Online Multi-type Multiple Knapsack Problem

October 14, 2025

Abstract

We consider the online multi-type multiple knapsack problem.

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, and energy management.

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 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. The heterogeneity of requests and storage units requires an online strategy that balances immediate rewards with future uncertainty.

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 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. Section 3 presents the bid-price and resolving dynamic primal policies to assign seats for incoming requests. Section 5 presents the experimental results and provides insights gained from implementing dynamic primal. Conclusions are shown in Section 6.

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 structure 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 a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of knapsacks, where each knapsack j has a capacity $c_j \in \mathbb{Z}^+$. There is also a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of distinct item types. Each item of type i has a size $w_i \in \mathbb{Z}^+$ and yields a profit $r_i \in \mathbb{Z}^+$ when placed entirely into a knapsack. The item types are ordered such that their profit-to-weight ratios, r_i/w_i, are monotonically increasing in i.

Requests for these items arrive sequentially. Upon the arrival of a request (which specifies its type), the seller must immediately decide whether to accept or reject it. If accepted, the seller must also assign it to a specific knapsack with sufficient remaining capacity. Each item must be placed whole into a single knapsack; partial assignments or reassignments are not permitted.

To model this problem, we adopt a dynamic programming framework based on discrete time periods $t = 1, 2, \ldots, T$. In each period, at most one request arrives. Let λ_i^t denote the probability that a request for an item of type $i \in \mathcal{M}$ arrives at time t. These probabilities satisfy $\sum_{i=1}^{M} \lambda_i^t \leq 1$ for all t, and we define $\lambda_0^t = 1 - \sum_{i=1}^{M} \lambda_i^t$ as the probability of no arrival in period t. Arrival events are assumed to be independent across time periods.

The system state is the vector of remaining capacities $\mathbf{C} = (c_1, \ldots, c_N)$. When an item of type i arrives, the seller chooses a decision variable $u_{i,j}^t \in \{0,1\}$ for each knapsack j. The feasible set $U^t(\mathbf{C})$ is defined by:

$$U^t(\mathbf{C}) = \left\{ u_{i,j}^t \in \{0,1\} \;\middle|\; \begin{array}{ll} \text{(a)} & \sum_{j=1}^N u_{i,j}^t \le 1 \quad \forall i \in \mathcal{M} \\ \text{(b)} & w_i u_{i,j}^t \le c_j \quad \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \end{array} \right\}.$$

Constraint (a) ensures the item is assigned to at most one knapsack. Constraint (b) ensures that if the item is assigned to knapsack j ($u_{i,j}^t = 1$), its weight w_i does not exceed c_j. The original vector form of this constraint, $w_i u_{i,j}^t \mathbf{e}_j \le \mathbf{C}$ (where $\mathbf{e}_j$ is the j-th standard basis vector), is mathematically equivalent to (b).

Let $v^t(\mathbf{C})$ denote the value function, representing the maximum expected revenue obtainable from period t onward, given the current remaining capacity vector $\mathbf{C}$. The Bellman equation is given by:

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

The boundary condition is $v^{T+1}(\mathbf{C}) = 0$ for all $\mathbf{C} \ge \mathbf{0}$, indicating that no more revenue can be earned after the final period T.

Let $\mathbf{C}_0 = (C_1, C_2, \ldots, C_N)$ be the initial capacity vector. The objective is to compute $v^1(\mathbf{C}_0)$, the maximum total expected revenue over the entire horizon from $t = 1$ to $t = T$, and to find the policy of item assignments that achieves this value.

Solving the dynamic programming problem presented in Equation (1) is computationally intractable for realistic problem sizes due to the curse of dimensionality inherent in the large state space.

To overcome this challenge, we propose a heuristic assignment policy. We first outline a traditional bid-price control policy. We then enhance this approach by introducing a novel bid-price control policy that leverages patterns.

3.1 BPC Policy

Bid-price control is a classical and widely studied methodology in network revenue management. The core idea is to set thresholds, known as bid prices, that represent the opportunity cost of consuming one unit of capacity. An item is accepted only if its revenue exceeds the estimated opportunity cost of the capacity it requires.

Typically, these bid prices are derived from the shadow prices of the capacity constraints in a deterministic approximation of the underlying stochastic problem. In this section, we detail the implementation of a bid-price control policy for our model.

We begin by formulating a deterministic linear programming (LP) approximation, specifically the LP relaxation of a multi-type multiple knapsack problem. This model uses expected demand over the horizon. Let x_{ij} denote the number of type i items assigned to knapsack j, and let $d_i = \sum_{t=1}^T \lambda_i^t$ represent the expected number of requests for type i. The formulation is as follows:

$$\max \quad \sum_{i=1}^{M}\sum_{j=1}^{N} r_i x_{ij} \tag{2}$$

$$\text{s.t.} \quad \sum_{j=1}^{N} x_{ij} \le d_i, \quad i \in \mathcal{M}, \tag{3}$$

$$\sum_{i=1}^{M} w_i x_{ij} \le c_j, j \in \mathcal{N}, \tag{4}$$

$$x_{ij} \ge 0, \quad i \in \mathcal{M}, j \in \mathcal{N}.$$

The objective (2) is to maximize total expected revenue. Constraint (3) ensures the total number of accepted type i items does not exceed its expected demand. Constraint (4) ensures the total weight in each knapsack j does not exceed its initial capacity C_j.

The monotonic increase of the profit-to-weight ratio r_i/w_i with type index i implies that items with a higher index are more profitable per unit of capacity. Consequently, the optimal solution to the LP relaxation exhibits a greedy structure, preferentially utilizing higher-indexed item types. This structural property is formalized in Proposition 1.

Lemma 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.

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

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

Algorithm 1: Bid-Price Control

```
1  for $t = 1, \dots, T$ do
2      Observe a request of item type $i$;
3      Solve problem (5) with $\boldsymbol{d}^{[t,T]}$ and $\mathbf{C}^t$;
4      Obtain $\tilde{i}$ such that the aggregate optimal solution is $xe_{\tilde{i}} + \sum_{i=\tilde{i}+1}^{M} d_i^t e_i$;
5      if $i \geq \tilde{i}$ and $\max_{j \in \mathcal{N}} c_j^t \geq w_i$ then
6          Set $k = \arg\min_{j \in \mathcal{N}} \{c_j^t | c_j^t \geq w_i\}$ and break ties;
7          Assign the item to knapsack $k$, let $c_k^{t+1} \leftarrow c_k^t - w_i$ ;
8      else
9          Reject the request;
10     end
11 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 capacities permit, the policy treats all knapsacks as equally preferable, making no distinction among them.

Example 1. *Consider an instance with $M = 3$, $N = 4$, $w_1 = 3, w_2 = 4, w_3 = 5$, $r_1 = 4, r_2 = 6, r_3 = 8$, $\boldsymbol{C} = [7, 8, 8, 4]$ and $T = 8$. The stationary arrival probability is $\lambda_1 = \lambda_3 = \frac{1}{4}$, $\lambda_2 = \frac{1}{2}$, giving an expected demand vector $\boldsymbol{d} = (2, 4, 2)$.*

Under traditional bid-price control, the bid-price for each knapsack j is $\beta_j^* = \frac{4}{3}$, yielding a total expected reward $z(BPC) = \frac{124}{3}$. This solution exhibits two major drawbacks:

It fails to differentiate between knapsacks, assigning them a uniform bid-price. It permits potentially infeasible assignments. For instance, even though $r_3 - \beta_4^* * w_3 > 0$, assigning a type 3 item (with weight 5) to knapsack 4 (with capacity 4) is infeasible.

3.2 BPC Policy Based on Patterns (BPP)

To better account for the differences between knapsacks when placing items, we propose an enhanced dynamic programming (DP) formulation. The key idea is to represent the state of each knapsack using feasible assignment patterns, which encode the combinations of items it can hold, rather than merely tracking its residual capacity. This approach more fully captures the knapsack's combinatorial structure.

Formally, a pattern for a knapsack is a vector $\boldsymbol{h} = [h_1, \dots, h_M]$ where each h_i denotes the number of type-i items assigned. A pattern $\boldsymbol{h}$ is feasible for a knapsack with capacity c_j if it satisfies $\sum_{i=1}^{M} w_i h_i \leq c_j$. We denote the set of all feasible patterns for knapsack j by $S(c_j)$.

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:

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

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 $\boldsymbol{h}$ 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:

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

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: $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta^\dagger_{ij} h_i$ and $h_i^* \geq 1$, we establish the value difference:

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

The acceptance threshold is then defined as:

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

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

Let $\gamma_j = \max_{\boldsymbol{h}\in S(c_j)}\{\sum_i \beta^\dagger_{ij} h_i\}$. 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 $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta^\dagger_{ij} h_i, h_i^* \geq 1$, the corresponding bid-price problem can be expressed as:

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

α_i represents marginal revenue for type i. $\beta^\dagger_{ij}$ 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 $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta^\dagger_{ij} h_i$ 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* $\mathcal{J}_i = \{j \in \mathcal{N} \mid r_i - \beta^{\dagger *}_{ij} \geq r_i - \beta^{\dagger *}_{ik}, \forall k \in \mathcal{N}, r_i - \beta^{\dagger *}_{ij} > 0\}$. *If* $\mathcal{J}_i = \varnothing$, *the first set of* *constraints for i is redundant. If $\mathcal{J}_i \neq \varnothing$, then there exists a $j' \in \mathcal{J}_i$ such that:*

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

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

The control policy is then defined as follows. For the arriving type-i item, if $\alpha_i > 0$, accept the type i. Assign the type i to a knapsack $j \in \mathcal{J}_i$ satisfying:

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

If $\alpha_i = 0$ and there exists a knapsack j such that: $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h} \in S(c_j)} \sum_i \beta_{ij}^{\dagger *} h_i, h_i^* \geq 1$, accept the type i and assign it to knapsack j; otherwise, reject the type i.

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

Proposition 2. *We have $z(BPC) \geq z(BPP)$.*

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

```
1  for $t = 1, \dots, T$ do
2      Observe a request of item type $i$;
3      Solve problem (10) with $\boldsymbol{d}^{[t,T]}$ and $\mathbf{L}^t$;
4      if $\alpha_i > 0$ then
5          Assign the type $i$ to a knapsack $j \in \mathcal{J}_i$ satisfying:
               $$\boldsymbol{h}^* \in \arg \max_{\boldsymbol{h} \in S(c_j)} \sum_i \beta_{ij}^{\dagger *} h_i, h_i^* \geq 1.$$
           Let $c_j^{t+1} \leftarrow c_j^t - w_i$;
6      else
7          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
8              Assign the item in knapsack $j$, let $c_j^{t+1} \leftarrow c_j^t - w_i$;
9          else
10             Reject the request;
11         end
12     end
13 end
```

Unlike the traditional BPC, the BPP does not require explicit feasibility checks on capacity. However, 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, the optimal bid price is $\boldsymbol{\beta}_1^* = [4,4,4,6]$, $\boldsymbol{\beta}_2^* = [6,6,6,6]$, $\boldsymbol{\beta}_3^* = [10,8,8,8]$. *We can verify that* $z(BPP) = 40 < z(BPC) = \frac{124}{3}$. *The reduced reward for each item type 3 is:* $r_3 - \boldsymbol{\beta}_3^* = [-2,0,0,0]$, *which indicates type 3 is excluded from knapsack 1. Also, the optimal patterns for knapsack 4,* $[0,1,0]$ *and* $[1,0,0]$. *Thus, 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 the dynamic primal formulation.

4 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:

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

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.

Proposition 3. *If the optimal solution* x_{ij}^* *to* (11) *satisfies* $x_{ij}^* > 0$ *for a pair* (i,j)*, then for that knapsack* j*, we have*

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

In contrast to Lemma 2, this lemma ensures the existance of a knapsack j to accommodate item type i when $x_{ij}^* > 0$. This eliminates the need for explicit existance verification.

Lemma 3. $z(DLP) \ge V^{HO}$ *results from the concave property.*

4.1 Solve the Dynamic Primal

The pattern $\boldsymbol{h}$ is efficient for knapsack j if and only if, for some $(\alpha_1, \dots, \alpha_M, \gamma_j)$ (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:

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

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). Therefore, 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 4. *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 $\sum_{i=1}^{M} d_i w_i \ge \sum_{j=1}^{N} c_j$, 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 dynamic primal, the optiaml solution is given by* $\boldsymbol{x}_1^* = [1, 1, 0, 0]$*,* $\boldsymbol{x}_2^* = [1, 0, 2, 1]$*,* $\boldsymbol{x}_3^* = [0, 1, 0, 0]$*. This indicates that type 1 can be assigned to knapsacks 1 or 2, type 2 cannot be assigned to knapsack 2, type 3 can only be assigned to knapsack 2.*

In the optimal solution, the efficient patterns for each knapsack are $[1, 1, 0]$*,* $[1, 0, 1]$*,* $[0, 2, 0]$*,* $[0, 1, 0]$*. For instance,* $[1, 1, 0]$ *shows that knapsack 1 can hold one item of type 1 and one of type 2.*

Using the decision variable x_{ij} *to represent these assignments is a straightforward and easily implementable approach.*

4.2 Type Analyses

When there is only one type, the optimal policy is straightforward, i.e., accept the request until the capacity is insufficient.

When there are two types,

When will the threshold policy be the optimal?

5 Computational Experiments

$N = 10$, $L_j = 20 \forall j$,

$D_1 = [0.2, 0.5, 0.3]$

$w = [3, 4, 5]$, $r = [4, 6, 8]$, $r_i / w_i > 1 \forall i$

$w = [3, 4, 5] = r$, $r_i / w_i = 1 \forall i$

$w = [8, 6, 4]$ $r = [5, 4, 3]$, $r_i/w_i < 1 \forall i$

$D_2 = [0.25, 0.25, 0.25, 0.25]$

$w = [3, 5, 7, 9]$, $r = [2, 4, 6, 8]$.

Table 1: Performance of Policies

Distribution	T	r_i/w_i	Primal (%)	BPC (%)	BPP (%)
	60	>1	99.15	95.30	97.98
	80	>1	99.37	95.70	98.18
D_1	60	=1	99.68	95.66	98.88
	80	=1	99.80	96.26	99.44
	60	<1	98.56	95.86	97.66
	80	<1	97.00	95.67	97.31
	60	<1	99.65	94.63	99.84
	80	<1	99.81	96.89	99.91
D_2	100	<1	99.94	99.41	99.97

6 Conclusion

We study the online multi-type multiple knapsack problem. We consider the bid-price control policy. Last but not least, to address the seat assignment with dynamic requests, we introduce the dynamic primal 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 different policies. Our proposed policy demonstrates superior and more consistent performance relative to these benchmarks.

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.

Dantzig, G.B., 1957. Discrete-variable extremum problems. Operations Research 5, 266–288.

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.

7 Proofs

Proof of Proposition 1

We model the problem as a special case of the multiple knapsack problem, then we consider the LP relaxation of this problem. In the model, groups are categorized into M distinct types. Each type i is characterized by a fixed size w_i, which serves as the weight, and an associated profit equal to r_i. For every type i, there are d_i items. Altogether, the total number of groups is given by $K = \sum_{i=1}^{M} d_i$. Each individual item k inherits its profit and weight from its type; specifically, if item k belongs to type i, then its profit p_k is r_i, and its weight W_k is w_i. To apply the greedy approach for the LP relaxation of (3), sort these items in non-increasing order of their profit-to-weight ratios: $\frac{p_1}{W_1} \geq \frac{p_2}{W_2} \geq \ldots \geq \frac{p_K}{W_K}$. The break item b is the smallest index such that the cumulative weight of item 1 to item b meets or exceeds the total capacity $\tilde{c}$: $b = \min\{j : \sum_{k=1}^{j} W_k \geq \tilde{c}\}$, where $\tilde{c} = \sum_{j=1}^{N} c_j$ is the total size of all knapsacks. For the LP relaxation of (3), the Dantzig upper bound (Dantzig, 1957) is given by $u_{\text{MKP}} = \sum_{j=1}^{b-1} p_j + \left(\tilde{c} - \sum_{j=1}^{b-1} W_j\right) \frac{p_b}{W_b}$. The corresponding optimal solution is to accept the whole groups from 1 to $b-1$ and fractional $(\tilde{c} - \sum_{j=1}^{b-1} W_j)$ item b. Suppose the item b belong to type $\tilde{i}$, then for $i < \tilde{i}$, $x_{ij}^* = 0$; for $i > \tilde{i}$, $x_{ij}^* = d_i$; for $i = \tilde{i}$, $\sum_{j=1}^{N} x_{ij}^* = (\tilde{c} - \sum_{i=\tilde{i}+1}^{M} d_i w_i)/w_{\tilde{i}}$. ■

Proof of Lemma 1

According to the Proposition 1, the aggregate optimal solution to LP relaxation of problem (3) takes the form $xe_{\tilde{i}} + \sum_{i=\tilde{i}+1}^{M} d_i e_i$, then according to the complementary slackness property, we know that $z_1 = \ldots = z_{\tilde{i}} = 0$. This implies that $\beta_j \geq \frac{r_i}{w_i}$ for $i = 1, \ldots, \tilde{i}$. Since $\frac{r_i}{w_i}$ increases with i, we have $\beta_j \geq \frac{r_{\tilde{i}}}{w_{\tilde{i}}}$. Consequently, we obtain $z_i \geq r_i - w_i \frac{r_{\tilde{i}}}{w_{\tilde{i}}} = \frac{r_i w_{\tilde{i}} - r_{\tilde{i}} w_i}{w_{\tilde{i}}}$.

Given that $\mathbf{d}$ and $\mathbf{L}$ are both no less than zero, the minimum value will be attained when $\beta_j = \frac{r_{\tilde{i}}}{w_{\tilde{i}}}$ for all j, and $z_i = \frac{r_i w_{\tilde{i}} - r_{\tilde{i}} w_i}{w_{\tilde{i}}}$ for $i = \tilde{i}+1, \ldots, M$. ■

Proof of Lemma 2

We consider two cases based on whether the set $\mathcal{J}_i$ is empty or not.

Case 1: $\mathcal{J}_i = \varnothing$ If $\mathcal{J}_i = \varnothing$, then for all $j \in \mathcal{N}$, we have $r_i - \beta_{ij}^{\dagger *} \leq 0$. This implies that the constraint: $\alpha_i \geq r_i - \beta_{ij}^{\dagger *}$ is automatically satisfied for all $j \in \mathcal{N}$ when $\alpha_i \geq 0$. Therefore, these constraints are redundant and can be removed without affecting the solution.

Case 2: $\mathcal{J}_i \neq \varnothing$ We prove by contradiction that there exists $h_i^* \geq 1$ for some $j' \in \mathcal{J}_i$.

Assumption for contradiction: Suppose that in the optimal solution, for all $j' \in \mathcal{J}_i$, we have $h_i^* = 0$.

Since $\mathcal{J}_i \neq \varnothing$, there exists at least one $j' \in \mathcal{J}_i$ such that $r_i > \beta_{ij'}^{\dagger *}$. From the constraint $\alpha_i \geq r_i - \beta_{ij'}^{\dagger *}$ and the optimality conditions, we must have $\alpha_i = r_i - \beta_{ij'}^{\dagger *} > 0$. Now consider the value:

$$\gamma_{j'} = \max_{\boldsymbol{h} \in S(c_{j'})} \sum_i \beta_{ij'}^{\dagger *} h_i$$

Under the contradiction assumption ($h_i^* = 0$ for all $j' \in \mathcal{J}_i$), we have: $\gamma_{j'} = \sum_i \beta_{ij'}^{\dagger *} h_i^*$.

Now examine the objective function:

$$\sum_i \alpha_i d_i + \sum_j \gamma_j.$$

Consider perturbing $\beta^{\dagger*}_{ij'}$ by increasing it slightly to $\beta^{\dagger*}_{ij'} + \delta$ (for some small $\delta > 0$ such that $r_i > \beta^{\dagger*}_{ij'} + \delta$ still holds). Then:

- Since α_i is exactly at the boundary ($\alpha_i = r_i - \beta^{\dagger*}_{ij'}$), we can now set $\alpha_i^{\text{new}} = r_i - (\beta^{\dagger*}_{ij'} + \delta) < \alpha_i$.
- The term $\gamma_{j'}$ remains unchanged because $h_i^* = 0$ for $j' \in \mathcal{J}_i$ (by our contradiction assumption), and the perturbation does not affect other terms.
- Since $d_i > 0$ (positive arrival rate for type i), the objective decreases by $\delta \cdot d_i > 0$.

This contradicts the optimality of the current solution. Therefore, our initial assumption must be false, and there must exist some $j' \in \mathcal{J} i$ such that $hi^* \geq 1$. ■

Proof of Proposition 2

We first testify that the relation satisfy the constraints in BPP. Then the objective value of BPP is no larger than that of BPC.

■

Proof of Proposition 3

If $x^*_{ij} > 0$, then $\sum_{\boldsymbol{h}\in S(c_j)} h_i y_{j\boldsymbol{h}} > 0$, then there exists $\boldsymbol{h}$ such that $y_{j\boldsymbol{h}} > 0$ and $h_i \geq 1$. Then $\boldsymbol{h} \in \arg\max(\sum_i (r_i - \alpha_i)h_i - \gamma_j)$.

According to the complementary slackness property, $\alpha_i + \beta_{ij} = r_i$, then $\max \sum_i (r_i - \alpha_i)h_i - \gamma_j$ equals $\max \sum_i \beta_{ij} h_i$. Thus, there exists j such that $\boldsymbol{h}^* \in \arg\max_{\boldsymbol{h}\in S(c_j)} \sum_i \beta^{\dagger}_{ij} h_i, h^*_i \geq 1$.

■

Proof of Lemma 3

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). $V^{\text{HO}} = E[\phi(\boldsymbol{d})] \leq \phi(E[\boldsymbol{d}]) = z(\text{DLP})$. ■