Soft Clustering-based Scenario Bundling for a Progressive Hedging Heuristic in Stochastic Service Network Design

Xiaoping Jiang[a], Ruibin Bai[a,*], Stein W. Wallace[b], Graham Kendall[c,d], Dario Landa-Silva[c]

[a] *School of Computer Science, University of Nottingham Ningbo China, Ningbo 315100, China*

[b] *Department of Business and Management Science, Norwegian School of Economics, NO-5045 Bergen, Norway*

[c] *School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK*

[d] *School of Computer Science, University of Nottingham Malaysia Campus, Semenyih 43500, Malaysia*

Abstract

We present a method for bundling scenarios in a progressive hedging heuristic (PHH) applied to stochastic service network design, where the uncertain demand is represented by a finite number of scenarios. Given the number of scenario bundles, we first calculate a vector of probabilities for every scenario, which measures the association strength of a scenario to each bundle center. This membership score calculation is based on existing soft clustering algorithms such as Fuzzy C-Means (FCM) and Gaussian Mixture Models (GMM). After obtaining the probabilistic membership scores, we propose a strategy to determine the scenario-to-bundle assignment. By contrast, almost all existing scenario bundling methods such as K-Means (KM) assume before the scenario-to-bundle assignment that a scenario belongs to exactly one bundle, which is equivalent to requiring that the membership scores are Boolean values. The probabilistic membership scores bring many advantages over Boolean ones, such as the flexibility to create various degrees of overlap between scenario bundles and the capability to accommodate scenario bundles with different covariance structures. We empirically study the impacts of different degrees of overlap and covariance structures on PHH performance by comparing PHH based on FCM/GMM with that based on KM and the cover method, which represents the state-of-the-art scenario bundling algorithm for stochastic network design. The solution quality is measured against the lower bound provided by CPLEX. The experimental results show that, GMM-based PHH yields the best performance among all methods considered, achieving nearly equivalent solution quality in a fraction of the run-time of the other methods.

Keywords

Network design; Stochastic mixed-integer programs; Progressive hedging; Scenario bundling

1. Introduction

Freight transportation concerns efficient distribution of commodities from one point to another utilizing an underlying network, such as the national highway system. Among various types of freight, less-than-truckload (LTL) transportation deals with shipments that are small relative to vehicle capacity (Hewitt et al., 2019). For LTL transportation, the underlying network is comprised of geographically dispersed end-of-line and break-bulk terminals. Each end-of-line terminal has a local service region and the shipments from various end-of-line terminals are usually combined to create truckloads of freight so that LTL carriers can spread the transportation cost out among as many customers as possible. This consolidation process is mainly performed at break-bulk terminals, where freight is unloaded, sorted, consolidated and then reloaded onto the same or different vehicles (Jiang et al., 2017).

To meet demands at certain end-of-line terminals from suppliers at other end-of-line terminals, LTL carriers need to determine which links connecting two terminals to travel along for a least-cost shipment of commodities. The selected links and their associated terminals constitute a service network over which LTL carriers operates. The design of the service network has a considerable bearing on the level of service provided and the transportation cost incurred. In addition to the spatial aspect, the temporal dimension of the demand should also be taken into account (Wang et al., 2019). A natural time constraint is that a commodity becomes available at some time and the delivery of this commodity must be completed no

* Corresponding author. *E-mail address*: ruibin.bai@nottingham.edu.cn

later than a prespecified deadline. In order to offer customers high-quality services at competitive prices, LTL carriers have to make smarter tactical decisions about the selection, routing and scheduling of services (Crainic, 2000).

There is quite a significant body of literature on deterministic service network design (e.g., Jarrah et al., 2009; Erera et al., 2013; Lindsey et al., 2016; Boland et al., 2017; Crainic, 2000; Wieberneit, 2008), where all parameters of the problem are known with certainty. However, the decision making in LTL transportation is subject to various uncertainties (Bai et al., 2014), among which the demand uncertainty is of particular importance (Hewitt et al., 2019). For LTL carriers, deterministic service network design under significant demand fluctuations is very likely to result in failure to satisfy actual customer demand or cause critical logistic assets such as vehicles to stand idle. Both situations are undesirable because they either miss opportunities to make more profits or waste capital investment in logistic resources (Gupta, 2003). Integrating uncertain demand into service network design can lead to a more robust service network, thereby not only providing reliable services to customers, but also reducing the expected cost for carriers.

Service network design with uncertain demand, also referred to as stochastic service network design, is an inherently two-stage decision problem (King and Wallace, 2012). The 'here-and-now' decisions in the first stage, which defines the structure of the service network, have to be made prior to the realization of the uncertain demand, whereas LTL carriers would wait to make decisions until after the quantities of the demand are observed in the second stage. Given customers' demand, these 'wait-and-see' decisions determine the most cost-effective way of commodity routing through the first-stage service network, which can be viewed as recourse actions taken after the uncertain demand is disclosed (Birge & Louveaux, 2011).

In most real-life applications, demand uncertainty is described by continuous probability distributions or discrete distributions with a large number of outcomes (Lium et al., 2009). Consequently, given a first-stage service network, the second-stage optimization problems are far too numerous to solve individually. A common approach to this problem is to approximate these distributions with a finite and often rather small number of scenarios, each of which has an associated probability. As the uncertain demand is realized over time, these scenarios can be organized into a hierarchical tree-like structure, referred to as a scenario tree (Høyland & Wallace, 2001). The two-stage stochastic program can then be reformulated by replacing the demand distribution with the scenario tree, giving rise to a deterministic model, known as the extensive form (Birge & Louveaux, 2011). Except for some trivial cases, directly solving the extensive form within acceptable run-time is typically beyond the capability of existing commercial solvers (e.g., CPLEX).

Fortunately, the extensive form contains some special decomposable structures that can be advantageously exploited to circumvent this difficulty. Various decomposition procedures have been developed to break up the extensive form into a set of smaller manageable subproblems (see, e.g., Santoso et al., 2005; Laporte & Louveaux, 1993; Beier et al., 2015). The scenario-based structure of the extensive form inspires a class of scenario-wise decomposition methods, which decompose the extensive form by scenarios into a set of single-scenario subproblems. A representative example is the progressive hedging algorithm (Watson et al., 2012), which iteratively solves each subproblem and aggregates subproblem solutions into an overall solution until a consensus solution among all subproblems is obtained or other stopping criteria such as the run-time limit are met. Since the individual scenario subproblems can be solved efficiently, the progressive hedging algorithm lends itself well to stochastic service network design. Although its desirable theoretical property of convergence to the global optimum in the convex case (Løkketangen & Woodruff, 1996) does not hold in this context, it can be effectively used as a heuristic to find high-quality solutions within an acceptable run-time (Watson & Woodruff, 2011).

Instead of working with scenario-wise decomposition, we can combine individual scenarios into bundles and decompose the extensive form by scenario bundles into multi-scenario subproblems, giving rise to bundle-wise decomposition. In contrast with scenario-wise decomposition, bundle-wise decomposition results in a smaller number of subproblems, albeit each of which is in a larger scale (Løkketangen & Woodruff, 1996). As long as the multi-scenario subproblems remain readily manageable, the significant reduction in the number of subproblems usually far outweighs the increased difficulty of addressing each subproblem, making bundle-wise decomposition an attractive algorithmic enhancement to the standard progressive hedging algorithm. In fact, several recent studies have shown that bundle-wise decomposition leads to a striking improvement in the computational performance of the progressive hedging heuristic (Crainic et al., 2014; Gade et al., 2016).

The effectiveness of bundle-wise decomposition has motivated a growing body of research on how to group scenarios into bundles for stochastic programs. Escudero et al. (2013) applied scenario bundling for solving the Lagrange dual of two-stage stochastic mixed 0-1 problems and obtained stronger lower bounds. In their work, the number of scenario bundles was defined as a proper divisor of the cardinality of the scenario set and the scenario set was partitioned into some mutually exclusive subsets with equal cardinalities. However, the scenario-to-bundle assignment was random in nature since no criterion concerning which scenarios would be grouped into the same bundle was given. Ryan et al. (2016) formulated the problem of determining the scenario-to-bundle assignment for two-stage stochastic mixed 0-1 programs as a mixed integer program (MIP), the objective function of which is to maximize the bound improvement. Their MIP formulation assumed that the intersection of any two scenario bundles was an empty set and the size of individual bundles was far less than the total number of scenarios. Crainic et al. (2014) proposed several scenario bundling strategies based on the k-means algorithm, most of which assigned each scenario to exactly one bundle. The scenario similarity was measured by the Euclidean distance between the commodity demand of different scenarios or between the solutions to different scenario subproblems. In particular, the authors presented the cover method, where each scenario was allocated to its two closest bundles in terms of Euclidean distance. These scenario bundling strategies were integrated as a preprocessing step into the progressive hedging-based meta-heuristic to address stochastic service network design problems. The results of their computational experiments showed that, the meta-heuristic with k-means-based scenario bundling performed better in terms of both the solution quality and computing efficiency than that with random scenario-to-bundle assignment or without scenario bundling. Among all of the proposed strategies, the cover method with commodity demand as scenario features yielded the best performance. Inspired by the cover method, we shall assume before the scenario-to-bundle assignment that a scenario belongs to multiple bundles. Unlike the cover method, each scenario is not restricted to belong to precisely two bundles in this paper. Instead, a scenario may be assigned to one or more scenario bundles, depending upon its membership scores. Besides these works, scenario bundling is being vigorously studied in other contexts, such as the multistage stochastic programs (Bakir et al., 2020; Carpentier et al., 2013; Escudero et al., 2016; Zenarosa et al., 2014) and the chance-constrained programs (Ahmed et al., 2017; Deng et al., 2017).

In almost all of the existing scenario bundling strategies, the assignment of scenarios to bundles is based on the assumption that a scenario belongs to exactly one bundle. This form of grouping is known as hard clustering in the field of pattern recognition and machine learning. One of most widely used hard clustering method is k-means. In this paper, we approach scenario bundling from a new perspective and the assignment of scenarios to bundles is based on the assumption that a scenario belongs to every bundle to a certain degree, referred to as soft clustering. If a scenario lies close to the center of a bundle, it will have a high degree of membership to that bundle. A fractional membership score is calculated to measure the association strength of a scenario to a bundle. We consider the two most commonly known types of soft clustering methods, namely fuzzy c-means and Gaussian mixture models, to calculate the membership scores. The probabilistic membership scores provide the flexibility to perform various scenario-to-bundle assignments. For example, we can assign each scenario to exactly two bundles by assigning a scenario to the bundles which have the highest and second highest membership scores. This scenario-to-bundle assignment is similar to the k-means-based cover method from Crainic et al. (2014), but we group scenarios based on their membership degrees. As we shall see later in Section 3, soft clustering-based scenario bundling also brings with it many other advantages, such as accurate reflections of the level of uncertainty over a scenario's bundle membership and the capability to accommodate scenario bundles with different covariance structures.

The main contribution of this paper is the development and empirical evaluation of the soft clustering-based scenario bundling strategies for a progressive hedging heuristic in the context of stochastic service network design. We believe that this is the first attempt to introduce the idea of soft clustering, which has its origin in the field of pattern recognition and machine learning, into scenario bundling and to study the impacts of soft clustering-based scenario bundling strategies on the computational performance of the progressive hedging heuristic. Specifically, we examine the impact of different degrees of overlap on the computational performance of the progressive hedging heuristic using fuzzy c-means, k-means and the cover method. Also, we make a comparison between the computational performance of Gaussian mixture models- and k-means-based progressive hedging heuristic to investigate whether there is any advantage in allowing scenario bundles to have different covariance structures. The observations and insights drawn from our experimental studies not only demonstrate the benefits of soft clustering-based scenario bundling strategies in improving the progressive hedging heuristic, but also provide some useful information for future research on scenario bundling strategies.

The rest of this paper is organized as follows. In Section 2, we present the mathematical model of stochastic service network design. In Section 3, we present the soft clustering-based scenario bundling method and discuss its potential advantages over k-means. In Section 4, we modify the standard progressive hedging algorithm to incorporate the resulting scenario bundles and to provide remedies for the undesirable conditions that may arise. The computational experiments comparing the progressive hedging heuristic based on different scenario bundling algorithms are described in Section 5 and 6. We conclude in Section 7 with a summary of the computational evidence and with some suggested directions for further study.

2. Problem Formulation

To present the mathematical model for stochastic service network design, we define the following notation. Let $\mathcal{N}$ represent the set of terminals in LTL transportation. The transportation service is scheduled over a time horizon of T periods, denoted by $\mathcal{T} := \{0,1,\dots,T-1\}$. For ease of illustration, we assume that the transport movement between every pair of terminals takes one time period and use t^- to represent the departure time of a movement that has arrival time t. So $t^- = T-1$ if t takes the value 0, otherwise $t^- = t-1$. The space-time network is constructed by duplicating each terminal in every time period (Bai et al., 2010). An illustrative space-time network with 3 terminals and 5 time periods is shown in Fig. 1. This network consists of (3*5=15) nodes. Every pair of nodes in two different nearby time periods is connected by an arc, representing the transport movement from one terminal at a certain time period to another terminal at the next period. We use a triplet (i,j,t) to denote an arc from terminal i at time period t^- to terminal j at time period t. In particular, an arc connecting the identical terminals in different time periods is referred to as a holding arc, which represents the activities of holding vehicles at a terminal for some time. Each arc (i,j,t) has an associated fixed cost c_{ij} incurred by the transportation or holding service. Except for the holding arcs, each arc (i,j,t) has a resource capacity denoted by u_{ij}.

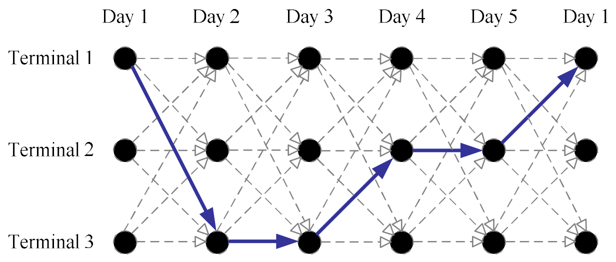

Fig. 1. A space-time network with 3 terminals over a time horizon of 5 days.

Through the underlying space-time network, there are some commodities to be delivered. We shall use $\mathcal{K}$ to denote the commodity set. Each commodity $k \in \mathcal{K}$ is characterized by its quantity, which is uncertain and will be explained later, its origin terminal $o(k)$ and the time period $\sigma(k)$ when it is available, as well as its destination terminal $s(k)$ and the delivery deadline $\tau(k)$.

To meet the transport demand in a profitable way, LTL carriers have to address the selection, routing and scheduling of services. The decision-making process has an intrinsically two-stage structure, which is illustrated in Fig. 2. In the first stage, LTL carriers make decisions about whether an arc would be included in the service network. For an arc (i,j,t), this decision is denoted by a binary variable x_{ij}^{t}. The corresponding vector $\mathbf{x}$ stands for such decisions on all of the arcs. After the first-stage decisions are made, a realization of the uncertain demand is observed. As mentioned in Section 1, we have a limited number of scenarios s for possible future demand, each with a probability of occurrence p_s. These probabilities are non-negative and sum to 1. The collection of these scenarios is denoted by $\mathcal{S}$. In Fig. 2, there are 8 scenarios altogether. The demand of commodity k in scenario s is represented by d_s^k and the vector $\mathbf{d}^s$ denotes the demand of all types of commodities in that scenario.

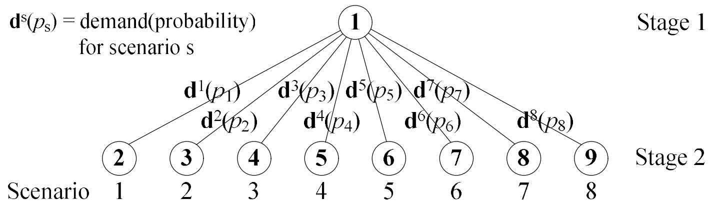

Fig. 2. Two-stage scenario tree describing the decision-making process in SSND.

In the second stage, LTL carriers determine the flow of commodities based on the demand realizations. We use the decision variable y_{ijk}^{st} to represent the flow of commodity k on arc (i,j,t) in scenario s. In addition, we assume that customer demand must be met. To deal with the situations where the demand exceeds the capacity of the first-stage service network, LTL carriers need the flexibility to outsource some of the orders to external suppliers. Let the decision variable $Z^s(k)$ denote the amount of outsourcing for commodity k in scenarios s, whereas λ stands for the cost of outsourcing one unit of the commodity.

With this notation, stochastic service network design can then be formulated as follows (Bai et al., 2014). We denote this model as *SSND* for later use.

$$\textbf{(SSND):} \min\left\{\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{t=0}^{T-1} c_{ij}x_{ij}^{t} + \lambda\sum_{s\in\mathcal{S}} p_s Q_1\left(\mathbf{x},\mathbf{d}^s\right)\right\} \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{N}} x_{ji}^{t^-} = \sum_{j\in\mathcal{N}} x_{ij}^{t} \quad \forall i\in\mathcal{N},\forall t\in\mathcal{T} \tag{2}$$

$$x_{ij}^{t} \in \{0,1\} \quad \forall i,j\in\mathcal{N},\forall t\in\mathcal{T} \tag{3}$$

where

$$Q_1\left(\mathbf{x},\mathbf{d}^s\right) = \min\sum_{k\in\mathcal{K}} Z^s(k) \tag{4}$$

$$\sum_{k\in\mathcal{K}} y_{ijk}^{st} \le u_{ij}x_{ij}^{t} \quad \forall i,j\in\mathcal{N},\forall t\in\mathcal{T},\forall i\ne j \tag{5}$$

$$-\sum_{j\in\mathcal{N}} y_{jik}^{st^-} + \sum_{j\in\mathcal{N}} y_{ijk}^{st} = \begin{cases} d_s^k - Z^s(k) & \text{if } (i,t) \text{ is supply node for } k \\ -d_s^k + Z^s(k) & \text{if } (i,t) \text{ is demand node for } k \\ 0 & \text{otherwise} \end{cases} \quad \forall i\in\mathcal{N},\forall t\in\mathcal{T},\forall k\in\mathcal{K} \tag{6}$$

$$y_{ijk}^{s\tau(k)} = 0 \quad \forall i,j\in\mathcal{N},\forall k\in\mathcal{K} \tag{7}$$

$$y_{ijk}^{st} \ge 0 \quad \forall i,j\in\mathcal{N},\forall k\in\mathcal{K},\forall t\in\mathcal{T} \tag{8}$$

$$Z^s(k) \ge 0 \quad \forall k\in\mathcal{K} \tag{9}$$

This formulation based on the scenario tree representation of the demand uncertainty is referred to as the extensive form (Crainic et al., 2014) or deterministic equivalent (Boland et al., 2018), which is essentially a deterministic model. The objective function (1) minimizes the cost of constructing the service network, plus the expected outsourcing cost across all of the scenarios. Constraint (2) is the design balance constraint (Pedersen et al., 2009), which ensures that the number of incoming vehicles is equal to that of outgoing vehicles at each node in Fig. 1. Constraint (3) enforces the binary restrictions on the design variables. The objective function (4) minimizes the outsourcing cost for a given scenario. Constraint (5) ensures that the total flow along each arc does not exceed its capacity. Constraint (6) makes sure that all customer demand is met, that is, all commodities are shipped from their origins to the intended destinations. Constraint (7) ensures that a commodity must not flow past its delivery deadline. Constraints (8) and (9) are used to guarantee non-negativity of the decision variables.

3. Soft Clustering-based Scenario Bundling

3.1. Motivations

The extensive form of stochastic service network design typically contains so many decision variables that it cannot be directly solved by existing commercial solvers within acceptable computational times (Crainic et al., 2014). Fortunately, the scenario tree representation of the demand uncertainty gives the extensive form a scenario-wise decomposable structure. The standard progressive hedging algorithm takes advantage of this special structure to decompose the extensive form into a set of easily solvable single-scenario subproblems. Fig. 3 illustrates the scenario-wise decomposition of the scenario tree in Fig. 2. To perform the scenario-wise decomposition, copies x_{ij}^{ts} of the design variable x_{ij}^{t} are introduced for each scenario, that is

$$x_{ij}^{t} = \sum_{s \in \mathcal{S}} p_s x_{ij}^{ts} \tag{10}$$

The copies of the design variable should be identical, that is,

$$x_{ij}^{tm} = x_{ij}^{tn}, \forall m, n \in \mathcal{S}, m \neq n \tag{11}$$

This is referred to as the non-anticipativity constraints (NACs) (Watson & Woodruff, 2011), which require that, if two scenarios are indistinguishable up to a specific time, then the decisions made before this time must be identical. After substituting (10) into (1) and rearranging, both the objective function and constraints are separable with respect to scenarios, except for the newly added NACs. The standard progressive hedging algorithm utilizes the augmented Lagrangian relaxation to incorporate the NACs into the objective function, so that the resulting model decomposes by scenarios. A detailed treatment of the decomposition is given in Section 4.

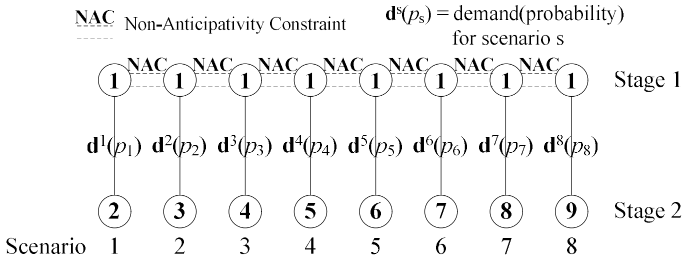
NAC Non-Anticipativity Constraint
$\mathbf{d}^s(p_s)$ = demand(probability) for scenario s
NAC
Stage 1
Stage 2
$\mathbf{d}^1(p_1)$ $\mathbf{d}^2(p_2)$ $\mathbf{d}^3(p_3)$ $\mathbf{d}^4(p_4)$ $\mathbf{d}^5(p_5)$ $\mathbf{d}^6(p_6)$ $\mathbf{d}^7(p_7)$ $\mathbf{d}^8(p_8)$
Scenario 1 2 3 4 5 6 7 8

Fig. 3. Scenario-wise decomposition.

One limitation of the scenario-wise decomposition is that the number of subproblems increases with the size of the scenario tree, which can make the NACs harder to satisfy and hence result in poor computational performance of the standard progressive hedging algorithm. The number of subproblems can be significantly reduced by combining individual scenarios into bundles and decomposing the extensive form according to scenario bundles into multi-scenario subproblems, but at the cost of increasing the size of each subproblem. Fig. 4 illustrates the bundle-wise decomposition for the scenario tree in Fig. 2.

The eight scenarios in Fig. 2 are grouped into three disjoint bundles in Fig.4. The scenarios belonging to the same bundle share common first-stage design variables. In this way, the NACs among the scenarios of a bundle are implicitly implemented and only the NACs among different bundles need to be explicitly considered. Compared to the scenario-wise decomposition in Fig.3, the bundle-wise decomposition in Fig.4 reduces the number of subproblems from eight to three, with the number of NACs decreasing from seven to two. The benefits of bundle-wise decomposition usually far outweigh the increased difficulty of solving each subproblem (Escudero et al., 2013; Crainic et al., 2014; Ryan et al., 2016), provided that multi-scenario subproblems remain readily manageable.

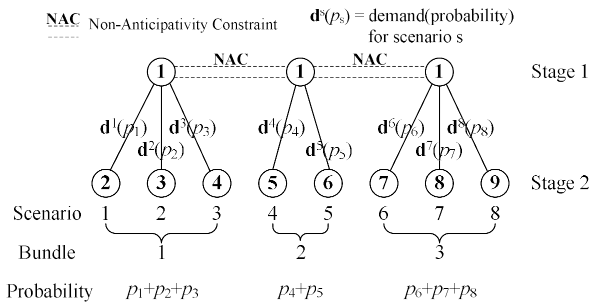
NAC Non-Anticipativity Constraint
$\mathbf{d}^s(p_s)$ = demand(probability) for scenario s
NAC
NAC
Stage 1
Stage 2
$\mathbf{d}^1(p_1)$ $\mathbf{d}^2(p_2)$ $\mathbf{d}^3(p_3)$ $\mathbf{d}^4(p_4)$ $\mathbf{d}^5(p_5)$ $\mathbf{d}^6(p_6)$ $\mathbf{d}^7(p_7)$ $\mathbf{d}^8(p_8)$
Scenario
Bundle
Probability
$p_1+p_2+p_3$
p_4+p_5
$p_6+p_7+p_8$

Fig. 4. Bundle-wise decomposition using disjoint scenario bundles.

In order to perform bundle-wise decomposition, we need to develop a method for grouping scenarios into bundles. As mentioned in Section 1, almost all scenario bundling strategies developed so far assume before the scenario-to-bundle assignment that each scenario belongs to exactly one bundle. For example, the k-means-based scenario bundling strategy assigns each scenario to its closest bundle center according to the Euclidean distance (Crainic et al., 2014). In fact, this form of scenario bundling, where a scenario is assumed to either belong to a bundle completely or not, falls within the class of hard clustering in the field of pattern recognition and machine learning. From a mathematical perspective, the membership relation between a scenario and a bundle is described by a Boolean value 0 or 1 (i.e., False or True) in hard clustering-based scenario bundling.

However, the Boolean assumption about the bundle membership for every scenario may turn out to be inappropriate for those scenarios that have some degree of uncertainty in their bundle membership. There may be some scenarios that lie roughly midway between the centers of several bundles. Logically, such a scenario would have approximately the same degree of membership in these bundles. Although there is no clear preference of one bundle over the other in such cases, the hard clustering-based scenario bundling would assign such a scenario to only one bundle. Therefore, it seems reasonable to soften the Boolean membership relation and estimate the probability of a scenario belonging to a certain bundle.

Moreover, with the Boolean membership relation replaced by a probabilistic one, a scenario is allowed to belong to multiple bundles, which could have a beneficial effect on the convergence rate of the progressive hedging algorithm. If two bundles have a few scenarios in common, the difference between the solutions of the corresponding multi-scenario subproblems is expected to be smaller by comparison with the case of disjoint bundles in hard clustering-based scenario bundling. The smaller difference between the first-stage decisions will normally lead to a faster convergence of the progressive hedging algorithm (Crainic et al., 2014). The bundle-wise decomposition based on overlapping scenario bundles for the scenario tree in Fig. 2 is illustrated in Fig. 5. In contrast with Fig.4, scenario 3, 4 and 5 belong to two bundles whereas scenario 6 has membership in three bundles.

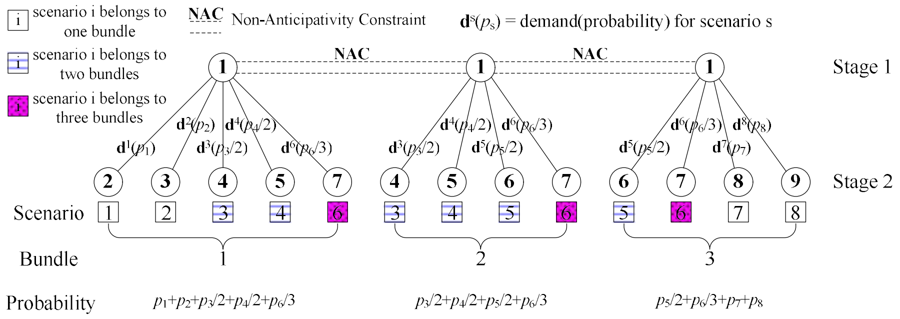
scenario i belongs to one bundle
scenario i belongs to two bundles
scenario i belongs to three bundles
NAC Non-Anticipativity Constraint
$\mathbf{d}^s(p_s)$ = demand(probability) for scenario s
NAC
NAC
Stage 1
Stage 2
$\mathbf{d}^1(p_1)$ $\mathbf{d}^2(p_2)$ $\mathbf{d}^3(p_3/2)$ $\mathbf{d}^4(p_4/2)$ $\mathbf{d}^5(p_5/2)$ $\mathbf{d}^6(p_6/3)$ $\mathbf{d}^7(p_7)$ $\mathbf{d}^8(p_8)$
Scenario
Bundle
Probability
$p_1+p_2+p_3/2+p_4/2+p_6/3$
$p_3/2+p_4/2+p_5/2+p_6/3$
$p_5/2+p_6/3+p_7+p_8$

Fig. 5. Bundle-wise decomposition using overlapping scenario bundles.

In addition, the existing scenario bundling strategies take no account of the covariances, which represent the geometric features of the scenario bundles, that is, the shape and orientation of a confidence ellipsoid drawn over a scenario bundle (see Fig. 6 for an illustrative example). For example, the k-means-based scenario bundling strategy does not estimate the covariances of the scenario bundles but only the bundle centers. In fact, it assumes that all ellipsoids of the scenario bundles have the same shape and orientation and differ from each other only in bundle centers (Bishop, 2006). By adopting a probabilistic approach, the covariances can be explicitly considered and the scenario bundles are allowed to have different covariance structures. Thus, scenario bundles with different shapes and orientations of ellipsoids can be better identified rather than erroneously merged into a single bundle.

These potential benefits motivate us to move from the hard clustering-based scenario bundling approaches to a probabilistic treatment, in which the bundle membership is described by probabilities (i.e., values between 0 and 1) instead of Boolean values (i.e., 0 and 1). We shall refer to scenario bundling based on probabilistic membership as soft clustering-based scenario bundling, as opposed to hard clustering-based scenario bundling rested upon Boolean membership.

3.2. Methodologies

In soft clustering-based scenario bundling, a scenario is assumed to have partial membership in each of the bundles, rather than full membership of a certain bundle in the case of hard clustering-based scenario bundling. More specifically, a fractional membership score is computed to measure the degree to which a scenario belongs to a bundle. The membership scores for all of the scenarios across all of the bundles can be organized into a matrix $\mathbf{\Delta}$, with the element δ_{sb} of row s and column b representing the degree of membership for scenario s in bundle b. Each membership score is less than or equal to 1 and the sum of the membership scores of a scenario is equal to 1, that is,

$$\delta_{sb} \in [0,1] \quad \forall s \in \mathcal{S}, \forall b \in \mathcal{B} \tag{12}$$

$$\sum_{b \in \mathcal{B}} \delta_{sb} = 1 \quad \forall s \in \mathcal{S} \tag{13}$$

Here $\mathcal{B}$ represents the set of scenario bundles b and thus $b \in \mathcal{B}$. The method for determining the number $|\mathcal{B}|$ of scenario bundles will be discussed later. Various soft clustering algorithms in the field of pattern recognition and machine learning can be applied to calculate the membership scores. In order to investigate the impacts of overlapping scenario bundles and covariance structures on the computational performance of the progressive hedging algorithm, we consider fuzzy c-means (Bezdek, Ehrlich & Full, 1984) and Gaussian mixture models (Bishop, 2006) in this paper, which will be further discussed later. It is worth noting that the resulting membership scores offer the flexibility to construct not only overlapping scenario bundles, but also disjoint bundles, whereas the Boolean membership relation allows disjoint bundles only. For example, the disjoint bundles can be obtained simply by assigning a scenario to the bundle with the highest membership score.

Having found the membership scores, we need a strategy to determine the scenario-to-bundle assignment. It is straightforward to define a membership score threshold and assign a scenario to those bundles in which the membership score of this scenario is greater than the specified threshold. However, the fixed and global threshold strategy suffers from severe limitations in practical applications. For example, some scenarios may not be assigned to any bundle when their maximum membership scores are lower than the threshold.

To correct this deficiency, we propose a scenario-dependent strategy that rests on the maximum membership score of a scenario. Let $h(s)$ denote the maximum membership score of a scenario s. We define a parameter η called the interval coefficient, which satisfies $\eta \in [0,1]$, and assign a scenario to those bundles in which its membership scores fall within the interval $[\eta * h(s), h(s)]$. Thus, each scenario will be assigned to at least one bundle. Compared with the hard clustering-based scenario bundling approach, our proposed strategy is better suited to the two types of scenarios discussed below. Some scenarios lie much closer to the center of a particular bundle than to any other center. For such a scenario, it is clear that the assignment to the nearest bundle center is the most appropriate. With our proposed strategy, such a scenario is assigned to exactly the nearest bundle because it has a much higher membership score for the closest bundle than the other bundles. This result is proved in **Proposition 1**. In addition, there may be other scenarios that lie in between the centers of the bundles and somewhat far away from the center of any bundle. It seems reasonable to assign such a scenario to multiple bundles. Since the largest membership scores are very close in magnitude, our proposed strategy assigns such a scenario to the corresponding bundles as expected.

Proposition 1. *Let* s *be a scenario and* η *the interval coefficient. If the maximum membership score of* s *is greater than* $1/(\eta+1)$, *then* s *would be assigned to exactly one bundle and this bundle has the highest membership score of* s.

Proof. See **Appendix A**.

In our proposed strategy, a scenario may appear multiple times in the resulting bundles. We borrow the mathematical term 'multiplicity' from the multiset theory (Blizard, 1989) to refer to the number of times that a scenario appears in the scenario bundles. If a scenario occurs only once, it has multiplicity one. On the other hand, the probabilities of each scenario bundle must sum to one regardless of the scenario-to-bundle assignment. Therefore, we cannot simply add together the probabilities of each scenario within a bundle to calculate the probability of this scenario bundle. To address this problem, we first count the total number of times a scenario appears in all final bundles to obtain the multiplicity of each scenario. Then the original probability of a scenario is divided by its multiplicity and we obtain the new probability of a scenario. As illustrated in Fig. 5, the original probabilities of scenario 3, 4 and 5 are divided by two while that of scenario 6 is divided by three. The probability of a scenario bundle is found by summing the new probabilities of each scenario within this bundle. It is worth noting that our approach to calculating the probability of a scenario bundle equally applies to the case of disjoint scenario bundles where the multiplicity of each scenario is one.

To summarize, the soft clustering-based scenario bundling is accomplished through three phases, which are shown in **Algorithm 1**. In the first phase, we calculate the bundle membership scores of each scenario using fuzzy c-means or Gaussian mixture models. In the following phases, we assign each scenario to the bundles based on their membership scores and calculate the probabilities of the resulting bundles.

Algorithm 1: Soft Clustering-based Scenario Bundling

```
Inputs:   the set $\mathcal{S}$ of scenarios $s$, the probability of each scenario $p_s$, the number of bundles $g$,
          the interval coefficient $\eta$.
Outputs: the set $\mathcal{B}$ of bundles $b$, the probability of each bundle $p_b$.
Begin
Phase 1: Membership score calculation.
1: Calculate the membership scores with fuzzy c-means or Gaussian mixture models.
Phase 2: Scenario-to-bundle assignment.
2: for each scenario $s \in \mathcal{S}$ do
3:      Find its maximum membership score $h(s)$.
4:      Assign $s$ to the bundles for which the corresponding membership scores fall within the interval
        $[\eta * h(s), h(s)]$.
5: end for
Phase 3: Bundle probability calculation.
6: Count the total number of times a scenario $s$ appears in all bundles to obtain its multiplicity.
7: Compute the new probability $q_s$ of $s$ through dividing its original probability $p_s$ by its multiplicity.
8: Calculate the probability of a bundle $p_b$ according to
   $p_b = \sum_{s \in b} q_s, \quad \forall b \in \mathcal{B}$
End
```

Before we perform the scenario bundling algorithm, we need to specify the scenario features based on which scenario similarity or dissimilarity is measured. A straightforward yet effective choice is the commodity demand (Crainic et al., 2014). In this paper, we choose the $|\mathcal{K}|$-dimensional vector $\mathbf{d}^s$, which represents the demand volumes of all commodities in scenario s, as the feature of a scenario s.

In addition to the scenario features, the soft clustering-based scenario bundling algorithm requires one to specify the number of scenario bundles beforehand. Although there exist various algorithms for determining the optimal number of clusters in the field of pattern recognition, almost all evaluation criteria for the quality of a clustering intrinsically amount to the clustering goal, that is, attaining high intra-cluster similarity and low inter-cluster similarity. For example, the well-known elbow method (Ketchen & Shook, 1996) considers the clustering objective as a function of the number of clusters and examines different numbers of clusters in ascending order to choose the one at which the graph of this function flattens markedly. However, an appropriate number of clusters in terms of such criteria does not necessarily translate into effectiveness for some later purpose. Actually, we are more concerned with how well different numbers of scenario bundles serve the downstream purpose of improving the computational performance of the progressive hedging algorithm, than with the scenario bundling itself. Therefore, we rely on this downstream purpose to provide an evaluation metric for choosing the number of scenario bundles. In view of this, we determine the number of clusters by rounding the square root of the number of scenarios to the nearest integer, that is, $\left\lceil \sqrt{|\mathcal{S}|} \right\rceil$ (resp. $\left\lfloor \sqrt{|\mathcal{S}|} \right\rfloor$) when the fractional part of the square root is greater (resp. less) than 0.5. In the case where the square root has a fractional part of exactly 0.5, we round it to the integer with larger magnitude. The rationale behind our proposed method is to balance the number and the size of multi-scenario sub-problems. With a larger number of clusters, the number of subproblems will increase accordingly, whereas the size of subproblems will generally decrease. As mentioned earlier, the increase in the number of sub-problems will generally make the NACs harder to satisfy and hence result in slow convergence of the progressive hedging algorithm. If the size of sub-problems is too small, we will not be able to take full advantage of the computing power of the computers. On the other hand, the size of the sub-problems would be too large to manage with a smaller number of clusters. Therefore, the computational performance of the progressive hedging algorithm is expected to be better when a proper balance between the number and the size of sub-problems is achieved.

3.3. Membership score calculation

3.3.1. Fuzzy c-means & Degree of overlap

Fuzzy c-means determines the membership scores by minimizing an objective function that is the sum of distance between a scenario and a bundle center weighted by that scenario's membership score over all possible scenario and bundle center pairs (Bezdek et al., 1984; Bai et al., 2016). The membership score matrix $\mathbf{\Delta}$ is referred to as the fuzzy partition matrix in fuzzy c-means. Let $\mathbf{v}_b$ represent the center of a bundle b and $\mathbf{V}$ the corresponding set of bundle centers. This objective function denoted by $J(\mathbf{\Delta},\mathbf{V})$ is given as follows.

$$J(\mathbf{\Delta},\mathbf{V}) = \sum_{s\in\mathcal{S}}\sum_{b\in\mathcal{B}} (\delta_{sb})^m \left\|\mathbf{d}^s - \mathbf{v}_b\right\|^2 \tag{14}$$

Here $m(m>1)$ is the exponent for the fuzzy partition matrix, which controls how much the resulting bundles can overlap with one another. A higher value of the exponent generally leads to a greater degree of overlap between the resulting bundles.

The most common method for solving the minimization problem (14) is the alternating optimization algorithm (Havens et al., 2012), the pseudocode of which is given in **Appendix B**. The alternating optimization algorithm is known to converge for any exponent $m\in(1,+\infty)$. As m goes to infinity, δ_{sb} approaches $1/g$ (see step 5 of **Algorithm B1**). This implies that each scenario has an equal chance of belonging to every bundle if m approaches infinity. On the other hand, if m approaches 1, the membership score of a scenario approaches 1 for its closest bundle and 0 for all other bundles. Therefore, if m approaches 1, the probabilistic membership scores become Boolean values and each scenario belongs to exactly one bundle, which corresponds to hard clustering-based scenario bundling method such as k-means.

We measure the degree of overlap by the overlap ratio, which is defined as the sum of the cardinality of each scenario bundle divided by the cardinality of the whole scenario set, that is, $\left(|b_1|+|b_2|+\cdots+|b_{|\mathcal{B}|}|\right)/|\mathcal{S}|$. For disjoint scenario bundles without overlap, the overlap ratio is therefore equal to one. In the case of overlapping scenario bundles, the overlap ratio would be greater than one and a larger quantity or higher multiplicity of repeated scenarios would lead to a greater value of the overlap ratio.

3.3.2. Gaussian mixture models & Covariance structures

Gaussian mixture models determine the membership scores by maximizing the log of the likelihood function, which is given by

$$l(\Phi,\boldsymbol{\mu},\boldsymbol{\Sigma})=\sum_{s\in\mathcal{S}}\log\left\{\sum_{b\in\mathcal{B}}\phi_b\mathcal{N}\left(\mathbf{d}^s\mid\boldsymbol{\mu}_b,\boldsymbol{\Sigma}_b\right)\right\} \tag{15}$$

Here, ϕ_b is the mixing coefficient which satisfies the constraint $0\le\phi_b\le 1$, together with $\sum_{b\in\mathcal{B}}\phi_b=1$. The corresponding set of ϕ_b is denoted by Φ. For a $|\mathcal{K}|$-dimensional vector $\mathbf{d}^s$, the multivariate Gaussian distribution takes the form

$$\mathcal{G}\left(\mathbf{d}^s\mid\boldsymbol{\mu}_b,\boldsymbol{\Sigma}_b\right)=\frac{1}{(2\pi)^{|\mathcal{K}|/2}}\frac{1}{\left(\det(\boldsymbol{\Sigma}_b)\right)^{1/2}}\exp\left\{-\frac{1}{2}\left(\mathbf{d}^s-\boldsymbol{\mu}_b\right)^{\top}\boldsymbol{\Sigma}_b^{-1}\left(\mathbf{d}^s-\boldsymbol{\mu}_b\right)\right\} \tag{16}$$

where $\boldsymbol{\mu}_b$ is a $|\mathcal{K}|$-dimensional mean vector, $\boldsymbol{\Sigma}_b$ is a $|\mathcal{K}|\times|\mathcal{K}|$ covariance matrix and $\det(\boldsymbol{\Sigma}_b)$ denotes the determinant of $\boldsymbol{\Sigma}_b$. The corresponding sets of $\boldsymbol{\mu}_b$ and $\boldsymbol{\Sigma}_b$ are denoted by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively. The membership score of a scenario s in the bundle b is then obtained by substituting the resulting parameters, namely $\boldsymbol{\mu}_b$, $\boldsymbol{\Sigma}_b$ and ϕ_b, into

$$\delta_{sb}=\frac{\phi_b\mathcal{N}\left(\mathbf{d}^s\mid\boldsymbol{\mu}_b,\boldsymbol{\Sigma}_b\right)}{\sum_{b\in\mathcal{B}}\phi_b\mathcal{N}\left(\mathbf{d}^s\mid\boldsymbol{\mu}_b,\boldsymbol{\Sigma}_b\right)} \tag{17}$$

An elegant method for finding the maximum likelihood solutions for the model (15) is the Expectation-Maximization (EM) algorithm (Redner & Walker, 1984; Bishop, 2006), the pseudocode of which is given in **Appendix C**.

One notable feature of Gaussian mixture models is that each component Gaussian density has a covariance matrix. The covariance matrices can be full rank or constrained to be diagonal. We can also specify whether all component Gaussian densities share a common covariance matrix. By contrast, the k-means-based scenario bundling strategy is close to a Gaussian mixture model in which a single covariance matrix is shared by all of the components, and is restricted to be a diagonal matrix with equal diagonal elements (Bishop, 2006). Geometrically, the values of the vector $\mathbf{d}^s$, at which the component Gaussian density (16) yields a constant value, form ellipsoids centered at the corresponding mean vector. The covariance structure determines the shape and orientation of the resulting ellipsoids (Banfield and Raftery, 1993). Consider the example of bivariate Gaussian distributions. For full covariance matrices, the orientation of an ellipse relative to the x - and y - axes can be arbitrary, whereas the major and minor axes of an ellipse are either parallel or perpendicular to the x - and y - axes in the case of diagonal covariance matrices. Furthermore, an ellipse becomes a circle when the corresponding diagonal matrix has equal diagonal elements. If a covariance matrix is shared among all components, all ellipses have the same shape and orientation. Conversely, the shape and orientation of all ellipses might vary if each component has its own unshared covariance matrix.

In comparison with k-means, Gaussian mixture models provide more flexibility by allowing unequal covariance structures and can therefore better identify scenario bundles that have different shapes and orientations. This advantage often translates into a better balance in the size of different scenario bundles. To illustrate this, consider the simulated data points from a mixture of two bivariate Gaussian distributions in Fig. 6, which are partitioned into 2 bundles by k-means and Gaussian mixture models.

The k-means algorithm fails to identify the two bundles corresponding to the two Gaussian components in the mixture models and erroneously merge part of the bundle 2 with bundle 1 into a single large bundle. As a result, the bundle 1 is nearly 4 times as large as bundle 2. By contrast, the difference between the size of the two bundles becomes much smaller and more balanced bundles are obtained when Gaussian mixture models with unshared full covariance matrices are used. Generally, the difficulty of solving a multi-scenario subproblem increases rapidly with the size of the subproblem, because the multi-scenario subproblem as a network design problem is NP-hard (Magnanti and Wong, 1984) and cannot be solved in polynomial time unless P = NP (Impagliazzo et al., 2001). This means that the total solution time for the two subproblems would be much longer in the case of two greatly unbalanced scenario bundles than that in the case of two relatively balanced scenario bundles. Thus, we would expect a reduction in the run-time of the progressive hedging algorithm when the scenario bundling method based on k-means is replaced by that based on Gaussian mixture models.

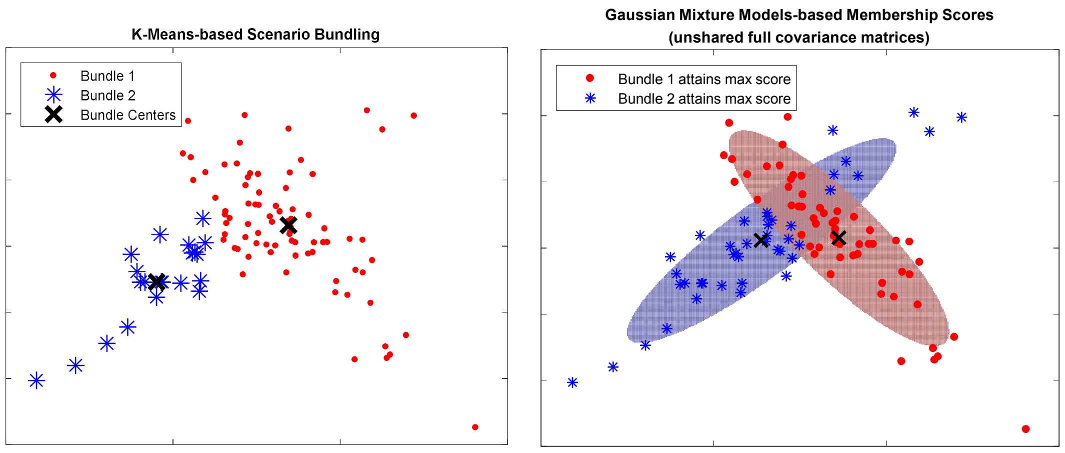

Fig. 6. A comparison of the scenario bundles from K-Means and Gaussian Mixture Models with respect to the extent of variability in the bundle size. In the illustrative example, the 100 two-dimensional data points are partitioned into 2 bundles. In the left plot, the bundle 1 and 2 consist of 79 and 21 data points, respectively. In the right plot, the bundle 1 and 2 attain maximum membership scores on 58 and 42 data points, respectively. We see that the difference between the size of the two bundles is much smaller in the right plot. The two shaded ellipses in the right plot correspond to the confidence regions of two fitted component Gaussian densities with probability threshold 99%. Since unshared full covariance structures are specified here, the two ellipses are arbitrarily oriented.

4. Progressive Hedging Heuristic

4.1. Extension of scenario-wise decomposition to bundle-wise decomposition

Having obtained the scenario bundles, we can now decompose the extensive form of the model *SSND* in a bundle-wise fashion, as is shown in Fig. 4 and 5. To do this, we shall introduce copies x_{ij}^{tb} of the design variable x_{ij}^{t} for each scenario bundle $b \in \mathcal{B}$, that is,

$$x_{ij}^{t} = \sum_{b \in \mathcal{B}} p_b x_{ij}^{tb} \tag{18}$$

The non-anticipativity constraints require that copies of the design variable should be identical, that is,

$$x_{ij}^{tb} = x_{ij}^{tb'}, \forall b, b' \in \mathcal{B}, b \neq b' \tag{19}$$

Substituting (18) into (1), the objective function can be decomposed by scenario bundles and the newly added non-anticipativity constraints are the only constraints that tie together different scenario bundles. Applying Augmented Lagrangian relaxation to the non-anticipativity constraints (Rockafellar & Wets 1991), we obtain an objective function of the form

$$\begin{aligned} \min\Bigg\{ & \sum_{b \in \mathcal{B}} p_b \left(\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{t=0}^{T-1} \left(c_{ij} x_{ij}^{tb} \right) + \lambda \sum_{s \in b} \sum_{k \in \mathcal{K}} \left(\frac{q_s}{p_b} Z^s(k) \right) \right. \\ & \left. + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{t=0}^{T-1} \left(w_{ij}^{tb} x_{ij}^{tb} \right) + \frac{1}{2} \rho \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{t=0}^{T-1} \left(x_{ij}^{tb} - \hat{x}_{ij}^{t} \right)^2 \right) \Bigg\} \end{aligned} \tag{20}$$

where w_{ij}^{tb} are weights related to Lagrange multipliers, $\rho > 0$ is the penalty factor and $\hat{x}_{ij}^{t}$ are given values (see (23)). Because $x_{ij}^{tb} \in \{0,1\}$, it follows that $\left(x_{ij}^{tb}\right)^2 = x_{ij}^{tb}$. By expanding the penalty term in (20), we have

$$\min \sum_{b\in\mathcal{B}} p_b L_b\left(x_{ij}^{tb}, Z^s(k)\right) \tag{21}$$

where

$$\begin{aligned} L_b\left(x_{ij}^{tb}, Z^s(k)\right) = &\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{t=0}^{T-1}\left(c_{ij}x_{ij}^{tb}\right) + \lambda\sum_{s\in b}\sum_{k\in\mathcal{K}}\left(\frac{q_s}{p_b}Z^s(k)\right) + \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{t=0}^{T-1}\left(w_{ij}^{tb}x_{ij}^{tb}\right) \\ &+\frac{1}{2}\rho\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{t=0}^{T-1}x_{ij}^{tb} - \rho\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{t=0}^{T-1}\left(\hat{x}_{ij}^{t}x_{ij}^{tb}\right) + \frac{1}{2}\rho\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N}}\sum_{t=0}^{T-1}\left(\hat{x}_{ij}^{t}\right)^2 \end{aligned} \tag{22}$$

From the procedure above, we see that the extensive form of the model *SSND* is decomposed by scenario bundles and the resulting multi-scenario sub-problems are mixed integer linear programs. It is worth noting that the scenario-wise decomposition in the basic progressive hedging algorithm can be viewed as a special case of the bundle-wise decomposition when the cardinality of each bundle is constrained to be one, or equivalently, the bundle-wise decomposition is a generalization of the scenario-wise decomposition.

With bundle-wise decomposition, the extensive form of the model *SSND* is divided into a system of multi-scenario subproblems. The solutions to these subproblems satisfy all of the constraints for the original two-stage model, referred to as admissible solutions (Listes & Dekker, 2005). A subproblem solution is said to be implementable if it fulfils the non-anticipativity constraints (Guo et al., 2015). However, the subproblem solutions are typically not implementable. Therefore, an approach to enforcing the implementability is needed.

Proposed for scenario-wise decomposition, the basic progressive hedging algorithm can be readily modified to address the implementability issue in bundle-wise decomposition. The implementability is enforced by iteratively aggregating the multi-scenario subproblem solutions into an implementable solution and penalizing the deviation of the subproblem solutions from this implementable solution until the subproblem solutions agree with the implementable solution. More specifically, after each multi-scenario subproblem is solved to optimality, the implementable solution for a first-stage design variable x_{ij}^{tb} is given by

$$\left(\hat{x}_{ij}^{t}\right)^{(r)} = \sum_{b\in\mathcal{B}} p_b\left(x_{ij}^{tb}\right)^{(r)} \tag{23}$$

Here variables with a parentheses-enclosed loop index (r) in the superscript represents their values in the r th iteration.

The deviation of the subproblem solutions from the implementable solution is then used to update the dual variables associated with the non-anticipativity constraints, so that

$$\left(w_{ij}^{tb}\right)^{(r)} = \left(w_{ij}^{tb}\right)^{(r-1)} + \rho\left(\left(x_{ij}^{tb}\right)^{(r)} - \left(\hat{x}_{ij}^{t}\right)^{(r)}\right) \tag{24}$$

The above process is repeated to generate a sequence of implementable solutions that provably converges to the global optimum in the convex case (Lamghari & Dimitrakopoulos, 2016). However, the integrality constraints on the first-stage decision variables render our problem non-convex and convergence is not guaranteed. In the presence of discrete decision variables, computational studies have shown that the progressive hedging algorithm can be used as an effective heuristic method (Watson & Woodruff, 2011; Gade et al., 2016; Boland et al., 2018).

4.2. Algorithmic enhancements

4.2.1. Solution generation

The standard progressive hedging algorithm relies on convergence to provide a solution to the model *SSND*. However, the integrality constraints on the decision variables may sometimes spoil convergence. The cycling behavior in Section 4.2.2 is an obvious example. In the case of non-convergence, the subproblem solutions fail to reach a consensus and the standard algorithm does not output any feasible solution at all, let alone a high-quality one. Crainic et al. (2014) attempted to derive a feasible solution to the stochastic network design problem by finding the union of all subproblem solutions. In other words, as long as an arc is included in any of the subproblem solutions, it will be selected to form a feasible solution. Unfortunately, the design balance constraints (2) of our model are not factored into this solution generation process, and hence the resulting solution can no longer be trusted for our model.

To derive a feasible solution in the case of non-convergence, we note that the solution to a subproblem satisfies all constraints of the original problem and is thus a feasible solution to the original problem. Based on this observation, we develop a simple solution generation method as follows. First of all, a variable is defined to store the incumbent, which is the best solution found so far. At the beginning, there is no incumbent. Then, at each iteration of the standard progressive hedging algorithm, we add a move to compare all of the subproblem solutions with the incumbent. This is done by solving a modification of the original model *SSND*, in which the design variables are fixed at the values they attained in a subproblem solution or the incumbent, and obtaining its optimal objective function value. If any subproblem solution from the current iteration has a better objective function value than the incumbent, we update the incumbent with this subproblem solution. At termination, the algorithm returns the incumbent as the final solution.

It is worth noting that the proposed solution generation process must be performed whether the algorithm converges or not. The reasons for this are twofold. First, in practice, until convergence is achieved, we will not be able to tell if the algorithm will converge. Second, the solution generation process ensures that we can have the best possible solution at termination. In the case of convergence, we see from the above process that the consensus solution among subproblems will also be compared with the incumbent, and so the final solution is at least as good as the consensus solution. Indeed, we observe in our experiments that the incumbent at termination is sometimes not the consensus solution among subproblems, but rather some subproblem solution during the iterative process. In order to obtain a solution of a higher quality, it is therefore worthwhile to include the solution generation process for the convergence case.

4.2.2. Cycle detection

As mentioned earlier, convergence of the progressive hedging algorithm is not assured due to the presence of integer decision variables. One important case, which arises in our numerical experiments, is that of cycling behavior. Once this happens, the progressive hedging algorithm gets trapped in cycles, continuously repeating the same sequence of implementable solutions and never finding a consensus solution. It is therefore necessary to detect the cycling behavior and stop the subsequent futile iterations.

To detect cycles, Watson and Woodruff (2011) proposed to check the repeated occurrences of the dual variables w_{ij}^{tb}, which are constantly changing before convergence. One major drawback of this method is that the number of the dual variables can be very large for some applications. In the context of stochastic service network design, the total number of the dual variables is given by $|\mathcal{N}|*|\mathcal{N}|*|\mathcal{T}|*|\mathcal{B}|$. A larger size of the vector of w_{ij}^{tb} generally leads to an increased complexity of the reoccurrence check, because element-wise comparisons between two w_{ij}^{tb} vectors are needed.

To resolve this problem, we consider instead the objective function values of each subproblem in one iteration. As with the dual variables, the subproblem objective values are also continually changing during the execution of the algorithm. However, the size of the subproblem objective value vector decreases considerably to $|\mathcal{B}|$ in comparison with that of the w_{ij}^{tb} vector, making the reoccurrence check much easier to implement. We check the reoccurrences at each iteration of the progressive hedging algorithm by performing an element-wise comparison of the subproblem objective value vector from the current iteration with that from each of the previous iterations for equality. A cycle is detected if two vectors are found to be equivalent. Our reoccurrence check method has the advantage of being conceptually simple and easy to implement.

There certainly exist other techniques for checking the reoccurrences, such as Floyd's tortoise and hare algorithm and the hashing scheme (Watson and Woodruff, 2011), but we are not interested in comparisons of these methods here.

4.3. Algorithm pseudocode

The progressive hedging heuristic based on bundle-wise decomposition is summarized in **Algorithm 2**.

Algorithm 2: Bundle-wise decomposition-based progressive hedging heuristic

Inputs: a multiplier θ for penalty factors ρ_{ij}^{tb}, parameters for stopping criteria.

Outputs: the incumbent at termination.

```
Begin
1:  $r \leftarrow 0$, $\left(w_{ij}^{tb}\right)^{(0)} \leftarrow 0$, $\left(\hat{x}_{ij}^{t}\right)^{(0)} \leftarrow 0$.
2:  Change the value of multiplier $\theta$ to zero so that $\rho_{ij}^{tb} = 0$.
3:  for all $b \in \mathcal{B}$ do
4:      $\left(x_{ij}^{tb}\right)^{(0)} \leftarrow \underset{x,Z}{\operatorname{argmin}} L_b\left(x_{ij}^{tb}, Z^s(k) \middle| \left(w_{ij}^{tb}\right)^{(0)}, \left(\hat{x}_{ij}^{t}\right)^{(0)}\right)$
5:      solve a restriction of the original model SSND with additional constraints
        $x_{ij}^{t} == \left(x_{ij}^{tb}\right)^{(0)}, \forall i,j \in \mathcal{N}, \forall t \in \mathcal{T}$.
6:      update the incumbent with this subproblem solution if appropriate.
7:  end for
8:  Restore the value of multiplier $\theta$ so that $\rho_{ij}^{tb} \neq 0$.
9:  Obtain implementable solutions according to (23).
10: for all $b \in \mathcal{B}$ do
11:     update the dual variables according to
        $\left(w_{ij}^{tb}\right)^{(0)} \leftarrow \rho_{ij}^{tb}\left(\left(x_{ij}^{tb}\right)^{(0)} - \left(\hat{x}_{ij}^{t}\right)^{(0)}\right)$
12: end for
13: while stopping criteria are not met do
14:     $r \leftarrow r+1$.
15:     for all $b \in \mathcal{B}$ do
16:         $\left(x_{ij}^{tb}\right)^{(r)} \leftarrow \underset{x,Z}{\operatorname{argmin}} L_b\left(x_{ij}^{tb}, Z^s(k) \middle| \left(w_{ij}^{tb}\right)^{(r-1)}, \left(\hat{x}_{ij}^{t}\right)^{(r-1)}\right)$
17:         solve a restriction of the original model SSND with additional constraints
            $x_{ij}^{t} == \left(x_{ij}^{tb}\right)^{(r)}, \forall i,j \in \mathcal{N}, \forall t \in \mathcal{T}$.
18:         update the incumbent with this subproblem solution if appropriate.
19:     end for
20:     Obtain implementable solutions according to (23).
21:     for all $b \in \mathcal{B}$ do
22:         update the dual variables according to (24).
23:     end for
24: end while
End
```

Note that we adopt the variable-dependent penalty factors proposed by Watson & Woodruff (2011). Rather than a global value ρ, each design variable x_{ij}^{tb} has its own penalty factor ρ_{ij}^{tb}. In the context of stochastic service network design, an effective value of ρ_{ij}^{tb} should be proportional to the fixed cost c_{ij} and so can be written in the form $\rho_{ij}^{tb} = \theta c_{ij}$. Here $\theta(\theta > 0)$ is a constant multiplier used for all ρ_{ij}^{tb}. Note also that, in order to obtain initial subproblem solutions, we temporarily change the value of the multiplier for penalty factors to zero in step 2.

We terminate the iterations based on the heterogeneity tolerance, which specifies the allowed proportion of arcs on which consensus has not been reached. The heterogeneity tolerance is more flexible than the convergence criteria in that the latter is equivalent to zero tolerance for heterogeneity among subproblem solutions. In order to hedge against the risk of non-convergence, we limit the maximum amount of time expended on iterating, as well as the maximum number of iterations. Besides, the iterations will be stopped when cycling behavior is detected.

5. Test Instances and Experimental Settings

5.1. Test instances

We empirically evaluated the various scenario bundling methods on a space-time network adapted from Bai et al. (2014), which consists of 6 terminals numbered consecutively from 1 to 6. As in Bai et al. (2014) and Lium et al. (2009), we use a fully connected network with no hub-and-spoke structure as the underlying network. The fixed costs for all of the arcs are arranged in matrix form, with the element of row i and column j representing the fixed charge of opening an arc between terminal i and j. To provide a diverse and representative set of test instances, we split the six terminals into three groups, namely $\{\{1,2\},\{3,4\},\{5,6\}\}$, and constructed two types of cost matrices, that is,

Type 1: Equal within-group and between-group costs.
Type 2: Low within-group costs while high between-group costs.
These two types of cost matrices are shown in Appendix D. As we shall see later in Section 6, among all of the test instances, those with Type 1 cost matrix are the relatively difficult ones, while those with Type 2 cost matrix are the relatively easy ones.

The planning horizon considered in this experiment is divided into 5 time periods and the transport movement between every pair of terminals takes one period. From the perspective of network scale, the experimental space-time network, therefore, consists of 30 nodes and 180 arcs.

Through the underlying space-time network, there is a set of 12 commodities to be delivered, but the volumes of these commodities vary stochastically. For each type of cost matrix, we created two commodity sets. The distributional properties of the uncertain demand are given in terms of marginal distributions for each commodity, as well as correlations among the different commodities. Analogous to the procedure in Lium et al. (2009) and Bai et al. (2014), the demand uncertainty in our experiment is described by the symmetric triangular distribution Tri(4, 12, 8) (min, max, mode) and three types of correlation settings, which include (1) all of the commodities are positively correlated, with the correlation coefficients taking on values 0.4 or 0.7, (2) a mixture of positively and negatively correlated commodities, with the correlation coefficients set at 0.5 and -0.5 respectively, (3) all of the commodities are uncorrelated. For simplicity's sake, these three correlation types are sequentially denoted by capital letters C, M and U, which will be used later in the instance identifiers.

Given the statistical description of an uncertain demand, we employ the publicly available scenario generator from Høyland et al. (2003) to construct scenario trees of varying size to approximate this uncertain demand. The scenario generator is based on the moment-matching algorithm, where scenarios are generated to match the first four marginal moments (i.e., the expected values, standard deviation, skewness and kurtosis) and the correlation matrix of the given uncertain demand. In our experiment, we generated 20, 30 and 40 scenarios for each uncertain demand. More details about scenario generation and evaluation can be found in the articles by Høyland et al. (2003) and Kaut & Wallace (2007).

By combining the three correlation types and three different sizes of the scenario tree, together with the two types of cost matrices and their associated commodity sets, we constructed 36 test instances. To further diversify the testbed, we constructed three larger instances by increasing the number of scenarios to 100. While the test instances are small in terms of the network size, we deliberately chose them because all associated multi-scenario subproblems can be solved to optimality by CPLEX V12.6.2 within several hours, thereby reducing the effect of subproblem solution quality and providing a clear understanding of the impact of scenario bundling on PHH performance. For the same reason, we chose a relatively small value (i.e., 0.5%) as the relative MIP gap tolerance of CPLEX, as we shall see shortly.

To differentiate between these instances, we define an instance identifier, which contains three dot-delimited fields. The instance identifier starts with a letter indicating the correlation type, followed by a positive integer specifying the number of scenarios. The second field represents the type index of cost matrices, whereas the last one stands for the index of the commodity set. For example, the identifier 'C100.1.2' represents the test instance where commodity set 2, whose uncertain demand is represented by 100 positively correlated scenarios, need to be shipped through the underlying network with type 1 cost matrix.

5.2. Experimental settings

On the test instances, we compared the progressive hedging heuristic based on the two soft clustering algorithms (i.e., FCM/GMM) with that based on the hard clustering algorithm (i.e., K-Means, or KM) and with the extensive formulation solved directly using CPLEX. We also benchmarked the soft clustering-based scenario bundling against the cover method proposed by Crainic et al. (2014), which represents the state-of-the-art scenario bundling algorithm for stochastic service network design. We chose K-Means as the hard clustering-based scenario bundling method because it is one of the most popular clustering algorithms in cluster analysis and was used in several papers on scenario bundling (e.g., Crainic et al., 2014; Deng et al., 2017). More importantly, it has been demonstrated by Crainic et al. (2014) that the K-Means-based scenario bundling method outperformed random bundling, which was another prevalent method for scenario bundling in the current literature (e.g., Escudero et al., 2013; Gade et al., 2016). Further details about the K-Means algorithm can be found in (Lloyd, 1982) and (Arthur & Vassilvitskii, 2007). We implemented the four scenario bundling algorithms in MATLAB R2015b, making use of the predefined functions for Fuzzy C-Means and Gaussian Mixture Models to estimate membership scores. Since the four scenario bundling algorithms are random, we repeated each algorithm 20 times for the 36 small instances and four times for the three larger instances, each time using a new set of initial values. Unless otherwise indicated, all of the computing was conducted on a personal computer with eight 2.80 GHz Intel Core i7 CPUs and 8 GB of RAM, under a 64-bit Windows 7 operating system.

The aim of our experiment is twofold. First, we aim to examine the impact of different degrees of overlap on PHH performance. In FCM-based scenario bundling, the amount of overlap between scenario bundles is jointly governed by the interval parameter η and fuzzy partition matrix exponent m. Since a high-valued interval parameter does not encourage overlap between scenario bundles, we consider a fairly small value of 0.4 for the interval parameter. Meanwhile, we varied the exponent to obtain seven different overlap ratios ranging from 1.2 to 2.4. By so doing, we were able to capture a reasonably wide range of performance potential for PHH. We did not consider even larger values of the overlap ratio because they can render the multi-scenario subproblems hard to solve due to a significant increase in the size of subproblems.

The second purpose of the experiments is to demonstrate the benefit of allowing scenario bundles to have different covariance structures. Consequently, we chose unshared full covariance matrices for all mixture components in the Gaussian Mixture Model so that each component had its own non-diagonal covariance matrix which allowed for correlation. At the same time, we attempted to minimize the effects of overlapping scenario bundles by assigning a relatively high value of 0.8 to the interval parameter. According to Proposition 1, the scenarios whose maximum membership scores are greater than 0.56 will then be assigned to only one bundle. Since the maximum membership score of almost every scenario is very close to 1, we actually obtained disjoint scenario bundles on our test instances, just as in the KM-based scenario bundling.

The progressive hedging heuristic based on bundle-wise decomposition was implemented with C++ programming language in Microsoft Visual Studio 2010, with the multi-scenario subproblems modeled by ILOG Concert Technology and solved by CPLEX Mixed Integer Programming Optimizer in version 12.6.2. Except for a relative MIP gap tolerance of 0.5%, all parameters controlling the behavior of CPLEX assume their default values. For all test instances, we set the weight multiplier for the vector of variable-dependent penalty factors equal to 1 when executing the PHH. To check for convergence of the PHH, we took 10^{-5} as the tolerance threshold to determine equality of any two first-stage decisions and 98% as the threshold for the level of consensus among the subproblem solutions. Regarding other termination settings, there are some differences between the 36 small instances and three larger instances. For the 36 small instances, we allow the PHH to run for a maximum of 3 hours in wall clock time or 30 iterations at most, whichever comes first. The maximum allowable number of iterations is set to a small value so that the PHH executes only the early iterations and discontinues the remaining numerous iterations, which play the role of fine-tuning to reconcile the already small discrepancies between subproblem solutions (Watson & Woodruff, 2011). As we have mentioned in Section 4, the best solution is usually found to be some subproblem solution instead of the final consensus solution and thus it does not seem worthwhile continuing the fine-tuning iterations. For the three larger instances, we accordingly set larger tolerances, with the time limit being 10 hours and iteration limit being 100.

6. Experimental Results and Analysis

We solved the problem instances for stochastic service network design using the progressive hedging heuristic based on bundle-wise decomposition and report the optimization results in Table 1-3. For simplicity, the Progressive Hedging Heuristic based on K-Means-, Fuzzy C-Means-, Cover- and Gaussian Mixture Models-based scenario bundling are represented by KM-PHH, FCM-PHH, Cover-PHH and GMM-PHH, respectively. We run each of the four algorithms 20 times for the 36 small problem instances and four times for the three larger instances, with a new scenario bundling result for every trial. To establish a benchmark against which the solution quality of our proposed method can be evaluated, we also solved the extensive formulations of these problem instances directly using CPLEX, in which case CPLEX was executed to reach an optimality gap of 0.5%. For the instances with no more than 40 scenarios, CPLEX can prove optimality within three hours. By contrast, it took more than 12 hours for CPLEX to solve the three 100-scenario instances to optimality. The quality of the solutions produced by the progressive hedging heuristic is then measured against the lower bound yielded by CPLEX and the relative optimality gap is calculated as (Final Objective Value – Lower Bound from CPLEX) / (Final Objective Value). We report the relative optimality gap in percentage with two decimal places, as well as the computing time (rounded to the nearest integer) and the number of iterations (rounded to one decimal place) consumed by the progressive hedging heuristic.

6.1. Impact of Different Degrees of Overlap

To study the impact of different degrees of overlap on the performance of the progressive hedging heuristic, we compare the performance results of FCM-PHH under different overlap ratios, which are shown in Table 1. The value '1' for the overlap ratio represents the scenario bundles obtained from K-Means, and there is no overlap between any two bundles. For the overlap ratio with value '2', we evaluate the scenario bundles obtained not only from FCM, but also from the cover strategy proposed in (Crainic et al., 2014).

Table 1. Performance results of FCM-PHH under different overlap ratios.

Overlap ratio	Small Instance									Larger instance		
	C40.1.2			M40.1.1			U40.1.1			C100.1.2		
	Gap CPLEX LB (%)	#Iter.	Time (s)	Gap CPLEX LB (%)	#Iter.	Time (s)	Gap CPLEX LB (%)	#Iter.	Time (s)	Gap CPLEX LB (%)	#Iter.	Time (s)
1(KM)	0.61	19.5	2741	0.56	12.7	1853	0.52	7.3	1491	0.60	51.5	15340
1.2	0.66	15	2601	0.56	9.4	1638	0.53	8.1	1563	0.57	25.3	9900
1.4	0.61	17.2	2790	0.58	9.2	1762	0.55	6.2	1403	0.60	32.3	11080
1.6	0.63	13	2523	0.50	7.3	1533	0.51	5.1	**1367**	0.53	18.5	**8300**
1.8	0.57	9.5	**2276**	0.50	6	**1477**	0.50	5.3	1429	0.53	15.3	9012
2	0.57	9.7	2503	0.50	6	1695	0.50	3	1386	0.52	14.5	10122
2(Cover)	0.57	9.5	2472	0.50	6	1608	0.50	3	1395	0.52	14.8	10386
2.2	0.57	6	3158	0.50	6	2165	0.50	3	1831	0.52	12	14563
2.4	0.57	6	4136	0.50	4	3072	0.50	3	2455	0.52	12	20328

'Cover' represents the cover strategy proposed for scenario bundling in (Crainic et al., 2014). Gap CPLEX LB is the relative optimality gap against the lower bound produced by CPLEX. Given an instance and an overlap ratio, the average performance of FCM-PHH over 20 (resp. 4) trials is reported for small (resp. larger) instances, with every trial using a new scenario bundling result. The shortest run-time is highlighted in boldface.

Let us consider first the solution quality and the number of iterations. It can be seen from Table 1 that, when the overlap ratio increases from 1 to 1.6, both the optimality gap and the number of iterations fluctuate. However, when the overlap ratio is greater than 1.6, the optimality gap becomes smaller and the number of iterations reduces dramatically in comparison with the case of small overlap ratios. Therefore, a high degree of overlap helps improve the solution quality and reduce the number of iterations required by FCM-PHH to achieve convergence. This tendency is consistently observed on the other test instances, whose results are not displayed here due to limited space.

We next look at the run-time of FCM-PHH. The impact of degree of overlap can be seen more clearly by plotting the run-time against the overlap ratio, as shown in Fig. 7. Here, we consider the larger instance C100.1.2 as an example. It can be seen that, as with the number of iterations, the run-time fluctuates dramatically when the overlap ratio rises from 1 to 1.6. However, different from the number of iterations, the run-time increases sharply when the overlap ratio goes above 2. For the overlap ratios between 1.6 and 2, FCM-PHH consumes much less run-time in comparison with the case of small and large overlap ratios in question.

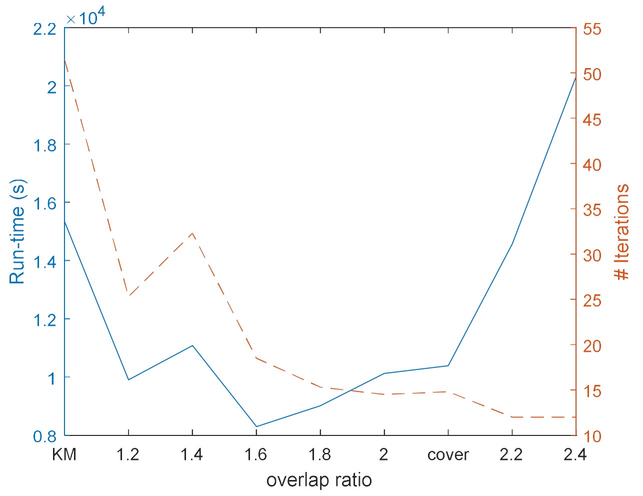

Fig. 7. The run-time and number of iterations consumed by FCM-PHH under different overlap ratios on the larger instance C100.1.2.

The general trend of the run-time demonstrates that there is a trade-off between the decreased number of iterations and the increased difficulty in solving each subproblem. Although a high degree of overlap may well lead to a reduction in the number of iterations, allowing some overlap between scenario bundles typically results in a greater size of each bundle, thus increasing the difficulty of solving each multi-scenario subproblem. When the overlap ratio is greater than 2, the advantage of fewer iterations is outweighed by the disadvantage of increased difficulty in solving each subproblem, leading to a substantial growth in the run-time eventually. By contrast, the advantage of fewer iterations dominates when the overlap ratio is between 1.6 and 2. For small values of the overlap ratio less than 1.6, the degree of overlap is too small to produce a definite advantage.

Considering both the solution quality and the run-time, the overlap ratios between 1.6 and 2 give relatively good performance among all the overlap ratios in question. However, the best choice for the overlap ratio depends on the problem instances, as can be seen in Table 1. For the sake of comparison with KM-PHH and GMM-PHH, we report in Table the performance results of FCM-PHH under the overlap ratio with value 1.8, in light of the observation that this overlap ratio produces good performance on most instances.

6.2. Impact of Covariance Structures

In order to investigate whether there is any advantage in specifying different covariance structures for scenario bundles, we then make a comparison between the results of GMM-PHH and KM-PHH, which are shown in Table 2 and 3. To save some space, we also put the performance results of Cover-PHH and FCM-PHH in these tables for later use in Section 6.3. It can be seen from Table 2 and 3 that there is no clear winner between KM-PHH and GMM-PHH in terms of solution quality. On some test instances, GMM-PHH yielded lower optimality gaps than KM-PHH, whereas KM-PHH gave smaller optimality gaps on other test instances. In terms of the number of iterations, we see from Table 2 and 3 that GMM-PHH is slightly better than KM-PHH. On all of the problem instances, GMM-PHH performed fewer iterations than KM-PHH, but the difference between these two methods is not big, except for two 100-scenario instances. When it comes to time consumption, GMM-PHH beats KM-PHH by a considerable margin. On all of the problem instances, GMM-PHH consumed much less time than KM-PHH did. Averaging the time ratios of GMM-PHH to KM-PHH, we find that GMM-PHH provides a significant saving of 26 percent on time for the 36 small instances and 54 percent for the three larger instances.

Table 2. Small instances: performance results of KM-, GMM-, Cover- and FCM-PHH on 36 instances with no more than 40 scenarios.

Small Instances		Gap CPLEX LB (%)				# Iterations				Time (s)			
Cost Matrix	# Scen.	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH
	40	0.54	0.59	0.52	0.52	11.9	7.6	4.8	4.9	1931	1411	1602	1528
Type 1	30	0.59	0.72	0.58	0.57	10.4	7.3	5.3	5.5	1199	840	912	889
	20	0.53	0.67	0.51	0.51	8.0	6.1	4.7	4.9	508	458	483	472
	40	0.57	0.50	0.50	0.50	8.3	4.5	3.6	3.9	1233	887	996	935
Type 2	30	0.52	0.51	0.50	0.50	6.2	5.5	4.2	4.2	977	673	799	739
	20	0.64	0.60	0.58	0.57	5.8	4.9	3.1	3.4	710	477	485	483

Gap CPLEX LB is the relative optimality gap against the lower bound produced by CPLEX. For each method, its average performance over the six instances having the same cost matrix and number of scenarios is reported, since the performance results on these instances are comparable.

Table 3. Larger instances: performance results of KM-, GMM-, Cover- and FCM-PHH on three 100-scenario instances.

Larger Instances	Gap CPLEX LB (%)				# Iterations				Time (s)			
	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH
C100.1.2	0.60	0.59	0.56	0.56	51.5	21.5	18.3	19.5	15,340	6324	8317	7631
M100.1.1	0.58	0.50	0.50	0.50	48.5	19.5	4	4	17,635	5920	7938	6182
U100.1.1	0.54	0.55	0.52	0.51	16.3	12.5	5.8	6.1	8722	5407	6287	5933

Gap CPLEX LB is the relative optimality gap against the lower bound produced by CPLEX. Given an instance and a method, the average performance over 4 runs of this method is reported, with every trial using a new scenario bundling result.

We can gain some insight into the run-time advantage of GMM-PHH by comparing the results of GMM- and KM-based scenario bundling. Both algorithms grouped similar scenarios together into one bundle while separating different bundles as well as possible and yielded up disjoint bundles on the test instances. However, a careful comparison of the scenario bundles from the two algorithms shows that there is a big difference in variability of bundle sizes. Take the case of 30 positively correlated scenarios for example. The sizes of the five scenario bundles from the first run of the KM- and GMM-based scenario bundling algorithm are $\{12, 1, 3, 2, 12\}$ and $\{3, 5, 8, 8, 6\}$, respectively. Obviously, the values in the set $\{12, 1, 3, 2, 12\}$ are widely scattered, while the set $\{3, 5, 8, 8, 6\}$ has a small dispersion of values.

To measure the relative size of the scenario bundles in relation to the whole scenario set, we applied the economic term 'concentration ratio' to the disjoint scenario bundles. The concentration ratio, expressed as a percentage, is calculated as the sum of the number of scenarios in the two largest scenario bundles divided by the size of the whole scenario set. It ranges from $2/|\mathcal{B}|$ to 100% and a lower value indicates a better balance in the size of different scenario bundles. We also measure how far the bundle sizes lie apart by the standard deviation and the range, which indicates how far from the average size each scenario bundle is and the difference between the largest and smallest sizes, respectively. The descriptive statistics of the KM- and GMM-based bundling results are displayed in Table 4.

Let us consider again the example of 30 positively correlated scenarios (i.e., scenario set 'C30'). It can be seen from Table 4 that the two largest scenario bundles account for nearly 80% of all scenarios on average in the 20 trials of KM-based scenario bundling, whereas the average concentration ratio in the case of GMM-based scenario bundling falls by one fourth to less than 60%. In terms of the range and standard deviation, the average difference between bundle sizes in KM-based scenario bundling is about twice as big as that in GMM-based scenario bundling. Similar results are consistently observed in other test instances.

Table 4. Descriptive statistics of the KM- and GMM-based scenario bundling results on the 12 scenario sets.

Scenario Set	KM-based scenario bundling			GMM-based scenario bundling		
	Concentration Ratio (%)	Range	Standard Deviation	Concentration Ratio (%)	Range	Standard Deviation
C40	65.00	14.45	5.77	51.25	8.65	3.40
C30	77.83	11.55	5.57	58.00	6.95	2.86
C20	90.00	10.00	4.90	65.50	5.20	2.30
M40	60.13	14.50	5.31	46.75	6.90	2.57
M30	75.50	14	5.92	55.67	6.30	2.58
M20	73.50	6.55	3.04	63.50	4.20	1.88
U40	52.63	9.75	3.73	49.63	7.55	3.01
U30	61.33	8.80	3.57	50.83	5.00	1.95
U20	66.50	5.35	2.45	61.50	3.85	1.70
C100	50.00	29.50	9.38	35.50	17.50	5.49
M100	48.00	23.50	8.77	28.00	10.50	3.84
U100	38.50	19.50	5.94	28.50	11	3.42

For each scenario set with no more than 40 scenarios and with 100 scenarios, the three descriptive statistics are averaged over 20 runs and 4 runs of the scenario bundling algorithm, respectively.

These results demonstrate that KM-based scenario bundling is prone to produce unbalanced scenario bundles, where there is a high concentration of scenarios in one or two bundles, while the other bundles contain very few scenarios. By contrast, the scenarios are more evenly distributed among the bundles in GMM-based scenario bundling and the resulting bundles are comparable in size. Recall that GMM provides more flexibility than KM by allowing arbitrary covariance structures for scenario bundles, so we might expect GMM to better accommodate scenario bundles with different correlation structures and thus to make the sizes of the resulting bundles more balanced.

When both the algorithms assign every scenario uniquely to one bundle, the result of KM-based scenario bundling would probably contain several much larger bundles, giving rise to some much harder subproblems. For GMM-based scenario bundling, all resulting bundles are most likely to be in relatively modest size, rendering all subproblems easier to solve. As a result, KM-PHH consumes more time than GMM-PHH, as we have seen in Table 2 and 3.

6.3. KM-PHH vs. Cover-PHH vs. FCM-PHH vs. GMM-PHH

In this section, we compare the performance results of KM-, Cover-, FCM- and GMM-PHH, which are shown in Table 2 and 3. We first examine the solution quality. Analyzing the relative optimality gaps presented in Table 2 and 3, we observe that there is little difference between Cover-PHH and FCM-PHH, with the maximum difference being 0.01%. Also, there is no clear winner between KM-PHH and GMM-PHH, as mentioned in Section 6.2. Among the four methods in question, Cover- and FCM-PHH outperform KM- and GMM-PHH on almost all test instances in terms of solution quality. This observation reinforces our previous findings in Section 6.1 that a high degree of overlap between scenario bundles helps improve the solution quality. However, the improvement in the solution quality is marginal. For example, the difference between the optimality gaps of FCM-PHH and GMM-PHH is no greater than 0.16% on all test instances. In other words, all four methods (i.e., KM-, Cover-, FCM- and GMM-PHH) found high-quality solutions to the test instances, with the maximum optimality gap being 0.64%, 0.58%, 0.57% and 0.72% respectively.

We look further into the relationship between the consensus solution among subproblems and the best solution found by the progressive hedging heuristic. Surprisingly, we observe from Table 5 that the best solution may not be the final consensus solution, but some subproblem solution during the iterative process. This is the case with more than one third of the total runs for 36 small instances and approximately half of the total runs for 3 larger instances. This observation demonstrates the important role of the newly added step in finding high-quality solutions, where we update the incumbent at each iteration by comparing it with each of the subproblem solutions. Also, we observe from Table 5 that the first appearance of the best solution is not very close to the PHH termination point, but takes approximately 90% of the total run-time for 36 small instances and about 80% for 3 larger instances. These two observations suggest that we don't have to wait until all subproblem solutions reach full agreement to provide a high-quality solution. Instead, we can stop the iterations in our proposed method earlier than in the standard progressive hedging algorithm, thereby reducing the number of iterations and the run-time without a decline in the solution quality.

Table 5. The relationship between PHH best solution and PHH consensus solution.

Instances	PHH consensus solution is ***not*** PHH best solution (percentage of total runs)				First appearance of PHH best solution (percentage of total run-time)			
	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH	KM-PHH	GMM-PHH	Cover-PHH	FCM-PHH
36 small instances	47.22%	41.67%	37.67%	36.11%	88.38%	91.37%	94.26%	92.63%
3 larger instances	66.67%	66.67%	41.67%	41.67%	73.12%	77.05%	86.73%	83.56%

We next consider the run-time and the number of iterations. As can be seen from Table 2 and 3, KM-PHH yielded the worst performance among the four methods, requiring a much longer run-time and more iterations than other methods on every test instance. Compared to the case of KM-PHH, the number of iterations needed by Cover-PHH reduces by 58% on average for 36 small instances and 74% for 3 larger instances. Indeed, Cover-PHH required the smallest number of iterations among the four methods. Regarding the run-time, Cover-PHH consumed on average 81% of the run-time of KM-PHH on 36 small instances and 57% on 3 larger instances. In contrast with Cover-PHH, FCM-PHH required slightly more iterations, but consumed less run-time. The run-time advantage is most easily seen on 3 larger instances, where FCM-PHH saved on average 12% of the run-time of Cover-PHH. Among the four methods, GMM-PHH consumed the least amount of run-time. On 36 small instances, GMM-PHH expended on average 74% of the KM-PHH run-time, 91% of the Cover-PHH run-time and 95% of the FCM-PHH run-time. The run-time advantage is even greater on 3 larger instances. Compared to KM-, Cover- and FCM-PHH, GMM-PHH provides on average a significant time saving of 54%, 21% and 10%, respectively.

Considering both the solution quality and the run-time, GMM-PHH yielded the best performance among the four methods, achieving nearly equivalent solution quality in a fraction of the run-time of KM-, Cover- and FCM-PHH.

7. Conclusions and Future Directions

We have presented a soft clustering-based scenario bundling method and modified the standard progressive hedging algorithm accordingly to incorporate the resulting scenario bundles. The proposed method differs from existing scenario bundling strategies in that the bundle membership is described by probabilities (i.e., values between 0 and 1) instead of Boolean values (i.e., 0 and 1), thereby bringing with it many advantages. We have empirically studied its impacts on the computational performance of a progressive hedging heuristic in the context of stochastic service network design by benchmarking it against k-means-based scenario bundling and the cover method, which represents the state-of-the-art scenario bundling algorithm for stochastic service network design.

We can draw the following conclusions from the computational evidence. Firstly, a moderate degree of overlap (with the overlap ratio between 1.6 and 2) helps improve the solution quality and reduce the run-time and number of iterations required by the progressive hedging heuristic. However, the best choice for the overlap ratio depends on the problem instances. More often, the overlap ratio around 1.8 yields up relatively good performance. Secondly, compared to k-means, Gaussian mixture models save on the average more than 20 percent of the run-time of the progressive hedging heuristic, since the advantage of allowing scenario bundles to have different covariance structures enables Gaussian mixture models to produce more balanced scenario bundles. Lastly, the progressive hedging heuristic based on Gaussian mixture models yields up the best performance among all methods in question, since it generally achieves nearly equivalent solution quality in a fraction of the run-time of the other methods.

For future research, the proposed method can be extended in two directions. First, the fact that soft clustering-based scenario bundling has proven beneficial in the context of two-stage stochastic mixed integer programming bodes well for its potential effects in other contexts. It is therefore natural to consider the extension of soft clustering-based scenario bundling to multi-stage stochastic programs or chance-constrained programs. Secondly, the progressive hedging heuristic has a natural advantage of being easily adaptable to parallel implementation, since the multi-scenario subproblems can be solved independently. Compared to the serial implementation in our experiments, parallel implementation has the potential to decrease the run-time considerably. Thus, the development of parallel approaches looks promising for future research.

Acknowledgements

This work is supported by the National Natural Science Foundation of China [grant number 71471092, 72071116], the Zhejiang Natural Science Foundation [grant number LR17G010001] and the Ningbo Science and Technology Bureau [grant numbers 2014A35006, 2017D10034, 2019B10026]. We are also grateful to the financial support from the International Doctoral Innovation Centre.

Declarations of interest

None.

Appendix A. Proof of proposition 1

Proposition 1. *Let* s *be a scenario and* η *the interval coefficient. If the maximum membership score of* s *is greater than* $1/(\eta+1)$*, then* s *would be assigned to exactly one bundle and this bundle has the highest membership score of* s.

Proof. We first want to show that there is a single bundle that has the highest membership score of s. Suppose for the sake of contradiction that there exist two or more scenario bundles that has the highest membership scores of s. Let $h(s)$ be the maximum membership score of s. By the definition of the interval coefficient, $\eta \in [0,1]$ and we have $0.5 \le 1/(\eta+1) \le 1$. If $h(s) > 1/(\eta+1)$, then $h(s) > 0.5$. From the supposition, it follows that the sum of the membership scores of s is greater than 1. But this contradicts the fact that the membership scores of a scenario across all bundles must sum to 1. Hence our supposition is incorrect and there is only one bundle that has the highest membership score of s.

We now want to show that s would be assigned to the bundle with the highest membership score of s. By the definition of our algorithm, s would be assigned to the bundles for which the corresponding membership scores fall within the interval $\left[\eta * h(s),\, h(s)\right]$. Let $h'(s)$ be the second maximum membership score of s. Since the membership scores of a scenario across all scenario bundles sums to 1, it follows that $h(s)+h'(s)\le 1$. If $h(s)>1/(\eta+1)$, then $h'(s)<\eta/(\eta+1)$, and thus $h'(s)<\eta h(s)$, since $\eta h(s)>\eta/(\eta+1)$. Therefore, among all membership scores of s, $h(s)$ is the only one that falls within the interval $\left[\eta * h(s),\, h(s)\right]$. We then conclude that s would be assigned to the bundle with the highest membership score of s. This completes the proof of the proposition. □

Appendix B. Membership score calculation based on fuzzy c-means

The pseudocode of membership score calculation based on fuzzy c-means is given in **Algorithm B1**. The equations in step 4 and 5 are necessary conditions for the objective function (14) to reach its minimum.

Algorithm B1: Membership score calculation based on fuzzy c-means

```
Inputs: the set of scenarios $\mathcal{S}$, the number of bundles $g$, the exponent for fuzzy partition matrix $m$.
Outputs: the fuzzy partition matrix $\mathbf{\Delta}$.
Begin
1: Initialize the matrix $\mathbf{\Delta}$ with uniformly generated random numbers between 0 and 1.
2: Normalize each entry in $\mathbf{\Delta}$ through dividing it by the sum of the corresponding column.
3: while stopping criteria are not met do
4:     Calculate the bundle centers according to
           $\mathbf{v}_b = \sum_{s\in\mathcal{S}} (\delta_{sb})^m \mathbf{d}^s \Big/ \sum_{s\in\mathcal{S}} (\delta_{sb})^m, \quad \forall b \in \mathcal{B}$
5:     Update the membership scores according to
           $\delta_{sb} = \dfrac{\left(1\big/\left\|\mathbf{d}^s - \mathbf{v}_b\right\|^2\right)^{\frac{1}{m-1}}}{\sum_{l=1}^{g}\left(1\big/\left\|\mathbf{d}^s - \mathbf{v}_l\right\|^2\right)^{\frac{1}{m-1}}}, \quad \forall s \in \mathcal{S}, \ \forall b \in \mathcal{B}$
6:     Calculate the objective function $J(\mathbf{\Delta}, \mathbf{V})$.
7: end while
End
```

Appendix C. Membership score calculation based on Gaussian mixture models

The pseudocode of membership score calculation based on Gaussian mixture models is given in **Algorithm C1**. For the initialization in step 1, we select several scenarios as the initial mean vectors using the k-means++ algorithm, details of which can be found in (Arthur and Vassilvitskii, 2007). All of the initial covariance matrices are diagonal, with the diagonal entries being the variances of the corresponding commodity demand. The initial mixing coefficients are chosen to be the uniform probability.

Algorithm C1: Membership score calculation based on Gaussian mixture models

```
Inputs: the set of scenarios $\mathcal{S}$, the number of bundles $g$.
Outputs: the membership score matrix $\mathbf{\Delta}$.
Begin
1: Initialize the mean vectors $\boldsymbol{\mu}_b$, covariance matrices $\boldsymbol{\Sigma}_b$ and mixing coefficients $\phi_b$.
2: while convergence criteria are not met do
```

```
3:      Compute the membership scores $\delta_{sb}$ according to (17).
4:      Update the parameters $\boldsymbol{\mu}_b$, $\boldsymbol{\Sigma}_b$ and $\phi_b$ according to
          $\boldsymbol{\mu}_b^{new} = \sum_{s\in\mathcal{S}} \delta_{sb}\mathbf{d}^s \Big/ \sum_{s\in\mathcal{S}} \delta_{sb}$
          $\boldsymbol{\Sigma}_b^{new} = \sum_{s\in\mathcal{S}} \delta_{sb}\left(\mathbf{d}^s - \boldsymbol{\mu}_b^{new}\right)\left(\mathbf{d}^s - \boldsymbol{\mu}_b^{new}\right)^T \Big/ \sum_{s\in\mathcal{S}} \delta_{sb}$
          $\phi_b^{new} = \sum_{s\in\mathcal{S}} \delta_{sb} \Big/ |\mathcal{S}|$
5:      Calculate the log likelihood $\ell(\Phi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$.
6: end while
End
```

Appendix D. The parameters for the test instances

Table D1. The parameters for the test instances. In each cost matrix, the main diagonal element in the i th row represents the fixed cost of holding service at terminal i, while the off-diagonal element in the i th row and j th column represents the fixed cost of transportation service from terminal i to j.

Parameters	Values	Type 1 cost matrix (c_{ij})						Type 2 cost matrix (c_{ij})					
$\lvert\mathcal{N}\rvert$	6	50	100	100	100	100	100	50	100	250	250	250	250
T	5	100	50	100	100	100	100	100	50	250	250	250	250
$\lvert\mathcal{K}\rvert$	12	100	100	50	100	100	100	250	250	50	100	250	250
u	20	100	100	100	50	100	100	250	250	100	50	250	250
λ	100	100	100	100	100	50	100	250	250	250	250	50	100
#scenarios	20/30/40	100	100	100	100	100	50	250	250	250	250	100	50

References

Ahmed S., Luedtke J., Song Y., Xie W., 2017. Nonanticipative duality, relaxations, and formulations for chance-constrained stochastic programs. Mathematical Programming 162(1), 51–81.

Arthur D., Vassilvitskii S., 2007. K-Means++: The advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), January 07-09. New Orleans, Louisiana, pp. 1027–1035.

Bai C., Dhavale D., Sarkis J., 2016. Complex investment decisions using rough set and fuzzy c-means: An example of investment in green supply chains. European Journal of Operational Research 248(2), 507–521.

Bai R., Kendall G., Li J., 2010. An efficient guided local search approach for service network design problem with asset balancing. In: Proceedings of the International Conference on Logistics Systems and Intelligent Management (ICLSIM 2010), January 09-10. Harbin, China, pp. 110-115.

Bai R., Wallace S.W., Li J., Chong A.Y.-L., 2014. Stochastic service network design with rerouting. Transportation Research Part B 60, 50-65.

Bakir I., Boland N., Dandurand B., Erera A., 2020. Sampling Scenario Set Partition Dual Bounds for Multistage Stochastic Programs. INFORMS Journal on Computing 32(1), 145–163.

Banfield, J. D., Raftery, A. E., 1993. Model-based gaussian and non-gaussian clustering. Biometrics 49, 803-821.

Beier E., Venkatachalam S., Corolli L., Ntaimo L., 2015. Stage- and scenario-wise Fenchel decomposition for stochastic mixed 0-1 programs with special structure. Computers & Operations Research 59, 94-103.

Bezdek J. C., Ehrlich R., Full W., 1984. FCM: The fuzzy c-means clustering algorithm, Computers & Geosciences, 10(2-3), 191–203.

Birge J. R., Louveaux F., 2011. Introduction to Stochastic Programming. 2nd ed. Springer, New York.

Bishop C. M., 2006. Pattern Recognition and Machine Learning. Springer-Verlag, Berlin.

Blizard W. D., 1989. Multiset theory. Notre Dame Journal of Formal Logic 30 (1), 36-66.

Boland N., Christiansen J., Dandurand B., Eberhard A., Linderoth J., J. Luedtke, 2018. Combining progressive hedging with a frank-wolfe method to compute lagrangian dual bounds in stochastic mixed-integer programming. SIAM Journal on Optimization 28(2), 1312–1336.

Boland N., Hewitt M., Marshall L., Savelsbergh M., 2017. The continuous-time service network design problem. Operations Research 65(5), 1303-1321.

Carpentier P.-L., Gendreau M., Bastin F., 2013. Optimal scenario set partitioning for multistage stochastic programming with the progressive hedging algorithm. Available at: http://www.optimization-online.org/DB_FILE/2013/10/4065.pdf (Accessed: 27 May 2018).

Crainic T. G., Hewitt M., Rei W., 2014. Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. Computers & Operations Research 43, 90–99.

Crainic T. G., 2000. Service network design in freight transportation. European Journal of Operational Research 122(2), 272–288.

Deng Y., Ahmed S., Lee J., Shen S., 2017. Scenario grouping and decomposition algorithms for chance-constrained programs. Available at: http://www.optimization-online.org/DB_FILE/2017/02/5853.pdf (Accessed: 27 December 2018).

Erera A., Hewitt M., Savelsbergh M., Zhang Y., 2013. Improved load plan design through integer programming based local search. Transportation Science 47(3), 412-427.

Escudero L. F., Garín M. A., Pérez G., Unzueta A., 2013. Scenario cluster decomposition of the lagrangian dual in two-stage stochastic mixed 0-1 optimization. Computers & Operations Research 40(1), 362–377.

Escudero L. F., Garín M. A., Unzueta A., 2016. Cluster Lagrangean decomposition in multistage stochastic optimization. Computers & Operations Research 67, 48–62.

Gade D., Hackebeil G., Ryan S. M., Watson J.-P., Wets R. J.-B., Woodruff D. L., 2016. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. Mathematical Programming 157(1), 47–67.

Guo G., Hackebeil G., Ryan S. M., Watson J.-P., Woodruff D. L., 2015. Integration of progressive hedging and dual decomposition in stochastic integer programs. Operations Research Letters 43(3), 311–316.

Gupta A., Maranas C. D., 2003. Managing demand uncertainty in supply chain planning. Computers & Chemical Engineering 27(8), 1219–1227.

Havens T. C., Bezdek J. C., Leckie C., Hall L. O., Palaniswami M., 2012. Fuzzy c-means algorithms for very large data. IEEE Transactions on Fuzzy Systems 20(6), 1130–1146.

Hewitt, M., Crainic, T. G., Nowak, M., & Rei, W., 2019. Scheduled service network design with resource acquisition and management under uncertainty. Transportation Research Part B: Methodological 128, 324-343.

Høyland K., Kaut M., Wallace S.W., 2003. A heuristic for moment-matching scenario generation. Computational Optimization and Applications 24(2), 169—185.

Høyland K., Wallace S.W., 2001. Generating scenario trees for multistage decision problems. Management Science 47(2), 205—336.

Impagliazzo R., Paturi R., Zane F., 2001. Which problems have strongly exponential complexity? Journal of Computer and System Sciences, 63(4), 512–530.

Jarrah A., Johnson E., Neubert L., 2009. Large-scale, less-than-truckload service network design. Operations Research 57(3), 609–625.

Jiang X., Bai R., Atkin J., Kendall G., 2017. A scheme for determining vehicle routes based on arc-based service network design. Information Systems and Operational Research 55(1), 16—37.

Kaut, M., Wallace, S.W., 2007. Evaluation of scenario-generation methods for stochastic programming. Pacific Journal of Optimization 3(2), 257—271.

Ketchen D., Shook C., 1996. The application of cluster analysis in strategic management research: an analysis and critique. Strategic Management Journal 17(6), 441—458.

King A. J., Wallace S. W., 2012. Modeling with Stochastic Programming: Springer Series in Operations Research and Financial Engineering. Springer, New York.

Lamghari A., Dimitrakopoulos R., 2016. Progressive hedging applied as a metaheuristic to schedule production in open-pit mines accounting for reserve uncertainty. European Journal of Operational Research 253(3), 843–855.

Laporte G., Louveaux F. V., 1993. The integer L-shaped method for stochastic integer programs with complete recourse. Operations Research Letters, 13(3), 133–142.

Lindsey K., Erera A., Savelsbergh M., 2016. Improved integer programming-based neighborhood search for less-than-truckload load plan design. Transportation Science, 50(4), 1360-1379.

Listes O., Dekker R., 2005. A scenario aggregation–based approach for determining a robust airline fleet composition for dynamic capacity allocation. Transportation Science 39(3), 367–382.

Lium A.-G., Crainic T. G., Wallace S. W., 2009. A study of demand stochasticity in service network design. Transportation Science 43(2), 144–157.

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Transactions on Information Theory 28(2), 129—137.

Løkketangen A., Woodruff D. L., 1996. Progressive hedging and tabu search applied to mixed integer (0,1) multistage stochastic programming. Journal of Heuristics 2(2), 111–128.

Magnanti T.L., Wong R.T., 1984. Network designs and transportation planning: models and algorithms. Transportation Science, 18, 1–55.

Pedersen M. B., Crainic T. G., Madsen O. B. G., 2009. Models and tabu search metaheuristics for service network design with asset-balance requirements. Transportation Science 43(2), 158–177.

Redner R., Walker H., 1984. Mixture densities, maximum likelihood and the EM algorithm. SIAM Review, 26(2), 195–239.

Rockafellar R. T., Wets R. J.-B., 1991. Scenario and policy aggregation in optimization under uncertainty. Mathematics of Operations Research 16, 119–147.

Ryan K., Ahmed S., Dey S. S., Rajan D., 2016. Optimization driven scenario grouping. Available at: http://www. optimization-online.org/DB_FILE/2016/03/5366.pdf (Accessed: 27 May 2018).

Santoso T., Ahmed S., Goetschalckx M., Shapiro A., 2005. A stochastic programming approach for supply chain network design under uncertainty. European Journal of Operational Research 167(1), 96-115.

Wang X., Crainic T. G., Wallace S. W., 2019. Stochastic network design for planning scheduled transportation services: the value of deterministic solutions. INFORMS Journal on Computing 31(1), 153-170.

Watson J.-P., Woodruff D. L., 2011. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. Computational Management Science 8(4), 355–370.

Watson J.-P., Woodruff D. L., Hart W. E., 2012. PySP: modeling and solving stochastic programs in Python. Mathematical Programming Computation 4(2), 109–149.

Wieberneit N., 2008. Service network design for freight transportation: A review. OR Spectrum 30(1), 77-112.

Zenarosa G. L., Prokopyev O. A., Schaefer A. J., 2014. Scenario-tree decomposition: bounds for multistage stochastic mixed-integer programs. Available at: http://www.optimization-online.org/DB_FILE/2014/09/4549.pdf (Accessed: 27 June 2018).