Michal Kaut; Arnt-Gunnar Lium

Scenario generation with distribution functions and correlations

Kybernetika, Vol. 50 (2014), No. 6, 1049–1064

Persistent URL: http://dml.cz/dmlcz/144122

KYBERNETIKA — VOLUME *50* (2014), NUMBER 6, PAGES 1049–1064

SCENARIO GENERATION WITH DISTRIBUTION FUNCTIONS AND CORRELATIONS

Michal Kaut and Arnt-Gunnar Lium

In this paper, we present a method for generating scenarios for two-stage stochastic programs, using multivariate distributions specified by their marginal distributions and the correlation matrix. The margins are described by their cumulative distribution functions and we allow each margin to be of different type. We demonstrate the method on a model from stochastic service network design and show that it improves the stability of the scenario-generation process, compared to both sampling and a method that matches moments and correlations.

Keywords: stochastic programming, scenario generation, moment matching, distribution functions, service network design

Classification: 90C15, 62G30, 62H20

INTRODUCTION

Stochastic programming is an area of optimization that deals with problems solved under uncertainty. In particular, a two-stage stochastic-programming model is an optimization problem of the form

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) + \mathbb{E}\left[g(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{\xi}), \boldsymbol{\xi})\right] \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X} \\ & \boldsymbol{y}(\boldsymbol{\xi}) \in \mathcal{Y}(\boldsymbol{x}, \boldsymbol{\xi})\,, \end{aligned}$$

where $\boldsymbol{\xi}$ is a random vector, $\boldsymbol{x}$ are the first-stage decisions, done before observing the value of $\boldsymbol{\xi}$, and $\boldsymbol{y}(\boldsymbol{\xi})$ are second-stage decisions, done after we have observed $\boldsymbol{\xi}$. Interested readers are referred to Kall and Wallace [12] or Birge and Louveaux [1].

For most practical applications, it is not possible to solve this problem exactly and the distribution of $\boldsymbol{\xi}$ has to be approximated by a discrete set of samples – usually called *scenarios* – $\boldsymbol{\xi}_s$, with probabilities P_s:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) + \sum_s P_s\, g(\boldsymbol{x}, \boldsymbol{y}_s, \boldsymbol{\xi}_s) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X} \\ & \boldsymbol{y}_s \in \mathcal{Y}(\boldsymbol{x}, \boldsymbol{\xi}_s) \quad \forall s\,. \end{aligned}$$

DOI: 10.14736/kyb-2014-6-1049

Clearly, the quality of the approximation has a significant impact on the quality of the obtained solutions. For this reason, the discretization – or *scenario generation*, as it is usually referred to – has enjoyed an increasing attention of both researchers and practitioners and established itself as an important part of the stochastic programming framework. During this time, several different scenario-generation methods have been presented, each with its strengths and weaknesses; examples include optimal discretization [18], property matching [10], or scenario reduction methods [3, 6, 7]. For an overview of all but the most recent methods, see Dupačová et all. [2].

It should be noted that the presented method could be considered an alternative to sampling also in other contexts where one needs to approximate a multi-variate integral with discrete samples. This is, however, out of the scope of this paper.

In this paper, we focus on the property-matching method [10], in particular on the moment-matching heuristic from Hoyland et all. [11]. There, the scenarios are generated to match the first four moments of the marginal distributions, plus the correlation matrix. This method has been successfully applied in many areas, including financial optimization [5, 22], electricity markets [4], supply chain modelling [19, 20, 23], or service network design [21].

On the other hand, there are situations where using moments does not provide sufficient control over the marginal distributions, leading to poor quality of the generated scenarios. We present a method that generalizes the algorithm from Hoyland et al. [11] in the sense that the margins can now be described directly by their distributions. Both continuous and discrete distributions are supported, even though the use of discrete distribution(s) can lead to larger errors in correlations. The different distributions can be freely mixed, i. e. each margin can have a different distribution, or it can be specified by its moments.

Throughout the paper, we assume that the target distributions, and the correlation matrix, are known. In practice, these would be obtained by assuming distribution families for the marginal distributions (possibly different for each margin) and estimating their parameters using some statistical model and/or data. An alternative approach is to use the empirical CDFs of the data, possibly interpolated to make them continuous, as the target CDFs in the method. The latter is appropriate in cases where the data represent all the information we have about the distribution.

The rest of the paper is organized as follows: we start by summarizing the scenario-generation algorithm by Hoyland et all. [11] In Section 2, we present our new algorithm, which is then in Section 3 compared to the original version as well as sampling, using a model from stochastic service network design.

1. SHORT SUMMARY OF THE MOMENT-MATCHING ALGORITHM

The algorithm generates one-period scenario trees[1] from multivariate distributions specified by their correlation matrix and the first four moments (mean, variance, skewness and kurtosis) of the marginal distributions. It achieves this by using two transformations, one correcting the moments and the other correcting the correlations. Since each transformation distorts the results of the other one, they are repeated iteratively, al-

[1]Note that we use the word 'tree', even if we generate only one-period scenarios.

ternating between the two. The algorithm stops on convergence, i. e. when the error of both the moments and correlations is below a given threshold, or when a maximal number of iterations is reached. The overall structure of the algorithm is thus

```
Generate initial sample, set i ← 0.
do
  if error-of-correlations > MaxErrorCorr
    Correct correlations.
  for j = 1, ..., n
    if error-of-moments(j) > MaxErrorMom
      Correct moments of margin j.
  set i ← i + 1.
while (error-of-correlations > MaxErrorCorr or error-of-moments > MaxErrorMoms)
      and (i < MaxNumIters).
```

Correlations are corrected using a variant of the standard method for generating correlated normal variates, based on Cholesky decomposition of the correlation matrix. In short, it uses the fact if we have random vector $\tilde{\boldsymbol{X}}$ with correlation matrix $C_p = L_p L_p^T$ and another correlation matrix $C = LL^T$, then the random vector $\tilde{\boldsymbol{Y}} = LL_p^{-1}\tilde{\boldsymbol{X}}$ has correlation matrix equal to C. This is true regardless of the distribution of $\tilde{\boldsymbol{X}}$, as long as C_p is non-singular and the marginal distributions are studentized, i. e. have zero mean and variance equal to one; we will come back to this in Section 2.3.

Moments of the marginal distributions are corrected one at a time, using a cubic transformation

$$\boldsymbol{y} = a + b\boldsymbol{x} + c\boldsymbol{x}^2 + d\boldsymbol{x}^3\,. \tag{1}$$

Finding the parameters a, b, c, d is the most challenging part of the algorithm, as it requires solving a system of four implicit nonlinear equations with four unknowns – see Hoyland et all. [11] for details.

There are several ways of generating the initial sample. The easiest option is sampling, if one has some data or an estimated distribution to sample from. An alternative is to sample from, or use fixed discretizations of, some parametric distribution, such as the standard normal distribution.

1.1. Limitations of using moments

Even though the method works in many practical situations, controlling moments using only margins has some limitations. Probably the most significant is the lack of control over the support of the generated scenarios, which causes problems in several different ways. Firstly, it makes it difficult (if not impossible) to generate scenarios for stochastic parameters with values in a specific intervals, such as positive values (demand, supply, etc.) or values that have to be within the $(0, 1)$ interval (c.f. efficiency of some machine or process). In all these cases, if the distribution is such that there is significant probability of being close to the boundary, it can be expected that the method will generate values on the wrong side of the boundary as well.

The lack of control of the support might also create problems in a more subtle way: consider the situation where the optimization model is sensitive to the minimal and/or

maximal value of some of the random variables. Without control of the supports, the extreme points might differ significantly if we run the algorithm several times, leading to instability of solutions of the optimization model – see Kaut and Wallace [13] or Heitsch et all. [8] for a definition and discussion of stability of scenario-generation methods.

Finally, if we happen to know the marginal distributions, using only the first four moments means throwing away a lot of information. In particular, if we have a large data set that we use to estimate the distribution, we should get better results by using the empirical distribution functions from the data, instead of using only the first four moments – assuming that the data set represents all we know about the distribution and that our method uses this information correctly. This observation serves as a motivation for the new method, which replaces the moments by cumulative distribution functions.

2. THE NEW ALGORITHM

The important thing to realize is that correcting moments is only one possible way of improving the marginal distributions – just as correcting correlations is one possible way of improving the dependence between the margins. In other words, the algorithm from Section 1 can be re-written as:

```
Generate initial sample.
set i ← 0.
do
  if error-of-correlations > MaxErrorCorr
    Correct correlations.
  for j = 1, ..., n
    if error-of-margin (j) > MaxErrorMarg
      Correct distribution of margin j.
  set i ← i + 1.
while (error-of-correlations > MaxErrorCorr or error-of-margins > MaxErrorMargs)
      and (i < MaxNumIters).
```

In this section, we describe how the margins can be controlled using their cumulative distributions functions (CDFs), instead of moments. While the method was originally developed for continuous distributions, we provide also an extension that can deal with discrete distributions as well – see Section 2.4. Since the corrections of margins are done independently for each margin, we can combine margins corrected by CDFs, moments, and possibly other methods, in one set of scenarios. In the rest of the section, however, we will only discuss the margins corrected using CDFs.

We start with continuous distributions. In other words, we assume that we have, for each margin j, its continuous CDF F_j and inverse CDF F_j^{-1} – or, if the CDF is not strictly increasing, the pseudo-inverse $F^{-1}(u) = \inf\{x : F(x) \geq u\}$. Since the margins are treated separately, we describe the correction for one margin only and therefore drop the j-subscript from all elements. Hence, we use F and F^{-1} for the CDFs and $\boldsymbol{x} = (x_1, \ldots, x_S)$ for the sample from one marginal distribution.

2.1. Correcting CDFs – finding the transformation and error definition

To implement the algorithm, we need a transformation that can change a sample to make it 'more similar' the target distribution and, at the same time, change the correlations as little as possible, allowing the algorithm to converge. In addition, we need a measure of the error of a current sample, i. e. its distance from the target distribution. Since we want to be able to mix margins with different distributions, the scale of the error should be independent of the type of distribution involved.

To achieve this, we propose a correction based on a fixed 'optimal' discretization, where the points are spread evenly in terms of percentiles. To correct a margin, we transform its distribution to the optimal discretization, while the error is computed as a distance from this discretization. The correction of margins then works as follows: we transform the margins to the standard uniform (U(0, 1)) distribution and spread the values uniformly on the interval $(0, 1)$. With S values per margin, this means placing the values into the centers of intervals of length $1/S$, i. e. at positions $\frac{2s-1}{2S}$, $s = 1 \dots S$. This can be also written as

$$u_s = F^e_{\boldsymbol{x}}(x_s) := \frac{2\,\mathrm{rank}(x_s, \boldsymbol{x}) - 1}{2S}, \quad s = 1 \dots S\,, \tag{2}$$

where $\boldsymbol{x}$ is the sample of margin's values and $\mathrm{rank}(x_s, \boldsymbol{x})$ denotes the rank of x_s in $\boldsymbol{x}$ (sometimes denoted $\mathrm{ord}(x_s, \boldsymbol{x})$), with minimum having the rank of one and ties resolved using the scenario index (so we have ordinal ranking). The function $F^e_{\boldsymbol{x}}$ can be viewed as a kind of empirical CDF, except that it is only defined on the values x_s from $\boldsymbol{x}$. Note also that $F^e_{\boldsymbol{x}}$ does not assign zero or one to any value.

Once we have the fixed U(0, 1) vector $\boldsymbol{u} = (u_1, \dots, u_S)$, we apply the inverse CDF to get the target distribution,

$$x_s = F^{-1}(u_s), \quad s = 1, \dots, S\,. \tag{3}$$

The whole correction of vector $\boldsymbol{x}$ can be thus written as

$$x_s \leftarrow F^{-1}\big(F^e_{\boldsymbol{x}}(x_s)\big), \quad s = 1, \dots, S\,, \tag{4}$$

or, in terms of ordered values $x_{(s)} \in \boldsymbol{x}$, $x_{(1)} \le x_{(2)} \le \dots \le x_{(S)}$,

$$x_{(s)} \leftarrow F^{-1}\big(\tfrac{2s-1}{2S}\big), \quad s = 1 \dots S\,.$$

Obviously, this transformation changes the correlations. On the other hand, both F_e and F^{-1} are rank-preserving, so the transformation does not change rank correlations and all the related measures of concordance. The changes in correlations tend therefore to be small, allowing the overall algorithm to converge. Note also that the rank preservation means that repeating the transformation would not change $\boldsymbol{x}$, as long as the $\mathrm{rank}(\cdot, \boldsymbol{x})$ function is unique – for example by adding a requirement that if $x_s = x_t$ and $s < t$, then $\mathrm{rank}(x_s, \boldsymbol{x}) < \mathrm{rank}(x_t, \boldsymbol{x})$.

Evaluating the error of a margin

To evaluate the error (level of mismatch) of each margin, we use a measure closely

related to the margin-correcting procedure: in addition to computing the 'discretized CDF' $F^e_{\boldsymbol{x}}(x_s)$, we compute the actual CDF $F(x_s)$ and define the error as their mean-square difference:

$$\text{error of margin } \boldsymbol{x} = \sqrt{\frac{1}{S}\sum_{s=1}^{S}\big(F(x_s) - F^e_{\boldsymbol{x}}(x_s)\big)^2}\,. \tag{5}$$

Note that the error of a margin after correction is equal to zero, as $F(x_s) = F^e_{\boldsymbol{x}}(x_s) = u_s$ for all s. One could ask why we have not used the more standard Kolmogorov distance,

$$\sup_x |F(x) - F^e_{\boldsymbol{x}}(x)| = \max_{1\le s\le S} \max\Big\{\frac{\operatorname{rank}(x_s,\boldsymbol{x})}{S} - F(x_s), F(x_s) - \frac{\operatorname{rank}(x_s,\boldsymbol{x})-1}{S}\Big\}. \tag{6}$$

The reason is that the error in moments is evaluated as a mean-square error and we wanted to have a similar measure. In addition, if the target distribution is continuous, the Kolmogorov distance will never be zero – again something we get from our definition. On the other hand, the following result suggests that it should be possible to use the Kolmogorov distance as an error measure in our algorithm, possibly with $\frac{1}{2S}$ subtracted from it:

Proposition.

(i) Transformation (4) minimizes the Kolmogorov distance from the target CDF, within a class of discrete distributions with support of cardinality S.

(ii) Kolmogorov distance of vector $\boldsymbol{x}$ from the target CDF after transformation (4) is equal to $\frac{1}{2S}$.

Proof. We start with (ii), which follows directly from (3) since $F(x_s) = u_s = \frac{2\operatorname{rank}(x_s,\boldsymbol{x})-1}{2S}$ and therefore

$$\max_{1\le s\le S} \max\Big\{\frac{\operatorname{rank}(x_s,\boldsymbol{x})}{S} - F(x_s), F(x_s) - \frac{\operatorname{rank}(x_s,\boldsymbol{x})-1}{S}\Big\} = \max_{1\le s\le S} \max\Big\{\frac{1}{2S}, \frac{1}{2S}\Big\} = \frac{1}{2S}\,.$$

For (i), observe that sum of the two values in the inner max is $\frac{1}{S}$, from which follows

$$\max\Big\{\frac{\operatorname{rank}(x_s,\boldsymbol{x})}{S} - F(x_s), F(x_s) - \frac{\operatorname{rank}(x_s,\boldsymbol{x})-1}{S}\Big\} \ge \frac{1}{2S}\,.$$

Alternatively, we could say that the distance is a maximum of $2S$ values with sum equal to one, so it can not be smaller than $\frac{1}{2S}$. □

Note that, despite the result above, our approach is still very different from a full minimization of the Kolmogorov distance from Henrion et all. [9]. There, the distance of the whole multivariate distribution is minimized at once, while our approach works only on margins. Their approach should therefore provide better results, given that one has information about the multivariate distribution, i. e. multivariate CDF or historical

data. Our method, on the other hand, makes it easy to combine margins with different distributions – we can have margins with distributions of the same type but with different parameters, margins with different distributions, or even combine margins specified by CDFs with margin specified by their moments, or other properties. In addition, our method is easier to implement and much faster to run: the authors report running time of over an hour for 25 scenarios for $n = 2$, while our algorithm normally takes a couple of seconds to run, even for thousands of scenarios and much higher dimensions.

2.2. Improving convergence

Since we use a fixed discretization of the marginal distributions, the scenario-generation problem becomes much more constrained as the correct correlations can be obtained only by pairing the fixed values against each other. While this will produce more stable (less random) results, it may also have a negative impact on convergence of the overall method, i. e. on the minimal size of the correlation error we are able to obtain. In such a case, we can improve the convergence by introducing the fixed discretization gradually by replacing (2) by

$$\boldsymbol{u} = \alpha F^e_{\boldsymbol{x}}(\boldsymbol{x}) + (1-\alpha)F(\boldsymbol{x})\,. \tag{2'}$$

The complete transformation (4) then becomes

$$\boldsymbol{x} \leftarrow F^{-1}\big(\alpha F^e_{\boldsymbol{x}}(\boldsymbol{x}) + (1-\alpha)F(\boldsymbol{x})\big)\,, \tag{4'}$$

where $\alpha \in [0,1]$ is a weight increasing during the algorithm. Note that (4') defaults to identity for $\alpha = 0$ and to (4) for $\alpha = 1$.

By increasing α slowly from zero to one, the correlations have a better chance to 'settle down', before we fix the margins by reaching $\alpha = 1$. It is, however, possible that we will not be able to get the full match in correlations with $\alpha = 1$, especially when the number of scenarios is small. In such a case, we can trade the stability of the margins for better correlations and exit the loop with some $\alpha < 1$. We would thus end up with values different from the 'optimal' discretization – though it does not mean that the margins would have wrong distributions; at least for reasonably high values of α the margins would still have the right distribution in the sense that the standard tests (cf. Kolmogorov–Smirnov) would not reject the null hypothesis.

Note that even with this adjustment, the margin correction (4') is very easy to implement, as there are ready-made implementations of both the sorting algorithm (needed to get the ranks in (2)) and CDFs and their inverses for all the standard distributions. This is a significant improvement over the moment-matching approach, where implementing the cubic transformation needed strong programming skills.

2.3. Studentized margins

So far, we have ignored one issue our algorithm shares with the original moment-matching method. The Cholesky decomposition, used for correcting the correlations in the algorithm, requires studentized samples, i. e. scenarios with zero means and variances equal to one. In the case of moments, this is easily done by specifying the first two moments as zero and one, respectively, and then rescaling the results to the correct

means and variances at the very end of the algorithm, using a simple linear transformation $\tilde{\boldsymbol{y}} \leftarrow \mu + \sigma\tilde{\boldsymbol{y}}$.

We can do the same for distributions like normal, Student's t, where studentizing the margins, i. e. replacing $\tilde{\boldsymbol{y}} \leftarrow \frac{\tilde{\boldsymbol{y}}-\mu}{\sigma}$, can be done simply by changing parameters of the distribution. For uniform distributions and the empirical CDF from a data set, the same effect is achieved by scaling all the defining points.

For other distributions, however, things are a bit more complicated. It is, for example, obvious that studentizing an exponential or beta distribution leads to a distribution outside of the given family, and therefore a distributions for which we might not have a formula for the CDF and its inverse. (Though for the latter case, we can solve this by using a generalized beta distribution with four parameters). If we cannot scale the CDF, we have to scale the values generated during the algorithm to keep them studentized during the correlations-correcting step, and scaled to the target means and variances during the margins-correcting step.

2.4. Discrete distributions

Discrete distributions are a special case, as their CDFs are piece-wise constant and discrete margins typically have many scenarios sharing the same value. This creates a possibility of major changes in values while correcting the margins, which can have a negative impact on the overall convergence of the algorithm. In addition, the margin-correcting transformations may cease to be rank-preserving, damaging the convergence even further – though this can be avoided by a slight change in the $\operatorname{rank}(\cdot, \boldsymbol{x})$ function, as mentioned in Section 2.1.

To improve the convergence in the discrete case, we use a smoothed CDF, denoted by F_s. This is simply a linear interpolation of F, connecting the average of the left and right limits at each value of the discrete distribution. Below the minimum and above the maximum it converges to 0 and 1, respectively, so $F_s(x_s)$ is always inside $(0, 1)$. See Figure 1 for an example. Unfortunately, we can not just use the inverse of F_s, since this is defined only on interval $(0, 1)$ – while values outside this interval are possible in the early stages of the algorithm. The function is therefore extended to the whole real axis $\mathcal{R}$, as shown in Figure 1. For the sake of brevity, we denote this function F_s^{-1} even if it is not an exact inverse of F_s.

The smoothed versions of the distribution functions are then used similarly to the 'empirical CDF' above: $F(\boldsymbol{x})$ is replaced by $\beta F(\boldsymbol{x}) + (1-\beta)F_s(\boldsymbol{x})$, where β increases from zero at the beginning of the algorithm to one at the end. Combined with (2'), the formula for $\boldsymbol{u}$ becomes

$$\boldsymbol{u} = \alpha F^e_{\boldsymbol{x}}(\boldsymbol{x}) + (1-\alpha)\big(\beta F(\boldsymbol{x}) + (1-\beta)F_s(\boldsymbol{x})\big)\,. \tag{2''}$$

Note that the correction of correlations may cause some x_s to be outside the support of its distribution. In such a case, $F_s(x_s)$ will be outside $[0, 1]$ and the same may be true for u_s, depending on the values of α and β in (2''). This is why we have to extend the definition of F_s^{-1} on the whole $\mathcal{R}$.

The same procedure is applied to the inverse CDF, in which case (3) becomes

$$\boldsymbol{x} = \beta F^{-1}(\boldsymbol{u}) + (1-\beta)F_s^{-1}(\boldsymbol{u})\,. \tag{3''}$$

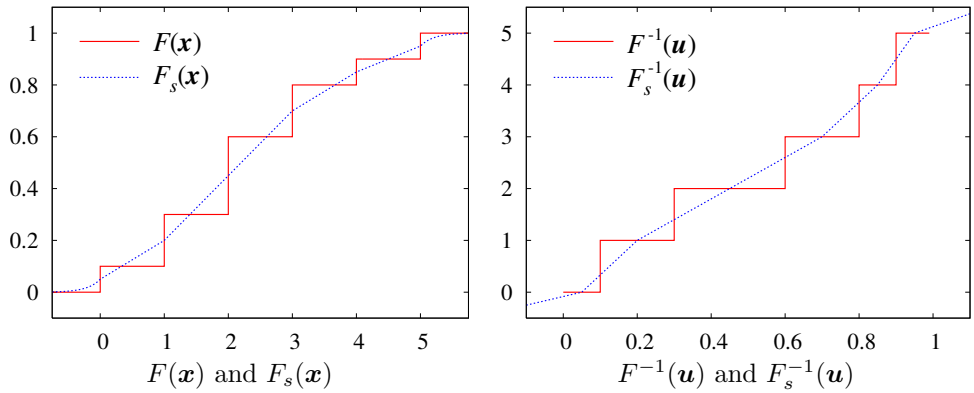

Fig. 1. Normal and smoothed version of CDF and inverse CDF for a discrete variable with values $\{0, \ldots, 5\}$ and probabilities $\{0.1, 0.2, 0.3, 0.2, 0.1, 0.1\}$.

This version of the algorithm has, so far, been tested (successfully) only on small test instances, so we are unable to provide any general guidelines for controlling the parameter β.

Note that the functions F_s and F_s^{-1} can also be useful for smoothing of the empirical distribution function in the case where we use a data set to specify the target margins. This makes sense especially if we believe that the underlying distribution is continuous.

2.5. Extensibility

Due to simple structure of the algorithm, it is easy to change and/or extend some of its functionality. Hence, if we have one or more margins with distributions requiring some special treatment, they can be handled by adding a new type of correction. For example, if the only information we have about the marginal distribution is a set of percentile values, we can use a transformation from Okunev and White [17], which stretches the margins to match a given set of percentiles.

3. TEST CASE

Quality of a scenario-generation method is judged by the quality of the solutions obtained using trees generated by the method, see [13, 16]. It follows that we can only test whether a given method is good for a a specific optimization problem; indeed, a given scenario-generation method can be very good for one optimization problem, but lead to poor decisions for another one. In this section, we therefore test the presented method on a small optimization problem. For this, we have chosen one of the problems that motivated us to create the presented method in the first place, i. e. a problem where the moment-based method does not work well.

Based on the discussion from Section 1.1, the probably most important advantage of the new model is that it can be used for stochastic parameters with values within a given interval. It would thus be natural to use such a model as our test case – but it would be difficult to compare the new method to the moment-matching approach, as the latter simply would not work in such a case. Instead, we have chosen a test case where the margins do not have any limits (or, more precisely, have distributions with the probability concentrated far from the limits), but the model turns out to be sensitive to the maximal values.

3.1. Test model formulation

The test case is a stochastic service-network design problem for a less-than-truckload trucking carrier. This is a two-stage stochastic integer program, where the first stage corresponds to routing of the trucks and the second stage to the flow of freight, under stochastic demands. The flow of freight is treated as continuous variables, so we have integrality only in the first stage. In addition, costs are associated only with the first stage, i. e. they depend only on the number of trucks and the lengths of their routes. As a result, the objective function is fully determined by the first-stage solution (truck routing), while the second stage can be seen as a recourse function. Hence, we only need the first-stage solutions in the following tests, so we refer to them simply as 'solutions'.

The model formulation uses a space-time network constructed by repeating the set of nodes (terminals) $\mathcal{N}$ in each of the periods $t \in \mathcal{T} = \{0, \ldots, T-1\}$. Each arc $(i,j) \in \mathcal{N}^2$ represents either a service, if $i \neq j$, or a holding activity if $i = j$; a complete network is assumed. A cost C_{ij} is associated to each arc (i,j), equal to the cost of driving a truck from terminal i to j if $i \neq j$, or to the cost of holding a truck at terminal i if $i = j$.

Each commodity $k \in \mathcal{K}$ is defined by its origin $\mathrm{o}(k)$ and destination $\mathrm{d}(k)$ nodes/terminals, as well as the time periods $\sigma(k)$ and $\tau(k)$ when it becomes available at its origin and when it must be delivered (at the latest) at its destination, respectively. The demand for each commodity D_k^s is the only stochastic parameter of the model, described using the set of equiprobable scenarios $s \in \mathcal{S}$.

The model has two sets of decision variables: the first-stage integer variables x_{ij}^t for the number of tracks driving from node i to node j in period t and second-stage continuous variables $y_{ij,k}^{t,s}$ for the amount of commodity k going from terminal i to terminal j in period t in scenario s. The commodity-flow variables are defined only for t between $\sigma(k)$ and $\tau(k) - 1$. The model is formulated in a time-circular fashion to avoid end-of-horizon issues.

$$\min \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} \sum_{t\in T} C_{ij}\, x_{ij}^t \tag{7a}$$

$$\sum_{i\in\mathcal{N}} x_{ij}^t = \sum_{i\in\mathcal{N}} x_{ji}^{t+1 \bmod T} \qquad \forall t \in \mathcal{T}, \forall j \in \mathcal{N} \tag{7b}$$

$$\sum_{i\in\mathcal{N}} y_{ij,k}^{t,s} = \sum_{i\in\mathcal{N}} y_{ji,k}^{t+1 \bmod T,s} \qquad \forall t \in \mathcal{T}, \forall j \in \mathcal{N}, \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \tag{7c}$$

$$\sum_{k\in\mathcal{K}} y_{ij,k}^{t,s} \leq M x_{ij}^t \qquad \forall (i,j) \in \mathcal{N}^2 \colon i \neq j,\ \forall t \in \mathcal{T}, \forall s \in \mathcal{S} \tag{7d}$$

$$\sum_{j\in\mathcal{N}} y^{\sigma(k),s}_{\mathrm{o}(k)j,k} = D^s_k, \qquad \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \tag{7e}$$

$$\sum_{i\in\mathcal{N}} y^{\tau(k)-1 \bmod T,s}_{i,\mathrm{d}(k);k} = D^s_k, \qquad \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \tag{7f}$$

$$x^t_{ij} \in \mathbb{N}^0 \qquad \forall t \in \mathcal{T}, \forall (i,j) \in \mathcal{N}^2 \tag{7g}$$

$$y^{t,s}_{ij,k} \geq 0 \qquad \forall (i,j) \in \mathcal{N}^2, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}, \forall s \in \mathcal{S}. \tag{7h}$$

The objective function (7a) minimizes the cost of moving the vehicles between the terminals and their holding cost at the terminals (for vehicles parked at the terminals). Constraints (7b) and (7c) represent conservation of flow for respectively trucks and commodities, while (7d)) are the usual linking and vehicle capacity constraints. Note that commodities can be held at nodes without a truck being present there. Constraints (7e) and (7f) represent the conservation of flow at the origin and destination nodes of a commodity, respectively. Finally, (7g) and (7h) define the value ranges for the decision variables.

For the stochastic demands, we use the same distribution of the demands as in [15]. This means that all demands are triangularly distributed and independent on each other. The triangular distribution is convenient for this kind of models, as it is defined by the minimal, maximal, and most likely value.

Note that we use hard constraints in the second stage, i. e. we are constructing routes so that the demands are satisfied in all scenarios. This increases the instability of the scenario-based solutions and therefore magnifies the differences between the different scenario-generation techniques. It is also the reason why the model reacts to the maximal values: a small increase in maximal demand can lead to a need of extra truck, and therefore a jump in the objective function. Note that the hard constraints also imply that a solution obtained using one scenario tree will most likely be infeasible in a model using a different set of scenarios.

3.2. Setup of the test

To test the influence of scenario-generation methods on the quality of the produced solutions, we use the methodology from Kaut and Wallace [13]. This means that we first look at *in-sample stability*: we generate K scenario sets, solve the problem on each of them and then look at the variation of the reported optimal objective values. Ideally, the variation should be as small as possible and converge to zero as the sample size increases to infinity.

Even more important than the stability of the *reported* performance of the solutions is the stability of the *true* performance, referred to as the *out-of-sample stability* (Kaut and Wallace [13]). Unlike the in-sample case, the out-of-sample objective values cannot be computed exactly, as this would mean integrating over a multivariate triangular distribution. Instead, we approximate the out-of-sample values by evaluating the solutions on a large scenario tree, which we refer to as the *target tree*. In our case, we use the same target tree as in Liem and Kaut [15]; this is a tree with 1000 scenarios, generated by sampling from the marginal distributions.

To measure the out-of-sample stability, we would then normally take the K solutions from the in-sample test and measure the variation of their objective values when evaluated on the target tree. However, this generic approach does not work in our case; as we have already pointed out, the hard constraints in the second stage will most likely cause the tested solution to be infeasible on the target tree. Hence, we use a modified version of the test where we compare the amount of infeasibility instead of the objective value, i. e. the amount of demand we are not able to satisfy with the selected truck routing.

We compare performance of the following scenario-generation methods:

- Standard sampling from the target tree.
- Moment matching, with moments and correlations estimated from the target tree.
- The new method, using triangular distributions with parameters estimated from the target tree. This method is initiated by sampling from the standard normal distribution, just as we do in the moment-matching case.
- The new method, using interpolated empirical distribution functions from the target tree. This method is initiated by sampling from the target tree, so it can be seen as a post-process for sampling.

While the first three methods have already been tested in [15] (even though the cdf-based method has not been described there), the last one is new to this paper.

We consider a case with twelve commodities. For each of the scenario-generation methods we measure the stability for scenario trees with tree sizes from 13 to 53, with steps of 8. We solve 20 different problems for each combination of a method and tree size, which equals to $20 \times 4 \times 6 = 480$ problems in total. The solution times per problem range from a couple of minutes for the smallest ones to a couple of days for the worst cases, using CPLEX 9 on a 3 GHz PC.

3.3. Results

The in-sample results are presented in Figure 2. We see that for three of the methods, the optimal objective value increases with the sample size. This is expected, as a model with only few scenarios tends to overestimate its own performance – see Mak et all. [16] for a mathematical formulation and proof of the effect. The absence of this effect in the case of the new method with triangular margins is therefore startling and suggests that the method either gets the right value already with 13 scenarios, or that it fails to improve the estimate even with 53 scenarios. In addition, we see that this method, together with moment-matching, seems to converge to significantly better objective values than the other two methods. Again, this means that the two methods either produce better solutions, or that they are much worse in estimating the quality of their solutions.

The other observation is that sampling produces significantly less stable results. Also this is expected, since the instability is generally the main argument against sampling methods in the case of small trees. We can thus conclude that, in terms of in-sample stability, sampling is clearly the worst method, while the new method with triangular margins is the most stable one.

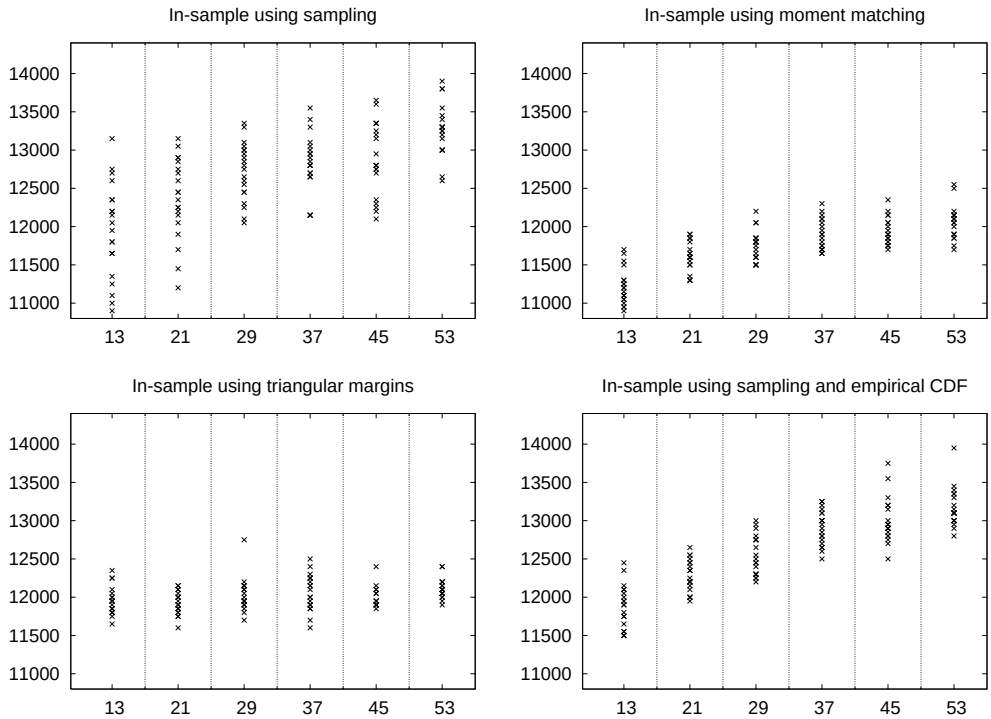

Fig. 2. In-sample optimal objective function values for the four tested methods. The horizontal axes shows the number of scenarios, the vertical axes the optimal objective function value.

The conclusion, however, changes dramatically when we look at the out-of-sample results presented in Figure 3. In this case, the moment-matching and the new method with triangular margins are clearly worse than the other two, both in terms on stability and convergence. This suggests that the methods do not capture the distribution properly and fail to improve as the tree size increases. It also shows that their good performance in the in-sample tests was due to their consistent overestimation of the quality of the produced solutions.

Sampling and the new method using empirical CDFs, on the other hand, show a much better performance, with results that clearly improve with the increasing size of the scenario tree. Out of these two, the new method is significantly better in the case of 13 scenarios and comparable or marginally better in the other cases. This implies that the new method is, overall, the best of the four tested methods.

To conclude, the new method using empirical CDF from the target tree and initiated by a sample from the target tree is clearly the best of the four methods, having the best out-of-sample stability, without exhibiting the strong in-sample instability seen in the sampled trees. In addition, the test illustrates the importance of using all the information available: the two methods that throw away some of the information in the

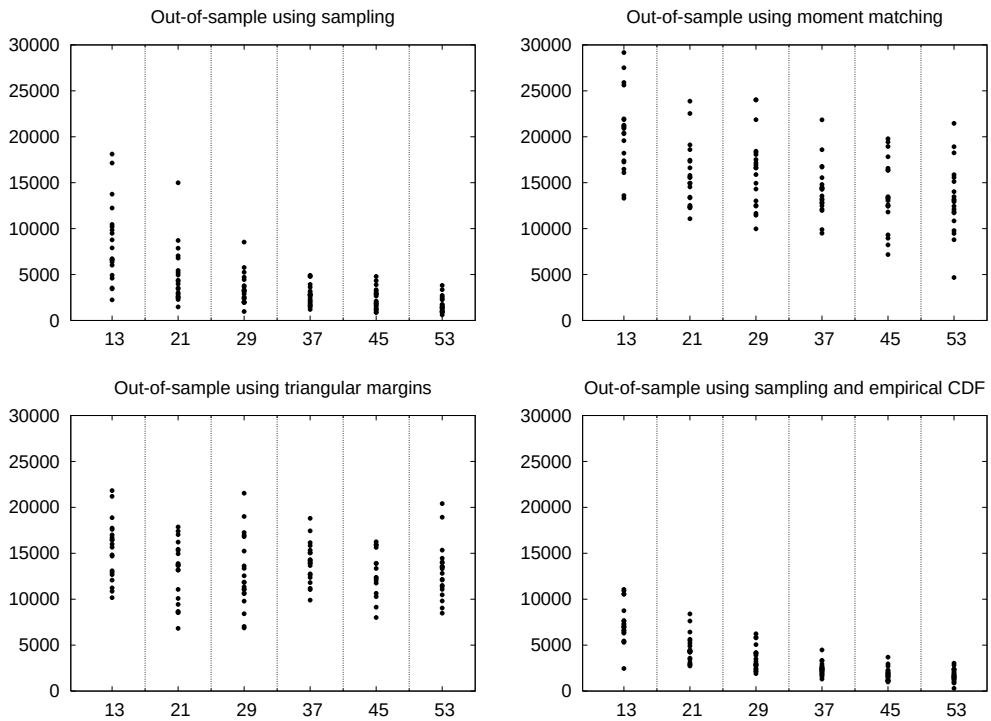

Fig. 3. Out-of-sample results for the four tested methods. The horizontal axes shows the number of scenarios, the vertical axes the slack required to maintain feasibility.

data (the target tree) by approximating the distribution in some way turned out to have a significantly worse out-of-sample performance, compared to the two methods that use the data directly.

CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a method for generating scenarios from distributions specified by their marginal distribution functions and the correlation matrix. This provides an alternative to the existing moment-based method and has a potential to be better in cases where we know the marginal distributions, for example from some theoretical model, or from historical data.

We have illustrated the method on an example from stochastic service network design, where the new method proved to be better than both sampling and the moment-matching approach. While these tests are, by definition, case-dependent, it shows that there are problems where the new method is better than the other two. We can thus conclude that the presented method should be considered when choosing a scenario-generation algorithm for a given problem.

While the presented method gives the user a significantly better control of the marginal distributions, the dependence between the margins is still controlled only by a correlation matrix. We thus believe that the next improvement should come there, possibly by using copula in the way suggested in Kaut and Wall [14] – something we leave for future research.

(Received August 1, 2013)

REFERENCES

[1] J. R. Birge and F. Louveaux: Introduction to stochastic programming. Springer-Verlag, New York 1997.

[2] J. Dupačová, G. Consigli, and S. W. Wallace: Scenarios for multistage stochastic programs. Ann. Oper. Res. *100* (2000), 1 – 4, 25–53. DOI: 10.1023/A:1019206915174.

[3] J. Dupačová, N. Gröwe-Kuska, and W. Römisch: Scenario reduction in stochastic programming: An approach using probability metrics. Math. Programming *95* (2003), 3, 493–511. DOI: 10.1007/s10107-002-0331-0.

[4] S.-E. Fleten and E. Pettersen: Constructing bidding curves for a price-taking retailer in the norwegian electricity market. IEEE Trans. Power Systems *20* (2005), 2, 701–708. DOI: 10.1109/TPWRS.2005.846082.

[5] A. Geyer, M. Hanke, and A. Weissensteiner: No-arbitrage conditions, scenario trees, and multi-asset financial optimization. European J. Oper. Res. *206* (2010), 3, 609–613. DOI: 10.1016/j.ejor.2010.03.022.

[6] H. Heitsch and W. Römisch: Scenario reduction algorithms in stochastic programming. Comput. Optim. Appl. *24* (2003), 2 – 3, 187–206. DOI: 10.1023/A:1021805924152.

[7] H. Heitsch and W. Römisch: Scenario tree modelling for multistage stochastic programs. Math. Programming *118* (2009), 2, 371–406. DOI: 10.1007/s10107-007-0197-2.

[8] H. Heitsch, W. Römisch, and C. Strugarek: Stability of multistage stochastic programs. SIAM J. Optim. *17* (2006), 2, 511–525. DOI: 10.1137/050632865.

[9] R. Henrion, C. Küchler, and W. Römisch: Scenario reduction in stochastic programming with respect to discrepancy distances. Comput. Optim. Appl. *43* (2009), 1, 67–93. DOI: 10.1007/s10589-007-9123-z.

[10] K. Høyland and S. W. Wallace: Generating scenario trees for multistage decision problems. Management Sci. *47* (2001), 2, 295–307. DOI: 10.1023/A:1021853807313.

[11] K. Høyland, M. Kaut, and S. W. Wallace: A heuristic for moment-matching scenario generation. Comput. Optim. Appl. *24* (2003), 2 – 3, 169–185.

[12] P. Kall and S. W. Wallace: Stochastic Programming. John Wiley and Sons, Chichester 1994.

[13] M. Kaut and S. W. Wallace: Evaluation of scenario-generation methods for stochastic programming. Pacific J. Optim. *3* (2007), 2, 257–271.

[14] M. Kaut and S. W. Wallace: Shape-based scenario generation using copulas. Comput. Management Sci. *8* (2011), 1 – 2, 181–199. DOI: 10.1007/s10287-009-0110-y.

[15] A.-G. Lium and M. Kaut: Scenario generation for obtaining sound solutions. In: Stochastic Service Network Design *2* (2006), Chapter 4.

[16] W. Mak, D. Morton, and R. Wood: Monte Carlo bounding techniques for determining solution quality in stochastic programs. Oper. Res. Letters *24* (1999), 47–56.

[17] J. Okunev and D. R. White: Moment matching for the masses. Available at SSRN: `http://ssrn.com/abstract=921451`, 2006.

[18] G. C. Pflug: Scenario tree generation for multiperiod financial optimization by optimal discretization. Math. Programming *89* (2001), 2, 251–271. DOI: 10.1007/PL00011398.

[19] P. Schütz and A. Tomasgard: The impact of flexibility on operational supply chain planning. Int. J. Production Economics *134* (2011), 2, 300–311. DOI: 10.1016/j.ijpe.2009.11.004.

[20] P. Schütz, A. Tomasgard, and S. Ahmed: Supply chain design under uncertainty using sample average approximation and dual decomposition. European J. Oper. Res. *199* (2009), 2 409–419. DOI: 10.1016/j.ejor.2008.11.040.

[21] B. K. Thapalia, T. G. Crainic, M. Kaut, and S. W. Wallace: Single-commodity stochastic network design with multiple sources and sinks. Inform. Systems Oper. Res. *49* (2011), 3, 193–211. DOI: 10.3138/infor.49.3.003.

[22] N. Topaloglou, H. Vladimirou, and S. A. Zenios: A dynamic stochastic programming model for international portfolio management. European J. Oper. Res. *185* (2008), 3, 1501–1524. DOI: 10.1016/j.ejor.2005.07.035.

[23] A. Werner, K. T. Uggen, M. Fodstad, A.-G. Lium, and R. Egging: Stochastic mixed-integer programming for integrated portfolio planning in the LNG supply chain. The Energy J. *35* (2014), 1. DOI:10.5547/01956574.35.1.5.

Michal Kaut, SINTEF Technology and Society, Applied Economics, PO Box 4760 Sluppen, N0-7465, Trondheim. Norway.
e-mail: michal.kaut@sintef.no

Arnt-Gunnar Lium, SINTEF Technology and Society, Applied Economics, PO Box 4760 Sluppen, N0-7465, Trondheim. Norway.
e-mail: arnt-gunnar.lium@sintef.no