

Scenario generation: Property matching with distribution functions

Michal Kaut

`michal.kaut@himolde.no`*

Arnt-Gunnar Lium

`Arnt-Gunnar.Lium@sintef.no`†

May 23, 2007

Abstract

In this paper we present a method for generating scenarios with specified margins and correlations. The margins are described by their distribution functions and we allow each margin to be of different type, including discrete distributions. In addition, it is possible to combine margins specified by their distributions and margins described by moments in one scenario set. We demonstrate the method on a model from stochastic network design and show that it improves the stability of the scenario-generation process, compared to both sampling and a method based on moments and correlations.

Keywords: scenario generation, stability, stochastic programming, logistics

Introduction

In the recent years, scenario generation 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; see for example Dupačová et al. (2003), Heitsch and Römisch (2003, 2005), Høyland and Wallace (2001), Høyland et al. (2003), Pflug (2001). For an overview of all but the most recent methods, see Dupačová et al. (2001).

In this paper, we will focus on the moment-matching algorithm from Høyland et al. (2003) and show how it can be improved and generalized. This algorithm has gained some popularity, probably because of its relative ease of use and implementation, as well as a good performance for some types of problems. Yet, even though the algorithm works in many practical settings, it has some shortcomings. In the first place, some may find describing the margins by moments inconvenient or inefficient: if one happens to know the marginal distribution exactly, using only the first four moments means wasting a lot of information.

In addition, there are settings where using the moments lead to instability of solutions of the optimization model—see Kaut and Wallace (2007) or Heitsch et al. (2006) for a definition and discussion of stability of scenario-generation methods. As an example, consider the situation where the model is sensitive to the minimal and/or maximal value of some of the random variables: since using moments does not provide any control over the support of the distribution, this could lead to substantial instability.

*Molde University College, P.O. Box 2110, N-6402 Molde, Norway (corresponding author)

†SINTEF Technology and Society, S.P. Andersensvei 5, N-7465 Trondheim, Norway

In this paper, we present a method that generalizes the algorithm from Høyland et al. (2003) 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 bigger 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.

To use the method, one needs to be able to evaluate the marginal cumulative distribution functions (CDFs) and their (generalized) inverses. Typically, 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 CDFs in the method. The latter is appropriate in cases where the data represent all the information we have about the marginal distributions, as in these cases any further assumptions on parametric families would mean wasting some of the information.

The rest of the paper is organized as follows: Section 1 summarizes the scenario-generation algorithm by Høyland et al. In Section 2, we present our improvement of the algorithm, which is then in Section 3 compared to the original version as well as sampling, based on a model from stochastic service network design.

1 Short summary of the algorithm by Høyland et al

The algorithm produces one-period scenario trees with specified correlations 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, alternating between the two. The algorithm stops 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 a starting point.
do
  if error_of_correlations > MaxErrorCorr
    Correct correlations.
  for  $i = 1, \dots, n$ 
    if error_of_moments_ $i$  > MaxErrorMom
      Correct moments of  $i$ .
while error_of_correlations > MaxErrorCorr or error_of_moments > MaxErrorMoms.

```

Correlations are corrected using a variant of the standard method for generating correlated normal variates, based on Cholesky decomposition of the correlation matrix. Moments of the marginal distributions are corrected one at a time, using a cubic transformation

$$\mathbf{y} = a + b\mathbf{x} + c\mathbf{x}^2 + d\mathbf{x}^3. \quad (1)$$

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 in four unknowns—see Høyland et al. (2003) for details.

For starting the algorithm, Høyland et al. (2003) uses a two-step approach, starting with margins with moments different from the target moments, to minimize the distortion caused by the correction

of correlations. Unfortunately, this two-stage approach does not work that well in practice, so it is both better and easier to start with some standard distribution instead. A natural choice is then the normal distribution, but other distributions might work better in certain situations—see Kaut (2003, pp. 95–100) for a further discussion.

2 The new algorithm

The important thing to realize is that correcting moments is only one possible way of correcting the marginal distributions—just as correcting correlations is one of possible ways of improving the multivariate structure (co-variation) of the distribution. In this section, we describe a way of controlling marginal distributions using their cumulative distributions functions (CDFs), to get a better control of the margins in situations where CDFs are available. While the method was mainly developed with continuous distributions in mind, we show how it can be adjusted to deal with discrete distributions as well.

Just as in Høyland et al. (2003), using Cholesky decomposition for correcting the correlations means that the margins have to be studentized (set to zero means and variances equal to one) inside the algorithm. The correct means and variances are then obtained at the very end of the algorithm, using a simple linear transformation $\tilde{y} \leftarrow \mu + \sigma\tilde{y}$ for each margin. For most standard distributions, setting mean to zero and variance to one represents a simple change of the distribution’s parameters. The algorithm from Section 1 thus becomes:

```

Studentize the margins' CDFs (make mean=0, variance=1).
Generate a starting point.
do
  if error_of_correlations > MaxErrorCorr
    Correct correlations.
  for  $i = 1, \dots, n$ 
    if error_of_margin. $i$  > MaxErrorMarg
      Correct the  $i$ -th margin.
while error_of_correlations > MaxErrorCorr or error_of_margins > MaxErrorMargins.
Transform margins to the target means and variances.

```

Note that the corrections of margins are done independently for each margin, so we can combine margins corrected by CDFs, moments, and possibly other methods, in one scenario tree. In the rest of the section, however, we will only discuss the margins corrected using CDFs. Note also that we assume that we can calculate the (generalized) inverse of the marginal CDF, denoted F^{-1} .

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 like” 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 on 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 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

$$u_s = F_{\mathbf{x}}^e(x_s) := \frac{2 \operatorname{rank}(x_s, \mathbf{x}) - 1}{2S}, \quad s = 1 \dots S, \quad (2)$$

where \mathbf{x} is the sample of margin's values and $\operatorname{rank}(x_s, \mathbf{x})$ denotes the rank/order of x_s in \mathbf{x} (sometimes denoted $\operatorname{ord}(x_s, \mathbf{x})$), with minimum having the rank of one. The function $F_{\mathbf{x}}^e$ can be viewed as a kind of empirical CDF, except that it is only defined on the values x_s from \mathbf{x} . Note also that $F_{\mathbf{x}}^e$ does not assign zero or one to any value.

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

$$\mathbf{x} = F^{-1}(\mathbf{u}). \quad (3)$$

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

$$\mathbf{x} \leftarrow F^{-1}(F_{\mathbf{x}}^e(\mathbf{x})), \quad (4)$$

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

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

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 \mathbf{x} , as long as the $\operatorname{rank}(\cdot, \mathbf{x})$ function is unique—for example by adding a requirement that if $x_s = x_t$ and $s < t$, then $\operatorname{rank}(x_s, \mathbf{x}) < \operatorname{rank}(x_t, \mathbf{x})$.

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” $F_{\mathbf{x}}^e(x_s)$, we compute the actual CDF $F(x_s)$ and define the error as their mean-square difference:

$$\text{error of margin } \mathbf{x} = \sqrt{\frac{1}{S} \sum_{s=1}^S (F(x_s) - F_{\mathbf{x}}^e(x_s))^2}. \quad (5)$$

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

$$\sup_x |F(x) - F_{\mathbf{x}}^e(x)| = \max_{1 \leq s \leq S} \max \left\{ \frac{\operatorname{rank}(x_s, \mathbf{x})}{S} - F(x_s), F(x_s) - \frac{\operatorname{rank}(x_s, \mathbf{x}) - 1}{S} \right\}. \quad (6)$$

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 true 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 true CDF, within a class of discrete distributions with support of cardinality S .
- (ii) Kolmogorov distance of vector \mathbf{x} from the true CDF after transformation (4) is equal to $\frac{1}{2S}$.

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

$$\max_{1 \leq s \leq S} \max \left\{ \frac{\text{rank}(x_s, \mathbf{x})}{S} - F(x_s), F(x_s) - \frac{\text{rank}(x_s, \mathbf{x}) - 1}{S} \right\} = \max_{1 \leq s \leq S} \max \left\{ \frac{1}{2S}, \frac{1}{2S} \right\} = \frac{1}{2S}.$$

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

$$\max \left\{ \frac{\text{rank}(x_s, \mathbf{x})}{S} - F(x_s), F(x_s) - \frac{\text{rank}(x_s, \mathbf{x}) - 1}{S} \right\} \geq \frac{1}{2S}.$$

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 $\frac{1}{2S}$. \square

Note that, despite the result above, our approach is still very different from a full minimization of the Kolmogorov distance from \mathbf{R} . Henrion and Römisch (2006). 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 data from different sources. In addition, our method is likely to be easier to implement and faster to run, especially for bigger values of S .

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

$$\mathbf{u} = \alpha F_{\mathbf{x}}^e(\mathbf{x}) + (1 - \alpha)F(\mathbf{x}). \tag{2'}$$

The complete transformation (4) then becomes

$$\mathbf{x} \leftarrow F^{-1}(\alpha F_{\mathbf{x}}^e(\mathbf{x}) + (1 - \alpha)F(\mathbf{x})), \tag{4'}$$

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 substantial programming skills.

2.3 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 big 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 $\text{rank}(\cdot, \mathbf{x})$ 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 at the early stages of the algorithm, as we will see later. The function is therefore extended to the whole real axis \mathcal{R} , again see 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 .

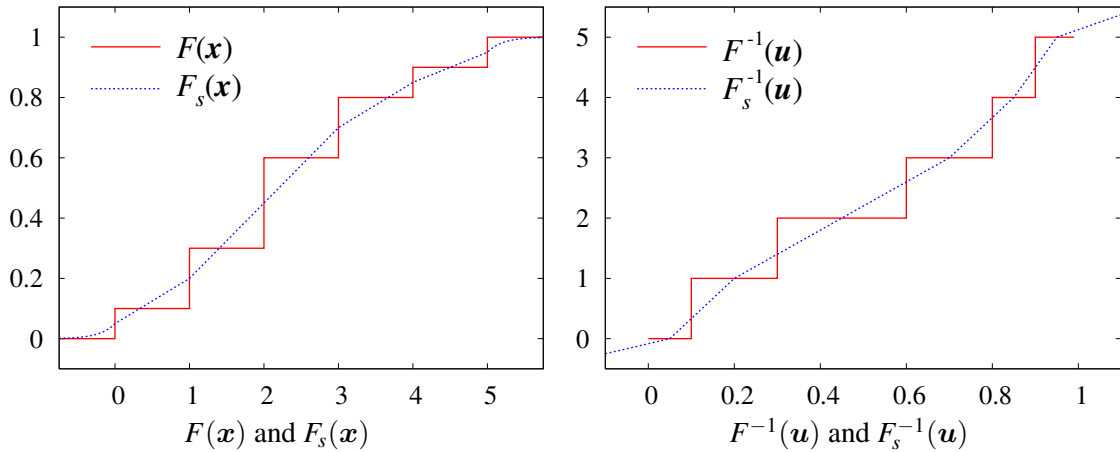


Figure 1: Normal and smoothed version of CDF and inverse CDF for a discrete variable with values $\{0, \dots, 5\}$ and probabilities $\{0.1, 0.2, 0.3, 0.2, 0.1, 0.1\}$.

The smoothed versions of the distribution functions are then used similarly to the ‘empirical CDF’ above: $F(\mathbf{x})$ is replaced by $\beta F(\mathbf{x}) + (1 - \beta)F_s(\mathbf{x})$, where β increases from zero at the beginning of the algorithm to one at the end. The same is applied to the inverse CDF, so (3) becomes

$$\mathbf{x} = \beta F^{-1}(\mathbf{u}) + (1 - \beta)F_s^{-1}(\mathbf{u}). \quad (3'')$$

This can be combined either with (2) or with an adjusted version of the improved formula (2'), in which case we get

$$\mathbf{u} = \alpha F_e(\mathbf{x}) + (1 - \alpha)(\beta F(\mathbf{x}) + (1 - \beta)F_s(\mathbf{x})). \quad (2'')$$

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} .

Since we have not had any real stochastic programming problem with discrete distributions, the method has been tested (successfully) only on a small model. We are therefore unable to provide any guidelines for controlling β —we are not even sure whether general guidelines are possible, or whether the control is case-dependent.

2.4 Generating a starting point

If we have historical data, we can use a sample from the data as our starting point. In addition to being simple to implement, we have a guarantee of starting from the correct distribution—not only correct in terms of margins and correlations, but in terms of the whole multi-variate structure, which we are not able to control inside the algorithm. In this context, we can view our algorithm as a post-process for sampled scenarios.

If we do not have data to sample from, we have to start “from scratch”, by generating $U(0,1)$ pseudo-random values. Then, we have two options: we can start by correcting either the margins, or correlations. If we start with the margins, we first spread the random values evenly on the interval $(0, 1)$, using (2). Then we transform the margins to their target distributions using the inverse CDFs or, for margin described by moments, we first transform them to an intermediate distribution (cf. normal) and then apply the cubic transformation. In this way, we get scenarios that are approximately independent, due to the fact that the margins were obtained one by one. This means that the first correction of correlations will have to be quite strong, resulting in a significant distortion of the marginal distributions.

If we start with the correlations instead, we first transform the margins to the standard normal distribution and then apply the correlation correction. The reason for using the normal distribution is that it is invariant with respect to the transformation, i.e. all the margins remain normal. This also means that we can, at this point, spread the margins using (4), with F^{-1} being the inverse CDF of $N(0, 1)$ distribution. Note that we can do this for both the margins described by CDFs and moments. When we then start the algorithm, the first correction of margins will get us to a better position than in the previous case, as the correlations will be closer to their target value. For this reason, we use this approach in our implementation.¹

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 (2006), which stretches the margins to match a given set of percentiles.

¹The difference between the two methods is negligible for margins described by CDFs, while margins described by moments, can benefit from the extra correction of the correlated normal margins. In practice, however, there is very little difference between the two starting methods.

3 Test case

In this section, we compare the performance of the new scenario-generation method to sampling and moment matching, using a stochastic network design problem for less-than-truckload trucking carrier. This is a two-stage stochastic integer program, where the first stage corresponds to routing of the trucks, while the second stage represents the flow of freight, satisfying 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 distances. 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, though we refer to them simply as “solutions”. For more information about the model, including full formulation, see Lium and Kaut (2006).

We use hard constraints in the second stage, i.e. we are constructing routes so that the demands can be satisfied in all scenarios. This increases the instability of the scenario-based solutions and therefore magnifies the differences between the different scenario-generation techniques. Note that this means that a solution obtained using one scenario tree will most likely be infeasible in a model using a different set of scenarios.

3.1 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 (2007). This means that we first look at *in-sample stability*, i.e. the variation of the reported optimal objective values resulting from using several scenario trees. 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*. Typically, we would use a simulator and evaluate the objective function on the scenario-based solutions, in our case the same optimization model using a tree with 1000 scenarios. In other words, this tree represents the true distribution and is thus referred to as the *truth tree*). However, we have already pointed out that our scenario-based solutions are infeasible on most scenario trees different from the ones they were computed on, so they are most likely infeasible on the truth tree as well. Hence, 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 current truck routing.

We consider a case of twelve commodities, so we have scenarios for a 12-dimensional random variable. The truth tree then consists of one thousand scenarios for the twelve variables, sampled independently from triangular distributions—a common choice of distributions in this type of problems.

We compare performance of the following scenario-generation methods:

- Standard sampling from the truth tree.
- Moment matching, with moments and correlations estimated from the truth tree.
- The new method, using triangular distributions with parameters estimated from the truth 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 truth tree. This method is initiated by sampling from the truth tree, so it can be seen as a post-process for

sampling.

Note that the first three methods have already been tested in Lium and Kaut (2006), while the last one is new to this paper.

For each of the scenario-generation methods we look at the stability for scenario trees with tree sizes from 13 to 53, with step of 8. We solve 20 different problems for each combination of a method and a 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 with 1 GB of memory.

3.2 Results

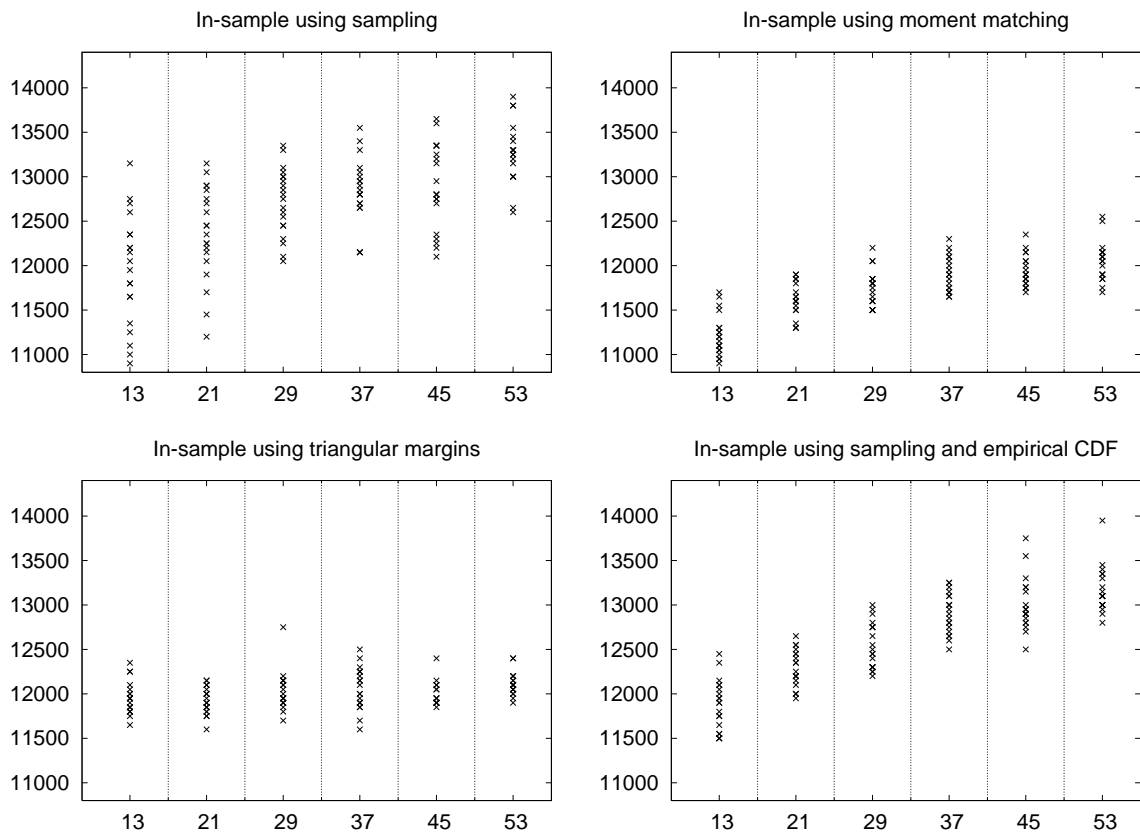


Figure 2: In-sample optimal objective function values using *sampling*, *moments*, *triangular margins* and *margins with empirical CDF*. The horizontal axes shows the number of scenarios, the vertical axes the optimal objective function value.

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 al. (1999) for a proper 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.

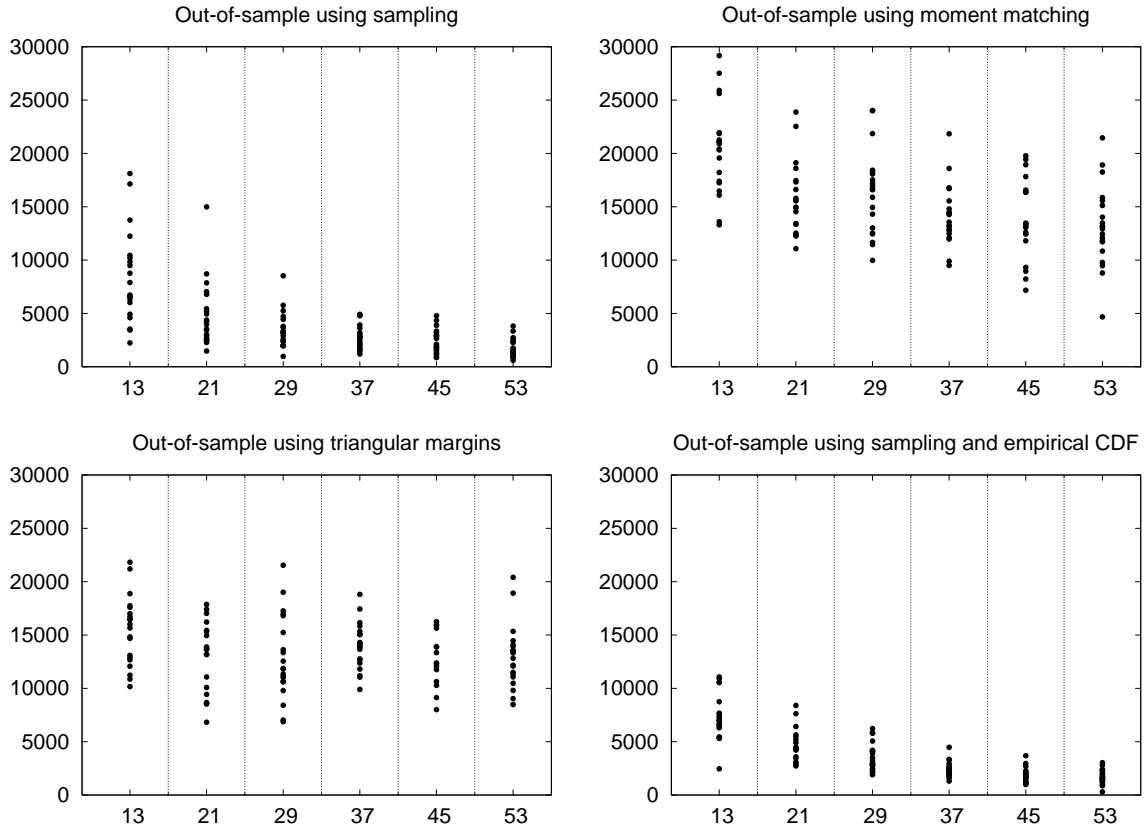


Figure 3: Out-of-sample results: amount of infeasibility of solutions based on *sampling*, *moments*, *triangular margins* and *margins with empirical CDF*. The horizontal axes shows the number of scenarios, the vertical axes the slack required to maintain feasibility.

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, which makes it a winner of the out-of-sample test.

To conclude, the new method using empirical CDF from the truth tree and initiated by a sample from the truth 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 data (the truth 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 with specified correlations and marginal distributions given by their distribution functions. This presents an improvement over matching only the moments of the distributions, both in the terms of approximating the distribution, and obtaining a better solution of stochastic programs. This has been illustrated on a case from service network design, where the new method proved to be better than both sampling and the moment-matching approach.

While the presented method gives the user a significantly better control of the marginal distributions, the co-distribution (multi-variate structure of the distribution) is still controlled only by a correlation matrix. We thus believe that the next improvement should come there—something we leave for future research.

References

- Jitka Dupačová, Giorgio Consigli, and Stein W. Wallace. Scenarios for multistage stochastic programs. *Ann. Oper. Res.*, 100:25–53, 2001. ISSN 0254-5330.
- Jitka Dupačová, Nicole Gröwe-Kuska, and Werner Römisch. Scenario reduction in stochastic programming. *Mathematical Programming*, 95(3):493–511, 2003.
- H. Heitsch and W. Römisch. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2–3):187–206, 2003.
- H. Heitsch and W. Römisch. Scenario tree modelling for multistage stochastic programs. Technical Report Preprint 296, DFG Research Center MATHEON, “Mathematics for key technologies”, Technische Universität Berlin, Germany, 2005.
- H. Heitsch, W. Römisch, and C. Strugarek. Stability of multistage stochastic programs. *SIAM Journal on Optimization*, 17(2):511–525, 2006.
- K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001.
- Kjetil Høyland, Michal Kaut, and Stein W. Wallace. A heuristic for moment-matching scenario generation. *Comput. Optim. Appl.*, 24(2-3):169–185, 2003. ISSN 0926-6003.
- Michal Kaut. *Scenario tree generation for stochastic programming: Cases from finance*. PhD thesis, Norwegian University of Science and Technology, Trondheim, 2003. Dr.ing. thesis 2003:55.
- Michal Kaut and Stein W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, to appear, 2007.

- Arnt-Gunnar Lium and Michal Kaut. Scenario generation for obtaining sound solutions. In *Stochastic Service Network Design*, chapter 4. Molde University College, 2006. PhD thesis by A.-G. Lium.
- W.K. Mak, D.P. Morton, and R.K. Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Oper. Res. Lett.*, 24:47–56, 1999.
- John Okunev and Derek R. White. Moment matching for the masses. Available from <http://www.ssrn.com/>, 2006.
- G. C. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89(2):251–271, 2001.
- C. Küchler R. Henrion and W. Römisch. Scenario reduction in stochastic programming with respect to discrepancy distances. Preprint, DFG Research Center MATHEON “Mathematics for key technologies”, 2006.