Forecast-based scenario-tree generation method

Michal Kaut*

March 10, 2017

Sometimes, the best available information about an uncertain future is a single forecast. On the other hand, stochastic-programming models need future data in the form of scenario trees. While a single forecast does not provide enough information to construct a scenario tree, a forecast combined with historical data does—but none of the standard scenario-generation methods is suited to handle this combination.

In this paper, we present a new scenario-generation method that combines a single forecast with historical forecast errors. The method is purely data driven and can take into account dependencies between errors of forecasts of different lengths.

Keywords: stochastic programming; scenario generation

Introduction

In stochastic programming, the standard way of describing the stochastic parameters is in the form of a scenario tree (Kall and Wallace, 1994; Birge and Louveaux, 1997; Pflug and Pichler, 2014). There are many methods to construct such trees, using different approaches to do so and requiring different type of information about the stochastic parameters (Dupačová et al., 2000; King and Wallace, 2012).

In this paper, we focus on the situation where the only (or the best) information about the future values of stochastic parameters comes in the form of a forecast. This can be a price coming from a deterministic equilibrium model, or a weather forecast in situation where the forecaster does not provide information about the uncertainty. In addition to the current forecast, we assume that we have access to historical forecasts and actual observed values, in order to estimate the the historical forecast errors.

*SINTEF Technology and Society, Trondheim, Norway; michal.kaut@sintef.no

In such a case, most of the standard scenario-generation methods cannot be applied, since they typically require more input data. In practice, this is usually tackled by one of the following approaches: if we have access to historical data and/or forecasts, one can try to find periods with forecasts or observations similar to the current forecasts and use data from those periods as extra scenarios. The challenge of this approach is the definition of 'similar', especially in multi-period and/or multi-variate cases. In addition, this approach generates a collection of scenarios (often referred to as a 'fan'), so one would need to use some reduction technique (Heitsch and Römisch, 2003; Dupačová et al., 2003; Heitsch and Römisch, 2009) if one needed multi-stage trees. An alternative is to construct the scenarios in the form $f_t \pm k\epsilon_t$ or $f_t(1 \pm k\epsilon_t)$, where f_t is the forecast and $k \in \{1, \ldots, K\}$. The step sizes ϵ_t are either fixed values, or they can be related to some measure of variability (such as the standard deviation) of historical forecast errors of the given length. Just like the above approach, this generates a 'fan', not a scenario tree.

A natural extension of the last method is computing the historical forecast errors and then using some of the standard scenario-generation tools to create a scenario tree for the error term. This tree would then be combined with the latest forecast to obtain a scenario tree for the future values. The challenge with this approach is that the historical forecast errors are different from usual historical data in that we normally have several forecasts errors for each observed value, coming from forecasts with different forecast lengths. In other words, the forecast errors do not form simple linear history, which prohibits direct use of standard data-analysis approaches.

In this paper, we describe a method that uses the historical forecast errors in a new way to generate the scenario trees for the errors. The method is purely data driven (no distributional assumptions) and allows modelling dependencies between errors of forecasts of different lengths.

1 The method

1.1 Notation

ω_t	historical observations (vector of size N)
$\tilde{\omega}_{t+\Delta t\|t}$	forecast for time $t + \Delta t$ made at time t
$\tilde{\varepsilon}_{t+\Delta t\|t}$	forecast error of $\tilde{\omega}_{t+\Delta t\|t}$
$\bar{\varepsilon}^s_{\Delta t}$	error of forecast of length Δt in scenario s
ω^s_t	scenario-tree value for scenario s at time t

The forecast error $\tilde{\varepsilon}_{t+\Delta t|t}$ can be defined in several different ways, dependent on the nature of the data. The most common choices are simple differences

$$\tilde{\varepsilon}_{t+\Delta t|t} = \tilde{\omega}_{t+\Delta t|t} - \omega_{t+\Delta t} \tag{1a}$$

and relative differences

$$\tilde{\varepsilon}_{t+\Delta t|t} = \frac{\tilde{\omega}_{t+\Delta t|t}}{\omega_{t+\Delta t}} - 1\,, \tag{1b}$$

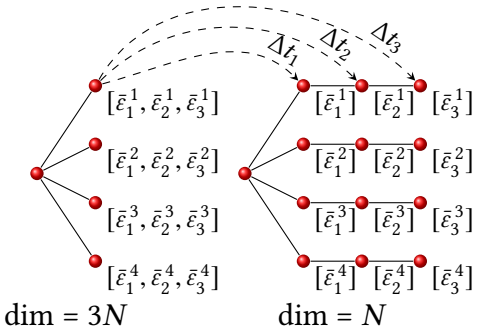


Figure 1: Generating a two-stage tree with four scenarios and three periods. Each $\bar{\varepsilon}_{\Delta t}^s$ is a vector of size N.

where the operations are meant to be done element-wise. Obviously, the latter definition cannot be used if zero is a feasible value for ω_t.

1.2 Two-stage case

Let's assume that we have historical forecasts $\tilde{\omega}_{t+\Delta t|t}$ for $t \in \{T_s, \ldots, T_e\}$ and $\Delta t \in \{1, \ldots, T\}$, together with observations ω_t for $t \in \{T_s, \ldots, T_e + T\}$. In addition, we have a forecast $\hat{\omega}_t$ for t periods ahead, i.e., for $t \in \{1, \ldots, T\}$. Based on this data, we want to generate S scenarios for T periods ahead.

First, we compute the historical forecast errors $\tilde{\varepsilon}_{t+\Delta t|t}$ for $t \in \{T_s, \ldots, T_e\}$ and $\Delta t \in \{1, \ldots, T\}$, using one of the formulas from (1). This means that for each time t from the historical data, we have T forecast errors for each element $n \in \{1, \ldots, N\}$. In total, we thus have $T\,N$ values for each $t \in \{T_s, \ldots, T_e\}$.

Now, we can view the historical forecast errors as a data set with dimension $T\,N$ and use any suitable method to generate S *two-stage* scenarios that approximate it; we denote the results by $\bar{\varepsilon}_{\Delta t}^s$. The next step is to 're-interpret' the two-stage scenarios with dimension $T\,N$ as scenarios T periods ahead with dimension N. This is illustrated in Fig. 1, with the generated two-stage tree on the left and the final, re-interpreted tree on the right.

The final step is to combine the generated prediction errors with the forecast $\hat{\omega}_t$, using the inverse of the formulas defining the forecast error. For the two examples from (1), this means

$$\omega_t^s = \hat{\omega}_t - \bar{\varepsilon}_{\Delta t}^s$$

and

$$\omega_t^s = \frac{\hat{\omega}_t}{1 + \bar{\varepsilon}_{\Delta t}^s}\,.$$

1.3 Extension to multi-stage trees

To generate multi-stage trees, we add the additional assumption that the employed two-stage scenario generator allows fixing of some margins to pre-specified values. This is not a limit-

ing assumption, since we can always take the fixed margins out of the scenario-generation process and generate scenarios for the rest.

The method is best explained on a simple example, illustrated in Fig. 2. There, the goal is to generate the three-stage tree from Fig. 2a. We start by identifying the first two-stage subtree (2b) and generating its values using the two-stage method described in the previous section. The next step (2c) duplicates the subtree in the target tree to remove the stage. Then we again identify the first two-stage subtree—in our case, this is the whole tree. Before generating scenarios for the new tree, we copy the values from the first two-stage tree and fix them (2d). Note that this means that the generated scenario tree will have duplicate values in the first two periods. Finally, after we have generated scenarios for the two-stage tree, we collapse the resulting 'fan' into the desired multi-stage tree. Note that we do not need any special reduction method for this, since all scenario-tree nodes we need to collapse are generated with equal values.

If the tree had more than three stages, we would repeat the steps (2c) and (2d) until we covered the whole tree. A pseudo-code for the complete algorithm is presented in Fig. 3. In the example in Fig. 2, we have $T = 4$ and the counter τ would have values 0, 2, and 4.

2 Our implementation

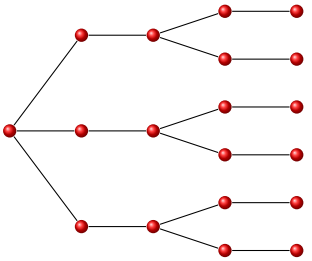
Our implementation uses the copula-based heuristic from Kaut (2014) as the two-stage scenario generator. This method generate scenarios by matching the (univariate) distribution functions of all the margins, plus pairwise dependencies using bivariate copulas. The latter can be done for all the margins, or only a specified subset. The main reasons for choosing this algorithm, apart from the fact that we are familiar with it, is that it can be made completely non-parametric, using only historical data. Moreover, the method works by adding one margin at a time, so it was easy to modify for this paper's multi-stage algorithm, where we need to fix the first τ margins.

The major challenge in using the method is the dimension of the generated scenarios, combined with the fact that the method matches pairwise copulas: with $N\,T$ stochastic variables (error terms), this means $N\,T(N\,T-1)/2$ pairs to match. For most applications, this is too many. Instead, we should focus on matching dependency only between pairs that we can expect to matter. For example, we might not be concerned about dependence between the error of the forecast of wind power at one location, one hour ahead, and the error of forecast at another location, 24 hours ahead.

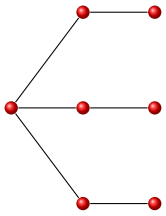
For this reason, we match dependency of a forecast error of variable x_i for forecasts Δt_1 ahead, denoted $(x_i, \Delta t_1)$, with:

- forecast errors of $(x_i, \Delta t_2)$ with $0 < \Delta t_2 - \Delta t_1 \leq U^{\mathrm{i}}$
- forecast errors of $(x_j, \Delta t_2)$ with $j \neq i$ and $\Delta t_2 - \Delta t_1 \leq U^{\mathrm{e}}$,

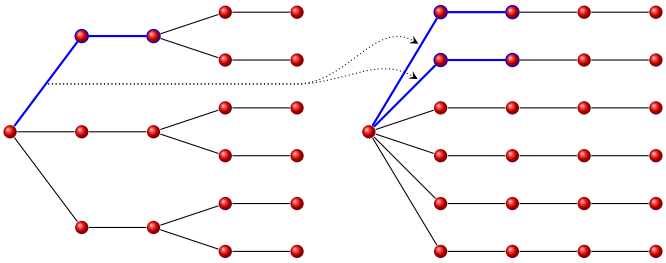
where $U^{\mathrm{i}} \geq 1$ and $U^{\mathrm{e}} \geq 0$ are case-dependent parameters. In our implementation, we use the minimal values, so we match dependencies between two consecutive forecast lengths for each variable and between all variables for each forecast length. This implies matching $N(T-1) + T\,N(N-1)/2 = N(N\,T+T-2)/2$ variable pairs.



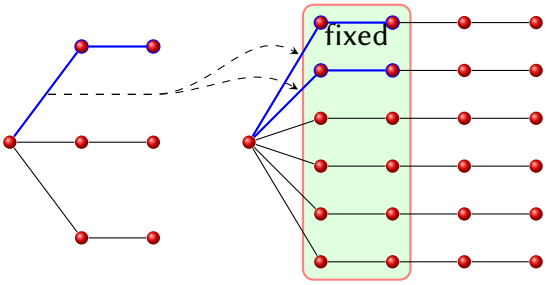
(a) the target three-stage tree



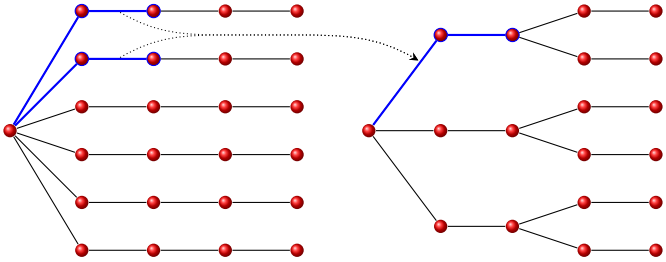
(b) first two-stage tree



(c) duplicating the subtree corresponding to (b)



(d) copying and fixing values from the first tree



(e) collapsing the equal nodes

Figure 2: Illustration of the method on a three-stage scenario tree

$\tau \leftarrow 0$
while $\tau < T$ **do**
  if $\tau > 0$ **then**
    duplicate subtree with $t \in \{1, \ldots, \tau\}$ to remove stage at τ ..... *See Fig. 2c.*
    copy values for $t \in \{1, \ldots, \tau\}$ from the last generated tree ..... *See Fig. 2d.*
  identify the first two-stage subtree
  generate values for the subtree, with the first τ periods fixed
  $\tau \leftarrow$ last period of the generated tree
collapse the tree ..... *See Fig. 2e.*

Figure 3: Pseudo-code of the multi-stage generator

We have integrated the presented method into the copula-based generator. The result is released under a dual LGPL/EPL open-source license and is available from the author's web page[1].

3 Test case

Ideally, we would like to test quality of solutions obtained from an optimization model using scenario trees generated with the presented method. This is not possible in our case, since have not yet integrated the method into the optimization model for which it was developed. Moreover, the optimization model is complex, making it difficult to separate the effects of the scenario-generation method. For this reason, we instead look at how well the generated scenarios represent the observed uncertainty, based on real data and forecasts. Note that, since our method adds noise to the externally provided forecast, the quality of the scenarios depends on the forecasting method, as well as the scenario-generation method.

We have tested the method on Norwegian electricity prices, with price forecasts from Statkraft, a major Norwegian power company. The data set includes forecasts generated at each working day from January 2014 until April 2016, always one week ahead with hourly resolution (168 observations), as well as the actual prices. From this we calculate relative forecast errors (1b) and then use a rolling window of the last 100 observations to estimate the distribution of forecast errors. This implies that we can test forecasts from April 2014, which gives us 450 test days.

At each day, we build a symmetric 5-stage scenario tree for the forecast errors, with 5 branches at $t \in \{1, 7\}$ and 2 branches at $t \in \{16, 25\}$; in total 100 scenarios. The forecast errors are then combined with the actual forecast at the test day, to create a scenario tree for the prices, one week ahead. We have used the relative errors from (1b) and estimated their distribution based on a rolling window of 100 days.

An example of such a tree is presented in Fig. 4, which shows a 'normal' week. The forecast was done on Friday, so the forecasted period is Saturday to Friday. In the chart, we can see the forecast, the 100-scenario tree, as well as the actual values.

[1] http://work.michalkaut.net/

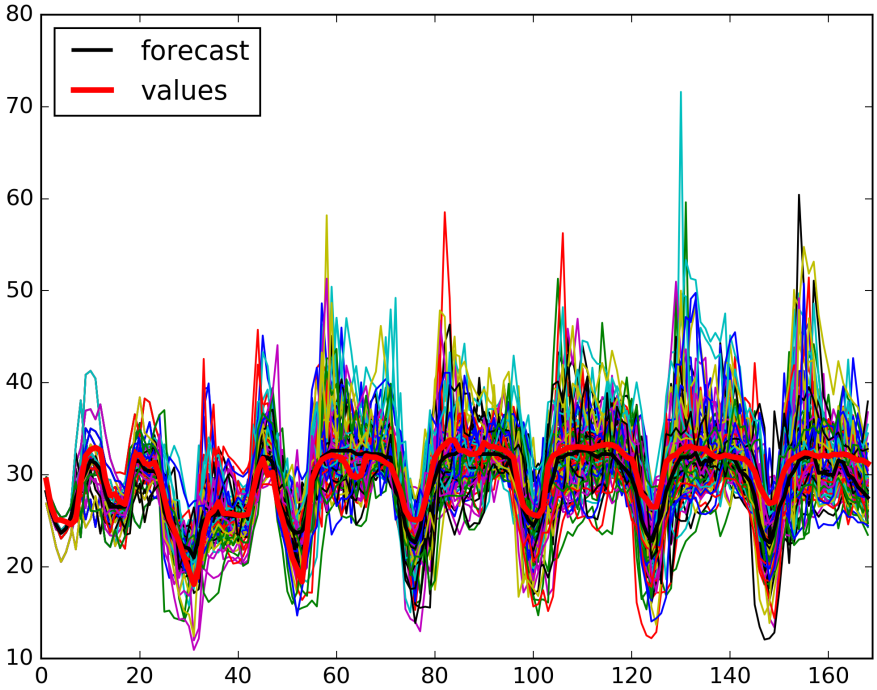


Figure 4: Example week showing the original forecast, 100 generated scenarios, and the actual observed prices.

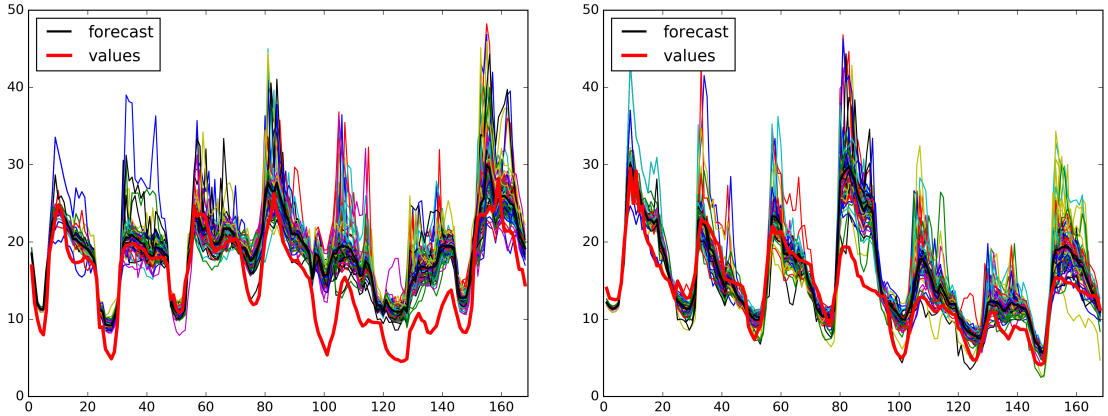


Figure 5: Two consecutive weeks, showing unexpectedly low prices and their effect on the scenarios.

Figure 5 shows how the method reacts to unexpectedly high forecast errors. In the left figure, we can see that the prices five and six days ahead are not only lower than the forecast, but also outside of the range covered by the scenarios. In the right figure, representing forecast one week later, we see that the scenario-generation method has reacted to the observed errors by increasing the downward variability of the longer forecast. As a result, when the prices again end up significantly lower than the forecast in the second half of the week, this drop is mostly covered by at least one scenario. In other words, if we use the method in an actual optimization model, the model would have taken into account the possibility of such a price drop. Naturally, the model reacts in the same way to unusual price peaks as well.

Naturally, we would like to know how well the scenarios represent the actual uncertainty. For this, we do the following percentile-based test: for each test day d and forecast length Δt, we compute the percentile of the *observed* value within the generated scenarios:

$$p_{d,\Delta t} = \left|\{s \in 1, \ldots, S_\Delta t : x^s_{\Delta t} \leq \bar{x}_{\Delta t}\}\right| / S_\Delta t \,, \tag{2}$$

where $x^s_{\Delta t}$ are the scenario values and $\bar{x}_{\Delta t}$ the actual observed value. This means that $p_{d,\Delta t}$ is zero/one if the value is lower/higher than in any scenarios.

Now, if the scenarios represented the uncertainty perfectly, we should see $p_{d,\Delta t} \leq q$ in $100\,q$ percent of the cases, for each Δt. To test this, we compute the frequencies of the quantiles

$$F_{\Delta t} = \left|\{d \in \mathcal{D} : p_{d,\Delta t} \leq q\}\right| / |\mathcal{D}| \,,$$

for each $q \in \{0, 0.1, \ldots, 1\}$ and then plot these against q. The result is a probability plot presented in Fig. 6. There, the horizontal axis represent q and the vertical axis $F_{\Delta t}$, and each line corresponds to one forecast length Δt.

We can see that most of the lines are close to the ideal line $(0, 1)$–$(1, 1)$. However, a couple of lines are different, with several intervals of constant values. These correspond to the shortest forecasts, where the scenario tree has only 5 scenarios. Since these are equiprobable, the values of $p_{d,\Delta t}$ must be multiples of 0.2. In other words, the percentiles have only five possible values, leading to the step-wise nature of the curves. For all the other forecasts, the lines are very close to the diagonal, so we can conclude that the scenario trees provide a good approximation of the actual distribution.[2]

Conclusions

We have presented a new scenario-generation method for situations where our information about future values of random parameters is limited to a single forecast. Our initial testing suggests that the method works as expected and generates scenario trees with distributions covering the actual observed uncertainty.

[2]Note that this is, in fact, a joint test of the forecasting method and the scenario generation. We could not achieve such good results if the forecasting method did not work.

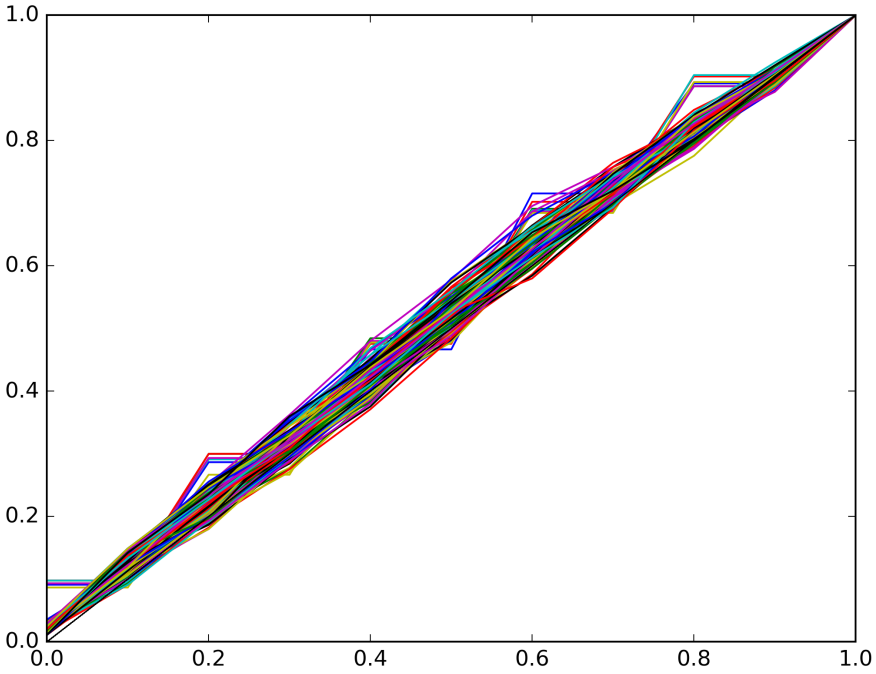


Figure 6: Probability plot showing actual vs. expected percentiles of the generated scenarios. Each line represents one forecast length.

Acknowledgements

This work was done as a part of KPN[3] project *Day-ahead Bidding with Multiple Short-term Markets*, a Research Council of Norway Project No. 243964/E20.

References

John R. Birge and François Louveaux. *Introduction to stochastic programming*. Springer-Verlag, New York, 1997. ISBN 0-387-98217-5.

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.

Jitka Dupačová, Nicole Gröwe-Kuska, and Werner Römisch. Scenario reduction in stochastic programming: An approach using probability metrics. *Mathematical Programming*, 95(3): 493–511, 2003. doi:10.1007/s10107-002-0331-0.

H. Heitsch and W. Römisch. Scenario reduction algorithms in stochastic programming. *Computational Optimization and Applications*, 24(2–3):187–206, 2003. doi:10.1023/A:1021805924152.

[3]From Norwegian 'Kompetanseprosjekt for næringslivet', meaning 'knowledge-building project for industry'.

Holger Heitsch and Werner Römisch. Scenario tree reduction for multistage stochastic programs. *Computational Management Science*, 6(2):117–133, 2009. doi:10.1007/s10287-008-0087-y.

Peter Kall and Stein W. Wallace. *Stochastic Programming*. John Wiley & Sons, Chichester, 1994.

Michal Kaut. A copula-based heuristic for scenario generation. *Computational Management Science*, 11(4):503–516, 2014. doi:10.1007/s10287-013-0184-4.

Alan J. King and Stein W. Wallace. *Modeling with Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, 2012. doi:10.1007/978-0-387-87817-1.

Georg Ch. Pflug and Alois Pichler. *Multistage Stochastic Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2014. doi:10.1007/978-3-319-08843-3.