Computational Management Science (2020) 17:241–275
https://doi.org/10.1007/s10287-020-00375-4

ORIGINAL PAPER

Evaluation of scenario reduction algorithms with nested distance

Markéta Horejšová[1] · Sebastiano Vitali[1,2] · Miloš Kopa[1] · Vittorio Moriggia[2]

Received: 5 March 2020 / Accepted: 20 July 2020 / Published online: 10 August 2020

Abstract
Multistage stochastic optimization is used to solve many real-life problems where decisions are taken at multiple times. Such problems need the representation of stochastic processes, which are usually approximated by scenario trees. In this article, we implement seven scenario reduction algorithms: three based on random extraction, named *Random*, and four based on specific distance measures, named *Distance-based*. Three of the latter are well known in literature while the fourth is a new approach, namely nodal clustering. We compare all the algorithms in terms of computational cost and information cost. The computational cost is measured by the time needed for the reduction, while the information cost is measured by the nested distance between the original and the reduced tree. Moreover, we also formulate and solve a multistage stochastic portfolio selection problem to measure the distance between the optimal solutions and between the optimal objective values of the original and the reduced tree.

Keywords Nested distance · Multistage stochastic optimization · Scenario tree reduction · Nodal clustering

Mathematics Subject Classification 90C15 · 60B05 · 62P05

The research was partially supported by Czech Science Foundation Grant 18-05631S, by MIUR ex-60% 2019 sci.resp. V. Moriggia, by MIUR ex-60% 2020 sci.resp. V. Moriggia and by MIUR ex-60% 2020 sci.resp. S. Vitali.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10287-020-00375-4) contains supplementary material, which is available to authorized users.

✉ Sebastiano Vitali
vitali@karlin.mff.cuni.cz

Extended author information available on the last page of the article

1 Introduction

In many real-life problems the parameters characterizing the structure of the future may be uncertain, e.g. the values of price or the level of demand. To solve this type of problem, the literature suggests exploiting the instruments provided by stochastic optimization, see Birge and Louveaux (1997) and Powell (2014). More precisely, we distinguish between single-stage stochastic optimization when only one decision is taken, and multistage stochastic optimization, when the solution is a sequence of decisions over a given span of times, see Shapiro et al. (2009) and Dupačová et al. (2002). To deal with these types of problems, we typically assume that the uncertain parameters are random variables with known probability distribution. This distribution is usually approximated by a discrete distribution represented by a scenario tree.

In order to better approximate the initial distribution, the scenario tree may become rather large making the problem computationally intractable. Therefore, several scenario tree reduction algorithms have been proposed in the literature to suit the available computational capabilities, e.g. Dupačová et al. (2003) and Pflug and Pichler (2014). In this article, we focus on comparing representative reducing algorithms with each other. In particular, we apply three reduction approaches based on random extraction and four more advanced reduction algorithms based on distance measures. With more sophisticated scenario reduction techniques the quality of the reduction is expected to be better than with random algorithms. We assume that the quality of the reduction is given by the closeness of the two scenario trees: the original and the reduced ones. To measure the distance between two scenario trees we use the Nested Distance (ND) proposed in Pflug and Pichler (2012, 2014). The aim of this work is then to observe how good the reduction algorithms are in relation to the computational effort, i.e. the computing time that each algorithm requires.

Moreover, it is well known that multistage stochastic optimization is particularly helpful for addressing portfolio management problems, see Dupačová et al. (2002) for an overview of the topic and the recent applications of Vitali et al. (2017), Kopa and Petrová (2017), Consigli et al. (2018a), Kopa et al. (2018), Moriggia et al. (2019) and references therein. For this reason, we also formulate and solve a multistage portfolio selection problem. The aim is to measure not only the distance between the initial tree and the reduced tree and the time needed to generate the reduced tree, but also the distances between the optimal objective values and between the here-and-now solutions.

This paper is structured as follows: the notation and definition of ND adopted throughout the paper is presented in Sect. 2. In Sect. 3, we present the reduction algorithms analysed in the subsequent sections. In particular, in Sect. 3.1 we present the most naive and non deterministic algorithms and in Sect. 3.2 we introduce three other algorithms which are more advanced and deterministic. Section 5 is devoted to the numerical analysis. In particular, Sect. 5.2 analyses the results of the nested distance, while Sect. 5.3 presents a multistage portfolio model used to analyse the objective value distance and the solution distance in Sects. 5.3.1 and 5.3.2, respectively. Section 6 concludes the paper.

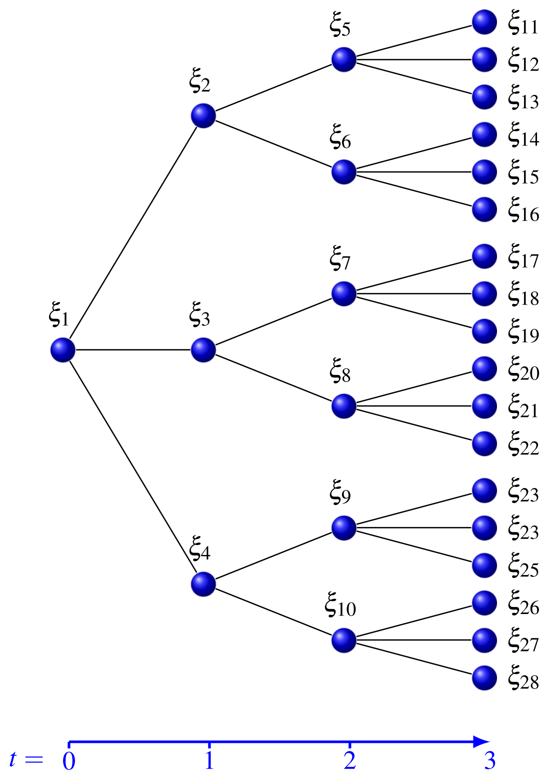

Fig. 1 Example of a four-stage stochastic tree and a 3-2-3 branching

2 Notation and nested distance

We represent a scenario tree as follows: let's suppose that the tree consists of N nodes $n = 1, \dots, N$. The root node is the node $n = 1$. For each node n, we define its stage $t(n)$, we denote the set of its direct successors (children) by $n+$, and, for each node, except for the root, we denote its direct predecessor (parent) by $n-$, the sequence of its ancestors $n-, n--, \dots, 1$ by $\mathscr{A}(n)$ and the union of the set of ancestors with the node itself by $\overline{\mathscr{A}}(n) = \mathscr{A}(n) \cup \{n\}$. A *subtree* is a set of nodes consisting of a node and its children. A particular case of subtree is a *degenerate subtree* which is a set composed of a single node. The leaves of the tree are degenerate subtrees. A tree is called *regular* if all the nodes in the same stage have the same number of children. In the case of regular trees, the number of children of each node in subsequent stages is called *branching structure* (or *bushiness*) . All nodes are divided into disjoint node sets $\mathscr{K}_t,\ t = 0, \dots, T$, collecting the nodes of each stage. In this notation, $\mathscr{K}_0 = \{1\}$ contains the root, $\mathscr{K}_T$ contains the leaves, and $\mathscr{K}_t,\ t = 1, \dots, T-1$, contain the nodes of the inner stages. By p_n we denote the absolute probability of the node n; the conditional probability of the node n (given $n-$) is denoted by q_n. Each node also carries an A-dimensional vector of values ξ_n which are the realizations of the random vector in that node.

In Fig. 1 we report an example of a regular tree spanning 4 stages, $t = 0, \dots, 3$ and having branching structure 3-2-3. The number of nodes is $N = 28$. For instance, node $n = 7$ is on the second stage, $t(7) = 2$, so it belongs to $\mathscr{K}_2$, it has children $7+ = \{17, 18, 19\}$, parent $7- = \{3\}$, ancestors $\mathscr{A}(7) = \{3, 1\}$ and $\overline{\mathscr{A}}(7) = \{7, 3, 1\}$, the corresponding vector of values of the random variable is ξ_7, and, assuming that in the whole tree the conditional probabilities are equally distributed among the children of the same node, it has conditional probability $q_7 = \frac{1}{2}$ and absolute probability $p_7 = q_3 \cdot q_7 = \frac{1}{6}$. The set of the leaves is $\mathscr{K}_3 = \{11, \dots, 28\}$.

To measure the distance between two scenario trees, we adopt the concept of the ND proposed in Pflug and Pichler (2012, 2014), and then further analysed in Pflug and Pichler (2015, 2016), Timonina (2015), Kovacevic and Pichler (2015) and Vitali (2018). In these papers, the ND is computed by backward iteration and can be interpreted as the optimal transportation cost for moving one scenario tree onto the other scenario tree considering the conditional probabilities of the nodes and, of course, the realization of the random variable in each node.

Given two scenario trees, we start computing the distance between the scenarios of the first tree and the scenarios of the second tree. We denote by single quote ($'$) all the elements of the second tree. For each pair of leaf nodes $i \in \mathscr{K}_T$ and $j \in \mathscr{K}'_T$ we compute the distance

$$\mathbb{d}_T(i, j) := d\left((\xi_{k_i}), (\xi'_{k_j})\right), \quad k_i \in \overline{\mathscr{A}}(i), k_j \in \overline{\mathscr{A}}(j) \tag{1}$$

where d is the classic ℓ^1 distance.

Then we move backward for $t = T - 1$ down to $t = 0$ and, for all combinations of $k \in \mathscr{K}_t$ and $l \in \mathscr{K}_t{}'$, we solve the following linear problem

$$\begin{aligned}
\mathbb{d}_t(k, l) := \min_{\pi} & \sum_{m \in k+, n \in l+} \pi(m, n|k, l) \cdot \mathbb{d}_{t+1}(m, n) \\
\text{s.t.} & \sum_{n \in l+} \pi(m, n|k, l) = q_m \quad \forall m \in k+, \\
& \sum_{m \in k+} \pi(m, n|k, l) = q'_n \quad \forall n \in l+, \\
& \pi(m, n|k, l) \geq 0.
\end{aligned}$$

The ND between the trees is the distance between the subtrees at level 0, i.e. at their roots,

$$\mathbb{d}(\mathbb{P}, \mathbb{P}') = \mathbb{d}_0(1, 1). \tag{2}$$

3 Scenario reduction algorithms

As mentioned above, in many situations we have a stochastic process represented by a huge scenario tree, and we would like to reduce (approximate) the tree to a smaller one, in order to be able to solve a stochastic optimization problem based

Table 1 Scenario reduction algorithms

Random	Distance-based
Nodal extraction	Single scenario reduction
Improved nodal extraction	Nodal clustering
Scenario extraction	Subtrees merging
	Single node reduction

on the scenario tree itself. Many scenario reduction algorithms are proposed in the literature. We have selected a set of representative algorithms to highlight their quality in terms of computational costs and information cost, i.e. computation time and loss in the information carried by the tree. In particular, we divide the scenario reduction algorithms into two groups and we name them *Random* and *Distance-based*, see Table 1. The random algorithms are based on a random extraction of some nodes from a huge tree to generate a smaller one. These algorithms are easy to implement and fast to run. However, the consequent reduced tree is uncertain because it is based on a random sample and, therefore, also the quality of the reduction could vary. The distance-based algorithms are representatives of different classes of reduction approaches according to the distance definition used to merge the closest elements or to eliminate the furthest: distance between scenarios, distance between nodes and distance between subtrees.

3.1 Random algorithms

Random algorithms create a reduced tree by randomly extracting the elements (nodes and/or scenarios) of the original tree. Indeed, these algorithms neither compute any distance measure nor solve any nested optimization. They are easy to implement and typically very fast as will be shown in the numerical section. All these algorithms, being based on a random extraction, do not produce a unique reduced tree, but they identify a different reduced tree every time they are run.

Nodal extraction

The first algorithm we propose takes as input an original tree and the desired structure of the reduced tree. This means that the number of nodes for each stage and the branching structure is given. Then, at each stage, we randomly select from the nodes of that stage of the original tree a set of nodes that will become the nodes of the same stage on the reduced tree. Using this approach, we can completely alter the parent-child relationship of the original tree. Therefore, it can be used only for those scenario trees that originally have an intra-stage independence. Considering the example depicted in Fig. 1 and assuming we want a reduced tree with branching 2-1-2, we could obtain the reduced tree proposed in Fig. 2, for instance. In this case, for the second stage we extracted the nodes {2, 4}, for the third the nodes {9, 7} and for the last stage the nodes {13, 27, 21, 19}. Clearly the parent-children relationships of the original tree are not conserved.

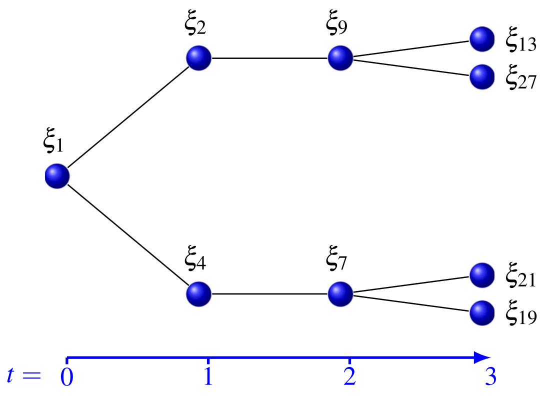

Fig. 2 Example of reduced tree obtained with nodal extraction algorithm

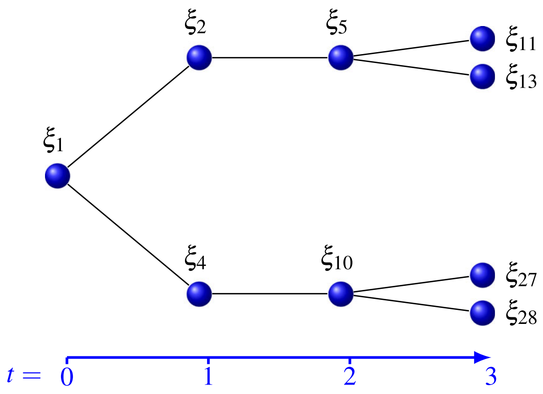

Fig. 3 Example of reduced tree obtained with improved nodal extraction algorithm

Improved nodal extraction

To address the issue of the previous approach, i.e. the loss of the parent-children relationships, nodes can be randomly sampled only from the children of the nodes already selected from the previous stage. Then, in each stage and for the children of a particular node, we observe the children of that node in the original tree and we sample from them the children of the node in the reduced tree. Therefore, this approach proceeds forward and preserves the original parent-children relationships. Considering the example shown in Fig. 1 and assuming we want a reduced tree with branching 2-1-2, we could, for example, obtain the reduced tree proposed in Fig. 3. In this case, we extract for the second stage the nodes $\{2, 4\}$, for the third stage as children of $\{2\}$ we extract $\{5\}$ and as children of $\{4\}$ we extract $\{10\}$, for the fourth stage as children of $\{5\}$ we extract $\{11, 13\}$ and as children of $\{10\}$ we extract $\{27, 28\}$. In this way, we keep the parent-children relationships of the original tree.

Scenario extraction

The scenario extraction approach requires as input an original tree and the number of scenarios of the reduced tree. Then, the algorithm consists of two steps. In the first

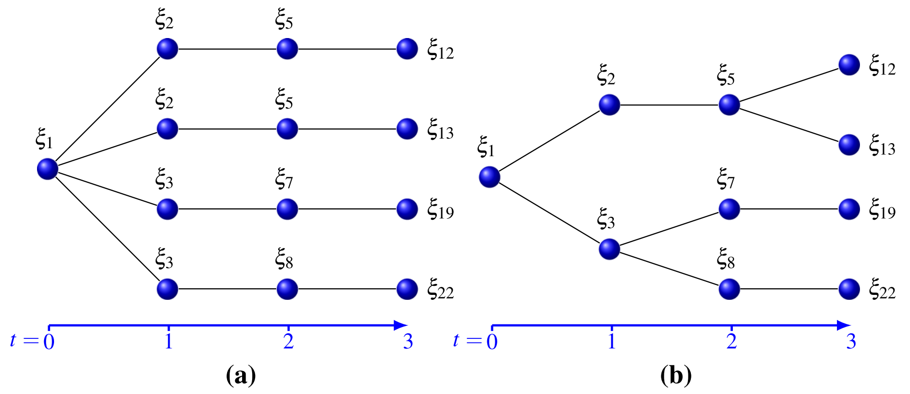

Fig. 4 Example of reduced tree obtained with scenario extraction algorithm

step, the algorithm randomly samples the given number of scenarios from the original tree and generates the reduced tree as a fan of such scenarios. In the second step, the algorithm merges the overlapping nodes and sets the probabilities of the leaves so that the former ratios are preserved. For example, if we select three leaf nodes with original probabilities 0.2, 0.1 and 0.3, in the reduced tree the probabilities will be 1/3, 1/6 and 1/2. The probabilities of nodes in previous stages are computed recursively as the sum of the probabilities of their children.

Using this approach, it can happen that many nodes may have only one child. Therefore, the reduced tree could have a total number of nodes that is larger than the trees generated with the other two random algorithms.

Considering the example depicted in Fig. 1 and assuming we want a reduced tree with 4 scenarios as in the previous examples, we could, for instance, in the first step extract from the original scenario tree the scenarios having as leaves the nodes {12, 13, 19, 22} and compose the fan of scenarios in Fig. 4a and then, in the second step, we obtain the final reduced scenario tree displayed in Fig. 4b. In this way, we keep the parent-children relationships of the original tree. However, we also notice that the size of the scenario tree increases compared to the other two random algorithms since the total number of nodes is 10 while with the other two approaches it was 9.

3.2 Distance-based algorithms

Since random algorithms do not use the information about the values of the nodes, it is natural to suspect that the reduced trees obtained by these methods might not be very close to the original tree. For this reason, we would like to think of a way to be able to take into account both the information of the values of the nodes and the information about the tree structure. In this section, we introduce four different approaches that tackle this issue.

Single scenario reduction

The single scenario reduction is proposed and described in detail in Dupačová et al. (2003). As the name suggests, it reduces the original tree scenario by scenario, taking as input an original scenario tree and the number of scenarios desired for the reduced scenario. At each iteration, this method eliminates one scenario which has a small distance from another scenario and also carries a small probability. To achieve this purpose, the algorithm performs the following three steps. First, the algorithm measures the distances between all the scenarios and constructs the matrix composed of the elements

$$d_{i,j} = d((\xi_{i_0}, \ldots, \xi_{i_T}), (\xi_{j_0}, \ldots, \xi_{j_T})), \quad i \neq j,$$

where $(i_0, \ldots, i_T)$ and $(j_0, \ldots, j_T)$ are the paths of the scenario i and j, respectively. We use distance ℓ^1, although as an alternative one could use any well-defined distance on $R^{(T+1)\times A}$. Second, for each scenario i the algorithm defines

$$D(i) = p_i \cdot \min_{j \neq i} d_{i,j}, \tag{3}$$

where p_i are the scenario probabilities, and the algorithm eliminates the scenario for which $D(i)$ is the smallest. If $\arg\min D(i)$ contains more than one element, we can either randomly choose the scenario that must be eliminated (which would lead to a non-deterministic solution) or we could simply eliminate the first one (which would lead to a deterministic solution).

Finally, in the third step, the algorithm identifies the scenario which is the closest to the eliminated one and increases its probability by the probability of the eliminated scenario, e.g. if the eliminated scenario is the scenario i^*, we find

$$j^* = \arg\min_{j \neq i^*} d_{i^*,j}$$

and set $p_{j^*} = p_{j^*} + p_{i^*}$. We repeat this procedure until we have the desired number of scenarios.

The disadvantage of this algorithm is that we cannot specify a particular parent-children structure for the reduced tree but only a given number of scenarios. Moreover, it uses only absolute probabilities. Finally, this method is computationally much more demanding than the previous random algorithms, because all random algorithms have linear complexity while this method has quadratic complexity with respect to the number of scenarios (or nodes).

Nodal clustering

In this part, we introduce a new algorithm based on clustering. Clustering is a way of grouping n points into $s < n$ groups, which are named clusters. There are many possible ways of clustering described in the literature, see Kaufman and Rousseeuw (2009) for a review. In this work we consider the following procedure: we start with each point being a cluster itself. Then we repeatedly join the two nearest clusters into one until we get the target number of clusters. The distance between clusters is defined as the maximum distance between the points of one cluster and the points of the other, i.e. if we have two clusters, say A and B, A consisting of points $\{a_1, \ldots, a_k\}$, B of $\{b_1, \ldots, b_l\}$, where k and l are positive integers, then the distance between A and B is

$$d(A, B) = \max\{d(a_i, b_j) : i \in \{1, \ldots, k\}, j \in \{1, \ldots, l\}\}, \tag{4}$$

where $d(a_i, b_j)$ is some distance between points a_i and b_j, e.g. the distance ℓ^1 or the distance ℓ^2.

Alternatively, we could represent each cluster by its centroid and compute the cluster distances as the distances between their centroids. Instead of centroids, we could use the points which have the smallest average distance to the other points of the cluster, or the smallest maximum distance to the other points of the cluster, etc.

Given as input the desired reduced tree structure, we proceed as described below.

Algorithm. We choose the final (regular) structure of the reduced tree and set $\mathscr{K}_0^* := \mathscr{K}_0$. Then, for $t = 0$ to $t = T - 1$ and for each node $n \in \mathscr{K}_t^*$, we cluster its children into the predetermined number of groups. As the representative of each group we take the group's median, i.e. a new artificial node that corresponds to a vector whose components are the medians of the components of the vectors of the nodes belonging to that group. We store the label of the new node in $\mathscr{K}_{t+1}^*$. Its probability is equal to the sum of the probabilities of all the members of the group and, since it is a new node, its children are all children of all the members of the group. The scheme of the process is given in Algorithm 1 (in Electronic Supplementary Material).

The reduced tree is represented by the nodes in $\mathscr{K}_t^*$, $t = 0, \ldots, T$. It has the required structure, but its scenarios do not need to be equiprobable. This is due to the fact that the clusters usually do not have the same number of points and the probability is taken as the sum of the probabilities of the points in the cluster.

The idea of apply clustering techniques to reduce the size of a scenario tree is not new in the literature. Many works propose clustering the scenarios directly, e.g. Growe-Kuska et al. (2003), Heitsch and Römisch (2003), Mandelli et al. (2013), Heitsch and Römisch (2009), Crainic et al. (2014), Beraldi and Bruni (2014), while others suggest merging the nodes by applying a nested optimization at each iteration, see Pflug and Pichler (2014) that proposes the subtree merging explained in the next paragraph and Chen and Yan (2018) that adopts the same methodology. However, the approach described in Algorithm 1 differs from all these reduction algorithms because it clusters nodes and not scenarios and because it does not need any nested optimization. Therefore, we name this reduction algorithm *nodal clustering*.

Subtree merging

As already mentioned, see Pflug and Pichler (2014) and Chen and Yan (2018), it is also possible to merge nodes considering their corresponding subtrees. These methods require a nested optimization to compute the distance between subtrees. As a representative algorithm for this class of techniques, we adopt the algorithm presented in Pflug and Pichler (2014) that uses the ND itself to measure the distance between subtrees.

The procedure starts computing the NDs between all feasible pairs of subtrees, i.e. having roots (say i_0 and j_0) at the same stage, $t(i_0) = t(j_0)$, and with the same parent,

$i_0- = j_0-$. We merge the pair with the lowest ND using the algorithm described in the following paragraph. Note that the closest pair could have roots at any stage and so the algorithm proceeds neither forwards nor backwards.

Suppose we want to merge two subtrees with distributions $\mathbb{P}_1$ and $\mathbb{P}_2$ and roots i_0 and j_0. If the nodes i_0 and j_0 are leaves, we take their mean value as the new node. If they are not leaves, we use the optimal transport plan between $\mathbb{P}_1$ and $\mathbb{P}_2$ which we already have from the computation of the ND. We start by setting the value of the new root as the mean value of the two old roots. For the successors, we sort the optimal transport plan into descending order, i.e., we get

$$\pi_{c_1} \geq \pi_{c_2} \geq \cdots,$$

where $c_1, c_2, \ldots$ represent all possible couples of successors. For instance, if i_0 and j_0 have three children each, $i_1, i_2, i_3 \in i_0+$ and $j_1, j_2, j_3 \in j_0+$, we get nine couples $(i_1, j_1), (i_1, j_2), \ldots, (i_3, j_3)$, which we reorder so that the inequalities above hold and we denote them by $c_1, c_2, \ldots$ for simplicity, i.e. if π_{i_2,j_1} is the biggest then $c_1 = (i_2, j_1)$. Then, we take the smallest m such that

$$\sum_{k=1}^{m} \pi_{c_k} \geq p,$$

where p is a chosen reducing parameter, $p \in (0, 1]$. The smaller the p, the greater the reduction. We obtain m pairs of subtrees with roots $c_1, \ldots, c_m$. We set their probabilities as

$$p_k = \frac{\pi_{c_k}}{\sum_{l=1}^{m} \pi_{c_l}}, \quad k = 1, \ldots, m,$$

so that $\sum_{k=1}^{m} p_k = 1$ holds. Now we proceed recursively by merging the pairs $c_1, \ldots, c_m$. The algorithm stops when the resulting number of scenarios reaches the required amount. Otherwise, we recalculate the NDs and merge the closest pair.

The computational effort of this algorithm is particularly intense, see numerical experiments in Sect. 5.1, since it does not control the number of nodes in each stage which could also increase compared with the original scenario tree. For this reason, we introduce two new versions of the subtree merging algorithm that change the algorithm and make it computationally more tractable.

Then, the three versions of the subtree merging algorithm considered are:

– *SubtreeM*: the original algorithm proposed in Pflug and Pichler (2014), takes as input the original scenario tree and the number of scenarios of the reduced tree, but it does not control the number of nodes in each stage;
– *nonDeg-SubtreeM*: the first new version of the algorithm that we propose. It takes as input the original scenario tree and the number of scenarios of the reduced tree, but it does not consider degenerate subtrees (subtrees consisting of one node only) when looking for the couple of subtrees to merge; in other words, at each iteration, the roots i_0 and j_0 of the subtrees to be merged could be at any stage excluding the last, $t(i_0) = t(j_0) \neq T$;

- *Adj-SubtreeM*: the second new version of the algorithm that we propose. The Adjusted version of the algorithm takes as input the original scenario tree and a specific branching structure for the reduced scenario tree (not only the number of scenarios). This algorithm differs from the original version because it proceeds forward, i.e. it starts by considering as possible subtrees to be merged only those with roots i_0 and j_0 at the first stage, $t(i_0) = t(j_0) = 1$, while the original algorithm considered $t(i_0) = t(j_0), t(i_0) \in \{1, \dots, T\}$. Then, the algorithm merges the closest pair of these subtrees at each iteration until the number of nodes required for the first stage is reached. After the first stage, the algorithm proceeds to the second stage by considering the subtrees having roots i_0 and j_0 at the second stage and common parent, i.e. $i_0- = j_0-$ and $t(i_0) = t(j_0) = 2$. The procedure continues forward in the same way until the desired branching is fulfilled. The algorithm stops at the last stage merging the leaves. When the algorithm identifies two subtrees to be merged, it follows the general procedure of the original algorithm.

A detailed analysis of the performance of the three versions of the subtree merging algorithm is reported in Sect. 5.1 where we also discuss the choice of the parameter p which is crucial for the performance of this algorithm.

Single node reduction

One of the most commonly used reduction algorithms is the algorithm proposed in Heitsch and Römisch (2009). The idea is to measure distances between nodes with the same parent, find the pairs which are close enough, reduce them, and repeat until the reduction stopping criterion is fulfilled. We modified the algorithm slightly so that it is as follows:

1. for each pair (i, j) of nodes with the same parent compute

$$\varepsilon_{i,j} = p_i \cdot \|\xi_i - \xi_j\| + \frac{2p_i p_j}{p_i + p_j},$$

   where p_i is the absolute probability of the node i and ξ_i the nodal value (note that the nodes (i, j) could be at any stage),
2. choose the pair (i, j) with the smallest $\varepsilon_{i,j}$,
3. set the node i as the parent of the children of the node j, set $p_i = p_i + p_j$ and remove the node j,
4. repeat until the desired number of scenarios is reached.

4 Portfolio selection problem, objective and solution distances

After measuring the ND between the original tree and the reduced tree, we want to understand if the reduction algorithms induce not only close scenario trees, but also close multistage stochastic problems. Therefore, for a given multistage stochastic model, we want to observe if more or less close scenario trees also induce more or less close solutions. To evaluate the distance between two stochastic problems using different scenarios trees Vitali (2018) already analyses not only the ND, but also the optimal solutions both in terms of optimal objective function values and in terms of first-stage solutions. Therefore, we also formulate and solve a multistage stochastic problem to compare the objective function values and the first-stage solution that we obtain by using the original scenario tree and the reduced scenario trees. Given the nature of our scenario trees in the numerical section - weekly returns of a set of assets - and considering that one of the main fields of application of multistage stochastic optimization is portfolio selection and management, see e.g. Dupačová et al. (2002), Kilianová and Pflug (2009), Vitali et al. (2017), Kopa and Petrová (2017), Consigli et al. (2018a, b), Kopa et al. (2018), Moriggia et al. (2019), Rusý and Kopa (2018), we formulate a multistage portfolio selection problem.

Given a set $i = 1, \dots, A$ of available assets and the set of nodes $n = 1, \dots, N$ each with probability p_n, we define the following variables:

$x_{n,i}$	is the amount of money invested in the asset i at the node n,
$\mathbf{W}_t$	is a random variable representing the wealth distribution at stage t,
W_n	are the realizations of the random variable $\mathbf{W}_t$ with $n \in \mathcal{K}_t$.

Then, considering the weekly returns $\rho_{i,n}$ realized between node n and its ancestor $n-$ and considering a diversification coefficient θ and a turnover coefficient λ, the problem is formulated as follows:

$$\max \quad f(\mathbf{W}_T) \tag{5}$$

$$\sum_{i=1}^{A} x_{n,i} = W_0, \tag{6}$$

$$\sum_{i=1}^{A} (1+\rho_{i,n}) x_{n-,i} - \sum_{i=1}^{A} x_{n,i} = 0 \quad \forall n \in \mathcal{K}_t, \quad t = 1, \dots, T-1, \tag{7}$$

$$\sum_{i=1}^{A} (1+\rho_{i,n}) x_{n-,i} = W_n \quad \forall n \in \mathcal{K}_T, \tag{8}$$

$$\theta \cdot W_0 - x_{n,i} \geq 0, \quad \forall i, \quad \forall n \in \mathcal{K}_0, \tag{9}$$

$$\theta \cdot \sum_{i=1}^{A} (1+\rho_{i,n}) x_{n-,i} - x_{n,i} \geq 0, \quad \forall i, \forall n \in \mathcal{K}_t, \quad t = 1, \dots, T-1, \tag{10}$$

$$|x_{n,i} - x_{n-,i}| \leq \lambda x_{n-,i} \quad \forall i, \ \forall n \in \mathcal{K}_t, \quad t = 1, \dots, T, \tag{11}$$

$$W_n \geq 0, \quad \forall n \in \mathcal{K}_T, \tag{12}$$

$$x_{n,i} \geq 0, \quad \forall i, \quad \forall n \in \mathcal{K}_t, \quad t = 0, \dots, T. \tag{13}$$

As objective function $f(\mathbf{W}_T)$ in (5) we implement both the mean value of the final wealth and the Average Value-at-Risk of the final wealth. Then, in the first case we have $f(\mathbf{W}_T) = \mathbb{E}[\mathbf{W}_T]$ and the objective function (5) takes the following form:

$$\max \quad \sum_{n} p_n \cdot W_n, n \in \mathcal{K}_T, \tag{14}$$

while in the second case we have $f(\mathbf{W}_T) = AV@R[\mathbf{W}_T]$ and, adopting the formulation proposed in Rockafellar and Uryasev (2000, 2002), we substitute the objective function (5) with the following:

$$\max \quad a - \frac{1}{\alpha}\sum_n (z_n \cdot p_n) \tag{15}$$

$$-a + W_n + z_n \geq 0, \qquad n \in \mathscr{K}_T \tag{16}$$

$$z_n \geq 0, \quad n \in \mathscr{K}_T \tag{17}$$

where a is an auxiliary variable, z_n is a slack variable and α is a given confidence level. Thus, we propose two different models and we observe the empirical evidence obtained by maximizing the final expected wealth and by optimizing a coherent risk measure of the final wealth. Equation (6) represents the first-stage budget constraint. Equation (7) expresses the financial growth of the portfolio due to the asset returns. Equation (8) calculates the wealth achieved in each node. Constraints (9) and (10) avoid an investment higher than θ for a single asset. Constraint (11) imposes a maximum turnover, i.e. it is not possible to change each position by more than $\lambda \cdot 100\%$. Clearly constraint (11) is then implemented with two inequality constraints to keep the model fully linear.

Once we have solved the above model for two scenario trees, the distance between the optimal objective values, namely *objective distance*, is measured using the ℓ^1 norm, i.e. for two optimal objective values obj_1^* and obj_2^* the distance is

$$d(\text{obj}_1^*, \text{obj}_2^*) = \left|\text{obj}_1^* - \text{obj}_2^*\right| \tag{18}$$

while the distance between the optimal solutions, namely *solution distance*, is intended as the ℓ^1 distance between the first stage decisions. That is, if $\mathbf{x}_1^1$ and $\mathbf{x}_1^2$ are two here-and-now optimal solutions, the distance between them is then

$$d(\mathbf{x}_1^1, \mathbf{x}_1^2) = \frac{\sum_{i=1}^{A}\left|x_{1,i}^1 - x_{1,i}^2\right|}{2}, \tag{19}$$

where A is the dimension of the vectors $\mathbf{x}_1^1$ and $\mathbf{x}_1^2$, i.e. the number of assets, and then $x_{1,i}^1$ and $x_{1,i}^2$, respectively, are the optimal amount of the initial wealth W_0 that should be invested in the asset i at time $t = 0$. By dividing by 2, we ensure that the solution distance is in the interval $[0, W_0]$.

5 Numerical study

We apply the algorithms described in the previous section to different original trees. All the original trees span over 4 stages and are regular trees. However, in many problems, in particular in Finance, for a given number of scenarios the scenario tree is built with more branching in the first stages and a lower number of children in the last stages to better describe the information in the short period. Therefore, we consider original trees with two different branching structures: 20-5-5-2, i.e. 1000 scenarios, or 25-10-10-4, i.e. 10,000 scenarios. Our aim is to obtain a reduced scenario tree having 100 scenarios and, when the algorithm allows, to use the following branching structure: 5-5-2-2. Moreover, for each size of the original scenario tree, in each node we consider different dimensions of the random variable. Indeed, we represent either 1-, 5-, 10- or 20-dimension random variable, i.e. we construct the evolution of either 1, 5, 10 or 20 assets.

To generate the scenario trees, we observe the weekly returns of a set of 20 assets over the period from the beginning of 2015 till the end of 2018. The list of the assets and the historical mean and standard deviation of the weekly returns are reported in Table S1. Then, in the 1-dimension case we consider the asset #1, UBI.PA, in the 5-dimension case we consider the assets #1-#5, in the 10-dimension case we consider the assets #1-#10, and in the 20-dimension case we consider all the assets.

To generate the original scenario tree, we consider two scenario generation approaches. The first approach is the Monte Carlo (MC) sampling method: observing also the historical correlation between the assets and considering the required structure, we sample the vector of the weekly returns of each node of the tree from a multivariate normal distribution having the parameters described in Table S1. Therefore, by construction, all the generated scenarios are equiprobable and there is no dependency between subsequent stages. The second approach is the Historical Extraction (HE) proposed in Kopa et al. (2018) that generates the scenario tree sampling from the historical series of the assets. In particular, given a time step and a given branching structure, it samples historical observations trying to reproduce possible financial cycles that consider both good and bad paths. Therefore, this approach not only maintains the dependency between the assets because the sampled observations carry the correlation between the assets naturally without any further estimation, but it also induces an intra-stage dependency due to the financial cycle that the algorithm reproduces, and generates equiprobable scenarios because the sampled historical observations are equiprobable.

Combining the two different sizes of the original tree (10,000 and 1000) with the dimensions of the vector in each node (1-, 5-, 10-, 20-dimension) and with the type of scenario generation approach (MC and HE), we obtain 16 original trees to which we apply the reduction methods described in the previous section.

Since the reduced tree cannot be determined uniquely, for the random algorithms we generate for each case 100 reduced scenario trees.

The numerical analysis starts with Sect. 5.1 where we compare the three versions of the subtree merging algorithm and the choice of the parameter p is discussed.

Then, the results of the ND between the original scenario trees and the reduced scenario trees, and the computing time needed to apply each reduction algorithm are presented and discussed in Sect. 5.2.

Finally, in Sect. 5.3 we implement and solve the multistage portfolio selection problem to compare the ND with the distances between the optimal objective values and the distances between the first-stage solutions.

Table 2 Comparison of the performance of the three versions of the subtree merging algorithm for different values of p considering the original tree generated with the HE method. We report the ND between the original tree and the reduced tree and the time needed to perform the reduction. We highlight in bold the smallest ND obtained for a given version of the algorithm

Value of p	Adj-SubtreeM		nonDeg-SubtreeM		SubtreeM	
	Time	ND	Time	ND	Time	ND
0.1	384	2.081	3732	1.473	10794	1.274
0.2	384	2.081	3742	1.473	10783	1.274
0.3	407	1.883	3741	1.475	10785	1.273
0.4	406	1.883	3745	1.475	10828	1.273
0.5	439	1.950	3812	**1.448**	10848	**1.265**
0.6	479	**1.782**	4015	1.456	11431	1.289
0.7	529	2.051	4116	1.520	11545	1.323
0.8	612	2.010	4120	1.486	11552	1.303
0.9	680	2.002	4307	1.497	11769	1.316
1.0	680	2.002	4313	1.497	11758	1.316

Throughout the study, we use the ND of order one and the ℓ^1 norm to measure distances between points and scenarios, i.e.

$$d((\xi_0,\dots,\xi_T),(\xi'_0,\dots,\xi'_T))=\sum_{t=0}^{T}\sum_{i=1}^{A}\left|\xi_{t,i}-\xi'_{t,i}\right|,$$

where $T=4$ is the number of stages and A is the dimension of the nodes, i.e. the number of assets considered.

All computation is performed in MATLAB on a computer with Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz processor and 8 GB memory. The multistage portfolio selection problem is implemented and solved in GAMS 23.2.1 using Cplex 12.1.0. The time is measured in seconds.

5.1 Analysis of the subtree merging algorithm

The aim of this section is to compare the three versions of the subtree merging algorithm and the choice of the parameter p to find which version and which value of p is more suitable for adoption in the subsequent analysis. As representative cases, we consider the original trees with 10-dimension nodes and 1000 scenarios created both with the MC method and with the HE method. We apply the three versions of the subtree merging algorithm to obtain a reduced tree with 100 scenarios and, for the Adj-SubtreeM, specifying the branching 5-5-2-2. We propose results fixing the parameter p to values 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 and considering the original tree generated with the HE method in Table 2 and with the MC method in Table 3.

The results in Tables 2 and 3 confirm various observations:

Table 3 Comparison of the performance of the three versions of the subtree merging algorithm for different values of p considering the original tree generated with the MC method. We report the ND between the original tree and the reduced tree and the time needed to perform the reduction. We highlight in bold the smaller ND obtained for a given version of the algorithm

Value of p	Adj-SubtreeM		nonDeg-SubtreeM		SubtreeM	
	Time	ND	Time	ND	Time	ND
0.1	309	3.261	4010	2.073	12847	1.852
0.2	310	3.261	4187	2.073	12850	1.852
0.3	322	2.779	4189	2.073	12850	1.852
0.4	322	2.779	4193	2.073	12855	1.852
0.5	340	2.672	4211	**2.069**	12841	1.853
0.6	372	2.580	4488	2.499	13318	1.865
0.7	501	2.574	4530	2.329	13356	**1.837**
0.8	629	2.514	4540	2.528	13365	1.850
0.9	679	**2.439**	4663	2.217	13492	1.850
1.0	682	**2.439**	4668	2.217	13454	1.850

– the Adj-SubtreeM version is on average 10 times faster than the nonDeg-SubtreeM which is on average 3 times faster than the SubtreeM.
– the ND obtained with the SubtreeM version is smaller than the one obtained with the nonDeg-SubtreeM which is smaller than the one obtained with the Adj-SubtreeM. Therefore, there is a clear trade-off between the ND and the computing time of the three versions of the subtree merging algorithm.
– for an increased value of the parameter p all versions show an increased computing time but not a decreased ND. Indeed, for almost all the cases, the minimum ND is touched for a value of p in the range $[0.5, 0.7]$. Therefore, focusing jointly on minimizing the ND and the computation time, we consider $p = 0.5$ in the subsequent analysis.

The comparison just shown could not be performed having the original scenario tree with 10,000 scenarios because both the nonDeg-SubtreeM and SubtreeM versions reached the limit time of 1 day for all values of p. Therefore, in the subsequent analysis, to be able to compare results for both 1000 and 10,000 cases, we adopt only the version Adj-SubtreeM.

5.2 Comparison of the ND and computing times

We compare the time required to run each of the seven reduction algorithms described in Sect. 3 and the NDs between the reduced trees generated by the algorithms and the original scenario trees.

First, we analyse the results of the cases that consider the original scenario trees generated with the MC approach, cf. Fig. 5 and Table 4. For the reduction of the tree with 1000 scenarios, we notice that the single node reduction algorithm is the best in terms of the ND for the 1- and 10-dimension cases, while for the 5- and 20-dimension

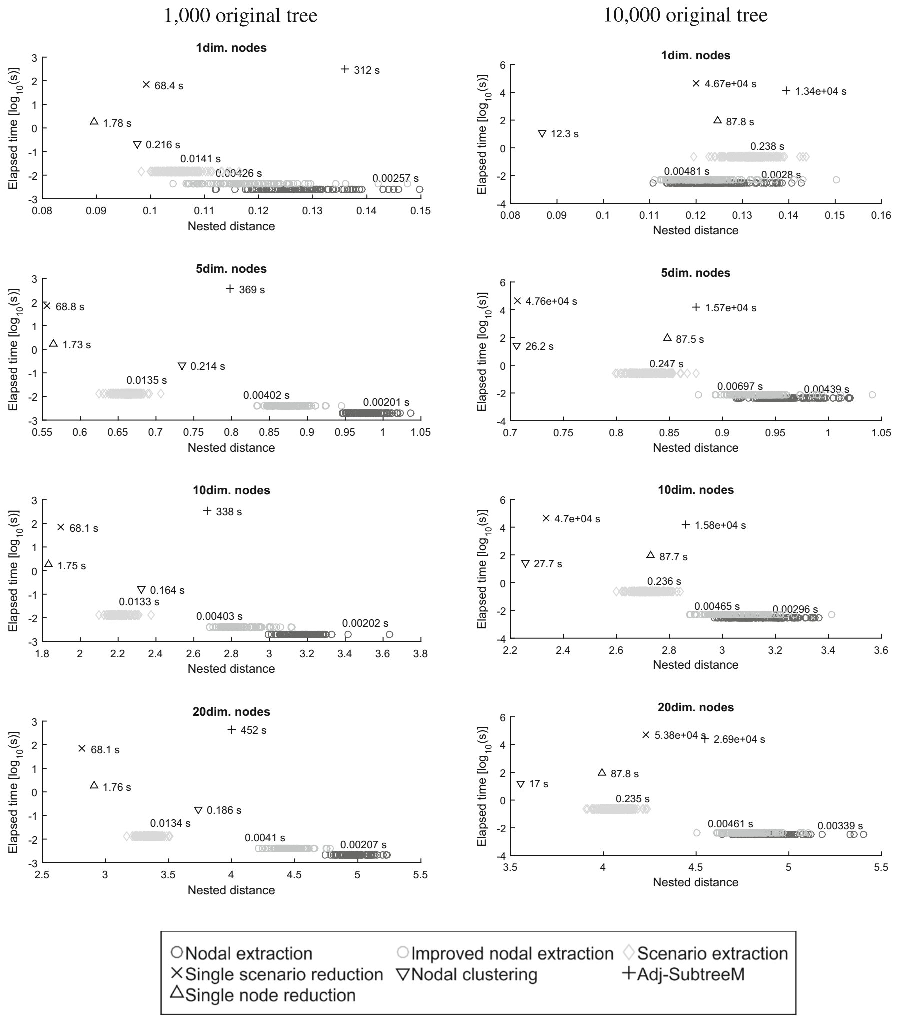

Fig. 5 The ND and the computation time for all the reduction algorithms considering the original trees generated with the MC approach. On the left, the result starting from the original tree with 1000 scenarios, on the right, the result starting from the original tree with 10,000 scenarios. Each row corresponds to a different dimension for the node of the trees

cases the single scenario reduction produces a reduced scenario tree closer to the original scenario tree (with the ND equal to 0.556 and 2.815, respectively). In terms of computing time, the single node reduction approach proves to be very efficient with an elapsed time of around 1.7 seconds. The nodal clustering algorithm is the fastest of the distance-based algorithms but the final ND is always greater than the one obtained with the single node reduction approach. The Adj-SubtreeM algorithm is computationally the most expansive, although it performs badly in terms of the ND.

Considering the reduction of the tree with 10,000 scenarios, the nodal clustering appears to be the most efficient for all the dimensions. In the 5-dimension case, the single scenario extraction gives the same result as the nodal clustering (the ND equal to 0.706) but it requires much more time to run (47,600 vs 26 seconds). Moreover, we notice that for the single node reduction algorithm and, even more for the nodal clustering, the computing times remain relatively low (between 12 and 87 seconds), while for the single scenario reduction and for the Adj-SubtreeM the computing times increase significantly (between 13,000 and 53,000 seconds). However, unlike the 1000 case, the single node reduction algorithm now performs worse than the nodal clustering for all dimensions.

The random algorithms generate a reduced tree in less than 1 second and, especially in the case of the 10,000 scenarios, they have a performance in terms of the ND similar to some of the distance-based algorithms. As expected, of the random algorithms, the scenario extraction performs better than the others showing, on average, a smaller ND, being more stable with a smaller standard deviation (cf. SD in Table 4), and requiring a computing time of the same magnitude as the other random algorithms.

In Figure S1 and in Table S2, we show the results obtained by reducing the original tree generated with the HE approach.

In general, we observe similar results to the MC case. When we reduce the original scenario tree with 1000 scenarios, the best performing algorithm is the single node reduction and the single scenario reduction is the second best in the 5-, 10- and 20-dimension cases. The nodal clustering has an even worse result than the scenario extraction algorithm apart from the 1-dimension case. The Adj-SubtreeM algorithm produces the highest ND of the distance-based algorithms.

In the case of the 10,000 scenario tree, the nodal clustering is the best only in the 1-dimension case, while in all the other dimension cases the best performing algorithm is the single scenario reduction. However, the single scenario reduction, the nodal clustering and the single node reduction appear to be relatively close to each other in terms of the ND, while in terms of required computing time the single node reduction is the fastest. Considering the random algorithms, again the scenario extraction performs better than the others both in terms of the ND on average, and in terms of stability of the results (cf. the SD in Table S2). Moreover, the ND obtained with the scenario extraction is comparable to the one obtained with the distance-based algorithms.

5.3 Comparison of NDs, objective distances and solution distances

To evaluate the nearness between the whole stochastic optimization problems considering the original stochastic tree and the reduced stochastic tree, we implement and solve the model proposed in Sect. 4 and we compute the objective and the here-and-now solution distances therein explained. In particular, the objective distances and the solution distances are computed between the result of the optimization when the original scenario tree is considered and the result of the optimization when the reduced scenario tree is considered. We adopt the original trees and the reduced trees discussed in the previous section, where each tree has four stages, $T = 4$. The initial wealth of the portfolio selection model is $W_0 = 100$, the diversification parameter $\theta = 0.65$, the turnover coefficient $\lambda = 0.30$ and, when we optimize the Average Value-at-Risk, the confidence $\alpha = 0.05$. Clearly, the dimension of the random variable of the scenario

Table 4 The ND between the original trees produced with the MC approach and the reduced trees for the different algorithm considered. Distinctions are made according to the reduction algorithm considered and the dimension of the node of the trees. In bold the best value among the distance-based algorithms for each case is presented

Algorithm		Nested Distance statistics							
		1000 → 100				10,000 → 100			
		Min	Mean	Max	SD	Min	Mean	Max	SD
Dimension 1	Nodal extraction	0.112	0.127	0.150	0.007	0.111	0.124	0.143	0.006
	Improved nodal extraction	0.104	0.118	0.148	0.007	0.111	0.122	0.150	0.006
	Scenario extraction	0.098	0.106	0.116	0.003	0.119	0.132	0.144	0.004
	Single scenario reduction		0.099				0.120		
	Nodal clustering		0.097				**0.087**		
	Adj-SubtreeM		0.136				0.139		
	Single node reduction		**0.090**				0.125		
Dimension 5	Nodal extraction	0.946	0.980	1.036	0.019	0.913	0.961	1.020	0.027
	Improved nodal extraction	0.833	0.876	0.945	0.020	0.877	0.936	1.042	0.023
	Scenario extraction	0.624	0.661	0.706	0.014	0.799	0.831	0.875	0.015
	Single scenario reduction		**0.556**				**0.706**		
	Nodal clustering		0.734				**0.706**		
	Adj-SubtreeM		0.798				0.875		
	Single node reduction		0.565				0.848		
Dimension 10	Nodal extraction	2.992	3.170	3.633	0.093	2.969	3.134	3.364	0.091
	Improved nodal extraction	2.683	2.845	3.117	0.093	2.875	3.044	3.411	0.092
	Scenario extraction	2.100	2.205	2.373	0.047	2.599	2.716	2.841	0.045
	Single scenario reduction		1.897				2.333		

Table 4 continued

Algorithm		Nested Distance statistics							
		1000 → 100				10,000 → 100			
		Min	Mean	Max	SD	Min	Mean	Max	SD
	Nodal clustering		2.323				**2.256**		
	Adj-SubtreeM		2.672				2.862		
	Single node reduction		**1.832**				2.729		
Dimension 20	Nodal extraction	4.742	4.979	5.235	0.093	4.641	4.895	5.405	0.113
	Improved nodal extraction	4.213	4.447	4.784	0.106	4.502	4.785	5.108	0.103
	Scenario extraction	3.170	3.354	3.509	0.070	3.906	4.067	4.238	0.064
	Single scenario reduction		**2.815**				4.230		
	Nodal clustering		3.734				**3.552**		
	Adj-SubtreeM		4.002				4.549		
	Single node reduction		2.907				3.994		

trees corresponds to the number of assets available for the portfolio selection, then we consider the cases with $A = 5$, 10, 20.

5.3.1 Objective distances

We start analysing the objective distance and considering the original tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the expected value of the final wealth. The results are in Fig. 6, where we represent the objective distances and the NDs, and in Table 5.

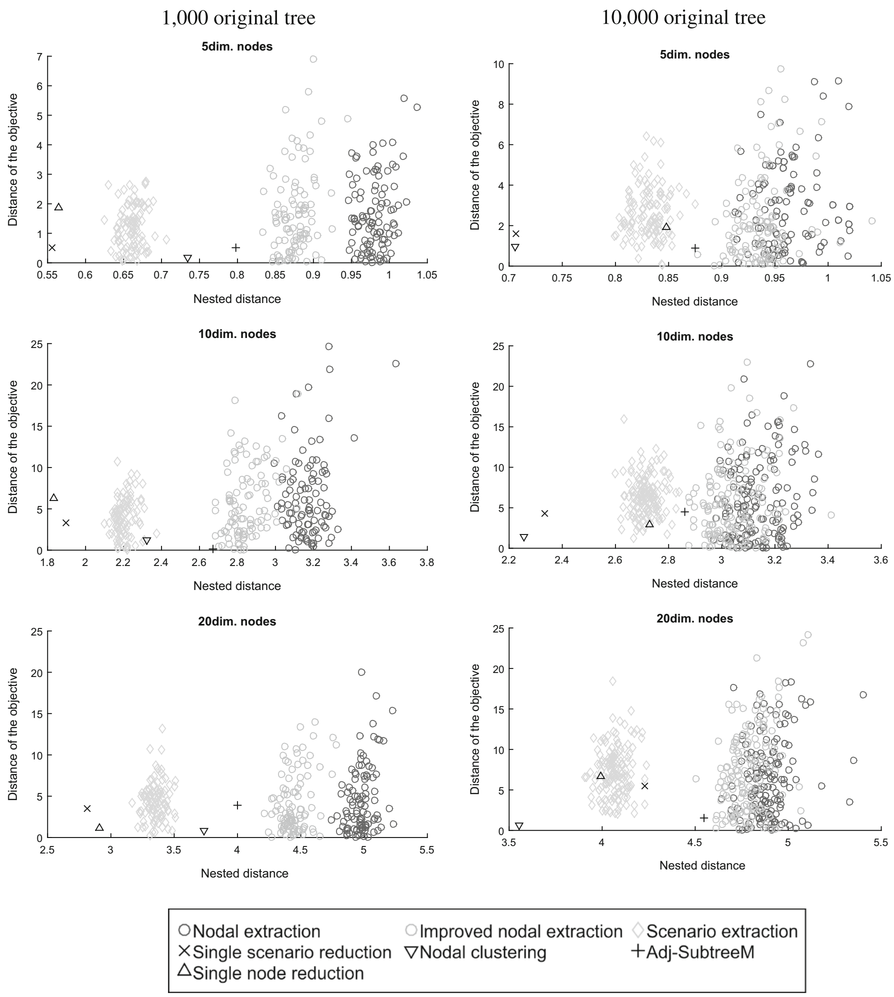

Fig. 6 The ND and the distance between the optimal objective values for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the expected value of the final wealth. On the left, the result starting from the original tree with 1000 scenarios, on the right, the result starting from the original tree with 10,000 scenarios. Each row corresponds to a different dimension for the node of the trees

Table 5 The objective distances for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the expected value of the final wealth. The best value of the distance-based algorithms is presented in bold for each case

Algorithm		Objective value distance statistics							
		1000 → 100				10,000 → 100			
		Min	Mean	Max	SD	Min	Mean	Max	SD
Dimension 5	Nodal extraction	0.020	1.670	5.570	1.240	0.100	2.733	9.140	2.049
	Improved nodal extraction	0.040	1.714	6.890	1.413	0.030	2.212	9.750	2.095
	Scenario extraction	0.060	1.223	2.740	0.712	0.070	2.664	6.430	1.237
	Single scenario reduction		0.510				1.610		
	Nodal clustering		**0.170**				0.970		
	Adj-SubtreeM		0.510				**0.870**		
	Single node reduction		1.870				1.910		
Dimension 10	Nodal extraction	0.020	6.490	24.670	5.131	0.010	6.061	22.790	4.877
	Improved nodal extraction	0.020	5.771	18.890	4.217	0.050	5.775	22.990	4.604
	Scenario extraction	0.080	3.953	10.750	2.293	1.260	6.468	15.940	2.370
	Single scenario reduction		3.270				4.250		
	Nodal clustering		1.200				**1.390**		
	Adj-SubtreeM		**0.100**				4.520		
	Single node reduction		6.290				2.880		
Dimension 20	Nodal extraction	0.090	4.577	20.040	4.112	0.070	6.412	18.370	4.733
	Improved nodal extraction	0.040	4.105	13.980	3.569	0.060	7.235	24.170	5.133
	Scenario extraction	0.200	4.453	13.160	2.300	2.060	7.291	18.390	2.960
	Single scenario reduction		3.550				5.470		
	Nodal clustering		**0.850**				**0.680**		
	Adj-SubtreeM		3.860				1.530		
	Single node reduction		1.100				6.700		

In the case of the 1000 scenario tree, on average, the objective distances observed for cases where the reduced tree comes from a distance-based reduction approach are smaller than the ones coming from the random reduction approaches. Of the distance-based approaches, the nodal clustering has the smallest objective distance in the 5- and 20-dimension cases and the second smallest in the 10-dimension case. The Adj-SubtreeM algorithm is the best in the 10-dimension case and equal to the single scenario reduction algorithm in the other cases. The single node reduction is the worst in the 5- and 10-dimension cases but its performance improves in the 20-dimension case.

In the case of the 10,000 scenario tree, the nodal clustering performs very well both in terms of the ND and in terms of objective distance. As in the 1000 case, we observe a stable behaviour of the nodal clustering algorithm meaning that the objective distance is constantly relatively low, while for the other distance-based algorithms we observe more volatile objective distances among the different cases.

The random algorithms have a very sparse behaviour in terms of objective distances. In particular, the scenario extraction algorithm proves to have a lower SD than the other random algorithms but, on average, a higher objective distance.

Considering the case of the original tree generated with the HE approach, see Figure S2 and Table S3, we observe similar results. In particular, the nodal clustering always gives a very low objective distance, the single scenario reduction is almost always the best, while the single node reduction is relatively volatile in terms of objective distance in all the cases. The significant difference is the worsening of the performance of the Adj-SubtreeM algorithm for which the objective distance proves to be relatively high. Of the random reduction algorithms, the single reduction algorithm is again the less volatile one but, in this case, the random reduction approach also produces on average the smallest objective distance.

Let’s consider the case when the original tree is generated with the MC approach and the formulation of the portfolio selection model is the optimization of the Average Value-at-Risk. The results are summarized in Fig. 7 and in Table 6. We notice that in all the cases the single node reduction generates the smallest objective distance in the distance-based reduction algorithms. The other algorithms show a more volatile behaviour. Only the single scenario reduction algorithm is second best in almost all the cases. Of the random reduction algorithms, the scenario extraction always has the smallest SD and the lowest objective distance on average. Moreover, we observe that in most of cases the average objective distance of the random algorithms is even better than the objective distance of the distance-based algorithms except for the single node reduction.

Now let’s consider the case when the original tree is generated with the HE approach and the formulation of the portfolio selection model is the optimization of the Average Value-at-Risk. The results are presented in Figure S3 and in Table S4. The results are very similar to the ones obtained in the MC case shown above. The most notable difference is that the best performing algorithm is not the single node reduction but the single scenario reduction. The Adj-SubtreeM still has the worst objective distance in most of the cases but the behaviour is better than in the MC case.

5.3.2 Solution distances

In this section we analyse the here-and-now solution distances in relation to the ND, and we start by analysing the cases in which the original tree is generated with the MC approach and the formulation of the portfolio selection model optimizes the expected value of the final wealth. The results are shown in Fig. 8 and in Table 7.

For those cases having the original scenario tree with 1000 scenarios, we observe that the single scenario reduction algorithm is always the one that produces the mini-

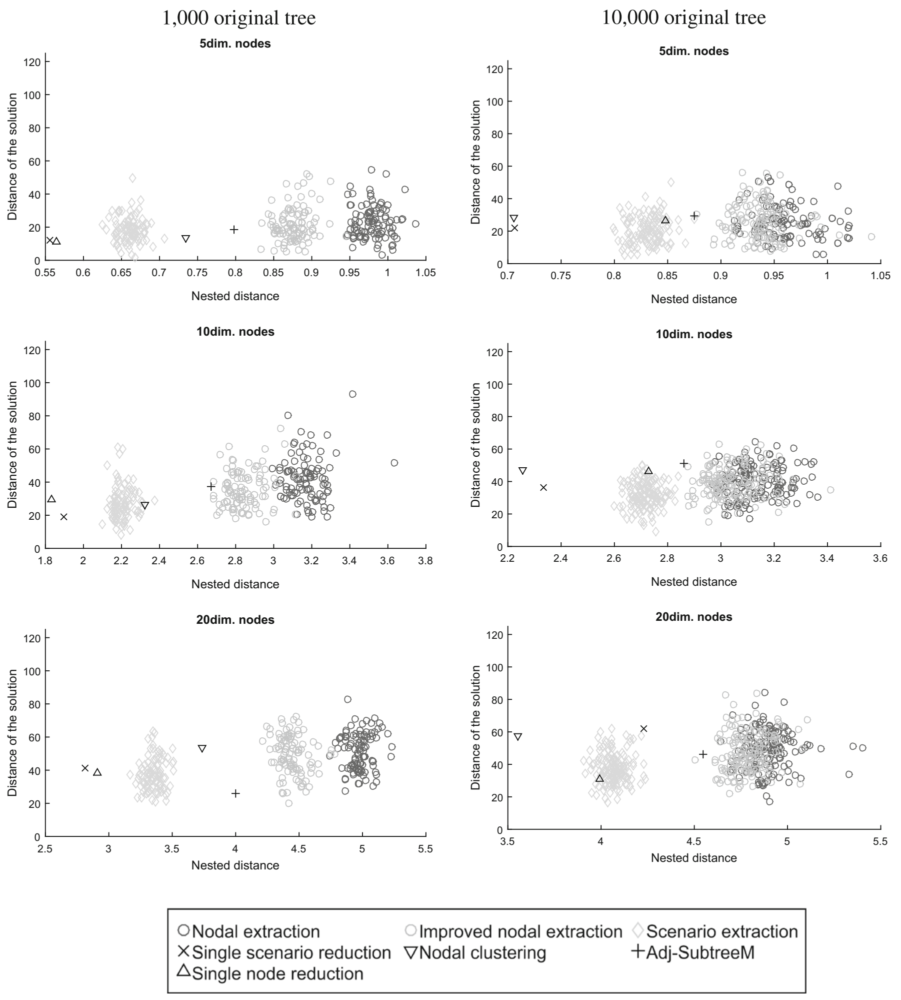

Fig. 7 The ND and the distance between the optimal objective values for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the Average Value-at-Risk of the final wealth. On the left, the result starting from the original tree with 1000 scenarios, on the right, the result starting from the original tree with 10,000 scenarios. Each row corresponds to a different dimension for the node of the trees

Table 6 The objective distances for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the Average Value-at-Risk of the final wealth. The best value of the distance-based algorithms is presented in bold for each case

Algorithm		Objective value distance statistics							
		$1000 \to 100$				$10{,}000 \to 100$			
		Min	Mean	Max	SD	Min	Mean	Max	SD
Dimension 5	Nodal extraction	0.000	1.584	4.970	1.188	0.020	2.095	5.790	1.377
	Improved nodal extraction	0.020	1.572	4.470	1.178	0.010	2.185	5.630	1.398
	Scenario extraction	0.010	1.407	2.910	0.734	0.040	1.946	4.490	1.041
	Single scenario reduction		**2.430**				3.410		
	Nodal clustering		4.390				5.130		
	Adj-SubtreeM		6.340				7.240		
	Single node reduction		2.440				**1.250**		
Dimension 10	Nodal extraction	0.080	2.635	12.470	2.064	0.080	3.222	7.250	1.578
	Improved nodal extraction	0.080	2.440	7.600	1.702	0.090	3.232	9.020	1.428
	Scenario extraction	0.020	2.187	5.140	1.108	0.590	2.829	5.630	0.985
	Single scenario reduction		3.140				4.720		
	Nodal clustering		4.440				6.180		
	Adj-SubtreeM		7.710				5.020		
	Single node reduction		**1.770**				**1.510**		
Dimension 20	Nodal extraction	0.490	2.806	5.900	1.162	0.560	4.056	8.400	1.541
	Improved nodal extraction	0.200	2.734	5.400	1.189	0.410	3.866	9.060	1.516
	Scenario extraction	0.820	2.620	5.130	0.776	1.430	3.662	6.030	0.878
	Single scenario reduction		1.300				8.220		
	Nodal clustering		3.410				4.910		
	Adj-SubtreeM		1.770				5.730		
	Single node reduction		**1.120**				**3.170**		

Table 7 The here-and-now solution distances for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the expected value of the final wealth. The best value of the distance-based algorithms is presented in bold for each case

Algorithm		Solution distance statistics							
		1000 → 100				10,000 → 100			
		Min	Mean	Max	SD	Min	Mean	Max	SD
Dimension 5	Nodal extraction	1.205	24.427	84.235	15.613	5.335	32.787	96.360	17.728
	Improved nodal extraction	5.305	23.517	59.175	13.369	2.080	34.574	90.915	18.231
	Scenario extraction	5.740	17.673	37.850	5.959	5.940	19.345	40.150	7.042
	Single scenario reduction		**3.740**				**12.170**		
	Nodal clustering		8.190				24.265		
	Adj-SubtreeM		8.190				18.740		
	Single node reduction		23.750				21.455		
Dimension 10	Nodal extraction	5.785	46.026	83.115	16.704	8.200	39.495	79.065	14.929
	Improved nodal extraction	4.255	44.803	83.115	14.072	1.700	40.405	82.145	18.929
	Scenario extraction	5.935	28.256	57.045	12.388	7.210	24.851	46.230	9.023
	Single scenario reduction		**25.505**				23.835		
	Nodal clustering		50.675				11.395		
	Adj-SubtreeM		39.105				34.100		
	Single node reduction		32.175				**8.770**		

Table 7 continued

Algorithm		Solution distance statistics							
		1000 → 100				10,000 → 100			
		Min	Mean	Max	SD	Min	Mean	Max	SD
Dimension 20	Nodal extraction	13.770	51.212	90.800	19.562	11.595	42.809	86.900	15.521
	Improved nodal extraction	3.330	45.838	96.530	17.727	7.465	49.510	83.365	15.435
	Scenario extraction	12.745	39.055	66.285	10.168	6.400	34.020	58.025	10.116
	Single scenario reduction		**16.665**				**20.585**		
	Nodal clustering		30.040				21.960		
	Adj-SubtreeM		78.500				69.685		
	Single node reduction		23.800				32.185		

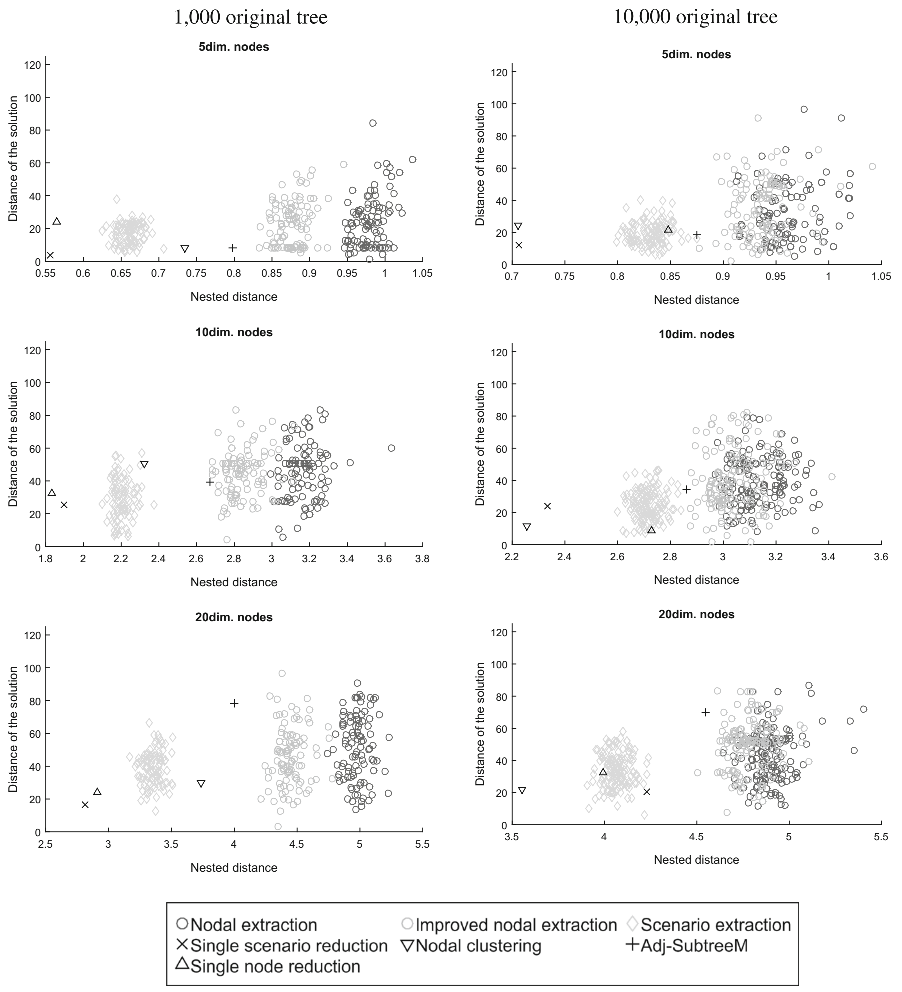

Fig. 8 The ND and the distance between the optimal here-and-now solutions for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the expected value of the final wealth. On the left, the result starting from the original tree with 1000 scenarios, on the right, the result starting from the original tree with 10,000 scenarios. Each row corresponds to a different dimension for the node of the trees

mum solution distance, while the other three distance-based algorithms have a volatile behaviour. Similarly, in the case of 10,000 scenarios, the general behaviour is still volatile. However, we can notice that the solution distance has values of around 10 and 30 for all the distance-based reduction algorithms, apart from the Adj-SubtreeM whose solution distance increases substantially. Considering the random algorithms, on average the scenario extraction gives the minimum solution distance and shows the lowest SD.

Let's consider the case when the formulation of the portfolio selection model is again the optimization of the expected value of the final wealth, but the original tree

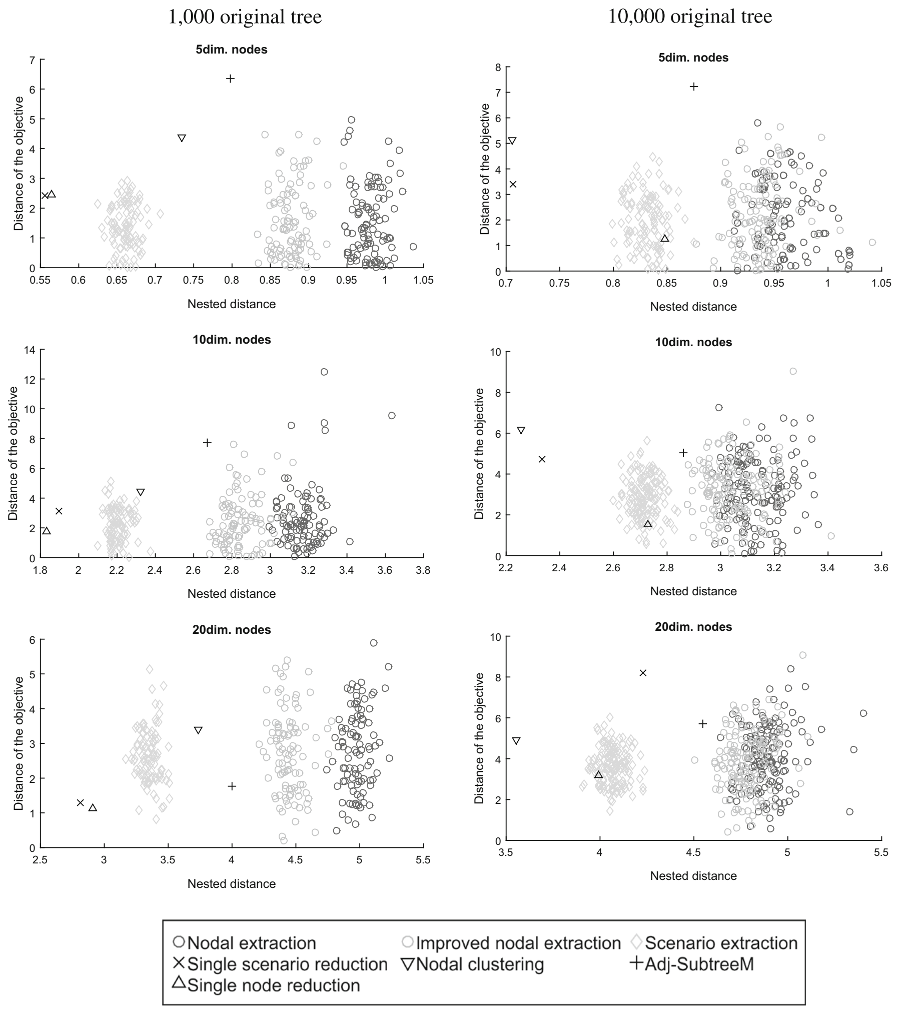

Fig. 9 The ND and the distance between the optimal here-and-now solutions for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the Average Value-at-Risk of the final wealth. On the left, the result starting from the original tree with 1000 scenarios, on the right, the result starting from the original tree with 10,000 scenarios. Each row corresponds to a different dimension for the node of the trees

is now generated with the HE approach. The results are shown in Figure S4 and in Table S5. We notice that the Adj-SubtreeM algorithm is the worst performing of the distance-based algorithms in terms of solution distance. Considering the other three distance-based algorithms, the solution distance is quite volatile and we do not observe any algorithm that is generally better than the others.

In Fig. 9 and in Table 8, we consider the case where the original tree is generated with the MC approach and the formulation of the portfolio selection model is the optimization of the Average Value-at-Risk. Again, the four distance-based algorithms have a volatile behaviour which means that it is not possible to identify one that performs better than the others. On the contrary, of the random algorithms, the scenario extraction still produces, on average, the lowest solution distance with the lowest SD.

In Figure S5 and in Table S6, we show the case where the original tree is generated with the HE approach and the formulation of the portfolio selection model is the optimization of the Average Value-at-Risk. As before, the distance-based algorithms perform in a similar way. In the 10,000 scenario case, we notice that the single scenario reduction always performs best. The scenario extraction algorithm is again the best of the random algorithms.

6 Conclusion

In this paper, we analyse the quality of several algorithms that can be used to reduce a multistage scenario tree. We consider four original multistage scenario trees by combining:

– an established number of scenarios, 1000 or 10,000;
– a scenario generating method, either Monte Carlo or Historical Extraction (Kopa et al. 2018).

To reduce these four scenario trees, we implemented seven scenario tree reduction algorithms, three random and four distance-based ones. Among the four distance-based algorithms, we proposed a new one based on a clustering technique, thus, we name it nodal clustering. Moreover, since one of the other distance-based algorithms, the Subtree Merging, appeared computationally intractable (for large number of scenarios) we introduced and analysed two more tractable versions of this algorithm and the role of the main parameter p. We found the Adj-SubtreeM algorithm with parameter $p = 0.5$ as the best choice. Finally, we compared the computational time and the Nested Distance between the original tree and the reduced trees applying the seven reduction algorithms.

In case of the distance-based algorithms, the results show that the nodal clustering and the single node reduction algorithms are typically the fastest, while the single scenario reduction and the Adj-SubtreeM are quite computationally demanding. The results can be summarized as follows:

– considering the original tree with 1000 scenarios, the single node reduction and the single scenario reduction produce the lowest Nested Distance in most of the cases;
– considering the original tree with 10,000 scenarios, the two best algorithms are the single scenario reduction and the nodal clustering;
– thus, in general, we could say that the single node reduction, the single scenario reduction and the nodal clustering are almost equivalent in terms of Nested Distance performance, but the single scenario reduction is the slowest

To perform a deeper comparison between the seven reduction algorithms, we also formulated a multistage portfolio selection problem and we solved it both with the original trees and with the reduced trees, and we compared the optimal objective values and the first stage solutions. We can report the following observations:

Table 8 The here-and-now solution distances for all the reduction algorithms considering the original scenario tree generated with the MC approach and the formulation of the portfolio selection model that optimizes the Average Value-at-Risk of the final wealth. The best value of the distance-based algorithms is presented in bold for each case

Algorithm		Solution distance statistics							
		1000 → 100				10,000 → 100			
		Min	Mean	Max	SD	Min	Mean	Max	SD
Dimension 5	Nodal extraction	3.195	23.047	54.565	9.914	5.480	25.909	53.210	10.411
	Improved nodal extraction	4.960	21.645	52.145	10.209	7.450	26.072	56.110	11.281
	Scenario extraction	1.890	18.327	49.765	7.451	5.410	20.707	50.005	8.407
	Single scenario reduction		12.120				**22.075**		
	Nodal clustering		13.455				28.490		
	Adj-SubtreeM		18.660				29.505		
	Single node reduction		**10.920**				26.595		
Dimension 10	Nodal extraction	19.100	42.118	92.975	13.332	16.955	39.852	64.300	9.895
	Improved nodal extraction	15.960	35.557	72.400	11.250	15.275	39.381	63.045	9.380
	Scenario extraction	7.950	27.297	61.190	10.006	9.060	31.640	50.005	7.818
	Single scenario reduction		**18.805**				**36.135**		
	Nodal clustering		26.285				46.955		
	Adj-SubtreeM		37.495				51.350		
	Single node reduction		29.265				46.060		

Table 8 continued

Algorithm		Solution distance statistics							
		1000 → 100				10,000 → 100			
		Min	Mean	Max	SD	Min	Mean	Max	SD
Dimension 20	Nodal extraction	27.245	51.222	82.660	11.184	17.170	48.892	84.190	11.783
	Improved nodal extraction	20.245	48.989	72.215	11.674	21.880	47.327	83.835	12.385
	Scenario extraction	21.030	37.542	63.675	9.483	16.740	37.579	61.830	9.235
	Single scenario reduction		41.435				62.180		
	Nodal clustering		53.655				57.495		
	Adj-SubtreeM		**26.160**				46.295		
	Single node reduction		38.460				**30.825**		

- considering the random algorithms, the scenario extraction algorithm appears to be the best in terms of the Nested Distance, objective value distance and solution distance; moreover, it is very fast, even if slightly slower than the other random algorithms;
- considering the distance-based algorithms, they perform similarly in terms of objective value distance and solution distance; only the Adj-SubtreeM suffers a relatively high objective value distance and solution distance in almost all the cases. We are aware that the authors of this algorithm, see Pflug and Pichler (2014), proposed a further improvement. However, since it is even more computationally demanding, it was not considered in this paper.

To conclude, we believe that the Nested Distance can be efficiently used to analyse the quality of scenario reduction algorithms. Furthermore, to fully compare scenario reduction algorithms, the distance between stochastic models should be measured also in terms of objective value distance and here-and-now solution distance.

Acknowledgements Open access funding provided by Università degli Studi di Bergamo within the CRUI-CARE Agreement.

References

Beraldi P, Bruni ME (2014) A clustering approach for scenario tree reduction: an application to a stochastic programming portfolio optimization problem. Top 22(3):934–949

Birge JR, Louveaux F (1997) Introduction to stochastic programming. Springer, Berlin

Chen Z, Yan Z (2018) Scenario tree reduction methods through clustering nodes. Comput Chem Eng 109:96–111

Consigli G, Moriggia V, Benincasa E, Landoni G, Petronio F, Vitali S, di Tria M, Skoric M, Uristani A (2018a) Optimal multistage defined-benefit pension fund management. In: Handbook of Recent Advances in Commodity and Financial Modeling, Springer, Berlin pp 267–296

Consigli G, Moriggia V, Vitali S, Mercuri L (2018b) Optimal insurance portfolios risk-adjusted performance through dynamic stochastic programming. CMS 15(3–4):599–632

Crainic TG, Hewitt M, Rei W (2014) Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. Comput Oper Res 43:90–99

Dupačová J, Hurt J, Štěpán J (2002) Stochastic modeling in economics and finance, applied Optimization. Kluwer,

Dupačová J, Gröwe-Kuska N, Römisch W (2003) Scenario reduction in stochastic programming. Math Program 95(3):493–511

Growe-Kuska N, Heitsch H, Romisch W (2003) Scenario reduction and scenario tree construction for power management problems. In: Power tech conference proceedings, 2003 IEEE Bologna, vol 3, p 7

Heitsch H, Römisch W (2003) Scenario reduction algorithms in stochastic programming. Comput Optim Appl 24(2–3):187–206

Heitsch H, Römisch W (2009) Scenario tree reduction for multistage stochastic programs. CMS 6(2):117–133

Kaufman L, Rousseeuw PJ (2009) Finding groups in data: an introduction to cluster analysis. Wiley, New York

Kilianová S, Pflug GC (2009) Optimal pension fund management under multi-period risk minimization. Ann Oper Res 166(1):261–270

Kopa M, Petrová B (2017) Multistage risk premiums in portfolio optimization. Kybernetika 53(6):992–1011

Kopa M, Moriggia V, Vitali S (2018) Individual optimal pension allocation under stochastic dominance constraints. Ann Oper Res 260(1–2):255–291

Kovacevic RM, Pichler A (2015) Tree approximation for discrete time stochastic processes: a process distance approach. Ann Oper Res 235(1):395–421

Mandelli D, Yilmaz A, Aldemir T, Metzroth K, Denning R (2013) Scenario clustering and dynamic probabilistic risk assessment. Reliab Eng Syst Saf 115:146–160

Moriggia V, Kopa M, Vitali S (2019) Pension fund management with hedging derivatives, stochastic dominance and nodal contamination. Omega 87:127–141

Pflug GC, Pichler A (2015) Convergence of the Smoothed Empirical Process in Nested Distance. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Mathematik, https://doi.org/10.18452/8448

Pflug GC, Pichler A (2012) A distance for multistage stochastic optimization models. SIAM J Optim 22(1):1–23

Pflug GC, Pichler A (2014) Multistage stochastic optimization. Springer, Berlin

Pflug GC, Pichler A (2016) From empirical observations to tree models for stochastic optimization: convergence properties. SIAM J Optim 26(3):1715–1740

Powell WB (2014) Clearing the jungle of stochastic optimization. In: Bridging data and decisions, Informs, pp 109–137

Rockafellar TR, Uryasev S (2000) Optimization of conditional value-at-risk. J Risk 2:21–42

Rockafellar TR, Uryasev S (2002) Conditional value-at-risk for general loss distributions. J Banking & Financ 26(7):1443–1471

Rusý T, Kopa M (2018) An asset-liability management stochastic program of a leasing company. Kybernetika 54(6):1247–1263

Shapiro A, Dentcheva D, Ruszczyński A (2009) Lectures on stochastic programing. Modeling and theory SIAM and mathematical programming society. https://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/SPbook.pdf

Timonina AV (2015) Multi-stage stochastic optimization: the distance between stochastic scenario processes. CMS 12(1):171–195

Vitali S (2018) Multistage multivariate nested distance: an empirical analysis. Kybernetika 54(6):1184–1200

Vitali S, Moriggia V, Kopa M (2017) Optimal pension fund composition for an Italian private pension plan sponsor. CMS 14(1):135–160

Affiliations

Markéta Horejšová[1] · Sebastiano Vitali[1,2] · Miloš Kopa[1] · Vittorio Moriggia[2]

Markéta Horejšová
horejsova@karlin.mff.cuni.cz

Miloš Kopa
kopa@karlin.mff.cuni.cz

Vittorio Moriggia
vittorio.moriggia@unibg.it

1 Department of Probability and Mathematical Statistics, Faculty of Mathematics and Physics, Charles University, Sokolovska 83, 186 75 Prague, Czech Republic

2 Department of Management, Economics and Quantitative Methods, University of Bergamo, Via dei Caniana 2, 24127 Bergamo, Italy