Multi-Stage Scenario Generation by The Combined Moment Matching and Scenario Reduction Method

Uladzimir Rubasheuski and Johan Oppen
Molde University College
N-6410 Molde Norway
Uladzimir.Rubasheuski,Johan.Oppen@himolde.no
and
David L. Woodruff *
Graduate School of Management
University of California Davis
Davis CA 95616 USA
DLWoodruff@UCDavis.edu
+1-530-752-0515
June 9, 2014

Abstract

We describe an opportunity to speed up multi-stage scenario generation and reduction using a combination of two well known methods: the moment matching method (Høyland and Wallace 2001) and the method for scenario reduction to approximately minimize a metric (Heitsch and Römisch 2009). Our suggestions is to combine them rather than using them in serial by making use of a stage-wise approximation to the moment matching algorithm. Computational results show that combining the methods can bring significant benefits.

Keywords: Scenario Generation, Multi-stage stochastic programming, moment matching, scenario reduction.

*Corresponding Author
1 Introduction

Multi-stage, stochastic, optimization models receive increased interest as solver technology improves. Many solvers require that stochastic data be presented in the form of discrete realizations with attached probabilities. In the multi-stage case, they are almost always organized into a tree with the property that scenarios with the same realization up to decision stage share a node at that stage. See, e.g., [8] for more discussion of scenario trees and stochastic optimization modeling.

In this note we describe an opportunity to speed up multi-stage scenario generation and reduction using a combination of two well known methods: the moment matching (MM) method [7] and the method for scenario reduction to approximately minimize a metric (SR) [5]. While the MM is designed to generate scenarios, SR is used to reduce an existing scenario tree to a tractable size. Although Monte Carlo methods can be used to generate a scenario tree before reducing it [3, 6, 9], a sensible alternative is to use MM to generate scenarios and then SR to make the scenario tree tractable (see [2]). In all cases, the scenario tree is constructed in full before reduction. Our suggestions is to combine MM and SR rather than using them in serial. This is done making use of a stage-wise approximation to the MM algorithm. We give a brief review of the methods in the next two sections and then give the combined method in Section 4. Experimental results that confirm significant speed improvements are presented in Section 5, along with concluding remarks.

2 The Moment Matching Method for Scenario Generation

The idea of MM is to match the statistical properties of the generated scenarios with those of the observed data process. Following the notation presented by Høyland and Wallace [7], define $\Gamma$ as a set of statistical properties to be matched, and $\Gamma_{VA]\lambda_i$ as the observed value of statistical property $i$ from $\Gamma$. Then let $N$ be the number of random variables, $T$ be the number of stages and $\Theta_t$ be the number of conditional outcomes in stage $t$. Define the outcome vector $x$ of dimension $N \cdot \Theta_1 + N \cdot \Theta_1 \cdot \Theta_2 + \ldots + N \cdot \Theta_1 \cdot \Theta_2 \cdot \ldots \Theta_T$, which means that there are $\Theta_1 \cdot \Theta_2 \cdot \ldots \Theta_t$ outcomes of each variable $n = \{1, \ldots, N\}$ in stage $t = \{1, \ldots, T\}$. The probability vector $\rho$ of dimension $\Theta_1 + \Theta_1 \cdot \Theta_2 + \ldots + \Theta_1 \cdot \Theta_2 \cdot \ldots \Theta_T$. The function $f^\prime (x, \rho)$ is the mathematical expression for statistical property $i$ in $\Gamma$. Finally, let $w_i$ be the weight for statistical property $i$ in $\Gamma$. 


We then construct vectors \( x \) and \( \rho \) by solving the non-linear optimization problem:

\[
\min_{x,\rho} \sum_{i \in \Gamma} w_i \left( f^i (x, \rho) - \Gamma_{VAL_i} \right)^2
\]

(1)

\[
\rho_{t,1} = 1
\]

(2)

\[
\sum_{k = (t-1)\Theta_t + 1}^{\Theta_t} \rho_{t,k} = 1, \quad \forall t = 2, \ldots, T, j = 1, \ldots, \prod_{h=1}^{t-1} \Theta_h
\]

(3)

\[
\rho_{t,k} > 0, \quad \forall t \in T, k \in 1, \ldots, \prod_{h=1}^{t} \Theta_h
\]

(4)

In this model, \( \rho_{t,k} \) expresses probability of the outcome \( k = 1, \ldots, \prod_{h=1}^{t} \Theta_h \) in stage \( t = 1, \ldots, T \).

In principle, one can use as many moments and state dependent statistical properties as desired. To be concrete, we will refer to an example where we are matching the three first moments of demand for a set of products \( N \) and the paired correlation function as statistical properties in the objective function, and where the mean value is the only state dependent statistical property. The data set used to compute the target values of the statistical properties was taken from the Norwegian company Stokke, and includes data on the demand of a set of products for children.

The statistical properties can be expressed in the following way:

\[
f_{\text{mean}}^{n,t,k}(x, \rho) = \sum_{g = (k-1)\Theta_t + 1}^{\Theta_t} x_{n,t,g} \rho_{t,g}, \quad \forall n \in N, \ t = 2, \ldots, T, \ k = 1, \ldots, \prod_{h=1}^{t-1} \Theta_h
\]

(5)

\[
f_{\text{variance}}^{n,t,k}(x, \rho) = \frac{\Theta_t}{\Theta_t - 1} \sum_{g = (k-1)\Theta_t + 1}^{\Theta_t} \left( (x_{n,t,g} - f_{\text{mean}}^{n,t,k}(x, \rho))^2 \rho_{t,g} \right),
\]

\[
\forall n \in N, \ t = 2, \ldots, T, \ k = 1, \ldots, \prod_{h=1}^{t} \Theta_h
\]

(6)

\[
f_{\text{skewness}}^{n,t,k}(x, \rho) = \sqrt{\frac{\Theta_t (\Theta_t - 1)}{\Theta_t - 2}} \sum_{g = (k-1)\Theta_t + 1}^{\Theta_t} \left( (x_{n,t,g} - f_{\text{mean}}^{n,t,k}(x, \rho))^3 \rho_{t,g} \right) \left( \sum_{g = (k-1)\Theta_t + 1}^{\Theta_t} ((x_{n,t,g} - f_{\text{mean}}^{n,t,k}(x, \rho))^2 \rho_{t,g}) \right)^{3/2},
\]

\[
\forall n \in N, \ t = 2, \ldots, T, \ k = 1, \ldots, \prod_{h=1}^{t} \Theta_h
\]

(7)
\[ f_{n,m,t,k}^{\text{correlation}}(x, \rho) = \frac{\sum_k \Theta_t k \left( (x_{n,t,g} - f_{n,t,k}^{\text{mean}}(x, \rho)) (x_{m,t,g} - f_{m,t,k}^{\text{mean}}(x, \rho)) \rho_{t,g} \right)}{\sqrt{\sum_k \Theta_t k \left( (x_{n,t,g} - f_{n,t,k}^{\text{mean}}(x, \rho))^2 \rho_{t,g} \right) \sum_k \Theta_t k \left( (x_{m,t,g} - f_{m,t,k}^{\text{mean}}(x, \rho))^2 \rho_{t,g} \right)}}. \]

\[ \forall n,m \in \mathcal{N}, \ t = 2, ..., T, \ k = 1, ..., \prod_{h=1}^{t-1} \Theta_h \quad (8) \]

In formulas (5-8) \( x_{n,t,g} \) expresses the value of variable \( x \) for product \( n \in \mathcal{N} \) in outcome \( g = 1, ..., \prod_{h=1}^{t-1} \Theta_h \) on stage \( t \in T \).

In addition to matching the first three moments and the correlation matrix, Høyland and Wallace [7] suggest using autocorrelation in a special way. Let \( \hat{i} \) be the index for the mean, so \( \Gamma_{VAL\hat{i}}^{VAL} \) is treated as the observed value for the mean. For a particular scenario tree parent with indexes \( (n,t-1,k) \), \( \forall n \in \mathcal{N}; \ t = 1, ..., T; \ k = 1, ..., \prod_{h=1}^{t-1} \Theta_h \) the computation of \( \Gamma_{VAL\hat{i}}^{VAL} \) for its children is as follows:

\[ \alpha_n \cdot x_{n,t-1,k} + (1 - \alpha_n) \cdot \mu_n, \quad (9) \]

where \( \mu_n \) represents the grand mean demand for product \( n \in \mathcal{N} \) and \( \alpha_n \in [0, 1] \) represents the mean reversion factor for product \( n \in \mathcal{N} \). The expected demand on the first stage can be defined in two different ways. The first option is to set it equal to \( \mu_n \). The second option is to specify the current actual value of demand and use the formula above to calculate the expected demand on the first stage. Note that it is the presence of state dependent factors such as mean reversion makes stage-wise decomposition an approximation. Such factors, and mean reversion in particular, can be very important in modeling multi-stage stochastic processes.

As pointed out by Høyland and Wallace [7], in general such an optimization problem is not convex. So, one is likely to find a locally optimal solution, rather than one that is globally optimal. However, for purposes of scenario generation finding a perfectly optimal match of properties may possible or necessary. Consequently, we propose decomposition of the presented problem into sub-problems, so that a matching problem can be solved for each stage separately. The stage-by-stage algorithm can be stated as follows:

**Step 1.** Set \( t = 1 \) and \( \Gamma_{VAL\hat{i}} = \mu_n \) for each \( x_{n,t,1}, \ \forall n \in \mathcal{N} \).

**Step 2.** Set \( t = t+1 \). For each parent \( x_{n,t-1,k}, \ \forall n \in \mathcal{N}, \ k = 1, ..., \prod_{h=1}^{t-1} \Theta_h \) compute \( \Gamma_{VAL\hat{i}} \) for its children using formula (9).

**Step 3.** Solve the moment matching optimization model (1-4) with respect to the \( t \)-th stage components only. If \( t < T \), go to step 2.
Step 4. Construct scenarios $S$, where $\tilde{x}_i^t$, $\forall i \in S$ is a vector of $n$ variables related to scenario $i$ at stage $t$, and $\rho_i^t$ is the conditional probability of obtaining vector $\tilde{x}_i^t$. Then $\rho^i = \prod_{t=1}^{T} \rho_i^t$ is the probability of a scenario $i \in S$.

We have compared the performance of the original MM with the performance of the decomposed MM based on five instances of each of two configurations types. The results are shown in Table 1. The configurations are labeled $(N, T, b)$ where $b$ is the branching factor, indicating the constant number of child nodes generated from each node at stage $t = 1, ..., T - 1$. Both algorithms were coded using Python and Pyomo [4]; the nonlinear minimization problems were solved using the IPOPT [10] nonlinear solver version 3.10.2. The CPU running times were computed for an 8x Intel(R) Core(TM) i7-2600 CPU 3400 GHz processor with 15 GB RAM. The same arbitrary initial solution was used as a starting point for both algorithms.

Table 1: The Original and the Decomposed MM: Comparison of Performance based on averages taken over five replicates for each configuration.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Original MM</th>
<th>Decomposed MM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU (sec)</td>
<td>Objective</td>
</tr>
<tr>
<td>(2,4,6)</td>
<td>333.84</td>
<td>0.317</td>
</tr>
<tr>
<td>(2,5,6)</td>
<td>6942.10</td>
<td>1.97 \cdot 10^{13}</td>
</tr>
</tbody>
</table>

One can see that the original MM algorithm requires more time to find a solution and the quality of this solution is actually much worse than quality of the solution found by the decomposed MM for these instances. This is because the original problems are so large. If we use the solution obtained by the decomposed MM as a starting point for the original MM, it was improved in approximately 10% of the cases.

Nevertheless, the average objective values do not present the entire picture of algorithms’ performance. In the case of the (2,4,6) configuration, the original MM got an objective value significantly different from 0 (1.5844) only once out of five instances. That resulted in a comparably high average objective value. For the same instance the decomposed MM resulted in an objective value of factor $10^{-6}$. In the case of the (2,5,6) configuration, when the original MM was used, the objective function for one of the instances was equal to 5.21, which is comparably close to 0. However, it didn’t affect the average objective value, since the smallest objective value among other instances was of factor $10^{6}$. In the same cases, when the decomposed MM was used the highest objective value was equal to 1.44.
We also conducted some experiments with configuration (44,4,12). IPOPT could not converge on a solution for the original MM for this instance, but it was able to for the decomposed MM (it took 10739.69 CPU seconds). The reason is in the size of the optimization problem to be solved. In the case of the original MM, instances of configuration (44,4,12) are characterized by one non-linear problem with a highly non-linear objective and 84825 variables. The decomposed MM for the same instance is characterized by 157 nonlinear problems with 540 variables each.

It is clear a priori that it is computationally expedient to decompose by stages. An important conclusion from these experiments is that for practical reasons, a decomposition by stages may be required in some situations.

3 The Scenario Reduction Method

When the number of scenarios generated is large, one could be interested in a scenario tree reduction in order to be able to run the stochastic optimization model in a shorter amount of time. The scenario reduction (SR) method proposed by Heitsch and Römisch [5] and the forward construction algorithms based on this method [1] are intended to accomplish this task. The forward construction algorithm successively computes partitions of scenario set $S$ into $\lambda_t$, $t = 1, \ldots, T$ clusters of the form:

$$\Delta_t := \{ \Delta_1^t, \ldots, \Delta_{\lambda_t}^t \}, \quad \lambda_t \in \mathbb{N},$$  \hspace{1cm} (10)

where $\mathbb{N}$ is any natural number not exceeding $\Theta_t$.

The elements of $\Delta_t$ are called clusters. Now let us consider the algorithm:

**Step 1.** Define $\Delta_1 = S$ and set $t := 2$

**Step 2.** For each cluster of scenarios $\Delta_{i-1}^t$, run the scenario reduction procedure with respect to $\tilde{x}_{it}^j$, $j \in \Delta_{i-1}^t$.

**Step 3.** Obtain the mapping $\beta_t^\lambda$ from the deleted scenarios $\mathcal{J}_t^\lambda$ to the remaining scenarios $S_t^\lambda$ such that:

$$\beta_t^\lambda(i) \in \arg\min_{j \in S_t^\lambda} \| \tilde{x}_{it}^j - \tilde{x}_{it}^i \|, \quad i \in \mathcal{J}_t^\lambda$$ \hspace{1cm} (11)

**Step 4.** Define the overall mapping $\alpha_t$ from the original set of scenarios $S$ to the new set of scenarios $S$:

$$\alpha_t(i) = \begin{cases} \beta_t^\lambda(i), & i \in \mathcal{J}_t^\lambda, \text{ for some } \lambda = 1, \ldots, \lambda_{t-1} \\ i, & \text{otherwise} \end{cases}$$ \hspace{1cm} (12)
Then a new partition at $t$ is:

$$\Delta_t := \{ \alpha_t^{-1}(i) \mid i \in S_t^\lambda, \lambda = 1, \ldots, \lambda_t \}$$  \hspace{1cm} (13)

If $t < T$, set $t := t + 1$ and continue from Step 2, otherwise go to Step 5.

**Step 5.** Define new set of scenarios according to the partition set $\Delta_T$ and mappings (12). Each scenario will be the centroid of a cluster. The probability of each scenario from a new set will then be equal to the sum of probabilities of the scenarios belonging to the same cluster, plus the probability of the scenario itself.

In Step 1 a single cluster consisting from all the initial scenarios is defined. In Steps 2-4 the number of scenarios is iteratively reduced. We consider all scenarios at a given stage and refer to a component of a scenario corresponding to a given stage as a node. Thus these steps can be understood as the selection of a given number of nodes $\lambda_t$ from nodes belonging to the scenarios remaining in the previous stage. For example, consider stage 3. Suppose, that in the previous stage we have defined 2 clusters: $\Delta_1^2$ and $\Delta_2^2$. This means that only scenarios having the same node as the centroid scenarios on the second stage were selected for further consideration. Suppose also that $\lambda_3 = 2$. Then from all the third stage nodes belonging to the remaining scenarios of a given cluster $\Delta_3^2$ we form two clusters on stage 3. So, in total we will have four clusters of scenarios after stage 3. Each of these clusters will have centroid scenarios with a given node at stage 3. Only scenarios which are going through these centroid nodes are selected for further consideration on the next stage.

Once the scenarios at all of the stages have been considered, the new set of scenarios from the remaining list of scenarios in the last stage are formed. To understand the procedure for selecting the centroid nodes of clusters in Step 2 and the mapping in Steps 3-4 of the forward construction algorithm, consider a set of nodes $I_t$ with given coordinates $\bar{x}_i^t$, $i \in I$ and probabilities $\rho_i^t$. Selecting $\lambda_t$ centroid nodes from them can be considered as a P-median problem, where the distance between the nodes $i$ and $j$ is defined as a norm of the difference between coordinates of the nodes $d_{i,j} = \| \bar{x}_i^t - \bar{x}_j^t \|$ and the weight of the node is denoted by probability of this node $\rho_i^t$.

The P-median problem is known to be $\mathcal{NP}$-hard, thus Eichhorn et al. [1] proposed use of a greedy forward selection algorithm:

**Step 1.** Set $J := I_t$
Step 2. Determine an index \( l \in J \) such that
\[
l \in \arg\min_{u \in J} \sum_{k \in J \setminus \{u\}} \rho_k \min_{j \notin J \setminus \{u\}} |x_t^k - x_t^j|,
\]
and set \( J := J \setminus \{l\} \). If the cardinality of \( J \) equals to desired number of clusters go to termination step. Otherwise continue with a further index selection step.

Step 3. Find a mapping from the original set \( J \) to a reduced set \( J \) by assigning each node that is not in the new \( J \) to the closest median.

Instead of applying the forward selection heuristic one can try to solve a MIP formulation of the P-median problem, using a commercially available solver, which gives a provably bounded, nearly exact solution in comparable time for the problem we use as an example. Define two sets of variables: \( \phi_i \in \{0, 1\} \, \forall \, i \in I_t \) which indicates if scenario \( i \) was selected as centroid of a cluster on stage \( t \) (\( \phi_i = 1 \)), and \( \tau_{i,j} \in \{0, 1\} \, \forall \, i \in I_t, \, j \in I_t \) which denotes if scenario \( j \) is mapped to scenario \( i \) on stage \( t \). Then the linear model for the P-median problem is:

\[
\min \sum_{i \in I_t} \sum_{j \in I_t} \tau_{i,j} d_{i,j} \rho_t^j
\]

subject to:
\[
\sum_{i \in I_t} \phi_i = \lambda_t \quad (16)
\]
\[
\sum_{i \in I_t} \tau_{i,j} = 1, \, \forall \, j \in I_t \quad (17)
\]
\[
\sum_{j \in I_t} \tau_{i,j} \leq \phi_i (|I_t| - 1), \, \forall \, i \in I_t \quad (18)
\]
\[
\tau_{i,j} \in \{0, 1\}, \, \forall \, i \in I_t, \, j \in I_t \quad (19)
\]
\[
\phi_i \in \{0, 1\}, \, \forall \, i \in I_t \quad (20)
\]

Applying the forward construction algorithm along with any algorithm for P-median problem will allow us to reduce the original set of scenarios to the desired size.
4 Combined Moment Matching Scenario Reduction method

It is easy to see that the steps of the stage-wise MM algorithm given in Section 2 can be combined with the steps of the forward construction algorithm for the SR method in two different ways. One way is to apply the MM algorithm and then apply the forward construction. Another way is to apply the two methods simultaneously while constructing the scenario tree stage by stage. The combined algorithm can significantly reduce the time required to construct the scenario tree.

The combined moment matching forward construction algorithm for the combined moment matching scenario reduction method is as follows:

Step 1. Set $t = 1$ and $\Gamma_{VAL_i} = \mu_n$ for each $x_{n,t,1}$, $\forall n \in \mathcal{N}$

Step 2. Set $t = 2$. Define set $\mathcal{K}_{t-1} = \{1, ..., \prod_{h=1}^{t-1} \Theta_h\}$

Step 3. Compute $\Gamma_{VAL_i}$ for the children of each parent $x_{n,t-1,k}$, $\forall n \in \mathcal{N}$, $k \in \mathcal{K}_{t-1}$ using formula (9).

Step 4. Solve the moment matching optimization model (1-4) with respect to the $t$-th stage components only. Create nodes with variable vector $\tilde{x}_i = (x_{1,t,i}, ..., x_{n,t,i})^T$, $i = (k-1) \cdot \Theta_t, ..., k \cdot \Theta_t$, $\forall k \in \mathcal{K}_{t-1}$ and probabilities $p_{t,i}$.

Step 5. Create $\lambda_t$ clusters of nodes with centroid nodes $\bar{x}_j^t$, $j \in \{1, ..., \lambda_t\}$ by solving the P-median problem for the nodes at stage $t$. Set $\mathcal{K}_t = \{1, ..., \lambda_t\}$. If $t < T$, set $t := t + 1$ and continue from Step 3, otherwise go to Step 6.

Step 6. Construct scenarios $\mathcal{S}$ from obtained centroid nodes.

In other words, at each stage except the first one, we create a set of nodes using the Moment Matching method, then we reduce the number of nodes using the Scenario Reduction method. To create the nodes at the next stage we branch only from the remaining nodes using the given branching factor.

5 Experiments and Conclusions

Let us consider the process of scenario creation by two separated methods and the combined method. If the MM and SR methods are applied sequentially, then we will have to solve $1 + 1 \cdot \Theta_2 + ... + 1 \cdot \Theta_2 \cdots \Theta_{T-1}$ matching problems.
and $1 + 1 \cdot \lambda_2 + \ldots + 1 \cdot \lambda_2 \cdots \lambda_{T-1}$ P-Median problems. If the two methods are combined, the total number of matching problems and the total number of P-median problems to solve will be the same and equal to $1 + 1 \cdot \lambda_2 + \ldots + 1 \cdot \lambda_2 \cdots \lambda_{T-1}$. Hence, the larger the difference between the branching factor $\Theta_t$ and the number of clusters $\lambda_t$, the greater time savings can be achieved.

Consider the following example: Suppose, that we have 4 stages in our scenario tree, the branching factor is chosen to be $\Theta_t = 12, \forall t = \{2, 3, 4\}$ and the number of clusters is $\lambda_t = 2, \forall t = \{2, 3, 4\}$. If the two methods are applied in series we will have to solve 157 nonlinear matching problems and 7 P-median problems. If we apply the combined Moment Matching Scenario Reduction method we will have to solve only 7 matching problems and the same number of P-median problems. It means that we will avoid solving 150 nonlinear matching problems.

Examples of time savings in seconds can be seen in Table 2. The configurations are labeled as $(N, T, b)$, where $b$ is the branching factor, indicating the constant number of child nodes generated from each node at stage $t = 1, \ldots, T - 1$. We formulated the models using Pyomo and solved them using IPOPT as a nonlinear solver and we used a greedy forward selection algorithm coded in Python for the P-median problems. The MM algorithm used in serial experiments (and in the combined method experiments) was decomposed by stages. For the simultaneous solution of all stages, the computational times are dramatically higher.

Table 2: CPU Time (seconds) for the Serial and Combined Moment Matching Scenario Reduction method

<table>
<thead>
<tr>
<th>Instance Configuration</th>
<th>Serial MM SR</th>
<th>Combined MM and SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,3,6)</td>
<td>0.54 0.12</td>
<td>0.36</td>
</tr>
<tr>
<td>(2,4,6)</td>
<td>1.73 0.13</td>
<td>0.57</td>
</tr>
<tr>
<td>(2,4,12)</td>
<td>8.89 0.15</td>
<td>0.92</td>
</tr>
<tr>
<td>(2,5,12)</td>
<td>104.52 0.29</td>
<td>1.69</td>
</tr>
<tr>
<td>(2,6,12)</td>
<td>3398.81 1.43</td>
<td>2.99</td>
</tr>
</tbody>
</table>

As we can see, whenever scenario generation may constitute a large and time consuming part of solving a stochastic optimization problem, using the combined moment matching scenario reduction method can bring significant benefits.
References


