Online Multi-type Multiple Knapsack Problem

September 12, 2025

Abstract

Keywords: multi-type multiple knapsack problem, revenue management, .

1 Introduction

We will address the online multi-type multiple knapsack problem. The Online Multi-type Multiple Knapsack Problem (OMMKP) extends the classical knapsack problem to a dynamic and multi-dimensional setting, where items of distinct types arrive sequentially, and decisions to accept or reject them must be made immediately without knowledge of future arrivals. Each item is characterized by a type-dependent size and value, and must be placed into one of multiple knapsacks with non-identical capacities. This framework captures critical resource allocation challenges across various domains, including cloud computing, advertising systems, production planning, energy management, and network bandwidth allocation.

In cloud computing environments, service providers manage numerous heterogeneous servers (knapsacks) with varying computational resources (e.g., CPU, memory, or storage capacities). User tasks (items) arrive dynamically, each belonging to a specific type (e.g., compute-intensive, memory-intensive, or I/O-bound) with deterministic resource requirements (size) and a value (e.g., revenue or priority). The OMMKP models the online scheduling problem of allocating incoming tasks to suitable servers to maximize total profit or resource utilization while respecting capacity constraints. This is particularly relevant in server clusters with specialized hardware or reserved instances, where task heterogeneity and server diversity must be efficiently handled in real time.

In online advertising platforms, advertisers bid for impression slots (knapsacks) that exhibit diverse audience reach and engagement capacities (e.g., banner ads, video ads, or native ads). Ad requests (items) arrive in streams and are categorized by types (e.g., industry verticals or creative formats), each with a size (e.g., required impressions or click-through rate) and a value (e.g., bid price or expected revenue). The OMMKP formalism helps optimize the real-time assignment of ads to available slots to maximize platform revenue while satisfying contractual delivery constraints. The problem is compounded by the need to handle multiple ad types and slot categories under uncertain demand.

Modern manufacturing systems often involve multiple production lines (knapsacks) with different

capabilities and capacities (e.g., throughput rates or machine hours). Customer orders (items) arrive dynamically and can be classified into types (e.g., urgent, standard, or custom orders), each with a processing time (size) and a profit margin (value). The OMMKP captures the challenge of accepting and scheduling orders across production lines to maximize total profit while adhering to capacity limits. This is especially critical in make-to-order environments where order heterogeneity and line specialization necessitate intelligent online decision-making.

In smart grids or distributed energy systems, multiple storage units (knapsacks) such as batteries or renewable energy buffers have distinct storage capacities. Energy requests (items)—e.g., from electric vehicles or industrial consumers—arrive online and are typed by priority or flexibility (e.g., urgent, deferrable, or intermittent), each with an energy demand (size) and a willingness-to-pay (value). The OMMKP models the problem of allocating energy requests to storage units to maximize revenue or minimize grid instability. The heterogeneity of requests and storage units requires an online strategy that balances immediate rewards with future uncertainty.

In telecommunications networks, multiple communication channels (knapsacks)—such as fiber-optic links or wireless spectra—have different bandwidth capacities. Data flows (items) arrive dynamically and are categorized by service type (e.g., video streaming, VoIP, or bulk transfer), each requiring a certain bandwidth (size) and offering a quality-of-service value (e.g., priority or revenue). The OMMKP formulation aids in designing online algorithms that assign flows to channels to maximize total utility or adherence to service-level agreements. The problem is exacerbated by the diversity of flow types and channel characteristics in multi-protocol label switching (MPLS) or software-defined networking (SDN) contexts.

These applications underscore the broad applicability of the OMMKP in real-world systems where heterogeneous resources must be allocated dynamically under uncertainty. Developing efficient online algorithms for this problem—with guarantees on competitive ratio, scalability, and robustness—is therefore of significant theoretical and practical interest. This paper aims to address this gap by proposing novel strategies for the OMMKP and validating them in one or more of the above domains.

We develop several policies for online MMKP.

The rest of this paper is structured as follows. We review the relevant literature in Section 2. Then we introduce the key concepts of seat planning with social distancing and formulate the . Section 3 presents the bid-price and resolving dynamic primal policies to assign seats for incoming requests. Section 4 presents the experimental results and provide insights gained from implementing social distancing. Conclusions are shown in Section 5.

2 Literature Review

Multiple Knapsack problem (Martello and Toth, 1990) is a practical problem that presents unique challenges in various applications. While existing literature has primarily focused on deriving bounds or competitive ratios for general multiple knapsack problems (Khuri et al., 1994; Ferreira et al., 1996; Pisinger, 1999; Chekuri and Khanna, 2005), our work distinguishes itself by analyzing the specific struc-

ture and properties of solutions to the MMKP problem.

While the dynamic stochastic knapsack problem (e.g., Kleywegt and Papastavrou (1998, 2001), Papastavrou et al. (1996)) has been extensively studied in the literature, these works primarily consider a single knapsack scenario where requests arrive sequentially and their resource requirements and rewards are unknown until they arrive. In contrast, the online MMKP problem extends this framework by incorporating multiple knapsacks, adding another layer of complexity to the decision-making process. Research on the dynamic or stochastic multiple knapsack problem is limited. Perry and Hartman (2009) employs multiple knapsacks to model multiple time periods for solving a multiperiod, single-resource capacity reservation problem. This essentially remains a dynamic knapsack problem but involves time-varying capacity. Tönissen et al. (2017) considers a two-stage stochastic multiple knapsack problem with a set of scenarios, wherein the capacity of the knapsacks may be subject to disturbances. This problem is similar to the SPSR problem in our work, where the number of items is stochastic.

Generally speaking, the online MMKP problem relates to the revenue management (RM) problem, which has been extensively studied in industries such as airlines, hotels, and car rentals, where perishable inventory must be allocated dynamically to maximize revenue (van Ryzin and Talluri, 2005). Network revenue management (NRM) extends traditional RM by considering multiple resources (e.g., flight legs, hotel nights) and interdependent demand (Williamson, 1992). The standard NRM problem is typically formulated as a dynamic programming (DP) model, where decisions involve accepting or rejecting requests based on their revenue contribution and remaining capacity (Talluri and van Ryzin, 1998). However, a significant challenge arises because the number of states grows exponentially with the problem size, rendering direct solutions computationally infeasible. To address this, various control policies have been proposed, such as bid-price (Adelman, 2007; Bertsimas and Popescu, 2003), booking limits (Gallego and van Ryzin, 1997), and dynamic programming decomposition (Talluri and van Ryzin, 2006; Liu and van Ryzin, 2008). These methods typically assume that demand arrives individually (e.g., one seat per booking). However, in our problem, customers often request multiple units simultaneously, requiring decisions that must be made on an all-or-none basis for each request. This requirement introduces significant complexity in managing group arrivals (Talluri and van Ryzin, 2006).

A notable study addressing group-like arrivals in revenue management examines hotel multi-day stays (Bitran and Mondschein, 1995; Goldman et al., 2002; Aydin and Birbil, 2018). While these works focus on customer classification and room-type allocation, they do not prioritize real-time assignment. The work of Zhu et al. (2023), which addresses the high-speed train ticket allocation and processes individual seat requests and implicitly accommodates group-like traits through multi-leg journeys (e.g., passengers retaining the same seat across connected segments).

3 Online MMKP

Consider N knapsacks, with each knapsack j containing $c_j \in \mathbb{Z}^+$ capacities, for $j \in \mathcal{N} \coloneqq \{1, 2, \ldots, N\}$. There are M distinct item types, where each item type i, $i \in \mathcal{M} \coloneqq \{1, 2, \ldots, M\}$, requiring $w_i \in \mathbb{Z}^+$ consecutive size in one capacity. The profit of each item type i is $r_i \in \mathbb{Z}^+$. Suppose that the profit-weight

ratio is increasing monotone from 1 to M for type i. Requests arrive sequentially over time, and the seller must immediately decide whether to accept or reject each request upon arrival. If a request is accepted, the seller must also determine the specific knapsack to assign. Importantly, each item must be either fully accepted or entirely rejected; once the item is assigned to a knapsack, it cannot be altered.

To model this problem, we formulate it using dynamic programming approach in a discrete-time framework. Time is divided into T periods, indexed forward from 1 to T. We assume that in each period, at most one request arrives and the probability of an arrival for an item type i is denoted as λ_i^t, where $i \in \mathcal{M}$. The probabilities satisfy the constraint $\sum_{i=1}^{M} \lambda_i^t \leq 1$, indicating that the total probability of any item arriving in a single period does not exceed one. We introduce the probability $\lambda_0^t = 1 - \sum_{i=1}^{M} \lambda_i^t$ to represent the probability of no arrival in each period. To simplify the analysis, we assume that the arrivals of different item types are independent and the arrival probabilities remain constant over time. This assumption can be extended to consider dependent arrival probabilities over time if necessary.

The remaining capacity in each knapsack is represented by a vector $\mathbf{C} = (c_1, c_2, \ldots, c_N)$, where c_j denotes the capacity of knapsack j. Upon the arrival of an item type i at time t, the seller needs to make a decision denoted by $u_{i,j}^t$, where $u_{i,j}^t = 1$ indicates acceptance of type i in knapsack j during period t, while $u_{i,j}^t = 0$ signifies rejection of that type in knapsack j. The feasible decision set is defined as

$$U^t(\mathbf{L}) = \left\{ u_{i,j}^t \in \{0,1\}, \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \middle| \sum_{j=1}^{N} u_{i,j}^t \leq 1, \forall i \in \mathcal{M}; w_i u_{i,j}^t \mathbf{e}_j \leq \mathbf{C}, \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \right\}.$$

Here, $\mathbf{e}_j$ represents an N-dimensional unit column vector with the j-th element being 1, i.e., $\mathbf{e}_j = (\underbrace{0, \cdots, 0}_{j-1}, 1, \underbrace{0, \cdots, 0}_{N-j})$. The decision set $U^t(\mathbf{C})$ consists of all possible combinations of acceptance and rejection decisions for each type in each knapsack, subject to the constraints that at most one item of each type can be accepted in any knapsack, and the size of each accepted item must not exceed the remaining capacity of the knapsack.

Let $V^t(\mathbf{C})$ denote the maximum expected revenue earned by the optimal decision regarding item assignments at the beginning of period t, given the remaining capacity $\mathbf{C}$. Then, the dynamic programming formulation for this problem can be expressed as:

$$V^t(\mathbf{C}) = \max_{u_{i,j}^t \in U^t(\mathbf{C})} \left\{ \sum_{i=1}^{M} \lambda_i^t \Big(\sum_{j=1}^{N} r_i u_{i,j}^t + V^{t+1}(\mathbf{C} - w_i u_{i,j}^t \mathbf{e}_j) \Big) + \lambda_0^t V^{t+1}(\mathbf{C}) \right\} \tag{1}$$

with the boundary conditions $V^{T+1}(\mathbf{C}) = 0, \forall \mathbf{C}$, which implies that the revenue at the last period is 0 under any capacity. The initial capacity is denoted as $\mathbf{C}_0 = (C_1, C_2, \ldots, C_N)$. Our objective is to determine item assignments that maximize the total expected revenue during the horizon from period 1 to T, represented by $V^1(\mathbf{C}_0)$.

Solving the dynamic programming problem in equation (1) presents computational challenges due to the curse of dimensionality that arises from the large state space.

We propose our policy for assigning arriving requests. First, we employ the traditional bid-price control policy. Then, we present the bid-price control policy based on the patterns.

3.1 BPC Policy

Bid-price control is a classical approach discussed extensively in the literature on network revenue management. It involves setting bid prices for different types, which determine the eligibility of items to take the sizes. Bid-prices refer to the opportunity costs of taking one size of capacity. As usual, we estimate the bid price of one size by the shadow price of the capacity constraint corresponding to some knapsack. In this section, we will demonstrate the implementation of the bid-price control policy.

First, we give the formulation of the LP relaxation of the multi-type multiple knapsack problem (MMKP). Let x_{ij} represent the number of items of type i placed in knapsack j over T periods. The whole request of each item type during T periods is represented by the expected demand $d_i = \sum_{t=1}^{T} \lambda_i^t$. Then, the problem can be expressed as:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{M}\sum_{j=1}^{N} r_i x_{ij} && (2)\\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} \le d_i, \quad i \in \mathcal{M}, && (3)\\
& \sum_{i=1}^{M} w_i x_{ij} \le c_j, j \in \mathcal{N}, && (4)\\
& x_{ij} \ge 0, \quad i \in \mathcal{M}, j \in \mathcal{N}.
\end{aligned}
$$

The objective function (2) is to maximize the revenue. Constraint (3) ensures the total number of accommodated items does not exceed the number of requests for each type. Constraint (4) stipulates that the total size in each knapsack does not exceed its capacity.

The increasing nature of the ratio $\frac{r_i}{w_i}$ with respect to type i leads to preferential inclusion of larger items in the optimal fractional assignment. This intuitive property is illustrated in Proposition 1.

Proposition 1. *For the LP relaxation of the* MMKP *problem, there exists an index $\tilde{i}$ such that the optimal solutions satisfy the following conditions: $x_{ij}^* = 0$ for all j, $i = 1, \ldots, \tilde{i} - 1$; $\sum_{j=1}^{N} x_{ij}^* = d_i$ for $i = \tilde{i} + 1, \ldots, M$; $\sum_{j=1}^{N} x_{ij}^* = \frac{L - \sum_{i=\tilde{i}+1}^{M} d_i w_i}{w_{\tilde{i}}}$ for $i = \tilde{i}$.*

The dual of LP relaxation of the MMKP problem is:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{M} d_i z_i + \sum_{j=1}^{N} c_j \beta_j \\
\text{s.t.} \quad & z_i + \beta_j w_i \ge r_i, \quad i \in \mathcal{M}, j \in \mathcal{N} \\
& z_i \ge 0, i \in \mathcal{M}, \beta_j \ge 0, j \in \mathcal{N}.
\end{aligned} \tag{5}
$$

In (5), β_j can be interpreted as the bid-price for one size in knapsack j. A request is only accepted if the revenue it generates is no less than the sum of the bid prices of the sizes it uses. Thus, if $r_i - \beta_j w_i \ge 0$, meanwhile, the capacity allows, we will accept the item type i. And choose knapsack $j^* = \arg\max_j \{r_i - \beta_j w_i\}$ to allocate that item.

Lemma 1. *The optimal solution to problem (5) is given by* $z_1 = \ldots = z_{\tilde{i}} = 0$, $z_i = \frac{r_i w_{\tilde{i}} - r_{\tilde{i}} w_{\tilde{i}}}{w_{\tilde{i}}}$ *for* $i = \tilde{i}+1, \ldots, M$ *and* $\beta_j = \frac{r_{\tilde{i}}}{w_{\tilde{i}}}$ *for all* j.

According to Lemma 1, the decision inequality becomes $r_i - \beta_j w_i = r_i - \frac{r_{\tilde{i}}}{w_{\tilde{i}}} w_i \geq 0$. This establishes the threshold policy: reject item type $i, i < \tilde{i}$ and accept item type $i, i \geq \tilde{i}$.

Algorithm 1: Bid-Price Control

for $t = 1, \ldots, T$ **do**
  Observe a request of item type i;
  Solve problem (5) with $\boldsymbol{d}^{[t,T]}$ and $\mathbf{C}^t$;
  Obtain $\tilde{i}$ such that the aggregate optimal solution is $xe_{\tilde{i}} + \sum_{i=\tilde{i}+1}^{M} d_i^t e_i$;
  if $i \geq \tilde{i}$ *and* $\max_{j \in \mathcal{N}} c_j^t \geq w_i$ **then**
    Set $k = \arg\min_{j \in \mathcal{N}} \{c_j^t | c_j^t \geq w_i\}$ and break ties;
    Assign the item to knapsack k, let $c_k^{t+1} \leftarrow c_k^t - w_i$;
  else
    Reject the request;
  end
end

Let $z(\text{BPC})$, $z(\text{DLP})$ denote the optimal value of (5) and the LP relaxation of the MMKP problem with expected demand, respectively. Then we have $z(\text{BPC}) = z(\text{DLP})$.

However, the BPC policy has two drawbacks. First, the capacity feasibility need to be checked when to accept a request. Second, when capacity permits, the policy treats all knapsacks as equally preferable, making no distinction among them.

Example 1. *Consider* $M = 3$, $N = 4$, $w_1 = 3, w_2 = 4, w_3 = 5$, $r_1 = 4, r_2 = 6, r_3 = 8$, $\boldsymbol{C} = [7, 8, 8, 4]$, $T = 8$, *stationary arrival probability:* $\lambda_1 = \lambda_3 = \frac{1}{4}$, $\lambda_2 = \frac{1}{2}$. *Then the expected demand for each type is* $\boldsymbol{d} = (2, 4, 2)$.

For the traditional bid-price control, for each j, $\beta_j^* = \frac{4}{3}$. $z(BPC) = \frac{124}{3}$. *Two drawbacks: there is no difference among knapsacks. Even* $r_3 - \beta_4^* * w_3 > 0$, *it is infeasible for type 3 assigned in knapsack 4.*

3.2 BPC Policy Based on Patterns (BPP)

To account for the differences in placing items across knapsacks, we propose an enhanced dynamic programming (DP) formulation. The key idea, as detailed next, is to represent knapsack configurations using patterns rather than merely tracking the residual capacities.

A feasible pattern $\boldsymbol{h} = [h_1, \ldots, h_M]$ for knapsack j satisfies $\sum_{i=1}^{M} w_i h_i \leq c_j$. Suppose that $S(c_j)$ is the set of all feasible patterns for knapsack j.

Let $v^t(\boldsymbol{C})$ denote the maximal expected value-to-go at time t, given the remaining capacity $\boldsymbol{C}$. The enhanced dynamic programming formulation is as follows:

$$v^t(\boldsymbol{C}) \geq \mathbb{E}_{i\sim\lambda^t}\left[\begin{cases} \max\left\{\max_{j:\boldsymbol{h}\in S(c_j),h_i\geq 1}\left\{v^{t+1}\left(\boldsymbol{C}-e_j^T\cdot w_i\right)+r_i\right\}, v^{t+1}(\boldsymbol{C})\right\}, & \exists j, \boldsymbol{h}\in S(c_j), h_i\geq 1, \\ v^{t+1}(\boldsymbol{C}) & \text{otherwise.} \end{cases}\right] \tag{6}$$

The DP formulation involves two layers of maximization when there exists at least one knapsack j satisfying $\boldsymbol{h} \in S(c_j), h_i \geq 1$. The inner maximization evaluates the optimal placement of item type i across all feasible knapsacks j where the pattern h is feasible for knapsack j (i.e., $\boldsymbol{h} \in S(c_j)$) and the item type i can be accommodated (i.e., $h_i \geq 1$). The outer maximization compares the value of accepting i (via the inner maximization) and rejecting i (retaining $v^{t+1}(\boldsymbol{C})$). If no such j exists, the request i is rejected.

For notational convenience, we define q_i as follows:

$$q_i = \begin{cases} \max_{j:h\in S(c_j),h_i\geq 1}\left\{v^{t+1}\left(C-e_j^T w_i\right)+r_i\right\} & \text{if } \exists j \text{ satisfying } h\in S\left(c_j\right), h_i\geq 1 \\ 0 & \text{otherwise} \end{cases}$$

We can solve the following program to compute $v^1(\boldsymbol{C})$ for any given capacity $\boldsymbol{C}$:

$$\begin{aligned} \min \quad & v^1(\boldsymbol{C}) \\ \text{s.t.} \quad & v^t(\boldsymbol{C}) \geq \mathbb{E}_{i\sim\lambda^t}\left[\max\left\{q_i, v^{t+1}(\boldsymbol{C})\right\}\right], \\ & v^{T+1}(\boldsymbol{C}) \geq 0. \end{aligned} \tag{7}$$

Solving (7) remains computationally prohibitive. Following from the ADP approach, we approximate $v^t(\boldsymbol{C})$ as

$$\hat{v}^t(\boldsymbol{C}) = \theta^t + \sum_{j=1}^{N} \max_{\boldsymbol{h}\in S(c_j)} \{\sum_{i=1}^{M} \beta_{ij}^{\dagger} h_i\}. \tag{8}$$

The term $\beta_{ij}^{\dagger}$ can be regarded as the approximated value for each type i in knapsack j. Unlike traditional linear approximations, our approach retains the linear term θ^t but introduces a nonlinear component for each knapsack j. Specifically, we maximize the linear combination $\sum_{i=1}^{M} \beta_{ij}^{\dagger} h_i$ over the feasible set $S(c_j)$.

Our approximation extends classical linear ADP by incorporating resource-specific nonlinear terms through constrained maximization over feasible allocations. While similar separable corrections appear in resource allocation ADP (e.g., Powell, 2007), our explicit use of $\max_{\boldsymbol{h}\in S(c_j)}$ captures local constraints more directly.

Substituting (8) into (7), we have:

$$\theta^t - \theta^{t+1} = \hat{v}^t(\boldsymbol{C}) - \hat{v}^{t+1}(\boldsymbol{C}) \geq \sum_i \lambda_i^t \max\left\{q_i - v^{t+1}(\boldsymbol{C}), 0\right\} \tag{9}$$

For the case where there exists a knapsack j satisfying both conditions: $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^\dagger h_i$ and $h_i^* \geq 1$, we establish the value difference:

$$\begin{aligned}
&v^{t+1}(\boldsymbol{C} - e_j^T w_i) - v^{t+1}(\boldsymbol{C}) \\
=&\max_{\boldsymbol{h}\in S(c_j - w_i)} \{\sum_i \beta_{ij}^\dagger h_i\} - \max_{\boldsymbol{h}\in S(c_j)} \{\sum_i \beta_{ij}^\dagger h_i\} \\
=&-\beta_{ij}^\dagger \leq 0
\end{aligned}$$

The acceptance threshold is then defined as:

$$\alpha_i = \max\left\{\max_j \left\{r_i - \beta_{ij}^\dagger\right\}, 0\right\},$$

with $\alpha_i = 0$ when no qualifying knapsack j exists.

Let $\gamma_j = \max_{\boldsymbol{h}\in S(c_j)}\{\sum_i \beta_{ij}^\dagger h_i\}$. This yields:

$$\begin{aligned}
\theta^1 &= \sum_{t=1}^{T}(\theta^t - \theta^{t+1}) \geq \sum_t \sum_i \alpha_i \lambda_i^t = \sum_i d_i \alpha_i \\
\hat{v}^1(\boldsymbol{C}) &= \sum_i d_i \alpha_i + \sum_j \gamma_j
\end{aligned}$$

Since $\hat{v}^1(\boldsymbol{C})$ constitutes a feasible solution to (7), we have $\hat{v}^1(\boldsymbol{C}) \geq v^1(\boldsymbol{C}) = V^{DP}$.

If all j exist for $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^\dagger h_i, h_i^* \geq 1$, the corresponding bid-price problem can be expressed as:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{M} \alpha_i d_i + \sum_{j=1}^{N} \gamma_j \\
\text{s.t.} \quad & \alpha_i + \beta_{ij}^\dagger \geq r_i, \quad \forall i, j, \\
& \sum_{i=1}^{M} \beta_{ij}^\dagger h_i \leq \gamma_j, \quad \forall j, \boldsymbol{h} \in S(c_j), \\
& \alpha_i \geq 0, \forall i, \quad \beta_{ij}^\dagger \geq 0, \forall i, j \\
& \gamma_j \geq 0, \quad \forall j.
\end{aligned} \tag{10}$$

α_i represents marginal revenue for type i. $\beta_{ij}^\dagger$ represents the cost for type i assigned in knapsack j. γ_j represents the capacity cost associated with knapsack j.

When no knapsack j exists satisfying both $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^\dagger h_i$ and $h_i^* \geq 1$, the first set of constraints for (i, j) should be removed from the formulation (10). Although these constraints appear difficult to enforce in advance, the following lemma reveals that in practice we can avoid imposing additional restrictions. Simply solving problem (10) will inherently satisfy all required conditions.

Lemma 2. *Define $\mathcal{J} = \{j \in \mathcal{N} \mid r_i - \beta_{ij}^{\dagger *} > 0\}$. Then there exists a $j_0 \in \mathcal{J}$ such that:*

$$\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_{j_0})} \sum_i \beta_{ij_0}^{\dagger *} h_i, h_i^* \geq 1.$$

This lemma guarantees that when such a j_0 exists, the first set of constraints for i becomes active; otherwise, it remains inactive. Consequently, we have $z(\text{BPP}) = \hat{v}^1(\boldsymbol{C})$.

Then, the control policy becomes as follow. If $\alpha_i > 0$, accept the type i. Find knapsack $k = \arg\max_{j\in\mathcal{N}}\{r_i - \beta_{ij}^{\dagger}\}$, if multiple maximizers exist, assign the type i to a knapsack j satisfying:

$$\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^{\dagger *} h_i, h_i^* \geq 1.$$

If $\alpha_i = 0$, check whether there exists a knapsack j such that: $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^{\dagger *} h_i, h_i^* \geq 1$. If no such j exists, reject the type i; otherwise, accept the type i and assign it to knapsack j.

Let $z(\text{BPC})$ and $z(\text{BPP})$ denote the expected optimal value of (5) and (10), respectively.

Lemma 3. *For the optimal β_j^* in (5), there exist optimal $\beta_{ij}^{\dagger *}$ in (10) satisfying $\beta_{ij}^{\dagger *} \leq w_i \beta_j^*$ for all i. Furthermore, $z(BPC) \geq z(BPP)$.*

Both bid-price approaches give upper bounds on the value function at any state, meanwhile it follows that BPP provides a tighter approximation to the value function more accurately than BPC does.

Under the approximation (8), the BPP policy operates as follows:

Algorithm 2: Bid-Price Control Based on Patterns

for $t = 1, \ldots, T$ **do**
  Observe a request of item type i;
  Solve problem (10) with $\boldsymbol{d}^{[t,T]}$ and $\mathbf{L}^t$;
  if $r_i - \beta_{ij}^{\dagger} > 0$ **then**
    Set $k = \arg\max_{j\in\mathcal{N}}\{r_i - \beta_{ij}^{\dagger}\}$;
    If multiple ks exist, assign the type i to a knapsack j satisfying:

$$\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^{\dagger *} h_i, h_i^* \geq 1.$$

    Let $c_j^{t+1} \leftarrow c_j^t - w_i$;
  else
    if *There exists j such that* $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^{\dagger} h_i, h_i^* \geq 1$ **then**
      Assign the item in knapsack j, let $c_j^{t+1} \leftarrow c_j^t - w_i$;
    else
      Reject the request;
    end
  end
end

Unlike the traditional BPC, the BPP does not require explicit feasibility checks on capacity. How-

ever, it still needs to verify whether the optimal pattern contains the given request. This introduces a new challenge, as bid-price policies are inherently derived from a dual formulation, which may inherently omit key information preserved in the primal problem.

Example 2. *Continue with the above example:*

For the BPP, $\boldsymbol{\beta}_1^* = [4,4,4,6]$, $\boldsymbol{\beta}_2^* = [6,6,6,6]$, $\boldsymbol{\beta}_3^* = [10,8,8,8]$. $z(BPP) = 40$. $\alpha = [0,0,0]$, $\gamma = [10,12,12,6]$.

$r_1 - \boldsymbol{\beta}_1^* = [0,0,0,-2]$, $r_2 - \boldsymbol{\beta}_2^* = [0,0,0,0]$, $r_3 - \boldsymbol{\beta}_3^* = [-2,0,0,0]$.

Drawback: need to check whether j *exists for* $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_{j_0})} \sum_i \beta_{ij_0}^{\dagger *} h_i, h_i^* \geq 1$.

For example, $r_3 - \boldsymbol{\beta}_3^* = [-2,0,0,0]$ *indicates type 3 cannot be assigned in knapsack 1. While the generated patterns for knapsack 4 are* $[0,1,0], [1,0,0]$*, which do not contain* h_3*, type 3 can only be assigned to knapsack 2 or 3.*

While this verification is cumbersome under pure bid-price frameworks, the primal problem offers a more direct way to guarantee the existence of such a request-pattern match. To address this gap, we propose a dynamic primal formulation.

3.3 Dynamic Primal Based on Patterns

Let $y_{j\boldsymbol{h}}$ denote the proportion of pattern $\boldsymbol{h}$ used in knapsack j. The primal problem can be formulated as:

$$\begin{aligned} \max \quad & \sum_{i=1}^{M}\sum_{j=1}^{N} r_i x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} \leq d_i, \quad i \in \mathcal{M}, \\ & x_{ij} \leq \sum_{\boldsymbol{h}\in S(c_j)} h_i y_{j\boldsymbol{h}}, \quad i \in \mathcal{M}, j \in \mathcal{N}, \\ & \sum_{\boldsymbol{h}\in S(c_j)} y_{j\boldsymbol{h}} \leq 1, \quad j \in \mathcal{N}. \end{aligned} \tag{11}$$

The first set of constraints demonstrate that for each item type i, the sum of assigned items and unassigned items equals the total demand. The second set of constraints shows that the number of items of type i assigned in knapsack j is not larger than the sum of h_i (the count of type i items in pattern $\boldsymbol{h}$) weighted by the pattern proportions $y_{i\boldsymbol{h}}$. The total proportion of patterns uesd in knapsack j cannot exceed 1.

Lemma 4. *The optimal solution* x_{ij}^* *to* (11) *satisfies* $x_{ij}^* > 0$ *if and only if there exists a knapsack* j *such that*

$$\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^{\dagger} h_i, h_i^* \geq 1.$$

In contrast to Lemma 2, this lemma demonstrates the equivalence between the condition $x_{ij}^* > 0$ and the existance of a knapsack j satisfying $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta_{ij}^{\dagger} h_i, h_i^* \geq 1$. This equivalence eliminate the need for explicit existance verification.

Lemma 5. *$z(DLP) \geq V^{HO}$ results from the concave property.*

Consider the standard linear program: $\phi(\boldsymbol{d}) = \{\max c^T \boldsymbol{x} : A\boldsymbol{x} \leq \boldsymbol{d}, \boldsymbol{x} \geq \boldsymbol{0}\}$. Suppose that $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are two demand vectors, the optimal solution is $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. For any $\lambda \in [0,1]$, $\boldsymbol{d}_\lambda = \lambda \boldsymbol{d}_1 + (1-\lambda)\boldsymbol{d}_2$. Let $\boldsymbol{x}_\lambda = \lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2$, then $A\boldsymbol{x}_\lambda = A(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda \boldsymbol{d}_1 + (1-\lambda)\boldsymbol{d}_2 = \boldsymbol{d}_\lambda$. Thus, $\boldsymbol{x}_\lambda$ is a feasible solution for $\boldsymbol{d}_\lambda$. Then, $\phi(\boldsymbol{d}_\lambda) \geq \boldsymbol{c}^T \boldsymbol{x}_\lambda = \lambda \boldsymbol{c}^T \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{c}^T \boldsymbol{x}_2 = \lambda \phi(\boldsymbol{d}_1) + (1-\lambda)\phi(\boldsymbol{d}_2)$, which indicates $\phi(\boldsymbol{d})$ is concave. Let $\phi(\boldsymbol{d})$ indicate the optimal value of the linear relaxation of the SPDR problem. Substitute $\boldsymbol{x}$ with $y_{j\boldsymbol{h}}$ and view $y_{j\boldsymbol{h}}$ as the decision variables, then the concave property still holds for (11). $V^{\mathrm{HO}} = E[\phi(\boldsymbol{d})] \leq \phi(E[\boldsymbol{d}]) = z(\mathrm{DLP})$.

3.3.1 Solve the dynamic primal

The pattern $\boldsymbol{h}$ is efficient for knapsack j if and only if, for some $(\alpha_1, \ldots, \alpha_M, \gamma_j)$ (except that $\alpha_i = r_i, \forall i$), $\boldsymbol{h}$ is the optimal solution to

$$\max_{\boldsymbol{h}} \sum_{i=1}^{M} (r_i - \alpha_i) h_i - \gamma_j$$

To generate all efficient patterns, we need to solve the subproblem for each knapsack j:

$$\begin{aligned} \max \quad & \sum_{i=1}^{M} (r_i - \alpha_i) h_i - \gamma_j \\ \text{s.t.} \quad & \sum_{i=1}^{M} w_i h_i \leq c_j, \\ & h_i \in \mathbb{N}, \quad i \in \mathcal{M}. \end{aligned} \tag{12}$$

If the optimal value of (12) is larger than 0, the primal (11) reaches the optimal. Otherwise, a new pattern can be generated.

One important fact is that only efficient sets are used in the solution to (11). Specifically,

Lemma 6. *If $y^*_{j\boldsymbol{h}} > 0$ is the optimal solution to (11), then $\boldsymbol{h}$ is an efficient pattern.*

A pattern $\boldsymbol{h}$ is dominant if there is no distinct pattern $\boldsymbol{h}'$ where every component of $\boldsymbol{h}'$ is greater than or equal to the corresponding component of $\boldsymbol{h}$. The efficient pattern is a dominating pattern. (If $\alpha_i = r_i$, (11) reaches the optimal and no pattern will be generated.)

The relation between the capacity and the demand shows the different structure of the optimal solution.

Lemma 7. *When $\sum_{i=1}^{M} d_i w_i < \sum_{j=1}^{N} c_j$, we have $\gamma^*_j = 0, \forall j$, $\beta^{\dagger *}_{ij} = 0, \forall i, j$ and $\alpha^*_i = r_i, \forall i$. There exists at least one knapsack j such that $\sum_{\boldsymbol{h} \in S(c_j)} y^*_{j\boldsymbol{h}} < 1$.*

When $\sum_{i=1}^{M} d_i w_i \geq \sum_{j=1}^{N} c_j$, we have $\sum_{\boldsymbol{h} \in S(c_j)} y^*_{j\boldsymbol{h}} = 1, \forall j$.

Algorithm 3: Dynamic Primal

for $t = 1, \ldots, T$ **do**
  Observe a request of type i;
  if $c_j^t = w_i, \exists j$ **then**
    Assign the item to knapsack j;
    continue
  end
  Solve problem (11) with $\boldsymbol{d}^{[t,T]}$;
  Obtain an optimal solution x_{ij} ;
  if $\max_j\{x_{ij}\} > 0$ **then**
    Set $k = \arg\max_j\{x_{ij}\}$ and break ties;
    Assign the item to knapsack k, let $c_k^{t+1} \leftarrow c_k^t - w_i$;
  else
    Reject the request;
  end
end

Meanwhile, it guarantees feasible placement. Once a request is accepted, the policy ensures it can be assigned to a suitable knapsack without additional feasibility checks.

Example 3. *For the primal,* $\boldsymbol{x}_1^* = [1,1,0,0]$, $\boldsymbol{x}_2^* = [1,0,2,1]$, $\boldsymbol{x}_3^* = [0,1,0,0]$. *It indicates that type 1 can be assigned in knapsacks 1 or 2, type 2 cannot be assigned in knapsack 2, type 3 can only be assigned in knapsack 2.*

It may contain multiple efficient patterns for one knapsack. In this example, there is exactly one efficient pattern for each knapsack: $[1,1,0]$, $[1,0,1]$, $[0,2,0]$, $[0,1,0]$. *It shows that knapsack 1 can assign type 1 and 2, so on and so forth.*

Using x_{ij} *to make the decision is straightforward and easy to implement.*

Example 4. *Consider* $M = 3$, $N = 4$, $w_1 = 3, w_2 = 4, w_3 = 5$, $r_1 = 4, r_2 = 6, r_3 = 8$, $\boldsymbol{C} = [7,8,8,4]$, $T = 8$, *stationary arrival probability,* $\lambda_1 = \lambda_3 = \frac{1}{4}$, $\lambda_2 = \frac{1}{2}$. *Then the expected demand for each type is* $\boldsymbol{d} = (2,4,2)$.

For the traditional bid-price control, for each j, $\beta_j^* = \frac{4}{3}$. $z(BPC) = \frac{124}{3}$. *Two drawbacks: there is no difference among knapsacks. Even* $r_3 - \beta_4^* * w_3 > 0$, *it is infeasible for type 3 assigned in knapsack 4.*

For the BPP, $\boldsymbol{\beta}_1^* = [4,4,4,6]$, $\boldsymbol{\beta}_2^* = [6,6,6,6]$, $\boldsymbol{\beta}_3^* = [10,8,8,8]$. $z(BPP) = 40$. $\alpha = [0,0,0]$, $\gamma = [10,12,12,6]$.

$r_1 - \boldsymbol{\beta}_1^* = [0,0,0,-2]$, $r_2 - \boldsymbol{\beta}_2^* = [0,0,0,0]$, $r_3 - \boldsymbol{\beta}_3^* = [-2,0,0,0]$.

Drawback: need to check whether j *exists for* $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h} \in S(c_{j_0})} \sum_i \beta_{ij_0}^{\dagger *} h_i, h_i^* \geq 1$.

For example, $r_3 - \boldsymbol{\beta}_3^* = [-2,0,0,0]$ *indicates type 3 cannot be assigned in knapsack 1. While the generated patterns for knapsack 4 are* $[0,1,0], [1,0,0]$, *which do not contain* h_3, *type 3 can only be assigned to knapsack 2 or 3.*

For the primal, $\boldsymbol{x}_1^* = [1,1,0,0]$, $\boldsymbol{x}_2^* = [1,0,2,1]$, $\boldsymbol{x}_3^* = [0,1,0,0]$. *It indicates that type 1 can be*

assigned in knapsacks 1 or 2, type 2 cannot be assigned in knapsack 2, type 3 can only be assigned in knapsack 2.

It may contain multiple efficient patterns for one knapsack. In this example, there is exactly one efficient pattern for each knapsack: $[1,1,0]$, $[1,0,1]$, $[0,2,0]$, $[0,1,0]$. *It shows that knapsack 1 can assign type 1 and 2, so on and so forth.*

Using x_{ij} *to make the decision is straightforward and easy to implement.*

Asymptotic loss:

Lemma 8. *Loss:* $V_\theta^{HO} - V_\theta^{BPC} = O(\sqrt{\theta})$.

Lemma 9. *Loss:* $V_\theta^{HO} - V_\theta^{DPP} = O(1)$.

Let $T_i = sup\{t \leq T : \lambda_i^t > 0\}$.

3.4 Static BLC Policy

Booking limit control policy:

$$\begin{aligned} \max \quad & \sum_{i=1}^{M}\sum_{j=1}^{N} r_i x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} \le d_i, \quad i \in \mathcal{M}, \\ & \sum_{i=1}^{M} w_i x_{ij} \le L_j, j \in \mathcal{N}, \end{aligned} \tag{13}$$

Let $d_i^* = \sum_j x_{ij}^*$, x_{ij}^* is an integral optimal solution to (13) with $d_i = \sum_t p_i^t$ (Expected demand).

Let d_i indicate the number of type i during time T. $d_i = \sum_t \mathbf{1}_{i_t=i}$. Let $val(I;\{d_i\})$ denote the optimal objective value of (13).

$V^{BL}(I) = E_{\{d_i\}}[\sum_i (n_i-\delta)\min\{d_i^*, d_i\}]$, $V^{OPT}(I) = E_{\{d_i\}}[val(I;\{d_i\})] \le val(I;\{E[d_i]\})$.

$val(I;\{d_i\})$ is concave in d_i.

$$\begin{aligned} & V^{OPT}(I) - V^{BL}(I) \\ \le & val(I;\{E[d_i]\}) - V^{BL}(I) \\ = & val(I;\{E[d_i]\}) - val(I;\{\lfloor E[d_i]\rfloor\}) + val(I;\{\lfloor E[d_i]\rfloor\}) - E_{\{d_i\}}[\sum_i (n_i-\delta)\min\{d_i^*, d_i\}] \\ \le & \sum_i (n_i-\delta) + N\sum_i i + E_{\{d_i\}}[\sum_i (n_i-\delta)(d_i^* - \min\{d_i^*, d_i\})] \\ = & \sum_i (n_i-\delta) + N\sum_i i + E_{\{d_i\}}[\sum_i \frac{1}{2}(n_i-\delta)(d_i^* - d_i + |d_i^* - d_i|)] \\ \overset{(a)}{\le} & \sum_i (n_i-\delta) + N\sum_i i + \frac{1}{2}\sum_i (n_i-\delta)(d_i^* - E[d_i] + |d_i^* - E[d_i]| + \sqrt{\mathrm{Var}[d_i]}) \\ \le & \sum_i (n_i-\delta) + N\sum_i i + \frac{1}{2}\sum_i (n_i-\delta)\sqrt{\mathrm{Var}[d_i]} \\ \le & \sum_i (n_i-\delta) + N\sum_i i + \frac{1}{2}\sum_i (n_i-\delta)\sqrt{Tp_i(1-p_i)} = O(\sqrt{T}) \end{aligned}$$

Thus, $\lim_{T\to\infty}(V^{OPT}(I) - V^{BL}(I))/T \to 0$.

$val(I;\{E[d_i]\}) - val(I;\{\lfloor E[d_i]\rfloor\}) \le val(I;\{\lceil E[d_i]\rceil\}) - val(I;\{\lfloor E[d_i]\rfloor\}) = \sum_i (n_i-\delta)$

$LP - IP \le \sum_i \sum_j (n_i-\delta)(x_{ij}^* - \lfloor x_{ij}^*\rfloor) \le N\sum_i i \Rightarrow val(I;\{\lfloor E[d_i]\rfloor\}) \le IP + N\sum_i i$.

$IP = \sum_i \sum_j (n_i-\delta)x_{ij}^* = \sum_i (n_i-\delta)d_i^*$

(a) results from the following inequalities: $|d_i^* - d_i| = |(d_i^* - E[d_i]) + (E[d_i] - d_i)| \le |d_i^* - E[d_i]| + |d_i - E[d_i]|$. Take the expectation, we have $E[|d_i^* - d_i|] \le |d_i^* - E[d_i]| + E[|d_i - E[d_i]|]$. $E[|d_i - E[d_i]|] \le \sqrt{\mathrm{Var}[d_i]}$(Since $E[|X|] \le \sqrt{E[X^2]}$). $d_i^* \le E[d_i]$.

Surrogate relaxation (0-1 single):

$$\max \quad \sum_{i=1}^{M} r_i x_i \tag{14}$$

$$\text{s.t.} \quad x_i \le d_i, \quad i \in \mathcal{M}, \tag{15}$$

$$\sum_{i=1}^{M} n_i x_i \le L. \tag{16}$$

LP optimal solution: $[0, \ldots, 0, X_{\tilde{i}}, d_{\tilde{i}+1}, \ldots, d_M]$, $X_{\tilde{i}} = \frac{L - \sum_{i=\tilde{i}+1}^{M} d_i w_i}{n_{\tilde{i}}}$.

One feasible IP optimal solution: $[0, \ldots, 0, \lfloor X_{\tilde{i}} \rfloor, d_{\tilde{i}+1}, \ldots, d_M]$.

$LP - IP \le \tilde{i}(X_{\tilde{i}} - \lfloor X_{\tilde{i}} \rfloor)$

$$\begin{aligned}
&V^{OPT}(I) - V^{BL}(I) \\
\le &val(I; \{E[d_i]\}) - V^{BL}(I) \\
= &val(I; \{E[d_i]\}) - val(I; \{\lfloor E[d_i] \rfloor\}) + val(I; \{\lfloor E[d_i] \rfloor\}) - E_{\{d_i\}}[\sum_i (n_i - \delta) \min\{d_i^*, d_i\}] \\
\le &\sum_i (n_i - \delta) + \tilde{i}(X_{\tilde{i}} - \lfloor X_{\tilde{i}} \rfloor) + E_{\{d_i\}}[\sum_i (n_i - \delta)(d_i^* - \min\{d_i^*, d_i\})] \\
\le &\sum_i (n_i - \delta) + \tilde{i}(X_{\tilde{i}} - \lfloor X_{\tilde{i}} \rfloor) + \frac{1}{2} \sum_i (n_i - \delta) \sqrt{T p_i (1 - p_i)}
\end{aligned}$$

$E[\text{loss}] = V^{\text{off}} - V_\pi^{on} \ge V^{\text{opt}} - V_\pi^{on}$

One sample path. d^r realization of M types.

$V_t(l) = \sum_{i=\hat{i}+1}^{M} r_i d_i^r + r_{\hat{i}}(l - \sum_{i=\hat{i}+1}^{M} d_i^r)$

Let $V^{\text{OPT}}(I)$ denote the expected value under offline optimal policy (relaxed) during T periods for instance I (capacity, probability distribution).

The revenue loss between the static deterministic heuristic and the optimal is bounded by $C\sqrt{T}$.

Let γ_i, γ_i^0 denote the number of type i accepted and rejected by some heuristic policy, respectively.

$$\begin{aligned}
\text{OPT}(L, \hat{d}, \gamma): \quad \max \quad & \sum_{i=1}^{M} r_i x_i \\
\text{s.t.} \quad & x_i^0 + x_i = \hat{d}_i, \quad i \in \mathcal{M}, \\
& x_i \ge \gamma_i, \quad i \in \mathcal{M}, \\
& x_i^0 \ge \gamma_i^0, \quad i \in \mathcal{M}, \\
& \sum_{i=1}^{M} w_i x_i \le L.
\end{aligned}$$

Heuristic policy: At time t, solve problem (14) with $d_i = d_i^t = (T - t) * p_i$, $L = L^t$. When $x_i \ge 1$ for the request of type i, accept the request.

$d^{[1,T]}$ is the demand realization during $[1,T]$. $\gamma^{[1,t)}$ represents the number of requests rejected and accpeted by some heuristic policy during $[1,t)$.

$OPT(L, d^{[1,T]}, \gamma^{[1,t+1)})$ can be interpreted as the total reward obtained under a virtual policy where we first follow the heuristic policy during $[1,t+1)$ and then from time $t+1$ we follow the optimal solution assuming that we know the future demands.

For one sample path of the requests, the revenue loss can be decomposed into T increments.

$$\begin{aligned}
&OPT(C, d^{[1,T]}, 0) - OPT(C, d^{[1,T]}, \gamma^{[1,T]}) \\
=&\sum_{t=1}^{T}[OPT(C, d^{[1,T]}, \gamma^{[1,t)}) - OPT(C, d^{[1,T]}, \gamma^{[1,t+1)})]
\end{aligned}$$

Let $c_j^t = c_j - \sum_{i=1}^{M} w_i \gamma_{ij}^{[1,t)}$.

The expected revenue loss can be upper bounded:

$$\begin{aligned}
&E[OPT(C, d^{[1,T]}, 0) - OPT(C, d^{[1,T]}, \gamma^{[1,T]})] \\
\leq& l \sum_{t=1}^{T} P(OPT(C, d^{[1,T]}, \gamma^{[1,t)}) - OPT(C, d^{[1,T]}, \gamma^{[1,t+1)}) > 0) \\
=&(n_M - \delta) \sum_{t=1}^{T} P(OPT(L^t, d^{[t,T]}, 0) - OPT(L^t, d^{[t,T]}, \gamma^{[t,t+1)}) > 0) \\
\leq&(n_M - \delta) \sum_{t=1}^{T} P(x_{i^t}^{*,t} < 1) \\
=&(n_M - \delta) \sum_{t=T_0}^{T} P(x_{i^t}^{*,t} < 1) \\
\leq&(n_M - \delta) \max_i \{\frac{1}{p_i}\}
\end{aligned}$$

Lemma 10. $OPT(L^1, \hat{d} + d^{[1,t_2)}, \gamma^{[1,t_2)}) = \sum_i (n_i - \delta)\gamma_i^{[1,t_1)} + OPT(L^t, \hat{d} + d^{[t_1,t_2)}, \gamma^{[t_1,t_2)})$

For any optimal solution x^* of $OPT(L^t, \hat{d} + d^{[t_1,t_2)}, \gamma^{[t_1,t_2)})$, $x^* + \gamma^{[1,t_1)}$ is a feasible solution of $OPT(L^1, \hat{d} + d^{[1,t_2)}, \gamma^{[1,t_2)})$. For any optimal solution x^* of $OPT(L^1, \hat{d} + d^{[1,t_2)}, \gamma^{[1,t_2)})$, $x^* - \gamma^{[1,t_1)}$ is a feasible solution of $OPT(L^t, \hat{d} + d^{[t_1,t_2)}, \gamma^{[t_1,t_2)})$ because $x^* - \gamma^{[1,t_1)} \geq \gamma^{[1,t_2)} - \gamma^{[1,t_1)} = \gamma^{[t_1,t_2)}$.

The first inequality results from $E[A] \leq r_M E[\mathbf{1}_{A>0}] = r_M P(A > 0)$.

The first equation follows from Lemma. (Let $t_1 = t_2 = t$, $\hat{d} = d^{[t,T]}$; let $t_1 = t, t_2 = t+1$, $\hat{d} = d^{[t+1,T]}$).

The second equation is as follows. If $x_{i^t}^{*,t} \geq 1$, then $x^{*,t}$ is still feasible for $OPT(L^t, d^{[t,T]}, \gamma^{[t,t+1)})$. (Because the optimal policy)

$x_{i^t}^{*,t}$ is the optimal solution for OPT$(L^t, d^{[t,T]}, 0)$ at time t.

Let $T - T_0 = \max_i\{\frac{1}{p_i}\}$

For N rows,

$$
\begin{aligned}
OPT(\boldsymbol{L}, \hat{d}, \gamma): \quad \max \quad & \sum_{i=1}^{M}\sum_{j=1}^{N} r_i x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{N} x_{ij} + x_{i0} = \hat{d}_i, \quad i \in \mathcal{M}, \\
& \sum_{j=1}^{N} x_{ij} \geq \gamma_i, \quad i \in \mathcal{M}, \\
& x_{i0} \geq \gamma_i^0, \quad i \in \mathcal{M}, \\
& \sum_{i=1}^{M} w_i x_{ij} \leq c_j, \quad j \in \mathcal{N}.
\end{aligned}
$$

4 Computational Experiments

4.1

5 Conclusion

We study the seating management problem under social distancing requirements. Specifically, we first consider the seat planning with deterministic requests problem. To utilize all seats, we introduce the full and largest patterns. Subsequently, we investigate the seat planning with stochastic requests problem. To tackle this problem, we propose a scenario-based stochastic programming model. Then, we utilize the Benders decomposition method to efficiently obtain a seat plan, which serves as a reference for dynamic seat assignment. Last but not least, to address the seat assignment with dynamic requests, we introduce the SPBA policy by integrating the relaxed dynamic programming and the group-type control allocation.

We conduct several numerical experiments to investigate various aspects of our approach. First, we compare SPBA with three benchmark policies: BPC, BLC, and RDPH. Our proposed policy demonstrates superior and more consistent performance relative to these benchmarks. All policies are assessed against the optimal policy derived from a deterministic model with perfect foresight of request arrivals.

Building upon our policies, we further evaluate the impact of implementing social distancing. By introducing the concept of the threshold of request-volume to characterize situations under which social distancing begins to cause loss to an event, our experiments show that the threshold of request-volume depends mainly on the mean of the group size. This leads us to estimate the threshold of request-volume by the mean of the group size.

Our models and analyses are developed for the social distancing requirement on the physical distance and group size, where we can determine a threshold of occupancy rate for any given event in a venue, and a maximum achievable occupancy rate for all events. Sometimes the government may impose a maximum allowable occupancy rate to tighten the social distancing requirement. This maximum allowable rate is effective for an event if it is lower than the threshold of occupancy rate of the event. Furthermore,

the maximum allowable rate becomes redundant if it is higher than the maximum achievable rate for all events. These qualitative insights are stable concerning the tightness of the policy as well as the specific characteristics of various venues.

Future research can be pursued in several directions. First, when seating requests are predetermined, a scattered seat assignment approach can be explored to maximize the distance between adjacent groups when sufficient seating is available. Second, more flexible scenarios could be considered, such as allowing individuals to select seats based on their preferences. Third, research could also investigate scenarios where individuals arrive and leave at different times, adding an additional layer of complexity to the problem.

References

Adelman, D., 2007. Dynamic bid prices in revenue management. Operations Research 55, 647–661.

Aydin, N., Birbil, S.I., 2018. Decomposition methods for dynamic room allocation in hotel revenue management. European Journal of Operational Research 271, 179–192.

Bertsimas, D., Popescu, I., 2003. Revenue management in a dynamic network environment. Transportation Science 37, 257–277.

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.

Chekuri, C., Khanna, S., 2005. A polynomial time approximation scheme for the multiple knapsack problem. SIAM Journal on Computing 35, 713–728.

Ferreira, C.E., Martin, A., Weismantel, R., 1996. Solving multiple knapsack problems by cutting planes. SIAM Journal on Optimization 6, 858–877.

Gallego, G., van Ryzin, G., 1997. A multiproduct dynamic pricing problem and its applications to network yield management. Operations Research 45, 24–41.

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 1, 207–219.

Khuri, S., Bäck, T., Heitkötter, J., 1994. The zero/one multiple knapsack problem and genetic algorithms, in: Proceedings of the 1994 ACM symposium on Applied computing, pp. 188–193.

Kleywegt, A.J., Papastavrou, J.D., 1998. The dynamic and stochastic knapsack problem. Operations Research 46, 17–35.

Kleywegt, A.J., Papastavrou, J.D., 2001. The dynamic and stochastic knapsack problem with random sized items. Operations Research 49, 26–41.

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

Martello, S., Toth, P., 1990. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc.

Papastavrou, J.D., Rajagopalan, S., Kleywegt, A.J., 1996. The dynamic and stochastic knapsack problem with deadlines. Management Science 42, 1706–1718.

Perry, T.C., Hartman, J.C., 2009. An approximate dynamic programming approach to solving a dynamic, stochastic multiple knapsack problem. International Transactions in Operational Research 16, 347–359.

Pisinger, D., 1999. An exact algorithm for large multiple knapsack problems. European Journal of Operational Research 114, 528–541.

van Ryzin, G.J., Talluri, K.T., 2005. An introduction to revenue management, in: Emerging Theory, Methods, and Applications. INFORMS, pp. 142–194.

Talluri, K., van Ryzin, G., 1998. An analysis of bid-price controls for network revenue management. Management Science 44, 1577–1593.

Talluri, K.T., van Ryzin, G.J., 2006. The Theory and Practice of Revenue Management. Springer Science & Business Media.

Tönissen, D.D., Van den Akker, J., Hoogeveen, J., 2017. Column generation strategies and decomposition approaches for the two-stage stochastic multiple knapsack problem. Computers & Operations Research 83, 125–139.

Williamson, E.L., 1992. Airline network seat inventory control: Methodologies and revenue impacts. Ph.D. thesis. Massachusetts Institute of Technology.

Zhu, F., Liu, S., Wang, R., Wang, Z., 2023. Assign-to-seat: Dynamic capacity control for selling high-speed train tickets. Manufacturing & Service Operations Management 25, 921–938.