in Figure 5, we observe that SBP generally outperforms the remaining solution methods. When we look into performances of SBP and DSM, we notice that the percentage differences are mostly affected by the load factor. Although both approaches use static bid prices for accept-reject decisions, DSM performs poor as it disregards the variance in the arrival process. An interesting result we have is that the performances of these two models are slightly affected by the increases in the stay-over probability and the price. When we compare the performances of PBD and SBD, we see that taking into account the stay-over requests 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 stay-overs brings additional revenue.

Figure 6 illustrates the average revenues for the test problems with 6 days. Comparing the plots for SBP and DSM in Figure 5 with those in Figure 6, we note that the average revenues obtained by SBP and DSM are significantly close. SBP decomposes the network into pairs when computing bid prices. As
the size of the network becomes larger, number of pairs increases rapidly and the performance of SBP deteriorates. This behavior can be attributed to the effect of capacity sharing. Since SBP partition the network problem into independent pairs, its performance is affected by the network cross-effect.

Figure 5: The average revenues for varying load-factor (ρ), stay-over probability and stay-over fare (κ) parameters (n = 4)

Figure 6: The average revenues for varying load-factor (ρ), stay-over probability and stay-over fare (κ) parameters (n = 7)
7. Conclusion. In this paper, we study the room allocation problem with walk-in and stay-over customers in a hotel network. Room allocation problem in hotel revenue management resembles to airline capacity allocation problem, however there are two important differences. First, demand structure in hotel RM is different. Customers can change the length of their reservation after their arrival or they can show-up without any reservation. Second, long-term stay is very common in hotel systems. Therefore, it is not a good practice to directly apply the models developed for airline problems to hotel context. In this study, we work on the dynamic room allocation problem. Due to the complexity of this problem, we concentrate on approximation methods. We analyze structural properties of the problem and present day-based and pair-based decomposition approaches which can handle walk-in and stay-over customers. In the first part of this paper, we utilize the decomposition idea of Kunnumkal and Topaloglu (2010) and study its linear programming formulation for the day-based decomposition model. We also work on the alternative solution approaches which are computationally efficient to solve. Since day-based decomposition generates independent subproblems for each day, it cannot store the number of reserved rooms for each product. Hence, incorporating stay-over customers becomes a challenge. In the second part, we work on stay-over extension and propose a pair-based decomposition model. By analyzing the structural properties of the model, we develop a solution method. The objective is to construct a two dimensional piecewise linear concave function. In our numerical results, we observe that our proposed model performs better than the deterministic model which is widely used in practice.

To the best of our knowledge, dynamic programming model of stay-over customers have not been proposed in the literature before. In this study, we assume that customers can request for at most one day extension on their reservations. Although, the stay-over probability of staying more than one day is low, relaxation of this assumption would result in a more realistic model. Moreover, hotel customers have the flexibility to leave at any time during their stay. These customers are known as early departures in hotel revenue management literature. Incorporation of early departure request in the check-in and check-out pair-based decomposition method would be another potential topic for future research.

Another future research direction would be investigating the error incurred by our fast heuristic approach. This approach divides the overall problem into smaller subproblems and each subproblem corre-
sponds to a different time period in the planning horizon. In our computational experiments, we observe that this method provides a good approximation and it is computationally efficient. To improve the performance of this solution method, we can decompose it into a smaller set of subproblems. In other words, instead of solving the overall problem at each time period in the planning horizon, we can divide the problem into block of time periods and find the optimal solution for each block. It would be interesting to investigate the dependency between the size of the subproblems and the incurred error.
Appendix A. Deterministic Models. An alternative solution to capacity allocation problem in hotel revenue management is to solve a deterministic linear program. DLP is formulated under the assumption that the arrivals of the product requests take on their expected values.

We start with the deterministic model without stay-over customers. Let $w_k$ be the number of reservations that we plan to accept for product $k$. Then, the deterministic linear program becomes

$$z_{DLP} = \text{maximize} \sum_{k \in K} f_k w_k \tag{30}$$

subject to

$$\sum_{k \in K} a_{ik} w_k \leq C_i, \quad i \in \mathcal{L}, \tag{31}$$

$$w_k \leq \sum_{t=1}^\tau p_{kt}, \quad k \in \mathcal{K}, \tag{32}$$

$$w_k \geq 0, \quad k \in \mathcal{K}. \tag{33}$$

DLP is a well-known solution method for the network revenue management problem which provides an upper bound on the optimal total expected revenue (Talluri and van Ryzin, 2004). Moreover, Talluri and van Ryzin (1998) discuss that the upper bound obtained by DLP is asymptotically optimal as the capacities on the days (flight legs) and the expected numbers of product requests increase linearly with the same rate. Following Proposition 4.1, we obtain that the model (5)-(6) is also asymptotically optimal.

Besides providing an upper bound, the optimal solution of DLP gives a decision policy. The dual variables associated to constraint 31 can be used to construct a policy to accept or reject the product requests. An important shortcoming of this model is that it ignores the variation in the random arrival process.

We can extend deterministic model (30)-(33) to include stay-over customer requests. Let $x_{ks}$ be the number of reservations that we plan to accept for product $k$ in pair $s$. Similarly, let $w_s$ be the number of reservations that we plan to accept for stay-over request of pair $s$. Then, the deterministic linear program has the following form:

$$\text{maximize} \sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} f_k x_{ks} + \sum_{s \in \mathcal{S}} \theta_s w_s \tag{34}$$

subject to

$$\sum_{s \in \mathcal{S}} \sum_{k \in \mathcal{K}_s} a_{is} x_{ks} + \hat{a}_{is} w_s \leq C_i, \quad i \in \mathcal{L} \tag{35}$$

$$x_{ks} \leq \sum_{t=1}^\tau p_{kt} \quad s \in \mathcal{S}, k \in \mathcal{K}_s, \tag{36}$$

$$w_s - \sum_{k \in \mathcal{K}_s} \pi_s x_{ks} \leq 0, \quad s \in \mathcal{S}, \tag{37}$$

$$x_{ks}, w_s \geq 0 \quad s \in \mathcal{S}, k \in \mathcal{K}_s, \tag{38}$$

The dual variables associated to capacity constraints are used as bid prices to construct a decision policy for the product requests.
References


