

# Scenario Generation and Obtaining Sound Solutions

Arnt-Gunnar Lium

Arnt.G.Lium@hiMolde.no\*

Michal Kaut

mail@michalkaut.net †

September 27, 2006

## Abstract

For any optimization problem, the quality of the solution depends heavily on the quality of the input data. In stochastic programming, the input for the models is usually given in the form of scenario trees based on an underlying statistical distribution. We look at examples where the discretization of the underlying distributions gives unstable results that are too optimistic and turn out to be infeasible when tested using the underlying distribution. Examples and suggestions for dealing with such problems are presented.

**Keywords:** scenario generation, stability, stochastic programming, logistics

## Introduction

For solving a stochastic program, one typically needs to represent the underlying stochasticity by a discrete sample in the form of a scenario tree. Since most of the common scenario-generation methods are not deterministic, repeating the scenario-generation process (with the same underlying distribution) will result in different scenario trees. Intuitively, we would expect (or at least hope) that the solution of the stochastic program should not depend on our choice of scenario tree, i.e. that it is *stable* with respect to the scenario-generation process—otherwise we are solving a problem that poorly represents the true problem.

Stochastic programs tend to have flat objective functions, so there can be many different solutions with objective values close to the optimal one(s)—see for example Higle and Wallace (2003). Stability in the sense of obtaining the same (or similar) solution therefore has to be replaced by stability in the sense of obtaining solutions with (approximately) the same objective values. In addition, we consider two types of stability. First *in-sample* stability, where we compare the objective values as given by the stochastic programs themselves. While this may be the most “natural” approach, it measures the performance of the solution based on the—possibly incorrect—scenario tree. As a result, even if two programs with different scenario trees lead to the same solution, we might consider them different, as the observed objective values can be different. Hence, we also study *out-of-sample* stability, which is based on the “true” performance of the solution, usually given by some simulator. Both notions are taken from Kaut and Wallace (2003), which provide a detailed discussion on the stability of stochastic programs.

When we do not have stable solutions, it is an indication that the scenario trees we use do not incorporate all relevant aspects of the underlying stochasticity. In this paper, we study the situation where

---

\*Molde University College, P.O. Box 2110, NO-6402 Molde, Norway (corresponding author)

†Molde University College, P.O. Box 2110, NO-6402 Molde, Norway

it is difficult—if not impossible—to obtain stability with the standard scenario-generation methods. Further, we discuss what can be done when being forced to make decisions without having stability.

The rest of the paper is organized as follows: in Section 1, we present our test problem, test its stability, and try several ways of improving it. In Section 2, we compare the true performances of the selected methods and discuss the observed differences. Finally, Section 3 presents some possible ways of handling instabilities.

## 1 Improving stability – a test case

As a test case, we use a two-stage stochastic integer problem from logistics: the first stage consists of integer variables (truck routes) and the second stage consists of continuous variables (the flow of freight). Costs are associated with the first but not the second stage, i.e. the costs depend on the number of trucks and distances, not on freight flow. Stochasticity is in the demand for a given sub-set of commodities, a demand which has to be fully satisfied for all scenarios. In the test case, we consider twelve commodities, so we have a 12-dimensional random vector, i.e. each scenario includes twelve values. See Appendix A.1 for a full formulation of the problem.

To measure the “true” performance of the solutions, we use a large scenario tree (1000 scenarios), with margins sampled independently from triangular distributions. In other words, we declare this scenario tree to be the *truth* and refer to it as the *truth tree*. This *truth tree* is then used both as input for the scenario generation (used as historical data) and as a simulator for evaluating the performance of the solutions obtained by using small(er) scenario trees. (Ideally, this tree should have been used as input when solving our stochastic integer problem; unfortunately its size would give us an intractable problem.)

A consequence of having an integer program is that we must use far fewer scenarios due to the combinatorial nature of the problem compared to, for example, Kaut and Wallace (2003) who solved LPs.

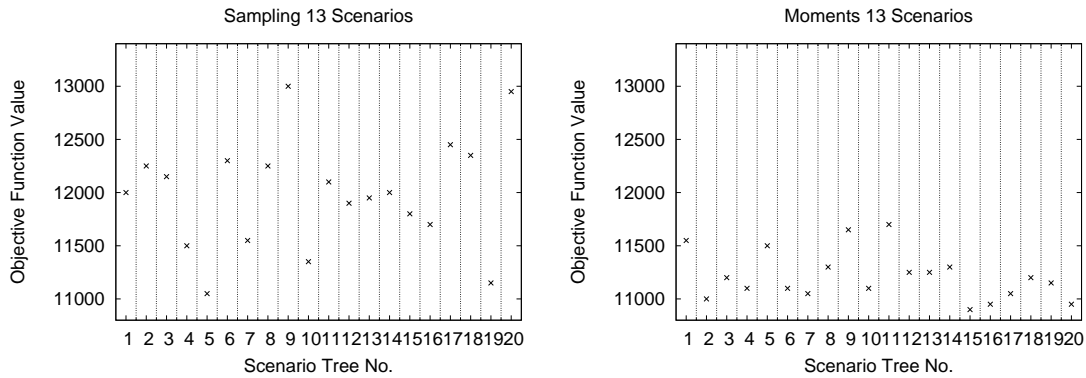
For scenario generation, we use two methods: sampling (with replacement) from the truth tree, and a property matching approach, in particular the moment-matching heuristics from Høyland, Kaut, and Wallace (2003). In the latter method, we estimate the first four moments of the marginal distributions, and in addition the correlation matrix, from the truth tree, and then generate scenario trees that match these values as closely as possible.

Figure 1 shows the in-sample instability of the optimal objective function values in the case of 13 scenarios. We can clearly see that, for these small trees, sampling leads to very unstable solutions—making a decision using any of these results would mean that the decision made depends not only on the underlying distribution but mostly on which tree is chosen. The moment-matching method produces more stable results, but we are still far from achieving in-sample stability. In such a case, Kaut and Wallace (2003) recommend two remedies: an increase of the size of the scenario trees, or a change/improvement of the scenario-generation method. In the following, we try both.

### 1.1 Increasing the number of scenarios

We increased the number of scenarios in the trees in steps of 8, and tested stability of trees with 13, 21, 29, . . . , 53 scenarios. To test in-sample stability, we generated 20 different trees of each size.

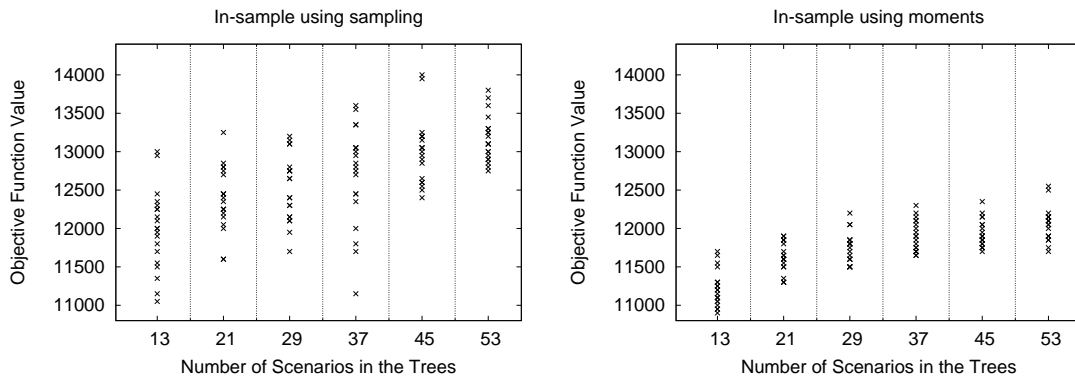
The results of the tests are in Figure 2. We can see that, in the case of sampling, increasing the number of scenarios seems to reduce the variance in the objective function values, with the conse-



**Figure 1:** In-sample optimal objective function values using *sampling* and *moments*. 20 different trees, each consisting of 13 scenarios

quence that there are fewer incidents of extremely high or low objective function values (relative to the other observations). In addition, there is an increase in the optimal objective function values as more scenarios are used. This increase may indicate that not all aspects of the stochasticity given by the truth tree are being captured by the smaller trees. Or, put in another way, solutions based on few scenarios seem to overestimate their own performance, as they “do not grasp” all aspects of the underlying uncertainty. A similar tendency can be observed for trees generated by the moment-matching method and has also been described in Kaut and Wallace (2003).

We can also note that the objective function values of solutions based on sampling are in general much higher than the ones of solutions based on moment matching. We will come back to this observation in Section 2.



**Figure 2:** In-sample optimal objective function values using *sampling* and *moments*, for different numbers of scenarios.

We can conclude that even if increasing the size of scenario trees improves the results, we are still far from achieving in-sample stability. We should also note that the moment-matching method leads to significantly more stable solutions than sampling—so maybe another method would be even better?

## 1.2 Improving the scenario generation method

When looking at Figure 2, we note that the more scenarios used the higher the objective function values become. This is interesting, as a higher number of scenarios should mean more information about the underlying problem. If this information could be provided by increasing the *quality* of the scenarios instead of their number, it would leave us with a smaller problem to solve. In this section, we will thus try to improve the scenario generation procedure, in our case the moment-matching method. To find out what should be improved, we first have to analyze the model and identify the possible source(s) of instability.

In our stochastic problem, costs are connected only to the integer variables in the first stage and not to the continuous variables in the second stage. However, as the second-stage decisions strictly constrain the first-stage variables, they indirectly drive the costs by adding capacity in the first stage. A consequence of having costs connected to the integer variables in the first stage is a piecewise continuous objective function where small changes in the random variables can cause jumps in the objective function value—increasing the demand a bit might lead to an increase in the number of trucks—making stability difficult to obtain.

In particular, it is enough to have one scenario with an unrealistically high demand (higher than in any scenario of the truth tree) to drive the number of needed trucks—and thus the costs—upwards. In other words, we should never generate values outside the support provided by the values of the truth tree. Unfortunately, the moment-matching method from Høyland et al. (2003) provides only control of the moments of the marginal distributions, so the generated values can fall outside the support. Adding this feature could therefore improve the performance of the method in our case.

On the other hand, why should we stop with moments and support? By visually analyzing histograms of the marginal distributions of the *truth tree* (see Appendix A.2), we note that the distributions are close to triangular distributions.<sup>1</sup> To be able to use this information, we have modified the scenario-generation method in such a way that the marginal distributions are specified by their distribution functions; instead of their moments. The workings of this new method are out of the scope of this paper, and can be found in Kaut and Lium (2006).

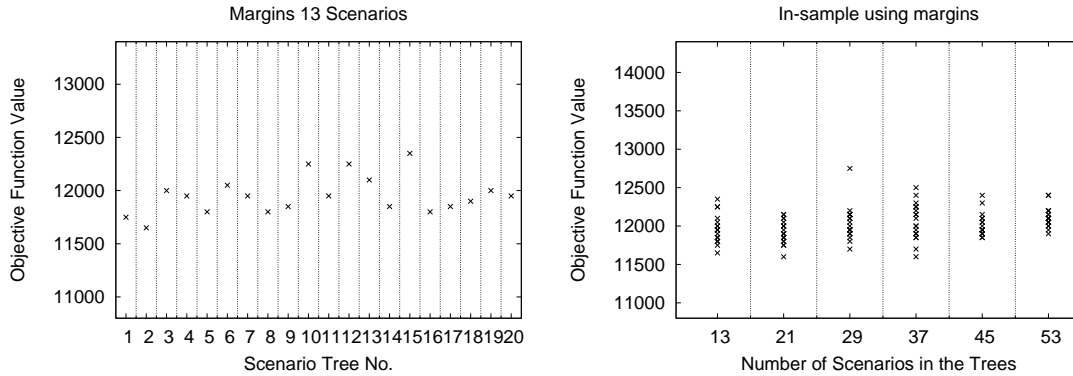
With this new method, we thus have to assume a distribution family of the margins (in our case triangular) and estimate its parameters (in our case min, max and mode), together with the correlation matrix, from the truth tree. The method then generates a scenario tree that matches the given input as closely as possible. Since we control the distribution functions, the support is obviously controlled as well.

The results of the new method are presented in Figure 3 and indicate a promising performance, compared to Figures 1 and 2. We can note that there is a decrease in the variance of the objective function values when moving from sampling to moments and to margins.

We also notice that, unlike sampling and moment-matching, the new method does not lead to an increase in the optimal objective function value as more scenarios are being used. This can be interpreted in two different ways: one possibility is that the new method simply fails to converge, and consider it a problem. On the other hand, we could interpret it so that the new method manages to include enough information even in the small trees, so the optimal objective value does not need to converge from the too-optimistic values to the true ones, as is the case with the other methods. This interpretation is supported by the fact that the objective value is around 12000, a level to which the results based on moments seem to converge (but reach it first with 37 scenarios), see Figure 2.

---

<sup>1</sup>This is hardly surprising in our case, as we have generated the truth tree ourselves by sampling from the triangular distribution.



**Figure 3:** In-sample optimal objective function values using *margins*. 20 different trees each consisting of 13 scenarios (left) and an increasing number of scenarios (right).

To decide which of the interpretations is correct, we have to evaluate the true performance of the solutions.

While we have managed to decrease the instability slightly, we have not achieved full in-sample stability. In fact, we are forced to admit that, with our scenario-generation methods, it seems to be unattainable. Instead we analyze the results in more detail and try to understand the differences between the three tested methods.

## 2 Evaluating the solutions

Having found the optimal solutions using the small trees, we know only their in-sample objective values and whether they are stable plus, of course, the solutions themselves. We do not know how good the solutions are when encountering the *truth* and we must therefore evaluate the performance of the solutions using the *truth tree* or, using the language of Kaut and Wallace (2003), perform an *out-of-sample* test.

However, combining a solution with a different scenario tree from the one used to produce the solution can easily lead to infeasibility, i.e. there can be scenarios with demand exceeding the capacity of the truck routes given by the solution. To avoid this, we introduce slack variables that allow us to maintain feasibility even when capacity is violated. The test is then done in the following way: we fix the integer variables of the solution, move the slack variables into the objective function and attach a “dummy” cost to their use. We now have an LP problem minimizing the “dummy” cost, which can be solved to optimality using the large *truth tree* as an input. For practical reasons, the “dummy” cost is chosen so that the objective function values of the LPs are equal to the actual slack required to maintain feasibility.

### 2.1 Performance of the scenario generation techniques

Evaluating the performance of the solutions is not straightforward as there is no clear and undisputable way of measuring it. How we measure the performance strongly affects the results we obtain. Focusing only on feasibility will imply that the most robust solution is perceived as the best one, independently of its cost. Focusing on cost efficiency can make less robust but cheaper solutions to be perceived as better. Our integer problem can thus be seen as a risk-aversion problem, trading

feasibility against investment costs.

The results of the evaluations are presented in Figure 4. Note that, unlike the previous figures, the y-axis now represents the total sum of all the slack variables over all scenarios. The first observation is that the slack required to maintain feasibility in the solutions based on sampling drops significantly when increasing the number of scenarios from 13 to 21 and continues to do so as more scenarios are used. On the other hand, we have to remember that this improvement in feasibility comes at a cost, since it corresponds to the increase of the in-sample objective function value (the cost of operating the trucks)—see Figure 2.

Similar, but much less extreme, results can be seen in the case of moment matching, where the observed decrease in infeasibility also corresponds to an increase in the in-sample objective function value. Unlike sampling, however, the variance of the infeasibility is not decreasing with an increasing number of scenarios. In addition the infeasibility does not seem to converge to zero, but to some “true” value, suggesting the moment matching method does not capture the shape of the distribution properly.

Just like in the in-sample case, the out-of-sample values of solutions based on trees generated using the marginal distribution functions are unaffected by the number of scenarios used to produce the solution. We can also notice that both the in-sample and the out-of-sample values of the solutions based on moment-matching seem to converge to a level that the new method attains already with 13 scenarios. This is good news, as it means we can use fewer scenarios, without it having an adverse effect on the quality of the solutions. Unfortunately, it also means that the method has the same problem as moment matching, i.e. it does not properly capture the shape of the underlying distribution, leading to a rather high (even if constant) level of infeasibility.

## 2.2 Why do the scenario generation techniques perform differently?

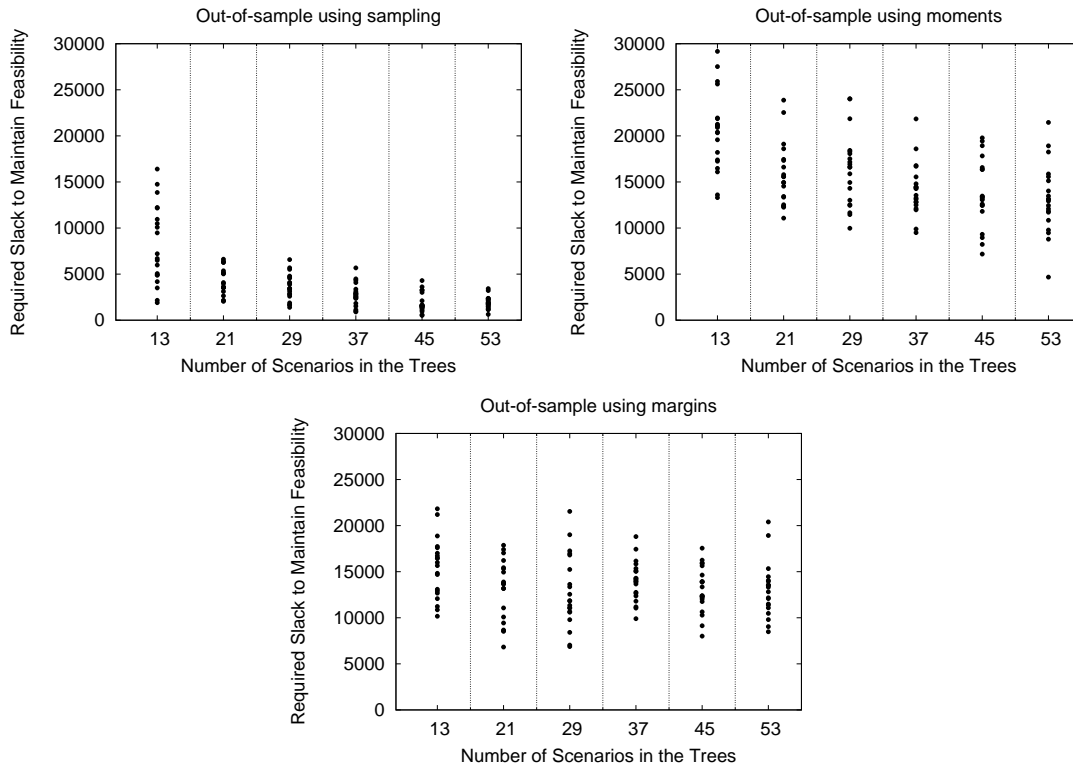
Using scenario trees based on sampling as input has been reported to produce poor and unstable results. This led to work on alternative scenario generation procedures—see for example Kaut and Wallace (2003), Kouwenberg (2001). Our results, in-sample, are in line with previous experience with sampling producing more unstable results compared to the two alternative methods. However, out-of-sample, i.e. looking at the actual performance, we note that sampling is by far the best one, with respect to stability and feasibility. This looks like a paradox, the least stable solution in-sample turn out to be the most stable out-of-sample.

One possible explanation is in the size of the *truth tree*: it consists of 1000 scenarios, which is too little to give a good approximation of a 12-dimensional distribution. As a result, the multi-dimensional shape of the distribution represented by the truth tree may be very strange and, more importantly, very far from a shape of a continuous 12-dimensional triangular distribution with independent margins. Consequently, both property-matching methods (moment matching and our new method) are unable to pick up the shape, as they, basically, assume that all the non-specified properties—shape being one of them—are “nice” and/or close to the ones of a normal distribution.<sup>2</sup> Sampling, on the other hand, from its very nature does not have any problems picking up the shape, converging slowly but surely to the true distribution.

Høyland et al. (2003) report that they obtained stability using their procedure, we can note from Figures 2 and 4 that it is not the case in our problem. While they worked with linear problems with

---

<sup>2</sup>This is a consequence of the algorithm from Høyland et al. (2003), not a general property of moment-matching methods. The reason is, with some simplification, that the method starts with a normal distribution and changes the specified properties to their target values, leaving the rest as it is, or close to it.



**Figure 4:** Out-of-sample objective function values of solutions based on *sampling*, *moments*, and *margins*.

continuous decision variables, we deal with integer decision variables. One reason for instability in our problem is that the use of an objective function consisting only of integer variables causes the objective functions to be step-wise constant as a function of the random variables. Hence, a change in demand may not change the truck routes at all, but can also result in a need for additional trucks (or that the “original” number of trucks are being used in another way at a different cost). While we have tried to improve the situation by incorporating the support of the distribution into our scenario-generation procedures, it turned out not to be enough. One possible explanation for the instability is that some decision variables can be quite sensitive to the actual realization of a certain combination of random variables, compared to other decision variables. For example if random variable number 1 and 2 take on a high value in one or more of the scenarios, it can have a more significant impact on the (integer) decision variables compared to if, lets say, random variable 1 and 3 take on similar high values.

### 3 Discussion

Ideally one would prefer to end up with both in- and out-of-sample stability, which would make sure that the solutions are consistent. Of course, solutions that are in-/out-of-sample stable can be wrong but at least they would be consistently wrong and not an “accidental result” of the scenario generation procedure. Not having in- or out-of-sample stability, the most obvious question is: “what do we do”?

### 3.1 Suggestions for choosing a solution

Having a tool such as a *truth tree* or a simulator makes it possible to evaluate the performance of the scenario generating method, making the choice less complicated. If we want to minimize infeasibility, for instance, we would choose one of the solutions obtained using 45 sampled scenarios, requiring 14205 units of slack to maintain feasibility—unfortunately at the relative high cost of 13250. On the other hand, if the in-sample objective value is the most important objective, we would choose one of the solutions obtained using 13 scenarios generated by the moment-matching method. The in-sample objective value would then be 10900, but the slack required to maintain feasibility is 27516 units. (In comparison the worst case requires 40319 units of slack.)

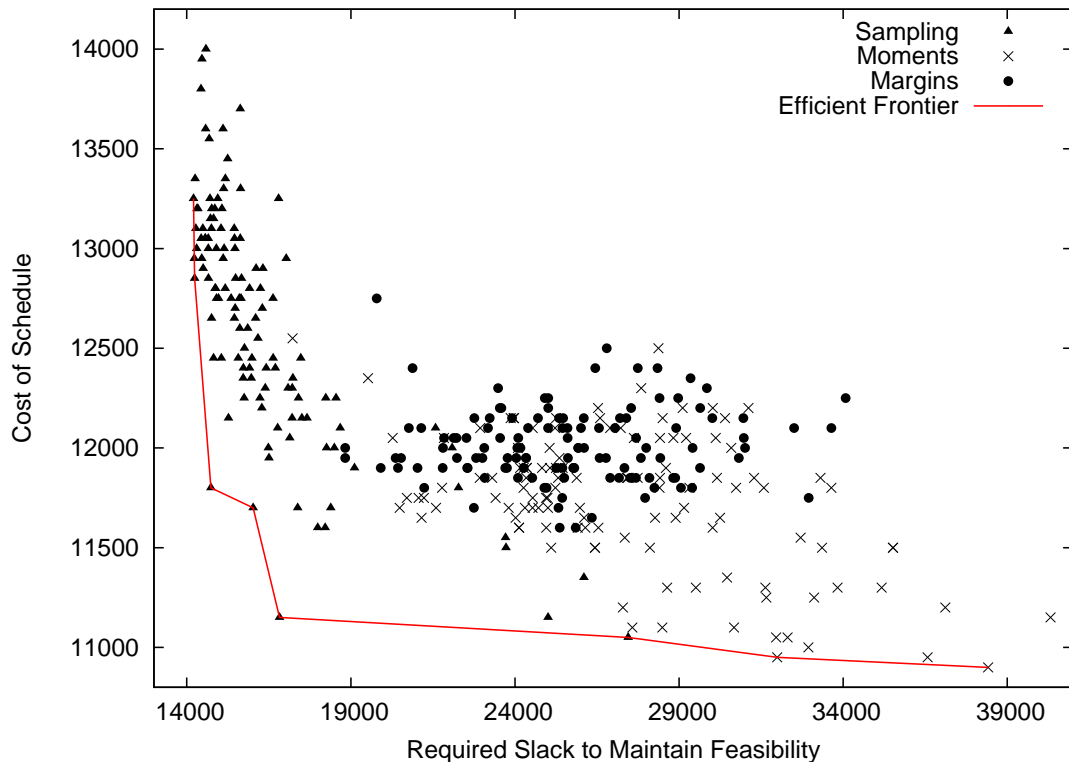
There are situations where there is a tradeoff between the immediate cost of a solution (in-sample) and its feasibility. The consequence of this is that choosing a more expensive solution can mean less violation of feasibility. However, due to the lack of in- and out-of-sample stability it is not guaranteed that choosing a more expensive solution will reduce infeasibility, at worst a solution can turn out to be more infeasible than a less expensive alternative. To omit this problem we display the cost of the solutions and their performance out-of-sample (Figure 5), and let the user choose the most suitable one with respect to cost and risk.

When looking at Figure 5 we note that many of the solutions created using moments and margins are far from the efficient frontier, which indicates that they do not grasp all of the underlying stochasticity. A very different performance can be observed on one of the solutions on the frontier, namely the one that cost 11150 and requires 16828 units of slack to maintain feasible. Compared to the other solutions either having approximately the same cost or risk, this is significantly better. There are no other solutions that come close in performance. A timely question in this respect would be why the solution is so good compared to the others, is it because it is so good or is it because the others are so poor? To the authors it seems that the scenario tree used to create the solution was of a higher quality compared to the others.

Note that the situation would be simplified if we could put a price (penalty) for not delivering freight, instead of the requiring the demand to be met in all scenarios. This would help us in more than one way: it would put all the costs into the objective, so there would be no problem choosing the best solution. In addition, we would get some continuous variables into the objective, which is likely to improve the continuity of the objective with respect to the random variables. On the other hand, to use this solution we would have to know the exact costs of not delivering, as they will have a strong influence on the solution.

There are situations when the decision maker does not have access to a *truth tree* or other “impartial” methods to evaluate the performance of the solutions. One possible proxy is to merge all the smaller trees used to produce the solutions and use the new tree in a similar way as we used the *truth tree*. However, using the same scenario trees to produce the solutions and to test their own performance can lead to errors. Alternatively, the decision maker can use a “cross-testing”, i.e. to take a solution from one scenario tree and test it using the other small trees, instead of the *truth tree*. The user can then choose the solution with most stable performance.

And what if the decision maker has no possibility to evaluate the performance of the solutions, before deciding on which to use, for example because of some properties of the model? Is it still possible to come with a general advice on which solution to choose? The intuitive reasoning in this case could be that not knowing anything about the performance of a solution, it makes sense to choose the one that appear to be the cheapest, i.e. the solution that is best in-sample. However, one of the strengths of stochastic programming is that it gives us robust solutions, rather than solutions that turn



**Figure 5:** The tradeoff between immediate cost and required slack to maintain feasibility

out to be very poor or very good in the hindsight. The use of—what seems to be—the cheapest solution can to some extent undermine this, as the performance out-of-sample of such a solution can be somewhat “accidental”.

In addition to performing numerical out-of-sample tests, we would like to stress the importance of experts or practitioners with practical knowledge of the problem. Such an expert can often provide a useful insight regarding the true performance of the solution. Typically this approach will produce statements like “*this solution is good/poor because...*”, which is useful when choosing the solution to be used. A slightly different approach is to look for good structures in the solution and find out which solution provides the best structures to deal with different realizations of the random variables. See Lium, Crainic, and Wallace (2005) for an example of such an approach.

## Conclusion

Decision support based on a potentially wrong input is of a limited value to the decision maker. It is not to say that the advice cannot be good, it can, but one may not *know* whether it is good or not. Being able to test a solution before implementation, for example by simulation or with the help of a *truth tree*, is helpful when deciding which solution to use. However, such quantitative forms of evaluation are not always possible, with the consequence that the decision maker may not know how the solution is expected to perform. One alternative is to make experts/practitioners analyze the solution candidates to see, in their view, which is the best.

There are situations where neither quantitative nor qualitative forms of pre-evaluation can be performed, with the implication that the decision maker will not know how the different solutions are expected to perform. Not being able to know which tree to use, all that can be done is to make sure that the tree is as good as it can be. That implies solving the problem using as many scenarios as possible, to give the best possible description of the underlying stochasticity with the methods currently available. Solving the problem using different number of scenarios (as we did) can give indications of whether the solution overestimates its own performance or not. If an increase in the number of scenarios leads to an increase in the objective function value there is an overestimation of the solution performance. However, this overestimation may not be detected with the use of only one scenario generation technique as some of the techniques result in objective function values that are unaffected by changes in the number of scenarios used. This can be good, if the resulting objective function values are correct, but the user might not know whether this is the case or not. Hence, the use of alternative scenario generation techniques can be useful to provide “guidance” to achieve better estimates of the *true* objective function value—estimates that hopefully lead to good decisions.

## Acknowledgments

We would like to thank Stein W. Wallace for his many ideas and suggestions and Halvard Arntzen for pointing out errors and providing suggestions to improve the paper.

## References

- J.L. Higle and S.W. Wallace. Sensitivity analysis and uncertainty in linear programming. *Interfaces*, 33:53–60, 2003.
- K. Høyland, M. Kaut, and S.W. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2-3):169–185, 2003. ISSN 0926-6003.
- Michal Kaut and Arnt-Gunnar Lium. Scenario generation with correlations and general margins. not yet finished, 2006.
- Michal Kaut and Stein W. Wallace. Evaluation of scenario-generation methods for stochastic programming. Stochastic Programming E-Print Series, <http://www.speps.info>, May 2003.
- R. R. P. Kouwenberg. Scenario generation and stochastic programming models for asset liability management. 134(2):51–64, 2001.
- A.-G. Lium, T. G. Crainic, and S.W. Wallace. Stochastic service network design: The importance of taking uncertainty into account. *Center for Research on Transportation, Working Papers*, 2005. To be published as a working paper at Center for Research on Transportation.

## A Appendix

### A.1 Model formulation

In our tests we used a stochastic and dynamic service network design formulation typically used in the tactical planning for a less-than-truckload trucking carrier. The model formulation use a space-

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

We describe the uncertainty in demand for our commodities by using scenarios  $s \in \mathcal{S}$ , where each scenario has probability  $p^s \geq 0$ , with  $\sum p^s = 1$ , and  $p^1 = p^2 = \dots = p^s$ . A scenario is  $|\mathcal{K}|$ -dimensional, as it contains one demand  $\lambda(k, s)$  for each commodity  $k \in \mathcal{K}$  in each scenario. In addition each commodity  $k \in \mathcal{K}$  is defined by its origin  $o(k)$  and destination  $d(k)$  nodes/terminals, as well as the time periods  $\sigma(k)$  and  $\tau(k)$  when it becomes available at its origin and when it must be delivered (at the latest) at its destination, respectively. We assume uniform vehicles with a capacity  $M$ , which is the same units as for demand.

The two main sets of decision variables are:

$X_{ij}^t$ : Number of trucks from node  $i$  in period  $t$  to in node  $j$  in period  $t+1$ ,  $\forall i, j, t$ .

$Y_{ij}^t(k, s)$ : Amount of commodity  $k$  in scenario  $s$  going from terminal  $i$  in period  $t$  to terminal  $j$  in period  $t+1$ , for  $\{i = o(k), t = \sigma(k), \forall j\} \cup \{j = d(k, s), t = \tau(k), \forall i\} \cup \{\sigma(k) < t < \tau(k), \forall i, j, s\}$ ;

The formulation is written in a circular fashion to deal with potential end of horizon problems. To improve readability, we present constraints according to the difference  $\Delta t$  in number of time periods between  $\sigma(k)$  and  $\tau(k)$ .

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{t \in \mathcal{T}} c_{ij} X_{ij}^t \quad (1)$$

$$\sum_{i \in \mathcal{N}} X_{ij}^t = \sum_{i \in \mathcal{N}} X_{ji}^{(t+1) \bmod T} \quad \forall t \in \mathcal{T}, \forall j \in \mathcal{N} \quad (2)$$

$$\sum_{k \in \mathcal{K}} Y_{ij}^t(k, s) \leq M X_{ij}^t \quad \forall (i, j) \in \mathcal{N}^2 : i \neq j, \forall t \in \mathcal{T}, \forall s \in \mathcal{S} \quad (3)$$

$$\sum_{j \in \mathcal{N}} Y_{o(k)j}^{\sigma(k)}(k, s) = \lambda(k, s), \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \quad (4)$$

$$\sum_{i \in \mathcal{N}} Y_{i,d(k)}^{(\tau(k)-1) \bmod T}(k, s) = \lambda(k, s), \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \quad (5)$$

$$(\Delta t = 2): Y_{o(k),j}^{\sigma(k)}(k, s) = Y_{j,d(k)}^{(\sigma(k)+1) \bmod T}(k, s) \quad \forall j \in \mathcal{N}, \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \quad (6)$$

$$(\Delta t \geq 3): Y_{o(k),j}^{\sigma(k)}(k, s) = \sum_{i \in \mathcal{N}} Y_{ji}^{(\sigma(k)+1) \bmod T}(k, s) \quad \forall j \in \mathcal{N}, \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \quad (7)$$

$$(\Delta t \geq 3): \sum_{i \in \mathcal{N}} Y_{ij}^{(\sigma(k)-2) \bmod T}(k, s) = Y_{j,d(k)}^{(\tau(k)-1) \bmod T}(k, s) \quad \forall j \in \mathcal{N}, \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \quad (8)$$

$$(\Delta t \geq 4): \sum_{i \in \mathcal{N}} Y_{ij}^{(t-1) \bmod T}(k, s) = \sum_{i \in \mathcal{N}} Y_{ji}^t(k, s) \quad \forall t \in \mathcal{T}, \forall j \in \mathcal{N}, \forall k \in \mathcal{K}, \forall s \in \mathcal{S} \quad (9)$$

$$X_{ij}^t \geq 0 \text{ and integer}, \quad \forall t \in \mathcal{T}, \forall i, j \in \mathcal{N} \quad (10)$$

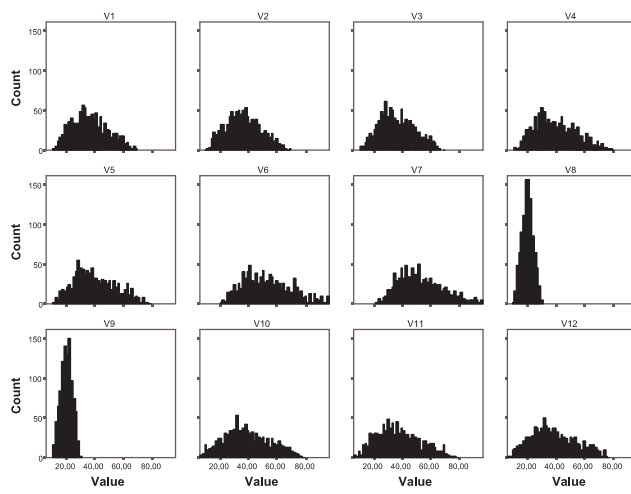
$$Y_{ij}^t(k, s) \geq 0 \quad \forall (i, j) \in \mathcal{N}^2, \forall k \in \mathcal{K}, \forall t \in \mathcal{T}, \forall s \in \mathcal{S} \quad (11)$$

The objective function (1) minimizes the cost of moving the vehicles between the terminals and their holding cost at the terminals (for vehicles parked at the terminals). Constraints (2) represent conservation of flow for trucks. Relations (3) are the usual linking and vehicle capacity constraints. Note that commodities can be held at nodes without a truck being present as  $i \neq j$ . Constraints (4) and

(5) represent the conservation of flow at the origin and destination nodes of a commodity, respectively. (6) is a conservation of flow constraint for the commodities when there are just two periods between  $\sigma(k)$  and  $\tau(k)$ . Equation (7) represents conservation of flow conditions at nodes one period after a commodity has left its origin node. The flow may come only from node  $o(k)$  in period  $\sigma(k)$ , but may go to any node in period  $\sigma(k) + 1$ . Similarly, equations (8) enforce conservation of flow at nodes one period before flow arrives at destination. Equations (9) are the general conservation of flow constraints. They are valid from the second period after a commodity has left its origin and up to two periods before arriving at its destination. (10) makes sure that non-negativity and integrality for the trucks is maintained while (11) is a non-negativity constraint for the flow of commodities.

## A.2 The truth tree

### Histograms of the truth tree



**The truth tree correlation matrix**

1.000	-0.007	-0.016	-0.007	-0.002	0.022	0.009	-0.044	0.017	0.003	-0.003	0.054
-0.007	1.000	0.029	0.020	0.063	0.004	-0.050	-0.033	0.022	-0.036	0.000	-0.018
-0.016	0.029	1.000	0.018	-0.042	-0.026	-0.014	0.001	-0.030	0.046	-0.026	-0.009
-0.007	0.020	0.018	1.000	-0.044	0.051	-0.010	-0.004	0.049	0.017	0.022	0.042
-0.002	0.063	-0.042	-0.044	1.000	0.013	-0.009	-0.007	0.005	0.035	-0.042	0.041
0.022	0.004	-0.026	0.051	0.013	1.000	0.041	0.009	-0.036	-0.008	-0.001	-0.023
0.009	-0.050	-0.014	-0.010	-0.009	0.041	1.000	-0.054	0.008	0.047	0.016	0.008
-0.044	-0.033	0.001	-0.004	-0.007	0.009	-0.054	1.000	0.008	0.046	0.016	0.032
0.017	0.022	-0.030	0.049	0.005	-0.036	0.008	0.008	1.000	0.013	0.012	0.073
0.003	-0.036	0.046	0.017	0.035	-0.008	0.047	0.046	0.013	1.000	0.005	0.038
-0.003	0.000	-0.026	0.022	-0.042	-0.001	0.016	0.016	0.012	0.005	1.000	-0.046
0.054	-0.018	-0.009	0.042	0.041	-0.023	0.008	0.032	0.073	0.038	-0.046	1.000

**The margins calculated from the truth tree**

Random Variable	Min	Max	Mode
RV1	11.36417	68.63482	29.41272381
RV2	10.29005	69.65291	29.21754854
RV3	10.64078	68.63514	29.23432171
RV4	10.54221	79.47084	29.62187472
RV5	10.78228	78.45055	31.20441136
RV6	20.57144	96.36365	42.13650293
RV7	21.88128	96.44737	40.02644969
RV8	10.1788	29.81557	19.44593959
RV9	10.38797	29.29183	20.51094606
RV10	5.63092	79.03162	31.75073788
RV11	6.23565	78.75777	26.86868787
RV12	6.5983	77.42379	30.66814994

**The moments calculated from the truth tree**

Random Variable	Mean	SD	Skewness	Kurtosis
RV1	36.47057127	12.60792202	0.315323469	2.408922033
RV2	36.38683618	12.26458919	0.238765555	2.428501871
RV3	36.17008057	12.30325875	0.3186025	2.392315233
RV4	39.87830824	14.59656137	0.389824825	2.420594263
RV5	40.14574712	14.8078755	0.358521542	2.327246924
RV6	53.02386431	16.95038639	0.440251235	2.506259806
RV7	52.78503323	15.72744129	0.519019478	2.696417211
RV8	19.81343653	4.04104733	0.034472974	2.511571461
RV9	20.06358202	4.061498319	-0.053427689	2.368795085
RV10	38.80442596	16.05284121	0.236924091	2.341612321
RV11	37.28736929	15.29095326	0.342886091	2.414006668
RV12	38.23007998	15.78440838	0.320312665	2.334780771