Capacity Planning to Cope with Demand Surges in Fourth-Party Logistics Networks under Chance-Constrained Service Levels

Songchen Jiang[a], Min Huang[a,*], Yunan Liu[b], Yuxin Zhang[a], Xingwei Wang[c]

[a] *College of Information Science and Engineering, Northeastern University, Shenyang, Liaoning, 110819, China*
[b] *Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC 27695-7906*
[c] *College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China*

Abstract

In this paper, we study a capacity planning problem for a *fourth-party logistics network* (4PLN) in the face of event-triggered demand surges. We aim to solve a stochastic optimization problem in order to minimize the total cost for the 4PLN under chance-constrained service-level targets, where the stochastic demand process is modeled as a summation of random variables with a Bernoulli term of jump processes. At the heart of our solution procedure is a greedy pricing and weighting strategy based cell-and-bound (G-C&B) algorithm designed for solving the SAA-based model. Compared to the standard C&B method, our G-C&B is able to largely reduce the number of non-essential cell enumerations and achieve reduced running time complexity. To mitigate the performance degradation due to large system scale and/or sample instance, we extend our base algorithm to a two-step ***L**ocal **E**xperimentation for **G**lobal **O**ptimization strategy based cell-and-bound* (LEGO-C&B) framework, in which we first solve a small-scale training problem to find the important scenarios (eliminating excessive cell enumerations) and then use the training results to expedite the full optimization problem. We evaluate the performance of our algorithms by conducting a comprehensive series of numerical experiments. Besides, our results also demonstrate how the effectiveness of our methods depends on various factors including (i) the algorithm's hyperparameters such as the sample size and training ratio, and (ii) the 4PLN's input parameters such as the network scale, surge demand frequency, and rental price of 3PL resource. Our results exhibit several qualitative insights.

Keywords: Logistics, 4PL network capacity planning, Demand surge, Chance-constrained stochastic programming, Local experimentation for global optimization (LEGO) framework

*Corresponding author
Email addresses: sjiang@stumail.neu.edu.cn (Songchen Jiang), mhuang@mail.neu.edu.cn (Min Huang), yliu48@ncsu.edu (Yunan Liu), neuzyx@stumail.neu.edu.cn (Yuxin Zhang), wxwang@mail.neu.edu.cn (Xingwei Wang)

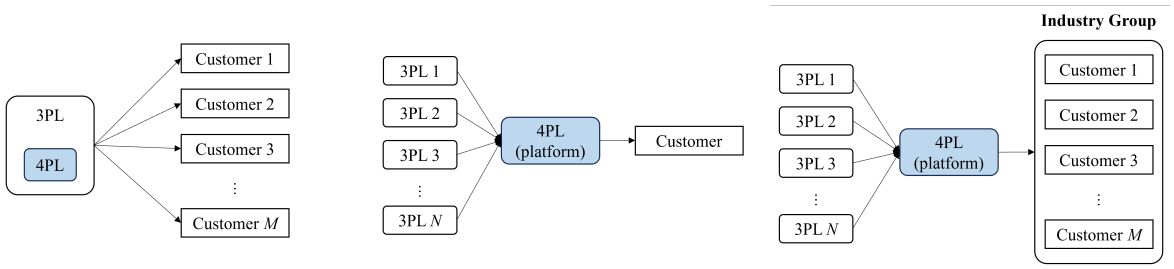
1. Introduction

In response to the escalating imperative for enhancing the quality of service in logistics and supply chain systems, *fourth-party logistics* (4PL) emerges as a new operational model that successfully integrates the traditional supply chain management technology and the contemporary mechanisms of the sharing economy. Expanding upon the conventional *third-party logistics* (3PL) model that focuses on its own customer cluster's services, 4PL adds the *platform* (i.e., the "fourth party") to the supply chain model in order to integrate the capabilities of multiple 3PLs and information from multiple customers. The 4PL model improves the cost-effectiveness of logistics enterprises and enhances service quality, offering unique advantages over emerging financing approaches such as bank loans and e-commerce platform funding (Bi et al., 2024). While the notion of 4PL is not recent[1], proper theoretical models of 4PL began to emerge in the field of operations research only recently. There are three extant models in the 4PL literature (Büyüközkan et al., 2009): (*i*) the *synergy plus* model where 4PL functions serve as auxiliary departments of the 3PL (Kutlu, 2007); here 4PL acts as the decision maker to help address operational management issues for the 3PL (Marasco, 2008); (*ii*) the *solution integrator* model where 4PL aims to integrate multiple 3PLs to provide a solution for a particular customer, examplified by the 4PL routing problem (Liu et al., 2014; Huang et al., 2015; Tao et al., 2017), and (*iii*) the *industry innovator* model where 4PL serves as a more comprehensive platform to integrate multiple 3PLs and additional customer information to provide integrated solutions, such as the risk management of outsourcing (Huang et al., 2019; Guchhait & Sarkar, 2025), resource auction (Yu et al., 2024), and 4PL network planning (Zhang et al., 2022; Jiang et al., 2024; Zhang et al., 2024a,b). Among the aforementioned three 4PL models, the industry innovator model has the largest information integration capabilities. One practical example of the industry innovator model is Cainiao Smart Logistics Network, a Chinese 4PL platform for Tmall[2]. To achieve high-quality quick delivery services, Cainiao integrates the resources of more than 30 major domestic 3PL entities including EMS, ZTO Express, and YUNDA. The effective integration of 3PL resources enables Cainiao to effectively reduce the courier cost, and shorten delivery times; allowing for same-day and next-day delivery services in most areas within China. In this study, we focus our attention on the industry innovator 4PL model.

The paramount business question integral to the design and management of 4PLN is how the service capacity can be planned so as to minimize the overall operational costs. Such an optimal capacity planning problem is often subject to some designated *service-level* (SL) constraints, which are crucial in many scenarios (see Sinha et al. (2023); Cao et al. (2023) for instances). One predominant SL metric is the *fraction of customer demand fulfilled*, in particular, the 4PL manager hopes to achieve the SL level in the form of a

[1]The first proposal of 4PL was introduced in 1998 by Accenture (Gattorna & Jones, 1998).

[2]Tmall is a major online marketplace for business-to-consumer (B2C) online retails in China, and for more details, refer to https://www.exportnow.com/2014/06/tmall-introduction-worlds-largest-e-commerce-marketplace/



(a) Synergy plus mode (b) Solution integrator mode (c) Industry innovator mode (in our study)

Figure 1: Three different modes of 4PLNs.

chance constraint

$$\mathbb{P}(S \geq D) \geq \alpha, \tag{1}$$

where S is the service capacity, D is the customer demand, and $\alpha \in (0, 1)$ is some desired probability target (e.g, $\alpha = 95\%$). Constraints structured in the form of (1) are also referred to as *type-1 service level* (Lyu et al., 2022) and are widely used in many service systems such as customer contact centers and healthcare systems; see Liu (2018); Liu et al. (2022) for examples.

The most challenging aspect of making optimal capacity plans for 4PL is the uncertainty in customer demand. Undesired violation of the SL chance constraint may occur due to demand surges. Demand surges manifest in two principal forms: (i) the random demand surge, propelled by the intrinsic stochastic nature of the demand process or unforeseeable natural causes, and (ii) the event-triggered demand surge, stemming from specific service-related events such as promotional activities or shopping festivals (*e.g.*, Black Friday, Amazon's Prime Day, Tmall's Double 11 shopping carnival). In 2023, sales achieved an astounding $160 billion in gross merchandise value, exceeding Amazon's Prime Day sales by over ten times and nearly eclipsing U.S. Black Friday spending by a similar margin [3]. By October 30, 2024, total sales for the "Double 11" shopping festival had already soared to an impressive 845 billion yuan [4]. For another example, we illustrate Wal-Mart's demand surges using its sales data from 45 stores[5]. Figure 2 clearly shows that demand surges occur on four event-triggered special days during a year: Super Bowl, Labor Day, Thanksgiving Day, and Christmas.

In response to the challenges posed by event-triggered demand surges, 4PL managers need to proactively deploy logistics resources to augment service capacity, ensuring the attainment of designated SL targets. To enhance service performance, the integration of functional facilities, such as transshipment centers, warehouses, logistics ports, and transportation providers into the 4PLN, becomes imperative. Two predominant

[3]https://tryon.kivisense.com/blog/double-11-2024-insights/

[4]https://www.globaltimes.cn/page/202411/1322863.shtml

[5]Data sourced from the Aliyun Tianchi open-source dataset: "Walmart Recruiting - Store Sales Forecasting." The dataset encompasses historical sales data from 45 Walmart stores across diverse regions, highlighting selected holiday promotional events that impact sales. https://tianchi.aliyun.com/dataset/dataDetail?dataId=90208.

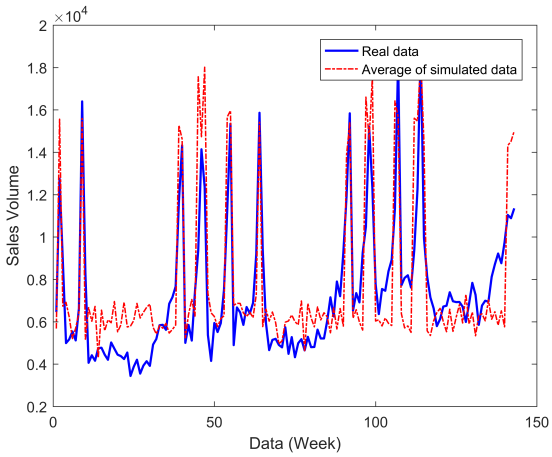


Figure 2: The weekly sales volume (the solid line) from 45 Wal-Mart stores (From February 5, 2010 to November 1, 2012).

options typically available to 4PL planners for capacity planning include: (*i*) self-building and (*ii*) renting. While self-building new facilities affords the advantage of achieving desired service capacities over the long term, it presents inflexibility in adjusting to short-term fluctuations. Conversely, opting to rent certain facilities from 3PL providers offers a temporary solution to fulfill demand surges. However, this approach often entails elevated operational costs.

The 4PL model in this study is in general applicable to enterprises' network capacity planning. It takes into consideration demand surges arising from planned discount promotions and various other activities. Our analysis serves as an operational tool, offering insights into the optimal balance between self-built and rented logistics resources in different surge scenarios, facilitating the development of a cost-effective network solution subject to constraints on desired service level targets.

1.1. Main Contribution

In this paper, we study an optimal service capacity planning problem for 4PL with event-triggered demand surges. We aim to seek the optimal service capacity (i.e., the number of facilities to build and the number of 3PL facilities to rent) so as to minimize the total operational costs subject to SL constraints in form of (1). We make the following contributions:

- We study a novel 4PL capacity planning problem with event-triggered demand surges subject to chance constraints on the SL. To characterize the surged demand process, we formulate it as a sum of Bernoulli terms of jump processes.

- We aim to solve a stochastic program under a joint chance-constrained service target. We reformulate the chance constraint as a SAA-based mixed integer program (MIP) knapsack sub-problem. To improve the solution efficiency of the knapsack problem, we give another reformulation of the problem using hyperplane arrangement to transform various knapsack schemes, i.e., the satisfaction of scenarios, into

cells of hyperplanes. Further, we devise two algorithms within the cell-and-bound (C&B) framework: greedy pricing and weighting strategy based C&B (G-C&B) and local experimentation for global optimization strategy based C&B (LEGO-C&B). We believe that the scope of these two methods goes beyond the present problem (they can potentially be used to solve other chance-constrained stochastic programs).

- We confirm the effectiveness of our methods by conducting a comprehensive set of numerical studies. These results show that the performance of our algorithms is robust to relevant model parameters. Moreover, we investigate the effect of these model parameters on the network design strategy and generate some interesting insights. For example, when the surge frequency is low, renting 3PLs is a more economic decision; on the other hand, as the surge frequency increases, the manage leans to use rented 3PLs to self-building.

1.2. Notation and Organization

Notably, we employ $\mathbb{P}(\cdot)$ to denote the probability of an event occurring, and $\mathbb{E}(\cdot)$ to signify the expectation of a random variable. All vectors are presented in boldface type (*e.g.*, $\mathbf{d}$), and $\mathbf{e}$ denotes the vector whose elements consist of all ones, expressed as $(1, 1, \cdots, 1)^n$. Additionally, we utilize the notation $[K] = \{1, 2, \ldots, K\}$ to represent the set of positive running indices ranging from 1 to K. Throughout this paper, a random variable is consistently denoted with a tilde symbol.

The remainder of this paper is organized as follows. In Section 2, we review the related literature. In Section 3, we introduce the capacity planning model for 4PLN with demand surges and formulate our capacity planning problem as a chance-constrained stochastic program. In Section 4, we propose two algorithms and a two-stage learning-based optimization framework to enhance the algorithm's effectiveness, further G-C&B algorithm and LEGO-C&B algorithm are designed separately. In Section 5, numerical experiments based on realistic scenarios are conducted to demonstrate the effectiveness of the developed model and algorithms, and we reveal the laws of 4PLN designing under demand surge. Section 6 concludes the paper. All proofs are relegated to Appendix A. In Appendix B, we present a 4PL network design example along with decision trees for each algorithm, demonstrating the efficiency of the proposed G-C&B algorithm. Appendix C provides additional details on the three other stochastic models used for comparison. In Appendix D, we elaborate on the goodness-of-fit test conducted on samples of varying sizes to illustrate the convergence of the SAA-based method.

2. Literature Review

This paper studies a 4PLN planning problem with premeditated demand surge, which is a supply chain network design (SCND) problem with demand uncertainty under 4PL mode. To further clarify our research orientation, we separately reviewed the literature on the issue of 4PLN planning and demand uncertainty in 4PLN.

2.1. 4PLN Planning

In essential, 4PLN is a kind of supply chain network, the literature on SCND is extensive, and for more details, refer to the survey of Melo et al. (2009). Here, we only review the literature related to the planning of 4PLN. In a 4PLN planning problem, different from SCND, the decision maker not only determines the optimal location of stationary facilities, and allocations among them but also focuses on the selection of the corresponding 3PL TP(s) between stationary facilities and the flow on each selected 3PL TP. The transportation capacity on each 3PL TP is one of the main factors impacting the network capacity. Although there are frequently several optional 3PL TPs between facilities, this aspect is frequently overlooked in the earlier studies of SCND. To better plan the network capacity, the selection of 3PL TPs has been thoroughly considered in the 4PLN planning problem. In the study by Wang et al. (2021), a 4PLN model is designed for determining DCs, 3PL TPs, and flow assignments to achieve the goal of maximizing the service satisfaction of suppliers and customers under a limited budget. Yin et al. (2021) studied the designing of 4PLN considering delivery time, in which a cost-saving 4PLN design scheme is obtained through the decoding of the opening of DCs and selection of 3PL TPs. Zhang et al. (2024a) utilized prospect theory to model customer satisfaction with logistics services and investigated the selection of DCs and 3PL providers within budget constraints to design a 4PLN that optimizes customer satisfaction. It can be seen that, as an extension of SCND, the introduction of decision variables for 3PL TPs' selection in 4PLN is the main difference from SCND, and aforementioned studies also point out that this added kind of decision variables significantly increase the difficulty of problem-solving. Therefore, in the study of 4PLN planning, how to solve the problem more effectively is a notable challenge.

2.2. Demand Uncertainty in 4PLN

It is necessary and meaningful to study 4PLN planning under uncertainty to make its performance satisfactory in a changing environment because, as a long-term strategy decision, the planning of 4PLN usually cannot be adjusted for a long period. The comprehensive review by Govindan et al. (2017) points out that supply chain networks face several uncertainties, the most significant of which is the uncertainty of customer demand. The uncertainty of customer demand is a widely considered factor in existing studies, and it has a clear impact on supply chain network design. In previous studies around the planning of 4PLN, customer demand is usually considered as a stochastic variable, and distribution is used to portray it (Wang et al., 2021; Yin et al., 2022). However, in today's e-commerce environment, demand surges triggered by man-made causes such as promotion occur all the time. While the occurrence of such event-triggered surges is predictable, the uncertainty of the magnitude of the surge still poses a challenge for 4PLN (supply chain network) planning. To accurately portray the demand surge, Huang et al. (2016) concluded some surge trajectories of demand and developed a demand model that captures those characteristics of surge trajectories, such as magnitude, duration, and intensity, and then solve a reactive capacity planning problem for the supply chain. This study provides an effective framework to model random demand surges

whose occurrence is considered completely random, and several studies have taken into account such kind of demand surges as a risk in their operation problem in different fields, such as inventory management (Roni et al., 2015, 2016) and resource allocation (Liu et al., 2021). When considering such random demand surges, the most important problem for companies is how to do reactive capacity planning. Different from the aforementioned studies, the occurrence of the event-triggered demand surges considered in this paper is predictable, this motivates us to consider, from a strategic level perspective, how to address this risk by doing rational network designing and capacity planning. Therefore, in this paper, we investigate a 4PLN planning problem considering event-triggered demand surges under the industry innovator mode of 4PL and give the analysis of how companies effectively cope with demand surges in a sharing economy background.

2.3. Methods for Chance-Constrained Programs

In operational management problems, service level constraints in the form of formula (1) are often used to limit the probability of stockouts; such models with probability constraints are referred to as *chance-constrained programming* (CCP) model. See for example Bookbinder & Tan (1988); Chen & Krass (2001); Lyu et al. (2022). Constrained optimization problems sometimes can be efficiently solved with some mild conditions (see Xiao et al. (2024) for example), however chance constraint is still hard to handle due to its probabilistic characteristics. The challenge in solving CCP lies in evaluating the probability values corresponding to every candidate solution. The relevant literature can be categorized into two streams specified by the availability of the distributions of the random elements in the stochastic program.

Distribution is known. Having the exact form and parameters of the distribution allows us to accurately characterize the random variable. Although it is possible to calculate probabilities using the cumulative distribution function if the distribution of the random variable is given, this is considered quite challenging, as they plays an important role in chance constraints. The probabilistic constraints can be reformulated into linear or second-order cone constraints under a Gaussian-like assumption (Nemirovski & Shapiro, 2007). For uncertain demand, the Gaussian distribution is widely used in supply chain network design (Shu et al., 2005; Chou et al., 2010), or one may directly assume that the demand follows a known discrete distribution (Yin et al., 2021, 2022; Zhang et al., 2022). This allows us to further treat the random terms, thus making it possible to generate closed-form expressions for the expectation or probability of the associated random variables.

Distribution is unknown. Unfortunately, in real-world problems, we often cannot obtain the exact distribution of random variables but only have access to their limited information such as moments or limited historical data. To solve the CCP problem with limited information, an effective approach is to develop a *mixed-integer programming* (MIP) reformulation. For CCPs with individual chance constraints, we refer to the comprehensive review Küçükyavuz & Jiang (2022). Here we focus on the reviewing CCPs with joint chance constraints as considered in our study (in the form of formula (1)). These methods can be generally divided into two categories:

(i) The first approach is the *sample average approximation* (SAA), which uses finite discrete distributions to transform the CCP as a MIP with knapsack sub-problem (Ruszczyński, 2002). This reformulation can be solved by state-and-the-art solvers. However, this model has been proven to be NP-hard (Luedtke et al., 2010), which drives the researchers to seek effective alternatives, such as branch-and-price and cutting planes, to accelerate the solution of the knapsack sub-problem (Liu et al., 2016; Deng et al., 2021). Another approach for the SAA-based reformulation or CCP directly is to build approximate models with valid inequalities, see Porras et al. (2023); Zhang et al. (2023). However, these approximations necessitate breaking down the multivariate distribution corresponding to joint chance constraints into multiple univariate distributions, resulting in estimation bias for the original distribution or conservative decision-making, such as CVaR approximation (Alexander & Baptista, 2004; Nemirovski, 2012) and Bonferroni approximation (Nemirovski & Shapiro, 2007; Chen et al., 2010).

(ii) Another approach for CCPs under limited information is to model the stochastic program as a distributionally robust program where the chance constraint is modeled under an ambiguity set. The main dowside of this approach is the conservativeness of its solution, because the essential approach is based on the analysis of the worst case. We refer to Rahimian & Mehrotra (2019) for a more comprehensive review. In our study, to address service level constraints on demand surges, we employ an SAA-based MIP reformulation, utilizing finite samples to solve the CCP model. To address the computational challenges due to the need for a large number of samples for accurate simulation, we propose several novel algorithms using a greedy strategy and a learning framework, specifically designed for the large-scale knapsack sub-problems.

3. Problem Description and Formulation

In this section, we introduce our 4PLN model, describe the model assumptions, and specify the capacity optimization problem.

3.1. The 4PLN Model

In a 4PLN, commodities are delivered from suppliers to customers through a *distribution center* (DC) by 3PL *transportation providers* (TPs). This network can be specified as a multi-graph (see in Figure 3), where suppliers, DCs, and customers are represented as nodes and the 3PL TPs as edges connecting these nodes. There can be multiple edges connecting two nodes in the graph under 4PL operation mode. In what follows, we sometimes refer to DCs and TPs collectively as 3PL facilities for ease of description.

Let $\mathcal{S}$, $\mathcal{F}$ and $\mathcal{D}$ denote the sets of suppliers, DCs and customers. We assume that all suppliers have stable supply capacity, and DCs have a fixed location cost, unit processing cost, and a finite processing capacity. The collection of nodes within the 4PLN network is denoted as $\mathcal{N} = \mathcal{S} \cup \mathcal{F} \cup \mathcal{D}$. All 3PL TPs possess predetermined construction costs, unit transportation expenses, and limited transport capacities. We define the quantity of 3PL entities as K, where $[K]$ signifies the group of 3PLs present on any compliant path connecting nodes, while all feasible paths are encompassed within the set $\mathcal{V}$. Then the 4PLN is fully

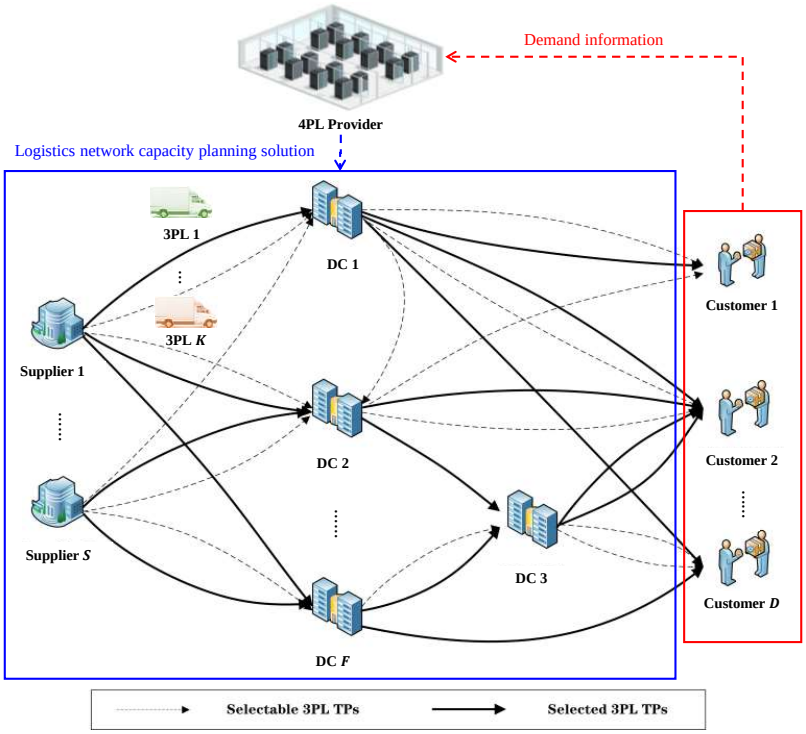


Figure 3: Multi-graph representation of 4PLN.

specified by the graph $\mathcal{G}(\mathcal{N}, \mathcal{V})$. The 4PLN planner aims to solve an optimal network design problem with the objective of minimizing the total capacity cost (of the service capacities that are self-built and rented) subject to constraints on the SL with designated targets (defined as the probability of all demand fulfilled). We assume that all decisions regarding self-building and renting are made at the beginning of a planning cycle. In the event of a demand surge, the pre-determined strategy to activate the rented 3PL resources is then implemented.

3.2. Stochastic Demand

In the above-mentioned 4PLN capacity optimization problem, we consider two demand types: regular demand and surged demand. Without loss of generality, we assume the demands among different customers are identically distributed.

The regular demand, denoted as $\tilde{\boldsymbol{d}}_r = (\tilde{d}_{r(1)}, \ldots, \tilde{d}_{r(|\mathcal{D}|)}) \in \mathbb{R}_+^{|D|}$, is the demand that occurs during the regular operational cycles. Following Govindan et al. (2017), we use Normal distributions to model the regular demand, that is, we assume that $\tilde{d}_{r(j)} \sim N\left(\mu_r, \sigma_r^2\right)$ for any $j \in \mathcal{D}$, so that

$$\mathbb{P}\left(\tilde{d}_{r(j)} = \xi_r\right) = \frac{1}{\sqrt{2\pi}\sigma_r} \cdot \exp\left(-\frac{\left(\xi_r - \mu_r\right)^2}{2\sigma_r^2}\right), \tag{2}$$

this can serve as a good approximation to the actual demand, especially when the business has a sufficient amount of demand data (following the law of large numbers). In our study, without loss of generality, we assume the enterprise has an adequate amount of regular demand data to fit the distribution parameters.

The surged demand, denoted by $\tilde{\boldsymbol{d}}_s = (\tilde{d}_{s(1)}, \ldots, \tilde{d}_{s(|\mathcal{D}|)}) \in \mathbb{R}_+^{|D|}$, represents demand triggered by events (*e.g.*, Black Friday). Unlike regular demand, surged demand occurs over a shorter duration and with significantly greater magnitude. Since surged demands are infrequent, it's challenging for firms to gather a substantial amount of data to explore their distribution. In our study, we model the dynamics of surged demand as a jump process, aiming to provide a work-well empirical model; see Cai & Yang (2021); Ewald & Zou (2021). Specifically, we assume that a demand surge in each customer $j \in \mathcal{D}$ occurs according to a jump process $\tilde{J}_{(j)}$ with the amplitude of the jump $\tilde{A}_{(j)}$ following a Normal distribution, namely,

$$\tilde{A}_{(j)} = \log(\tilde{J}_{(j)}) \sim N\left(\mu_a, \sigma_a^2\right), \tag{3}$$

where μ_a and σ_a^2 represent the mean and variance of the jump amplitude. Hence, the the surge demand can be written as

$$\tilde{\boldsymbol{d}}_s = \tilde{\boldsymbol{d}}_r + \tilde{\boldsymbol{J}} \tag{4}$$

We consider the planning for the 4PLN as a long-term project in a given planning horizon, as it is a strategy-level problem. A decision cycle is thought to be divided into numerous equal periods (denote the number as N), thus planners can temporarily rent 3PL facilities during each of those periods if necessary. Based on whether there is a surge in customer demand, all periods can be easily classified as *surged demand period* and *regular demand period*. For the ease of notation, we use an indicator variable η_t, with $\eta_t = 1$ representing there is an event leading to a surge in demand in period t, and $\eta_t = 0$ otherwise. The stochastic customer demand in period t, comprising both regular and surge components, is formulated as

$$\tilde{\boldsymbol{d}}_t = (1 - \eta_t) \cdot \tilde{\boldsymbol{d}}_r + \eta_t \cdot \tilde{\boldsymbol{d}}_s = \tilde{\boldsymbol{d}}_r + \eta_t \cdot \tilde{\boldsymbol{J}}, \tag{5}$$

hence the demand for each customer over the entire planning horizon is represented as a matrix $\tilde{\boldsymbol{D}} = (\tilde{\boldsymbol{d}}_1, \ldots, \tilde{\boldsymbol{d}}_N)$.

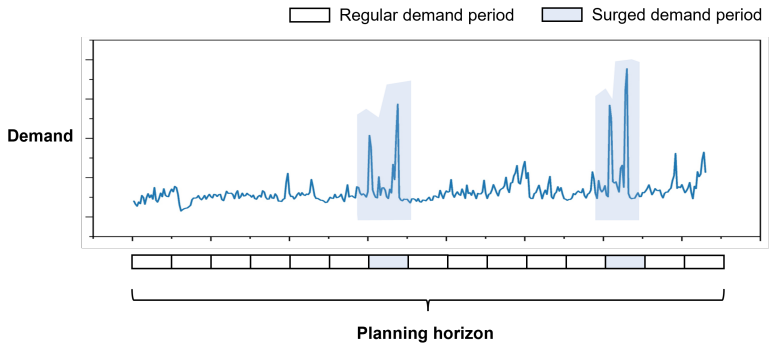


Figure 4: Stochastic demand in a planning horizon.

In the context of event-triggered demand surges, we assume that these events (*e.g.*, promotions) are prearranged, and their surge warnings are expected in advance, implying that the values of η_t are known to the planner. Additionally, without loss of generality, we assume that customer demands are identically distributed and the distribution is stationary.

To validate the above model of the surged demand, we conduct computer simulations of our demand model using parameters estimated from the Wal-Mart sales data. We show that our simulated results can effectively match with the real data. See Section 5.1 for details.

3.3. Rental of 3PL Facilities

To enhance network capacity to cope with demand surges, a 4PL planner can choose between self-building and renting 3PL facilities, where only temporary renting is allowed. And we assume that the rentable facilities have been determined before each decision cycle and are used when needed during each surge period. High construction expenses can be avoided by temporary renting, however, there will be additional rentals for 3PLs to use those rented facilities. Due to the different usage scenarios, usually, the rental for DCs and TPs are calculated differently, and we consider they are calculated as follows:

(i) To rent a DC, only the contractual cost is charged as rental based on a proportional θ_c of its construction cost. (ii) To rent a 3PL TP, besides the contractual cost will charged based on a proportional θ_t of its construction cost, additional unit transportation cost will charged on a proportional γ.

3.4. Modeling

All notations are summarized in Table 1.

We next describe four operational costs for the 4PLN.

(i). Base costs of 4PLN (including facility costs and construction costs of 3PL transportation carriers)

$$C_F = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^f_{ijk}\cdot x^r_{ijk} + \sum_{i\in\mathcal{F}} c^l_i \cdot y^r_i. \tag{6}$$

(ii). Costs of renting 3PL facilities and transportation carriers

$$C_R = \theta_c \cdot \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^f_{ijk}\cdot\left(x^s_{ijk} - x^r_{ijk}\right) + \theta_t \cdot \sum_{i\in\mathcal{F}} c^l_i\cdot\left(y^s_i - y^r_i\right). \tag{7}$$

(iii). Total operational costs during the regular demand period (including operating cost of 3PL transportation carriers and processing costs of 3PL facilities)

$$C_{PR} = \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot z^r_{ijk} + \sum_{i\in\mathcal{F}} c^p_i\cdot\left(\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} z^r_{ijk}\right). \tag{8}$$

(iv). Total operational costs during the surged demand period (including operating costs of 3PL transportation carriers, processing costs of 3PL facilities and remuneration of resource renting)

$$C_{PS} = \sum_{i\in\mathcal{F}} c^p_i\cdot\left(\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} z^s_{ijk}\right) + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot v_{ijk} + (1+\gamma)\cdot\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot w_{ijk}, \tag{9}$$

where $v_{ijk}\in\mathbb{R}_+$ and $w_{ijk}\in\mathbb{R}_+$ are auxiliary variables representing the transport volumes of the self-built and rented 3PL resource separately, and it satisfies $z^s_{ijk} = v_{ijk} + w_{ijk}$.

Table 1: Summary of notations.

Sets	
$\mathcal{S}$	Set of suppliers
$\mathcal{F}$	Set of DCs
$\mathcal{D}$	Set of customers
$\mathcal{V}_{ij}$	Set of compliant path between node i and j
$[K]$	Set of 3PLs (the quantity of 3PLs included is K)
Model parameters	
$\tilde{d}_i$	Stochastic demand of customer $i \in \mathcal{D}$
c_i^l	Fixed location cost of DC $i \in \mathcal{F}$
c_i^p	Unit processing cost in DC $i \in \mathcal{F}$
c_{ijk}^f	Fixed construction cost of 3PL TP $k \in \mathcal{V}$ between node i and node j
c_{ijk}^t	Unit transportation cost of 3PL TP $k \in \mathcal{V}$ between node $i \in \mathcal{N}$ and node $j \in \mathcal{N}$
h_{ijk}	Transport capacity of 3PL TP $k \in \mathcal{V}$ between node i and node j
θ_t	Proportion of contractual cost for 3PL TP charged at the time of renting
γ	Proportion of extra processing cost for 3PL TP charged at the time of renting
Decision variables	
x_{ijk}^m	Binary variable; the selection of 3PL TP $k \in \mathcal{V}$ between node $i \in \mathcal{N}$ and node $j \in \mathcal{N}$ during regular demand period denoted as $m = r$ and surged demand period as $m = s$
y_i^m	Binary variable; the selection of DC $i \in \mathcal{F}$ during regular demand period denoted as $m = r$ and surged demand period as $m = s$
Auxiliary decision variables	
z_{ijk}^m	Non-negative integer variable; constructed capacity of the 3PL TP $k \in \mathcal{V}$ between node $i \in \mathcal{N}$ and node $j \in \mathcal{N}$ during regular demand period denoted as $m = r$ and surged demand period as $m = s$

Denote the number of periods as N, and recall that the vector $\boldsymbol{\eta} = (\eta_1, \eta_2, \ldots, \eta_N)$ represents the surge warning information

The number of occurrences of the surged demand and regular demand are $N_S = \sum_{i=1}^{N} \eta_i$ and $N_R = N - N_S$. Define the frequency of the surged demand as $\varphi \equiv \frac{N_S}{N}$, then the total cost is derived as

$$C = C_F + N \cdot \Big[(1-\varphi) \cdot C_{PR} + \varphi \cdot (C_R + C_{PS}) \Big]. \tag{10}$$

The goal of the planning is to minimize the total cost specified in (10) subject to a constraint on the SL, which is defined as the probability of demand fulfilled. As we assume that the regular demand is normally distributed and there are sufficient historical data to estimate the moments of the distribution, hence the SL constraint for regular demand is presented in the following linear form using the z-score method, which is commonly used in inventory control or capacity planning to approximate demands (see, for example, Mak et al. (2013))

$$\sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{V}_{ij}} z_{ijk}^r \geq \mu_r + z_{\alpha_r} \cdot \sigma_r, \tag{11}$$

where $\sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{V}_{ij}} z_{ijk}^r$ represents the total capacity required for commodities fulfillment (those successfully

delivered to customer j), and z_α is the corresponding quantile value for type-1 service level α_r. And under constraint (11), we approximately calculate the long-run average operational cost during regular demand periods as formula (8), i.e., we consider this cost only at the target demand level of $\mu_r + z_{\alpha_r} \cdot \sigma_r$ to satisfy the regular service level requirement. Thus the total cost function is reformulated as

$$C = C_F + N \cdot \Big[(1-\varphi) \cdot C_{PR} + \varphi \cdot (C_R + C_{PS}) \Big].$$

For the surged demand, we write the SL constraint as follows

$$\mathbb{P}\left(\sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{V}_{ij}} z^s_{ijk} - \tilde{d}_{s(j)} \geq 0, \ \forall j \in \mathcal{D} \right) \geq \alpha_s, \tag{12}$$

where $\sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{V}_{ij}} z^s_{ijk}$ represents the total capacity required for commodities fulfillment for customer j in surged demand periods, and α_s is the designated target of type-1 service level for demand surges to ensure an acceptable SL. The SL constraint in (12) is a joint chance constraint; it provides a population-level control for the overall quality of the service rather than focusing on the individual experience of the customers. Specifically, when the demand from each customer is independent, we can equivalently use a set of individual chance constraints for each customer.

Putting things together, we give the complete version of our optimization problem as below.

$$\min \quad C_F + N \cdot \Big[(1-\varphi) \cdot C_{PR} + \varphi \cdot (C_R + C_{PS}) \Big] \tag{13a}$$

$$\text{s.t.} \quad \mathbb{P}\left(\sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{V}_{ij}} z^s_{ijk} - \tilde{d}_{s(j)} \geq 0, \ \forall j \in \mathcal{D} \right) \geq \alpha_s, \tag{13b}$$

$$\sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{V}_{ij}} z^r_{ijk} \geq \mu_r + z_{\alpha_r} \cdot \sigma_r, \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}_{ij}, \tag{13c}$$

$$x^r_{ijk} \leq x^s_{ijk}, \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}_{ij}, \tag{13d}$$

$$y^r_i \leq y^s_i, \qquad \forall i \in \mathcal{F}, \tag{13e}$$

$$\sum_{j \in N} \sum_{k \in V_{ij}} z^m_{ijk} \leq y^m_i \cdot q_i, \qquad \forall i \in \mathcal{F}, \forall m \in \{r, s\}, \tag{13f}$$

$$z^r_{ijk} + v_{ijk} \leq x^r_{ijk} \cdot h_{ijk}, \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}_{ij}, \tag{13g}$$

$$w_{ijk} \leq x^s_{ijk} \cdot h_{ijk}, \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}_{ij}, \tag{13h}$$

$$\sum_{j \in N} \sum_{k \in V_{ij}} z^m_{ijk} = \sum_{j \in N} \sum_{k \in V_{ij}} z^m_{jik}, \qquad \forall i \in \mathcal{F}, \forall m \in \{r, s\}, \tag{13i}$$

$$z^m_{ijk}, v_{ijk}, w_{ijk} \in \mathbb{R}_+, \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}_{ij}, \forall m \in \{r, s\}, \tag{13j}$$

$$x^m_{ijk} \in \{0, 1\}, \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{N}, \forall k \in \mathcal{V}_{ij}, \forall m \in \{r, s\}, \tag{13k}$$

$$y^m_i \in \{0, 1\}, \qquad \forall i \in \mathcal{F}, \forall m \in \{r, s\}. \tag{13l}$$

Constraint (13b) is a stochastic chance constraint, representing that the joint probability to meet surged demand in each customer is at least α_s; Constraint (13c) corresponds to the SL constraint for customers'

regular demand; Constraints (13d) and (13e) relate the self-built network to the rented network, requiring that the rented network is formed by incorporating additional 3PL resources into the self-built network; Constraint (13f)–(13h) are capacity constraints, imposing finite capacities for the 3PL transportation carriers, processing quantity in each 3PL facility; Constraint (13i) ensures flow conservation among all nodes, and constraints (13j)–(13l) specify the domains for all decision variables.

Let $\boldsymbol{x} = \left(x_{ijk}^m\right)_{i\in\mathcal{N},j\in\mathcal{N},k\in\mathcal{V}_{ij},m\in\{r,s\}}$, $\boldsymbol{y} = (y_i^m)_{i\in\mathcal{F},m\in\{r,s\}}$, $\boldsymbol{z} = \left(z_{ijk}^m\right)_{i\in\mathcal{N},j\in\mathcal{N},k\in\mathcal{V}_{ij},m\in\{r,s\}}$, with those notations, we rewrite the objective function that minimize the total cost, by defining

$$C(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) := C_F + N\cdot\Big[(1-\varphi)\cdot C_{PR} + \varphi\cdot(C_R + C_{PS})\Big],$$

and we use $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\in\boldsymbol{\chi}$ to denote the compact feasible set determined by constraints (13c)–(13l), except the joint chance constraint (13b). Then we give the equivalent representation of problem (14) as follows

$$\begin{aligned}(\mathbf{CCP}) \quad \min \quad & C(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z_{ijk}^s - \tilde{d}_{s(j)} \geq 0,\ \forall j\in\mathcal{D}\right) \geq \alpha_s, \\ & (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\in\boldsymbol{\chi}.\end{aligned} \tag{14}$$

Solving problem (14) directly is challenging, because even with an exact probability distribution $\mathbb{P}$ and a given $\boldsymbol{z}\in\boldsymbol{\chi}$, computing the probability requires multi-dimensional integration, let alone the distribution is unknown (as discussed in Section 2.3). To efficiently solve this joint chance-constrained programming problem with limited information on the distribution, and without relying on Gaussian-like assumptions, we establish an SAA-based mixed-integer programming reformulation for (14) in the next section based on scenario simulations of demand surges, following the approaches of Ruszczyński (2002) and Benati & Rizzi (2007).

3.5. Scenario-based MIP Reformulation

Considering a realized scenario set $R = \{\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_L\}$, where $L = |R|$ represents the number of realized scenarios of demand surges. Typically, the value of L is small, leading to an unexpected approximation error (*e.g.*, $L = 11$ among a total of over 200 scenario points of a customer in the Wal-Mart dataset). Hence, we first fit the parameters of distribution (4) as the empirical distribution $\hat{\mathbb{P}}_{\boldsymbol{\psi}}$ using the demand set R and then generate a scenario set $S = \{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_M\}$ of $\tilde{\boldsymbol{d}}_s$ via *Latin hypercube sampling*, where $M = |S|$ is the number of sampled scenarios. Then we generate $\bar{S} = S\cup R = \{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_M,\boldsymbol{\xi}_{M+1},\ldots,\boldsymbol{\xi}_{M+L}\}$ where we let $\boldsymbol{\xi}_{M+q} = \boldsymbol{\psi}_q$ for any $1\leq q\leq L$, and use $\bar{M} = M + L$ to denote the set size. The corresponding probability of the scenarios are $\boldsymbol{p} = (p_1,\ldots,p_{\bar{M}})^T$ with $\sum_{j=1}^{\bar{M}} p_j = 1$. Notice that $\boldsymbol{\xi}_q$ is a $|\mathcal{D}|$-dimensional vector and we denote the demand of customer $j\in\mathcal{D}$ in scenario $\boldsymbol{\xi}_q$ by $\xi_{q(j)}$.

As we use the finite scenario set to simulate the stochastic surged demand, only partial scenarios need to be satisfied to touch the SL constraint. Define a binary vector $\boldsymbol{u}\equiv(u_1,u_2,\cdots,u_{\bar{M}})^{\mathrm{T}}$, our chance constraint

in problem (14) can be reformulated as follows:

$$\sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^s_{ijk} - u_q \cdot \xi_{q(j)} \geq 0,\ \forall j \in \mathcal{D}, \forall q \in [\bar{M}], \tag{15}$$

$$\sum_{q=1}^{\bar{M}} p_q \cdot (1-u_q) \leq 1-\alpha_s, \tag{16}$$

where the binary variable $u_q = 1$ when scenario $\boldsymbol{\xi}_q$ is selected to be satisfied, and otherwise, $u_q = 0$. Let $K_\alpha \equiv \lfloor N_\alpha \rfloor$ be the minimum number of scenarios to be satisfied under targeted SL α_s.

$$\begin{aligned}
(\textbf{SAA-MIP}) \quad \min \quad & C(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \\
\text{s.t.} \quad & \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^s_{ijk} - u_q \cdot \xi_{q(j)} \geq 0, \quad \forall j \in \mathcal{D}, \forall q \in [\bar{M}], \\
& \sum_{q\in[\bar{M}]} p_q \cdot (1-u_q) \leq 1-\alpha_s, \\
& (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) \in \boldsymbol{\chi},\ \boldsymbol{u} \in \{0,1\}^{\bar{M}}.
\end{aligned} \tag{17}$$

which is also known as the *sample average approximation* (SAA) model, and $\boldsymbol{u}$ is the binary auxiliary decision variable. In problem (17), the chance constraint is handled as a *knapsack sub-problem* in a linear form, and then the reformulated version of the model can be solved straightly.

Then we show some convergence properties of the SAA model. Let v^*_α and v^*_R represent the optimal value of CCP model (14) with target service level α and SAA model (17) with realization set R, respectively.

Theorem 1 (Convergence of the SAA model)**.** Solving the SAA model (17) with the desired service level target α under realizations set R is asymptotically optimal, approaching the optimal solution almost surely as the sample size $|R|$ increases, i.e.,

$$\lim_{|R|\to\infty} \mathbb{P}\left(v^*_\alpha = v^*_R\right) \to 1.$$

Theorem 1 demonstrates the qualitative behavior of the sample approximation method, specifically stating that as the sample size of realizations approaches infinity, solving the SAA model will yield the exact optimal solution with high probability. Furthermore, in the limiting case where $|R| \to \infty$, additional sampling from the empirical distribution becomes unnecessary, as the empirical distribution over the set R sufficiently approximates the true distribution. In this asymptotic state, further sampling based on the empirical distribution does not affect the accuracy of the SAA model, since the sample set S can be regarded as a realization of the true distribution, consistent with the law of large numbers.

Although the SAA model will provide a high-quality approximate solution and is straightly solvable, the required scenario size is still not negligible for large-scale problems, especially those involving multiple customers. Solving such a (large-scale) problem with a knapsack sub-problem remains challenging. This motivates us to propose two effective algorithms in the next section to improve the solution procedure further.

4. Solution Methodologies

In this section, we design two algorithms building on the cell-and-bound framework (Zheng et al., 2017). First, we give the MIP reformulation for our transformed model (17) based on the hyperplane arrangement in Section 4.1. In Section 4.2, to improve the efficiency of the algorithm, we propose a *greedy pricing and weighting strategy* to speed up cell enumerations; we call this method the greedy pricing and weighting strategy based cell-and-bound (G-C&B) algorithm. In Section 4.3, to further improve the efficiency of G-C&B when treating large-scale problems with a large sample size, we develop a local experimentation for global optimization (LEGO) framework, dubbed LEGO-C&B.

4.1. Cell-and-Bound Framework and Hyperplane Arrangement based MIP Reformulation

The cell-and-bound method is introduced by Zheng et al. (2017) for solving math programming problems with binary variables; also see Sleumer (1999). Under the cell-and-bound framework, each inequality (corresponding binary variable) is mapped into a hyperplane, so that the value selection for binary variables is equivalent to picking a cell in the positive or negative side of a hyperplane. For ease of expression, we use π to denote a cell. In such a framework, the cell-and-bound algorithm continuously enumerates across all neighboring cells, computes the solution for each cell, and generates the optimal solution when identifying the optimal cell. See Figure 5 for a schematic presentation.

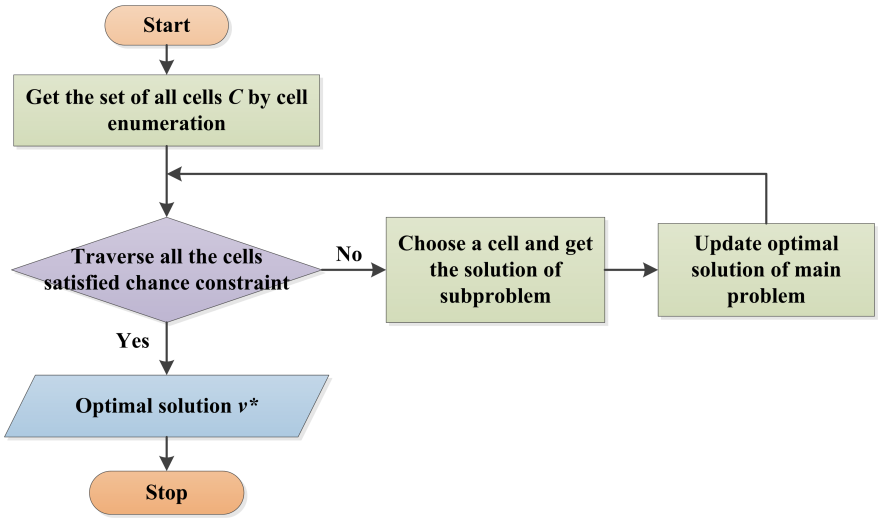


Figure 5: The schematic presentation of the cell-and-bound method.

According to the knapsack sub-problem (15)-(16), each customer demand ξ_q^j can be represented as a hyperplane, and a scenario vector $\boldsymbol{\xi}_q$ corresponds to a hyperplane class. The hyperplane class of each demand scenario $\boldsymbol{\xi}_q$ and the generated positive polyhedron in the joint chance-constrained version are generated as follows.

Definition 1 (Hyperplane Class for Demand Scenario and its Positive Polyhedron)**.** Given a demand scenario $\boldsymbol{\xi}_q \in \mathbb{R}^{|\mathcal{D}|}$, its hyperplane class including hyperplanes for each customer demand is defined as

$$\boldsymbol{H}_q = \left\{ H_{qj} := \left\{ \boldsymbol{z}^s \in \mathbb{R}^+ \middle| \sum_{i\in\mathcal{F}} \sum_{k\in\mathcal{V}_{ij}} z^s_{ijk} - \xi^j_q = 0 \right\}, \forall j \in \mathcal{R} \right\}, \tag{18}$$

and the corresponding positive polyhedron $P(\boldsymbol{\xi}_q)$ is defined as the intersection of the positive halfspace H^+_{qj} of each hyperplane $H_{qj} \in \boldsymbol{H}_q$, i.e.,

$$P\left(\boldsymbol{\xi}_q\right) := \bigcap_{j\in\mathcal{R}} H^+_{qj}, \tag{19}$$

here we use $\varphi_{lq} = 1$ to represent $\pi_l \subseteq P(\boldsymbol{\xi}_q)$, and otherwise $\varphi_{lq} = 0$.

In model (17), the scenario selection is represented as binary variables ($\boldsymbol{u}$) each of which is mapped into its corresponding hyperplane class. Hence, we reformulate (17) as the following equivalent hyperplane arrangement (HA) version.

$$\begin{aligned}
(\textbf{HA-MIP}) \quad \min \quad & \left\{ v(\pi_l) \mid \pi_l \in \Phi,\ \boldsymbol{p}^T \cdot \boldsymbol{\varphi}_l \geq \alpha^s \right\} \\
\text{where } v(\pi_l) := \min \quad & C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \\
\text{s.t.} \quad & \sum_{i\in\mathcal{F}} \sum_{k\in\mathcal{V}_{ij}} z^s_{ijk} - u_q \cdot \xi^j_q \geq 0, \qquad \forall j \in \mathcal{D}, \forall q \in [\bar{M}] \\
& (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \boldsymbol{\chi},\ \boldsymbol{u} = \boldsymbol{\varphi}_l \in \{0,1\}^{\bar{M}}
\end{aligned} \tag{20}$$

here $v(\cdot)$ is the optimal objective value of the problem. Also, Φ represents the set of all cells and the optimal solution of (20) is iteratively obtained by exploring sub-problems among all feasible cells.

Proposition 1 (Complexity Bound for C&B framework)**.** The maximum number of cells to be explored under the C&B framework is bounded by $O(|\overline{S}|^{|\mathcal{D}|})$, where $|\overline{S}|$ is the number of demand scenarios and $|\mathcal{D}|$ is the number of customers.

According to Proposition 1, the optimal solution can be obtained by exploring no more than $O(|\overline{S}|^{|\mathcal{D}|})$ cells under the C&B framework. This is a significant performance improvement compared to the B&B algorithm which has a worst-case complexity of $O(2^{|\overline{S}|})$. In practice, oftentimes $|\overline{S}|$ may be much larger than $|\mathcal{D}|$, which guarantees that C&B requires searching a much smaller number of cells than the B&B algorithm. However, we remark that every sub-problem is an NP-hard network design problem. As the sample size or network size increases, solving sub-problems on these cells still poses challenges. In order to improve the solution efficiency for (20), we next propose the G-C&B method.

4.2. Greedy Pricing and Weighting Strategy based Cell-and-Bound Algorithm

The cell-and-bound method intends to search all feasible cells and continuously update the upper or lower bounds of the problem by calculating the optimal value in each cell. The enumeration of the neighboring cells gives the interior points of all neighboring cells around a given cell at the same time (Zheng et al., 2017). Hence, this is equivalent to a tree search of all feasible cells as depicted in Figure B2.

It should be noted that the C&B tree does not necessarily have a binary structure because a cell may have more than two neighboring cells, for example, the tree generated by choosing (+,+,+,+) as the root cell in Sleumer (1999)'s example is a multi-branch tree.

To solve problem (20), we consistently set the all-negative cells (i.e. those cells satisfying $\pi_l \cap \{\cup_q P(\boldsymbol{\xi}_q)\} = \phi$) as the root cells. Then we know that each layer in the obtained search tree has the same number of negative signs '-', meaning that these cells correspond to the same SL (i.e., satisfying the same number of scenarios). It is apparent that the optimal total cost of a cell having a greater depth in the search tree will not be superior to that having a smaller depth, because it needs to satisfy more scenarios. Following this rule, we can remove some cells based on the upper and lower bounds in the C&B framework.

Throughout the solution process, finding a more optimal cell can help immediately remove those branches associated with less optimal cells. To help reduce the running time of the C&B algorithm, we next propose a greedy pricing and weighting strategy using the depths of cells. The main idea of G-C&B is to (i) compute weights for all scenarios based on their demand information (these weights are also referred to as the *price* of scenarios), (ii) rank all scenarios, and (iii) prioritizing the enumeration and exploration of scenarios corresponding to lower prices. In this way, we can expedite the updating process of the problem's bounds, thus eliminating unnecessary cells at early stages to avoid excessive computational complexity otherwise spent on solving redundant sub-problems.

In details, we calculate the price of each scenario through following *pricing sub-problem*,

$$
\begin{aligned}
(\textbf{Pricing Problem}) \quad \Delta c_e = \min \quad & \frac{C(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}'|\overline{\boldsymbol{d}}) - C(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\overline{\boldsymbol{d}})}{\Delta d} \\
\text{s.t.} \quad & \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^{s}_{ijk} - \overline{d}_j \geq 0, && \forall j \in \mathcal{D}, \\
& \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^{'s}_{ijk} - \overline{d}_j \geq 0, && \forall j \in \mathcal{D}\setminus\{e\}, \\
& \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^{'s}_{ijk} - \left(\overline{d}_j + \Delta d\right) \geq 0, && \forall j \in \{e\}, \\
& (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}), (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}') \in \boldsymbol{\chi}.
\end{aligned}
\tag{21}
$$

where Δc_e is the unit price of demand change in customer $e \in \mathcal{D}$, $\overline{\boldsymbol{d}} \in \mathbb{R}_+^{|\mathcal{D}|}$ is a given demand vector (typically, we can set the value as $\overline{d}_j = \boldsymbol{e}^{\mu_a}$ for all $j = 1, \ldots, |\mathcal{D}|$), and $\Delta d \geq 0$ is a positive value. Hence the price of a demand scenario $\boldsymbol{\xi}_q$ to any given demand vector $\hat{\boldsymbol{d}}$ is calculated as

$$
w_q = \sum_{j\in\mathcal{D}} \Delta c_j \cdot \left(\boldsymbol{\xi}_q^j - \hat{d}_j\right). \tag{22}
$$

Then we set the initial value $\hat{\boldsymbol{d}}_0 = \mathbf{0}$ and iteratively choose the scenario $\boldsymbol{\xi}_q$ with the smallest price w_q^t to the benchmark demand $\hat{\boldsymbol{d}}_t$ to satisfy (i.e., let $u_q = 1$) in iteration t until touching the SL constraint. We refer to this heuristic algorithm as the Greedy Pricing and Weighting Strategy based Cell-and-Bound (G-C&B) algorithm, and we provide a formal description of G-C&B in Algorithm 1. Additionally, we present an example and the corresponding decision tree in Figure B2 to illustrate the efficiency of G-C&B.

Proposition 2 (Complexity Bound for G-C&B)**.** The maximum number of sub-problems to be solved in G-C&B algorithm is bounded by $O(|\overline{S}|)$.

Algorithm 1: G-C&B algorithm

Input: 4PLN structure $\mathcal{N}$ and $\mathcal{V}$; SL α; all model parameter in Table 1; Demand scenarios set $\overline{S}$.

Output: Optimal 4PLN solutions $(\boldsymbol{x}, \boldsymbol{y})$.

Set all-negative cells as the root cells.

Initialize *lowerBound* $= \infty$, depth $t = 1$, $\hat{\boldsymbol{d}}_0 = \mathbf{0}$, and scenarios set $S' = \phi$.

while $t \leq K_\alpha$ **do**

  Calculate w_q^t for each demand scenario in $\overline{S}$, rank scenarios in descending order of weights, and generate their associated hyperplane classes.

  Solve the sub-problem of (20) in the first cell and obtain the solution value v_t for current cell.

  Update the *lowerBound*;

  Enumerate the neighbouring cells of the current cell;

  Update detph $t = t + 1$;

end

Set the optimal solution value v_{best} as the updated *lowerBound*.

Output the optimal decision variables $(\boldsymbol{x}, \boldsymbol{y})$ corresponding to v_{best}.

In addition, G-C&B needs to search a fewer number of cells than the base C&B algorithm (as illustrated in Figure B2). On the other hand, in each cell, the corresponding sub-problem is NP-hard with the computational complexity of $O(2^{|\mathcal{V}|})$, where $|\mathcal{V}|$ is the number of 3PL TPs (i.e., a metric of the scale of the problem). In order to further enhance the solution efficiency as the scenario size $|\overline{S}|$ is large, we next develop the LEGO framework and the corresponding LEGO-C&B method.

4.3. Local Experimentation for Global Optimization Framework and LEGO-C&B Algorithm

Although G-C&B is able to achieve a significant performance improvement over C&B and B&B, the total number of cells that G-C&B needs to search through may still be quite big when (i) the network scale or (ii) the size of the demand scenario is large. In order to achieve additional reduction for the computational complexity, we propose a two-stage learning-based optimization framework to further refine G-C&B. Our main idea is to explore the cell space using a small subset of samples, and then exploit the gained knowledge to expedite the optimization of the full problem with complete samples (to avoid extensive cell enumeration). See Figure 7 for a schematic representation of this idea.

Our two-stage approach, called *local experimentation for global optimization (LEGO)*, takes two stages:

(i). Local experimentation (exploration).

Step 1: Given the historical scenario set R, learn the parameters of demand distributions, and generate sampled scenario set S through LHS, and obtain $\overline{S} = R \cup S$;

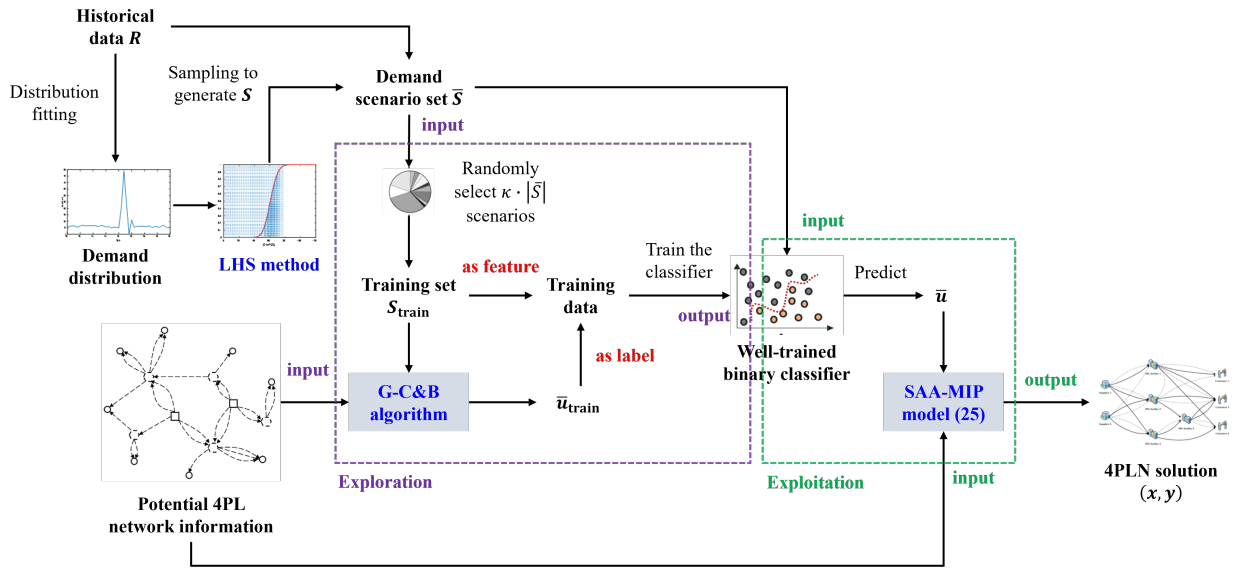


Figure 6: Two-stage local experimentation for global optimization (LEGO) framework.

Step 2: Given the training ratio $\kappa \in (0, 1)$, randomly select $\kappa \cdot |\overline{S}|$ scenarios from set $\overline{S}$ to generate training set S_{train};

Step 3: Recall G-C&B to solve the network design problem under set S_{train} in form of (20), and obtain the output $\boldsymbol{u}_{\text{train}}$;

Step 4: Generate the training data set Θ_{train} by letting each demand vector $\boldsymbol{\xi}_q \in S_{\text{train}}$ as the features and the corresponding u_q as the label;

Step 5: Train the 0-1 classification model $\widehat{f}$ on Θ_{train}, i.e., for any given demand scenario $\boldsymbol{\xi}_i$ we have $u_i = \widehat{f}(\boldsymbol{\xi}_i)$.

(ii). Global optimization (exploitation).

Step 6: Utilize the well-trained model $\widehat{f}$ to predict the decision $\overline{\boldsymbol{u}}$ for problem (17) over the set $\overline{S}$, i.e., $\overline{\boldsymbol{u}} = \widehat{f}(\overline{S})$. We denote the set of scenarios with $u_q = 1$ as S_{learn}. If $|S_{\text{learn}}| \equiv \|\overline{\boldsymbol{u}}\| < K_\alpha$, we generate the demand vector $\hat{\boldsymbol{d}} := \{d_j | d_j = \max_q\{\xi_q^j\},\ \forall j \in \mathcal{D}\}$. We then calculate the price of each scenario in $\overline{S} \setminus S_{\text{learn}}$ through (21) and select the top $K_\alpha - \|\overline{\boldsymbol{u}}\|$ scenarios ξ_q in ascending order of price, setting the corresponding $\overline{u}_q = 1$.

Step 7: Solve following SAA-MIP model with fixed $\overline{\boldsymbol{u}}$ to obtain the 4PLN solution $(\boldsymbol{x}, \boldsymbol{y})$

$$\begin{aligned} \min \quad & \hat{C}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \overline{\boldsymbol{u}}) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{F}} \sum_{k \in \mathcal{V}_{ij}} z_{ijk}^s - \overline{u}_q \cdot \xi_q^j \geq 0, \quad \forall j \in \mathcal{D}, \forall q \in [\bar{M}], \\ & (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \boldsymbol{\chi}. \end{aligned} \tag{23}$$

Using the above LEGO framework, we can learn knowledge about the satisfaction of demand scenarios to meet a given service level (SL) on a subset of samples. This allows us to predict the values of decision variables (i.e., providing a solution to the knapsack sub-problem) in model (20), thereby reducing computational complexity. Driven by the above-mentioned idea, we propose a further refined version of G-C&B, called two-stage LEGO-based cell-and-bound (LEGO-C&B). The pseudo-code of LEGO-C&B is described in Algorithm 2.

Algorithm 2: LEGO-C&B algorithm

Input: 4PLN structure $\mathcal{N}$ and $\mathcal{V}$; SL α; All the model parameter shown in Table 1; Set of demand scenarios $\overline{S}$ obtained by sampling; Training ratio κ.

Output: An 4PLN design solution $(\boldsymbol{x}, \boldsymbol{y})$.

Select $\kappa \cdot |\overline{S}|$ demand scenarios randomly to construct the training set S_{train};
Solve HA-MIP model (20) under S_{train} by using G-C&B algorithm, and obtain $\overline{\boldsymbol{u}}_{\text{train}}$;
Train binary classifier $\widehat{f}_B$ through input S_{train} as the features and $\overline{\boldsymbol{u}}_{\text{train}}$ as the label;
Predict $\overline{\boldsymbol{u}} = \widehat{f}(\overline{S})$ over full scenario set $\overline{S}$, and generate satisfied scenatio set S_{learn};
if $\|\overline{\boldsymbol{u}}\| \geq K_\alpha$ **then**
    Randomly reduce $\|\overline{\boldsymbol{u}}\| - K_\alpha$ scenarios from set S_{learn};
else
    Calculate the price of each scenario in $\overline{S} \setminus S_{\text{learn}}$ through (21) and select the top $K_\alpha - \|\overline{\boldsymbol{u}}\|$ scenarios ξ_q in ascending order of price, setting the corresponding $\overline{u}_q = 1$;
end
Solve problem (23) with fixed $\overline{\boldsymbol{u}}$;
Output decision $(\boldsymbol{x}, \boldsymbol{y})$.

In LEGO-C&B, we can reduce the number of cells to be explored using the results generated from partial demand scenario samples. In the first stage, the computational complexity bound of the training problem is $O(\kappa \cdot |\overline{S}|)$, i.e., the maximum number of sub-problems to be solved. It is apparent that, the training ratio κ plays a role in the efficiency of the algorithm; the choice of κ will be a balancing factor for the training accuracy and solution efficiency. In the second stage, the prediction of scenarios reduces the computational complexity, requiring the solution of only a 4PLN problem, i.e., with the complexity of $O(1)$. Consequently, we conclude that the overall number of cells to be explored in the LEGO-C&B algorithm is $O(\kappa \cdot |\overline{S}|)$, which is smaller than that under G-C&B. algorithm. As we will show soon in our numerical experiments, LEGO-C&B is able to significantly reduce the computational complexity for large-sample problems by properly adjusting the value of the training ratio.

5. Numerical Studies

In this section, we evaluate the performance of the proposed model and algorithms. We do so using a sequence of scale-specific 4PLN examples. To substantiate the effectiveness of our proposed algorithms, we benchmark LEGO-C&B with G-C&B (Section 5.2). Further, we investigate the performance of the CCP model (Section 5.3) and the efficiency of SAA approximation (Section 5.4). We also investigate the impact of the demand volume, surge intensity, demand surge frequency, and 3PL resources rental price on the system performance (Section 5.5) and the value of 4PL (Section 5.6). In these experiments, all algorithms are implemented in MATLAB with an Intel(R) Core (TIM) i5-7500 CPU @ 3.40GHz PC.

5.1. Experiment Settings

Motivated by China's Cainiao supply chain network[6], we consider a data set that includes 8 different instances specified by the scale and geographical distribution. The details of the network structure are given in Table 2, where Instances 1-4 can be used to simulate 4PLNs at provincial and municipal levels, while Instances 5-6 can represent a bigger region such as East China (including eight provinces) and North China (including five provinces), and instance 7-8 can emulate a nationwide network. The location of suppliers, DCs, and customers are uniformly distributed over $[0, 100] \times [0, 100]$.

Table 2: 4PLN instances.

	# Supplier	# DC	# Customer	# 3PL TP
Instance 1	3	6	5	3
Instance 2	3	6	5	5
Instance 3	8	12	10	3
Instance 4	8	12	10	5
Instance 5	15	24	18	3
Instance 6	15	24	18	5
Instance 7	20	38	56	3
Instance 8	20	38	56	5

Figure 7 shows an instance with 15 suppliers, 24 DCs, and 18 customers. We assume that the transport cost is proportional to the Euclidean distance in the plane. When generating the costs and capacities of 3PL facilities and 3PL transportation carriers, we assume that $c_i^l \sim N(100000, 10000)$, $c_i^p \sim N(0.5, 0.05)$, $q_i \sim N(500, 65)$, $c_{ijk}^f \sim N(25000, 2000)$, $c_{ijk}^t \sim N(0.1, 0.02) \times distance$, and $h_{ijk} \sim N(100, 15)$. Unlike a supply chain network, the complexity of a 4PLN is determined not only by the number of nodes but also by the number of edges (i.e. 3PL transportation carriers). A 4PLN can be seen as a transformed network with the 3PL transportation carriers regarded as additional layers. Thus, the complexity of Instances 7 and 8 is nearly equivalent to that of a 1000-node network, which is challenging to solve in practice due to the substantial number of decision variables and constraints (typically exceeding millions) involved.

[6]The details are provided on its official website: https://www.cainiao.com/land.html

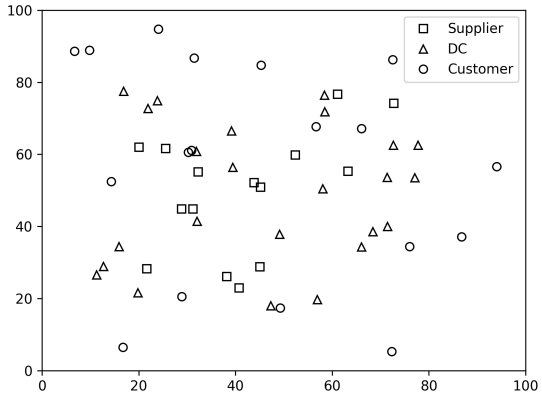


Figure 7: Supplier-DC-customer locations.

To generate the demand data, we first fit the real-world Walmart data[5] to our surge demand model, which gives

$$\tilde{d}_r \sim N\left(6572, 150^2\right), A \sim N\left(8.723, 0.182^2\right).$$

Next, we use Latin hypercube sampling to generate demand data according to the above model for the trace simulation. See Figure 2 for a comparison of the simulated data and the real data, which verifies the effectiveness of our demand model.

5.2. Performance Analysis

To evaluate the performance of our proposed algorithms, we conduct numerical experiments for LEGO-C&B, G-C&B, and C&B, branch-and-bound (B&B), and the state-of-the-art solver Gurobi. Furthermore, we also investigate how the sample size and network scale impact the performance of proposed algorithms. We also investigate the effect of the training ratio on the LEGO-C&B algorithm. Each numerical experiment was independently repeated 30 times, and the average results were reported.

5.2.1. Algorithm Accuracy and Effectiveness

We first test the effectiveness of LEGO-C&B and G-C&B. The settings of numerical experiments are as follows: the number of periods is $N = 50$, the surge frequency is $\varphi = 0.1$, and the desired service-level targets in the regular demand period and surge demand period are $\alpha_r = 0.95$ and $\alpha_s = 0.9$. And we set the training ratio $\kappa = 0.2$ for LEGO-C&B. In all experiments, the limitation of the total algorithm running time is set as 7200 seconds (2 hours). For LEGO-C&B, the optimal classification learner parameters and the best classifier are obtained during learner data training in which a series of classification learning methods (i.e. Logistic Regression, KNN, and SVM et. al) are tested.

We evaluate the accuracy and efficiency of LEGO-C&B, G-C&B, C&B, B&B, and Gurobi. In Table 3, we report their optimality gaps (%) and running times (sec). Here, we use *Gap** to denote the optimality gap reported by Gurobi directly, and let *Gap* represent the solution gap relative to Gurobi for other algorithms.

Table 3: Comparison of LEGO-C&B, G-C&B, C&B, B&B, and Gurobi.

Instance	# Scenario	LEGO-C&B		G-C&B		C&B		B&B		Gurobi	
		Gap	CPU Time	Gap	CPU Time	Gap	CPU Time	Gap	CPU Time	Gap*	CPU Time
Instance 1	100	0.22	0.58	0.01	1.18	0.00	14.79	0.00	6.23	0.00	1.95
	500	0.08	0.58	0.01	5.13	0.00	226.79	0.00	1584.56	0.01	444.18
	1000	0.10	0.73	0.01	18.39	0.00	1428.95	0.05	7200*	0.01	2417.86
	2000	0.49	1.35	0.02	58.09	0.00	2668.50	0.10	7200*	0.04	7200*
Instance 2	100	0.35	0.39	0.10	2.08	0.00	29.71	0.00	3.11	0.01	1.62
	500	0.07	0.85	0.02	6.65	0.00	507.96	0.00	4739.93	0.01	809.21
	1000	0.30	1.07	0.03	23.40	0.00	2501.63	0.03	7200*	0.01	1736.36
	2000	0.48	1.29	0.03	290.82	0.00	3565.90	0.04	7200*	0.03	7200*
Instance 3	100	0.23	2.64	0.01	17.56	0.00	124.28	0.00	32.92	0.00	9.84
	500	0.17	6.76	0.01	49.00	0.00	1334.53	0.01	7200*	0.01	154.88
	1000	0.13	5.98	0.02	57.85	0.00	3314.38	0.01	7200*	0.01	799.45
	2000	0.36	6.58	0.02	176.19	0.62	7200*	0.01	7200*	0.01	2251.39
Instance 4	100	0.14	4.45	0.02	71.68	0.00	510.86	0.00	1416.29	0.00	180.47
	500	0.24	7.44	0.03	82.44	0.00	2197.24	0.01	7200*	0.01	3488.23
	1000	0.17	13.47	0.01	126.15	0.00	3967.66	0.01	7200*	0.01	7200*
	2000	0.22	8.72	0.01	313.02	0.94	7200*	0.01	7200*	0.01	7200*
Instance 5	100	0.30	34.97	0.00	49.68	0.00	98.67	0.00	139.40	0.00	93.37
	500	0.24	55.16	0.01	320.70	0.12	7200*	0.00	7200*	0.01	163.35
	1000	0.37	76.56	0.01	379.76	0.03	7200*	0.04	7200*	0.01	310.37
	2000	0.27	90.14	0.01	708.71	0.01	7200*	0.01	7200*	0.01	1447.45
Instance 6	100	0.23	115.26	0.00	458.35	0.00	967.23	0.00	1860.92	0.00	404.11
	500	0.12	227.27	0.01	993.01	0.02	7200*	0.02	7200*	0.01	7200*
	1000	1.70	506.57	0.01	1397.36	0.01	7200*	0.01	7200*	0.01	7200*
	2000	0.54	139.72	-0.52	3195.62	0.05	7200*	0.25	7200*	1.74	7200*
Instance 7	100	0.37	972.34	0.25	7200*	2.24	7200*	3.29	7200*	0.00	7200*
	500	0.78	1089.01	0.74	7200*	4.62	7200*	4.99	7200*	0.01	7200*
	1000	0.53	1050.34	0.02	7200*	4.87	7200*	8.07	7200*	3.73	7200*
	2000	-4.55	854.31	-5.02	7200*	2.03	7200*	1.21	7200*	12.46	7200*
Instance 8	100	0.09	3214.97	0.00	7200*	3.32	7200*	3.78	7200*	5.10	7200*
	500	-1.60	4544.66	-1.26	7200*	0.89	7200*	0.79	7200*	4.45	7200*
	1000	-1.06	4991.83	-1.94	7200*	6.02	7200*	6.90	7200*	10.47	7200*
	2000	-6.62	4129.88	-0.99	7200*	5.23	7200*	6.40	7200*	8.68	7200*

Firstly, the results indicate that C&B outperforms B&B in all cases; therefore, we exclusively compare our algorithms to C&B. Table 3 demonstrates that G-C&B runs faster than C&B, especially when the scale is large while maintaining a high solution accuracy. G-C&B has shown to be more efficient in solving small-sample problems, and as the sample size increases, LEGO-C&B becomes significantly advantageous. When compared to the state-of-the-art solver Gurobi, G-C&B exhibits faster speeds with a negligible error (less than 0.1%) in most experiments. LEGO-C&B, while maintaining an acceptable error, demonstrates even faster speeds, usually nearly a hundred times faster than Gurobi. It is noteworthy that in large-scale problems, both LEGO-C&B and G-C&B may achieve better solutions than Gurobi within the specified time. In summary, G-C&B outperforms C&B in all cases, and LEGO-C&B, as an enhanced version of G-C&B for large-sample problems, exhibits a significant improvement in solution efficiency.

We also report the iterative output of G-C&B for Instance 1 with a total of 500 scenarios as an example to

show how the algorithm converges to the near-optimal solution iteratively. Figure 8 illustrates the iterative output over a total of 450 iterations (as we terminate the algorithm after $\alpha_s \cdot |\overline{S}|$ iterations), with the dotted line representing the results obtained by directly solving the SAA-MIP model using Gurobi as a benchmark.

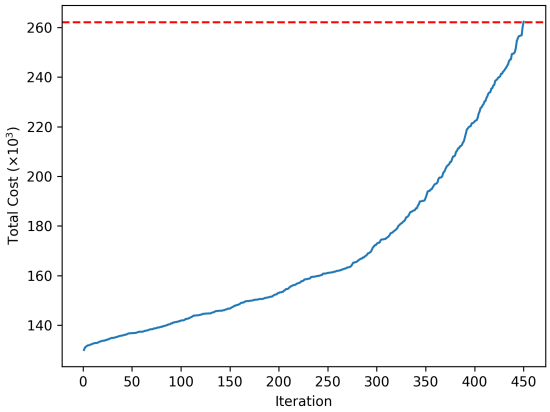


Figure 8: Iterative output of G-C&B.

5.2.2. On the Scenario Size

To investigate the effect of the sample size on the performance of LEGO-C&B and G-C&B, we report their running times under different sample sizes on the Instance 1 setting.

Table 4: Running times under different scenario sizes.

# Scenario	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
LEGO-C&B ($\tau = 0.1$)	0.231	0.467	0.523	0.805	0.962	1.093	1.096	1.743	1.948	2.022
LEGO-C&B ($\tau = 0.2$)	0.328	0.602	0.908	0.976	1.212	1.611	1.867	1.896	1.963	2.800
LEGO-C&B ($\tau = 0.3$)	0.446	0.636	1.109	1.207	1.248	1.669	2.016	2.862	2.896	3.998
G-C&B	1.495	4.703	10.367	22.531	32.461	45.426	61.969	79.955	106.686	154.159

Table 4 indicates that the computation time for both LEGO-C&B and G-C&B algorithms increases with the growth of scenario size. However, LEGO maintains a relatively low rate of increase, where an increase in the training ratio leads to time increases, providing satisfactory computational speed even for very large scenario sizes. Hence, it is evident that our LEGO-C&B is the most advantageous algorithm when dealing with large scenario (sampling) size problems.

5.2.3. On the Scale of the Network

To investigate the performance of algorithms under different network scales, we consider a series of network models having an # Supplier $= n$, # DC $= 2n$ and # Customer $= n$ structure (where # TP $= 3$ is a constant), in which all nodes between different echelons are fully connected. We report the running times of LEGO-C&B and G-C&B algorithms in Table 5 (under 1000 scenarios setting).

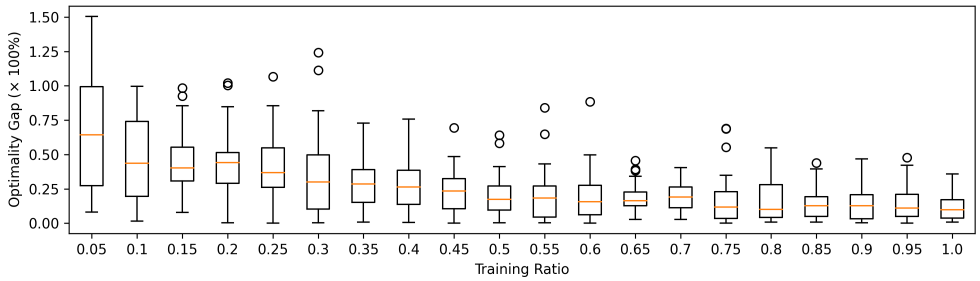
Table 5: Running times under different network scales.

n	1	3	5	7	9	11	13	15
LEGO-C&B ($\tau = 0.1$)	0.017	0.069	0.216	0.504	0.790	1.994	3.502	5.272
LEGO-C&B ($\tau = 0.2$)	0.018	0.071	0.240	0.609	0.873	2.177	3.767	5.485
LEGO-C&B ($\tau = 0.3$)	0.019	0.083	0.263	0.703	0.957	2.297	4.065	5.628
G-C&B	0.038	0.173	0.657	2.002	2.602	5.077	9.425	12.680

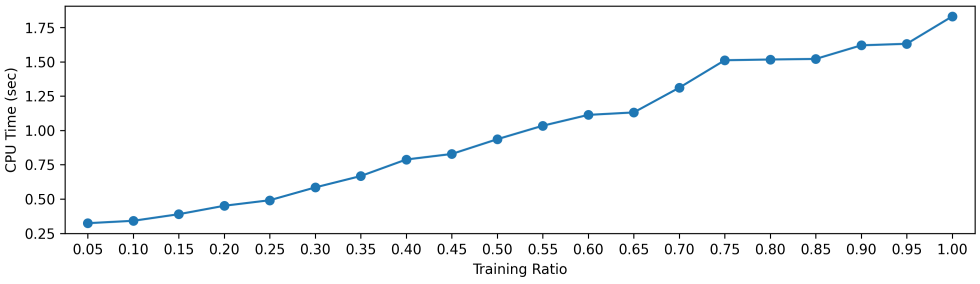
First, G-C&B is the most computationally expensive algorithm of which the running time grows explosively as n increases. Next, between G-C&B and LEGO-C&B, the latter is more stable in n and has a smaller growth rate, and the increase in the training ratio has a negligible impact on the computation time. Hence, we conclude that LEGO-C&B exhibits the highest efficiency when dealing with large-scale problems.

5.2.4. On the Training Ratio of LEGO-C&B

In LEGO-C&B, we investigate the impact of how many scenarios are selected for training on its performance. We conduct a set of numerical experiments to report the optimality gap (let the output of G-C&B as a benchmark) and running time under different training ratios (ratio of the size of the training set to the total number of scenarios). In all cases, LEGO-C&B is tested 30 times under every training ratio as the setting of instance 1, and the results are shown in Figure 9.



(a) Objective Gap



(b) Running Time

Figure 9: Performance of LEGO-C&B under different training ratios.

According to Figure 9, as the training ratio κ increases, the gap between the LEGO-C&B solution and the G-C&B solution gradually narrows and diminishes. On the other hand, the running time of the algorithm

increases due to the larger training set size for the exploration step. In fact, when the training ratio reaches a certain level, further increasing κ does not guarantee a notable improvement in the accuracy of the algorithm, and it comes at the expense of sacrificing algorithm efficiency. Therefore, a proper training ratio should be identified to strike a balance between solution accuracy and algorithm efficiency.

5.3. Analysis on the Chance-Constrained Model

To investigate the decision-making performance of the chance-constrained programming model (refer to CCP model), we compared its performance with the commonly used expected value (EV) model and worst-case model, including two versions: the worst scenario (WS) model and the worst marginal demand (WMD) model in stochastic programming (we provide the specific models in Appendix C). Given the realized scenario set, we solved the different models, and the output of the CCP model is used as a benchmark (in this experiment we let $\overline{S} = R$). We report the SLs and the cost difference of the aforementioned models compared to the CCP model (defined as $\frac{Cost_{(\cdot)} - Cost_{CCP}}{Cost_{CCP}} \times 100\%$) in Table 6.

Table 6: Comparison result of models for demand surge management (SL target is set as $\alpha_s = 0.9$).

# Realized Scenario	CCP model	EV model		WS model		WMD model	
	SL	SL	Cost (%)	SL	Cost (%)	SL	Cost (%)
50	0.932	0.358	-8.91	0.691	-4.37	0.985	4.56
60	0.929	0.358	-8.92	0.732	-3.35	0.989	5.78
70	0.959	0.358	-9.78	0.609	-5.44	0.997	5.96
80	0.932	0.358	-8.91	0.652	-4.24	0.996	6.99
90	0.959	0.358	-9.78	0.693	-4.69	0.999	6.78
100	0.959	0.358	-9.78	0.652	-4.93	1.000	7.07

From Table 6, it can be observed that, compared to the overly optimistic EV model and WS model, which fail to meet the service level constraints, the CCP model can obtain effective network design solutions without being overly conservative like the WMD model, which would result in excessively high total costs.

5.4. Impact of Scenario Size on the Performance of SAA

To investigate the impact of scenario size on the performance of SAA, we initially examine its convergence behavior. We first conduct a goodness-of-fit test for sampled scenarios (realizations) of varying sizes to assess the fit to the underlying distribution (specific details are provided in Appendix D). To numerically show the impact of scenario size on the optimization error, we create finite discrete support set S_{real} comprising a total of 1000 scenarios to represent the given empirical distribution $\hat{\mathbb{P}}_{\boldsymbol{\psi}}$, then randomly generate scenarios from S_{real} to construct the sampled scenario set S. Subsequently, we solve the network design problem under S to obtain a network solution and evaluate its SL under S_{real}. Figure 10 reports the SL performance under varying sampling numbers.

Figure 10 demonstrates that with an increasing number of samples, the SL of the network gradually approaches the preset level (set at 0.9 in this experiment). This result indicates that a larger number

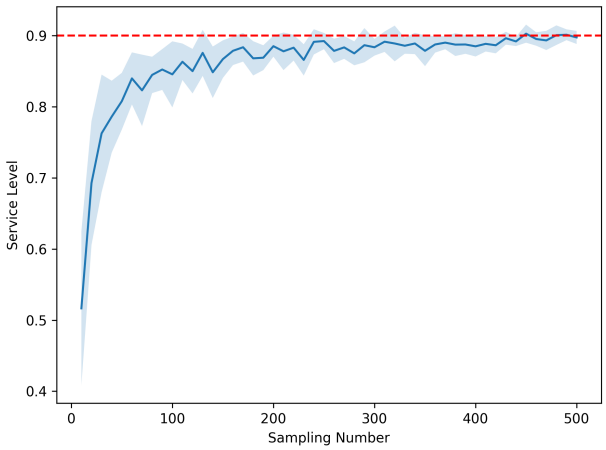


Figure 10: Convergence of SAA model under different scenario size.

of samples allows for a better simulation of the given distribution. Simultaneously, we observe that as the number of customers increases, a greater number of samples is needed to effectively simulate this multivariate distribution (typically more than 500 in our case).

Next, we analyze the performance of the SAA model with and without supplementary sampling from the empirical distribution. In this experiment, we generate a realized scenario set R from the underlying true demand distribution, and estimate the empirical distribution $\hat{\mathbb{P}}_{\psi}$ on R. Then we obtain the supplementary sample set S by sampling from $\hat{\mathbb{P}}_{\psi}$, and do the network decision based on $\overline{S} = R \cup S$ (here we set $|\overline{S}| = 500$ as a fixed value). Then we compare the SL performance of the network obtained by with sampling from the empirical distribution (hereafter referred to as SL-S) and only using realized scenarios (hereafter referred to as SL-R). The comparison results are shown in Figure 11.

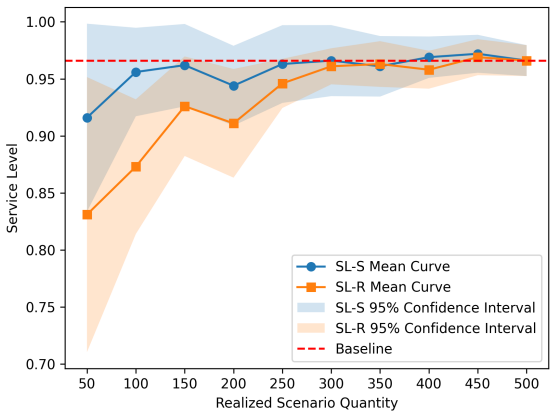


Figure 11: (Color Online) Service level performance of CCP model with and without sampling.

Figure 11 generates the following insights: (i) Whether using samples from the empirical distribution or not, the performance of network design solutions in terms of service level converges to the preset level with

the increasing number of realized scenarios. (*ii*) When the number of realized scenarios is limited, sampling from the empirical distribution can still improve the performance of network solutions, even if the empirical distribution may be imprecise at this point. This is because obtaining a certain number of samples from the empirical distribution can help avoid the omission of low-probability scenarios due to insufficient samples. This, to some extent, compensates for the limited discovery of the demand distribution when there are fewer realized scenarios.

5.5. Analysis of Model Parameters

In this section, we conduct sensitivity analysis on several other factors including the parameters in the demand (*e.g.*, demand volume, surge intensity, surge frequency), and 3PL resources rental prices. All experiments in this section are conducted under instance 1.

5.5.1. On the Demand Level

The demand volume is the representation of the average customer demand of a 4PLN, which affects the demand in both regular demand period and surge demand period. The numerical results are shown in Figure 12.

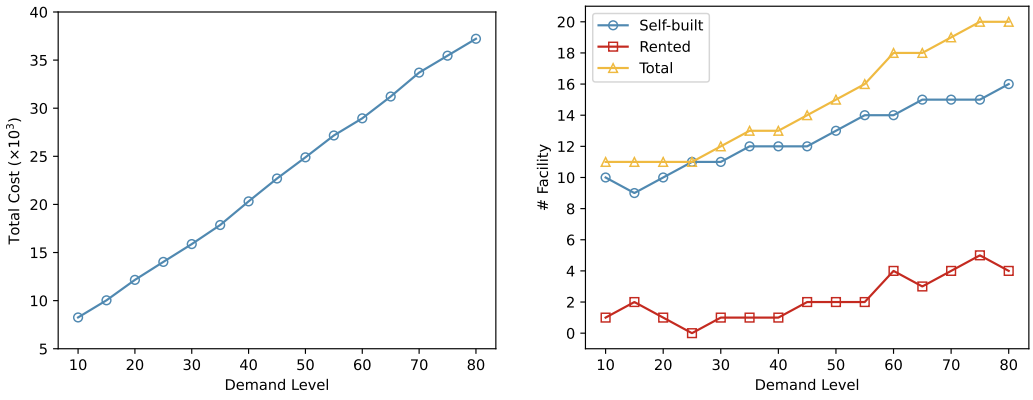


Figure 12: Effect of stochastic demand level.

It can be seen from Figure 12 that, as the demand level increases, the overall network cost grows significantly. Consequently, the 4PLN is under higher operational pressure, so that the planner should select to build or rent more facilities to warrant a sufficient network capacity. In the face of a higher demand level, the number of self-built facilities in the network rises, which is the main driving force for the rise of the total number of network facilities. In addition, the number of rented facilities fluctuates with the number of self-built facilities. It can be seen that increasing the capacity of the self-built facilities is a more effective solution to cope with a higher demand level.

5.5.2. On the Surge Intensity

In our surge demand model, more than one factor affects the surge intensity (i.e. μ_a and σ_a). To better understand the effect of the surge intensity, we focus on the surge coefficient $k = \frac{\mathbb{E}(\tilde{D}_s)}{\mathbb{E}(\tilde{D}_r)}$, which represents the intensity of demand surge compared with regular demand. The results of numerical experiments of surge coefficient are shown in Figure 13.

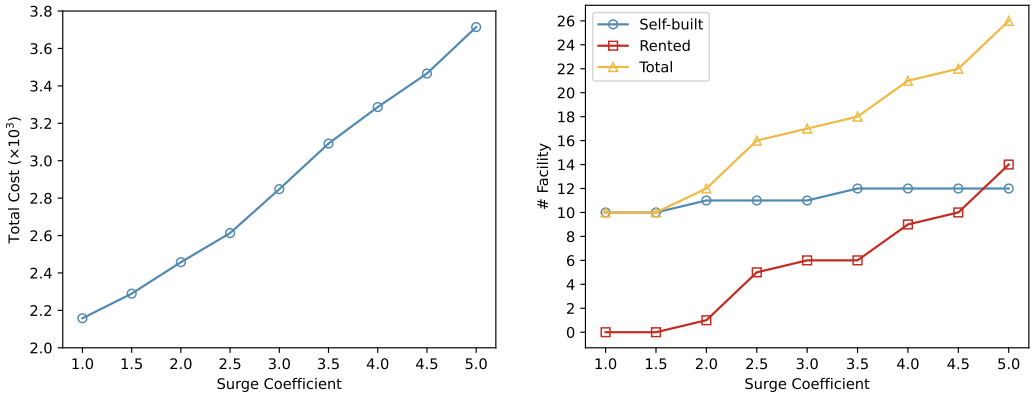


Figure 13: Effect of surge coefficient.

According to Figure 13, we observe that a bigger surge coefficient leads to an increase in the network's total cost because the increase in the surge coefficient gives rise to a higher volume of the surge demand, so the 4PL planner needs to expand the network capacity and thus incur higher operating costs to handling the orders. Specifically, the number of rented facilities rises significantly, which is the main reason for the total number of facilities to boost. But when the surge demand occurs less frequently, the impact of the surge coefficient on the number of self-built facilities is less evident. In response to the higher surge in demand, expanding the network capacity through temporary rental facilities seems a more effective decision.

5.5.3. On the Demand Surge Frequency

In each decision-making cycle, surge demand triggered by events may appear multiple times. The demand surge frequency describes the occurrence times of surge demand in a cycle. In this section, numerical experiments are conducted under different surge frequencies, the results are shown in Figure 14.

It can be seen that when the frequency increases, the greater the pressure on the 4PLN to withstand strong disturbances of demand surging. As a result, the total cost of the network increases as more facility resources are needed to satisfy more customer demand. The increasing demand surge frequency in the decision-making cycle means that the planner needs to focus on the management of the capacity of the 4PLN to fulfill the surged demand. With the objective of minimizing the network's total cost, we see that a self-built 4PLN with a bigger capacity is a more effective solution. In addition, the number of self-built facilities increases significantly as the surge demand frequency increases. However, the number of rented facilities looks quite insensitive: demand surges occur frequently, and it is not an economic option to rent facilities because of

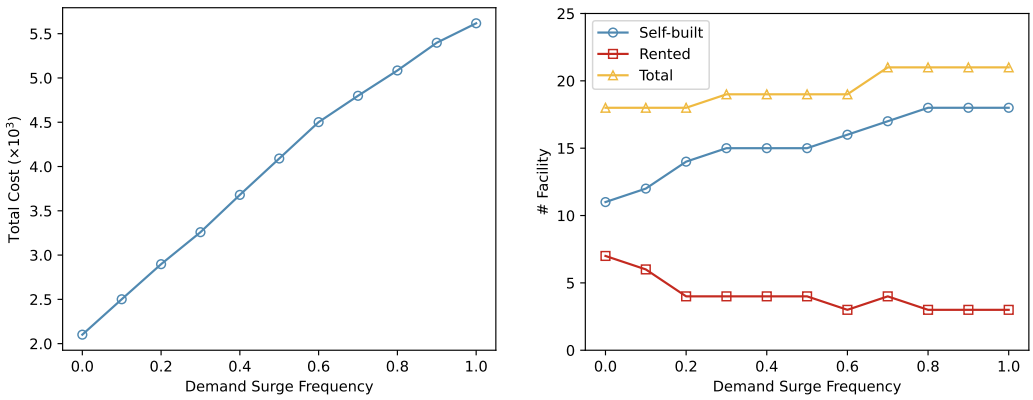


Figure 14: Effect of demand surge frequency.

the extra costs and remunerations. In extreme cases, when the demand surge frequency reaches one, the network is faced with high-level regular demand, so it becomes necessary to self-built facilities to increase network capacity.

5.5.4. On the Rental Price

The 3PL resources rental price is the key factor that affects the 4PLN planner's choice of self-built or rented facilities. To investigate the effect of 3PL resource's rental price on 4PLN designing, we conduct numerical experiments of which the results are reported in Figure 15.

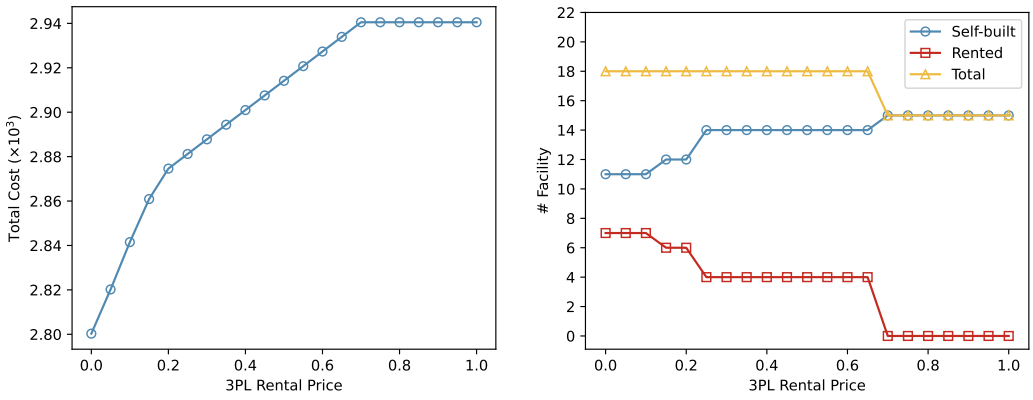


Figure 15: Effect of 3PL resource's rental price.

According to Figure 15, as the price increases, the overall network cost first increases and then plateaus after the rental price reaches a certain threshold. To understand this, let's focus on how the rental price influences the 4PLN's decision-making. In the right-hand panel of Figure 15, we see that the number of self-built facilities in the 4PLN increases in price, while the number of rented facilities decreases and eventually diminishes to zero, which validates our observation in the left-hand panel (the total cost is independent of the price changes when no rental is needed due to its expensiveness).

The costs incurred in renting facilities during the surge demand periods can be seen as revenue for the 3PL resource owners. We explored the total revenue of 3PL resource owners at different demand levels and calculated the optimal rental price for 3PL resource holders. The results are shown in the Figure 16.

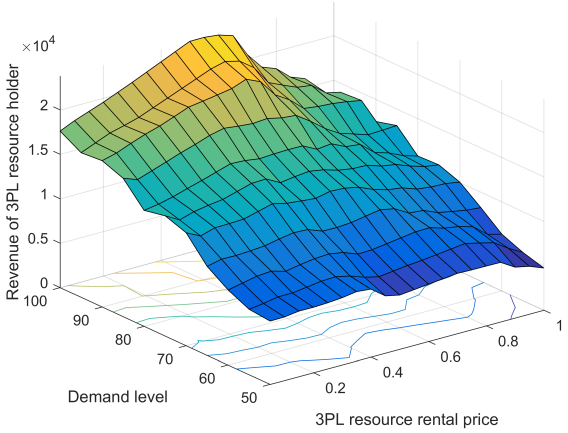


Figure 16: Effect of demand level and 3PL resources rental price on revenue of 3PL resource owners.

The total revenue of 3PL resources increases significantly as the demand level becomes larger. However, the optimal rental price for 3PL resources is not monotonically increasing in the demand level. When the demand level is sufficiently low or high, 3PL resource owners should attract 4PL planners to rent their facilities by lowering the rental prices; When the demand volume is above a certain level, 4PL planners need to trade off between self-building and renting, so 3PL resource owners are likely to increase their revenue by raising rental price.

5.5.5. Adapting the Network to Diverse Surge Scenarios

To understand the trade-off between self-building and renting in 4PL networks under different demand surge scenarios, we established four distinct situations by combining the following factors: (i) surge intensity (measured by the surge coefficient defined in Section 5.5.2): high level with $k = 2.5$, low level with $k = 1.2$; (ii) surge frequency: high level with $\varphi = 0.25$, low level with $\varphi = 0.05$. We conducted numerical experiments on Instance 1, and the results are illustrated in Figure 17.

The above results suggest: (i) As the surge intensity increases, the total quantity of 3PLs used in the network also increases. Specifically, when the surge frequency is low, renting 3PLs is preferred, while when the surge frequency is high, self-building 3PLs is more favorable. (ii) Correspondingly, as the surge frequency increases, the preference shifts from using rented 3PLs to self-building.

5.6. The Value of 4PL

In order to investigate the advantages of network solutions under the 4PL mode, we conducted comparative experiments in an environment with a total of 5 3PLs. In this setting, we specified that under the 3PL mode, all network facilities must belong to the same 3PL. The comparative results are shown in Figure 18.

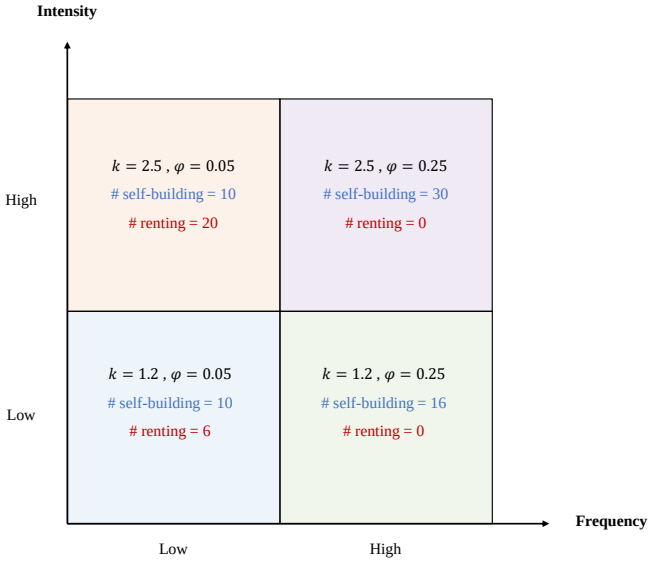


Figure 17: Network construction under different surge scenarios.

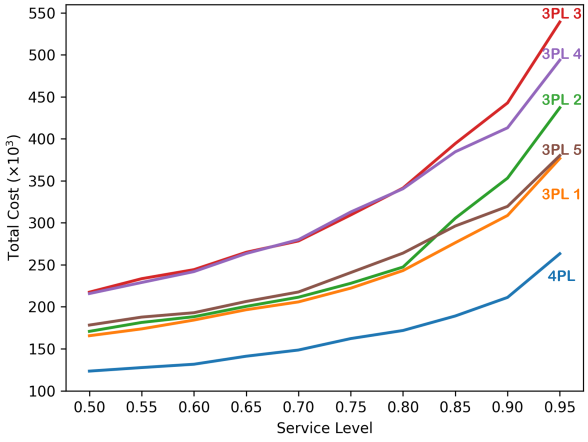


Figure 18: (Color Online) Systemic cost under 4PL and different 3PLs.

The results indicate that the total cost of the 4PL network is consistently smaller than that of relying on a single 3PL. This is because, as a platform, the 4PL can select the optimal solution from a potential network composed of multiple 3PLs. This also reflects the *pooling effect* of the 4PL as a platform.

6. Conclusion

In this paper, we study a novel 4PLN capacity planning problem in the face of a surged demand. We investigate the impact of demand surges on network design and study the optimal ways to design a 4PLN under the trade-off between self-building and renting from 3PL resource holders. We propose to model the event-triggered surge demand by a jump process and we solve a stochastic optimization problem subject to SL requirements in the form of chance constraints, i.e., we stipulate that the probability of satisfying customer stochastic demand be at least at certain designated target. To facilitate the computation, we give a scenario-based MIP reformulated model in the form of a knapsack problem. Then, under the cell-and-bound framework, we propose a new weight strategy and scenarios learning strategy, which can help improve the solving efficiency by reducing redundant cell enumeration and sub-problem solution complexity. We design two new algorithms: G-C&B and LEGO-C&B, and we conduct a series of numerical studies to test their performance.

Through numerical experiments, we investigate how key model parameters impact the performance of these algorithms. Our analysis generates several useful insights that are of practical interest. For instance, in the face of stochastic demand with surge characteristics, self-built facilities, and transportation carriers are the primary methods to warrant a sufficient network capacity, and temporary rental of resources is an effective secondary option. We also demonstrate that the rental price of 3PL resources has an interesting impact on 4PLN design, even though rented resources are not the primary way for network construction. The detailed managerial insights for capacity planning with event-triggered demand surges can be summarized as follows: (*i*) With the increase of surge intensity, the total quantity of 3PLs employed in the network should also increase. Specifically, in cases with low surge frequency, opting for rented 3PLs is preferable, whereas in cases with high surge frequency, self-building becomes more advantageous. (*ii*) When the surging intensity remains constant but the surge frequency increases, expanding network capacity with more self-built facilities is more cost-effective. (*iii*) When the rental price is high, the manager should consider reducing the frequency of promotional events to reduce excessive costs.

Acknowledgments

This work is supported by the NSFC Key Supported Project of the Major Research Plan Grant No. 92267206; the NSFC Grant No. 62032013; the Liaoning Revitalizing Talent Program Grant No. XLYC2202045; and the China Scholarship Council Program Grant No. 202206080038.

References

Alexander, G. J., & Baptista, A. M. (2004). A comparison of var and cvar constraints on portfolio selection with the mean-variance model. *Management Science*, *50*, 1261–1273.

Benati, S., & Rizzi, R. (2007). A mixed integer linear programming formulation of the optimal mean/value-at-risk portfolio problem. *European Journal of Operational Research*, *176*, 423–434.

Bi, G., Shen, F., & Xu, Y. (2024). Third-party logistics firm's technology investment and financing options in platform-based supply chain with 4pl service. *Naval Research Logistics (NRL)*, .

Bookbinder, J. H., & Tan, J.-Y. (1988). Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science*, *34*, 1096–1108.

Büyüközkan, G., Feyzioğlu, O., & Ersoy, M. Ş. (2009). Evaluation of 4PL operating models: A decision making approach based on 2-additive choquet integral. *International Journal of Production Economics*, *121*, 112–120.

Cai, N., & Yang, X. (2021). A computational approach to first passage problems of reflected hyperexponential jump diffusion processes. *INFORMS Journal on Computing*, *33*, 216–229.

Cao, C., Liu, J., Liu, Y., Wang, H., & Liu, M. (2023). Digital twin-driven robust bi-level optimisation model for COVID-19 medical waste location-transport under circular economy. *Computers & Industrial Engineering*, *186*, 109107.

Chen, F. Y., & Krass, D. (2001). Inventory models with minimal service level constraints. *European Journal of Operational Research*, *134*, 120–140.

Chen, W., Sim, M., Sun, J., & Teo, C.-P. (2010). From cvar to uncertainty set: Implications in joint chance-constrained optimization. *Operations Research*, *58*, 470–485.

Chou, M. C., Chua, G. A., Teo, C.-P., & Zheng, H. (2010). Design for process flexibility: Efficiency of the long chain and sparse structure. *Operations Research*, *58*, 43–58.

Deng, Y., Jia, H., Ahmed, S., Lee, J., & Shen, S. (2021). Scenario grouping and decomposition algorithms for chance-constrained programs. *INFORMS Journal on Computing*, *33*, 757–773.

Ewald, C., & Zou, Y. (2021). Analytic formulas for futures and options for a linear quadratic jump diffusion model with seasonal stochastic volatility and convenience yield: Do fish jump? *European Journal of Operational Research*, *294*, 801–815.

Gattorna, J., & Jones, T. (1998). *Strategic supply chain alignment: best practice in supply chain management*. Gower Publishing, Ltd.

Govindan, K., Fattahi, M., & Keyvanshokooh, E. (2017). Supply chain network design under uncertainty: A comprehensive review and future research directions. *European Journal of Operational Research*, *263*, 108–141.

Guchhait, R., & Sarkar, B. (2025). Economic evaluation of an outsourced fourth-party logistics (4pl) under a flexible production system. *International Journal of Production Economics*, *279*, 109440.

Huang, L., Song, J.-S., & Tong, J. (2016). Supply chain planning for random demand surges: Reactive capacity and safety stock. *Manufacturing & Service Operations Management*, *18*, 509–524.

Huang, M., Ren, L., Lee, L. H., & Wang, X. (2015). 4PL routing optimization under emergency conditions. *Knowledge-Based Systems*, *89*, 126–133.

Huang, M., Tu, J., Chao, X., & Jin, D. (2019). Quality risk in logistics outsourcing: A fourth party logistics perspective. *European Journal of Operational Research*, *276*, 855–879.

Jiang, S., Huang, M., Zhang, Y., Wang, X., & Fang, S.-C. (2024). Fourth-party logistics network design with demand surge: A greedy scenario-reduction and scenario-price based decomposition algorithm. *International Journal of Production Economics*, *269*, 109135.

Küçükyavuz, S., & Jiang, R. (2022). Chance-constrained optimization under limited distributional information: A review of reformulations based on sampling and distributional robustness. *EURO Journal on Computational Optimization*, *10*, 100030.

Kutlu, S. (2007). *Fourth Party Logistics: Is It the Future of Supply Chain Outsourcing?*. best global publishing.

Liu, Q., Zhang, C., Zhu, K., & Rao, Y. (2014). Novel multi-objective resource allocation and activity scheduling for fourth party logistics. *Computers & Operations Research*, *44*, 42–51.

Liu, S., Hua, G., Cheng, T., & Dong, J. (2021). Unmanned vehicle distribution capacity sharing with demand surge under option contracts. *Transportation Research Part E: Logistics and Transportation Review*, *149*, 102320.

Liu, X., Küçükyavuz, S., & Luedtke, J. (2016). Decomposition algorithms for two-stage chance-constrained programs. *Mathematical Programming*, *157*, 219–243.

Liu, Y. (2018). Staffing to stabilize the tail probability of delay in service systems with time-varying demand. *Operations Research*, *66*, 514–534.

Liu, Y., Sun, X., & Hovey, K. (2022). Scheduling to differentiate service in a multiclass service system. *Operations Research*, *70*, 527–544.

Luedtke, J., & Ahmed, S. (2008). A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, *19*, 674–699.

Luedtke, J., Ahmed, S., & Nemhauser, G. L. (2010). An integer programming approach for linear programs with probabilistic constraints. *Mathematical Programming*, *122*, 247–272.

Lyu, G., Chou, M. C., Teo, C.-P., Zheng, Z., & Zhong, Y. (2022). Stochastic knapsack revisited: The service level perspective. *Operations Research*, *70*, 729–747.

Mak, H.-Y., Rong, Y., & Shen, Z.-J. M. (2013). Infrastructure planning for electric vehicles with battery swapping. *Management Science*, *59*, 1557–1575.

Marasco, A. (2008). Third-party logistics: A literature review. *International Journal of Production Economics*, *113*, 127–147.

Melo, M. T., Nickel, S., & Saldanha-Da-Gama, F. (2009). Facility location and supply chain management–a review. *European Journal of Operational Research*, *196*, 401–412.

Nemirovski, A. (2012). On safe tractable approximations of chance constraints. *European Journal of Operational Research*, *219*, 707–718.

Nemirovski, A., & Shapiro, A. (2007). Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, *17*, 969–996.

Porras, Á., Domínguez, C., Morales, J. M., & Pineda, S. (2023). Tight and compact sample average approximation for joint chance-constrained problems with applications to optimal power flow. *INFORMS Journal on Computing*, *35*, 1454–1469.

Rahimian, H., & Mehrotra, S. (2019). Distributionally robust optimization: A review. URL: `https://doi.org/10.48550/arXiv.1908.05659`.

Roni, M. S., Eksioglu, S. D., Jin, M., & Mamun, S. (2016). A hybrid inventory policy with split delivery under regular and surge demand. *International Journal of Production Economics*, *172*, 126–136.

Roni, M. S., Jin, M., & Eksioglu, S. D. (2015). A hybrid inventory management system responding to regular demand and surge demand. *Omega*, *52*, 190–200.

Ruszczyński, A. (2002). Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra. *Mathematical Programming*, *93*, 195–215.

Shu, J., Teo, C.-P., & Shen, Z.-J. M. (2005). Stochastic transportation-inventory network design problem. *Operations Research*, *53*, 48–60.

Sinha, P., Kumar, S., & Chandra, C. (2023). Strategies for ensuring required service level for covid-19 herd immunity in indian vaccine supply chain. *European Journal of Operational Research*, *304*, 339–352.

Sleumer, N. H. (1999). Output-sensitive cell enumeration in hyperplane arrangements. *Nordic Journal of Computing*, *6*, 137–147.

Tao, Y., Chew, E. P., Lee, L. H., & Shi, Y. (2017). A column generation approach for the route planning problem in fourth party logistics. *Journal of the Operational Research Society*, *68*, 165–181.

Wang, H., Huang, M., Ip, W., & Wang, X. (2021). Network design for maximizing service satisfaction of suppliers and customers under limited budget for industry innovator fourth-party logistics. *Computers & Industrial Engineering*, *158*, 107404.

Xiao, N., Liu, X., & Toh, K.-C. (2024). Dissolving constraints for Riemannian optimization. *Mathematics of Operations Research*, *49*, 366–397.

Yin, M., Huang, M., Qian, X., Wang, D., Wang, X., & Lee, L. H. (2021). Fourth-party logistics network design with service time constraint under stochastic demand. *Journal of Intelligent Manufacturing*, *34*, 1203–1227.

Yin, M., Huang, M., Wang, X., & Lee, L. H. (2022). Fourth-party logistics network design under uncertainty environment. *Computers & Industrial Engineering*, *167*, 108002.

Yu, H., Huang, M., & Yue, X. (2024). Sharing the shared rides: Multi-party carpooling supported strategy-proof double auctions. *Production and Operations Management*, *33*, 1569–1590.

Zhang, Y., Gao, Z., Huang, M., Jiang, S., Yin, M., & Fang, S.-C. (2022). Multi-period distribution network design with boundedly rational customers for the service-oriented manufacturing supply chain: A 4PL perspective. *International Journal of Production Research*, (pp. 1–20).

Zhang, Y., Gao, Z., Huang, M., Jiang, S., Yin, M., & Fang, S.-C. (2024a). Multi-period distribution network design with boundedly rational customers for the service-oriented manufacturing supply chain: A 4pl perspective. *International Journal of Production Research*, *62*, 7412–7431.

Zhang, Y., Huang, M., Gao, Z., Jiang, S., Fang, S.-C., & Wang, X. (2024b). Multi-period fourth-party logistics network design from the viability perspective: a collaborative hyper-heuristic embedded with double-layer q-learning algorithm. *International Journal of Production Research*, (pp. 1–31).

Zhang, Z., Gao, C., & Luedtke, J. (2023). New valid inequalities and formulations for the static joint chance-constrained lot-sizing problem. *Mathematical Programming*, *199*, 639–669.

Zheng, X., Wu, B., & Cui, X. (2017). Cell-and-bound algorithm for chance constrained programs with discrete distributions. *European Journal of Operational Research*, *260*, 421–431.

Appendix A Technical Proofs

A.1 Proof of Theorem 1

Our proof primarily builds upon the results presented in Luedtke & Ahmed (2008). To facilitate the analysis, we first express the original CCP and SAA models in the same form as the model in Luedtke & Ahmed (2008). Without loss of generality, we collectively denote all decision variables as $\boldsymbol{w} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}^r, \boldsymbol{z}^s)$, and let $\boldsymbol{\Omega}$ represent the corresponding feasible domain. Denote $z^s_{(j)} = \sum_{i\in\mathcal{F}} \sum_{k\in\mathcal{V}_{ij}} z^s_{ijk}$ and $\tilde{d}_{s(j)}$ to represent the designed capacity and stochastic demand at customer j, respectively. Define function

$$G(\boldsymbol{w}, \tilde{\boldsymbol{d}}_s) := \left(z^s_{(1)} - \tilde{d}_{s(1)}, \ldots, z^s_{(|\mathcal{D}|)} - \tilde{d}_{s(|\mathcal{D}|)}\right)^T \in \mathbb{R}^{|\mathcal{D}|\times 1}.$$

Recall $C(\boldsymbol{w})$ is the total cost function, we can then equivalently express the CCP model as follows:

$$v^*_\alpha = \min\left\{C(\boldsymbol{w}) : \mathbb{P}\left(G(\boldsymbol{z}^s, \tilde{d}_s \geq \boldsymbol{0})\right) \geq \alpha_s,\ \boldsymbol{w} \in \boldsymbol{\Omega}\right\} \tag{A.1}$$

Given a sample set $R = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N\}$, the SAA model with a specific service level target α_s is formulated as follows

$$\bar{v}^*_R = \min\left\{C(\boldsymbol{w}) : \frac{1}{N}\sum_{q=1}^{N} \mathbb{I}\left(G(\boldsymbol{z}^s, \tilde{\boldsymbol{\xi}}_q) \geq \boldsymbol{0}\right) \geq \alpha_s,\ \boldsymbol{w} \in \boldsymbol{\Omega}\right\} \tag{A.2}$$

Our SAA model (17) with service level target α_s can be expressed as

$$v^*_R = \min\left\{C(\bar{\boldsymbol{w}}) : \bar{\boldsymbol{w}} \in \Omega_u\right\} \tag{A.3}$$

where

$$\boldsymbol{\Omega}_u = \left\{ \bar{\boldsymbol{w}} \in \boldsymbol{\Omega} \,\middle|\, \begin{array}{l} \displaystyle\sum_{i\in\mathcal{F}} \sum_{k\in\mathcal{V}_{ij}} \bar{z}^s_{ijk} - u_q \cdot \xi_{q(j)} \geq 0,\ \forall j \in \mathcal{D}, \forall q \in [N] \\ \displaystyle\sum_{q=1}^{N} u_q \geq \alpha_s, \\ \boldsymbol{u} \in \{0,1\}^N. \end{array} \right\}.$$

We then establish the equivalence between models (A.2) and (A.3), as outlined in the following lemma. The proof of Lemma 1 will be presented after the completion of Proposition 1's proof.

Lemma 1. Model (A.2) and model (A.3) are equivalent, i.e.,

$$v^*_R = \bar{v}^*_R, \quad \text{and} \quad X^*_R = \bar{X}^*_R,$$

where X^*_R is the collection of all optimal solutions $\boldsymbol{w}^*$ of model (A.3), and $\bar{X}^*_R$ is that of model (A.2).

Lemma 1 demonstrates that our SAA model is equivalent to the model in the form of (A.2), which is consistent with the formulation (see equation xx) in Luedtke & Ahmed (2008). We now apply their results to demonstrate the convergence of the SAA model to the true problem.

Lemma 2 (Luedtke & Ahmed (2008)'s convergence result)**.** Since $\mathbf{\Omega}$ is a finite set, let $X_\alpha^* \subseteq \mathbf{\Omega}$ represent the collection of optimal solutions for (A.1). Assume $\max\left\{\mathbb{P}\left(G(\boldsymbol{w}, \tilde{\boldsymbol{d}}_s) \not\geq \mathbf{0}\right), \boldsymbol{w} \in X_\alpha^*\right\} < 1 - \alpha_s$. Then, it holds that

$$\mathbb{P}(v_R^* = \bar{v}_\alpha^*) \geq 1 - (|\mathbf{\Omega}| + 1)\exp\{-2\kappa^2 \cdot |R|\}, \tag{A.4}$$

where $\kappa = \min\left\{\alpha_s - \max\left\{\mathbb{P}\left(G(\boldsymbol{w}, \tilde{\boldsymbol{d}}_s) \not\geq \mathbf{0}\right), \boldsymbol{w} \in X_\alpha^*\right\}, \min\left\{\mathbb{P}\left(G(\boldsymbol{w}, \tilde{\boldsymbol{d}}_s) \not\geq \mathbf{0}\right), \boldsymbol{w} \in \mathbf{\Omega} \setminus X_\alpha^*\right\} - \alpha_s\right\}$ is a positive constant.

Lemma 2 indicates that the SAA model will almost surely obtain the optimal objective function value as the sample size $|R|$ increases. Furthermore, by applying Lemma 1, we can ensure that our SAA model (A.2) will converge to the true optimal objective function value as $|R| \to \infty$, i.e.,

$$\lim_{|R|\to\infty} \mathbb{P}(v_R^* = v_\alpha^*) \to 1.$$

This completes the proof.

Finally, we present the proof of Lemma 1.

Proof of Lemma 1: The proof is carried out using a contradiction argument. Assume the optimal solution set for model (A.2) is $\bar{X}_R^*$. First, we verify that any $\bar{\boldsymbol{w}}^* \in \bar{X}_R^*$ is a feasible solution for model (A.3). Let $u_q = \mathbb{I}\left(\sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} \bar{z}_{ijk}^{s*} - \xi_{q(j)} \geq 0,\ \forall j \in \mathcal{D}\right)$ for all $q \in [N]$, it is easy to see that $\bar{\boldsymbol{w}}^*$ is feasible for (A.3) under this specifically defined $\boldsymbol{u}$. Conversely, the optimal solutions of model (A.3) are also feasible for (A.2). Next we check the optimality of any $\bar{\boldsymbol{w}}^* \in \bar{X}_R^*$ for (A.3). Suppose there is an optimal solution $\boldsymbol{w}^*$ for (A.3) satisfies $C(\boldsymbol{w}^*) < C(\bar{\boldsymbol{w}}^*)$. As we known, $\boldsymbol{w}^*$ and the corresponding $\boldsymbol{u}^*$ are feasible for (A.2). This implies that $\bar{\boldsymbol{w}}^*$ is not optimal for (A.2), which contradicts the assumption. Conversely, any $\boldsymbol{w}^* \in X_R^*$ is also optimal for (A.2). This completes the proof.

■

A.2 Proof of Proposition 1

Assume that there are m hyperplanes in the n-dimensional Euclidean space, which is divided into at most $\sum_{j=0}^{n} C_m^j$ cells by these hyperplanes. We write

$$\sum_{j=0}^{n} C_m^j = \left(a_n^{(n)} \cdot m^n + a_{n-1}^{(n)} \cdot m^{n-1} + \cdots + a_1^{(n)} \cdot m\right) + \cdots + \left(a_2^{(2)} \cdot m^2 + a_1^{(2)} \cdot m\right) + a_1^{(1)} \cdot m + 1,$$

because $C_m^n = \frac{m!}{n!\cdot(m-n)!} = a_n \cdot m^n + a_{n-1} \cdot m^{n-1} + \cdots + a_1 \cdot m$. This expression can be further simplified by combining terms such as $b_n \cdot m^n + b_{n-1} \cdot m^{n-1} + \cdots + b_1 \cdot m^1 + 1$. It is straightforward to see that $C_m^n \leq (b_n + b_{n-1} + \cdots + b_1 + 1) \cdot m^n = c \cdot m^n$. This concludes that the maximum number of cells for hyperplane division is bounded by $O(m^n)$. When applied to our 4PLN problem, each scenario corresponds to a hyperplane class. Each hyperplane class generates a positive polyhedron defined as (19), and we are only concerned with whether a cell lies in each positive polyhedron. Notice that the number of regions enclosed by positive polyhedrons will not exceed $O(m^n)$. This is because, within a hyperplane class with size

h, the family of hyperplanes forms at most $O(h^n)$ regions, but for the corresponding positive polyhedron, the number of regions will be reduced to 2. Therefore, for a total of m positive polyhedrons, the number of 'cells' is bounded by $O(m^n)$. In our problem, each demand scenario includes the demand of all customers and is therefore $|\mathcal{D}|$-dimensional, and each scenario gives a positive polyhedron, totaling $|\overline{S}|$. Hence, we have shown that the maximum number of cells to be explored when applying the G-C&B algorithm to the 4PLN problem is bounded by $O(|\overline{S}|^{|\mathcal{D}|})$.

■

A.3 Proof of Proposition 2

We know that a decision tree is generated within the C&B framework to explore feasible cells, with the depth of cells corresponding to the SL. In Algorithm 1, we assess the weight (i.e., price) of all cells with equal depth and then explore only the cell with the smallest price. Since we terminate the algorithm once the depth of the cell reaches the SL, we can calculate that only $\alpha_s \cdot |\overline{S}|$ cells are explored. That is, the number of cells to be explored in G-C&B is bounded by $O(|\overline{S}|)$.

■

Appendix B An Example of 4PLN Design and Algorithms' Decision Tree

We present an example of 4PLN to demonstrate the efficiency of our proposed G-C&B algorithm. The potential network is defined as Figure B1, and we set the capacity of DCs and TPs uniformly to 381.6 and 190.8, respectively.

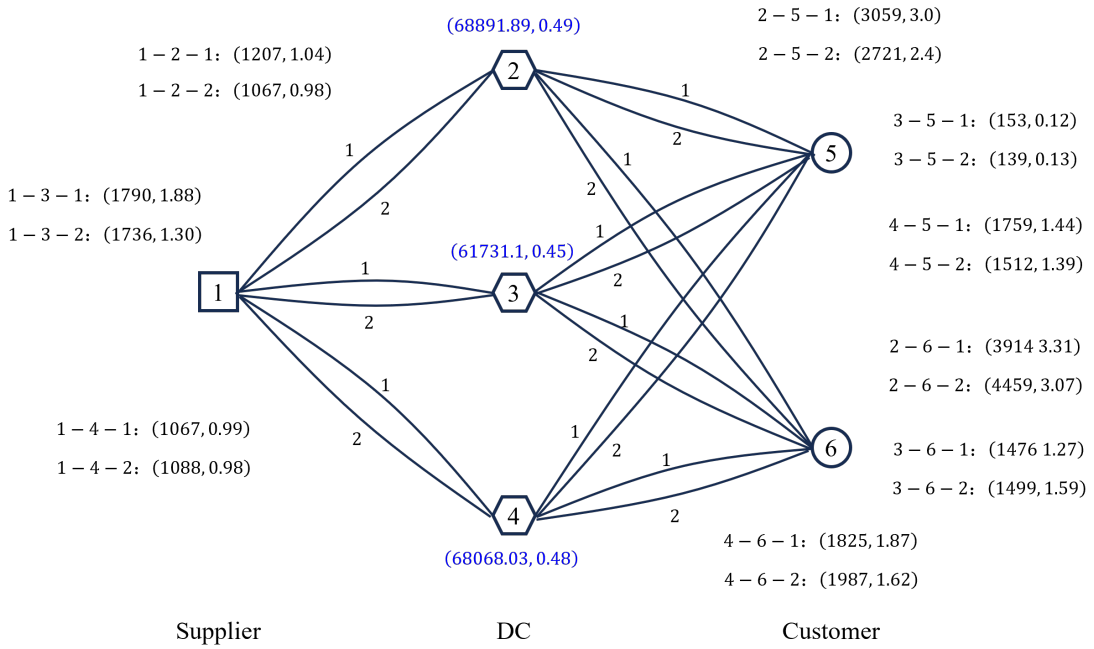


Figure B1: Network structure of 4PLN example.

Then, given demand scenario set as

$$\left\{\begin{array}{l}\boldsymbol{\xi}_1 = (52.63, 52.33)^T, \boldsymbol{\xi}_2 = (59.59, 64.61)^T, \boldsymbol{\xi}_3 = (72.58, 60.64)^T \\ \boldsymbol{\xi}_4 = (62.32, 68.88)^T, \boldsymbol{\xi}_5 = (64.03, 44.15)^T, \boldsymbol{\xi}_6 = (42.63, 40.12)^T\end{array}\right\}, \tag{B.1}$$

we provide the decision tree of B&B, C&B, and G-C&B algorithm in Figure B2.

In this example, it can be observed that the number of cells to explore in the C&B algorithm is smaller than the number of branching nodes explored by the B&B algorithm. Furthermore, with the adoption of the greedy pricing strategy, the G-C&B algorithm requires even fewer cells to explore. This intuitively demonstrates the improvement in algorithm efficiency.

Appendix C Benchmark Models Used in Section 5.3

C.1 The Expected-Value Model

Recall that $\boldsymbol{\chi}$ is the compact feasible set determined by constraints (13c)-(13l). Then given a realized surged demand scenario set S, the operational cost for surged demand is

$$\begin{aligned} C_{PS} &= \mathbb{E}_{(\boldsymbol{\xi}\in S)}\left[\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^p_i\cdot z^s_{ijk} + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot v_{ijk} + (1+\gamma)\cdot\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot w_{ijk}\right] \\ &= \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^p_i\cdot\mathbb{E}_{(\boldsymbol{\xi}\in S)}\left(z^s_{ijk}\right) + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot\mathbb{E}_{(\boldsymbol{\xi}\in S)}\left(v_{ijk}\right) + (1+\gamma)\cdot\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot\mathbb{E}_{(\boldsymbol{\xi}\in S)}\left(w_{ijk}\right). \end{aligned} \tag{C.1}$$

Then the objective function of the expected value model can be formulated as

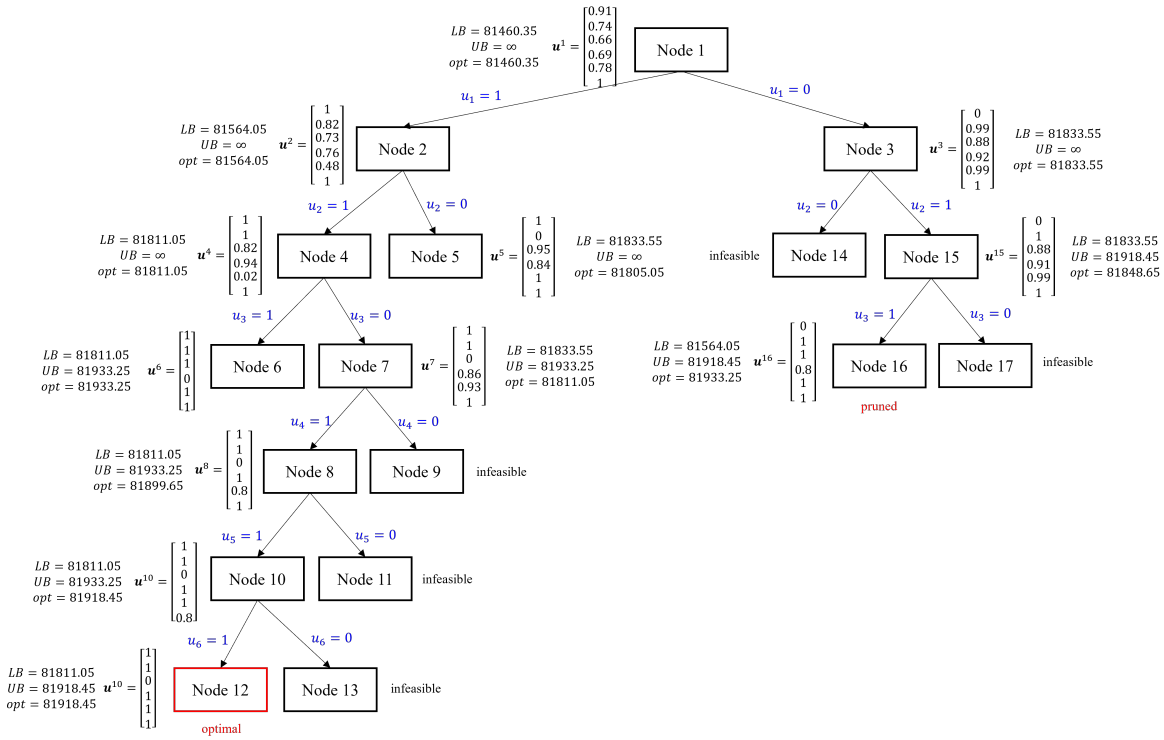
$$\begin{aligned} C_E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{\xi}) = &\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^f_{ijk}\cdot x^r_{ijk} + \sum_{i\in\mathcal{F}} c^l_i\cdot y^r_i + \theta_c\cdot\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^f_{ijk}\cdot\left(x^s_{ijk}-x^r_{ijk}\right) + \theta_t\cdot\sum_{i\in\mathcal{F}} c^l_i\cdot\left(y^s_i-y^r_i\right) \\ &+\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot z^r_{ijk} + \sum_{i\in\mathcal{F}} c^p_i\cdot\left(\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} z^r_{ijk}\right) + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^p_i\cdot z^s_{ijk} \\ &+\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot v_{ijk} + (1+\gamma)\cdot\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot w_{ijk}, \end{aligned} \tag{C.2}$$

and the expected value model is formulated as

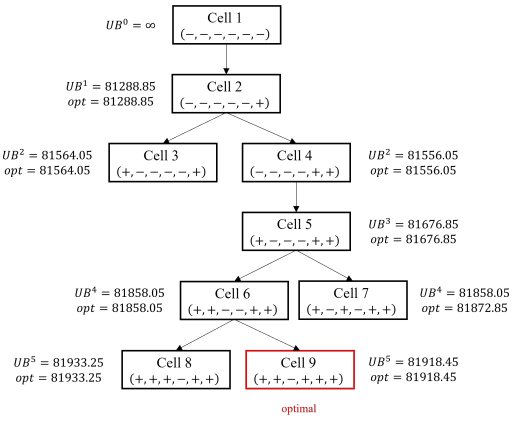
$$\begin{aligned} (\textbf{Expected Value Model}) \quad \min \quad & C_E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{\xi}) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^s_{ijk} - \mathbb{E}_{\boldsymbol{\xi}\in S}\left(\boldsymbol{\xi}^j\right) \geq 0,\ \forall j\in\mathcal{D}, \\ & (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\in\boldsymbol{\chi}. \end{aligned} \tag{C.3}$$

C.2 The Worst-Case Model

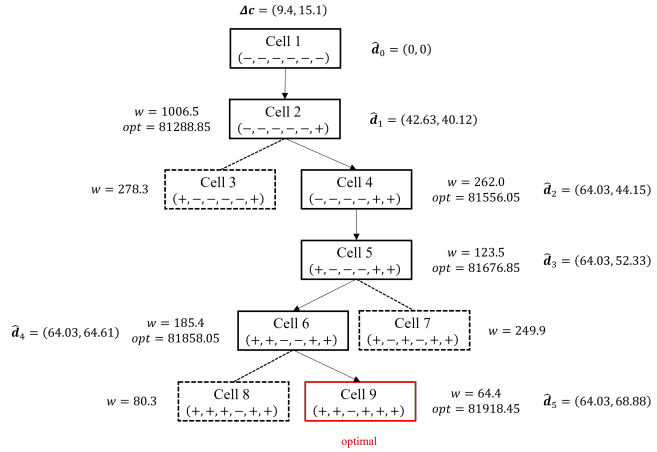
Similarly, recall that $\boldsymbol{\chi}$ is the compact feasible set determined by constraints (13c)-(13l), and given a realized surged demand scenario set S, the objective function of the worst-scenario model is derived as



(a)



(b)



(c)

Figure B2: The schematic presentations of the decision tree of (a) B&B, (b) C&B, and (c) G-C&B.

$$\min \sup_{\boldsymbol{\xi}\in S} C_W(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{\xi}) = \min \sup_{\boldsymbol{\xi}\in S} \left[\begin{array}{l} \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^f_{ijk}\cdot x^r_{ijk} + \sum_{i\in\mathcal{F}} c^l_i \cdot y^r_i + \theta_c \cdot \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^f_{ijk}\cdot\left(x^s_{ijk}-x^r_{ijk}\right) \\ + \theta_t \cdot \sum_{i\in\mathcal{F}} c^l_i\cdot(y^s_i - y^r_i) + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot z^r_{ijk} + \sum_{i\in\mathcal{F}} c^p_i \cdot \left(\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} z^r_{ijk}\right) \\ + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^p_i\cdot z^s_{ijk} \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot v_{ijk} + (1+\gamma)\cdot\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{k\in\mathcal{V}_{ij}} c^t_{ijk}\cdot w_{ijk} \end{array} \right], \tag{C.4}$$

then the worst scenario model can be equivalently formulated as

$$\begin{aligned} (\textbf{Worst Scenario Model}) \quad & \min \sup_{\boldsymbol{\xi}\in S} v(\boldsymbol{\xi}) \\ & \text{where } v(\boldsymbol{\xi}) = \min \quad C_W(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{\xi}) \\ & \qquad\qquad \text{s.t.} \quad \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^s_{ijk} - \xi^j \geq 0,\ \forall j\in\mathcal{D}, \\ & \qquad\qquad\qquad (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\in\boldsymbol{\chi}. \end{aligned} \tag{C.5}$$

Considering the worst-case scenario in the marginal distribution for each customer, we can formulate the following worst demand vector model

$$\begin{aligned} (\textbf{Worst Marginal Demand Model}) \quad & \min \quad C_W(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{\xi}) \\ & \text{s.t.} \quad \sum_{i\in\mathcal{F}}\sum_{k\in\mathcal{V}_{ij}} z^s_{ijk} - \max_{\boldsymbol{\xi}\in S}\left\{\xi^j\right\} \geq 0,\ \forall j\in\mathcal{D}, \\ & \qquad (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\in\boldsymbol{\chi}. \end{aligned} \tag{C.6}$$

In the worst scenario model, we aim to minimize the total cost under the worst marginal demand scenario, while in the worst demand model, the objective is to minimize the total cost under the worst (i.e., largest) demand for each customer individually. Notably, the worst demand vector (consisting of the largest demand in each customer among all scenarios) may break the correlation between customer demands (which can be reflected in demand scenarios) and could result in a relatively conservative solution under the worst-case for the marginal distribution.

Appendix D Goodness-of-Fit Test for Samples of Different Sizes

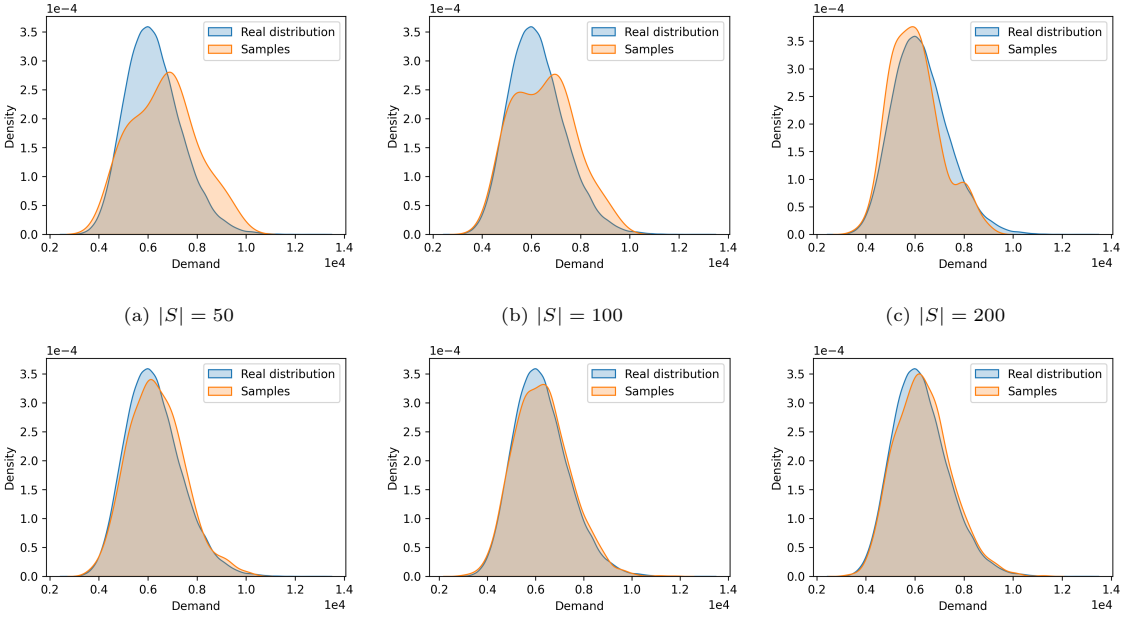
As given the real distribution of demand surge as

$$A \sim N\left(8.723, 0.182^2\right),$$

we perform random sampling to generate sample sets of various sizes and then conduct the Kolmogorov-Smirnov test to assess the fit of samples to the specified distribution. We utilize the KS statistic and p-value as metrics for goodness-of-fit, reporting their average values and worst-case values over 100 repeated trials in Table D1.

Table D1: Kolmogorov-Smirnov test for samples with different sizes.

# Samples	Average KS statistic	Average p-value	Worst KS statistic	Worst p-value
50	0.1204	0.4879	0.2392	0.0052
100	0.0868	0.4811	0.1645	0.0078
200	0.0606	0.4870	0.1160	0.0083
500	0.0384	0.4921	0.0650	0.0282
1000	0.0271	0.5073	0.0536	0.0065
2000	0.0188	0.5256	0.0341	0.0201



(a) $|S| = 50$ (b) $|S| = 100$ (c) $|S| = 200$

(d) $|S| = 500$ (e) $|S| = 1000$ (f) $|S| = 2000$

Figure D1: (Color Online) Probability Density Function (PDF) of real distribution and samples.

It is noteworthy that the results presented here are specific to the instance data employed in our experiments. For cases involving larger scale or greater system variability, a more extensive sample size might be essential to guarantee the accuracy of the fitting.