# A copula-based heuristic for scenario generation

Michal Kaut
NTNU Trondheim, Norway
and
SINTEF Technology and Society, Trondheim, Norway

in: Computational Management Science. See also BibTeX entry below.

BibTeX:

```
@article{Kaut11,
  author = {Michal Kaut},
  title = {A copula-based heuristic for scenario generation},
  journal = {Computational Management Science},
  volume = {11},
  number = {4},
  pages = {503--516},
  year = {2014},
  doi = {10.1007/s10287-013-0184-4}
}
```



Document created on September 29, 2014.
Cover page generated with CoverPage.sty.

# A copula-based heuristic for scenario generation

Michal Kaut*

September 2011, updated July 2013 and September 2014

This paper presents a new heuristic for generating scenarios for two-stage stochastic programs. The method uses copulas to describe the dependence between the marginal distributions, instead of the more common correlations. The heuristic is then tested on a simple portfolio-selection model, and compared to two other scenario-generation methods.

Keywords: stochastic programming; scenario generation; copulas

## Introduction

In most practical applications of stochastic programming, we have to approximate the probability distribution of the stochastic parameters by a discrete distribution, i.e. a list of realizations (scenarios) and their probabilities. The process of generating this discrete distribution is usually referred to as '*scenario generation*'. As with any other approximation, the quality of the scenarios is an important determinant of the quality of the solutions obtained from the model—bad scenarios can ruin an otherwise flawless model. This issue is further complicated by the fact that we typically want to keep the number of scenarios as low as possible, to be able to solve the problem in a reasonable time.

There are many methods for generating scenarios, see Dupačová et al. (2000) for an overview. Some of these methods try to generate scenarios that match a given set of specifications for their marginal distributions and the dependence between them, where the latter is almost always specified using the correlation (or variance-covariance) matrix. This is sufficient for elliptical distributions such as normal or the Student's $t$-distribution, but might fail in the general case: it cannot, for example, model asymmetric dependence (a situation where the dependence in down-turns differs from the up-turns) or tail dependence—phenomena known

*Norwegian University of Science and Technology (NTNU), Trondheim, Norway;
SINTEF Technology and Society, Trondheim, Norway; michal.kaut@sintef.no.

to exist in financial data (Hu, 2006; Longin and Solnik, 2001; Patton, 2004), or industries with changing trends, such as the apparel industry (Vaagen and Wallace, 2008).

The effect of using correlations in such situations is investigated in Kaut and Wallace (2011), which confirms that correlations can indeed lead to sub-optimal solutions to the stochastic model. The paper also shows that the problem can be addressed by modelling the dependence using *copulas* instead of correlations, and describes how such scenario-generation methods could work. It does not, however, present any new method for generating scenarios for the copula itself; the tests are done using sampling. In this paper, we try to fill this gap by presenting a heuristic for generating scenarios from a given copula. This heuristic then forms a base for a new copula-based scenario-generation method. It should be noted that the presented method is meant for two-stage stochastic programs.

The rest of the paper consists of the following parts: first, we describe copulas and their potential for scenario generation, plus a general structure of the proposed method. To simplify the presentation of the heuristic, we then discuss the bivariate case in Section 2, before presenting the general method in Section 3. Section 4 presents several ways of converting the resulting values from copulas to the specified target distributions. Finally, we test the scenarios produced by the presented method on a simple portfolio-optimization model in Section 5, before concluding the paper.

# 1 Copulas and scenario generation

Copula is the joint cumulative distribution function (cdf) of any $n$-dimensional random vector with standard uniform margins, i.e. a function $C : [0,1]^n \to [0,1]$. The *Sklar's theorem* (Sklar, 1996) states that for any $n$-dimensional cdf $F$ with marginal distribution functions $F_1, \ldots, F_n$, there exists a copula $C$ such that

$$F(x_1, \ldots, x_n) = C\big(F_1(x_1), \ldots, F_n(x_n)\big)\,.$$

Moreover, if all the marginal cdfs $F_i$ are continuous, then $C$ is unique. An immediate consequence of the theorem is that, for every $\mathbf{u} = (u_1, \ldots, u_n) \in [0,1]^n$,

$$C(u_1, \ldots, u_n) = F\big(F_1^{-1}(u_1), \ldots, F_n^{-1}(u_n)\big)\,,$$

where $F_i^{-1}$ is the generalized inverse of $F_i$.

It follows that a multivariate cdf is fully determined by the marginal cdfs and the copula; in other words, the copula is a full description of the dependence between the margins. This is in contrast to correlations that measure only the level of *linear* dependence. Furthermore, it shows that copulas, again unlike correlations, are independent on the marginal distributions, so we can model the two independently.

This suggests the following two-step *procedure for generating scenarios* (Kaut and Wallace, 2007): first, generate scenarios for the desired copula. Since a copula is a multivariate distribution with standard uniform (U$(0,1)$) margins, the margins of the scenario-distribution will constitute a sample from the U$(0,1)$ distribution. In the second step, we thus only need

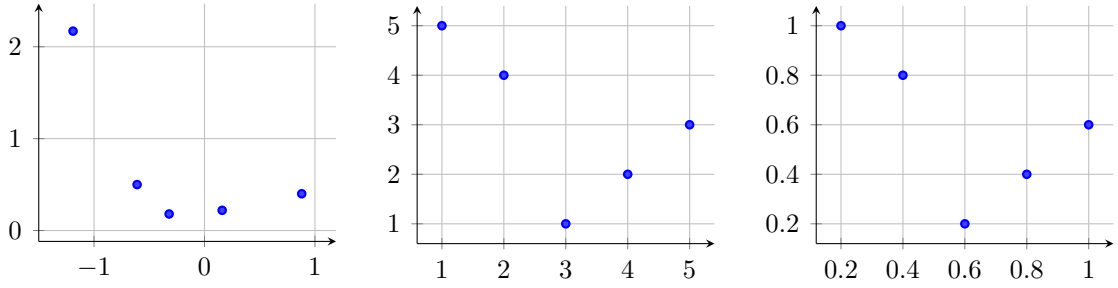


**Figure 1:** Relation between a sample (left), its empirical copula in terms of ranks (center) and the empirical copula with margins scaled to $(0, 1]$ (right).

to transform the margins using the inverse cdfs, to get scenarios that have both the correct copula and marginal distributions—and therefore the correct multivariate distribution.

While the second step is trivial, generating scenarios from a given copula is a more difficult task. The only available method we are aware of is sampling; there are readily available codes for sampling from all the major copula families.[1] The problem with sampling, however, is that one needs a lot of scenarios to get a reasonable approximation of the distribution and hence reliable results from stochastic-optimization models using these samples. Moreover, there are many applications where we simply cannot solve problems with the required number of scenarios (Kaut and Wallace, 2007). We thus need a 'smarter' way of creating samples from a given copula, which can achieve a comparable quality of solutions with fewer scenarios.

## 1.1 Measuring the distance of a copula sample from its cdf

For the purpose of this paper, a copula sample can be viewed as a sample with all the information about the original marginal distributions removed. It follows that the values of the sample do not matter; as long as we do not change the order of the values, the copula remains the same. One natural option is to change the values to the ranks of the values in the sample, with 1 denoting the minimum and $S$ the maximum. The copula sample is thus equivalent to an assignment between the ranks of the margins, as illustrated in Fig. 1.

From now on, a *copula sample* means a set

$$\mathcal{C} = \{\mathbf{r} = (r_1, \ldots, r_n) : 1 \leq r_i \leq S,\ \forall i \leq n\}\,, \tag{1}$$

where each value appears exactly once in each dimension—it is an assignment. To get the cumulative distribution function (cdf) of one sample point $\mathbf{r} \in \mathcal{C}$, we have to scale the values back to the interval $[0, 1]$:

$$\begin{aligned} C_r(\mathbf{r}) &= C(\tfrac{r_1}{S}, \ldots, \tfrac{r_n}{S}) = \Pr\big([0, \tfrac{r_1}{S}] \times \cdots \times [0, \tfrac{r_n}{S}]\big) \\ &= \tfrac{1}{S}\big|\{\mathbf{r}' = (r'_1, \ldots, r'_n) : r'_i \leq r_i,\ \forall i \leq n\}\big|\,, \end{aligned} \tag{2}$$

where $C_r$ denotes the sample cdf with input in terms of ranks and $C$ the sample cdf with the sample values in $[0, 1]$.

[1] For some available codes, see, for example, http://www.mathfinance.cn/tags/copula.

To assess the 'quality' of the sample, we compare these values to cdf of the target copula $C^*$, denoted as $C^*_r(\mathbf{r}) = C^*(\mathbf{r}/S)$. In other words, we compute deviations

$$\mathrm{dev}(\mathbf{r}) = \mathrm{dev}(r_1, \ldots, r_n) = C_r(r_1, \ldots, r_n) - C^*_r(r_1, \ldots, r_n) \tag{3}$$

The overall quality is then computed as a function of the measured differences. We consider the following two measures:

$$\mathrm{d}_{\mathrm{avg}}(\mathcal{C}, C^*) = \frac{1}{S^n} \sum_{r_1=1}^{S} \cdots \sum_{r_n=1}^{S} \big|\mathrm{dev}(r_1, \ldots, r_n)\big| \tag{4a}$$

$$\mathrm{d}_{\max}(\mathcal{C}, C^*) = \max_{\substack{1 \le r_i \le S \\ i=1,\ldots,n}} \big|\mathrm{dev}(r_1, \ldots, r_n)\big| \tag{4b}$$

This approach has one obvious problem: to evaluate the quality of the sample, we need to evaluate the copula at $S^n$ grid points, which is not feasible for most practical problems. Instead, we propose working only with bivariate copulas, i.e. specify the dependence structure pairwise. This will decrease the numerical complexity from $\mathcal{O}(S^n)$ to $\mathcal{O}(n^2\, S^2)$, a much more manageable number.

The price we pay for this simplification is that we no longer have a complete description of the dependency—this approach will not model any higher-order dependencies. On the other hand, it should be more powerful than using correlations, as we move from one number per pair of random variables to one cdf per pair.

We will thus first focus on the bivariate case and present two different methods for generating scenarios that minimize the bivariate versions of distances (4). We then extend these methods to the multivariate case in Section 3.

In the rest of the paper, we assume that we know the target copulas. In reality, obtaining the copulas is a nontrivial task, analogous to estimating distribution functions for sampling. If we have enough data, the easiest option is to use the *empirical copula*, as illustrated in Fig. 1, directly as the target. Otherwise, we have to use some parametric method, that is to first find a matching copula model and then estimate its parameters—just as what we would do when estimating distributions. This, however, is a complicated topic, out of the scope of this paper. Interested readers can find more information in, for example, Nelsen (1998); Bouyé et al. (2000); Romano (2002).

## 2 Generating scenarios for bivariate copulas

In the previous section, we have shown that the problem of finding the best copula sample can be expressed as an assignment problem in terms of ranks of the marginal distributions. In the bivariate case, this can be easily formulated as a mixed-integer programming (MIP) model, which we present next. Note that we change the notation slightly for the rest of this section and use the usual $(i, j)$ indices instead of $(r_1, r_2)$ from the multivariate case.

## 2.1 MIP models for the bivariate problem

The assignment is modelled by binary variables $x_{ij}$ that are equal to one if the $j$-th rank of the second margin is assigned to the $i$-th rank of the first margin, i.e. if there is a scenario with $r = (i, j)$. The $C_r$ function from (2) is then equal to

$$C_r(i,j) = \frac{1}{S} \sum_{k=1}^{i} \sum_{l=1}^{j} x_{kl}$$

and deviation $\mathrm{dev}(\mathbf{r})$ from Eq. (3) becomes

$$\mathrm{dev}(i,j) = C_r(i,j) - C_r^*(i,j)\,. \tag{5}$$

Next, we decompose the deviation into its positive and negative parts,

$$\mathrm{dev}(i,j) = \frac{1}{S}(y_{ij}^+ - y_{ij}^-)\,, \tag{6}$$

which gives

$$S\,\mathrm{dev}(i,j) = \sum_{k=1}^{i} \sum_{l=1}^{j} x_{kl} - S\,C_r^*(i,j) = y_{ij}^+ - y_{ij}^- \tag{7}$$

and, consequently,

$$S\,\big|\mathrm{dev}(i,j)\big| = y_{ij}^+ + y_{ij}^-\,. \tag{8}$$

The problem of minimizing the average absolute deviation $\mathrm{d}_{\mathrm{avg}}$ from Eq. (4a), omitting the scaling factor $1/S$, can then be written as

$$\min \quad \sum_{i,j=1}^{S} \left(y_{ij}^+ + y_{ij}^-\right) \tag{9a}$$

$$\text{s. t.} \quad \sum_{i=1}^{S} x_{ij} = 1 \qquad \forall\, j \in \{1 \ldots S\} \tag{9b}$$

$$\sum_{j=1}^{S} x_{ij} = 1 \qquad \forall\, i \in \{1 \ldots S\} \tag{9c}$$

$$\sum_{k=1}^{i} \sum_{l=1}^{j} x_{kl} - y_{ij}^+ + y_{ij}^- = S\,C_r^*(i,j) \qquad \forall\, i,j \in \{1 \ldots S\} \tag{9d}$$

$$y_{ij}^+ \geq 0,\ y_{ij}^- \geq 0,\ x_{ij} \in \{0,1\} \qquad \forall\, i,j \in \{1 \ldots S\} \tag{9e}$$

The objective function (9a) and constraints (9d) come from Eqs. (7) and (8), while constraints (9b) and (9c) define the assignment by ensuring that each rank is used only once. Note, however, that problem (9) differs from the standard assignment problem, so we cannot expect polynomial solution time.

The problem of minimizing the maximum deviation $\mathrm{d}_{\mathrm{max}}$ is very similar to minimizing $\mathrm{d}_{\mathrm{avg}}$, we just add a new variable $y$ for the maximum deviation, defined by constraints

$$y_{ij}^+ + y_{ij}^- \leq y \qquad \forall\, i,j \in \{1 \ldots S\}\,, \tag{10}$$

and change the objective from (9a) into minimizing the new variable $y$.

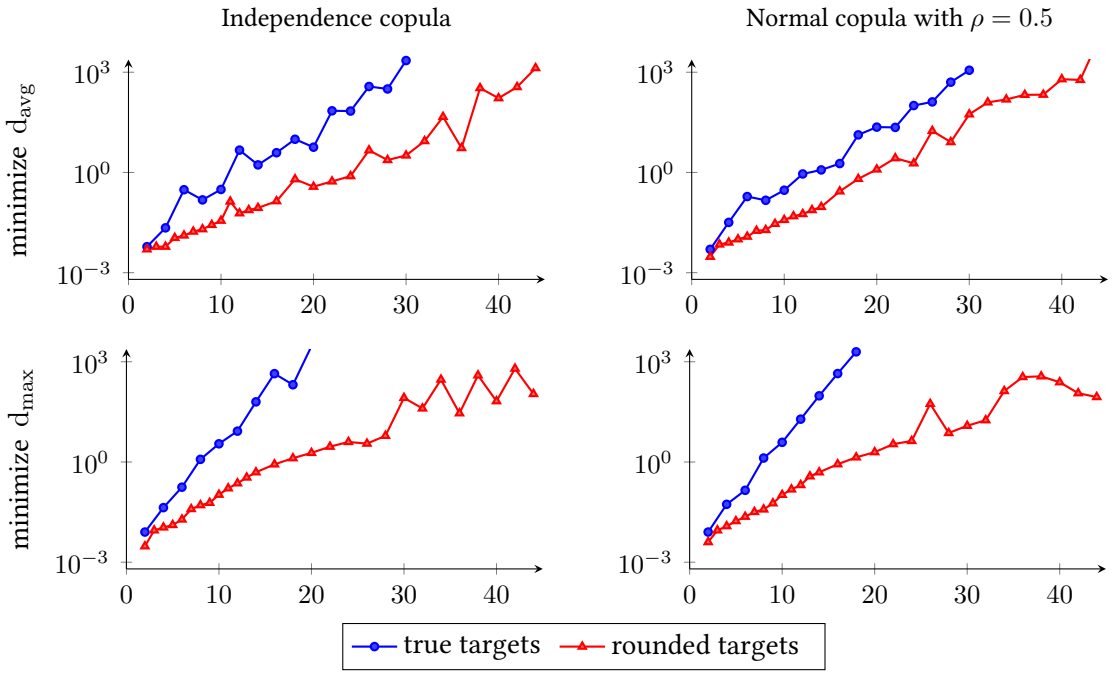


**Figure 2:** Solution times for problems (9) and (10), for two different copulas. Series '—▲—' shows the result of the default model with target cdfs rounded to the nearest integer. Note the logarithmic scale one the value axis—a straight line means exponential growth.

### Solution times

Both models were implemented in C++ using the FlopC++ library (Hultberg, 2007) and solved using Cbc 2.5 (Forrest and Lougee-Heimer, 2005) on a 12-core 2.4 GHz AMD Opteron with 24 GB RAM, solving several instances simultaneously. We have tested also other solvers and the results were qualitatively very similar.

An important observation is that for both models, all the coefficients in the LP matrix are integers, except for the target cdf values $C_r^*(i, j)$. It can thus be expected that the models will solve faster if we round the target values to the nearest integer. This is confirmed in our test results, presented in Fig. 2: the rounding decreases the solution times dramatically, especially in the case of minimizing the maximum deviation $\mathrm{d}_{\mathrm{max}}$. Yet even with the rounded targets, the method is only practical for up to 35–50 scenarios, depending on model and copula type.

In addition, we have to remember that in the general $n$-dimensional case, these models would have to be expanded to take into accounts the other margins, and we would have to solve $\frac{n(n-1)}{2}$ of them. It follows that the MIP-based approach does not work for problems of even moderate size. As a result, we have developed a heuristic that solves the problem (9) approximately.

```
1: 𝓘 ← {1, ..., S};  δr* ← ∞                                  initializations
2: for j ∈ {1, ..., S} do
3:     for i ∈ 𝓘 do                                           loop through unused ranks
4:         calculate the deviation δr(i, j)
5:         if δr(i, j) < δr* then                             new best assignment
6:             i* ← i;  δr* ← δr(i, j)
7:         end if
8:     end for
9:     r_j ← i*                                               assign rank j to rank i*
10:    𝓘 ← 𝓘 \ {i*}                                           mark i* as used
11: end for
```

**Figure 3:** Heuristic for constructing a bivariate copula sample with minimal $\mathrm{d}_{\mathrm{avg}}$.

## 2.2 Heuristic for the average-deviation problem

In this section, we present a heuristic for the bivariate case of the average-deviation problem. Just like the MIP formulation (9) from the previous section, the heuristic works on the ranks of the copula and uses the difference between the cdfs as a measure of distance between the sample and the target. It is based on the observation from Eq. (2) that the rank cdf $C_r(i,j)$ depends only on points $\{(i',j') : i' \leq i \;\&\; j' \leq j\}$ of the grid. This means that we can construct the sample 'row-wise': for each rank $j$, we compute the deviation caused by its pairing to rank $i$, denoted $\delta_{\mathrm{r}}(i,j)$,[2] as the sum of absolute values of deviations at grid points $(\cdot, j)$:

$$\delta_{\mathrm{r}}(i,j) = \sum_{l=1}^{S} \big|\mathrm{dev}(l,j)\big| = \sum_{l=1}^{S} \big|C_r(l,j) - C_r^*(l,j)\big| \,. \tag{11}$$

We do this for all unused ranks $i$ and then choose the one with the smallest deviation. The resulting greedy heuristic is presented in Fig. 3.

The most time-critical part of the heuristic is the calculation of $\delta_{\mathrm{r}}(i,j)$ on line 4: if we simply use the definition from Eq. (11), the calculation would require $\mathcal{O}(S)$ operations, making the whole heuristic $\mathcal{O}(S^3)$. Fortunately, this can be improved upon using the fact that, for $i > 1$, we have

$$C_r(i,j) = C_r(i,j-1) + \begin{cases} \frac{1}{S} & \text{if we couple } j \text{ to some } i' \leq i \\ 0 & \text{otherwise}\,, \end{cases}$$

[2] In the published version of the paper, we did not introduce this extra notation for the pairing-based deviation and continued using the existing symbol $\mathrm{dev}(i,j)$ instead.

which, in combination with (11), gives

$$\begin{aligned}\delta_{\mathrm{r}}(i,j) &= \sum_{l=1}^{S}\left|C_r(l,j-1) + \mathrm{if}(l \geq i, \tfrac{1}{S}, 0) - C_r^*(l,j)\right| \\ &= \sum_{l=1}^{i-1}\left|C_r(l,j-1) - C_r^*(l,j)\right| + \sum_{l=i}^{S}\left|C_r(l,j-1) + \tfrac{1}{S} - C_r^*(l,j)\right|\end{aligned}$$

and therefore

$$\delta_{\mathrm{r}}(i-1,j) = \sum_{l=1}^{i-2}\left|C_r(l,j-1) - C_r^*(l,j)\right| + \sum_{l=i-1}^{S}\left|C_r(l,j-1) + \tfrac{1}{S} - C_r^*(l,j)\right|.$$

Subtracting the two then gives

$$\begin{aligned}\delta_{\mathrm{r}}(i,j) = \delta_{\mathrm{r}}(i-1,j) &+ \left|C_r(i-1,j-1) - C_r^*(i-1,j)\right| \\ &- \left|C_r(i-1,j-1) + \tfrac{1}{S} - C_r^*(i-1,j)\right|,\end{aligned} \tag{12}$$

which is then initialized with

$$\delta_{\mathrm{r}}(0,j) = \sum_{l=1}^{S}\left|C_r(l,j-1) + \tfrac{1}{S} - C_r^*(l,j)\right|.$$

This recursive formula allows us to compute $\delta_{\mathrm{r}}(i,j)$, except $\delta_{\mathrm{r}}(0,j)$, in a constant number of operations, so the heuristic becomes $\mathcal{O}(S^2)$—a big difference to the exponentially growing solution time of the MIP formulations. In practice, this means that we can generate five thousand samples in less than a second on the same hardware we used for the MIP models. Note that the recursion implies that we have to evaluate $\delta_{\mathrm{r}}(i,j)$ for all $i \in \{1,\ldots,S\}$, instead of using only $i \in \mathcal{I}$ as we do in the algorithm presented in Fig. 3.[3]

# 3 Heuristic for the multivariate case

In this section, we extend the heuristic from a bivariate to a general multivariate case. This is done by starting with two margins and then adding one margin at a time. The procedure is best described in an inductive manner: assume that we have already generated values for $m$ margins and want to add a new margin $m+1$, using the bivariate copulas of variable pairs $(1,m+1),\ldots,(m,m+1)$ as targets. Unlike the bivariate case, we cannot simply connect rows and columns, since we have to take into account that the $m$ margins have already been connected together. Instead, we assign the ranks of the new margin to *scenarios*; in other words we assume that for the $m$ processed margins, each rank has been connected with one

[3] In the published version of the paper, we use a variant of (12) where the recursion is in the second argument:

$$\begin{aligned}\delta_{\mathrm{r}}(i,j) = \delta_{\mathrm{r}}(i,j-1) &+ \left|C_r(i-1,j-1) - C_r^*(i,j-1)\right| \\ &- \left|C_r(i-1,j-1) + \tfrac{1}{S} - C_r^*(i,j-1)\right|.\end{aligned}$$

This formula has slightly different properties than (12): on the plus side, we would only need to evaluate $\delta_{\mathrm{r}}(i,j)$ for the active rows $i \in \mathcal{I}$, saving some running time. On the other hand, the formula uses $\delta_{\mathrm{r}}(i,j-1)$ from the previous column, which requires extra bookkeeping. For this reason, we use the form (12) in our implementation, while the other form is left for future versions of the code.

```
1: 𝒮 ← {1, ..., S}; δr* ← ∞                                  initializations
2: for j ∈ {1, ..., S} do
3:     for s ∈ 𝒮 do                                          loop through unused scenarios
4:         for k ∈ {1, ..., m} do
5:             calculate the deviation δr^k(r_s^k, j)
6:         end for
7:         δrs ← Σ_{i=1}^{m} δr^k(r_s^k, j)                  dev. of putting j into scen. s
8:         if δrs < δr* then                                 new best assignment
9:             s* ← s; δr* ← δrs
10:        end if
11:    end for
12:    r_j^{m+1} ← s*                                        assign rank j to scenario s*
13:    𝒮 ← 𝒮 \ {s*}                                          mark s* as used
14: end for
```

**Figure 4:** Step $m$ of the multivariate heuristic, adding a new margin $m+1$ to the set of already constructed margins $1, \dots, m$.

scenario. The goal of this step of the algorithm is thus to connect the ranks of margin $m+1$ to scenarios.

Since we now work with $m$ different bivariate copulas, instead of one, we need to extend the copula notation by an additional superscript for margin. Hence, $r_s^k$ becomes the rank of variable $k$ assigned to scenario $s$, ${C_r}^k$ and ${C_r}^{*k}$ denote respectively the sample and target cdfs of the bivariate copula of variables $k$ and $(m+1)$, and ${\delta_r}^k$ the deviation function of this copula, ${\delta_r}^k(i,j) = {C_r}^k(i,j) - {C_r}^{*k}(i,j)$.

The assignment is then done in analogously to the bivariate case: each rank $j$ of the $(m+1)$-st margin is assigned to the scenario $s \in \mathcal{S}$ causing the smallest deviation. The main difference is that the deviation is defined as the sum of all bivariate deviations ${\delta_r}^k(r_s^k, j)$ coming from copulas ${C_r}^k$, for $k \in \{1, \dots, m\}$. The resulting algorithm is presented in Fig. 4. We can see that the bivariate heuristic from Fig. 3 is a special case of the new code, with $m = 1$ and $r_s^1 = s$ for all $s$. This means that we can use the new code for the whole scenario-generation process: we first initialize the first margin, for example to $r_s^1 \leftarrow s$, and then execute the code from Fig. 4 for $m \in \{1, \dots, S-1\}$.

Since the ${\delta_r}^k()$ functions at line 5 of the algorithm are the same as in the bivariate case, we can again use the recursive formula from Eq. (12)—once more assuming that we adjust the heuristic so that the deviations are calculated for all $s \in \{1, \dots, S\}$, instead of $s \in \mathcal{S}$. This way, the heuristic becomes $\mathcal{O}(n\,S^2)$ for each margin and therefore $\mathcal{O}(n^2\,S^2)$ in total.[4]

Note that the heuristic in the form presented in Fig. 4 does not take into account the possibility of several scenarios having the same deviation; in such a case, it would always pick the first one. In our actual implementation, we instead store all the best scenarios found

[4]This assumes that we match all the possible pairs; more generally, we can replace $n^2$ by the number of matched bivariate copulas. For example, if we specify copulas only for pairs of margins $(i,j)$ with $|i-j| \le k$, for some fixed $k$, the heuristic becomes $\mathcal{O}(n\,S^2)$.

at line 8 and then choose one of them randomly at line 12.

# 4 Transformation to the target marginal variables

So far, we have been concerned only about generating scenario for the *copula* and ignored the fact that these have to be transformed to the target variables. Fortunately, this is usually quite easy, depending on the information we have about the marginal distributions. Assuming we have generated scenarios in terms of rank couplings $\mathbf{r}^s = (r_s^1, \ldots, r_s^n)$ for $s \in \mathcal{S}$, with $1 \leq r_s^i \leq S$, we have at least the following options:

1. We know the cdfs $F_i$ and they are invertible. In this case, we can choose between:
   a) $x_i^s = F_i^{-1}\left(\frac{r_s^i}{S}\right)$ would be the natural choice, but $r_s^i = S$ gives $x_i^s = F_i^{-1}(1)$ that is infinite for distributions with unlimited support.
   b) $x_i^s = F_i^{-1}\left(\frac{r_s^i}{S+1}\right)$ is one way of avoiding the above problem.
   c) $x_i^s = F_i^{-1}\left(\frac{r_s^i - 0.5}{S}\right)$ is another way; here, the resulting $x_i^s$ values are conditional medians of intervals $\left[F_i^{-1}\left(\frac{r_s^i - 1}{S}\right), F_i^{-1}\left(\frac{r_s^i}{S}\right)\right]$. This is our choice.
   d) $x_i^s = \mathrm{E}\left\{X_i \mid F_i^{-1}\left(\frac{r_s^i - 1}{S}\right) \leq X_i \leq F_i^{-1}\left(\frac{r_s^i}{S}\right)\right\}$, i.e. the conditional means of the same intervals; this implies that the samples will have the correct means.

   Note that out of the last three methods, method 1(c) produces the most extreme values for most distributions, since $F_i^{-1}(\frac{1}{2S}) < F_i^{-1}(\frac{1}{S+1})$ and mean is above the median because of the skewness of the tail distribution. However, the difference is noticeable only for small values of $S$.

2. We have historical data and therefore the empirical cdfs $F_{\mathrm{e}i}$. Unfortunately, these are not invertible, so we have to do something extra:
   a) using the standard pseudo-inverse $F_{\mathrm{e}i}^{(-1)}$; this could produce repeating values because $F_{\mathrm{e}i}$ are piece-wise constant functions.
   b) interpolated/smoothed version of $F_{\mathrm{e}i}^{(-1)}$: with $D$ data points, we would only use the values at $\frac{k}{D}$, shifted to the left by $\frac{1}{2D}$ to make the method consistent with the continuous case. For all other values $r$, we would use some interpolation—in our case cubic splines, but a simple linear interpolation should work just as well.

3. We have some other method for controlling the marginal distributions, such as the moment-matching approach from Høyland et al. (2003). In this case, we can simply generate the values of the margins, compute their ranks and then assign the values to scenarios with corresponding ranks $r_s^i$. This process is illustrated in Table 1, where the output values $r_s^i$ and $X_{s,i}$ correspond to the second and first plot in Fig. 1, respectively.

Of all the cdf-based methods above, only 1(d) guarantees that the scenarios will have the correct expected values; the rest will produce scenarios with means slightly different from the correct values, with the error inversely proportional to the number of scenarios. Since

**Table 1:** Scenario generation using pre-generated margins. The left table presents the pre-generated margin samples $M_{r,i}$, sorted for each margin $i$. The right table presents the generated copula in terms of ranks $r_s^i$ and the resulting scenario values $X_{s,i}$.

| $r$ | $M_{r,1}$ | $M_{r,2}$ |
|---|---|---|
| 1 | −1.28 | 0.36 |
| 2 | −0.52 | 0.11 |
| 3 | 0.00 | 0.69 |
| 4 | 0.52 | 1.20 |
| 5 | 1.28 | 2.30 |

| $s$ | $r_s^1$ | $r_s^2$ | $X_{s,1}$ | $X_{s,2}$ |
|---|---|---|---|---|
| 1 | 1 | 5 | −1.28 | 2.30 |
| 2 | 4 | 2 | 0.52 | 0.36 |
| 3 | 3 | 1 | 0.00 | 0.11 |
| 4 | 2 | 4 | −0.52 | 1.20 |
| 5 | 5 | 3 | 1.28 | 0.69 |

stochastic-programming problems are typically sensitive to errors in the means, it is probably a good idea[5] to shift the generated values to get the correct expected values $\mu_i^*$ using $x_{s,i} \leftarrow x_{s,i} - \overline{\mu}_i + \mu_i^*$, where $\overline{\mu}_i$ denotes the actual mean of the $i$-th margin. We could extend this to adjust variances as well, $x_{s,i} \leftarrow \mu_i^* + \frac{\sigma_i^*}{\overline{\sigma}_i}(x_{s,i} - \overline{\mu}_i)$, but the benefit of such a correction is likely to be case-dependent.

# 5 Test case: CVaR-based portfolio optimization model

We test the heuristic on a simple portfolio-optimization problem, using a conditional value-at-risk (CVaR) as a risk measure. This is the model used in Kaut and Wallace (2011); we also use the same data, so the results are directly comparable.

The model decides investments $x_i \geq 0$ into financial instruments $i \in \mathcal{I}$, with stochastic returns $\tilde{r}_i$. The returns are modelled using discrete scenario values $R_i^s$ with probabilities $P^s$, for $s \in \mathcal{S} = \{1, \ldots, S\}$. The goal is to maximize the expected final wealth, given a constraint on CVaR. We normalize the problem so that the initial wealth is equal to one.

We use the standard LP formulation of CVaR from Rockafellar and Uryasev (2000), using auxiliary variables $z^s \geq 0$ for the shortfalls below the value of variable $\alpha$, which in turn is equal to the value-at-risk (VaR) at the optimum solution. The complete model is then:

$$\max_{x_i} \quad \sum_{s\in\mathcal{S}} P^s \sum_{i\in\mathcal{I}} R_i^s \, x_i \tag{13a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}} x_i = 1 \tag{13b}$$

$$z^s + \sum_{i\in\mathcal{I}} R_i^s \, x_i \geq \alpha\,, \quad \forall s \in \mathcal{S} \tag{13c}$$

$$\alpha - \frac{1}{1-\beta} \sum_{s\in\mathcal{S}} P^s \, z^s \geq \gamma\,, \tag{13d}$$

where the left-hand side of (13d) is equal to CVaR at confidence level $\beta$; in our case $\beta = 0.95$. The CVaR model was written in the GNU MathProg language (Makhorin, 2013b) and solved

[5] Except for distributions with limited support, where such a correction might give infeasible values.

by `glpsol`, both parts of GNU Linear Programming Kit (GLPK); see Makhorin (2013a). The data set used for the tests consists of 4476 daily prices of seven stock and three government bonds, which gives 4455 monthly returns; the data was kindly provided by Kjetil Høyland from DNB Nor, Oslo, Norway.

We use the model to compare three different scenario-generation methods: sampling from the historical data, the moment-matching algorithm from Høyland et al. (2003) and the heuristic presented in this paper. For the heuristic, the margins were obtained from the generated copula samples by applying an inverse of an interpolated version of the empirical cdfs, as described in item 2(b) on page 10. After the inverse, we have used a linear transformation to match the means and variances of the historical data; note that this makes a difference only for small sample sizes, as mentioned in the previous section. The same scaling was applied also to the sampled scenarios—without it, the sampled scenarios would perform significantly worse, even in the biggest test cases.

## 5.1 Test methodology

The three methods are compared in terms of *in-sample* and *out-of-sample* stability (Kaut and Wallace, 2007): for each method, we generate 100 scenario sets and solve the model (13) on each of them. We collect the resulting objective and CVaR values and look how much they vary between the different scenario sets—the less the better. This kind of stability is important for reporting of the results: with perfect in-sample stability (no variation at all), it would be sufficient to solve the model on one scenario set and report the obtained optimal objective value. If, on the other hand, the values vary significantly between the runs, we would have to solve the models on many scenario sets and then report the distribution of the obtained objective values—there would be no single optimal objective value.

If the tested scenario sets are small, the scenario-based objective function might not be a good approximation of the 'true' objective function. To get a better estimate of the true quality of the obtained solutions, one can evaluate them on some large *reference tree*: for each such solution $x^*$, we fix the model variables to $x_i \leftarrow x_i^*$ and re-solve the model on the reference tree; in our case, we have used the whole set of historical returns. This procedure is referred-to as the out-of-sample tests. Again, we want the obtained objective values to have as little variation as possible: with perfect out-of-sample stability, we know that it is sufficient to solve the model once. With significant instability, we would need to solve the model on many scenario sets, estimate the true quality of all the obtained solutions and then choose the best one.

Since our reference tree has less than five thousand scenarios, we can actually solve the model on this tree as well. We then use the obtained solution as a proxy for the true optimum, which gives us an even better way to evaluate the quality of the scenario-based solutions. We have solved the reference model with varying values of the CVaR bound $\gamma$, which has given us the 'true' efficient frontier for model (13).

### 5.2 Test results

We have tested the CVaR model with 50, 250, and 1000 scenarios. The new heuristic needed respectively 0.1 s, 0.4 s and 6.1 s to generate the scenarios, using the same hardware as in the previous tests. Note that with 10 random variables, there are $10 \times 9/2 = 45$ bivariate copulas to match.

Results of the stability tests, comparing the new heuristic to sampling and the moment-matching heuristic, are presented in Fig. 5. Each plot shows results of 100 scenario sets; the reason there seems to be fewer points in the copula-based charts is the inherently low randomness of the method: the only random element is tie-breaking at line 12 of the algorithm in Fig. 4, as described at the end of Section 3.

From Fig. 5, we can see that the copula-based method is clearly the best one: it gives better results with 50 scenarios than the other two methods achieve with 1000 scenarios. This, however, does not mean that we should use only fifty scenarios; as we can see from Fig. 6, we still need over one thousand scenarios to get consistently near-optimal solutions.

From the results with one thousand scenarios, we can also see that the moment-matching method leads to a bias towards a more conservative solutions, i.e. solutions with smaller profit and risk. This is caused by the fact that some of the data series have higher dependency in the down-turns than in the up-turns, which is not captured by correlations, combined with the fact that CVaR is sensitive to misrepresentation of the lower tail. It should be noted that the sign of the bias is unpredictable; when tested with different data, the moment-matching heuristic led to solutions with too much risk and profit.

We have repeated the same tests for a different value of the CVaR bound $\gamma$, as well as using a completely different data set, and received similar results. This increases our confidence in the quality of the presented heuristic, at least for this class of stochastic-programming models.

## Conclusions

In this paper, we have presented a heuristic for generating scenarios from distributions with the dependence between the margins described pairwise using bivariate copulas. Compared to using correlations, copulas give a better control of the dependencies. Our testing confirms that the scenarios generated using the presented method proved to be significantly better than scenarios generated with both sampling and a moment-matching method using correlations: in our test case, the new method needs only fifty scenarios to obtain better solutions than the other two methods produce with one thousand scenarios.

It remains to be seen how well the new heuristic performs for other types of stochastic-programming models. We have therefore made the code freely available from the author's web page, `http://work.michalkaut.net`, so others can test it on their problems.

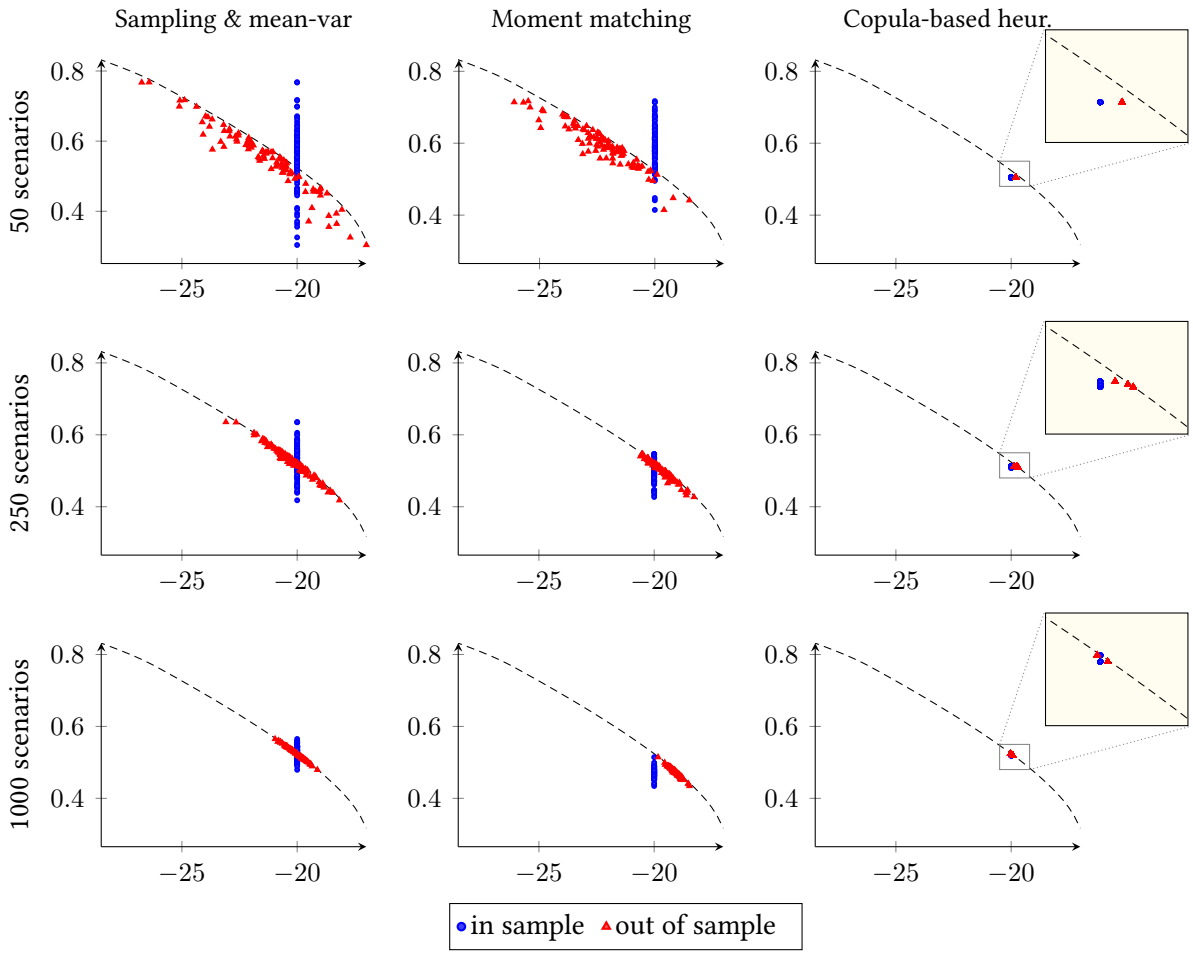


**Figure 5:** CVaR model: stability results for three different scenario-generation methods and increasing number of scenarios. CVaR is on the horizontal axis, expected return on the vertical. All values are in per cents, i.e. multiplied by 100.
Note how all the in-sample points '•' align along the vertical line CVaR $= -0.2$, showing that the constraint (13d) is always active.

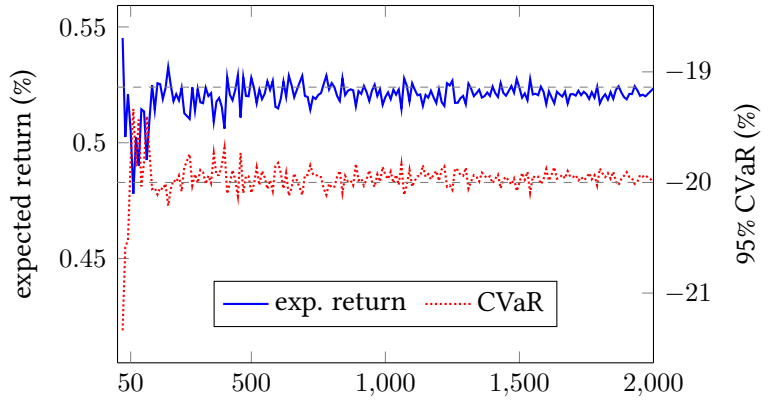


**Figure 6:** Out-of-sample expected return and CVaR for increasing number of scenarios. The thin dashed horizontal lines show the optimal values.

# References


Eric Bouyé, Valdo Durrleman, Ashkan Nikeghbali, Gaël Riboulet, and Thierry Roncalli. Copulas for finance: A reading guide and some applications. working paper, Crédit Lyonnais, Paris, 2000. Available at SSRN: http://ssrn.com/abstract=1032533.

Jitka Dupačová, Giorgio Consigli, and Stein W. Wallace. Scenarios for multistage stochastic programs. *Annals of Operations Research*, 100(1–4):25–53, 2000. ISSN 0254-5330. doi:10.1023/A:1019206915174.

John Forrest and Robin Lougee-Heimer. CBC user guide. In Harvey J. Greenberg and J. Cole Smith, editors, *Tutorials In Operations Research*, chapter 10, pages 257–277. INFORMS, 2005. URL http://test21.informs.org/Tutorials_CD/.

Kjetil Høyland, Michal Kaut, and Stein W. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2–3):169–185, 2003. doi:10.1023/A:1021853807313.

Ling Hu. Dependence patterns across financial markets: A mixed copula approach. *Applied Financial Economics*, 16(10):717–729, 2006. doi:10.1080/09603100500426515.

Tim H. Hultberg. FLOPC++ An algebraic modeling language embedded in C++. In Karl-Heinz Waldmann and Ulrike M. Stocker, editors, *Operations Research Proceedings*, volume 2006, pages 187–190. Springer, 2007. doi:10.1007/978-3-540-69995-8_31.

Michal Kaut and Stein W. Wallace. Evaluation of scenario-generation methods for stochastic programming. *Pacific Journal of Optimization*, 3(2):257–271, 2007.

Michal Kaut and Stein W. Wallace. Shape-based scenario generation using copulas. *Computational Management Science*, 8(1–2):181–199, 2011. doi:10.1007/s10287-009-0110-y.

François Longin and Bruno Solnik. Extreme correlation of international equity markets. *The Journal of Finance*, 56(2):649–676, 2001.

Andrew Makhorin. *GNU Linear Programming Kit – Reference Manual, Version 4.48*. Free Software Foundation, Inc., 2013a.

Andrew Makhorin. *GNU Linear Programming Kit – Modeling Language GNU MathProg, Version 4.48*. Free Software Foundation, Inc., 2013b.

Roger B. Nelsen. *An Introduction to Copulas*. Springer-Verlag, New York, 1998.

Andrew J. Patton. On the out-of-sample importance of skewness and asymmetric dependence for asset allocation. *Journal of Financial Econometrics*, 2(1):130–168, 2004. doi:10.1093/jjfinec/nbh006.

R. Tyrrell Rockafellar and Stan Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41, 2000.

Claudio Romano. Calibrating and simulating copula functions: An application to the italian stock market. working paper 12, Centro Interdipartimale sul Diritto e l'Economia dei Mercati, 2002.

Abe Sklar. Random variables, distribution functions, and copulas – a personal look backward and forward. In L. Rüschendorff, B. Schweizer, and M. Taylor, editors, *Distributions with Fixed Marginals and Related Topics*, volume 28 of *Lecture Notes – Monograph*, pages 1–14. Institute of Mathematical Statistics, Hayward, CA, 1996. URL http://www.jstor.org/stable/4355880.

Hajnalka Vaagen and Stein W. Wallace. Product variety arising from hedging in the fashion supply chains. *International Journal of Production Economics*, 114(2):431–455, 2008. doi:10.1016/j.ijpe.2007.11.013.