Scenario Grouping in a Progressive Hedging-Based Meta-Heuristic for Stochastic Network Design

Teodor Gabriel Crainic
Mike Hewitt
Walter Rei

August 2013

CIRRELT-2013-52

Scenario Grouping in a Progressive Hedging-Based Meta-Heuristic for Stochastic Network Design[†]

Teodor Gabriel Crainic[1,*], Mike Hewit[3] , Walter Rei[1]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

[2] Department of Information Systems & Operations Management, Quinlan School of Business, Loyola University Chicago， 1 E. Pearson St., Maguire Hall, Chicago, Il 60611, U.S.A.

Abstract. We propose a methodological approach to build strategies for grouping scenarios as defined by the type of scenario decomposition, type of grouping, and the measures specifying scenario similarity. We evaluate these strategies in the context of stochastic network design by analyzing the behavior and performance of a new progressive hedging-based meta-heuristic for stochastic network design that solves subproblems comprised of multiple scenarios. We compare the proposed strategies not only among themselves, but also against the strategy of grouping scenarios randomly and the lower bound provided by a state-of-the-art MIP solver. The results show by solving multi-scenario subproblems generated by the strategies we propose, the meta-heuristic produces better results in terms of solution quality and computing efficiency than when either single-scenario subproblems or multiple-scenario subproblems that are generated by picking scenarios at random are solved. The results also show that, considering all the strategies tested, the covering strategy with respect to commodity demands leads the highest quality solutions and the quickest convergence.

Keywords: Stochastic programs, network design, progressive hedging, scenario clustering, machine learning.

Acknowledgements. Partial funding for this project has been provided by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds de recherche du Québec – Nature et technologies (FRQNT). This support is gratefully acknowledged.

[†]Revised version of the CIRRELT-2012-41.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1 Introduction

Network design models define an important class of combinatorial optimization problems with a wide gamut of applications. These problems naturally appear in various forms in the planning of complex systems, e.g., logistics, transportation and telecommunications, at the strategic, tactical and operational levels. In such contexts, network design models are used to produce plans that define the structure, services, allocation of resources, or adjustments to be applied to the respective networks. Such plans are then used for varying periods of time depending on the decision level considered. In the case of either strategic or tactical planning, decisions are made for relatively long periods of time. Therefore, managers responsible for such planning decisions generally face uncertainty (i.e., stochastic parameters) at the moment when plans are being drawn.

Demand usually entails a certain level of uncertainty for network design problems, defining stochastic parameters relative to, e.g., origins, destinations, or demand volumes. In this paper, we assume origins and destinations are known, but volumes are uncertain. To account for such uncertainty, forecasting is traditionally used to obtain estimates in replacement of stochastic parameters. Managers may also apply some form of sensitivity analysis to provide alternative plans and networks. This type of approach can lead to arbitrarily bad solutions, however (Wallace, 2000). Moreover, recent studies have shown that cost-effective networks obtained in stochastic settings are structurally different than the ones obtained in deterministic settings (Lium et al., 2009; Thapalia et al., 2012).

Stochastic programming has become the methodology of choice to properly account for uncertainty in planning problems. The goal of stochastic programming approaches for network design is to build a single design that remains cost-effective when different demand realizations are encountered. To do so, uncertainty in demand is typically modeled with a finite set of scenarios, which must be generated with care to ensure that they, collectively, closely approximate the uncertainty in the planning setting (e.g., Crainic et al., 2011b). Once the appropriate set of scenarios is generated, a two-stage stochastic network design problem is generally solved, the first stage modeling the choice of design (e.g., selection of transportation services and schedules to operate during the next season or of components to include in the logistics network), the second modeling its cost-effectiveness in servicing the demand (e.g., routing) for each scenario. See Klibi et al. (2010) for a thorough review of such models.

Even though such problems have recently received increased attention from the scientific community, they remain notoriously hard to solve. The reasons for this are that, on the one hand, deterministic network design problems are NP-Hard in all but trivial cases and, thus, formulations of even moderate size are generally difficult to address and, on the other hand, modeling uncertainty with scenarios generally yields very large instances. Thus, much like with the deterministic case, we need heuristic methods for producing high-quality designs for realistically-sized instances, and we need efficient methodology to address problem settings involving large numbers of scenarios.

Strategies decomposing the set of scenarios have been used to address these challenges and build solution methods for stochastic combinatorial optimization problems. Meta-heuristics based on this approach and the progressive hedging idea (Rockafellar and Wets, 1991) have proved computationally successful in several such settings, including stochastic network design (e.g., Crainic et al., 2011b; Haugen et al., 2001; Løkketangen and Woodruff, 1996). Yet, the instances addressed are still of relatively modest dimensions, as the scenario decomposition used by these meta-heuristics generated single-scenario subproblems, each a NP-Hard deterministic network design formulation.

Revisiting the scenario-decomposition strategy by grouping scenarios to yield multi-scenario subproblems thus appears methodologically interesting as it could reduce significantly the number of subproblems, at the price of an increase in the difficulty of addressing each subproblem. The interest is also motivated by recent studies showing that exact methods using scenario decomposition to address integer stochastic problems can be significantly improved by applying a decomposition strategy based on scenario groups (Escudero et al., 2012, 2013). The general idea was to adapt the Lagrangean relaxation to produce subproblems defined on randomly defined groups of scenarios. Escudero et al. (2013) observed improved lower bounds obtained by solving the associated Lagrange dual problem, and have numerically shown that the computational burden of solving the Lagrange dual problem is also reduced when grouping is used. Although grouping scenarios has clearly proven to be efficient in this case, no studies that we are aware of have focused on how such strategies should be defined and scenario groups be created. Furthermore, no scenario-decomposition strategy based on scenario grouping has yet been applied in the context of meta-heuristics.

The goal of this paper therefore is twofold. First, to propose a systematic methodological approach to build strategies for grouping scenarios as defined by the type of scenario decomposition (partition or cover), type of grouping (similar or dissimilar scenarios), and the measures specifying scenario similarity. Second, to study these strategies experimentally in the context of stochastic network design in order to characterize their-behavior and recommend how to use the proposed methodology. To perform this study, we introduce a new progressive hedging-based meta-heuristic for stochastic network design that solves subproblems comprised of multiple scenarios as defined by the proposed scenario-grouping methodology. Notice that, while the scenario-grouping strategies we propose are derived in the context of addressing stochastic network design problems, they can be applied to other stochastic programs as well.

We evaluate the performance of the proposed strategies for grouping scenarios through the effectiveness of the resulting progressive hedging meta-heuristic. We use a state-of-the-art commercial solver to solve subproblems to optimality. This enables us to focus the analysis on the benefits of solving multi-scenario subproblems and reduce any potential noise that may occur by solving them with a heuristic method. The results show that solving multi-scenario subproblems based on a random partition of scenarios often enables the progressive hedging meta-heuristic to achieve a solution that is 25% better than when single-scenario subproblems are solved, and in fewer than half the iterations. Moreover, the results also show that, compared to grouping scenarios randomly, partitioning scenarios with the grouping strategy we propose can enable the meta-heuristic to obtain a solution that is 16% better. The number of iterations and time required to achieve this result is always reduced, often by half. Finally, the results show that a covering of scenarios enables the meta-heuristic to find solutions that are 16% better than partitioning them. Thus, covering produces an overall improvement in solution quality of approximately 27% with respect to the original strategy of solving single-scenario subproblems.

The rest of this paper is organized as follows. We recall the two-stage formulation of the stochastic network design problem in Section 2 and review the related literature. We analyze the issue of grouping scenarios in Section 3 and introduce the various strategies we propose. We start Section 4 dedicated to the description of the computational experiments we performed by introducing the new progressive hedging-based meta-heuristic that can solve subproblems comprised of multiple scenarios. We then proceed to computationally study the behavior and performance of the proposed methodology for grouping scenarios. We draw conclusions based on these experiments and outline future efforts in in Section 5.

2 A Brief Tour of Stochastic Network Design

We first recall the two-stage formulation of the stochastic network design problem and review the main applications and models present in the literature (Subsection 2.1), and then provide a general description of the algorithmic strategies used to develop the solution methods proposed for these problems (Section 2.2).

2.1 Stochastic Network Design Models

Network design models entail two general groups of decisions: design decisions, that define the structure and characteristics of the network, and flow decisions, which relate to how the network is used to perform the operational activities considered, see Crainic and Laporte (1997). When using the *a priori* approach (Birge and Louveaux, 2011) in a stochastic setting, decisions are made in stages according to when stochastic parameters become known. Problems are therefore formulated by defining which decisions are taken before all information is available (first stage decisions) and which decisions are made afterwards (second stage decisions and onward). Traditionally, in the case of stochastic two-stage network design models, design decisions define the first stage (i.e., the *a priori* solution) and flow decisions the second (i.e., the available recourse), see Klibi et al. (2010).

Formally, given a directed network with node set N, arc set A, commodity set K, and scenario set S, we wish to

$$\min \sum_{(i,j)\in A} f_{ij}y_{ij} + \sum_{s\in S} p_s(\sum_{k\in K}\sum_{(i,j)\in A} c_{ij}^k x_{ij}^{ks})$$

subject to

$$\sum_{j\in N^+(i)} x_{ij}^{ks} - \sum_{j\in N^-(i)} x_{ji}^{ks} = d_i^{ks} \quad \forall i \in N, k \in K, s \in S, \tag{1}$$

$$\sum_{k\in K} x_{ij}^{ks} \leq u_{ij}y_{ij} \quad \forall (i,j) \in A, s \in S, \tag{2}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A, \tag{3}$$

$$x_{ij}^{ks} \geq 0 \quad \forall (i,j) \in A, k \in K, s \in S, \tag{4}$$

where, y_{ij} indicates whether arc (i,j) is installed in the network, f_{ij} is the cost (often called the fixed charge) of doing so, x_{ij}^{ks} is the amount of commodity k's demand that flows on arc (i,j) in the resulting solution for scenario s, and c_{ij}^k is the cost per unit of demand flowed on arc (i,j). Constraints (1) ensure that in each scenario s, each commodity's demand may be routed from its origin node to its destination node. Constraints (2) ensure that the same design is used in each scenario, and that arc capacity (u_{ij}) is never violated. When $d^{ks} < u_{ij}$, the disaggregate inequalities $x_{ij}^{ks} \leq d^{ks}y_{ij}$ can be added to the formulation to strengthen its linear relaxation. We refer to this problem as the $CMND(S)$; its optimal solution is a single design that is cost-effective for all scenarios.

Various applications of stochastic network design models can be found in the literature. Most, but not all, of the existing research may be found in the fields of logistics or telecommunications. In transportation service network design with stochastic demands, two stage formulations decide on the structure of services to offer in the first stage, while the routing of flows is decided at the second stage (e.g., Lium et al., 2009; Crainic et al., 2011b). More complex formulations are encountered in multi-tiered transportation systems, e.g., two-tier City Logistics, where the first stage targets the design of the first tier service network, and the second stage addresses both the design of the corresponding service network and the routing of flows (Crainic et al., 2011a).

In the context of logistics, two-stage models have been proposed to formulate a wide range of planning problems related to the design of multi-echelon supply chain networks under uncertainty, see, e.g., Alonso-Ayuso et al. (2003) and Bidhandi and Yusuff (2011) for thorough studies on the subject of strategical and tactical planning in this context. In such models, the first stage traditionally involves decisions that define the topology and capabilities of the supply chain. Such decisions may include: the choice of facilities (plants, warehouses and distribution centers) (Tsiakis et al., 2001), the selection of machinery to use at processing points (Santoso et al., 2005), the choice of market segments to target (Vila et al., 2007) and the selection of operational processes to apply at each location in the supply chain (Schütz et al., 2009). The second stage variables generally group all flow decisions that define how the supply chain is used to perform a given set of operations (i.e., the supply, production and distribution of commodities) once all stochastic parameters become known. Finally, the objective in such models is usually to minimize the sum of both the total fixed cost incurred given the designed supply chain and the resulting expected total flow cost.

As for applications in telecommunications, stochastic programming models have been proposed to solve a wide gamut of network-related problems, see Gaivoronski (2005) for a thorough review. However, when compared to the supply chain context, fewer design models, where either the size or the structure of the network is changed, have been proposed. This is explained by the fact that in telecommunications applications, it is usually assumed that a physical network already exits and that it cannot be easily expanded or reduced. Stochastic models are therefore used to determine what equipment to install in the network to provide services when given parameters are stochastic (e.g., the demands for the considered services). Many such problems take the form of stochastic location models where in the first stage a series of equipment is installed, while the following stages formulate the operations performed to provide the considered services. The planning of an internet based information service, as presented in Gaivoronski (2005), defines such a problem. Stochastic network models can also be used to solve capacity planning problems under uncertainty. Riis and Andersen (2002) propose a two-stage formulation for such a problem when demands are stochastic. In this case, the first stage decisions determine how much capacity is included on the links of the network and the second stage contains the flow decisions that establish how the information transits in the network to satisfy demands. Finally, specific design models have also been proposed, as in the case of Smith et al. (2004), who developed an integer stochastic programming model for a ring design problem in an optical network. Consider a network, defined here as a set of client nodes, a set of client-pair demand edges and, for each client-pair demand, a set of rings on which the data of the demand can transit. The ring design problem consists in choosing which rings to use in order to satisfy demands which, in this case, are stochastic. A two-stage model is proposed in Smith et al. (2004) to formulate this problem. The first stage decisions define the assignment of the client nodes to rings that are then used in the second stage to transit the demands. In the model proposed in Smith et al. (2004), it is assumed that all demands are not necessarily satisfied (a penalty being applied for unfulfilled demands). Therefore, the second stage decisions are defined as the proportions for each demand that are either satisfied or not.

2.2 Solution Approaches

The solution methods developed to address stochastic network design problems generally share two common features. First, as is traditionally done in stochastic programming, scenarios are used to estimate the distributions of the stochastic parameters. Therefore, stochastic network design problems are either formulated using a static set of scenarios, or, they are solved using algorithms that employ sampling. If a static set of scenarios is used, then either such a set is assumed available (Tsiakis et al., 2001; Alonso-Ayuso et al., 2003; Smith et al., 2004), or, it needs to be generated using appropriate methods. For example, in Høyland and Wallace (2001) and Høyland et al. (2003), methods are proposed to generate sets of scenarios that enforce specified statistical properties. In contrast to using a static set of scenarios, sampling based solution methods, such as the sample average approximation (SAA) algorithm, dynamically generate sets of representative scenarios for the problem. In the SAA algorithm, these sets are used to obtain different approximations of the original stochastic model that are then solved to produce feasible solutions. In this method, sampling is also applied to evaluate the optimality gaps associated with the solutions obtained. The SAA solution approach has been applied to stochastic network design by, e.g., Santoso et al. (2005); Azaron et al. (2008); Vila et al. (2007); Schütz et al. (2009); Bidhandi and Yusuff (2011). Whenever scenarios are used, it may happen that the set of scenarios that is considered produces a model whose solution time is prohibitive. Scenario reduction techniques, as the frameworks defined in Heitsch and Römisch (2007, 2009), have been proposed to reduce the scenario set, and thus produce easier models to address, while limiting the errors caused by such reductions.

Once a scenario set is defined, it can be used to formulate the stochastic problem as a large-scale deterministic model, referred to as the extensive form in Birge and Louveaux (2011). Commercial solvers have been applied directly to the models formulated using scenario sets, see Tsiakis et al. (2001); Vila et al. (2007); Azaron et al. (2008). However, given the complexity of these problems, the direct use of commercial solvers generally limits the size of the instances that can be solved in acceptable times. Therefore, a second common feature among algorithms specifically developed for stochastic network design models is the use of decomposition strategies taking advantage of the block structure characterizing the extensive form models (blocks being defined according to the considered scenarios Birge and Louveaux (2011)).

Two such strategies have been successfully applied for network design. The first is based on Benders decomposition (Benders, 1962) which, when applied in the stochastic setting, is referred to as the L-shaped algorithm, introduced in Slyke and Wets (1969), see also e.g., Birge and Louveaux (2011). Following this strategy, the stochastic network design model is first projected onto the space defined by the first stage variables (i.e., the design variables). By doing so, the problem decomposes according to the considered scenarios (i.e., a flow model for each scenario). The problem is then solved by reformulating the scenario subproblems using an outer linearization approach and then applying a relaxation algorithm on the resulting equivalent model, see Riis and Andersen (2002); Smith et al. (2004); Santoso et al. (2005); Bidhandi and Yusuff (2011). The second decomposition strategy used for stochastic network design models is referred to as scenario decomposition, see Birge and Louveaux (2011). Scenario decomposition is obtained by applying Lagrangean relaxation to the non-anticipativity constraints (i.e., the constraints ensuring that a single design is used under all considered scenarios). The original stochastic problem is again decomposed per scenario (i.e., a deterministic network design defined for each scenario). The resulting scenario subproblems can then be used to obtain a general lower bound, by solving the Lagrangean dual as in Schütz et al. (2009), or as a means to produce a more efficient solution approach, as in the case of the branch-and-fix with coordination algorithm developed in Alonso-Ayuso et al. (2003) and strongly refined in Escudero et al. (2012), or the progressive hedging-based metaheuristics proposed in Crainic et al. (2011b).

To conclude, while scenario decomposition has been applied to stochastic network design, as in many other problem settings, the applications aimed to generate single-scenario subproblems. Moreover, despite advances in MIP solver sophistication and power, the inherent difficulty of network design and the number of scenarios required for a correct representation of uncertainty still drastically limits the problem dimensions one can address in acceptable computing times. Revisiting the decomposition of the scenario set into multi-scenario subproblems then appears a methodological avenue worth studying, as also indicated by the work of Escudero et al. (2013) on randomly grouped scenarios.

We now proceed to define a comprehensive set of grouping strategies and experimentally evaluate their behavior and performance on stochastic network design models using a progressive hedging-based meta-heuristic.

3 Scenario Grouping

Strategies for grouping scenarios aim to enhance solution methods, here, a progressive hedging-based meta-heuristic for stochastic network design. Several questions need to be answered in order to achieve achieve this goal, e.g., how many groups should be created? should scenarios that are grouped be similar or not? should the groups induce a partition of the scenario set or not? what criteria should be used to measure similarity among scenarios, and so on.

We start the presentation of the proposed strategies with a general discussion of these issues (Section 3.1), and then proceed to address the issues of the number of groups to create and of grouping similar scenarios (Section 3.2), grouping dissimilar scenarios (Section 3.3), creating covers or partitions of the scenario set (Section 3.4) and, finally, of the measures we use to decide scenario similarity (Section 3.5).

3.1 Issues and General Strategies

We begin by providing some general insights regarding how the proposed grouping strategies for the new scenario-decomposition approach are developed. Recall that the evaluation of these strategies will be performed by means of the new progressive hedging meta-heuristic solving multi-scenario subproblems (Algorithm mS-PH of Section 4.1).

The main question one desires to answer can be stated as follows: is grouping scenarios efficient, that is, does solving subproblems comprised of multiple scenarios enable our proposed algorithm to be more efficient when compared to the original procedure where single scenario subproblems were solved? This question has already been partially addressed in Escudero et al. (2013), where the authors showed that grouping scenarios in progressive hedging can produce a more efficient strategy for obtaining lower bounds for scenario-based integer stochastic problems. Our aim is to see if such benefits are also possible in the context of stochastic network design. Therefore, the first grouping strategy is to simply generate scenario groups randomly. This strategy will serve as a benchmark to both evaluate the possible gains in grouping versus not grouping and investigate the more refined strategies that we propose to generate the groups.

The second main question we investigate is how to create groups. Recall that, by solving multi-scenario subproblem, the proposed progressive hedging algorithm produces a single solution for each group used in the decomposition. Therefore, part of the differences in the solutions of the subproblems, which would have been observed had scenarios been treated separately, are now directly reconciled within each cluster. Less iterations of the algorithm are therefore expected to be necessary to reach consensus over all subproblem solutions but, at the same time, more effort is expected to be required to address each subproblem, as multi-scenario subproblems are generally harder to solve than single-scenario ones. We also expect the choice of grouping strategy to significantly influence the solutions obtained at each iteration. The efficiency of the algorithm will thus be measured both in terms of the computational effort needed and the quality of the overall solution produced.

In order to design grouping strategies that attain the right balance between the effort needed to solve the subproblems and the quality of the final solution obtained, two general principles are established. The first principle concerns whether groups should consist of scenarios that are similar to or dissimilar from each other. The general idea is that by including similar scenarios in the groups, the subproblems obtained are expected to be easier to solve at each iteration of the progressive hedging algorithm. On the other hand, differences between the subproblem solutions are likely to be more pronounced, as scenario groups tend to be different from one another when similar scenarios are grouped.

In such a case, more iterations are needed to reach consensus. In contrast, if groups are made up of dissimilar scenarios, the subproblems are expected to be harder to solve but fewer iterations are likely to be needed to reach consensus. To investigate this issue, three grouping strategies are proposed: group similar scenarios, group similar scenarios but introduce a dissimilar group, and group dissimilar scenarios. The first and last of these strategies will help to test the two polar opposite approaches to scenario grouping, while the second qualifies as an in-between strategy that enables to measure the marginal impact of introducing dissimilarity in grouping.

The second principle established is the characteristics of the resulting groups. We mainly investigate whether the groups should define a *partition*, wherein each scenario is included in a single group, or a *cover*, wherein scenarios are allowed to appear in several groups. Or, put another way, should scenarios be used in the definition of more than one subproblem? By using a partition in the decomposition strategy, the resulting subproblems are necessarily different from one another and consensus is expected to be harder to obtain. In contrast, when a cover is used and scenarios are replicated in more than one subproblem, then the iterative process of the progressive hedging meta-heuristic can be accelerated.

3.2 Grouping Similar Scenarios

In this section, we discuss the methods we use for grouping scenarios and determining the number of groups. Let w_s be the vector of descriptive statistics associated to each of the $s = 1, \ldots, |S|$ scenarios to be grouped. The statistics, described in Section 3.5, are then used to measure scenario similarity and assemble them into g groups, $\bar{C} = \{C_1, \ldots, C_g\}$, corresponding to the groups of descriptive statistics. We then refer to a scenario s being in group C_i when its vector w_s is in that cluster.

Our methodology for creating groups of similar scenarios is inspired by the k-means clustering algorithm proposed to partition n data points into m clusters such that each data point is in the cluster whose mean is closest (Marsland, 2009). The standard form of this method is displayed in Algorithm 1.

```
**Algorithm 1** K-Means
**Require:** $n$ data points;
**Require:** $k$, the number of clusters to create;
1: Find an initial clustering of $n$ points into $k$ clusters;
2: **while** have a new clustering **do**
3:     Calculate the mean of each cluster;
4:     Assign each point to the cluster with the closest mean;
5: **end while**
```

The number of groups, k, is not known a priori. To determine it, we assume we are provided with a lower and upper bound on the number of groups, and execute Algorithm 1 for each number of groups between those bounds. We then choose the number k such that the difference between the error associated with groups k and $k-1$ is the greatest. For example, when searching between 4 and 8 groups with the group error for each size given in Fi

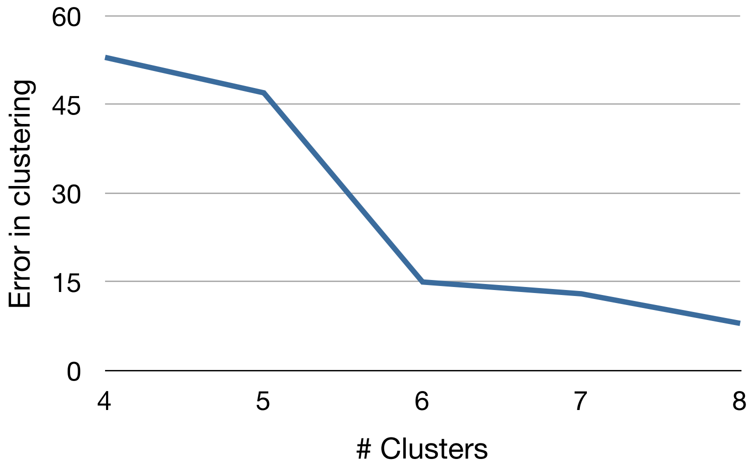


Figure 1: Comparing Clustering Errors to Choose k

We determine the distance from a point (scenario) to another point or from a point to the mean of a group based on the Euclidean distance. The error associated with a group is then the total distance of the scenarios from the mean of the group to which they are assigned.

To create an initial set of k groups, we first identify k core scenarios, each of which will represent the mean of a group. We then assign each remaining scenario to the closest core scenario. To identify the k core scenarios, we use the methodology developed by Arthur and Vassilvitskii (2007), which iteratively chooses the core scenarios randomly from the set of scenarios, but with weighted probabilities that reflect the distance of each scenario (not selected as core) from the previous core scenario chosen. Along with approximation guarantees relating to the error of the resulting clustering, the authors provided compelling computational evidence that this methodology is superior to simply choosing at random.

We present the method in detail in Algorithm 2. The core scenario s^i is chosen randomly in Step 2 through a lottery process in which the probability of picking a given scenario p is defined as $\frac{\|w_{s_p}-w_{s^{i-1}}\|}{\sum_{q=1}^{|S|}\|w_{s_q}-w_{s^{i-1}}\|}$, where function $\|\cdot\|$ is the Eucledean distance defined on the number of dimensions in the vectors of statistics. In all generality, assuming that the vectors of statistics associated with scenarios $s \in S$ are $|J|-$dimensional vectors defined as $w_s = [w_s^j]_{j\in J}$ then, given two scenarios s and s', we have $\|w_s - w_{s'}\| = \sqrt{\sum_{j \in J} \left(w_s^j - w_{s'}^j\right)^2}$. Therefore, scenarios that are distant when compared to the previous core point s^{i-1} (i.e., scenarios p that have a high $\|w_{s_p} - w_{s^{i-1}}\|$ value) have a higher probability of being picked than scenarios that are close to it (i.e., scenarios p with low $\|w_{s_p} - w_{s^{i-1}}\|$ values). Diversity is thus favored in the choices made concerning the core scenarios.

Algorithm 2 Choose Initial Set of Core Scenarios

```
Require: Statistics $w_s$ for describing scenario $s = 1, \ldots, |S|$;
Require: Number of core scenarios, $k$, to find;
1: Take one scenario, $s^1$, chosen randomly among $s_1, \ldots, s_{|S|}$;
2: Define a new core scenario, $s^i$, by randomly selecting among the scenarios not yet
   core with probabilities $s_p = \frac{\|w_{s_p} - w_{s^{i-1}}\|}{\sum_{q=1}^{|S|} \|w_{s_q} - w_{s^{i-1}}\|}$;
3: Repeat Step 2 until $k$ core scenarios are selected.
```

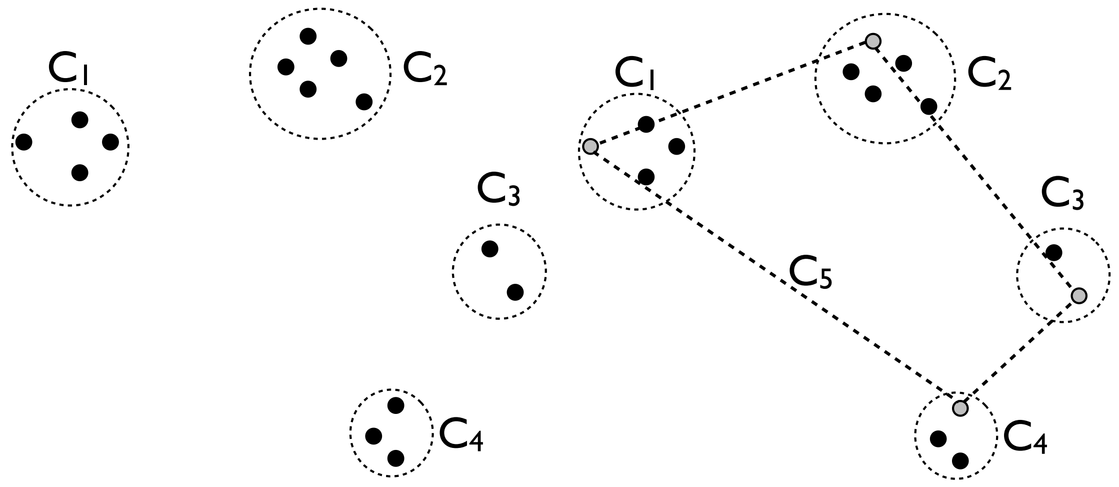
3.3 Considering Scenario Dissimilarity

We consider two approaches in introducing dissimilarity into our scenario-grouping strategies. The first, addresses the issue "locally", by starting with groups of similar scenarios and creating a new group that introduces dissimilarity to the grouping. The second is a "global" approach where groups are created by directly considering dissimilarity measures.

Introducing Dissimilarity into Groups of Similar Scenarios. We introduce dissimilarity into the groups of similar scenarios produced by the method presented in the previous subsection by creating an additional group that consists of one scenario from each of those created groups. We illustrate this method with Figures 2(a) and 2(b). Four scenario groups, C_1, C_2, C_3, and C_4, were created. A fifth scenario group, called the *Dissimilarity Group* (C_5 in Figure 2(b)), is created by choosing the scenario from each group that is closest to the center of that group. Let $\overline{w}_i = [\overline{w}_i^j]_{j \in J}$ be the $|J|-$dimensional vector representing the center of each group C_i, where $\overline{w}_i^j = \sum_{s \in C_i} \frac{p_s}{p_{C_i}} \times w_s^j$, $\forall j \in J$, with $p_{C_i} = \sum_{s \in C_i} p_s$. The *Dissimilarity Group* is then created by adding the scenario $s^i = \arg\min_{s \in C_i} \|w_s - \overline{w}_i\|$ from each group C_i.

Whether the scenarios chosen for the *Dissimilarity Group* are removed from their initial clusters $(C_1, \ldots, C_4)$ is an algorithm parameter determined by the selection of grouping type, partition or cover (Section 3.4).

Grouping Dissimilar Scenarios. Our methodology for creating such groups is very similar to how we group similar scenarios in Section 3.2. Specifically, for a fixed



(a) Initial Set of Similar Scenarios Groups (b) With Dissimilarity Group

Figure 2: Introducing Dissimilarity Into Similar Scenario Groups

number of groups, we execute an algorithm for grouping dissimilar scenarios that only differs from Algorithm 1 in Step 4, where points are instead assigned to the group whose mean is the furthest away. Similarly, when finding the initial groupings, we use Algorithm 2 to find the initial core scenarios, but then assign the remaining scenarios to the groups that are the furthest away. To choose the number of groups, we again assume a lower and upper bound on the possible number of groups, and execute our method for each number between those bounds. We then evaluate the quality of a grouping of fixed cardinality by calculating for each group the distance between the mean of that group and the scenario closest to that mean and then finding the average of these distances. We choose the number k such that this average distance is greatest.

3.4 Grouping Scenarios into Covers or Partitions

In (nearly) all of the methods we have discussed so far we create groups of scenarios that partition the set of scenarios. As indicated previously, however, the methodology may also consider covers such that a scenario may appear in multiple groups.

To create covers of scenarios, we first create groups of similar scenarios (Section 3.2). Next, for each scenario $s \in S$, we find the group C_i, other than the one it is currently assigned, for which value $\|w_s - \overline{w}_i\|$ is minimal, and add s to it. Note that, we do not update the vector $\overline{w}_i$ associated to C_i after adding this new scenario. Thus, we have that each scenario will appear in exactly two groups. Note that, the grouping algorithm building partitions calculated the probability of scenario group p_{C_τ} as $\sum_{s \in C_\tau} p_s$. With each scenario appearing in exactly two groups we set $p_{C_\tau} = \sum_{s \in C_\tau} p_s/2$.

3.5 Descriptive Statistics

The proposed scenario-grouping methodology requires that one specifies the scenario characteristics according to which scenario similarity or dissimilarity is to be measured. Such statistics can relate either to structural properties of the scenario, or a solution to the deterministic network design problem associated with that scenario. In this paper we consider the following statistics for scenario s:

Demand-based. The first descriptive statistic we use relates to commodity demands. Thus, $w_s^{demand} = [d^{ks}]_{k \in K}$ is a $|K|-$dimensional vector, with the k^{th} element containing the demand volume associated to commodity k in scenario s.

Solution-based. The second descriptive statistic we use relates to attributes of a solution, (x^s, y^s), produced by solving $CMND(s)$ with the original fixed charges, f_{ij}. Specifically, we consider the total commodity volume that flows on each arc in the solution (x^s, y^s). In this case, $w_s^{arc-flow} = [x_{ij}^s]_{(i,j) \in A}$ is an $|A|$-dimensional vector, where the element corresponding to arc (i, j) is calculated as $x_{ij}^s = \sum_{k \in K} x_{ij}^{ks}$.

4 Computational Experiments

We experimentally evaluated the proposed scenario-decomposition approach and the strategies for grouping scenarios. Our goal is twofold. First, to explore the performance of the proposed strategies with respect to the various choices described in Section 3. Second, to evaluate the performance of scenario grouping in addressing stochastic network design problems by means of a new progressive hedging-based meta-heuristic for stochastic network design that solves subproblems comprised of multiple scenarios.

We initiate the section by introducing the multiple-scenario progressive hedging meta-heuristic, followed by the description of the experimental environment (Section 4.2), and the presentation of the comparative results for grouping similar scenarios (Section 4.3), introducing dissimilarity into grouping (Section 4.4), grouping dissimilar scenarios (Section 4.5), and building partitions or covers (Section 4.6).

4.1 A Progressive Hedging-Based Metaheuristic

Progressive hedging (Rockafellar and Wets, 1991) has computationally proven to both produce effective meta-heuristics (Crainic et al., 2011b; Haugen et al., 2001; Løkketangen and Woodruff, 1996) and efficiently compute lower bounds (Escudero et al., 2013) for scenario-based stochastic integer problems. The method may also be an integral part of larger sampling-based solution procedures (Birge and Louveaux, 2011). The general strategy is based on the use of scenario decomposition (obtained through Lagrangean relaxation), which produces a set of single-scenario subproblems. A solution to the stochastic problem is then constructed by applying an iterative procedure which updates the Lagrange multipliers of each subproblem to tend toward a consensus solution among subproblems.

When applied to network design, each single-scenario subproblem solved at each iteration of the progressive hedging procedure represents a deterministic network design problem yielding a (potentially different) design (Crainic et al., 2011b). These designs are reconciled in order to create a single reference point. Then, at the beginning of the next iteration, the fixed cost associated with each arc is altered through an augmented Lagrangian-type technique to hopefully induce the resulting subproblems to yield solutions closer to the current reference point. Differences between the designs are thus reconciled indirectly. A progressive hedging approach converges when all subproblems yield the same design. While convergence is guaranteed for continuous optimization problems, the presence of integer variables in stochastic network design models eliminates that guarantee.

We introduce a new meta-heuristic, named *mS-PH* and displayed in Algorithm 3, that solves subproblems that may be comprised of multiple scenarios produced by the scenario-grouping strategies. The new meta-heuristic generalizes the method proposed by Crainic et al. (2011b) for the $CMND(S)$.

Specifically, the algorithm begins by creating a list of scenario groups, $\bar{C} = \{C_1, \dots, C_g\}$, where $C_\tau \subseteq S$ and $\cup_{\tau=1}^{g} C_i = S$. Note , the list $\bar{C}$ is static throughout the course of Algorithm 3. Then, at each iteration, we set $p_s = p_s/p_{C_\tau}$ where $p_{C_\tau} = \sum_{s \in C_\tau} p_s$, and solve $CMND(C_\tau)$, for $\tau = 1, \dots, g$, to produce the designs $y^{C_\tau \nu}$.

Having determined the designs $y^{C_\tau \nu}$ for $\tau = 1, \dots, g$ at iteration ν, we can derive a single design for $CMND(S)$, $\tilde{y}^\nu$, by setting the design variables $\tilde{y}^\nu_{ij}$ to

$$\tilde{y}^\nu_{ij} = \begin{cases} 1, & \text{if } y^{C_\tau \nu}_{ij} = 1 \text{ for any } \tau = 1, \dots, g \\ 0, & \text{otherwise,} \end{cases} \qquad \forall (i,j) \in A. \tag{5}$$

A feasible solution is thus obtained in Step 11, at each iteration of Algorithm mS-PH. As consensus is driven through the updating strategy applied on the fixed costs of the subproblems (i.e., Step 14), the differences observed between the designs $y^{C_\tau \nu}$, for $\tau = 1, \dots, g$, are expected to gradually decrease as the number of iterations ν increases. In Step 12, the best network found y^{Best} is updated based on the quality (i.e., total cost) of the feasible solution obtained at the current iteration.

To update the fixed costs of the subproblems in Step 14 of Algorithm mS-PH, we use the same strategy as Crainic et al. (2011b). Namely, with parameters, c^{low}, c^{high}, and β,

Algorithm 3 Multi-Scenario Progressive Hedging Meta-heuristic *mS-PH*

```
1: *Initialization: Determine the $g$ scenario groups;*
2: $\nu \leftarrow 0$;
3: $f_{ij}^{\nu} \leftarrow f_{ij}, \ \forall (i,j) \in A$;
4: Construct the list of scenario groups $\bar{C} = \{C_1, \ldots, C_g\}$;
5: *First phase: Seek consensus on the arcs $(i,j)$ that should exist in the design*
6: while stopping criterion not met do
7:     for all $\tau = 1 \rightarrow g$ do
8:         Solve $CMND(C_\tau)$ for design $y^{C_\tau \nu}$;
9:     end for
10:    $\nu \leftarrow \nu + 1$;
11:    Construct a feasible network $\tilde{y}^{\nu}$ by applying (5);
12:    Update best solution, $y^{Best} = \tilde{y}^{\nu}$, if appropriate;
13:    $\bar{y}_{ij}^{\nu} \leftarrow \sum_{\tau=1}^{g} p_{C_\tau} y_{ij}^{C_\tau \nu}, \ \forall (i,j) \in A$;
14:    Global update of the fixed costs $f_{ij}^{\nu}$ using the reference point $\bar{y}_{ij}^{\nu}, \forall (i,j) \in A$, by
       applying (6);
15: end while
16: Let $\bar{y} = \bar{y}^{\nu}$ and $\tilde{y} = \tilde{y}^{\nu}$
17: *Second phase: Solve a restriction of $CMND(S)$ as a mixed integer program if a
    consensus solution was not obtained*
18: if $\bar{y} \neq \tilde{y}$ then
19:    Fix design variables in $CMND(S)$ for which consensus is obtained in $\bar{y}$;
20:    Solve $CMND(S)$ as a restricted mixed integer program for design $y^{Final}$;
21:    Update best solution, $y^{Best} = y^{Final}$ if appropriate;
22: end if
23: return $y^{Best}$
```

we set

$$f_{ij}^{\nu} = \begin{cases} \beta f_{ij}^{\nu-1}, & \text{if } \bar{y}_{ij}^{\nu-1} < c^{low}, \\ \frac{1}{\beta} f_{ij}^{\nu-1}, & \text{if}, \bar{y}_{ij}^{\nu-1} > c^{high}, \\ f_{ij}^{\nu-1}, & \text{otherwise} \end{cases} \quad \forall (i,j) \in A. \tag{6}$$

Parameters c^{low} and c^{high} define thresholds on the level of consensus that are considered to be sufficient to warrant either an increase of the value of the fixed cost of an arc (if consensus is towards not including the arc in the design) or, a decrease of the value of the fixed cost of an arc (if consensus is towards including the arc in the design). As for parameter β, it defines the factor by which the fixed costs are adjusted. At iteration ν, a measure of the current level of consensus for each arc is obtained in Step 13 by computing the values $\bar{y}_{ij}^{\nu}, \ \forall (i,j) \in A$ (i.e., the expected value of including each arc in the network). We consider consensus to have been reached regarding arc (i,j) at iteration ν when $\bar{y}_{ij}^{\nu} \in \{0,1\}$. Otherwise, $0 < \bar{y}_{ij}^{\nu} < 1$ and thus consensus is not yet observed over all networks $y^{C_\tau \nu}, \tau = 1, \ldots, g$.

We use similar stopping criteria in Step 6 as presented in Crainic et al. (2011b). Namely, we stop after 1,000 iterations, 25 iterations without an improving solution, 10 hours of CPU time, or there are fewer than γ $(0 \leq \gamma \leq 1)$ of the arcs for which consensus has not been reached. Therefore, when the *First phase* of Algorithm mS-PH is completed, consensus on all arcs may not be observed (i.e., $\bar{y} \neq \tilde{y}$). When such a situation occurs, the *Second phase* is used to resolve it. This phase produces the network y^{Final}, which represents the final solution obtained once the restricted $CMND(S)$ is solved. In Step 21, y^{Best} is updated one last time if y^{Final} improves it.

4.2 Experimental environment

To avoid “noise” introduced by addressing subproblems approximately, we solve single and multi-scenario subproblems to (near-) optimality with CPLEX version 12. When solving subproblems, CPLEX was executed with an optimality tolerance of 1% and a time limit of 1,800 seconds. Other parameters were left at their default values. We provided the disaggregate inequalities $x_{ij}^{ks} \leq d^{ks} y_{ij}$ to CPLEX as *User Cuts*, meaning CPLEX will only add them to the formulation when they are violated by a solution to the linear relaxation. Algorithm mS-PH was executed with $\gamma = .1$, meaning that Phase 1 of the algorithm terminates when consensus has been reached for at least 90% of the arcs.

We also compare the performance of the strategies we propose to that of grouping scenarios randomly (Escudero et al., 2012, 2013). To do so, we randomly determine the number of groups between $|S|/2$ and $|S|/4$ (i.e., between 4 and 8 for 16 scenarios and 8 and 16 for 32 scenarios) and then randomly assign scenarios to groups.

All experiments were performed on a machine with 8 Intel Xeon CPUs running at 2.66 GHz with 32 GB RAM. Unless otherwise noted, computation times are reported in seconds. To evaluate the quality of the solutions produced by Algorithm mS-PH, we also solved these instances with CPLEX, in which case CPLEX was executed with an optimality tolerance of 1% and a time limit of 10 hours. We refer to the dual bound produced by CPLEX in these experiments as CPLEX LB. For the scenario grouping methods requiring the commodity flows on each arc ($w_s^{arc-flow}$) in a deterministic solution to each scenario-based instance, we solve the single scenario network design problem with CPLEX.

We used 6 problem classes (groups 4 to 9) from the set of R instances taken from Crainic et al. (2011b). The attributes of each class are given in Table 1. Each of these classes contains five networks, labeled 1, 3, 5, 7, 9, yielding a total of 40 networks. The labels 1, 3, 5, 7, and 9 reflect an increasing ratio of fixed to variable costs. For each of these networks, there are instances with 16 and 32 scenarios ($|S| = 16, 32$). While the instances in our computational study are rather small, and in some cases, easily solved by CPLEX, we chose them because we want to run experiments where CPLEX can solve both the single and multi-scenario subproblems to near-optimality in a reasonable time. We believe that by doing so, the experiments provide a clearer understanding of the impact of grouping scenarios and solving multi-scenario subproblems.

Table 1: Problem Class Characteristics

Group	$\|N\|$	$\|A\|$	$\|K\|$
4	10	60	10
5	10	60	25
6	10	60	50
7	10	82	10
8	10	83	25
9	10	83	50

In all the experiments reported in the next subsections, we ran Algorithm mS-PH multiple times for each instance, once with single-scenario subproblems and once for every strategy for creating multiple-scenario subproblems. We allow for between $|S|/2$ and $|S|/4$ groups of scenarios of size between 1 and 8. The tables displayed in these subsections report summary statistics (averages) for these instances when solving single-scenario subproblems (rows *Single*), multiple-scenario subproblems with randomly built groups (rows *Random*), and multiple-scenario subproblems with the strategies proper to each analysis/subsection.

We report the average number of CPU seconds (*P1 Time*) and iterations (*P1 # Iter*) required to complete Phase 1, the average time spent solving the restricted MIP in Phase 2 (*P2 Time*), the average optimality gap (*P1 Gap CPLEX LB*) of the best solution produced during Phase 1 of Algorithm mS-PH as measured against the dual bound produced by CPLEX, calculated as (Phase 1 Solution Value - CPLEX LB)/(Phase 1 Solution Value), and the average optimality gap (*Gap CPLEX LB*) of the final solution produced by Algorithm mS-PH as measured against the dual bound produced by CPLEX, and calculated as (Phase 2 Solution Value - CPLEX LB)/(Phase 2 Solution Value).

4.3 Effectiveness of Grouping Similar Scenarios

We report the results for the strategies yielding groups of similar scenarios (inducing a partition of S), when similarity is measured with respect to the vectors of demand and arc flows in rows *Similar* w_s^{demand}) and *Similar* $w_s^{arc-flow}$, respectively. Tables 2 and 3 display these results (averages) for the instances with 16 and 32 scenarios, respectively.

Comparing the *Single* row with the others in both tables, we see that solving multi-scenario subproblems in Algorithm mS-PH, regardless of how the scenarios are grouped,

Table 2: Performance when Grouping Similar Scenarios - 16 Scenarios

Method	P1 Time (sec.)	P1 # Iter.	P2 Time (sec.)	P1 Gap CPLEX LB	Gap CPLEX LB
Single	3,682.34	91.28	5.48	4.22	1.51
Random	3,524.70	40.70	35.43	3.21	1.14
Similar w^{demand}	1,701.97	5.53	9.63	2.14	1.07
Similar $w^{arc-flow}$	1,224.93	36.07	3.00	2.04	1.08

Table 3: Performance when Grouping Similar Scenarios - 32 Scenarios

Method	P1 Time (sec.)	P1 # Iter.	P2 Time (sec.)	P1 Gap CPLEX LB	Gap CPLEX LB
Single	7,176.86	128.55	44.62	4.65	1.85
Random	8,113.73	60.33	26.20	4.30	1.64
Similar w^{demand}	5,459.50	25.73	9.97	3.02	1.37
Similar $w^{arc-flow}$	4,371.77	48.70	69.37	2.91	1.26

enables Phase 1 of Algorithm 3 to converge in significantly fewer iterations, and find a better solution. Overall, when solving multi-senario subproblems, Algorithm mS-PH always finds a better solution than when solving single-scenario subproblems.

While grouping scenarios randomly is better (in terms of solution quality and number of iterations required for Phase 1 to converge) than not grouping at all, a more refined grouping strategy enables Algorithm mS-PH to produce an even higher quality solution and for Phase 1 to converge in fewer iterations. We also see that the grouping strategies we propose enable the algorithm to terminate in less time than when solving single-scenario subproblems or multi-scenario subproblems created by grouping scenarios randomly. Ultimately, we see that each grouping strategy proposed outperforms both random grouping and not grouping at all with respect to the quality of the solution produced and the number of iterations and time required for the proposed meta-heuristic to complete.

4.4 Effectiveness of Introducing Dissimilarity into Grouping

We next study whether introducing dissimilarity into a grouping of similar scenarios with respect to the w^{demand} and $w^{arc-flow}$ scenario characteristics can improve the performance of Algorithm mS-PH. Tables 4 and 5 report results when introducing a Dissimilarity Group, as illustrated in Figure 2(b), for instances with 16 and 32 scenarios, respectively. We report on the two variants of the Dissimilarity Group approach producing a partition or a cover of the scenario set. The first, rows *Similar* w^- - DG (P), yields partitions by creating the Dissimilarity Group and removing the scenarios in that group from the groups to which they were initially assigned. The second, rows *Similar* w^- - DG (R), yields covers by not removing the scenarios assigned to the Dissimilarity Group from the scenarios of their initial groups.

The results show that, for any scenario statistic, introducing the Dissimilarity Group often leads to higher quality solutions at the expense of longer run times. In fact, for both the 16 and 32 scenario instances, a method that includes the Dissimilarity Group yielded the highest quality solution (*Similar* w^{demand}- DG (P) for 16 scenario instances, *Similar* $w^{arc-flow}$-DG(R) for 32 scenario instances). We can also observe that, for any particular statistic, building a cover, i.e., repeating the scenarios in the Dissimilarity Group often results in fewer iterations needed for Phase 1 to converge than when partitions are built. However, the same conclusion can not be drawn regarding solution quality.

Table 4: Impact of Introducing Dissimilarity - 16 Scenarios

Method	P1 Time (sec.)	P1 # Iter.	P2 Time (sec.)	P1 Gap CPLEX LB	Gap CPLEX LB
Similar w^{demand}	1,701.97	5.53	9.63	2.14	1.07
Similar w^{demand} - DG (P)	1,872.73	10.20	4.53	2.63	0.93
Similar w^{demand} - DG (R)	2,021.27	6.90	5.70	3.33	1.02
Similar $w^{arc-flow}$	1,224.93	36.07	3.00	2.04	1.08
Similar $w^{arc-flow}$ - DG (P)	3,026.83	83.63	4.47	2.72	1.03
Similar $w^{arc-flow}$ - DG (R)	2,722.37	36.80	3.83	2.30	1.01

Table 5: Impact of Introducing Dissimilarity - 32 Scenarios

Method	P1 Time (sec.)	P1 # Iter.	P2 Time (sec.)	P1 Gap CPLEX LB	Gap CPLEX LB
Similar w^{demand}	5,459.50	25.73	9.97	3.02	1.37
Similar w^{demand} - DG (P)	4,508.17	23.43	20.93	2.92	1.26
Similar w^{demand} - DG (R)	6,097.63	35.63	21.47	3.79	1.32
Similar $w^{arc-flow}$	4,371.77	48.70	69.37	2.91	1.26
Similar $w^{arc-flow}$ - DG (P)	4,790.33	59.03	81.45	3.38	1.27
Similar $w^{arc-flow}$ - DG (R)	5,035.63	56.47	48.10	4.10	1.25

4.5 Effectiveness of Grouping Dissimilar Scenarios

We now turn to comparing grouping of similar or dissimilar scenarios, for the two scenario statistics. Tables 6 and 7 display the average results for these cases for instances with 16 and 32 scenarios, respectively. Based on these results, it appears that, while grouping dissimilar scenarios is superior to grouping them randomly or not at all, we can not conclude that doing so is better than grouping similar scenarios with respect to the impact on solution quality. Yet, it does appear that grouping dissimilar scenarios can significantly reduce the number of iterations needed for Phase 1 to converge.

Table 6: Group Similar or Dissimilar Scenarios - 16 Scenarios

Method	P1 Time (sec.)	P1 # Iter.	P2 Time (sec.)	P1 Gap CPLEX LB	Gap CPLEX LB
Similar w^{demand}	1,701.97	5.53	9.63	2.14	1.07
Dissimilar w^{demand}	2,045.97	3.07	2.40	1.76	1.08
Similar $w^{arc-flow}$	1,224.93	36.07	3.00	2.04	1.08
Dissimilar $w^{arc-flow}$	1,926.03	7.10	3.90	2.42	1.17

Table 7: Group Similar or Dissimilar Scenarios - 32 Scenarios

Method	P1 Time (sec.)	P1 # Iter.	P2 Time (sec.)	P1 Gap CPLEX LB	Gap CPLEX LB
Similar w^{demand}	5,459.50	25.73	9.97	3.02	1.37
Dissimilar w^{demand}	5,875.40	9.13	51.47	4.41	1.41
Similar $w^{arc-flow}$	4,371.77	48.70	69.37	2.91	1.26
Dissimilar $w^{arc-flow}$	4,708.30	11.43	14.23	3.28	1.35

4.6 Effectiveness of Covering Instead of Partitioning Scenarios

The last part of the experimental results address the question whether one should create groups of scenarios that cover or partition the set of scenarios. We report in Tables 8 and 9 the average results obtained for grouping similar scenarios within covers and partitions, for instances with 16 and 32 scenarios, respectively. The results show that, for both 16 and 32 scenarios and each statistic, using a cover of scenarios both increases solution quality and significantly decreases the number of iterations necessary for Phase 1 of Algorithm mS-PH to converge.

We then summarize in Table 10 the results obtained with the best cover strategy (i.e., Similar w^{demand}) and benchmark them against the original progressive hedging strategy of solving single-scenario subproblems. Overall, on the instances with 16 scenarios, the algorithm using the cover strategy converges to solutions that are, on average, 29% better than the ones obtained when solving single-scenario subproblems. Furthermore, these results were obtained in less than half the time. As for the instances with 32 scenarios, the average improvement in solution quality is 26% for the cover strategy while solution times are again significantly reduced by 24%.

Table 8: Cover or Partition Similar Scenarios - 16 Scenarios

Method Statistic	Type	P1 Time	P1 # Iter.	P2 Time	P1 Gap CPLEX LB	Gap CPLEX LB
w^{demand}	partition	1,701.97	5.53	9.63	2.14	1.07
w^{demand}	cover	680.13	3.50	7.43	2.36	0.82
$w^{arc-flow}$	partition	1,224.93	36.07	3.00	2.04	1.08
$w^{arc-flow}$	cover	2,386.63	4.50	14.50	2.50	1.03

Table 9: Cover or Partition Similar Scenarios - 32 Scenarios

Method Statistic	Type	P1 Time	P1 # Iter.	P2 Time	P1 Gap CPLEX LB	Gap CPLEX LB
w^{demand}	partition	5,459.50	25.73	9.97	3.02	1.37
w^{demand}	cover	6,299.00	12.27	84.93	4.07	1.20
$w^{arc-flow}$	partition	4,371.77	48.70	69.37	2.91	1.26
$w^{arc-flow}$	cover	5,777.20	11.37	23.70	4.43	1.21

Table 10: Not grouping vs. Covering

	16 scenarios		32 scenarios	
Method	Total time (sec.)	Gap CPLEX LB	Total time (sec.)	Gap CPLEX LB
Single	3,687.82	1.51	7,221.48	1.85
Similar w^{demand} cover	1,711.60	1.07	5,469.47	1.37

5 Conclusions and future work

We proposed a methodological approach to build strategies for grouping scenarios as defined by the type of scenario decomposition (partition or cover), type of grouping (similar or dissimilar scenarios), and the measures specifying scenario similarity. We experimentally studied these strategies in the context of stochastic network design by comparing the behavior and performance of a new progressive hedging-based meta-heuristic for stochastic network design that solves subproblems comprised of multiple scenarios. We compared the proposed strategies not only among themself, but also against the strategy grouping scenarios randomly and the lower bound provided by the state-of-the-art MIP solver, CPLEX.

We have shown that by solving multi-scenario subproblems, the metaheuristic produces better results in terms of solution quality and computing efficiency. Grouping scenarios is always beneficial and doing it the smart way even more so, as the manner in which the multi-scenario subproblems are constructed has a definite impact on the performance of the algorithm. Thus, the results show that solving multi-scenario subproblems based on a random partition of scenarios often enables the progressive hedging meta-heuristic to achieve a solution that is 25% better than when single-scenario subproblems are solved, and in fewer than half the iterations. Moreover, the results also show that, compared to grouping scenarios randomly, partitioning scenarios with the grouping strategy we propose can enable the meta-heuristic to obtain a solution that is 16% better. The number of iterations and time required to achieve this result is always reduced, often by half. Finally, the results show that a covering of scenarios enables the meta-heuristic to find solutions that are 16% better than partitioning them. Thus, considering all strategies tested, the covering strategy with respect to commodity demands leads the highest quality solutions, with an overall improvement in solution quality of approximately 27% with respect to the original strategy of solving single-scenario subproblems, and the quickest convergence. Notice that, while the scenario-grouping strategies we propose are derived in the context of addressing stochastic network design problems, they can be applied to other stochastic programs as well, and we believe similar performance will be observed.

The significant benefits to the performance of the progressive hedging-based meta-heuristics brought by the proposed scenario decomposition strategy, resulting in solving multi-scenario subproblems, suggest multiple avenues for future research. Network design problems of more than a modest size are notoriously difficult to solve with even the best integer programming solvers, and thus many effective meta-heuristics have been developed. Thus, adapting one of these to multi-scenario network design problems will be one of our next efforts. We also think that grouping scenarios dynamically, together with a method that incorporates memory, will be beneficial for larger instances, and thus we will investigate methods for doing so. The extension of our grouping strategies to multistage stochastic problems will also be considered. In this context, determining how the methods should be adapted to multistage scenario trees will both define interesting questions to investigate and pose important methodological challenges. Lastly, we believe that scenario decomposition strategies based on grouping scenarios may be beneficial for a broader set of solution methods. We are therefore in the process of investigating how these principles can be used in the context of Benders decomposition applied to stochastic problems.

Acknowledgments

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC) and by the Fonds québécois de la recherche sur la nature et les technologies (FQRNT). This support is gratefully acknowledged.

References

Alonso-Ayuso, A., Escudero, L. F., Garìn, A., Ortuño, M. T., and Pérez, G. (2003). An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization*, 26:97–124.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

Azaron, A., Brown, K. N., Tarim, S. A., and Modarres, M. (2008). A multi-objective stochastic programming approach for supply chain desing considering risk. *International Journal of Production Economics*, 116:129–138.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

Bidhandi, H. M. and Yusuff, R. M. (2011). Integrated supply chain planning under uncertainty using an improved stochastic approach. *Applied Mathematical Modelling*, 35:2618–2630.

Birge, J. R. and Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer, 2nd edition.

Crainic, T.G.., Errico, F., Rei, W., and Ricciardi, N. (2011a). Modeling Demand Uncertainty in Two-Tiered City Logistics Planning. Publication CIRRELT-20-54, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal, QC, Canada.

Crainic, T.G.., Fu, X., Gendreau, M., Rei, W., and Wallace, S. (2011b). Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58(2):114–124.

Crainic, T.G.. and Laporte, G. (1997). Planning models for freight transportation. *European Journal of Operational Research*, 97(3):409–438.

Escudero, L. F., Garín, M. A., Merino, M., and Pérez, G. (2012). An algorithmic framework for solving large-scale multistage sotchastic mixed 0-1 problems with nonsymmetric scenario trees. *Computers & Operations Research*, 39:1133–1144.

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

Gaivoronski, A. A. (2005). Stochastic optimization problems in telecommunications. In Wallace, S. W. and Ziemba, W. T., editors, *Applications of Stochastic Programming*, volume 5 of *MPS/SIAM Series on Optimization*, pages 669–704. SIAM, Philadelphia, PA.

Haugen, K. K., Løkketangen, A., and Woodruff, D. L. (2001). Progressive hedging as a meta-heuristic applied to stochastic lot-sizing. *European Journal of Operational Research*, 132:116–122.

Heitsch, H. and Römisch, W. (2007). A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters*, 35:731–738.

Heitsch, H. and Römisch, W. (2009). Scenario tree reduction for multistage stochastic programs. *Computational Management Science*, 6:117–133.

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

Høyland, K. and Wallace, S. W. (2001). Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307.

Klibi, W., Martel, A., and Guitouni, A. (2010). The design of robust value-creating supply chain networks: A critical review. *European Journal of Operational Research*, 203:283–293.

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

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

Marsland, S. (2009). *Machine learning: an algorithmic perspective.* Chapman & Hall/CRC.

Riis, M. and Andersen, K. A. (2002). Capacitated network design with uncertain demand. *INFORMS Journal on Computing*, 14(3):247–260.

Rockafellar, R. and Wets, R. J.-B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147.

Santoso, S., Ahmed, S., Goetschalckx, M., and Shapiro, A. (2005). A stochastic programming approach for supplu chain network design under uncertainty. *European Journal of Operational Research*, 167:96–115.

Schütz, P., Tomasgard, A., and Ahmed, S. (2009). Supply chain design under uncertainty using sample average approximation and dual decomposition. *European Journal of Operational Research*, 199:409–419.

Slyke, R. V. and Wets, R. J.-B. (1969). L-shaped linear programs with application to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663.

Smith, J. C., Schaefer, A. J., and Yen, J. W. (2004). A stochastic integer programming approach to solving a synchronous optical network ring design problem. *Networks*, 44(1):12–26.

Thapalia, B. K., Wallace, S. W., Kaut, M., and Crainic, T. G. (2012). Single source single-commodity stochastic network design. *Computational Management Science*, 9:139–160.

Tsiakis, P., Shah, N., and Pantelides, C. C. (2001). Design of multi-echelon supply chain networks under demand uncertainty. *Industrial & Engineering Chemistry Research*, 40:3585–3604.

Vila, D., Martel, A., and Beauregard, R. (2007). Taking market forces into account in the design of production-distribution networks: A positioning by anticipation approach. *Journal of Industrial and Management Optimization*, 3(1):29–50.

Wallace, S. W. (2000). Decision making under uncertainty: is sensitivity analysis of any use? *Operations Research*, 48(1):20–25.