**Submitted to *INFORMS Journal on Computing***
**manuscript (Please, provide the manuscript number!)**

**Authors are encouraged to submit new papers to INFORMS journals by means of a style file template, which includes the journal title. However, use of a template does not certify that the paper has been accepted for publication in the named journal. INFORMS journal templates are for the exclusive purpose of submitting to an INFORMS journal and should not be used to distribute the papers in print or online or to submit the papers to another publication.**

# Pseudo-random Instance Generators in C++ for Deterministic and Stochastic Multi-commodity Network Design Problems

Eric Larsen
CIRRELT and Université de Montréal, eric.larsen.1@umontreal.ca

Serge Bisaillon
CIRRELT and Université de Montréal, serge.bisaillon@cirrelt.ca

Jean-François Cordeau,
CIRRELT and HEC Montréal, jean-francois.cordeau@hec.ca

Emma Frejinger
CIRRELT and Université de Montréal, emma.frejinger@umontreal.ca

Network design problems constitute an important family of combinatorial optimization problems for which many exact and heuristic algorithms have been developed over the last few decades. Two central problems in this family are the multi-commodity, capacitated, fixed charge network design problem (MCFNDP) and its stochastic counterpart, the two-stage stochastic MCFNDP with recourse. These are standard problems that often serve as workbenches for devising and testing models and algorithms in stylized but near-realistic settings. The purpose of this paper is to introduce two flexible, high-speed generators capable of simulating a wide range of settings for both the deterministic and stochastic MCFNDPs. We hope that, by facilitating systematic experimentation with new and larger sets of instances, these generators will facilitate a thorough assessment of the performance achieved by exact and heuristic solution methods in both deterministic and stochastic settings. We also hope that making these generators available will promote the reproducibility and comparability of published research.

*Key words*: Instance generator; network design; capacitated; fixed cost; stochastic programming
*History*:

## 1. Introduction

Network design problems (Crainic et al. 2021a) are concerned with choosing which arcs to open in a network and how to route flows from origins to destinations so as to satisfy a set of demands while respecting arc capacities. These flows can represent objects, people

or information, and may be finely or coarsely differentiated depending on the application. Most variants of network design can be characterized by the following four attributes: (i) the amounts of commodities demanded and supplied at the nodes of the network, (ii) the fixed costs associated with the opening of arcs, (iii) the flow-dependent and commodity-specific costs on the arcs, and (iv) the global and commodity-specific capacities on the arcs.

NP-hard network design problems are ubiquitous and occur in many areas of human activity such as supply chain management, transportation, and telecommunications. The size and complexity of these networks tend to increase as economic processes grow and become more integrated. Hence, the development of powerful exact and heuristic solution methods for network design is a very active field of research in combinatorial optimization. In this context, the linear, deterministic, multi-commodity, capacitated, fixed charge network design problem (MCFNDP) (Gendron et al. 1999) and its stochastic counterpart, the two-stage stochastic MCFNDP with recourse (Crainic et al. 2011), play a central role. They often serve as work benches for the design and testing of new mathematical formulations and solution algorithms. In addition, the successes achieved on these two fundamental problems often carry over to other related problems such as facility location problems.

In this article, we introduce two flexible, high-speed generators capable of simulating a wide range of settings for both deterministic and stochastic MCFNDPs. By facilitating systematic experimentation with new and larger sets of instances, we hope that these generators will lead to more thorough assessments and comparisons of the performance achieved by exact and heuristic solution methods in deterministic and stochastic settings. We also hope that making these generators publicly available will advance the reproducibility and comparability of published research.

To the best of our knowledge, only one generator for the deterministic MCFNDP was available until very recently (CommaLAB 2023b). This so-called "Mulgen" generator was conceived in Fortran more than 30 years ago by Bernard Gendron and his collaborators at Université de Montréal. In turn, it inherited from code included in the Appendix of Bertsekas (1991). Whereas the resulting instances (CommaLAB 2023a) have since been used in many publications by several groups of authors (see, e.g., Chouman et al. 2016, Crainic et al. 2001, Hewitt et al. 2010), the generator itself had fallen in disuse. A salient cause of this situation was poor usability due to lack of documentation. We have aimed to remedy

this by modernizing the code and extensively documenting the use of the generator as well as the code itself. In addition, we have substantially extended the available functionalities. We believe that the resulting generator has become a more flexible, capable, as well as more usable tool that will facilitate current research on MCFNDPs. This new generator is presented in Section 2. We view it as a general tool that stands on its own and that can also be used to supply basic deterministic instances to the stochastic generator presented later in Section 3.

Very recently, the existence of the new pre-publication Bonnet et al. (2025) proposing a generator of deterministic MCFNDPs and schedule-augmented MCFNDPs (SMCFNDPs) came to our attention. This generator follows a top-down approach: The user sets targets in terms of certain metrics taken from the literature on Complex Network Analysis (CNA) and the generator is intended to closely match these targets and other metrics from CNA with an MCFNDP (or SMCFNDP) featuring a hub-and-spoke topology. The approach followed with our generator is bottom-up: The user sets elementary characteristics of the MCFNDP as well as values or ranges in which its individual elements may fall, and the generator pseudo-randomly selects the individual elements inside the specified ranges.

Also to the best of our knowledge, there does not exist a publicly available generator for the two-stage stochastic MCFNDP. Papers published in this area have resorted to ad hoc randomizations of linear deterministic instances that are often based on one of the algorithms of Høyland et al. (2003) (see, e.g., Crainic et al. 2021b). In this context, ensuring public availability of a new stochastic instance generator that would facilitate systematic experimentation and foster reproducibility and comparability of published research appears to be particularly useful. We present a stochastic generator building stochastic realizations about a subset of parameters of a specified deterministic instance of MCFNDP. This basic deterministic instance is supplied in a generic format and can arise from any source, including the deterministic generator of Section 2. We devote the bulk of our attention to describing this stochastic generator in Section 3. Finally, Section 4 concludes the paper.

## 2. A generator of linear deterministic MCFNDPs

This section first introduces the linear deterministic MCFNDP (borrowing notation from Crainic et al. 2021a) and follows with a description of the corresponding deterministic generator.

### 2.1. The linear deterministic MCFNDP

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ denote a graph composed of arcs $(i,j) \in \mathcal{A}$ and nodes $i,j \in \mathcal{N}$. With each arc $(i,j) \in \mathcal{A}$ are associated a fixed cost $f_{ij}$ and a capacity $u_{ij} \geq 0$ limiting the total amount of flow on the arc. Demands are associated with commodities $k \in \mathcal{K}$ and defined over the nodes of the graph. The unit costs for using any arc $(i,j) \in \mathcal{A}$ are denoted $c_{ij}^k$. It is commonly assumed that each commodity is characterized by a single origin node $O(k) \in \mathcal{N}$, a single destination node $D(k) \in \mathcal{N}$ and a volume $d^k$. In this setting, the net outgoing flow of commodity $k$ at node $i$ is given by

$$w_i^k = \begin{cases} d^k, & \text{if } i = O(k), \\ -d^k, & \text{if } i = D(k), \\ 0, & \text{otherwise.} \end{cases}$$

The problem may also include commodity-specific capacities $b_{ij}^k \geq 0$ limiting the flow of specific commodities on the arcs. Note that the generator introduced below allows the presence of multiple sources and sinks for each commodity.

The linear deterministic MCFNDP comprises two sets of decision variables – binary design variables $y_{ij},\ (i,j) \in \mathcal{A}$, and continuous multi-commodity flow variables $x_{ij}^k \geq 0,\ (i,j) \in \mathcal{A}, k \in \mathcal{K}$. The problem can be defined as

$$\min_{\boldsymbol{y},\boldsymbol{x}} \left\{ \sum_{(i,j)\in\mathcal{A}} f_{ij} y_{ij} + \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k \right\} \tag{1}$$

$$\text{subject to} \sum_{j\in\mathcal{N}_i^+} x_{ij}^k - \sum_{j\in\mathcal{N}_i^-} x_{ji}^k = w_i^k, \qquad \forall i \in \mathcal{N}, \forall k \in \mathcal{K}, \tag{2}$$

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq u_{ij} y_{ij}, \qquad \forall (i,j) \in \mathcal{A}, \tag{3}$$

$$x_{ij}^k \leq b_{ij}^k y_{ij}, \qquad \forall (i,j) \in \mathcal{A}, \forall k \in \mathcal{K}, \tag{4}$$

$$x_{ij}^k \geq 0, \qquad \forall (i,j) \in \mathcal{A}, \forall k \in \mathcal{K}, \tag{5}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A}, \tag{6}$$

where $\boldsymbol{y}, \boldsymbol{x}$ respectively stand for all design variables $y_{ij},\ (i,j) \in \mathcal{A}$, and all flow variables $x_{ij}^k,\ (i,j) \in \mathcal{A}, k \in \mathcal{K}$. The objective function (1) minimizes the total costs expressed as a summation of fixed arc opening costs and a summation of flow-dependent transportation

costs. Constraints (2) enforce flow conservation at each node $i \in \mathcal{N}$, where $\mathcal{N}_i^+$ and $\mathcal{N}_i^-$ identify, respectively, the successor and predecessor nodes of $i$. Constraints (3) enforce overall capacity limits and act as linking constraints. Finally, (4) are commodity-specific capacity constraints (a.k.a. strong forcing constraints) restricting the flow of commodity $k \in \mathcal{K}$ on arc $(i, j)$. The latter are optional in the generator.

### 2.2. The generator

The description of the generator is divided into three parts: *first*, we outline its design and work flow, *second*, we list its functionalities, *third*, we describe its use and provide an assessment of its performance.

**2.2.1. Design and workflow.** A modest difference between the original Fortran code and our C++ version of the Mulgen generator consists in the use of a modern pseudo-random number generation library. The PCG (i.e., "permuted congruential generator") C++ library (see O'Neill 2023, Lemire 2023) is faster, statistically more reliable and more convenient (by making room for random streams that facilitate indexing and retrieval of a series of generated network problems).

Five principal improvements distinguish the new generator. We list them without particular order. *First*, we provide the option to read as well as write the graph $\mathcal{G}$ (i.e., nodes and arcs, without costs or capacities) from and to a text file. Hence, if desired, the user can start from bespoke or from actual topologies. In contrast, the original generator only allowed to specify general characteristics of the topology of $\mathcal{G}$ (i.e., numbers of nodes and either grid-like, circular or random connections). All aforementioned options are currently available whereas it is still possible to add arcs randomly, but in a greater variety of ways. *Second*, whereas selecting a grid-like topology necessarily entailed that it was superposed over a torus, which is a quite strong assumption to make about the prevailing connectivity, we also make it possible to apply grid-like topologies to cylinders (as in cyclical schedules in space-time settings, provided the graph is pruned so as to make time unidirectional) and to flat surfaces (as in bounded networks). *Third*, we offer the option to generate graphical representations in TeX format of the generated graphs that include volumes, capacities and costs. *Fourth*, we provide automatic safeguards and responses guiding the selection of options by the user. *Fifth*, the use of the new generator and its code are extensively documented. In contrast, the lack of documentation impeded usability of the original generator.

The generation proceeds through three main steps, each comprising options set by the user (for additional details on the groups of options hereafter highlighted in bold, see the corresponding paragraph in Section 2.2.2): *First*, generate the structure of the problem according to **core-structure options**. *Second*, if requested, adjust random arcs according to **tuning-random-arcs options** in order to alter the overall complexity. *Third*, if requested, adjust fixed costs upward and/or adjust arc capacities downward according to **tuning-design-and-flow-problems options** in order to alter complexity of design and flow problems. The generated instance can be saved using two MCFNDP-specific formats, a generic human-readable LP format, and the generic machine-readable Mathematical Programming System (MPS) format. Also, a graphical representation of the instance can be saved in the TeX format so as to be rendered in PDF format by a LaTeX compiler. This representation consists of a main pane containing the graph, a pane containing its legend, and eventually a third pane, whenever variable costs and/or capacities are differentiated between commodities. See Figure 1 for a small-scale example of a main pane featuring a 3 x 4 flat grid topology, 3 commodities, and one source and one sink for each commodity.

**2.2.2. Available functionalities.** The generator offers the following *options*. Their settings can be left at default values, specified in configuration files or overridden in the command line.

***Basic options:***

- Sourcing of configuration parameters from specified file(s) and/or command line.
- Output file name and choice of machine-readable, human-readable or graphical output format(s).
- Name of file where graph $\mathcal{G}$ of network is optionally read or saved.
- Seed and stream of pseudo-random number generator.

***Core-structure options:***

- Imposing either (i) random, (ii) grid-like (on a torus, a cylinder or a bounded flat surface), (iii) circular topology of $\mathcal{G}$, or (iv) reading $\mathcal{G}$ from file.
- Length of grid along X- and Y-axis if grid-like topology.
- Number of nodes $|\mathcal{N}|$ if not grid-like topology.
- Number of commodities $|\mathcal{K}|$.
- Number of additional random arcs (beyond those required by grid-like or circular connections when requested).

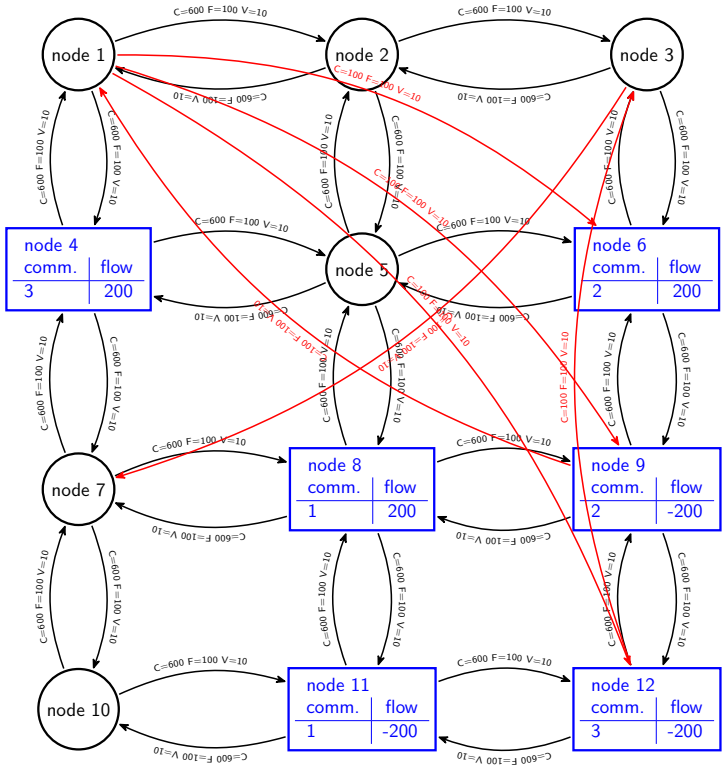


**Figure 1** **Main pane of a small-scale example of graphical representation**
**Reproduced from the legend pane:**
**Black arcs are part of the base, grid topology; red arcs are additional, randomly generated; $C = x$ along arc: overall capacity is equal to x; $F = y$ along arc: fixed cost of arc equal to y; $V = z$ along arc: variable cost of arc equal to z; capacities and variable costs are not differentiated between commodities; circular nodes are transit nodes; rectangular nodes are source ($flow > 0$) or sink ($flow < 0$) nodes for stated commodities.**

- Precluding (or not) parallel arcs.
- Minimum and maximum number of sources and sinks for each commodity.
- Imposing either (i) single source and single sink for each commodity $k \in \mathcal{K}$, or (ii) identical sources and identical sinks over all commodities, or (iii) sources and sinks randomly selected from $\mathcal{N}$ for all commodities.
- Minimum and maximum value of $d^k,\ k \in \mathcal{K}$.
- Minimum and maximum value of $f_{ij},\ (i,j) \in \mathcal{A}$, before tuning adjustments.
- Minimum and maximum value of $c^k_{ij},\ (i,j) \in \mathcal{A},\ k \in \mathcal{K}$, before tuning adjustments.
- Minimum and maximum value of $u_{ij},\ (i,j) \in \mathcal{A}$, before tuning adjustments.
- Minimum and maximum value of $b^k_{ij},\ (i,j) \in \mathcal{A}, k \in \mathcal{K}$, before tuning adjustments.
- Requiring that design variables $y_{ij},\ (i,j) \in \mathcal{A}$, appear in Objective (1) and Constraints (3), in the linear program described by output formats LP and MPS.
- Requiring that design variables $y_{ij},\ (i,j) \in \mathcal{A}$, appear in Objective (1) and Constraints (3) and that Constraints (4) be created, in the linear program described by output formats LP and MPS.

***Tuning-random-arcs options:*** Imposing specified percentages of random arcs where

- fixed cost must be set to zero,
- capacity must be set to total volume,
- one commodity-specific capacity must be set to zero,
- one commodity-specific capacity must be set to maximum capacity.

***Tuning-design-and-flow-problems options:*** Imposing specified

- uniform upward proportional adjustment of fixed costs over arcs,
- uniform downward proportional adjustment of overall capacities over arcs.

**2.2.3. Use and performance.** The generator is available in the depot located at **https://github.com/larseeri/determ-MCFNDP-gener**. The included **README.md** file supplies detailed information about building and running the generator from the Linux command line and running the generator from the included Docker container. Help is displayed on screen upon request or automatically in case of erroneous command line statement. For example, the following command line instruction:

```
./determ_gen +F newConfigFile.par -stream 1234 -seed 4567 -nbCom 10
```

would launch the generation of an MCFNDP instance according to options specified in configuration file newConfigFile.par, except that the random stream, random seed and number of commodities therein or their default values would respectively be superseded by 1234, 4567 and 10.

The generator achieves high speed computations. For example, generating and saving instances similar to the largest ones among the historical R and C series available at CommaLAB (2023a), that is r18 (20 nodes, 315 arcs, 200 commodities) and c64 (30 nodes, 700 arcs, 400 commodities), requires respectively less than 1 second and less than 4 seconds, using a single core of an Intel(R) Xeon(R) E5-2637 v4 @ 3.50GHz CPU running on Alma-Linux 9.2. The advanced user interested in the inner workings of the code will find useful that it has been extensively commented and documented.

## 3. A generator of linear two-stage stochastic MCFNDPs with recourse

The generator for the linear two-stage stochastic MCFNDP with recourse that we propose operates based on a supplied instance of the linear deterministic MCFNDP. The generator synthesizes a joint probability distribution governing a set of parameters from this base instance that are identified by the user as varying stochastically. In other words, the

generator randomizes a user-specified set of parameters of the base deterministic instance. To ensure computational tractability, the synthesized probability distribution is atomic. That is, it can be described by a finite set of realizations (i.e. "scenarios") and respective probabilities.

The synthesized distribution is shaped jointly (i) by primitive, easily interpretable requirements that are specified by the user and (ii) by pseudo-random determinations. Clearly, those requirements must be selected in order to promote the relevance and usefulness of assessments based on the generated problem instances. In turn, these qualities depend on the theoretical questions or contexts of application that are judged of interest. The instance of deterministic MCFNDP upon which the synthetic probability distribution is grafted is generic. It can arise from either actual, bespoke or synthetic MCFNDPs. If it is synthetic, then its structure and the values of its non-stochastic parameters also result (i) from the requirements that are specified at creation in order to promote relevance and usefulness in view of the tasks at hand, and (ii) possibly from pseudo-random determinations. The generator presented in Section 2 as well as any other existing generator of deterministic MCFNDPs can fulfill this role in a precise way.

In the endeavor to synthesize the probability distribution, we are naturally drawn to the scenario construction methods available in the abundant and growing literature investigating the computation of approximate solutions to stochastic programming problems. These methods fall in loosely demarcated classes called "scenario sampling", "scenario reduction" (a.k.a. "discrete scenario reduction", when scenarios are selected among existing ones) or "scenario generation" (a.k.a. "continuous scenario reduction", when scenarios are built freely). However, this inclination is tempered by an important caveat: In stochastic programming, the objective pursued by constructing sets of scenarios is to approximate the solution of problems whose underlying probability distribution is judged too complex to ensure tractability, while still being *known exactly or at least lending itself to sampling or estimation from historical data*. In contrast, our task is to synthesize an *a priori unknown* underlying distribution. The synthesized distribution is embodied in a set of scenarios constructed based on user specified distributional requirements coupled with pseudo-random determinations.

The Appendix provides an overview of the methods proposed in the stochastic programming literature for constructing scenarios and points to those among the latter that are

suited to our goal of synthesizing a distribution. It explains why we choose to construct scenarios with the Høyland-Kaut-Wallace (HKW) algorithm of Høyland et al. (2003) by matching targets for the first four moments and the linear correlations for all stochastic parameters appearing in the base linear deterministic MCFNDP at hand. We shall see momentarily how this information may be derived from a small set of assumptions and supplied to the generator.

### 3.1. The linear two-stage stochastic MCFNDP with recourse

The linear two-stage stochastic MCFNDP with recourse can be defined as follows. In comparison with the linear deterministic MCFNDP of Section 2.1, $\omega$, $p(\omega)$ and $\Omega$ respectively identify a stochastic realization (i.e., a scenario), the probability of a realization and the set of all stochastic realizations. The latter is assumed to be finite. We also assume that parameters $\boldsymbol{w}$, $\boldsymbol{u}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ may all have been randomized. Decision variables $\boldsymbol{y}$ and $\boldsymbol{x}(\omega)$ are respectively selected before and after disclosure of the stochastic realization $\omega$. The formulation becomes the following, where we have assumed for simplicity as in Section 2.1 that there is a single source node and a single sink node for each commodity:

$$\min_{\boldsymbol{y},\boldsymbol{x}(\omega),\forall\omega\in\Omega} \left\{ \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{\omega\in\Omega} p(\omega) \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k(\omega)x_{ij}^k(\omega) \right\} \tag{7}$$

$$\text{subject to } \sum_{j\in\mathcal{N}_i^+} x_{ij}^k(\omega) - \sum_{j\in\mathcal{N}_i^-} x_{ji}^k(\omega) = w_i^k(\omega), \qquad \forall i\in\mathcal{N}, \forall k\in\mathcal{K}, \forall\omega\in\Omega, \tag{8}$$

$$\sum_{k\in\mathcal{K}} x_{ij}^k(\omega) \le u_{ij}(\omega)y_{ij}, \qquad \forall(i,j)\in\mathcal{A}, \forall\omega\in\Omega, \tag{9}$$

$$x_{ij}^k(\omega) \le b_{ij}^k(\omega)y_{ij}, \qquad \forall(i,j)\in\mathcal{A}, \forall k\in\mathcal{K}, \forall\omega\in\Omega, \tag{10}$$

$$x_{ij}^k(\omega) \ge 0, \qquad \forall(i,j)\in\mathcal{A}, \forall k\in\mathcal{K}, \forall\omega\in\Omega, \tag{11}$$

$$y_{ij}\in\{0,1\}, \qquad \forall(i,j)\in\mathcal{A}, \tag{12}$$

where

$$w_i^k(\omega) = \begin{cases} d^k(\omega), & \text{if } i = O(k), \\ -d^k(\omega), & \text{if } i = D(k), \\ 0, & \text{otherwise.} \end{cases}$$

### 3.2. The generator

The description of the stochastic generator follows the same structure as in the previous section: *first*, outline its design and work flow, *second*, list its functionalities, *third*, describe its use and provide an assessment of its performance.

**3.2.1. Design and workflow.** The program initially reads the deterministic MCFNDP instance based on which the stochastic MCFNDP instance will be generated. It also reads which subsets of parameters among the following will be randomized (i.e., will vary between scenarios): (i) demands, (ii) total capacities of arcs, (iii) commodity-specific capacities of arcs, (iv) variable costs over arcs and commodities. If requested by the user, the program proceeds to generating target moments and correlations. Otherwise, target moments and correlations are read from files.

When generating target moments with moment generation parameters $\alpha$ and $\beta$, if the distributional characterization specified for the target moments is *uniform*, then each individual randomized parameter is assumed to have first four moments equal to those of a *uniform*$(a, b)$ distribution where $a = D - (\alpha \cdot D)$, $b = D + (\beta \cdot D)$. If the distributional characterization is *triangular*, then each randomized individual parameter is assumed to have first four moments equal to those of a *triangular*$(a, b, c)$ distribution where, $a = D - (\alpha \cdot D)$, $b = D + (\beta \cdot D), c = D$. Here, $D$ is the value taken by the parameter in the base deterministic MCFNDP instance and $\alpha \in [0, 1)$, $\beta \in [0, \infty)$. The distributions are symmetric around $D$ when $\alpha = \beta$.

When the generation of target moments and correlations is requested, the values of target correlations are specified in the command line for whole blocks of the correlation matrix. For example, a value specified for the block of correlations between demands and commodity-specific capacities will be shared by all correlations between parameters describing demands and parameters describing commodity-specific capacities. Similarly, a value specified for the correlations within demands will be shared by all correlations among parameters describing demands. (Of course, all parameters will have a self-correlation equal to one.) By default, correlations within and between blocks of parameters are equated to zero.

Once all target moments and correlations have been generated, the algorithm described in Section 2.5 of Høyland et al. (2003) is supplied with the matrices of target moments and correlations and with the values of the **HKW-algorithm options** (see next subsection for details). From a starting set of scenarios (determined quasi-randomly or fed back from

a previous run), the algorithm iteratively applies cubic transformations (to alter moments) and Cholesky factor transformations (to alter correlations) until a sufficiently close match with target moments and correlations is attained. If the algorithm fails to achieve sufficient convergence within the prescribed number of iterations, then a new attempt is made from new starting scenarios.

The feasibility of each scenario returned by the HKW algorithm is tested by verifying with either the CPLEX commercial solver or the HIGHS free, open-access solver (Hall and Galabova 2025, Huangfu and Hall 2018) if a solution to the second-stage problem exists under the given scenario when all arcs are open. CPLEX and HIGHS perform equally well. Infeasible scenarios are rejected and the numbers of tested and rejected scenarios are displayed on screen. This ensures that every retained scenario is compatible with at least one set of admissible first stage values. (Of course, increasing the number of rejections is expected to enlarge the gap between realized moments and correlations and target moments and correlations.) The generator does not ensure relatively complete recourse of the generated problem instances (i.e., that for every set of admissible first stage values, all scenarios be feasible).

Retained scenarios are written to the specified output file. In this file, complete deterministic-like problem instances, one for each scenario, are superposed and written under the same format as that used to supply the base deterministic input instance. Each such instance associated to a scenario is preceded by a header stating the scenario number. Proceeding in this manner involves writing to file more information than strictly necessary to describe the scenarios since the non-randomized parameters appearing in the base instance are repeated for each superposed instance. However, the small inconvenience caused by the additional volume and time that are required is outweighed by allowing the user to easily grasp, read and retrieve in its entirety the first and second-stage information associated with a scenario.

**3.2.2. Available functionalities.** The generator offers the following functionalities. Settings can be left at default values or specified in the command line.

***Basic options:***

- Name and format of the input file supplying the deterministic MCFNDP instance based on which scenarios are calculated.

- Which subset of parameters of the deterministic MCFNDP instance are to be randomized (one or more among the following: demand, total capacities of arcs, commodity-specific capacities of arcs, fixed costs over arcs, variable costs over arcs and commodities).
- Name of output file containing generated scenarios of the MCFNDP.
- Switches indicating if target moments and correlations are to be read from files or generated.
- Name of file holding target moments (if target moments are generated they will be written to this file; otherwise, they will be read from this file).
- Name of file holding target correlations (if target correlations are generated, they will be written to this file; otherwise, they will be read from this file).
- The following options are active only when generations of target moments and correlations are specified:
  - Distributional characterization of the first four target moments for the randomized parameters (either uniform or triangular).
  - Moment generation parameters $\alpha$ and $\beta$.
  - Value of target linear correlation shared in each block of the matrix of linear correlations.

***HKW-algorithm options:***

- Number of scenarios $|\Omega|$.
- Maximum error allowed when matching moments (scaled to $Variance = 1$).
- Maximum error allowed when matching correlations (scaled to $Variance = 1$).
- Level of verbosity displayed by the program.
- Maximum number of attempts to generate scenarios from new starting values.
- Maximum number of iterations in each attempt of the algorithm.
- Seed and stream of the pseudo-random number generator.
- Name of input file containing probabilities (optional: if absent, scenarios are assumed equiprobable).
- Name of output file where resulting matrix of raw scenarios is written in HKW format.
- Name of input file where to read a matrix of raw scenarios saved in HKW format in a previous run (optional: if present, content will be used as starting values, otherwise, starting values will be sampled).

**3.2.3. Use and performance.** The generator is available in the depot located at **https://github.com/larseeri/stoch-MCFNDP-gener**. The included **README.md** file supplies detailed information about building and running the generator from the Linux command line and running the generator from the included Docker container. Help is displayed on screen upon request or automatically in case of erroneous command line statement. For example, the following command line instruction:

```
/stoch_gen -I instB.std -F G -S 3 -G -T U -A 0.25 -B 0.3 -K -XDD 0.5 -XDA -0.3 -XAA 0.7
```

specifies that (i) the base deterministic MCFNDP instance is stored in file *instB.std*, (ii) under generic STD format (-F G), (iii) parameters describing demands and total capacities of arcs must be randomized through scenarios (-S 3), (iv) target moments must be generated rather than read from file (-G), (v) distributional characteristic of target moments is uniform (-T U), (vi) target moment generation parameters $\alpha$ and $\beta$ are equal to 0.25 (-A 0.25) and 0.3 (-B 0.3), (vii) target correlations are directly specified in blocks rather than read from file (-K), (viii) target correlations within demand parameters are all equal to 0.5 except for self-correlations being equal to 1.0 (-XDD 0.5), between parameters describing respectively demands and total capacities of arcs are all equal to -0.3 (-XDA -0.3), and within parameters describing total capacities of arcs are all equal to 0.7 except for self-correlations being equal to 1.0 (-XAA 0.7). All other options are set to their default values.

The computing speed achieved by the generator is high in view of its intended low-repetitions use. Table 1 reports scenario generation and feasibility verification times (columns *gener.* and *check*, respectively) when requesting specified numbers of scenarios (col. *requ.scen.*) for particular instances of the historical R series of MCFNDPs (col. *instance*) (CommaLAB 2023a). The latter range from a small r05.3 (10 nodes, 60 arcs, 25 commodities) to the largest r18.3 (20 nodes, 315 arcs, 200 commodities). Demands ($D$) and overall arc capacities ($A$) are randomized between scenarios. Target correlations among demands, among capacities and between capacities and demands are as specified in columns *corr. DD*, *corr. AA* and *corr. AD* respectively. Target moments are those of a uniform distributions with $\alpha = \beta = 0.25$. Hence, each randomized parameter is assumed to obey a uniform distribution whose bounds are respectively 25% below and 25% above the values taken by the corresponding parameters in the base deterministic problem. Column *feas. scen.* reports the number of scenarios that are feasible. Column *rand. elems* indicates

the number of parameters that are randomized between scenarios. Notice that the number of requested scenarios must usually exceed the latter if target moments and correlations must be satisfied.

| instance | rand. elems | requ. scen. | corr. DD | corr. AA | corr. AD | gener. (sec) | check (sec) | feas. scen. |
|---|---|---|---|---|---|---|---|---|
| r05.3 | 85 | 10000 | 0.0 | 0.0 | 0.0 | 2.0 | 13.0 | 10000 |
| r05.3 | 85 | 5000 | 0.0 | 0.0 | 0.0 | 0.5 | 7.7 | 5000 |
| r05.3 | 85 | 1000 | 0.0 | 0.0 | 0.0 | 0.1 | 1.6 | 1000 |
| r05.3 | 85 | 500 | 0.0 | 0.0 | 0.0 | 0.1 | 0.7 | 500 |
| r05.3 | 85 | 200 | 0.0 | 0.0 | 0.0 | 0.1 | 0.3 | 200 |
| r05.3 | 85 | 100 | 0.0 | 0.0 | 0.0 | 0.2 | 0.2 | 100 |
| r05.3 | 85 | 10000 | 0.5 | 0.5 | -0.2 | 8.0 | 6.0 | 10000 |
| r05.3 | 85 | 5000 | 0.5 | 0.5 | -0.2 | 0.9 | 7.6 | 5000 |
| r05.3 | 85 | 1000 | 0.5 | 0.5 | -0.2 | 0.2 | 1.4 | 1000 |
| r05.3 | 85 | 500 | 0.5 | 0.5 | -0.2 | 0.1 | 0.8 | 500 |
| r05.3 | 85 | 200 | 0.5 | 0.5 | -0.2 | 0.1 | 0.3 | 200 |
| r05.3 | 85 | 100 | 0.5 | 0.5 | -0.2 | 0.7 | 0.2 | 100 |
| r09.3 | 133 | 10000 | 0.0 | 0.0 | 0.0 | 3.0 | 16.0 | 10000 |
| r09.3 | 133 | 5000 | 0.0 | 0.0 | 0.0 | 0.7 | 17.8 | 5000 |
| r09.3 | 133 | 1000 | 0.0 | 0.0 | 0.0 | 0.2 | 3.5 | 1000 |
| r09.3 | 133 | 500 | 0.0 | 0.0 | 0.0 | 0.2 | 1.7 | 500 |
| r09.3 | 133 | 200 | 0.0 | 0.0 | 0.0 | 0.2 | 0.7 | 200 |
| r09.3 | 133 | 10000 | 0.5 | 0.5 | -0.2 | 9.0 | 18.0 | 10000 |
| r09.3 | 133 | 5000 | 0.5 | 0.5 | -0.2 | 1.5 | 17.9 | 5000 |
| r09.3 | 133 | 1000 | 0.5 | 0.5 | -0.2 | 0.3 | 3.5 | 1000 |
| r09.3 | 133 | 500 | 0.5 | 0.5 | -0.2 | 0.2 | 1.7 | 500 |
| r09.3 | 133 | 200 | 0.5 | 0.5 | -0.2 | 0.2 | 0.7 | 200 |
| r14.3 | 320 | 10000 | 0.0 | 0.0 | 0.0 | 20.0 | 184.0 | 10000 |
| r14.3 | 320 | 5000 | 0.0 | 0.0 | 0.0 | 2.4 | 95.8 | 5000 |
| r14.3 | 320 | 1000 | 0.0 | 0.0 | 0.0 | 1.3 | 18.8 | 1000 |
| r14.3 | 320 | 500 | 0.0 | 0.0 | 0.0 | 1.4 | 9.4 | 500 |
| r14.3 | 320 | 10000 | 0.5 | 0.5 | -0.2 | 82.0 | 162.0 | 10000 |
| r14.3 | 320 | 5000 | 0.5 | 0.5 | -0.2 | 5.4 | 92.1 | 5000 |
| r14.3 | 320 | 1000 | 0.5 | 0.5 | -0.2 | 1.6 | 18.5 | 1000 |
| r14.3 | 320 | 500 | 0.5 | 0.5 | -0.2 | 1.8 | 9.0 | 500 |
| r18.3 | 515 | 10000 | 0.0 | 0.0 | 0.0 | 51 | 529 | 10000 |
| r18.3 | 515 | 5000 | 0.0 | 0.0 | 0.0 | 6.3 | 270.7 | 5000 |
| r18.3 | 515 | 1000 | 0.0 | 0.0 | 0.0 | 4.0 | 57.8 | 1000 |
| r18.3 | 515 | 10000 | 0.5 | 0.5 | -0.2 | 160.0 | 470.0 | 10000 |
| r18.3 | 515 | 5000 | 0.5 | 0.5 | -0.2 | 13.2 | 274.0 | 5000 |
| r18.3 | 515 | 1000 | 0.5 | 0.5 | -0.2 | 5.0 | 54.0 | 1000 |

**Table 1 Scenario computation and verification times**

## 4. Conclusion

We introduced two flexible, high-speed generators capable of simulating wide ranges of deterministic and stochastic MCFNDP instances. We believe that they constitute highly effective and practical tools and hope that they may in the future facilitate systematic experimentation and foster reproducibility and comparability of published research. The

generator of deterministic MCFNDPs extends, modernizes and extensively documents the existing Mulgen generator so as to turn it into a more flexible, capable and readily usable instrument that stands on its own and that can also be used to supply basic deterministic instances to the stochastic generator.

The generator of stochastic MCFNDPs is completely new. It synthesizes the joint probability distribution governing a subset of parameters of a specified deterministic MCFNDP instance that are identified by the user as varying stochastically, using a moment-matching algorithm and based on primitive, easily interpretable requirements specified by the user. The basic deterministic instance can arise generically from any source. It suffices that it is supplied in one of the admissible formats.

As a natural extension of this line of research, we envision the creation of similar generators simulating instances of other standardized problem families of high interest. In terms of relevance and reliability, the advantages of collecting evidence from a range of problem families when assessing and comparing the performance achieved by exact and heuristic solution methods are manifest.

## Acknowledgments

This research was funded by the Canadian National Railway Company (CN) Chair in Optimization of Railway Operations at Université de Montréal, the Canada Research Chair program [950-232244] and IVADO fundamental research program on integrated machine learning and optimization for decision making under uncertainty. We are grateful to Antonio Frangioni for his stewardship of the Canad instances and the Mulgen generator on the CommaLAB website. We are also grateful for important insights provided by verifications and testing conducted by Ammar Metnani. The original code of the Mulgen generator was written by Bernard Gendron who passed away at a too young age. We are deeply grateful for the invaluable discussions on network design we had the privilege to have with him.

## Appendix. Existing Methods for the Construction of Scenarios

We overview the methods proposed in the literature on linear two-stage stochastic programming for constructing scenarios. We aim to identify those that are suited for the purpose of synthesizing a distribution and, among the latter, discuss usability in our particular context of application. Detailed reviews of the literature on scenario construction for the purposes of stochastic programming are available in Löhndorf (2016), Bounitsis et al. (2022), Kaut (2021), Keutchayan et al. (2023), Rujeerapaiboon et al. (2022), Bertsimas and Mundru (2022). We distinguish the following operating principles and classify the existing methods accordingly.

*Sampling scenarios.* Surveys of the literature on random and quasi-random sampling of scenarios are available in Shapiro (2003), Glasserman (2004), Bayraksan and Morton (2011), de Mello and Bayraksan (2014). Later advances are available in Leövey and Römisch (2015), Löhndorf (2016) (with stratification through Voronoi sampling). Methods in this class sample among existing scenarios included in an assumed known probability distribution. They are therefore inadequate for the purpose of synthesizing a distribution based on primitive requirements.

*Clustering scenarios.* Existing scenarios included in an assumed known probability distribution are clustered with k-means, k-medians, k-medoids, either based on values of parameters appearing in scenarios or on values of second stage outcomes associated to individual scenarios (Keutchayan et al. 2023, Hewitt et al. 2022, Abouelrous et al. 2022). These methods are inadequate for our purposes as they operate on scenarios of an assumed known probability distribution.

*Minimization of probabilistic distance.* This area of research is substantial and currently the most active. It also extends to multistage stochastic programming and scenario trees. Those topics exceed our scope. Surveys are available in Löhndorf (2016), Rujeerapaiboon et al. (2022), Bertsimas and Mundru (2022). Theoretical advances and methodological overviews are available from Dupačová et al. (2003), Heitsch and Römisch (2003), Heitsch and Römisch (2007), Henrion et al. (2008, 2009), Pflug and Pichler (2015), Henrion and Römisch (2022), Rujeerapaiboon et al. (2022). Large scale, industrial applications are presented in Li and Floudas (2014, 2016), Rujeerapaiboon et al. (2022), Bertsimas and Mundru (2022), Abouelrous et al. (2022). Methods in this class aim to minimize or show convergence of the Wasserstein (a.k.a. Kantorovich-Rubinstein) or Fortet-Mourier probabilistic distances or an upper bound thereof between a known, exact distribution and a new, approximate distribution resulting from a constructed set of scenarios. There is a partial overlap between this class and the preceding ones as sampling and clustering may be involved in building the sets of scenarios aimed at controlling the probabilistic distance (see for instance Dupačová et al. 2003, Heitsch and Römisch 2003, Heitsch and Römisch 2007, Henrion et al. 2008, 2009, Henrion and Römisch 2022, Rujeerapaiboon et al. 2022, Bertsimas and Mundru 2022, Abouelrous et al. 2022). Methods in this class control the probabilistic distance either (i) between the exact and approximate distributions of the second stage parameters while relying on stability analysis to characterize the probabilistic

distance between the resulting outputs, or (ii) directly between the distributions of the outputs achieved under the exact and approximate distributions of the parameters. Whereas it would be conceivable to devise constructive methods based on primitive requirements aiming to minimize probabilistic distances, this would exceed the scope of our proposition.

*Matching moments, correlations and distribution functions.* Methods in this class build scenarios by matching target moments and correlations or matching target distributions for the stochastic parameters (see Fleishman 1978, Høyland and Wallace 2001, Høyland et al. 2003, Mehrotra and Papp 2013). This type of method is applicable for our purposes as it is conceivable to state target moments and correlations for the stochastic parameters of the MCFNDP. A large capacity, high-speed computational application has been made available by one of the authors of Høyland et al. (2003) and the latter has been used successfully to build scenarios for MCFNDPs in a number of publications (see, e.g., Crainic et al. 2011, 2021b). This application requires supplying targets about first four moments and linear correlations for all stochastic parameters in the MCFNDP at hand. We saw in Section 3.2.1 how this information may be derived from a small set of assumptions. Using the more general method proposed in Mehrotra and Papp (2013) is also conceivable. In contrast with Høyland et al. (2003), Mehrotra and Papp (2013) can match the distribution function rather than only first four moments and linear correlations. However, its greater informational prerequisites are difficult to postulate convincingly in practice and its computational tractability stands at a prototypical stage. These reasons lead us to prefer the method of Høyland et al. (2003) to that of Mehrotra and Papp (2013). Kaut (2021) proposes a MILP solution to the moment-matching problem that might be viewed as an alternative to the iterative NLP algorithm of Høyland et al. (2003). However, we prefer the latter as the former is prototypical. Moment-matching methods can also be joined with clustering methods (Li and Zhu 2016), principal component analysis (Chopra and Selvamuthu 2020), distribution matching (Calfa et al. 2014, Bounitsis et al. 2022). These variants require a detailed knowledge of underlying scenarios and distribution that is unavailable in our context of application.

*Copula sampling.* A number of scenario selection methods account for dependencies in the assumed available underlying joint distribution through copula sampling and may combine it with matching of moments, matching of marginal distributions or clustering to account for marginal distributions (see Sutiene and Pranevicius 2007, Kaut and Wallace 2011, Kaut

2014, 2015, Qiu et al. 2019). In our context of application, while it would be conceivable to select a copula based on primitive assumptions and conduct sampling from it, this would presuppose a level of knowledge about the dependencies among the stochastic parameters of the MCFNDP that may be difficult to justify in practice.

## References

Abouelrous, A., Gabor, A. F., and Zhang, Y. Optimizing the inventory and fulfillment of an omnichannel retailer: a stochastic approach with scenario clustering. *Computers & Industrial Engineering*, 173: 108723, 2022.

Bayraksan, G. and Morton, D. P. A sequential sampling procedure for stochastic programming. *Operations Research*, 59(4):898–913, 2011.

Bertsekas, D. P. *Linear Network Optimization: Algorithms and Codes*. MIT Press, Cambridge, 1991.

Bertsimas, D. and Mundru, N. Optimization-based scenario reduction for data-driven two-stage stochastic optimization. *Operations Research*, 71(4):1343–1361, 2022.

Bonnet, L., Belieres, S., Hewitt, M., and Ngueveu, S. U. A Complex Network Analysis Approach for Generating Realistic Instances of the Scheduled Service Network Design Problem. *Hal open science hal-05345829*, November 2025.

Bounitsis, G. L., Papageorgiou, L. G., and Charitopoulos, V. M. Data-driven scenario generation for two-stage stochastic programming. *Chemical Engineering Research and Design*, 187:206–224, 2022.

Calfa, B., Agarwal, A., Grossmann, I., and Wassick, J. Data-driven multi-stage scenario tree generation via statistical property and distribution matching. *Computers & Chemical Engineering*, 68:7–23, 2014.

Chopra, I. and Selvamuthu, D. Scenario generation in stochastic programming using principal component analysis based on moment-matching approach. *OPSEARCH*, 57(1):190–201, 2020.

Chouman, M., Crainic, T. G., and Gendron, B. Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science*, 51:650–667, 2016.

CommaLAB. The Canad problems. `https://commalab.di.unipi.it/datasets/mmcf/#Canad`, 2023a. Accessed: 2023-11-28.

CommaLAB. Mulgen generator. `https://commalab.di.unipi.it/files/Data/MMCF/Canad.tgz/`, 2023b. Accessed: 2023-11-28.

Crainic, T. G., Frangioni, A., and Gendron, B. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112:73–99, 2001.

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

Crainic, T. G., Gendreau, M., and Gendron, B. Fixed-charge network design problems. In Crainic, T. G., Gendreau, M., and Gendron, B., editors, *Network Design with Applications to Transportation and Logistics*, pages 15–28. Springer, 2021a.

Crainic, T. G., Hewitt, M., Maggioni, F., and Rei, W. Partial Benders decomposition: General methodology and application to stochastic network design. *Transportation Science*, 55(2):414–435, 2021b.

de Mello, T. H. and Bayraksan, G. Monte Carlo sampling-based methods for stochastic optimization. *Surveys in Operations Research and Management Science*, 19(1):56–85, 2014.

Dupačová, J., Gröwe-Kuska, N., and Römisch, W. Scenario reduction in stochastic programming. *Mathematical Programming*, 95(3):493–511, 2003.

Fleishman, A. I. A method for simulating non-normal distributions. *Psychometrika*, 43(4):521–532, 1978.

Gendron, B., Crainic, T. G., and Frangioni, A. Multicommodity capacitated network design. In *Telecommunications network planning*, pages 1–19. Springer, 1999.

Glasserman, P. *Monte Carlo methods in financial engineering*. Springer, New York, 2004.

Hall, J. and Galabova, I. Highs - high performance software for linear optimization. `https://highs.dev/#top`, 2025. Accessed: 2025-10-14.

Heitsch, H. and Römisch, W. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2):187–206, 2003.

Heitsch, H. and Römisch, W. A note on scenario reduction for two-stage stochastic programs. *Operations Research Letters*, 35(6):731–738, 2007.

Henrion, R. and Römisch, W. Problem-based optimal scenario generation and reduction in stochastic programming. *Mathematical Programming*, 191(1):183–205, 2022.

Henrion, R., Küchler, C., and Römisch, W. Scenario reduction in stochastic programming with respect to discrepancy distances. *Computational Optimization and Applications*, 43(1):67–93, 2009.

Henrion, R., Küchler, C., and Römisch, W. Discrepancy distances and scenario reduction in two-stage stochastic mixed-integer programming. *Journal of Industrial and Management Optimization*, 4(2): 363–384, 2008.

Hewitt, M., Nemhauser, G. L., and Savelsbergh, M. W. P. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, 22:314–325, 2010.

Hewitt, M., Ortmann, J., and Rei, W. Decision-based scenario clustering for decision-making under uncertainty. *Annals of Operations Research*, 315(2):747–771, 2022.

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

Huangfu, Q. and Hall, J. A. J. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, Mar 2018.

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

Kaut, M. A copula-based heuristic for scenario generation. *Computational Management Science*, 11(4): 503–516, 2014.

Kaut, M. Erratum to: A copula-based heuristic for scenario generation. *Computational Management Science*, 12(2):341–343, 2015.

Kaut, M. Scenario generation by selection from historical data. *Computational Management Science*, 18(3): 411–429, 2021.

Kaut, M. and Wallace, S. W. Shape-based scenario generation using copulas. *Computational Management Science*, 8(1):181–199, 2011.

Keutchayan, J., Ortmann, J., and Rei, W. Problem-driven scenario clustering in stochastic optimization. *Computational Management Science*, 20(1):13, 2023.

Lemire, D. Testing non-cryptographic random number generators: my results. `https://lemire.me/blog/2017/08/22/testing-non-cryptographic-random-number-generators-my-results/`, 2023. Accessed: 2023-11-28.

Leövey, H. and Römisch, W. Quasi-Monte Carlo methods for linear two-stage stochastic programming problems. *Mathematical Programming*, 151(1):315–345, 2015.

Li, J. and Zhu, D. Combination of moment-matching, cholesky and clustering methods to approximate discrete probability distribution of multiple wind farms. *IET Renewable Power Generation*, 10:1450–1458(8), 2016.

Li, Z. and Floudas, C. A. Optimal scenario reduction framework based on distance of uncertainty distribution and output performance: I. single reduction via mixed integer linear optimization. *Computers & Chemical Engineering*, 70:50–66, 2014. Manfred Morari Special Issue.

Li, Z. and Floudas, C. A. Optimal scenario reduction framework based on distance of uncertainty distribution and output performance: II. sequential reduction. *Computers & Chemical Engineering*, 84:599–610, 2016.

Löhndorf, N. An empirical analysis of scenario generation methods for stochastic optimization. *European Journal of Operational Research*, 255(1):121–132, 2016.

Mehrotra, S. and Papp, D. Generating moment matching scenarios using optimization techniques. *SIAM Journal on Optimization*, 23(2):963–999, 2013.

O'Neill, M. PCG, a family of better random number generators. `https://www.pcg-random.org/`, 2023. Accessed: 2023-11-28.

Pflug, G. C. and Pichler, A. Dynamic generation of scenario trees. *Computational Optimization and Applications*, 62(3):641–668, 2015.

Qiu, Y., Li, Q., Pan, Y., Yang, H., and Chen, W. A scenario generation method based on the mixture vine copula and its application in the power system with wind/hydrogen production. *International Journal of Hydrogen Energy*, 44(11):5162–5170, 2019. The 6th International Conference on Energy, Engineering and Environmental Engineering.

Rujeerapaiboon, N., Schindler, K., Kuhn, D., and Wiesemann, W. Scenario reduction revisited: fundamental limits and guarantees. *Mathematical Programming*, 191(1):207–242, 2022.

Shapiro, A. Monte Carlo sampling methods. In *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*, pages 353–425. Elsevier, 2003.

Sutiene, K. and Pranevicius, H. Scenario generation employing copulas. In *World Congress on Engineering*, 2007.