

Article

Priority-Based Capacity Allocation for Hierarchical Distributors with Limited Production Capacity

Jun Tong ¹, Xiaotao Zhou ¹ and Lei Lei ^{2,*}

¹ School of Management, Zhejiang University of Technology, Hangzhou 310023, China; tongjun20@zjut.edu.cn (J.T.); tao928256@126.com (X.Z.)

² School of Economics and Business Administration, Chongqing University, Chongqing 400044, China

* Correspondence: leilei312@cqu.edu.cn

Abstract: This paper studies the issue of capacity allocation in multi-rank distribution channel management, a topic that has been significantly overlooked in the existing literature. Departing from conventional approaches, hierarchical priority rules are introduced as constraints, and an innovative assignment integer programming model focusing on capacity selection is formulated. This model goes beyond merely optimizing profit or cost, aiming instead to enhance the overall business orientation of the firm. We propose a greedy algorithm and a priority-based binary particle swarm optimization (PB-BPSO) algorithm. Our numerical results indicate that both algorithms exhibit strong optimization capabilities and rapid solution speeds, especially in large-scale scenarios. Moreover, the model is validated through empirical tests using real-world data. The results demonstrate that the proposed approaches can provide actionable strategies to operators, in practice.

Keywords: hierarchicalchannel management; capacity allocation; limited production capacity; PSO algorithm

MSC: 37M05; 65Z05



Citation: Tong, J.; Zhou, X.; Lei, L. Priority-Based Capacity Allocation for Hierarchical Distributors with Limited Production Capacity. *Mathematics* **2024**, *12*, 2237. <https://doi.org/10.3390/math12142237>

Academic Editors: Jian Luo, Zheming Gao and Xin Yan

Received: 3 June 2024

Revised: 8 July 2024

Accepted: 9 July 2024

Published: 18 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Hierarchical channel management is a strategic approach used to enhance operational efficiency and sales performance by segmenting channel distributors into distinct groups. This method is widely employed across various industries. For example, as [1] reported, Huawei, a global leader in information and communications technology, categorizes its channel partners into gold, silver, and certified agents based on their capabilities, scale, market coverage, and industry influence. Similarly, as reported by ITWeb [2], Procter & Gamble (P&G), a major consumer goods company, classifies its partners into different tiers based on their sales capabilities, market coverage, and customer relationships. This approach is also common in other sectors, such as the beverage industry and service sectors.

In this management model, higher-ranked distributors receive more favorable terms, such as greater profit margins, enhanced market support, and preferential product supply. For instance, when Huawei launches new products, top-tier distributors receive a larger allocation, while medium- and lower-ranked distributors are allocated products based on historical performance, market potential, and future growth plans. Companies may adjust allocations based on regional sales performance and distributor achievements.

Capacity shortfalls are a common challenge across many industries, including high-end fashion, electronics, and products that unexpectedly surge in popularity [3,4]. In such scenarios, marketing departments must prioritize distributors and optimally allocate capacity to balance supply and demand across regions and minimize negative impacts on channel partners.

Typically, as highlighted by [4], three approaches are widely adopted to address capacity shortfall issues. The first approach involves manufacturers imposing restrictions

on certain terms when distributors procure products, commonly seen in the semiconductor and steel industries. The second approach entails manufacturers setting differentiated wholesale prices for distributors to influence their purchasing decisions, as observed in the toy and clothing industries. The third approach involves manufacturers directly allocating production capacity to distributors, a practice frequently employed in the electronics and semiconductor industries. This paper primarily focuses on the third method, specifically on making decisions regarding capacity allocation among ranked distributors.

Prior studies have explored capacity allocation within supply chains using two primary mechanisms: the proportional mechanism, which allocates capacity based on the size of distributors' orders, and the lexicographic mechanism, which allocates capacity based on a predetermined priority sequence. Practically, the proportional mechanism is significantly easier to implement than the lexicographic one. Although ref. [5,6] found that the proportional mechanism incentivizes retailers to increase their orders beyond their needs, resulting in higher profits for both the supplier and the entire supply chain, ref. [7] demonstrated that the supplier benefits more from the lexicographic mechanism than the proportional mechanism by inducing high-priority retailers to order more than they need. For a comprehensive discussion on this topic, readers are referred to the reviews by [4,5,7,8].

Despite the significant achievements in the literature on production capacity allocation, several shortcomings remain. Firstly, and most importantly, the existing literature has largely overlooked the specific structure of hierarchical channel management modes, which can significantly impact production capacity allocation. While the lexicographic mechanism is analogous to hierarchical channel management when each rank contains only one distributor, it differs substantially when a ranked group includes multiple distributors. Secondly, almost all existing studies typically formulate capacity allocation decisions within a game theory framework, aiming to maximize profits (e.g., [4,7–11]) or minimize costs (e.g., [12]) for the supply chain involved. However, it is noteworthy that, in practice, many firms prioritize the long-term maintenance of product brand value during capacity allocation over merely maximizing profit or minimizing cost.

Considering these factors, this study investigates a make-to-order supply chain [4], wherein a manufacturer produces a singular product type in response to orders originating from multiple channel distributors. Diverging from prior research that employed game theory, this paper primarily focuses on the development of an optimization model for production capacity allocation within the hierarchical structure of distributors. To be practical, we assume that channel distributors at the same rank receive identical capacity allocations while substantial disparities in final orders within the same rank arise due to variances in market demand and operational capabilities among channel distributors. Note that establishing either a high or low capacity for a rank of distributors can engender a discordance between a firm's orientation and a distributor's demands, which may compromise the efficacy of the entire supply chain. To address this concern, our model mainly focuses on precise alignment between the firm's business orientation and customer demands, rather than solely optimizing for profit or cost, as is commonly considered in the literature.

Specifically, an assignment integer programming model with decision variables corresponding to capacity selection is developed, placing emphasis on the actual impact of capacity allocation as the primary objective for decision making. To assess the effectiveness of decisions, a comprehensive index is developed that incorporates key factors, such as purchase fulfillment rate, product coverage, order fulfillment coverage, and distributor ordering rate. This index essentially integrates the company's profit objectives with its business orientation to a significant extent.

It is widely acknowledged that solving integer programming problems presents a great challenge, particularly when addressing large-scale real-world scenarios. Hence, to work out high-quality solutions for marketing decision makers within practically acceptable time frames, we center on designing effective heuristic algorithms, a direction that has received significant attention in the existing literature.

Particle swarm optimization (PSO) is known as one of the most effective heuristic approaches for its fast convergence, easy implementation, and strong global search capabilities [13–15]. This swarm intelligence algorithm mimics the foraging behavior of birds, to find optimal solutions through information sharing and position updates among particles in each iteration. Discrete PSO extends the classical PSO algorithm, to handle discrete and combinatorial optimization problems where a particle's position is represented as a set of discrete values. In recent decades, there have been remarkable research achievements in this area. Notably, studies such as [16,17] have harnessed PSO to enhance performance metrics, such as resource utilization, response time, and load balancing within the realms of cloud computing and internet of things (IoT) platforms. Additionally, ref. [18] devised a hybrid approach that combined the PSO algorithm with other heuristic approaches, to tackle expansive mixed-integer linear programming problems. Ref. [19] introduced a modified PSO algorithm, MICPSO-TS, seamlessly integrated with tabu search to optimize intricate inverse scheduling predicaments. Ref. [15] proposed a probability-based discrete PSO algorithm tailored to addressing challenges in product portfolio planning. To learn more about the latest progress in this field of research, readers can refer to the reviews by [20]. Nevertheless, to the best of our knowledge, the application of PSO in production capacity allocation problems remains relatively unexplored in the literature.

In this paper, a priority-based mechanism is proposed for updating the position of each encoded particle, based on the model's structure. In this mechanism, higher-ranked distributors are prioritized over lower-ranked distributors in capacity allocation. To ensure that only one capacity is selected for each rank from all corresponding candidates, the multinomial logit choice method, which differs from the sigmoid function used in existing studies, is employed to update the position of each particle. Recognizing the effectiveness of the greedy algorithm in constructing initial solutions in the field of optimization, a customer enthusiasm-based greedy algorithm is proposed, to find high-quality solutions, particularly in large-scale scenarios.

Numerical experiments were conducted, to assess the accuracy and speed of the greedy method and the priority-based binary particle swarm optimization (PB-BPSO) algorithm, and to compare them with a commercial solver. The findings indicate that for scenarios with a small number of production capacity candidates, the solver can swiftly identify the optimal solution. However, as the scale of candidates increases, these algorithms exhibit superior performance. Notably, the greedy algorithm generates high-quality approximate solutions in mere milliseconds, whereas the PB-BPSO algorithm provides approximations within seconds. Notably, the PB-BPSO algorithm achieves greater accuracy while maintaining low error levels, even with a large number of production capacity candidates. Consequently, the PB-BPSO algorithm is a more effective approach for large-scale problems, particularly in parallel computing environments, where its advantages become more pronounced.

Moreover, the model was applied to develop production capacity allocation strategies for a product during its adjustment and steady-state phases, using real data from our partner company. The empirical results reveal that during the adjustment phase, production capacity tended to be allocated to higher-rank distributors. Conversely, during the steady-state phase, the strategies focused more on balancing the demands of distributors across different ranks. Notably, these strategies received positive feedback from the company's operators, indicating their alignment with the company's overall business orientation.

The remainder of the paper is structured as follows. Section 2 introduces the fundamental model and provides rigorous definitions of the relevant indicators. Section 3 mathematically presents the formulations of the model. In Section 4, a greedy algorithm is initially designed, and a priority-based BPSO algorithm is proposed. Section 5 analyzes the proposed algorithms and compares them through both numerical and empirical experiments. Finally, Section 6 concludes the study.

2. Problem Description

Consider a make-to-order supply chain operating within a single period, where a manufacturer (supplier) distributes products to customers across various regions through channel distributors. These distributors are grouped into several ranks. Within this ranking sequence, the higher the rank of a channel distributor, the higher their priority in obtaining products and the larger their allocated capacity (or procurement quota). Channel distributors at the same rank share the same capacity.

The process of allocating production capacities can be described as follows:

- *Stage 1: Establishing priority rules.* The manufacturer first establishes clear priority rules for allocating production capacities among different ranks.
- *Stage 2: Preliminary orders.* Distributors then submit their preliminary orders to the manufacturer.
- *Stage 3: Capacity allocation.* The manufacturer determines specific allocation strategies by setting the maximum quantity each rank of channel distributors is allowed to order, based on the priority rules.
- *Stage 4: Revising orders.* Finally, each distributor revises their order and procures products from the manufacturer, ensuring that the quantity does not exceed their allocated production capacities.

In Stage 1, two priority rules, which are widely adopted in the business landscape, are taken into consideration. The first rule is that the capacity allocated for high-rank channel distributors must be greater than that for low-rank distributors. The second rule is that the allocation of capacity must adhere to a sequential approach among adjacent ranks. For instance, if capacity is allocated to rank A and rank C, it is essential that the intervening rank B also receives an allocation. These rules enable the manufacturer to maintain the stability of its distribution strategy in the long term, fostering stronger and more harmonious relationships among its channel partners.

In Stages 2 and 4, disparities in orders can arise significantly among distributors of the same rank, due to variations in their operational capabilities and local market demands. Allocating excessive production capacity to a rank of channel distributors with low demand can lead to under-utilization, resulting in significant inventory stockpiles for the manufacturer. Conversely, allocating insufficient capacity to distributors with high demand may result in a shortage of market supply. Therefore, in Stage 3 the manufacturer must carefully allocate production capacities to different ranks of channel partners, taking into account the constraints of limited product supply.

In Stage 3, the manufacturer's distribution strategy must be aligned with their marketing orientation. During the product incubation phase, for instance, the allocation of production capacity is primarily focused on broadening the market reach of the product, emphasizing the expansion of the network of channel distributors. However, if the average market performance of the product at specific ranks of distributors, such as the inventory-to-sales ratio, falls short of the company's predefined expectations, the marketing focus shifts towards reducing capacity at those ranks. This adaptive marketing strategy is particularly prevalent among monopolistic enterprises.

As is widely acknowledged in the industry of our served partner company, four pivotal indicators serve to assess the post-implementation efficacy of product distribution: coverage ratio, procurement ratio, fulfillment rate, and order rate. Table 1 outlines the specific formulas for calculating these metrics. Specifically, the coverage ratio assesses the extent of market coverage, while the procurement ratio measures the success of achieving distribution strategies. Typically, higher ratios after implementing distribution strategies indicate a greater degree of precision in strategy execution.

Based on our analysis of real-world data, achieving optimal performance across all ratios concurrently poses considerable challenges. Typically, allocating greater production capacity to specific ranks boosts market supply but tends to compromise the fulfillment rate. Conversely, reducing production capacity enhances the fulfillment rate of distributors but

can curtail market supply. Therefore, in practical applications, manufacturers must strike a balance among these ratios, taking into account their marketing orientation, especially when production capacity is limited.

Table 1. Explanation of Indicators.

Indicator	Explanation
Coverage Ratio	Proportion of distributors at a specific rank that meet their full capacity to the total number of distributors at that rank.
Procurement Ratio	The number of distributors who have procured products compared to the total number of distributors.
Fulfillment Rate	The total procurement quantity of all distributors at a specific rank as a proportion of the total available production capacity for that rank.
Order Rate	The percentage of distributors at a specific rank who need to procure products, relative to the total number of distributors at that rank.

3. The Model

The problem described in the previous section can be formulated as an assignment model. Specifically, all feasible production capacities for each rank i are first enumerated as candidates. Then, the decision to set a capacity for that rank is essentially equivalent to assigning the best candidate from all options, such that the business orientation of the manufacturer is most effectively achieved. Here, to make the model more practical, we use the weighted ratios defined in Table 1 as the indicators for business orientation.

Table 2 collects the notations used in the model along with their explanations.

In this table, for any given rank i , the indicators of ξ_{ij} , ψ_{ij} , ζ_{ij} , and χ_{ij} are random variables, which can only be observed after allocating capacity j . Typically, higher values of these indicators suggest that more customers are likely to procure more products. Thus, these indicators effectively depict customer (channel distributor) enthusiasm at rank i when allocated capacity j . This enthusiasm is a key factor in understanding consumer behavior and predicting market trends. So, it is meaningful to set the expected weighted sum of these four key indicators as the objective function.

The capacity allocation decision model can be expressed as follows:

$$\max \sum_{i=1}^m \sum_{j=0}^n (\lambda_1 E[\xi_{ij}] + \lambda_2 E[\psi_{ij}] + \lambda_3 E[\zeta_{ij}] + \lambda_4 E[\chi_{ij}]) x_{ij} + M \sum_{i=1}^m \sum_{j=0}^n j x_{ij} \eta_i \quad (1)$$

$$\text{s.t.} \quad \sum_{i=1}^m \sum_{j=0}^n j x_{ij} \eta_i \leq C \quad (2)$$

$$\sum_{j=0}^n x_{ij} = 1 \quad \forall i = 1, \dots, m \quad (3)$$

$$\sum_{j=0}^n j x_{i-1j} \leq \sum_{j=0}^n j x_{ij} \quad \forall i = 2, \dots, m \quad (4)$$

$$x_{ij} \in \{0, 1\} \quad \forall i = 1, \dots, m, \quad \forall j = 0, \dots, n. \quad (5)$$

In this formulation, the objective function (1) maximizes the expected weighted sum of the key indicators for the allocated capacities while also maximizing the total quantity of allocated capacities across all ranks. Here, the weights λ_1 , λ_2 , λ_3 , and λ_4 reflect the relative importance of the four ratios, and M serves as a reward for allocating one unit more capacity to customers. It is undeniable that the practical application of setting these parameters poses significant challenges. Nonetheless, there exist strategies that seamlessly blend qualitative and quantitative indicators to tackle this issue. One such approach is the expert scoring method, where marketers leverage their extensive experience to fine-tune the parameters. Additionally, the entropy method and the fuzzy comprehensive evaluation method offer alternative frameworks that integrate both qualitative and quantitative aspects. Equation (2)

guarantees that the total allocated capacity does not exceed the total production capacity C . Constraint (3) ensures that each rank i receives one and only one strategy of capacity allocation, including the strategy without allocating capacity for $j = 0$. Equation (4) ensures that products are prioritized to satisfy higher-rank distributors and that cross-rank distribution of products is prohibited. Equation (5) signifies that all decision variables take values of either 0 or 1, identifying whether each rank i is assigned capacity j or not.

Table 2. Notations.

Notation	Explanation
m	Number of ranks
n	Maximum quantities of capacity allocated to each rank
i	The i -th rank, $i = 1, \dots, m$
j	Quantities of the allocated capacity, $j = 0, \dots, n$
ξ_{ij}	Fulfillment rate at rank i with allocated capacity j
ψ_{ij}	Coverage ratio at rank i with allocated capacity j
ζ_{ij}	Order rate at rank i with allocated capacity j
χ_{ij}	Procurement ratio at rank i with allocated capacity j
η_i	Number of distributors at rank i
C	Total quantity of production capacity
λ_u	Weight of indicator u , $u = 1, 2, 3, 4$
x_{ij}	0–1 decision variables indicating the assignment of capacity j to rank i

As evident, the scalability of the model relies on the values of m and n . In fact, in the real business world, companies often collaborate with a large number of channel distributors. This necessitates the delineation of multiple ranks to facilitate refined management strategies. For those companies in personalized management approaches, each channel distributor can be considered as a rank and, thus, m is exactly equal to the total count of the channel distributors. Furthermore, the magnitude of n is typically substantial in practice and is dictated by the chosen unit of measurement for production capacity. For instance, our partner company, specializing in fast-moving consumer goods, allocates capacity weekly, distributing around 10,000 boxes of products to distributors in a city, on average.

Large values of m and n pose significant challenges in solving the model. To circumvent this and render the model applicable for business operators, the subsequent section endeavors to employ heuristic methodologies in devising solution algorithms that are operable and capable of providing high-quality approximate solutions within an acceptable time frame.

4. Solution Method

Mathematically, it has been widely documented in the literature that the 0–1 integer programming problem is NP-hard. Consequently, no algorithm can find an exact optimal solution in polynomial time. In this context, imprecise approaches, such as greedy algorithms and heuristic algorithms, are often regarded as more attractive approaches both in the academic and the business communities. In this section, two approximate solution methods are devised: a greedy algorithm and a priority-based discrete particle swarm optimization heuristic algorithm.

4.1. Greedy Algorithm

This algorithm employs the greedy idea to design a mechanism for seeking a high-quality solution. For ease of expression, let $f_{ij} \doteq \lambda_1 E[\xi_{ij}] + \lambda_2 E[\psi_{ij}] + \lambda_3 E[\zeta_{ij}] + \lambda_4 E[\chi_{ij}]$. As mentioned in the previous section, a higher value of f_{ij} indicates greater customer enthusiasm at rank i when allocated capacity j . Given this observation, we first sort f_{ij} in descending order for each rank i to obtain $f_{ij_1} \geq \dots \geq f_{ij_n}$. Based on the generated sequence, the specific process of the greedy algorithm is outlined in Algorithm 1.

In Step 2, the specific criteria for stopping the algorithm are as follows: (i) If the remaining capacity is insufficient to allocate at any rank; (ii) If the remaining capacity is

greater than the number of distributors at some ranks but allocating to those ranks would violate the priority rules.

Algorithm 1 Greedy Algorithm.

Step 1: Initialization of parameters. Set the optimal capacity for rank i as $q_i = 0$ for all i ; set $q_{m+1} = \infty$ and the remaining capacity as C . Let $l = m$.

Step 2: If the remaining capacity cannot be further allocated then the algorithm stops. Otherwise, if $q_{(l+1)} > q_l$ then set $q_l = q_l + 1$. Calculate the remaining capacity and proceed to Step 3.

Step 3: Let $l' = l - 1$. If $l' = 0$ then set $l = m$. Otherwise, if $f_{(l'-1)(q_{l'}+1)} > f_{lq_{l+1}}$ then let $l = l'$. If $f_{(l'-1)(q_{l'}+1)} \leq f_{lq_{l+1}}$ and $f_{lq_l} \geq f_{(l+1)q_l}$ then set $l = m$. Go back to Step 2.

In Step 3, an assessment is conducted to determine whether it is more advantageous to allocate one additional unit of the remaining capacity to a lower rank. This evaluation is based on a comparison of customer enthusiasm at the current rank with that at the subsequent lower rank, assuming an incremental unit of capacity is added to the latter. Then, the rank exhibiting higher customer enthusiasm is selected for the additional capacity allocation. If the remaining capacity is still sufficient to fully cover at least one rank but the allocation to that rank violates established priority rules then the allocation process for the remaining capacity is restarted from the highest rank. This procedure is repeated until one of the stop criteria outlined in Step 2 is met.

4.2. Heuristic Algorithm

Given the effectiveness and simplicity of the particle swarm optimization (PSO) algorithm in various applications, this part aims to explore this type of algorithm in detail. To enhance comprehension, we first provide a concise overview of the discrete PSO algorithm.

4.2.1. Overview of Discrete Particle Swarm Optimization

The fundamental concept of classical particle swarm optimization (PSO) was inspired by the foraging behavior of birds, with each particle representing a candidate solution. It has two attributes: position and velocity, which, respectively, denote each particle's location and movement direction in the search space. During each iteration, the particles update their positions and velocities based on their own best positions and velocities as well as the best positions and velocities of the swarm. Through the sharing of positional and velocity information within the swarm, the particles collectively move towards an optimal solution.

Velocity Update. The flight velocity of a particle consists of three components: inertia, cognitive, and social. The inertia component represents the tendency for particles to maintain their current velocities and is composed of an inertia weight and the particle's own velocity. The cognitive component represents the adjustment of velocity based on the particle's own experience and is composed of an individual learning factor, a random number, and the difference between the particle's historical best position and its current position. The social component represents the adjustment of velocity based on swarm information and is composed of a swarm learning factor, a random number, and the difference between the swarm's historical best position and the particle's current position. Mathematically, the velocity of a particle is updated as follows:

$$v_{id}^{k+1} = \omega v_{id}^k + c_1 r_1 (p_{id,pbest}^k - x_{id}^k) + c_2 r_2 (p_{d,gbest}^k - x_{id}^k), \quad (6)$$

where v_{id}^k is the velocity of particle i in dimension d at iteration k , ω is the inertia weight, c_1 and c_2 are cognitive and social learning factors, respectively, r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$, $p_{id,pbest}^k$ represents the best position of particle i in dimension d at iteration k , $p_{d,gbest}^k$ represents the best-known position of the entire swarm in dimension d at iteration k , and x_{id}^k is the current position of particle i in dimension d at iteration k .

Position update. The position of a particle is updated according to the following equation:

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1},$$

where x_{id}^{k+1} is the updated position of particle i in dimension d at iteration $k + 1$, x_{id}^k is the current position of particle i in dimension d at iteration k , and v_{id}^{k+1} represents the updated velocity of particle i in dimension d at iteration $k + 1$.

To adapt PSO for combinatorial optimization problems, the binary particle swarm optimization (BPSO) algorithm was introduced. Unlike classical PSO, where particles move in a continuous search space, BPSO operates in a binary search space where each particle's position is represented by binary values (0 or 1).

In BPSO, the velocity update equation remains similar to the classical PSO but requires a transformation to map velocities to probabilities that determine the binary state. However, the position update formula requires random selection based on the probability values of the velocity vector, to ensure that the position vector of each particle can only take a value of 0 or 1.

The formula of probabilistic function, utilizing the sigmoid function, is given by

$$s(v_{id}^k) = \frac{1}{1 + e^{-v_{id}^k}}.$$

The position update for each particle is then defined by

$$x_{id}^k = \begin{cases} 1, & \text{if } s(v_{id}^k) > r_3; \\ 0, & \text{otherwise} \end{cases}$$

where $r_3 \in (0, 1)$ is a random number.

4.2.2. Priority-Based Binary Particle Swarm Optimization

To effectively tailor the BPSO algorithm to the model, which encompasses rank priority constraints, we investigate a priority-based binary particle swarm optimization (PB-BPSO) algorithm. This novel approach introduces a position and velocity update mechanism that adheres to rank priority rules within the classical BPSO framework, aiming to elevate the overall optimization efficiency of the particle swarm within the constrained space.

The basic framework of the algorithm is illustrated in Algorithm 2:

Algorithm 2 Framework of the Priority-Based Binary Particle Swarm Optimization (PB-BPSO) Algorithm.

Step 1: Initialize parameters, including the maximum number of iterations $T_{\max}$, the size of the particle swarm N , the range of particle positions and velocities, and the coefficients ω , c_1 , and c_2 .

Step 2: Compute the fitness value for each particle. If a particle's current fitness value exceeds its historical best, update its individual best position. Based on the updated individual best positions, update the global best position for the swarm.

Step 3: Generate the new velocity for each particle and update the position for each particle, using the priority-based update mechanism. Proceed to Step 4.

Step 4: If the current iteration count exceeds $T_{\max}$ then terminate the algorithm; otherwise, return to Step 2.

In this framework, the adjustments of particle velocity and particle position stand as a pivotal element of the algorithm. In the subsequent sections, we will delve into the detailed design and implementation of this process, where the objective function of the model is designated as the fitness function.

Particle Encoding. Given that the model aims to select the optimal production capacities for m ranks, each with n capacity candidates, it is appropriate to encode the position and velocity of each particle i using $m \times n$ matrices. These matrices are defined as follows:

$$X_i = \begin{bmatrix} x_{i11} & \dots & x_{i1n} \\ \vdots & \ddots & \vdots \\ x_{im1} & \dots & x_{imn} \end{bmatrix}_{m \times n}, \quad V_i = \begin{bmatrix} v_{i11} & \dots & v_{i1n} \\ \vdots & \ddots & \vdots \\ v_{im1} & \dots & v_{imn} \end{bmatrix}_{m \times n}$$

Here, X_i represents the position matrix of the i -th particle, where each element x_{ijd} is a binary value indicating whether the i -th particle is assigned capacity d at rank j . The matrix V_i represents the velocity of the i -th particle, with each element v_{ijd} corresponding to the velocity component of particle i in dimension d at rank j .

Velocity Update. The update process for each velocity component at the $k + 1$ -th iteration is similar to that of classical PSO, and it is defined as follows:

$$v_{ijd}^{k+1} = \omega v_{ijd}^k + c_1 r_1 (p_{id,pbest}^k - x_{ijd}^k) + c_2 r_2 (p_{id,gbest}^k - x_{ijd}^k).$$

Position Update. To ensure that particles consistently search for optimal solutions within the feasible domain, it is crucial to design an iterative mechanism that updates particle positions in accordance with the model's constraints. Given that Constraint (3) requires that only one capacity can be selected from the available candidates for each rank j , we utilize the multinomial logit model to determine the binary elements in each row of the position matrix X_i for each particle i . This method differs from traditional BPSO, which utilizes the sigmoid function.

The binary elements in each row of the position matrix X_i are determined by Constraint (4), which requires that the production capacity allocated to higher-rank distributors must not be less than that allocated to lower-rank distributors. So, to ensure that the updated particles comply with priority rules, it is necessary to introduce a priority-based mechanism into the update process.

We denote the probability of selecting capacity d at rank j as $s(v_{ijd}^k)$ and let the production capacity of the rank immediately above rank j (if it exists) be denoted as $d_{(j+1)}$. To meet Constraint (2), the maximum capacity available to rank j , denoted by $\tilde{d}_j$, is calculated as follows:

$$\tilde{d}_j = \min\{d_{(j+1)}, \lfloor \frac{C - \sum_{i=j+1}^m \sum_{d=0}^n d \eta_i x_{ijd}^k}{\eta_j} \rfloor\}$$

Here, the operator $\lfloor \cdot \rfloor$ represents rounding down to the nearest integer. Then, the probabilistic function at rank j can be formulated as follows:

$$s(v_{ijd}^k) = \begin{cases} \frac{1}{1 + \sum_{d=2}^n e^{-v_{ijd}^k}}, & \text{if } d = 0; \\ \frac{e^{-v_{ijd}^k}}{1 + \sum_{d=2}^n e^{-v_{ijd}^k}}, & \text{if } d = 1, \dots, \tilde{d}_j. \end{cases}$$

Based on this probability distribution, it is suitable to employ the roulette wheel selection method to determine capacity. Let F_{ijd} denote the cumulative probability corresponding to rank j with allocated capacity d , formulated as below:

$$F_{ijd} = F_{ij(d-1)} + s(v_{ijd}^k), \quad d = 1, \dots, \tilde{d}_j \quad (7)$$

where $F_{ij0} = 0$.

The position update for particle i in dimension d , $d = 0, \dots, n$, at rank j is then determined by

$$x_{ijd}^k = \begin{cases} 1, & \text{if } F_{ijd} \leq r_{ij} < F_{ij(d+1)}; \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where r_{ij} is a random number in the interval $[0, 1]$ and $F_{ij(\tilde{d}_j+1)} = \infty$.

The procedure of the priority-based mechanism for updating the position of particle i at the k -th iteration is detailed in Algorithm 3.

This procedure ensures that the particle's position is updated systematically while adhering to the given constraints and priority rules. Moreover, the combination of cumulative probability calculation and random selection effectively balances exploration and exploitation within the search space.

Algorithm 3 Priority-based position update for the i -th particle.

Step 1: Initialize v_{ijd}^k ($j = 1, \dots, m; d = 1, \dots, n$) and let X_i be a $m \times n$ zero matrix. Set $j = m$, $t = 0$, and $\tilde{d} = n$;

Step 2: Compute F_{ijd} using Equation (7) for $d = 0, \dots, \tilde{d}$; generate r_{ij} in the interval $[0, 1]$. Update x_{ijd}^k using Equation (8). Let $\tilde{d} = d_j$, $j = j - 1$, and proceed to Step 3.

Step 3: If $j = -1$ or $\tilde{d} = 0$ then terminate the iteration; otherwise, return to Step 2.

5. Computational Experiments

This section evaluates the performance of the model and algorithms using actual product sales data provided by our project partner, a leading Chinese fast-moving consumer goods company. The company organizes its channel distributors into 30 different levels across various cities, with each level containing a varying number of distributors. To address product supply constraints, the company employs customized strategies to establish distinct production capacities for distributors at different levels.

The dataset covers data on capacity quotas, the number of channel partners, preliminary procurement quantities, final procurement quantities, and the districts of the channel partners for each level, spanning from 1 January 2021 to 31 December 2022. All computational experiments were conducted using MATLAB software 2021b and executed on a personal computer equipped with a CPU running at 2.30 GHz and 4 GB of RAM.

5.1. Numerical Analysis

To enhance the algorithm's convergence rate, an adaptive weight policy was employed at each iteration k , calculated using the formula $\omega^k = \omega_{\max} - k \frac{\omega_{\max} - \omega_{\min}}{T_{\max}}$, where $\omega_{\min} = 0.4$, $\omega_{\max} = 0.8$, and the maximum number of iterations $T_{\max} = 200$ were suitable for our experiments through pre-tested trails. The algorithm's parameters were configured as follows: we let the particles' velocities range from $v_{\min} = -5$ to $v_{\max} = 5$ and we set the cognitive and social learning factors both to be $c_1 = c_2 = 2$. The basic parameters of the model were defined as follows: the number of ranks was $m = 30$; the four indicators were randomly generated from the interval $[0, 1]$ with the coefficients specified as $\lambda_1 = 1$, $\lambda_2 = 1$, $\lambda_3 = 0.001$, and $\lambda_4 = 0$, respectively; we set $M = 2$.

Convergence. First, we examined the solution accuracy and speed performance of the two proposed algorithms under scenarios with different n and total capacity C . For ease of comparison, the built-in integer programming solver in MATLAB was employed to obtain the theoretically precise solutions. Given the randomness of the PB-BPSO algorithm, its performance was evaluated from a statistical perspective over 30 replications.

Table 3 presents the numerical results obtained by the solver, the greedy algorithm, and the BP-PBSO algorithm, respectively, for different scales of experiments.

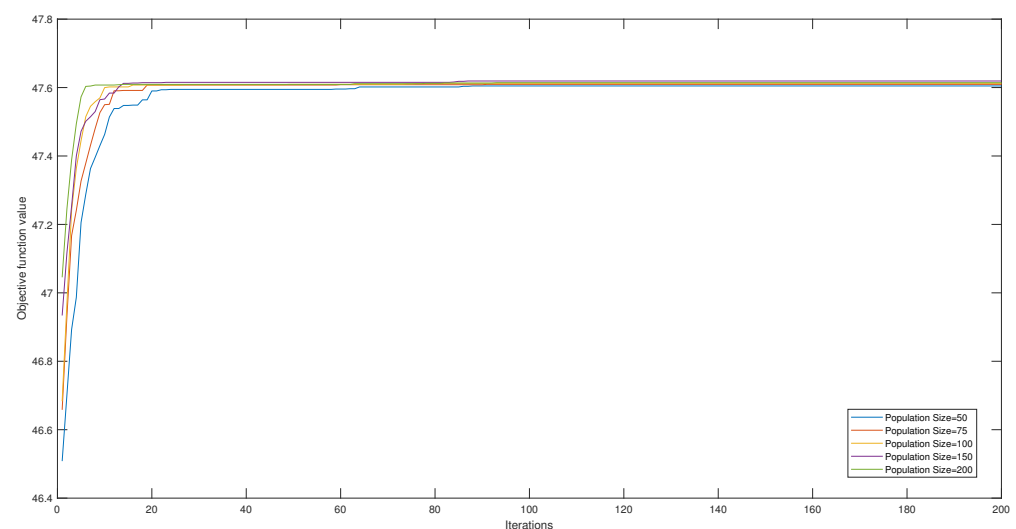
Table 3. Performances of Greedy and PB-BPSO algorithms under different n .

n	C	Solver		Greedy			PB-BPSO		
		Opti.	Time (s)	Appr.	Devi. (%)	Time (s)	Mean	RMSE (%)	Time (s)
3	8000	47.5926	0.0399	45.9106	3.5341	0.0052	47.5926	0	2.2344
5	8000	47.6221	0.6892	45.9106	3.5939	0.0046	47.6162	0.0132	2.2330
10	20,000	48.3768	1.0052	45.9106	5.0978	0.0028	48.2075	0.1924	3.9817
50	80,000	51.9821	5.3182	46.8466	9.8792	0.0052	50.9839	1.0132	6.5242
100	200,000	54.6425	64.8616	46.8466	14.2670	0.0035	51.3357	3.3927	8.9803
200	400,000	-	3600	46.8466	-	0.0044	51.8096	-	16.1217

Specifically, Columns 3 and 4 correspond to the optimal objective function value and computation time obtained by the solver. Columns 5 to 7 represent the approximate objective function value, percentage deviation, and computation time achieved by the greedy algorithm. Columns 8 to 10 correspond to the average objective function value, the root mean square error (RMSE), and the average computation time of the PB-BPSO algorithm, derived from 30 replications.

Three findings can be observed from the table: (1) While the solver demonstrated robust performance, in terms of speed and accuracy, for smaller problem sizes (where $n \leq 100$), its effectiveness diminished exponentially as the scale gradually expanded. Remarkably, for a problem size of $n = 200$ with $C=400,000$, corresponding to a model encompassing 6000 decision variables, the built-in solver failed to yield an optimal solution after an hour of iterations. (2) In contrast to the solver, both the PB-BPSO algorithm and the greedy algorithm demonstrated swift solution speeds, albeit with relatively lower accuracy. Notably, their performance did not deteriorate significantly as n increased. Specifically, the average computational time of the PB-BPSO algorithm remained under 20 s, even for $n = 200$. (3) The PB-BPSO algorithm outperformed the greedy algorithm in achieving superior solutions with significantly smaller errors across all scenarios. The error of the greedy algorithm grew rapidly with the increase of n , whereas the PB-BPSO algorithm exhibited a less significant increase in error. Although the PB-BPSO algorithm required a longer computation time compared to the greedy algorithm, it was still capable of providing high-quality solutions in a very short period. Therefore, the PB-BPSO algorithm was significantly more suitable for tackling large-scale problems.

In addition, we explored the influence of varying population sizes on the performance of the PB-BPSO algorithm, considering sizes of 50, 75, 100, 150, and 200 particles, respectively. Figure 1 illustrates the trajectories of the average optimal values achieved by the particle swarms across these different population sizes.

**Figure 1.** Convergence trajectory of PB-BPSO with varying population sizes.

It can be observed that the PB-BPSO algorithm could quickly converge to the optimal solution, and that an increased population size correlated with a faster convergence.

Table 4 further consolidates these findings by presenting detailed calculation results.

Table 4. Performance of PB-BPSO with varying population sizes.

<i>N</i>	Mean	RMSE (%)	Time (s)
50	47.6028	0.0367	0.5506
75	47.6067	0.0350	0.7795
100	47.6087	0.0342	1.3117
150	47.6152	0.0143	2.3131
200	47.6162	0.0132	2.2330

From the table, a clear trend emerges: as the population size increased from 50 to 200 the RMSE value of the particle swarm decreased significantly, from 0.0367% to 0.0132%, indicating an enhanced optimization capability. Meanwhile, the computation time scaled linearly, increasing from approximately 0.55 s to 2.23 s. However, in a parallel computing environment, the proposed PB-BPSO algorithm is anticipated to achieve superior performance, due to its ability to harness the increased population size effectively.

5.2. Performance Comparison on Benchmark Functions

To further evaluate the ability of the PB-BPSO algorithm developed in this study to seek the global optimal solution, six academically recognized standard test functions were selected for testing. Table 5 provides the details of these six benchmark functions, including their formulations, feasible domains, and theoretical global optimal solutions. Among these, f_1 to f_4 are unimodal functions, while f_5 and f_6 are multimodal functions.

Table 5. Benchmark functions.

Function	Formula	<i>n</i>	Range	f_{min}
f_1	$\sum_{i=1}^n x_i^2$	20	$-100 \leq x_1 \leq \dots \leq x_n \leq 100$	0
f_2	$\sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	20	$-10 \leq x_1 \leq \dots \leq x_n \leq 10$	0
f_3	$\max\{ x_i , 1 \leq i \leq n\}$	20	$-100 \leq x_1 \leq \dots \leq x_n \leq 100$	0
f_4	$\max\{ x_i , 1 \leq i \leq n\}$	20	$-1.28 \leq x_1 \leq \dots \leq x_n \leq 1.28$	0
f_5	$\sum_{i=1}^n x_i^2 - 10 \cos 2\pi x_i + 10$	20	$-5.12 \leq x_1 \leq \dots \leq x_n \leq 5.12$	0
f_6	$-20 \exp -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} - \exp \frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) + 20 + e$	20	$-32 \leq x_1 \leq \dots \leq x_n \leq 32$	0

A comparative analysis was conducted with four mainstream heuristic algorithms, including the classic genetic algorithm (GA), elitist genetic algorithm (MGA), pareto search algorithm (PS), and simulated annealing (SA). The core parameter settings for these algorithms are listed in Table 6. Note that these benchmark functions and models are primarily used for continuous optimization problems. In this comparative experiment, the particle swarm optimization (PSO) algorithm was adapted, based on priority-based principles, to be suitable for continuous optimization, referred to as the PB-PSO algorithm. The maximum number of iterations for stopping was set to 200 for all the algorithms.

Table 6. Parameter settings for algorithms.

Algorithm	Parameter	Value
GA	Population	200
	Crossover Fraction	0.8
	Elite Count	10
MGA	Population	200
	Crossover Fraction	0.8
	Pareto Fraction	0.35
	Max Stall Generations	50
PS	Pareto Set Change Tolerance	0.0001
	Pareto Set Size	60
	Constraint Tolerance	10^{-6}
	Constraint Tolerance	10^{-6}
SA	Initial Temperature	100
	Max Function Evaluations	60,000
	Max Stall Iterations	10,000
	Reanneal Interval	100
PB-PSO	population	200
	c_1, c_2	2
	w_{min}	0.4
	w_{max}	0.8

Table 7 presents the computational results of all the models on the test functions.

Table 7. Performances on six benchmark functions.

Method	Metric	f_1	f_2	f_3	f_4	f_5	f_6
GA	Avg.	3425.70	15.25	76.92	0.94	95.17	15.51
	Std.	17,989,309.41	61.02	593.10	0.17	1835.57	34.46
	Time (s)	26.54	27.3	26.02	26.74	26.77	27.07
MGA	Avg.	11,712.56	114.09	87.40	1.04	224.09	18.73
	Std.	238,221,728.54	1959.07	715.34	0.06	9325.08	3.67
	Time (s)	25.80	25.23	25.22	25.54	25.54	23.39
PS	Avg.	3.53	0	0	0	0	0
	Std.	190.75	0	0	0	0	0
	Time (s)	81.16	72.39	50.21	33.22	58.87	72.84
SA	Avg.	20.00	21.00	1.00	1.00	20.00	3.63
	Std.	0	0	0	0	0	0
	Time (s)	0.06	0.07	0.08	0.07	0.07	0.09
PB-PSO	Avg.	0	0	0	0	0	0
	Std.	0	0	0	0	0	0
	Time (s)	0.46	0.49	0.43	0.46	0.55	0.74

The proposed PB-PSO algorithm consistently found the global optimal solution in all tests, regardless of whether the functions were unimodal or complex multimodal, with computation times not exceeding 0.8 s. In contrast, the GA-type approaches demonstrated weaker optimization abilities. Even for simple unimodal functions like f_1 , the GA and MGA algorithms showed considerable deviation from the global optimal solution and high variability in the best solutions obtained across 10 repeated experiments, with computation times exceeding 20 s. While the PS algorithm exhibited good optimization capability, its iteration time was nearly 100 times that of the PB-PSO algorithm. Although the SA algorithm had a significantly shorter computation time than the proposed algorithm, its error from the optimal solution was relatively large. Therefore, overall, the algorithm proposed in this paper is more suitable for solving global optimization problems with sequential constraints.

Furthermore, Figures 2 and 3 depict the trajectory variations of the optimal solutions for six algorithms on the unimodal benchmark function f_2 and the multimodal benchmark

function f_6 , respectively. The horizontal axis indicates the number of trials, while the vertical axis represents the best value obtained by each algorithm. It can be observed that the proposed PB-PSO algorithm significantly outperformed the other algorithms on both types of test functions:

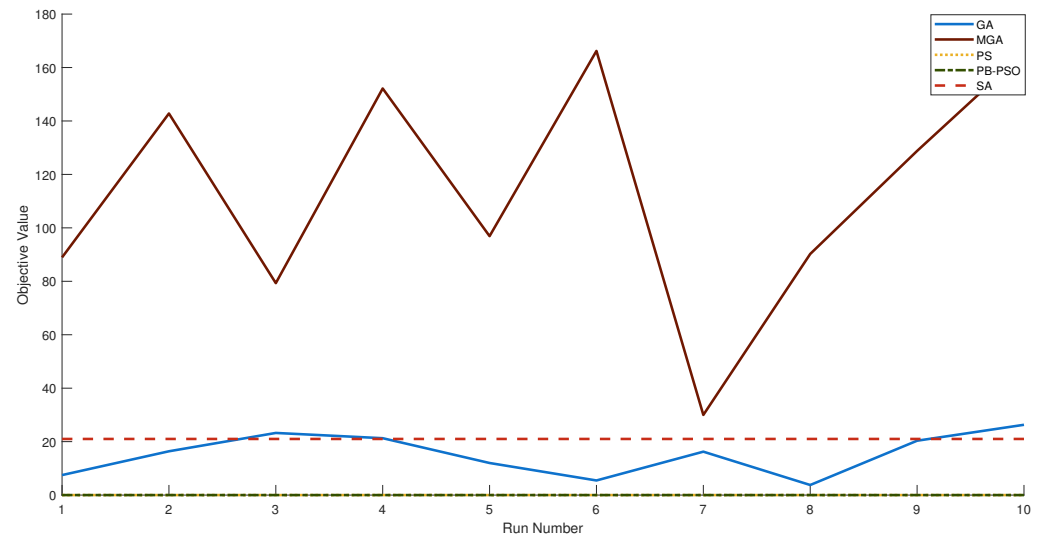


Figure 2. Trajectories of objective values for six algorithms tested on f_2 .

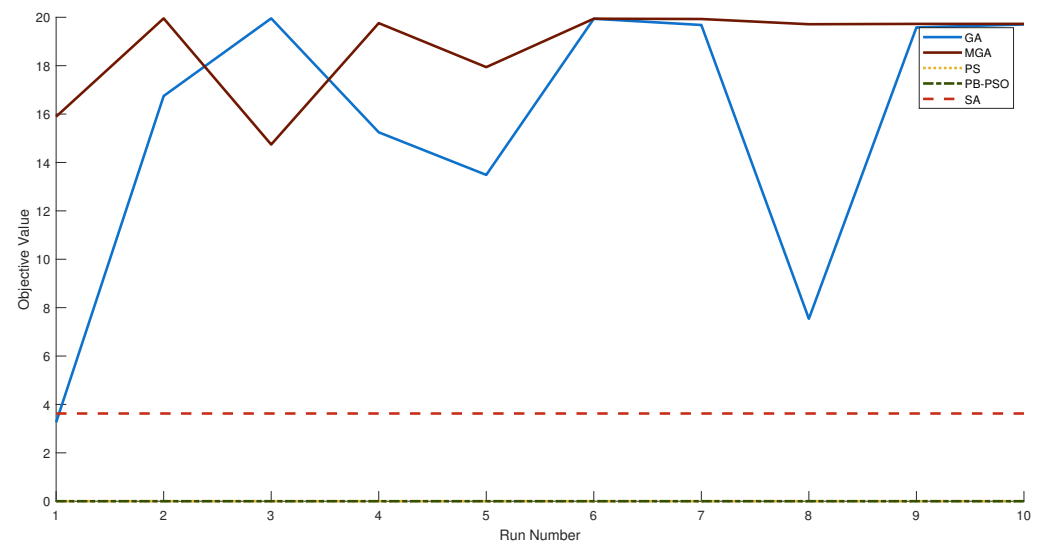


Figure 3. Trajectories of objective values for six algorithms tested on f_6 .

5.3. Empirical Analysis

This empirical study assessed the practical effectiveness of the model and the lexicographic mechanism (see [7]) through its application in Changzhou City, Jiangsu Province, China. In this city, a total of 1538 retailers procure products from the company weekly. These retailers are categorized into 30 distinct priority levels, ranging from the 1st level with the lowest priority to the 30th level with the highest. Given the company's limited resources, allocations to customers are governed by a strict priority-based system.

Historical data spanning from May 2021 to July 2022 were utilized to compute the average values of the four crucial indicators in the model. Recognizing that the company adapts its business strategy dynamically based on the product's market conditions, this analysis incorporated both the adjustment and steady-state periods of the product life cycle. During the adjustment period, as total market sales declined, the company tended to prioritize order fulfillment rates. Conversely, in the steady-state period, where the market

price of the product remained relatively stable, the company aimed to balance both order fulfillment rates and coverage ratios.

Initially, a classic lexicographic mechanism (see [7]) was employed, to develop a capacity allocation strategy, with the results presented in Table 8. Using this mechanism, during the adjustment period, production capacity was allocated to distributors with higher grades in a centralized manner. Specifically, channel partners ranked from 25 to 30 received a capacity allocation of 3 units, while those ranked from 22 to 24 received 1 unit. The remaining 21 distributors with lower grades were not allocated any production capacity. In contrast, during the steady-state period, the allocation of production capacity was relatively balanced, with distributors ranked above 14 each receiving 1 or 2 units of capacity.

Table 8. Capacity allocations using the lexicographic mechanism.

Capacity	Adjustment Period			Steady-State Period		
	3	1	0	2	1	0
Rank	25–30	22–24	1–21	29–30	14–28	1–13

The proposed model was then used to determine the optimal capacity allocation for distributors across 30 ranks. Based on practical considerations and interviews with business personnel, the coefficients for the four indicators were set as follows: During the adjustment period, the weights were set to $\lambda_1 = 1$, $\lambda_2 = \lambda_3 = \lambda_4 = 0.001$; During the steady-state period, the weights were set to $\lambda_1 = \lambda_2 = 1$, $\lambda_3 = \lambda_4 = 0.001$. The total capacity was set at $C = 4000$ during the adjustment period and $C = 16,000$ during the steady-state period. Additionally, in accordance with the company's regulations, regardless of the retailer's priority level, the weekly order quantity was capped at three boxes of product. Therefore, in this experiment, n was fixed at 3.

The outcomes are summarized in Table 9. Columns 2 through 5 display the results during the adjustment period, whereas Columns 6 through 9 present the outcomes during the steady-state period. The row labeled 'Capacity' enumerates all potential capacity allocations in descending order of 3, 2, 1, and 0 for both periods. The row labeled 'Rank' identifies the ranks of distributors who selected each respective capacity allocation.

Table 9. Capacity allocations using the integer programming method.

Capacity	Adjustment Period			Steady-State Period		
	3	1	0	2	1	0
Rank	29–30	20–28	1–19	26–30	14–25	1–13

From this table, it can be observed that during the adjustment period, the company strategically allocated its primary production capacity to the top two distributor ranks, specifically Rank 30 and Rank 29. Meanwhile, the subsequent nine ranks, spanning from Rank 20 to Rank 28, received a lesser share of the capacity. This strategy aimed to concentrate production resources on retailers with robust business capabilities, ultimately enhancing product sales. In contrast, during the steady-state period, the company's production capacity was distributed more evenly. Retailers ranked from 14th to 25th each received a capacity of one box of products, while those ranked from 26th to 30th each received a capacity of two boxes of products. This strategy suggests that the company should focus on balancing market supply and demand during the steady-state period. A comparison between Tables 8 and 9 reveals that the capacity allocation strategy derived from the integer programming model is more balanced than that obtained using the dictionary method.

6. Conclusions and Future Work

Hierarchical channel management is a prevalent marketing strategy in the fast-paced consumer goods industry, categorizing channel distributors into various ranks based on their operational capabilities, market demands, and other factors. However, this management strategy has received little attention in the literature. This study developed optimal production capacity allocation strategies for companies operating under capacity constraints while adhering to priority rules.

We abstracted this problem into a binary integer programming model, which stands out in two key respects: it prioritizes the company's business orientation and reshapes the capacity allocation challenge into an assignment-oriented problem. Based on the special structure of constraints, two efficient approaches are proposed: a greedy algorithm and a priority-based particle swarm optimization algorithm (PB-BPSO). The greedy algorithm allocates production capacity from higher to lower ranks, guided by customer enthusiasm, while PB-BPSO leverages a rank priority mechanism.

The main findings include that PB-BPSO performs well on both unimodal and multimodal benchmark functions, particularly in terms of accuracy and computational time. Real-world data from a large logistics company shows that the optimization model aligns with rank priority rules and effectively maximizes the company's business orientation, offering more balanced allocation strategies than the lexicographic mechanism that prevails in industries.

However, the proposed method has practical limitations. The model relies heavily on indicators that are difficult to predict accurately, impacting the precision of capacity allocation decisions. Setting appropriate parameters for these indicators is also challenging, requiring reasonable trade-offs based on experience, which reduces objectivity. These issues necessitate further in-depth research in subsequent work.

Author Contributions: Conceptualization, J.T. and L.L.; Methodology, J.T.; Software, X.Z.; Data curation, X.Z.; Writing—original draft, X.Z.; Writing—review & editing, J.T. and L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Zhejiang Provincial Philosophy and Social Sciences Planning Program in China, grant number 23NDJC094YB.

Data Availability Statement: The datasets presented in this article are not readily available because no new data were created or analyzed in this study.

Acknowledgments: The authors are grateful for helpful comments from the editor and anonymous reviewers.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

PSO	particle swarm optimization
BPSO	binary particle swarm optimization
PB-BPSO	priority-based particle swarm optimization

References

1. Thongyod, S. *The Strategic Analysis of Huawei Investment & Holding Co., Ltd.*; Technical Report; Stamford International University: Stanford, CA, USA, 2018. Available online: https://www.researchgate.net/publication/335146657_The_Strategic_Analysis_of_Huawei_Investment_Holding_Co_Ltd (accessed on 7 July 2024).
2. Kasimov, M. *Procter and Gamble Second Tier Distributors Use Pastel Evolution to Drive Efficiency, Sales*; Technical Report; ITWeb: Sandton, South Africa, 2009. Available online: <https://www.itweb.co.za/article/procter-and-gamble-second-tier-distributors-use-pastel-evolution-to-drive-efficiency-sales/4r1lyMR1Bed7pmda> (accessed on 7 July 2024).
3. Iyer, A.V.; Deshpande, V.; Wu, Z. A postponement model for demand management. *Manag. Sci.* **2003**, *49*, 983–1002. [[CrossRef](#)]

4. Hall, N.; Liu, Z. Capacity allocation and scheduling in supply chains. *Oper. Res.* **2010**, *58*, 1711–1725. [\[CrossRef\]](#)
5. Lee, H.L.; Padmanabhan, V.; Whang, S. Information distortion in a supply chain: The bullwhip effect. *Manag. Sci.* **1997**, *43*, 546–558. [\[CrossRef\]](#)
6. Cachon, G.P.; Lariviere, M.A. Capacity allocation using past sales: When to turn-and-earn. *Manag. Sci.* **1999**, *45*, 685–703. [\[CrossRef\]](#)
7. Chen, F.; Li, J.; Zhang, H. Managing downstream competition via capacity allocation. *Prod. Oper. Manag.* **2013**, *22*, 426–446. [\[CrossRef\]](#)
8. Li, J.; Cai, X. Allocating capacity with demand competition: Fixed factor allocation. *Decis. Sci.* **2017**, *48*, 523–560. [\[CrossRef\]](#)
9. Fu, K.; Hsu, V.; Lee, C.Y. Inventory and production decisions for an assemble-to-order system with uncertain demand and limited assembly capacity. *Oper. Res.* **2006**, *54*, 1137–1150. [\[CrossRef\]](#)
10. Alsabab, H.; Bernard, B.; Capponi, A.; Iyengar, G.; Sethuraman, J. Multiregional oligopoly with capacity constraints. *Manag. Sci.* **2020**, *67*, 4789–4808. [\[CrossRef\]](#)
11. Tana, L.; Wanga, F.; Lib, K. Optimal multichannel distribution strategies with limited capacity and uncertain demand. *Comput. Ind. Eng.* **2023**, *185*, 109659. [\[CrossRef\]](#)
12. Glasserman, P. Allocating production capacity among multiple products. *Oper. Res.* **1996**, *44*, 724–734. [\[CrossRef\]](#)
13. Laskari, E.; Parsopoulos, K.; Vrahatis, M. Particle swarm optimization for integer programming. In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; Volume 2, pp. 1582–1587.
14. Hsieh, F.S. Development and comparison of ten differential-evolution and particle swarm-optimization based algorithms for discount-guaranteed ridesharing systems. *Appl. Sci.* **2022**, *12*, 9544. [\[CrossRef\]](#)
15. Liu, X.; Li, A.D. An improved probability-based discrete particle swarm optimization algorithm for solving the product portfolio planning problem. *Soft Comput.* **2024**, *28*, 2535–2562.
16. Li, L.; Chen, J.; Yan, W. A particle swarm optimization-based container scheduling algorithm of docker platform. In Proceedings of the 4th International Conference on Communication and Information Processing, Qingdao, China, 2–4 November 2018; pp. 12–17.
17. Goudarzi, P.; Rahmani, A.; Mosleh, M. A mixed-integer programming model using particle swarm optimization algorithm for resource discovery in the cloudiot. *Peer Peer Netw. Appl.* **2022**, *15*, 2326–2346. [\[CrossRef\]](#)
18. Şahin, M.; Kellegöz, T. A new mixed-integer linear programming formulation and particle swarm optimization based hybrid heuristic for the problem of resource investment and balancing of the assembly line with multi-manned workstations. *Comput. Ind. Eng.* **2019**, *133*, 107–120. [\[CrossRef\]](#)
19. Zhang, Q.; Yu, L.; Xiahou, T.; Huang, H.Z. A heuristic maintenance scheduling framework for a military aircraft fleet under limited maintenance capacities. *Reliab. Eng. Syst. Saf.* **2023**, *235*, 109239. [\[CrossRef\]](#)
20. Elbes, M.; Alzubi, S.; Kanan, T.; Al-Fuqaha, A.; Hawashin, B. A survey on particle swarm optimization with emphasis on engineering and network applications. *Evol. Intell.* **2019**, *12*, 113–129. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.