CIRRELT
Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport
Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

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

Teodor Gabriel Crainic
Mike Hewitt
Walter Rei

August 2012

CIRRELT-2012-41

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Scenario Clustering 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] Kate Gleason College of Engineering, Rochester Institute of Technology, James E. Gleason Building, 77 Lomb Memorial Drive, Rochester, NY, USA 14623-5603

Abstract. We present a technique for enhancing a progressive hedging-based metaheuristic for a network design problem that models demand uncertainty with scenarios. The technique uses machine learning methods to cluster scenarios and, subsequently, the metaheuristic repeatedly solves multi-scenario subproblems. With an extensive computational study we see that solving multi-scenario subproblems leads to a significant increase in solution quality and that how you construct these multi-scenario subproblems directly impacts solution quality.

Keywords: Stochastic programs, network design, progressive hedging, 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.

---

* Corresponding author: TeodorGabriel.Crainic@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1 Introduction

Network design models define an important class of combinatorial optimization problems which have a wide gamut of applications. These problems naturally appear in various forms in the planning of complex systems (e.g., logistics, transportation and telecommunications), and at the strategic, tactical and operational levels. In such contexts, network design models are used to produce plans that can define the structure, allocation of resources or adjustments to be applied to the networks. As such, plans are 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.

In the case of network design models, demands usually entail a certain level of uncertainty and define stochastic parameters (i.e., origins, destinations or volumes of the demands). 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 (i.e., networks). However, this type of approach can lead to arbitrarily bad solutions, see Wallace (2000). Recent studies conducted by Lium et al. (2009) and Thapalia et al. (2012) have clearly shown that cost-effective networks obtained in stochastic settings are structurally different then the ones obtained in deterministic settings.

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. As an example, see Crainic et al. (2011b). Once the appropriate set of scenarios is generated, a two-stage stochastic network design problem is solved, with the first stage modeling the choice of design and the second modeling its cost-effectiveness for each scenario.

However, such problems remain notoriously hard to solve with off-the-shelf optimization software, both because deterministic network design models are difficult to solve, and because modeling uncertainty with scenarios can yield very large instances. Thus, much like with the deterministic case, we need heuristic methods for producing high-quality designs for realistically-sized instances. Progressive hedging (Rockafellar and Wets, 1991) has (computationally) proven (Crainic et al., 2011b) to form the basis of an effective meta-heuristic for two-stage, scenario-based, stochastic network design problems. As such, the progressive hedging algorithm proposed in Crainic et al. (2011b) can either be used as a stand-alone solution procedure for the considered problem, or, as an integral

part of larger sampling-based solution strategies Birge and Louveaux (2011).

In an iteration of a progressive hedging approach for network design, multiple subproblems are solved, with each subproblem solution representing a (potentially different) design. These designs are reconciled in order to create a single reference point, and, at the beginning of the next iteration the fixed cost associated with each arc is altered with an augmented lagrangian-type technique in hopes that the solutions to the resulting subproblems will match the current reference point. 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.

In the approach presented by Crainic et al. (2011b), a subproblem was solved for each scenario and took the form of a deterministic network design problem with demand levels corresponding to those seen in that scenario. We build on this work to develop a progressive hedging-based approach that instead solves subproblems comprised of multiple scenarios. We do so for two reasons: (1) By solving fewer subproblems at an iteration fewer designs need to be reconciled and potentially fewer iterations needed to reach convergence, and, (2) By solving multi-scenario subproblems the differences between subsets of scenarios are considered explicitly, yielding a design for each subproblem that may be closer to the final, consensus design.

Knowing how to group scenarios to form multi-scenario subproblems in a manner that reduces the number of iterations required to converge is not obvious. Should groups consist of scenarios that are "similar" or "dissimilar?" We study these questions computationally and develop techniques for grouping scenarios that, while derived in the context of solving stochastic network design problems, can be applied to other stochastic programs. To group "similar" scenarios, we use a *k-means* clustering technique (Marsland, 2009) to partition scenarios into clusters, with the clusters then used to define the multi-scenario subproblems. We develop a similar technique for grouping "dissimilar" scenarios. We also study whether the groups of scenarios should partition or cover the set of scnearios. We cluster scenarios based on various statistical representations, such as the vector of commodity demands for scenario s, and structural properties of the solutions to the individual scenario problems. We study the effectiveness of the methodologies that we have developed computationally. Using instances solved in Crainic et al. (2011b), we evaluate the impact of the various strategies for clustering scenarios on the effectiveness of the resulting progressive hedging approach.

The rest of this paper is organized as follows. In Section 2 we review the literature related to stochastic programming for network design. In Section 3 we define the problem we wish to solve and a progressive hedging-based metaheuristic that can solve subproblems comprised of multiple scenarios. In Section 4 we discuss how we group scenarios in order to create these multi-scenario subproblems. In Section 5 we computationally study

the effectiveness of our methodologies, and in Section 6 we draw conclusions based on these experiments and outline future efforts.

2 Literature Review

In this section, we first review the main stochastic network design models that have been proposed and then discuss solution approaches for such models. Two subsections are included here. In subsection 2.1, we present the models by focussing on the diversity of applications for which they are used. As for subsection 2.2, we provide a general description of the algorithmic strategies used to develop the solution methods proposed for these problems.

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. Therefore, problems are formulated by defining which decisions are taken before all information is available (i.e., first stage decisions) and which decisions are made afterwards (i.e., second stage decisions and onwards). 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).

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, e.g., see 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 solve 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. In the context of solving stochastic network design models, the SAA solution approach has been widely applied Santoso et al. (2005); Azaron et al. (2008); Vila et al. (2007); Schütz et al. (2009); Bidhandi and Yusuff (2011). Finally, whenever scenarios are used, an important and related topic is the problem of scenario reduction. It may happen that the set of scenarios that is considered produces a model whose solution time is prohibitive. When such a situation occurs, scenario reduction techniques, as the frameworks defined in Heitsch and Römisch (2007) and Heitsch and Römisch (2009), can be applied to obtain smaller sets that, while producing easier models to solve, limit 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. The resulting model, referred to as the extensive form in Birge and Louveaux (2011), can then be solved using various solution strategies. In the case of stochastic network design problems, 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 may limit 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. Given the block structure characterizing the extensive form models (blocks being defined according to the considered scenarios Birge and Louveaux (2011)), various decomposition strategies have proven efficient to solve these problems.

In the particular case of network design models, two such strategies have been successfully applied. The first is based on Benders decomposition (Benders, 1962) which, when applied in the stochastic setting, is referred to as the L-shaped algorithm, see 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 that is 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 algorithm developed in Alonso-Ayuso et al. (2003) or the progressive hedging-based metaheuristics proposed in Crainic et al. (2011b).

3 A Progressive Hedging-Based Metaheuristic

The solution approach we propose in this paper seeks to solve a problem that includes a finite set of known scenarios. Therefore, it is assumed that the process of producing the scenarios has been applied (i.e., sampling, scenario generation and reduction) and we now solve the problem with the resulting scenarios. In this section we formally define this problem and present the approach.

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

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

subject to

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

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

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

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

Here, 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) ensures 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.

We next present in Algorithm 1 a metaheuristic that generalizes the method proposed by Crainic et al. (2011b) for solving $CMND(S)$, as it allows for the subproblems solved at an iteration to be comprised of multiple scenarios. Specifically, when the algorithm begins, it creates a list of scenario groups, $\bar{S} = (S_1, \ldots, S_g)$, where $S_\tau \subseteq S$ and $\cup_{\tau=1}^{g} S_i = S$.

Note that in Algorithm 1 the list $\bar{S}$ is static throughout the course of the algorithm. At an iteration of Algorithm 1, we solve $CMND(S_\tau)$ for $\tau = 1, \ldots, g$ to produce the design $y^{S_\tau \nu}$. When solving $CMND(S_\tau)$ we set $p_s = p_s / p_{S_\tau}$ where $p_{S_\tau} = \sum_{s \in S_\tau} p_s$.

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

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

In Step 11 of Algorithm 1, we use the same update strategy as presented in Crainic et al. (2011b). Namely, with parameters, c^{low}, c^{high}, and β, we set

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

We consider consensus to have been reached regarding arc (i, j) at iteration ν when $\bar{y}_{ij}^{\nu} \in \{0, 1\}$. 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 been reached. To implement Algorithm 1, we must determine how to construct the list of scenario groups, $\bar{S} = (S_1, \ldots, S_g)$. We next discuss the methods we have developed for doing so.

Algorithm 1 Grouped Scenario Meta-heuristic

Require: Number of scenario groups, g.

Initialization: We first determine scenario groups.
$\nu \leftarrow 0$
$f_{ij}^{\nu} \leftarrow f_{ij},\ \forall (i,j) \in A.$
Construct list of scenario groups, $\bar{S} = (S_1, \ldots, S_g)$.
First phase: We next seek consensus on which arcs (i,j) should exist in the design
while stopping criterion not met **do**
  for all $\tau = 1 \rightarrow g$ **do**
    Solve $CMND(S_\tau)$ for design $y^{S_\tau \nu}$
  end for
  $\nu \leftarrow \nu + 1$
  global update of fixed costs f_{ij}^{ν}
  $\bar{y}_{ij}^{\nu} \leftarrow \sum_{\tau=1}^{g} p_{S_\tau} y_{ij}^{S_\tau \nu},\ \forall (i,j) \in A$
  Update best solution, $y^{Best} = y^{S\nu}$ when appropriate
end while
Let $\bar{y} = \bar{y}^{\nu}$
Second phase: We next solve a restriction of $CMND(S)$ as a mixed integer program
Fix design variables for which consensus is obtained
Solve $CMND(S)$ as a restricted mixed integer program

4 Scenario Grouping

We view grouping scenarios as a type of clustering problem, and use a methodology similar to k-means clustering (Marsland, 2009). The purpose of k-means clustering is to partition n data points into m clusters such that each data point is in the cluster whose mean is closest. Often times, and in our application, m is not fixed ahead of time, and thus the appropriate number of clusters must also be determined. While determining the optimal set of k clusters is NP-Hard, many computationally effective and efficient heuristics have been developed. In particular, we use the heuristic k-means++, (Arthur and Vassilvitskii, 2007) which has been shown to have approximation guarantees.

In Section 4.1 we discuss the methods we use for clustering scenarios and determining the number of clusters. To use them to partition the $|S|$ scenarios into g groups, $\bar{S} = (S_1, \ldots, S_g)$, we represent each scenario with a vector of descriptive statistics. Then, we cluster those vectors, and create groups of scenarios that correspond to the clusters of descriptive statistics. While it is these vectors of statistics, w_s, for scenarios $s = 1, \ldots, |S|$, that are clustered, we refer to a scenario s being in a cluster C_i when in fact its vector w_s is in that cluster. We describe the different types of descriptive statistics we use for clustering in Section 4.5.

4.1 Grouping Similar Scenarios

In Algorithm 2 we present the standard k-means clustering algorithm. In general, when determining the distance from a point to another point or a point to the mean of a cluster, we do so based on Euclidean distance. The error associated with a clustering is the total distance of points from the mean of the cluster to which they are assigned.

Algorithm 2 K-Means

Require: n data points
Require: The number of clusters to create, m
1: Find an initial clustering of n points into m 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**

To begin the k-means algorithm we must create an initial set of k clusters. We do so by first identifying k core scenarios, each of which will represent the mean of a cluster. 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 each scenarios' distance from the previous core scenario chosen. Along with approximation guarantees relating to the error of the resulting clustering, Arthur and Vassilvitskii (2007) provide compelling computational evidence that this methodology for choosing an initial set of core points is superior to simply choosing at random. We present the methodology in detail in Algorithm 3.

Algorithm 3 Choose Initial Set of Core Points

Require: Statistics w_s for describing scenario $s = 1, \ldots, |S|$
Require: Number of core points, k to find
1: Take one scenario, s^1, chosen randomly from $s_1, \ldots, s_{|S|}$.
2: Take a new scenario, s^i, randomly, with the probability of choosing $s_p = \frac{\|w_{s^p} - w_{s^{i-1}}\|}{\sum_{q=1}^{|S|} \|w_{s^q} - w_{s^{i-1}}|}$
3: Repeat Step 2 until k core points are found.

Lastly, to determine the number of clusters, we assume we are provided with a lower and upper bound on the number of clusters, and execute Algorithm 2 for each number of clusters between those bounds. We then choose the number k such that the difference between the error associated with k and $k-1$ clusters is the greatest. For example, if we were considering between 4 and 8 clusters, with the error of the resulting cluster for each size given in Figure 4.1, then we would choose $k = 6$.

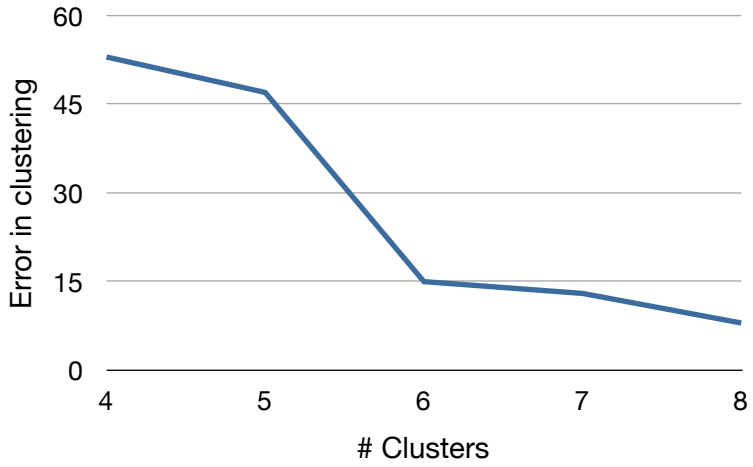


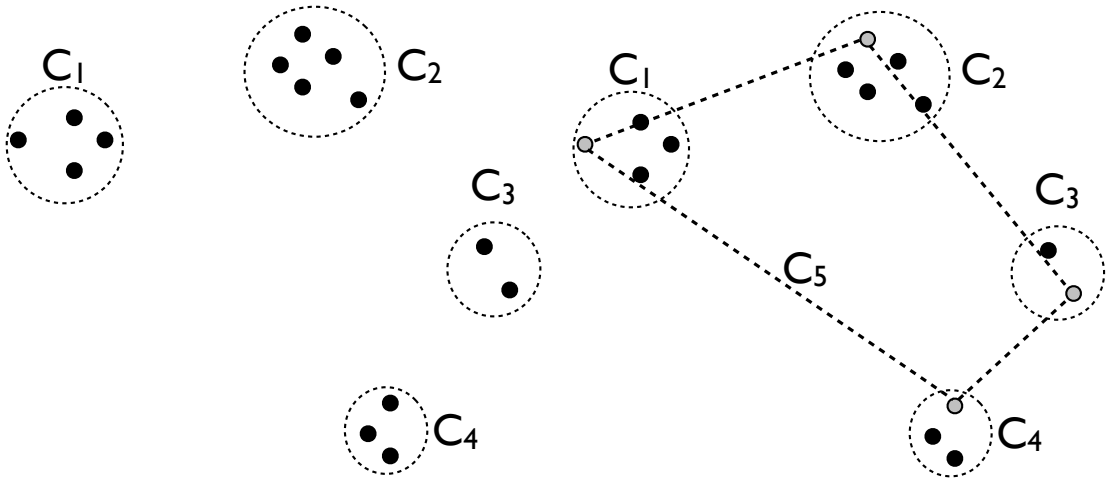
Figure 1: Comparing Errors of Clusterings to Choose k

4.2 Introducing Dissimilarity into Groups of Similar Scenarios

We also consider systematically introducing dissimilarities into the clusters of similar scenarios produced by Algorithm 2, and do so by creating one more cluster that consists of one scenario from each of those created clusters. We illustrate this method with Figures 2(a) and 2(b). Here, Algorithm 2 has output 4 scenario clusters, C_1, C_2, C_3, and C_4. A fifth scenario group, (C_5 in Figure 2(b)) called the *Dissimilarity Group*, is created by choosing one scenario from each of the clusters. Whether the scenarios chosen for the *Dissimilarity Group* are removed from their initial clusters $(C_1, \ldots, C_4)$ is an algorithm parameter. In our experiments, we choose the scenario from each cluster that is closest to the center of that cluster.

4.3 Grouping Dissimilar Scenarios

Instead of introducing dissimilarities into groups of similar scenarios we can instead consider grouping together scenarios that are dissimilar. Our methodology for creating such groups is very similar to how we group similar scenarios. Specifically, for a fixed number of groups, we execute an algorithm for grouping dissimilar scenarios that only differs from Algorithm 2 in Step 4, where points are instead assigned to the cluster whose mean is the furthest away. Similarly, when finding the initial clustering, we use Algorithm 3 to find the initial cluster centers and then assign the remaining points to the centers 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 methodology 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



(a) Grouping of Scenarios After Clustering (b) With Dissimilarity Group

Figure 2: Introducing Dissimilarity Into Similar Scenario Groupings

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.

4.4 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. However, we could also consider covers, or that a scenario may appear in multiple groups. To create covers of scenarios, we first creaate groups of scenarios by clustering similar scenarios with Algorithm 2. Next, for each scenario s^i, we find the cluster C_j other than the one it was assigned to by Algorithm 2 that minimizes $\|w_{s^i} - \mu_j\|$. We then add s^i to the group corresponding to C_j. We note that we do not update the mean of C_j after adding this new scenario. Thus, we have that each scenario will appear in exactly two groups. Previously, Algorithm 1 calculated the probability of scenario grouping p_{S_τ} as $\sum_{s \in S_\tau} p_s$. With each scenario appearing in exactly two groups we let $p_{S_\tau} = \sum_{s \in S_\tau} p_s/2$.

4.5 Descriptive Statistics

To execute Algorithm 2 we need statistics describing each scenario. Such statistics can relate either to structural properties of the scenario, or, of 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} is a $|K|-$dimensional vector, with the j^{th} element containing d^{js}.

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 demand that flows on each arc in the solution (x^s, y^s). In this case, $w_s^{arc-flow}$ is an $|A|$-dimensional vector, where the element corresponding to arc (i, j) is calculated as $\sum_{k \in K} x_{ij}^{ks}$.

5 Computational Experiments

We computationally study whether grouping scenarios and, subsequently, solving multi-scenario subproblems improves the performance of Algorithm 1. To focus our experiments on the impact of grouping scenarios, we solve single and multi-scenario subproblems with CPLEX (unlike what was done in Crainic et al. (2011b), where a Tabu Search heuristic was used to produce high quality designs for each single-scenario subproblem). 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 provide 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 1 was executed with $\gamma = .1$, meaning that Phase 1 of the algorithm will terminate when consensus has been reached for at least 90% of the arcs.

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 calculate the quality of the solutions produced by Algorithm 1 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. For the scenario grouping methods that require 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.

For our computational study, we use 6 problem classes (4-9) from the set of R instances seen in Crainic et al. (2011b). 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 charges. 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 doing so will 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

5.1 Effectiveness of Grouping Similar Scenarios

To understand whether one should have Algorithm 1 solve subproblems in Step 8 that contain multiple scenarios and how those groups of scenarios should be chosen we ran Algorithm 1 multiple times for each instance, once with single scenario subproblems and once for every strategy for creating multiple scenario subproblems. We report in Tables 2 and 3 summary statistics (averages) for these instances when solving single scenario subproblems (Single), and multiple scenario subproblems when clustering is done based on similarity with respect to the vectors $w_s^{demand}, w_s^{arc-flow}$. Table 2 contains results for the instances with 16 scenarios and Table 3 contains results for those with 32. We allow for between $|S|/2$ and $|S|/4$ clusters of scenarios of size between 1 and 8. We also study the effectiveness of grouping scenarios randomly. To do so, we randomly determine the number of groups (again between 4 and 8 for 16 scenarios and 8 and 16 for 32 scenarios), and then randomly assign scenarios to groups.

We report the average number of 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 1 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 1 as measured against the dual bound produced by CPLEX, and calculated as (Phase 2 Solution Value - CPLEX LB)/(Phase 2 Solution Value).

For each table, by comparing the "Single" row with the others, we see that solving multi-scenario subproblems in Algorithm 1, regardless of how the scenarios are grouped, enables Phase 1 of Algorithm 1 to converge in significantly fewer iterations, and find

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

a better solution. Overall, when solving multi-senario subproblems Algorithm 1 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, clustering scenarios nearly always enables Algorithm 1 to produce an even higher quality solution and for Phase 1 to converge in fewer iterations. We also see that clustering scenarios enables Algorithm 1 to terminate more quickly than when solving single scenario subproblems or multi-scenario subproblems created by grouping scenarios randomly. Ultimately, we see that each clustering-based method outperforms both randomly grouping and not grouping at all with respect to the quality of the solution produced and the number of iterations and time required for Algorithm 1 to complete.

5.2 Effectiveness of Introducing Dissimilarity into Grouping

We next study whether introducing dissimilarity into a grouping of similar scenarios with respect to w^{demand} and $w^{arc-flow}$ vectors can improve the performance of Algorithm 1. In Tables 4 and 5 we report results when introducing a Dissimilarity Group, as illustrated in Figure 2(b), for instances with 16 and 32 scenarios. We report two variants of the Dissimilarity Group approach. The first, Similar w^- - DG (P), introduces the Dissimilarity Group and removes the scenarios in that group from the groups to which they were

initially assigned. The second, Similar w^- - DG (R), introduces the Dissimilarity Group and does not remove the scenarios from their initial groups.

We see that for a given statistic, introducing the Dissimilarity Group often leads to higher quality solutions at the expense of longer run times. In fact, for both the 16 scenario 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 also note that for a given statistic, repeating the scenarios in the Dissimilarity Group (and thus having groups of scenarios that are a cover as opposed to a partition) often results in fewer iterations needed for Phase 1 to converge than not repeating them. 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

5.3 Effectiveness of Grouping Dissimilar Scenarios

We next study whether it is better to group scenarios that are dissimilar (as discussed in Section 4.3) or similar. Thus, we present in Table 6 and 7 results when dissimilar scenarios are grouped together along with the results regarding when similar scenarios are grouped together for instances with 16 and 32 scenarios. While it is difficult to draw conclusions regarding the impact of grouping dissimilar scenarios on solution quality, it

does appear that doing so 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

5.4 Effectiveness of Covering Instead of Partitioning Scenarios

Lastly, we study whether one should create groups of scenarios that cover or partition the set of scenarios. We report in Tables 8 and 9 results obtained with groupings of scenarios that were created by first clustering similar scenarios and then adding each scenario to the next nearest cluster as well as those obtained with groupings of scenarios that were created by clustering similar scenarios. We see 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 1 to converge.

Table 8: Cover or Partition Similar Scenarios - 16 Scenarios

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

Table 9: Cover or Partition Similar Scenarios - 32 Scenarios

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

6 Conclusions and future work

We have presented a technique for enhancing a progressive hedging-based metaheuristic for a network design problem that models demand uncertainty with scenarios. The technique uses machine learning techniques to cluster scenarios and, subsequently, the metaheuristic repeatedly solves multi-scenario subproblems (as opposed to single-scenario subproblems as is done in existing work). With an extensive computational study we saw that solving multi-scenario subproblems leads to an increase in solution quality, and that how you construct these multi-scenario subproblems also has an impact. Ultimately, we saw that creating a cover of the scenarios by first clustering them with respect to commodity demands and then adding each scenario to one other group lead to the highest quality solutions and the fewest number of iterations for the first phase of the metaheuristic to converge.

The benefits seen from 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 metaheuristics 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, and with a method that incorporates memory, will be beneficial for larger instances, and thus we will investigate methods for doing so. Lastly, we think grouping scenarios can be incorporated into exact methods. For example we are investigating using groups of scenarios to aggregate optimality cuts in a Bender's-based method.

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.

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.

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.

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.

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*, pages 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.

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.