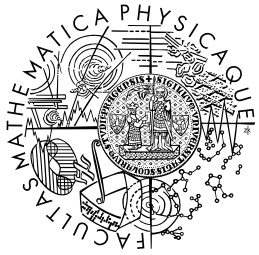


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Markéta Horejšová

Multistage nested distance in stochastic optimization

Department of Probability and Mathematical Statistics

Supervisor of the master thesis:	Sebastiano Vitali, Ph.D.
Study programme:	Mathematics
Study branch:	Probability, Mathematical Statistics and Econometrics

Prague 2018

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

Prague, 6th May 2018 signature of the author

Title: Multistage nested distance in stochastic optimization

Author: Bc. Markéta Horejšová

Department: Department of Probability and Mathematical Statistics

Supervisor: Sebastiano Vitali, Ph.D., Department of Probability and Mathematical Statistics

Abstract: Multistage stochastic optimization is used to solve many real-life problems where decisions are taken at multiple times, e.g., portfolio selection problems. Such problems need the definition of stochastic processes, which are usually approximated by scenario trees. The choice of the size of the scenario trees is the result of a compromise between the best approximation and the possibilities of the computer technology. Therefore, once a master scenario tree has been generated, it can be needed to reduce its dimension in order to make the problem computationally tractable. In this thesis, we introduce several scenario reduction algorithms and we compare them numerically for different types of master trees. A simple portfolio selection problem is also solved within the study. The distance from the initial scenario tree, the computational time, and the distance between the optimal objective values and solutions are compared for all the scenario reduction algorithms. In particular, we adopt the nested distance to measure the distance between two scenario trees.

Keywords: nested distance, multistage stochastic optimization, scenario tree reduction

I would like to thank my supervisor, Dr. Vitali, for finding me an interesting topic for my thesis, and for his guidance and advice.

Contents

Introduction

In many real-life optimization problems the data are uncertain, e.g., they could represent some future value such as price or demand, and decisions are made on several time occasions, e.g., board meetings every week. To solve these problems the theory of multistage stochastic optimization is used. Unlike single stage stochastic optimization where only one decision is made, multistage stochastic optimization considers that several decisions are to be made. It is assumed that the data are random variables with known distribution. This distribution is usually approximated by a discrete distribution represented by a scenario tree.

Since, in order to approximate the initial distribution as much as possible, scenario trees can be rather huge, so much that solving the optimization problem is beyond the computational capabilities, several scenario reduction algorithms were proposed in the literature, e.g., [2], [4], or [7]. In this thesis we focus on comparing different reducing algorithms. For this reason also a simple portfolio selection problem ([3]) is solved. For each of the algorithms we measure the distance between the initial tree (the master tree) and the reduced tree, the time needed to generate the reduced tree, and the distances between the optimal objective values and solutions. To measure the distance between two scenario trees we use the nested distance from the theory of probability measures.

The first two chapters of this thesis are dedicated to the theory. The first chapter summarizes the fundamentals of multistage stochastic problems and the nested distance; the second chapter introduces various scenario reduction algorithms and apply them on an example. All the algorithms are thoroughly compared in the third chapter of this thesis.

It is expected that with more sophisticated scenario reduction algorithms the quality of the reduction should be better than for simple algorithms. The aim of this thesis is to observe how big is the improvement in relation to the increased computational complexity and the time demand.

1. Fundamentals

In this chapter we summarize the basics of multistage stochastic optimization, the theory of probability measures, and the nested distance. For more thorough explanation see [7] or [3].

1.1 Multistage stochastic programs

In general, when we talk about a T-stage stochastic program we think of a stochastic data process

$$\xi = (\xi_1, \dots, \xi_T)$$

and a decision process

$$x = (\boldsymbol{x}_0, \dots, \boldsymbol{x}_T).$$

The realizations of ξ are usually called *trajectories* or *scenarios*. The probability distribution of ξ will be denoted by P and its support by Ω. The sequence of decisions and observations is then

$$\boldsymbol{x}_0, \xi_1, \boldsymbol{x}_1, \dots, \xi_T, \boldsymbol{x}_T.$$

Information available during the stages is represented by a filtration $\mathcal{F} = (\mathcal{F}_1, \dots, \mathcal{F}_T)$, where $\mathcal{F}_t,\ t = 1, \dots, T$, are σ-algebras and $\mathcal{F}_s \subset \mathcal{F}_t\ \forall\ 1 \leq s \leq t \leq T$. We assume that the decision process does not depend on future realizations of ξ nor on future decisions. The past and the probability distribution of ξ, however, are known. This assumption could be written as

$$\boldsymbol{x}_t \lhd \mathcal{F}_t\ \ \forall\ t = 1, \dots, T,$$

and it is called the *nonanticipativity constraint*. It means that $\boldsymbol{x}_t$ is $\mathcal{F}_t$-measurable for every t.

Note that the time intervals do not need to be equidistant and the stages can also correspond to steps in the decision process. Very often we also add a deterministic observation ξ_0 and a trivial σ-algebra $\mathcal{F}_0 = \{\emptyset, \Omega\}$. The notation varies in the literature, e.g., the stage is indexed from 1 or the sequence of decisions and observations is

$$\boldsymbol{x}_1, \xi_1, \boldsymbol{x}_2, \dots, \xi_{T-1}, \boldsymbol{x}_T.$$

However, if we speak of T-stage stochastic programs, we mean programs in which we make T decisions. Therefore, we disregard the decision $\boldsymbol{x}_T$ in our notation.

Using the notation defined above, T-stage stochastic program could be written in the form

$$\min\{Q(x, \xi) : x \in \mathbb{X}, \boldsymbol{x}_t \lhd \mathcal{F}_t\ \ \forall\ t = 0, \dots, T\},$$

where $Q(x, \xi)$ is a cost function and $\mathbb{X}$ is the set of constraints. A special case of T-stage stochastic programs is a linear T-stage stochastic program with recourse, which has the following form

$$\min\{\boldsymbol{c}_0^\top \boldsymbol{x}_0 + \mathsf{E}_{\xi_1}\, \phi_1(\boldsymbol{x}_0, \xi_1) : A_0 \boldsymbol{x}_0 = \boldsymbol{b}_0,\ \boldsymbol{l}_0 \leq \boldsymbol{x}_0 \leq \boldsymbol{u}_0\},$$

where for $t = 1, \dots, T-1$ the function ϕ_t is defined by

$$\phi_t(\boldsymbol{x}_{t-1}, \xi_t) = \min\{\boldsymbol{c}_t(\xi_t)^\top \boldsymbol{x}_t + \mathsf{E}_{\,\xi_{t+1}}\, \phi_{t+1}(\boldsymbol{x}_t, \xi_{t+1}) : \\ A_t(\xi_t)\boldsymbol{x}_{t-1} + B_t\boldsymbol{x}_t = \boldsymbol{b}_t(\xi_t),\ \boldsymbol{l}_t \le \boldsymbol{x}_t \le \boldsymbol{u}_t\}$$

and

$$\phi_T(\boldsymbol{x}_{T-1}, \xi_T) = \min\{\boldsymbol{c}_T(\xi_T)^\top \boldsymbol{x}_T : \\ A_T(\xi_T)\boldsymbol{x}_{T-1} + B_T\boldsymbol{x}_T = \boldsymbol{b}_T(\xi_T),\ \boldsymbol{l}_T \le \boldsymbol{x}_T \le \boldsymbol{u}_T\}.$$

1.1.1 Scenario based stochastic programs

Assume now that the probability distribution of ξ is finite and discrete, i.e., concentrated on a finite number of scenarios. A scenario is denoted by

$$\omega^s = (\omega_0^s, \ldots, \omega_T^s).$$

Further, we assume a specific organization of data in the form of a scenario tree. By $\tilde{\omega}_{k_t},\ k_t \in \mathcal{K}_t$, we denote all possible realizations of ξ_t, where $\mathcal{K}_t$ are disjoint sets of indices, $t = 0, \ldots, T$, by S we denote the total number of scenarios, which is equal to the number of elements in $\mathcal{K}_T$, and by k_t- we denote the index of the unique ancestor of $\tilde{\omega}_{k_t}$. The program then has the following form

$$\begin{aligned} \min\{\boldsymbol{c}_0^\top \boldsymbol{x}_0 + \sum_{k_1 \in \mathcal{K}_1} p_{k_1} \boldsymbol{c}_{k_1}^\top \boldsymbol{x}_{k_1} + \cdots + \sum_{k_T \in \mathcal{K}_T} p_{k_T} \boldsymbol{c}_{k_T}^\top \boldsymbol{x}_{k_T} : \\ A_0\boldsymbol{x}_0 = \boldsymbol{b}_0 \\ B_{k_1}\boldsymbol{x}_0 + A_{k_1}\boldsymbol{x}_{k_1} = \boldsymbol{b}_{k_1},\ k_1 \in \mathcal{K}_1 \\ B_{k_2}\boldsymbol{x}_{k_2-} + A_{k_2}\boldsymbol{x}_{k_2} = \boldsymbol{b}_{k_2},\ k_2 \in \mathcal{K}_2 \\ \vdots \\ B_{k_T}\boldsymbol{x}_{k_T-} + A_{k_T}\boldsymbol{x}_{k_T} = \boldsymbol{b}_{k_T},\ k_T \in \mathcal{K}_T \\ \boldsymbol{l}_{k_t} \le \boldsymbol{x}_{k_t} \le \boldsymbol{u}_{k_t},\ k_t \in \mathcal{K}_t,\ t = 0, \ldots, T\}, \end{aligned}$$

where $p_{k_t} > 0$ signify the path probabilities, $\sum_{k_t \in \mathcal{K}_t} p_{k_t} = 1,\ t = 1, \ldots, T$. The path probabilities may be obtained by stepwise multiplication of the marginal probabilities p_{k_1} by the related conditional probabilities $\pi_{p_\tau, p_{\tau+1}}$.

Example. One of the typical examples of multistage stochastic programs is the problem of a private investor. A private investor has some initial wealth, he wants to maximize his expected return in some time by investing in assets, and he has the possibility to reinvest his money during the time period. See [3] for more details and additional examples.

1.2 Scenario trees

As was already mentioned above, scenario trees are a way of defining a multistage stochastic optimization on finite probability spaces. The most relevant feature of a tree is its topology. This way, trees could be viewed as equivalence classes with respect to bijective mappings which preserve the precedence topology, i.e., the numbering and ordering of the nodes is irrelevant, any numbering or ordering of the nodes generates the same tree. For this reason we can always take one representative of a class and label its nodes.

1.2.1 Representation of a scenario tree

Suppose that the tree consists of N nodes $\{1, \dots, N\}$. The root node is the node 1. For each node n we define its stage, which is its distance from the root, we denote its direct successors (children) by $n+$, and for each node except for the root we also denote its direct predecessor by $n-$. All nodes are divided into disjoint node sets, which we denote by $\mathcal{K}_t,\ t = 0, \dots, T$, according to the stages. In this notation, $\mathcal{K}_0 = \{1\}$ is the root, $\mathcal{K}_T$ are the leaves, and $\mathcal{K}_t,\ t = 1, \dots, T-1$, are the inner nodes.

If we look at the scenario trees as at special directed finite graphs, they carry the probability valuations on the nodes (the unconditional probabilities) and arcs (the conditional branching probabilities). It suffices to assign the unconditional probabilities p_n only to the leaf nodes, i.e., $n \in \mathcal{K}_T$, since for all inner nodes they could be attained recursively by

$$p_m = \sum_{n \in m+} p_n.$$

The conditional arc probabilities are then defined by

$$q_n = \pi_{n,n-} = \frac{p_n}{p_{n-}}.$$

In the opposite direction, we can get the unconditional probabilities from the conditional ones recursively by

$$p_n = q_n \cdot p_{n-},$$

where, obviously, we set the root probabilities $p_1 = q_1 = 1$.

Apart from the information about the structure, each node could also carry a vector of scenario values $\xi(n) \in \mathbb{R}^m$.

Tree processes. A scenario tree could be equivalently described as a tree process $(\nu_t,\ t = 0, \dots, T)$ with values in the state space $\mathcal{K}_t,\ t = 0, \dots, T$, where $\mathcal{K}_t$ are pairwise disjoint, $\mathcal{K}_0$ is a singleton, and it satisfies

$$\sigma(\nu_t) = \sigma(\nu_0, \dots, \nu_t)\ \forall t.$$

The tree process defined above induces a probability distribution P on $\mathcal{K}_T$. Without loss of generality we may set the image space $\mathcal{K}_T$ as the basic probability space, i.e., $\Omega = \mathcal{K}_T$. The filtration $(\mathcal{F}_0, \dots, \mathcal{F}_T)$ then consists of the degenerated sigma algebras $\mathcal{F}_t := \sigma(\nu_t)$.

Example. Example of a two-stage scenario tree representation is shown in Figure 1.1 on the following page. The conditional probabilities are included in the graph, the unconditional could be easily calculated, e.g., $\mathsf{P}(\{\omega_1\}) = p_4 = 0.8 \cdot 0.2 = 0.16$, $\mathsf{P}(\{\omega_2\}) = p_5 = 0.8 \cdot 0.3 = 0.24$, etc.

As we can see, the tree consists of 8 nodes and has 5 scenarios. Each path of the tree is represented by $\omega_k,\ k = 1, \dots, 5$. For instance, ω_1 corresponds to the path (1,2,4), or ω_4 to the path (1,3,7). The node sets are as follows

$$\mathcal{K}_0 = \{1\},$$

$$\mathcal{K}_1 = \{2, 3\}, \text{ and}$$
$$\mathcal{K}_2 = \{4, 5, 6, 7, 8\}.$$

Figure 1.2 shows the corresponding filtration $(\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2)$ induced by this tree process. The filtration is defined by

$$\mathcal{F}_0 = \sigma(\{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}) = \{\emptyset, \Omega\},$$
$$\mathcal{F}_1 = \sigma(\{\omega_1, \omega_2, \omega_3\}, \{\omega_4, \omega_5\}), \text{ and}$$
$$\mathcal{F}_2 = \sigma(\{\omega_1\}, \{\omega_2\}, \{\omega_3\}, \{\omega_4\}, \{\omega_5\}).$$

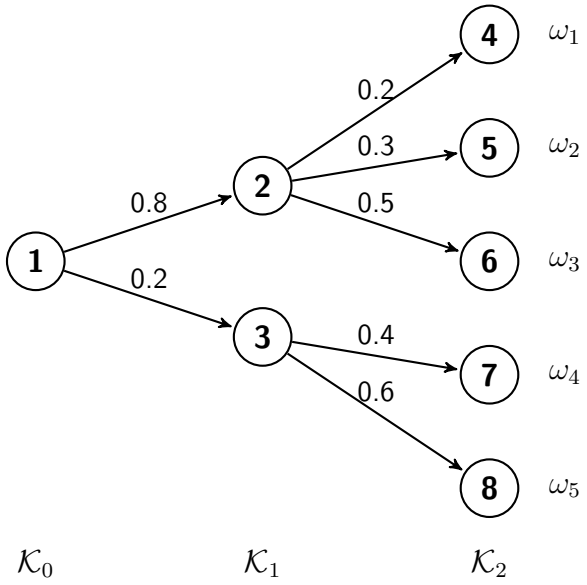


Figure 1.1: An example of a tree structure

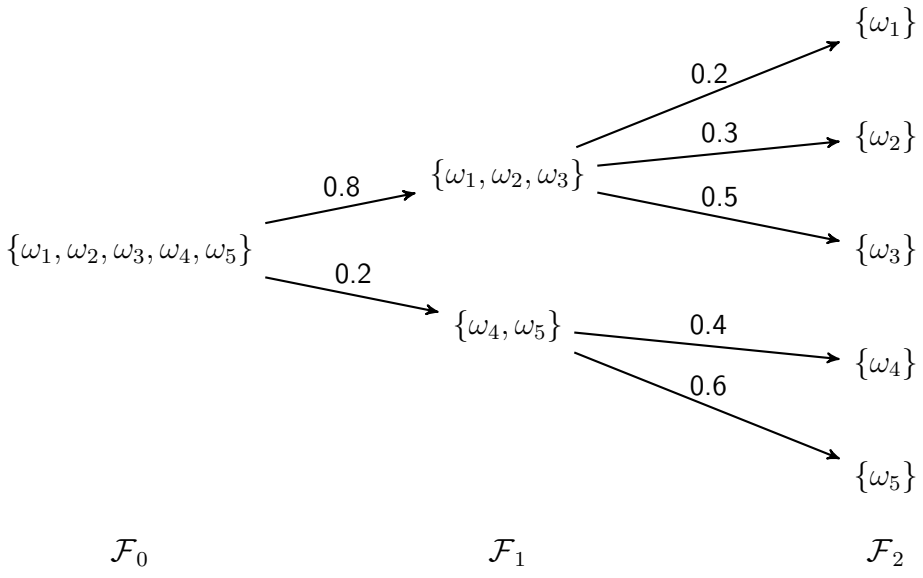


Figure 1.2: Induced filtration

1.2.2 Nested distribution

Now we would like to connect the scenario process $\xi_t,\ t = 0, \ldots, T$, to the tree process defined in the preceding subsection. The usual approach is to choose a standard representation of the values of the tree process as the conditional distribution of the scenario process ξ. One can do this, since only the distribution of the tree process matters. Naturally, we assume that the scenario process ξ is adapted to the filtration $\mathcal{F}$ induced by the tree process, i.e., $\xi_t \lhd \mathcal{F}_t,\ t = 0, \ldots, T$.

The structure $(\Omega, \mathcal{F}, \mathsf{P}, \xi)$ is called the *value-and-information structure*, where the tree process represents the information structure in the decision process, and the scenario process ξ, which is a function of the tree process ν, is the basis of the decisions.

Let us assume that the scenario process ξ has values in $\mathbb{R}^m$. Then we recursively define spaces:

$$\mathcal{X}_1 = \mathbb{R}^m,$$

representing one fixed value at the root node ξ_0,

$$\mathcal{X}_2 := \mathbb{R}^m \times \mathcal{P}(\mathcal{X}_1),$$

representing the root value $\xi_0 \in \mathbb{R}^m$ and the scenario distribution $\xi_1 \in \mathcal{P}(\mathcal{X}_1)$ at time 1 in one object, where $\mathcal{P}$ signifies all Borel probability measures on $(\mathbb{R}^m, \mathtt{d})$, where $\mathtt{d}$ is some metric on $\mathbb{R}^m$,

$$\mathcal{X}_3 := \mathbb{R}^m \times \mathcal{P}(\mathcal{X}_2) = \mathbb{R}^m \times \mathcal{P}(\mathbb{R}^m \times \mathcal{P}(\mathbb{R}^m)),$$

representing the initial root value ξ_0, the distribution of ξ_1, and the conditional distribution of ξ_2. We then iterate T times to finally get

$$\mathcal{X}_T := \mathbb{R}^m \times \mathcal{P}(\mathcal{X}_{T-1}).$$

The probability distribution $\mathbb{P}$ on $\mathcal{X}_T$ is called the *nested distribution of depth T*.

Example. Continuing the example of a tree process from the preceding section (Figures 1.1 and 1.2), Figure 1.3 shows a nested distribution displayed by a tree.

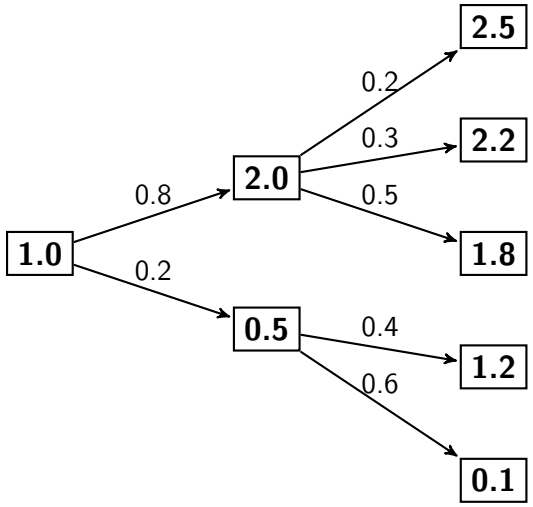


Figure 1.3: A nested distribution displayed by a tree

1.3 Nested distance

Nowadays, in most applications we want to approximate some stochastic process, e.g., to be able to solve some optimization problem. In order to know how good or bad our approximation is, we measure the distance between the induced probability measures. In other words, we have some distribution P on $\mathbb{R}^m$, we are approximating it by another distribution P', which is usually simpler, and we want to measure the distance between P and P'.

1.3.1 Probability measures

There exists a lot of different distances between probability measures in the literature, see [9] and [10]. In stochastic optimization, the distance should extend to general stochastic processes, measure distances between distributions, be independent of different underlying probability spaces, enable discrete approximations, and it should be possible to implement it.

Distance. Let $\mathcal{P}$ be a set of probability measures on $\mathbb{R}^m$. A measure $\mathtt{d}$ defined on $\mathcal{P} \times \mathcal{P}$ is called a *distance* if it satisfies:

1. (*nonnegativity*) for all $\mathsf{P}_1, \mathsf{P}_2 \in \mathcal{P}$:

$$\mathtt{d}(\mathsf{P}_1, \mathsf{P}_2) \geq 0;$$

2. (*symmetry*) for all $\mathsf{P}_1, \mathsf{P}_2 \in \mathcal{P}$:

$$\mathtt{d}(\mathsf{P}_1, \mathsf{P}_2) = \mathtt{d}(\mathsf{P}_2, \mathsf{P}_1);$$

3. (*triangle inequality*) for all $\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3 \in \mathcal{P}$:

$$\mathtt{d}(\mathsf{P}_1, \mathsf{P}_2) \leq \mathtt{d}(\mathsf{P}_1, \mathsf{P}_3) + \mathtt{d}(\mathsf{P}_3, \mathsf{P}_2);$$

4. (*strictness*) if $\mathtt{d}(\mathsf{P}_1, \mathsf{P}_2) = 0$, then $\mathsf{P}_1 = \mathsf{P}_2$.

If only conditions 1-3 are satisfied, then $\mathtt{d}$ is called a *semi-distance*.

1.3.2 The Wasserstein distance

A popular distance used in multistage stochastic optimization is the so-called *Wasserstein distance*, which is a generalization of the Kantorovich distance (see [7]).

Suppose we have two probability spaces $(\Omega, \mathcal{F}, \mathsf{P})$ and $(\Omega', \mathcal{F}', \mathsf{P}')$, and ξ and ξ' are $\mathbb{R}^m$-valued random variables on Ω and Ω', respectively. Then we define an *inherited distance* between the elements of Ω and Ω' by

$$\mathtt{d}(\omega, \omega') := c(\omega, \omega') = \mathtt{d}(\xi(\omega), \xi'(\omega')),$$

where $c : \Omega \times \Omega' \to \mathbb{R}$ is a cost function and $\mathtt{d}$ some distance on $\mathbb{R}^m$.

Distances between elements could be extended to the distance between probabilities. We define the *optimal transportation cost* as

$$\inf_{\pi} \iint_{\Omega\times\Omega'} c(\omega,\omega')d\pi(\omega,\omega'),$$

where π is a bivariate probability measure on $\Omega \times \Omega'$ satisfying

$$\pi(A \times \Omega') = \mathsf{P}(A),\ \pi(\Omega \times B) = \mathsf{P}'(B)$$

for all measurable sets $A \in \mathcal{F}$ and $B \in \mathcal{F}'$. In other words, π is a bivariate probability measure on $\Omega \times \Omega'$ with P and P' as marginals. The optimal measure π^* is called the *optimal transportation plan.*

Wasserstein distance. Now we can define the *Wasserstein distance of order* r ($r \geq 1$) as

$$\mathtt{d}_r(\mathsf{P},\mathsf{P}') := \left(\inf_{\pi} \iint_{\Omega\times\Omega'} \mathtt{d}(\omega,\omega')^r d\pi(\omega,\omega')\right)^{\frac{1}{r}},$$

where the infimum is taken over all joint probability measures π on $\Omega \times \Omega'$ with P and P' as marginals.

Sufficient condition for the infimum to exist is that both measures P and P' are tight, since the integrand is continuous (see [5]).

The Wasserstein distance satisfies several properties – it is monotone, convex, bounded, and it controls all moments. See [7] for proofs and more details. We can also use alternative distances as its basis.

Wasserstein distance in a discrete framework. It is quite common that the considered measures are discrete measures with finite supports. For two discrete measures $\mathsf{P} = \sum_{i=1}^{n} p_i\delta_{x_i}$ and $\mathsf{P}' = \sum_{j=1}^{n'} p'_j\delta_{x'_j}$, where $\{x_1,\ldots,x_n\}$ and $\{x'_1,\ldots,x'_{n'}\}$ are the corresponding finite supports, the Wasserstein distance is equal to

$$\begin{aligned}
&\min_{\pi} \sum_{i,j} \pi_{i,j}\cdot d_{i,j}^r \\
&\text{s.t.} \sum_{j=1}^{n'} \pi_{i,j} = p_i,\ i = 1,\ldots,n, \\
&\quad \sum_{i=1}^{n} \pi_{i,j} = p'_j,\ j = 1,\ldots,n', \\
&\quad\quad \pi_{i,j} \geq 0,
\end{aligned}$$

where the matrix $d_{i,j} = \mathtt{d}(x_i, x'_j)$ carries the distances between each pair (x_i, x'_j). The matrix $\pi_{i,j}$ corresponds to the bivariate probability measure

$$\pi = \sum_{i,j} \pi_{i,j}\cdot\delta_{(x_i,x'_j)}.$$

It holds that $\pi_{i,j} \geq 0$ and $\sum_i\sum_j \pi_{i,j} = 1$.

1.3.3 Multistage generalization

So far we defined the Wasserstein distance only for problems with one stage. In this subsection we extend it for stochastic processes $(\xi_t, t = 0, \dots, T)$ and $(\xi'_t, t = 0, \dots, T)$ with stochastic bases $(\Omega, \mathcal{F} = (\mathcal{F}_0, \dots, \mathcal{F}_T), \mathsf{P})$ and $(\Omega', \mathcal{F}' = (\mathcal{F}'_0, \dots, \mathcal{F}'_T), \mathsf{P}')$.

Consider now a stochastic process

$$\xi_t : \Omega \to (\Xi_t, \mathtt{d}_t),\ t = 0, \dots, T,$$

or in the compound form

$$\begin{aligned} \xi : \Omega &\to \Xi_0 \times \cdots \times \Xi_T \\ \omega &\mapsto (\xi_0(\omega), \dots, \xi_T(\omega)), \end{aligned}$$

where the spaces $(\Xi_t, \mathtt{d}_t)$ do not need to be the same for every $t = 1, \dots, T$ and ξ_0 is considered deterministic. If we take another process ξ' with the same state spaces Ξ_t, we can define the inherited distance analogously to the single-stage one. For instance, the ℓ^1 distance:

$$\mathtt{d}(\omega, \omega') := \sum_{t=0}^{T} w_t\, \mathtt{d}_t(\xi_t(\omega), \xi'_t(\omega')),\ w_t > 0\ \forall t,$$

the ℓ^2 distance:

$$\mathtt{d}(\omega, \omega') := \left(\sum_{t=0}^{T} w_t\, \mathtt{d}_t(\xi_t(\omega), \xi'_t(\omega'))^2 \right)^{\frac{1}{2}},\ w_t > 0\ \forall t,$$

or the ℓ^∞ distance:

$$\mathtt{d}(\omega, \omega') := \max_{t=0,\dots,T} w_t\, \mathtt{d}_t(\xi_t(\omega), \xi'_t(\omega')),\ w_t > 0\ \forall t.$$

For any choice of $\mathtt{d}$ there is a cost function on $\Omega \times \Omega'$ so that the Wasserstein distance

$$\mathtt{d}_r(\mathsf{P}^\xi, \mathsf{P}^{\xi'})$$

could be attained.

The main setback of the Wasserstein distance is that it does not take into account the conditional probabilities; it only considers the final probabilities. That is why an extension is made, which considers all σ-algebras included in the filtration, not only the final one.

Definition. Let $\mathbb{P} = (\Omega, (\mathcal{F}_t), \mathsf{P})$ and $\mathbb{P}' = (\Omega', (\mathcal{F}'_t), \mathsf{P}')$ be two probability spaces and $\mathtt{d} : \Omega \times \Omega' \to \mathbb{R}$ a distance defined on these spaces. We then define the *nested distance of order* r $(r \geq 1)$ as the optimal value of the optimization problem

$$\begin{aligned} \min_{\pi} & \left(\int \mathtt{d}(\omega, \omega')^r d\pi(\omega, \omega') \right)^{\frac{1}{r}} \\ \text{s.t.} \quad & \pi(A \times \Omega' |\, \mathcal{F}_t \otimes \mathcal{F}'_t) = \mathsf{P}(A |\, \mathcal{F}_t),\ A \in \mathcal{F}_t, t \in \{1, \dots, T\}, \\ & \pi(\Omega \times B |\, \mathcal{F}_t \otimes \mathcal{F}'_t) = \mathsf{P}'(B |\, \mathcal{F}'_t),\ B \in \mathcal{F}'_t, t \in \{1, \dots, T\}, \end{aligned} \tag{1.1}$$

where π is a bivariate probability measure from $\mathcal{P}(\Omega \times \Omega')$ defined on $\mathcal{F}_T \otimes \mathcal{F}'_T$. A feasible measure π is called a *nested transport plan.* We denote the optimal value by

$$\mathbb{d}_r(\mathbb{P}, \mathbb{P}').$$

Again, it could be proofed that the nested distance is both monotone and convex and that it satisfies the triangle inequality. See [7] for more details.

1.3.4 The nested distance for trees

For trees, we can reformulate the problem 1.1 as

$$\begin{aligned} \min_{\pi} & \sum_{i,j} \pi_{i,j} \cdot d_{i,j}^r \\ \text{s.t.} \quad & \sum_{j \succ l} \pi(i,j|k,l) = \mathsf{P}(i|k),\ \forall k \prec i,\ l, \\ & \sum_{i \succ k} \pi(i,j|k,l) = \mathsf{P}'(j|l),\ \forall l \prec j,\ k, \\ & \pi_{i,j} \geq 0,\ \forall i,j, \\ & \sum_{i,j} \pi_{i,j} = 1, \end{aligned}$$

where $\pi = (\pi_{i,j})$ is a matrix on the leaf nodes $i \in \mathcal{K}_T$, $j \in \mathcal{K}'_T$, k, l are arbitrary nodes of the same stage t, the notation $i \succ k$ signifies that the node k is a predecessor of the node i, and

$$\pi(i,j|k,l) = \frac{\pi_{i,j}}{\sum_{i' \succ k, j' \succ l} \pi_{i',j'}}$$

are the conditional probabilities. Note that many constraints are linearly dependent.

Since we can replace the conditions

$$\pi(A \times \Omega' | \mathcal{F}_t \otimes \mathcal{F}'_t) = \mathsf{P}(A | \mathcal{F}_t),\ \forall A \in \mathcal{F}_T$$

in 1.1 by

$$\pi(A \times \Omega' | \mathcal{F}_t \otimes \mathcal{F}'_t) = \mathsf{P}(A | \mathcal{F}_t),\ \forall A \in \mathcal{F}_{t+1}$$

(see [7], Lemma 2.43), the problem could be again rewritten considering the conditional probabilities only at subsequent stages.

Algorithm. For two tree processes the nested distance can be computed by the backward iteration. First, we compute the distance between the scenarios of the first tree and the scenarios of the second one, i.e., for each pair of leaf nodes $i \in \mathcal{K}_T$ and $j \in \mathcal{K}'_T$ we compute

$$\mathbb{d}_T^r(i,j) := \mathsf{d}((\xi_{i_0}, \xi_{i_1}, \ldots, \xi_i), (\xi'_{j_0}, \xi'_{j_1}, \ldots, \xi'_j))^r,$$

where $(i_0, \ldots, i_{T-1})$ and $(j_0, \ldots, j_{T-1})$ are the predecessors of the leaves i and j, respectively.

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

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

where $k+$ are the direct successors (children) of the node k and q_k are the conditional probabilities.

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

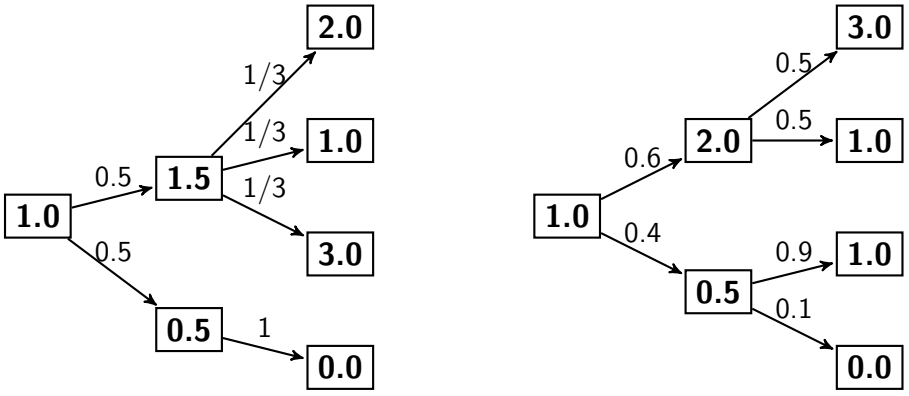
$$\mathbb{d}_r(\mathbb{P}, \mathbb{P}') = \mathbb{d}_0^r(1,1).$$

The optimal transport plan at the leaf nodes $i \in \mathcal{K}_T$ and $j \in \mathcal{K}'_T$ is the product

$$\pi(i,j) := \pi_1(i_1, j_1|i_0, j_0) \cdots \pi_{T-1}(i, j|i_{T-1}, j_{T-1}),$$

where π_t is the optimal transport plan at the stage t.

Example. We will compute the nested distance between the two trees which are depicted in Figure 1.4 below.



tree 1 tree 2

Figure 1.4: The trees considered in the example

We start with computing the distances between the leaf nodes. If we use the ℓ^1 distance, we get:

$$\mathbb{d}_2^1 = \begin{pmatrix} 1.5 & 1.5 & 2.0 & 3.0 \\ 2.5 & 0.5 & 1.0 & 2.0 \\ 0.5 & 2.5 & 3.0 & 4.0 \\ 4.5 & 2.5 & 1.0 & 0 \end{pmatrix}.$$

Now we solve the optimization problem for each combination of the nodes of the stage 1. We get:

$$\mathbb{d}_1^1 = \begin{pmatrix} 0.833 & 2.100 \\ 3.500 & 0.900 \end{pmatrix}.$$

We continue with the stage 0, i.e., the roots. We solve the optimization problem, and finally we get the nested distance between the tree 1 and the tree 2

$$\mathrm{d}_0^1(1,1) = \mathtt{d}_1(\mathbb{P}, \mathbb{P}') = 1.1267.$$

Obviously, the nested distance can be computed between trees which have different branching, different number of scenarios, or different number of nodes. However, both trees need to have the same number of stages.

2. Scenario reduction algorithms

As was mentioned earlier, in many situations we have a stochastic process represented by a huge tree, and we would like to reduce (approximate) the tree by a smaller (simpler) one, which we can use to solve some optimization problems. There are many scenario reduction algorithms in the literature; in this chapter we describe the most frequently used.

In the first section, we introduce algorithms which are based on a random extraction of nodes, and which do not consider the nodal values at all. Such algorithms, we call them *simple* in this thesis, are easy to implement and fast, but the quality of the result is uncertain. For that reason we introduce other algorithms, we call them *advanced*, in the second section of this chapter. These algorithms use different techniques based on the distances between the scenarios, the nodes, or the subtrees to merge the closest pair or to eliminate the furthest one. They are, naturally, more computationally demanding than the simple algorithms, but we expect a better approximation of the initial tree. We dedicate Chapter 3 of this thesis to study how much better is the approximation of the advanced algorithms compared to the simple algorithms in relation to the increase in the computational complexity.

Since it is best to show the algorithms on an example, we consider the following master tree:

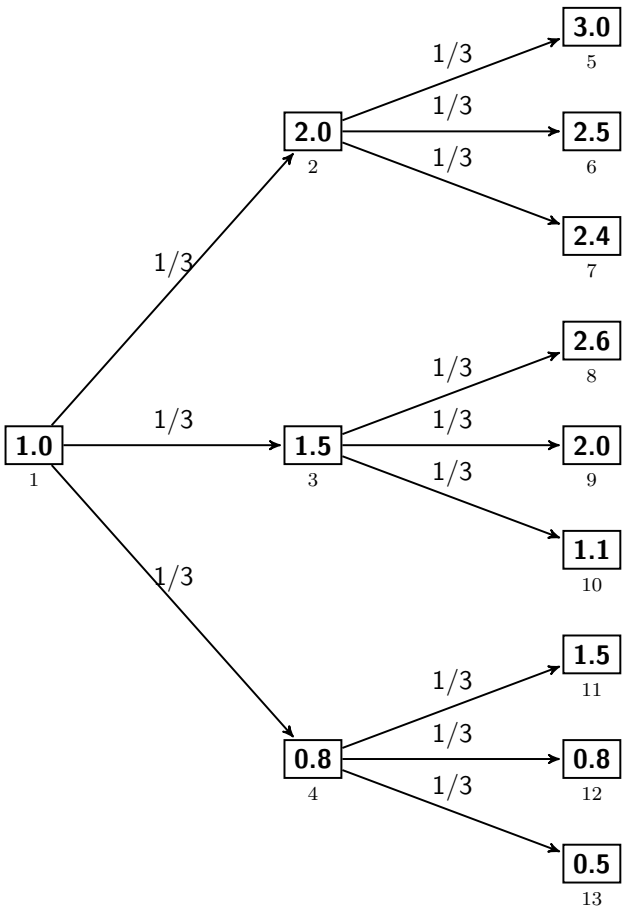


Figure 2.1: The master tree considered throughout Chapter 2

As we can see, it is a regular two-stage tree with branching 3-3 and it has nine equiprobable scenarios and thirteen nodes. In the following subsections we assume that we want to reduce it to a tree with four scenarios using alternative reduction algorithms.

2.1 Simple algorithms

We begin with the algorithms that do not consider the nodal values and are purely based on a random extraction of nodes. These algorithms do not compute any distances. They usually simply define an uniform distribution on the nodal labels (or, if the nodes are not equiprobable, they define a discrete probability distribution with the nodes being its support and the nodal probabilities the probabilities) and randomly select from the nodes. Therefore, they are very easy to implement and one can get a reduced tree in less than a second. Obviously, all of these algorithms are not deterministic, i.e., if run multiple times, they produce different reduced trees.

2.1.1 Nodal extraction

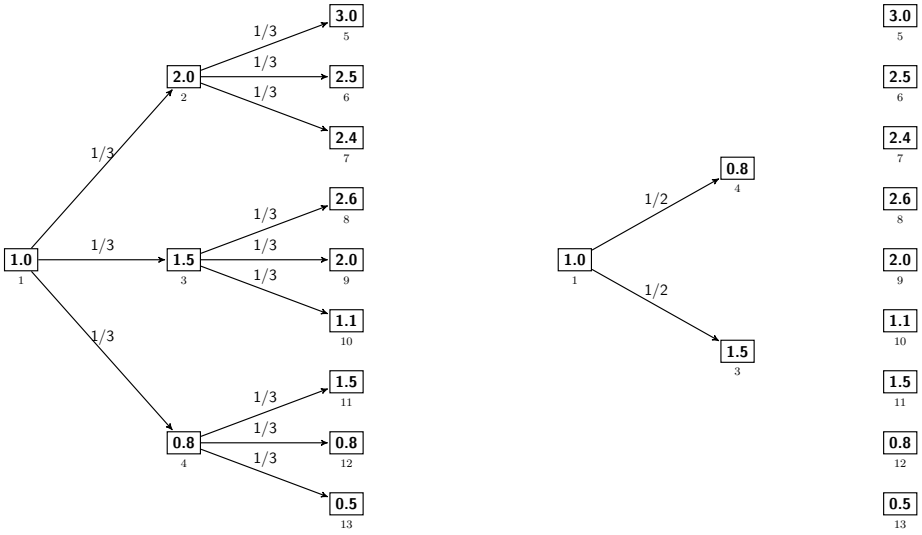
The first algorithm we propose is to simply choose the required tree structure, then at each stage define a discrete distribution on the nodes of that stage of the master tree, and randomly select from them.

Example. Consider the master tree defined at the beginning of this section (Figure 2.1). We would like to reduce it to a tree with four scenarios and regular branching 2-2. We divide the node labels according to the stages. We get the following sets

$$\begin{aligned}\mathcal{K}_0 &= \{1\},\\ \mathcal{K}_1 &= \{2,3,4\}, \text{ and}\\ \mathcal{K}_2 &= \{5,\ldots,13\}.\end{aligned}$$

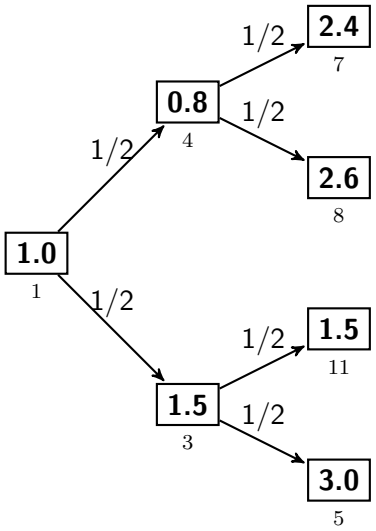
The desired number of nodes at each stage is 1, 2, and 4. We take the root node, and then randomly select 2 nodes from $\mathcal{K}_1$ and 4 nodes from $\mathcal{K}_2$. In this particular case we chose the nodes 4 and 3 from the stage 1, and the nodes 7, 8, 11 and 5 from the stage 2.

Note that the order of the nodes is important, since we connect the nodes to the nodes of the previous stage according to that order, i.e., in our case we connect the nodes 4 and 3 with the root (we set the root as the parent of the nodes 4 and 3), the nodes 7 and 8 with the node 4, and the nodes 11 and 5 with the node 3. The reduced tree is in Figure 2.2 below.



(a) Master tree

(b) Step 1



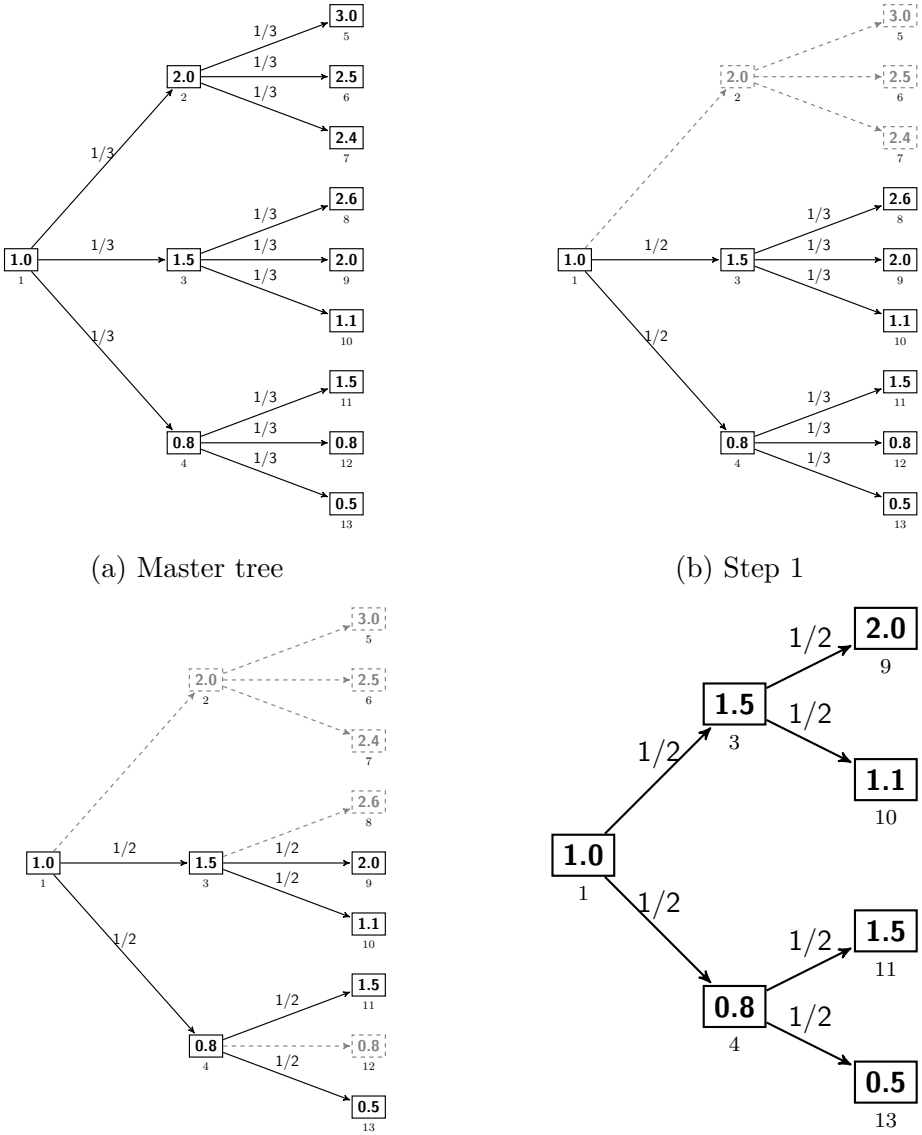
(c) Step 2

Figure 2.2: A reduced tree using the nodal extraction

2.1.2 Nodal extraction improved

One way to improve the algorithm described in the previous part (the nodal extraction) is to select nodes only from the children of the already selected nodes from the previous stage instead of selecting from all the nodes from the stage. This approach seems to be a bit smarter as it somehow preserves the original parent-child relations.

Example. Let us start from the master tree 2.1 again. We want to reduce it to a tree with 4 scenarios regularly branched 2-2. We take the root and randomly select 2 of its children $1+ = \{2, 3, 4\}$. In this particular case we get the nodes 3 and 4. Then we take the node 3 and randomly select from its children $3+ = \{8, 9, 10\}$. We do the same with the node 4. We get the nodes 9, 10 and 11, 13. The reduced tree is in Figure 2.3.



(a) Master tree

(b) Step 1

(c) Step 2

(d) A reduced tree

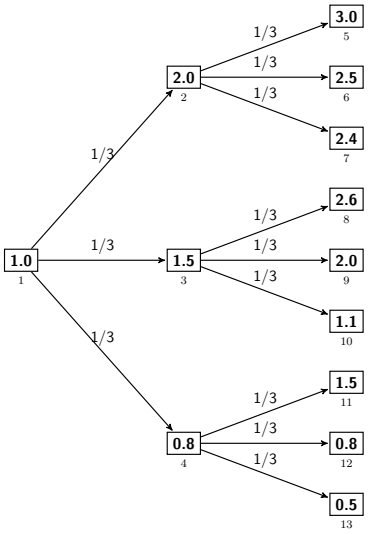
Figure 2.3: A reduced tree using the improved nodal extraction

2.1.3 Scenario extraction

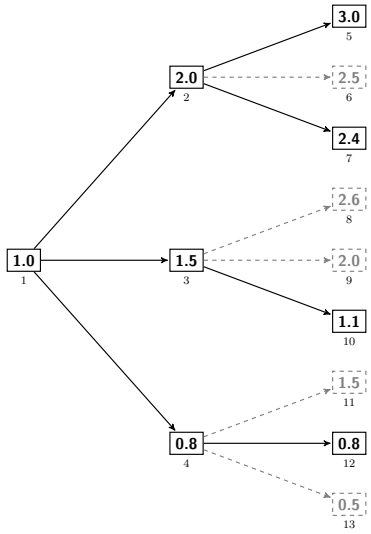
We might not want to determine any specific structure for the reduced tree. In that case we can simply randomly select the desired number of scenarios. We define a discrete distribution on the leaf nodes and randomly select the number of leaf nodes according to the number of scenarios we want. The reduced tree then contains the whole paths to the selected leaf nodes. We set the probabilities of the leaves so that the former ratios are preserved, but they sum up to 1. For example, if we select three leaf nodes with probabilities 0.2, 0.1 and 0.3, the new probabilities would be 1/3, 1/6 and 1/2. The probabilities of the remaining nodes are computed recursively as the sum of the probabilities of their children.

By this approach, the reduced tree is generally larger than the trees generated by the two algorithms described earlier. The reason is that, especially when reducing from very large trees, many nodes have only one child.

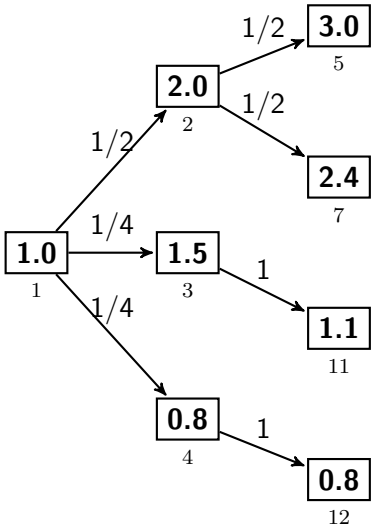
Example. Let us consider the master tree 2.1. We take the leaf nodes of the master tree (labeled $\{5, \ldots, 13\}$) and randomly select 4 of them. In this particular case we selected the nodes 5, 7, 10, and 12. We preserve the whole paths to the leaf nodes, i.e., all the nodes 2, 3, and 4. Since we have equiprobable nodes, we set the probabilities of the new leaf nodes equally to 1/4. Then we compute the probabilities of their parents – the node 2 has the probability 1/2, the nodes 3 and 4 have equally the probability 1/4. We compute the conditional probabilities from the absolute ones in the way that was shown in Subsection 1.2.1 to finally get the reduced tree, which is shown in Figure 2.4.



(a) Master tree



(b) Step 1



(c) A reduced tree

Figure 2.4: A reduced tree using the scenario extraction

2.2 Advanced algorithms

Since the algorithms described in the preceding section 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 that reason, we would like to think of a way how to use that information and the information about the tree structure in a cleverer way. In this section, we introduce three different approaches to the problem.

2.2.1 Single scenario reduction

The single scenario reduction, as the name suggests, reduces the master tree scenario by scenario. The idea is to iterate the procedure until the desired number of scenarios is reached. This method is in detail described in [2]. The outline of its general version is described at the end of this subsection.

First, we measure the distances between all scenarios. We get a matrix with the elements

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

where $(i_0, \dots, i_T)$ and $(j_0, \dots, j_T)$ are the paths of the scenario i and j, respectively.

Next, for each scenario i we define

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

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

Lastly, we find the scenario which was the closest to the eliminated one and increase its probability by the probability of the eliminated one, 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 with the new tree until we have the desired number of scenarios.

Example. We consider the master tree 2.1. We measure the scenario distances and get

$$d = \begin{pmatrix} - & 0.5 & 0.6 & 0.9 & 1.5 & 2.4 & 2.7 & 3.4 & 3.7 \\ 0.5 & - & 0.1 & 0.6 & 1.0 & 1.9 & 2.2 & 2.9 & 3.2 \\ 0.6 & 0.1 & - & 0.7 & 0.9 & 1.8 & 2.1 & 2.8 & 3.1 \\ 0.9 & 0.6 & 0.7 & - & 0.6 & 1.5 & 1.8 & 2.5 & 2.8 \\ 1.5 & 1.0 & 0.9 & 0.6 & - & 0.9 & 1.2 & 1.9 & 2.2 \\ 2.4 & 1.9 & 1.8 & 1.5 & 0.9 & - & 1.1 & 1.0 & 1.3 \\ 2.7 & 2.2 & 2.1 & 1.8 & 1.2 & 1.1 & - & 0.7 & 1.0 \\ 3.4 & 2.9 & 2.8 & 2.5 & 1.9 & 1.0 & 0.7 & - & 0.3 \\ 3.7 & 3.2 & 3.1 & 2.8 & 2.2 & 1.3 & 1.0 & 0.3 & - \end{pmatrix}.$$

Then we compute the D and get

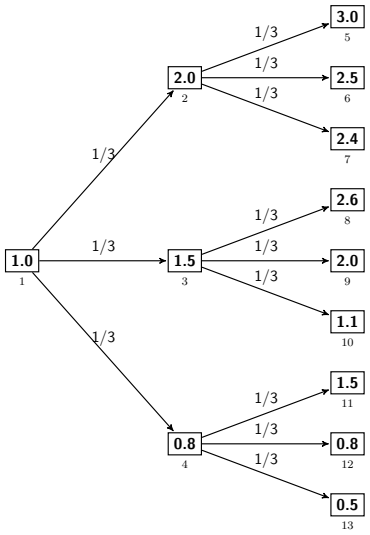
$$D = \frac{1}{9} \cdot (0.5, 0.1, 0.1, 0.6, 0.6, 0.9, 0.7, 0.3, 0.3).$$

D is the lowest at two scenarios – the scenarios 2 and 3 represented by the paths (1,2,6) and (1,2,7). We eliminate the first one, i.e., the scenario 2. Then we find the closest scenario, which in this case is the scenario 3, and increase its probability to $p_3 = p_3 + p_2 = \frac{2}{9}$.

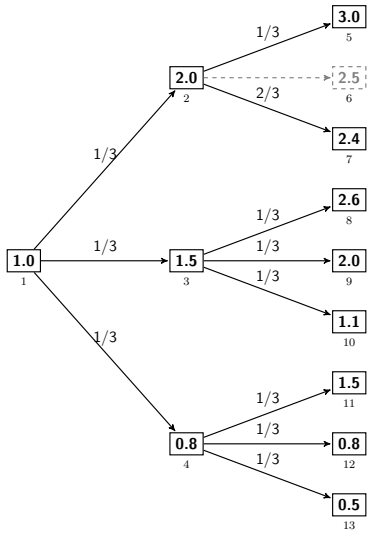
We recalculate the D for the reduced tree:

$$D = \frac{1}{9} \cdot (0.6, 2 \cdot 0.6, 0.6, 0.6, 0.9, 0.7, 0.3, 0.3).$$

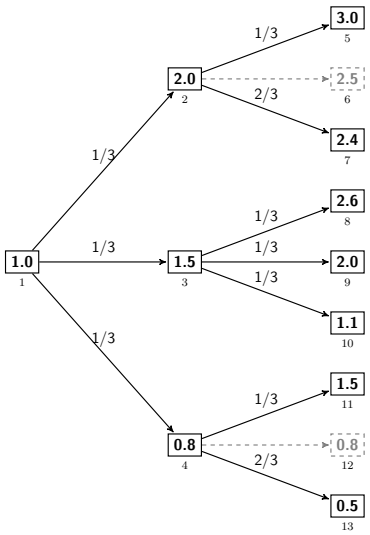
The $\arg\min D$ again contains two scenarios. We eliminate the first one, i.e., the one represented by the path (1,4,12), and readjust the probabilities. We iterate until we get the final tree (Figure 2.5).



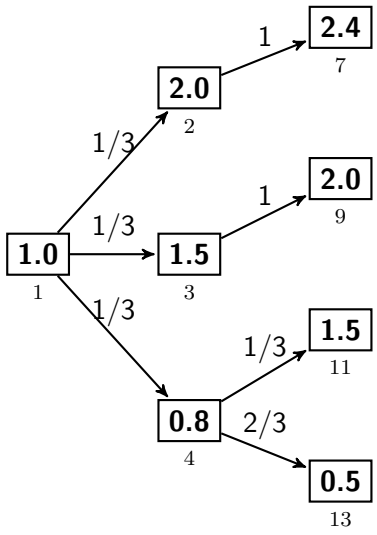
(a) Master tree



(b) Step 1



(c) Step 2



(d) Step 5 - the reduced tree

Figure 2.5: The reduced tree using the single scenario reduction

Generalization. We can generalize this method to a multiple scenario reduction. Suppose we want to eliminate k scenarios. We define

$$D_J = \sum_{i \in J} p_i \min_{j \notin J} d_{i,j},$$

where p_i is the probability of the scenario i, $d_{i,j}$ is the distance between the scenarios i and j, and $J \subset \{1, \ldots, S\}$, $\#J = k$, where S is the total number of scenarios. We eliminate the k scenarios from

$$J^* = \arg\min\{D_J : J \in \{1, \ldots, S\},\ \#J = k\}.$$

As in the single scenario reduction, we redistribute the probabilities to the non-eliminated ones. We iterate until the desired number of scenarios is reached. In each iteration, k could be different.

The apparent disadvantage of this method is that it is quiet huge and laborious. It is practically impossible to apply it on larger trees. Aside from this, it uses absolute probabilities only and does not consider the tree structure. As well as in the case of the scenario extraction described in Subsection 2.1.3, the reduced tree contains many nodes with only one child.

2.2.2 The clustering algorithm

In this part we introduce an algorithm based on *clustering*. Clustering is a way of grouping n points to $s < n$ groups, which are named *clusters*. There are many possible ways of clustering described in the literature (see [4]), however, in this thesis we consider the following version. 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 to 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

$$\mathtt{d}(A, B) = \max\{\mathtt{d}(a_i, b_j) : i \in \{1, \ldots, k\}, j \in \{1, \ldots, l\}\}, \tag{2.1}$$

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

Alternatively, we could represent each cluster by its centroid and compute the cluster distances as the distances between their centroids. Or, 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.

Now, we need to apply it to a tree. The idea is to determine the target structure of the reduced tree, and then proceed stage after stage. The algorithm is described below.

Algorithm. We choose the final (regular) structure of the reduced tree. When this is settled, we set $\mathcal{K}_0^* := \mathcal{K}_0$. Then for $t = 0$ to $T - 1$ and for each node n from $\mathcal{K}_t^*$ we cluster its children into the predetermined number of groups. As the

representative of each group we take the group's median, and we store its label in $\mathcal{K}^*_{t+1}$. Its probability is equal to the sum of the probabilities of all members of the group, its children are all children of all members of the group.

The reduced tree is represented by the nodes in $\mathcal{K}^*_t,\ t = 0, \dots, T$. It has the required structure, but its scenarios do not need to be equiprobable. This is the consequence of 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.

Example. Let us start with the master tree 2.1. We would like to reduce it to a tree with regular branching 2-2. We follow the algorithm and set $\mathcal{K}^*_0 := \mathcal{K}_0 = \{1\}$. We take its children $\{2, 3, 4\}$ and compute the ℓ^1 distances between each pair to get the matrix

$$d = \begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 4 \end{array} \\ \begin{pmatrix} - & 0.5 & 1.2 \\ 0.5 & - & 0.7 \\ 1.2 & 0.7 & - \end{pmatrix} \end{array}.$$

The closest nodes are the nodes 2 and 3, so we join them into one cluster. We have 2 clusters $\{2, 3\}$ and $\{4\}$, which is the number of clusters we wanted. For the first cluster we take the median (= 1.75), label it 2^*, and store it in $\mathcal{K}^*_1$ along with 4. The node 4 has the same probability as before, the probability of the node 2^* is $p_2 + p_3 = \frac{2}{3}$.

We move to the next stage. We take the children of the node 2^* (the nodes $\{5, \dots, 10\}$) and compute the distances between each pair. We get

$$d = \begin{array}{c} \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 5 & 6 & 7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} - & 0.5 & 0.6 & 0.4 & 1.0 & 1.9 \\ 0.5 & - & 0.1 & 0.1 & 0.5 & 1.4 \\ 0.6 & 0.1 & - & 0.2 & 0.4 & 1.3 \\ 0.4 & 0.1 & 0.2 & - & 0.6 & 1.5 \\ 1.0 & 0.5 & 0.4 & 0.6 & - & 0.9 \\ 1.9 & 1.4 & 1.3 & 1.5 & 0.9 & - \end{pmatrix} \end{array}.$$

The closest are the nodes 6, 7 and 6, 8. We join the first pair into one cluster $\{6, 7\}$ and recalculate the distances. We get

$$d = \begin{array}{c} \\ 5 \\ 6,7 \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{ccccc} 5 & 6,7 & 8 & 9 & 10 \end{array} \\ \begin{pmatrix} - & 0.6 & 0.4 & 1.0 & 1.9 \\ 0.6 & - & 0.2 & 0.5 & 1.4 \\ 0.4 & 0.2 & - & 0.6 & 1.5 \\ 1.0 & 0.5 & 0.6 & - & 0.9 \\ 1.9 & 1.4 & 1.5 & 0.9 & - \end{pmatrix} \end{array}.$$

We join the group $\{6, 7\}$ with the node 8 into one. The recalculated distances then are

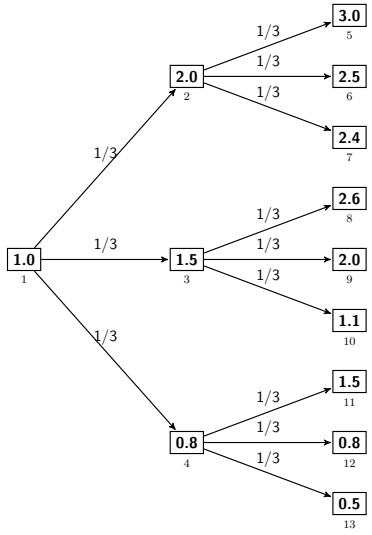
$$d = \begin{array}{c} \\ 5 \\ 6,7,8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccc} 5 & 6,7,8 & 9 & 10 \end{array} \\ \begin{pmatrix} - & 0.6 & 1.0 & 1.9 \\ 0.6 & - & 0.6 & 1.5 \\ 1.0 & 0.6 & - & 0.9 \\ 1.9 & 1.5 & 0.9 & - \end{pmatrix} \end{array}.$$

We add the node 5 into the big group. The distances are

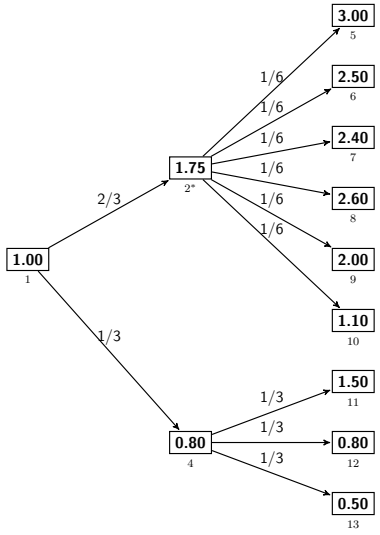
$$d = \begin{array}{c} \\ 5,6,7,8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{ccc} 5,6,7,8 & 9 & 10 \end{array} \\ \begin{pmatrix} - & 1.0 & 1.9 \\ 1.0 & - & 0.9 \\ 1.9 & 0.9 & - \end{pmatrix} \end{array}.$$

Now we join the nodes 9 and 10 into one group. Finally, we have two clusters. The group medians are 2.55 and 1.55. We store them under the labels 5^* and 9^* into $\mathcal{K}_2^*$. Their conditional probabilities are $\frac{2}{3}$ and $\frac{1}{3}$.

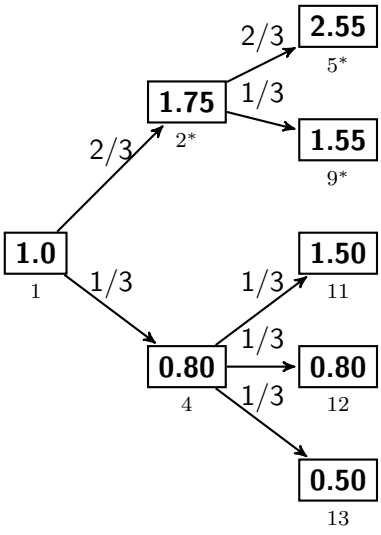
We do the same with the other node of $\mathcal{K}_1^*$. We get two points (1.5 and 0.65) labeled 11 and 12^*, which have the conditional probabilities $\frac{1}{3}$ and $\frac{2}{3}$. The final reduced tree is in Figure 2.6.



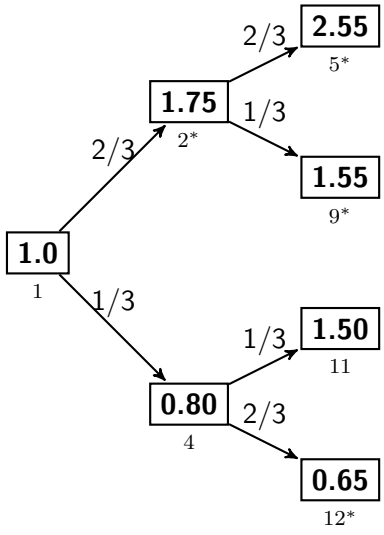
(a) Master tree



(b) Step 1



(c) Step 2



(d) Step 3 - the reduced tree

Figure 2.6: The reduced tree using the clustering algorithm

2.2.3 Subtrees merging

One may go one step further and use the nested distance between subtrees instead of the distance between points as the basis of the reduction. The algorithm which we will now describe is, along with its improvements, presented in [7].

We start with computing the nested distances between all non-degenerate subtrees which have the same parent. Then we merge the closest pair into one using the following algorithm.

Merging. Suppose we have two subtrees with distributions $\mathbb{P}_1$ and $\mathbb{P}_2$, which we want to merge into a new tree. The value at the new root is the mean value of the two old roots. For the successors, consider the optimal transport plan between $\mathbb{P}_1$ and $\mathbb{P}_2$, and sort it in descending order, i.e., if the roots are i_0 and j_0, respectively, then we get

$$\pi_{i_1,j_1} \geq \pi_{i_2,j_2} \geq \cdots ,$$

where $i_1, i_2, \ldots \in i_0+$, $j_1, j_2, \ldots \in j_0+$. Then we take the smallest m such that

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

where p is a chosen reducing parameter, $p \in (0, 1]$. The smaller the p, the bigger the reduction.

We have m points $(i_1, j_1), \ldots, (i_m, j_m)$. We set their values as the mean values of the corresponding values of the original two trees, i.e., if the values of the first tree are represented by ξ and the values of the second by ξ', we take $\frac{1}{2}(\xi_{i_k} + \xi'_{j_k})$, $k = 1, \ldots, m$. Their probabilities are

$$p_k = \frac{\pi_{i_k,j_k}}{\sum_{l=1}^{m} \pi_{i_l,j_l}}, \ k = 1, \ldots, m,$$

so that it holds $\sum_{k=1}^{m} p_k = 1$. Now we proceed recursively by merging the points $(i_1, j_1), \ldots, (i_m, j_m)$, i.e., by merging the subtrees which have the nodes i_k and j_k, $k = 1, \ldots, m$, as their roots. When the leaf level is reached, only the mean value is taken.

The algorithm stops if the reduced tree is small enough. Otherwise, we recalculate the nested distances and merge the closest pair.

The reduction of the original tree strongly depends on the choice of the reducing parameter p, which could be a problem, especially if we are aiming for a tree with a specific number of scenarios.

Example. Let us have the master tree 2.1. We compute the nested distances of order one. For the stage 1 we get

$$d_1 = \begin{array}{c} \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} 2 & 3 & 4 \end{array} \\ \begin{pmatrix} - & 1.233 & 2.900 \\ 1.233 & - & 1.667 \\ 2.900 & 1.667 & - \end{pmatrix} \end{array}.$$

So, we merge the subtrees with the nodes 2 and 3 as roots. We set their mean value as a new node 2*, and for their children we look at the optimal transportation plan, which is

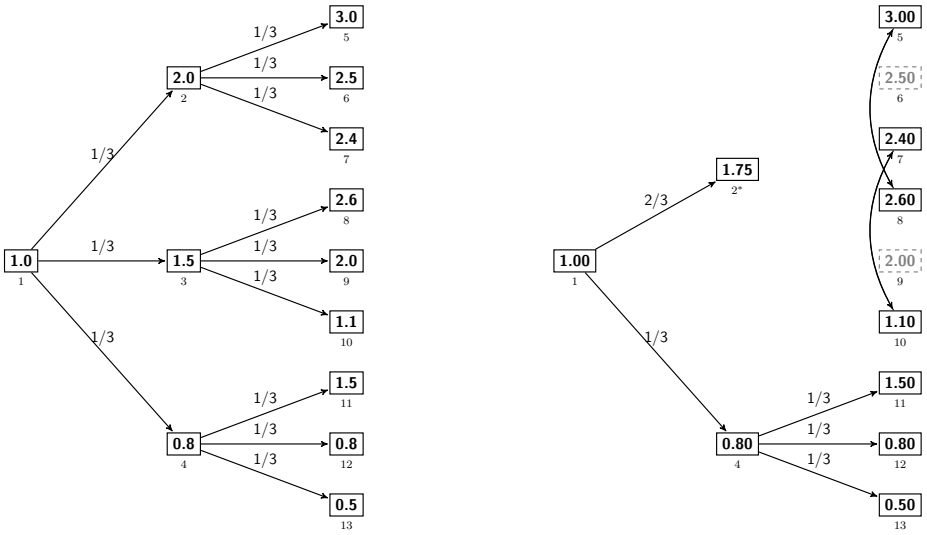
$$\pi = \begin{array}{c} \\ 8 \\ 9 \\ 10 \end{array} \begin{array}{c} \begin{array}{ccc} 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1/3 & 0 & 0 \\ 0 & 1/6 & 1/6 \\ 0 & 1/6 & 1/6 \end{pmatrix} \end{array}.$$

We sort π in descending order. Then it depends on the choice of p. For $p = 1$ we would have to merge all the points (5,8), (6,9), (6,10), (7,9), and (7,10). For $p = 0.5$ it suffices to merge only two points. Say we choose $p = 0.5$ and merge the points (5,8) and (7,10). We take their means and set them as new nodes labeled 5* and 7*, which we connect to the node 2* as its children. Their conditional probabilities are $\frac{1}{3} \cdot \frac{1}{2} = \frac{2}{3}$ and $\frac{1}{6} \cdot \frac{1}{2} = \frac{1}{3}$.

Now we have five scenarios, which is still too much, so we have to recompute the nested distances. Since we have only one possibility left, we merge the subtrees with the nodes 2* and 4 as roots. For their children we again follow the transportation plan

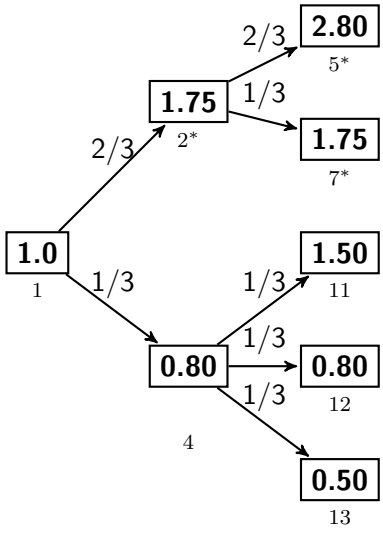
$$\pi = \begin{array}{c} \\ 11 \\ 12 \\ 13 \end{array} \begin{array}{c} \begin{array}{cc} 5^* & 7^* \end{array} \\ \begin{pmatrix} 2/9 & 1/9 \\ 2/9 & 1/9 \\ 2/9 & 1/9 \end{pmatrix} \end{array}$$

and merge the points (5*,11), (5*,12), and (5*,13) to finally get the reduced tree in Figure 2.7. However, this tree has only three scenarios, since 2 scenarios were removed in the last iteration for $p = 0.5$.

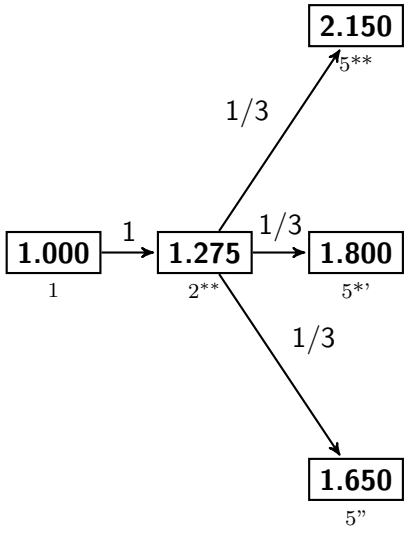


(a) Master tree

(b) Step 1



(c) Step 2



(d) Step 3 - the reduced tree

Figure 2.7: The reduced tree using the subtrees merging algorithm

For the choice $p = 1$ we obtain a tree with 15 scenarios and no possible subtrees to merge except for the degenerate ones (so the tree is actually larger than the master tree), for the choice $p = 0.8$ we get a tree with 9 scenarios, and for the choice $p = 0.6$ we get a tree with 5 scenarios. We see that even for this simple example the choice of p is very important, since each choice of p leads to a different solution.

If we allow the merging of the degenerate subtrees, i.e., the leaf nodes, we arrive to the tree depicted in Figure 2.8. We see that it has the desired number of scenarios.

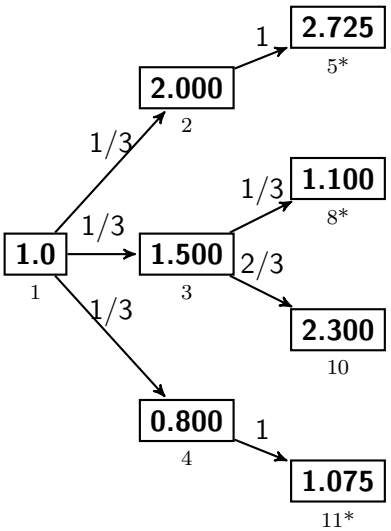


Figure 2.8: The reduced tree using the subtrees merging algorithm with allowed degenerate subtrees for p=1

2.3 Comparison of the algorithms

Let us now compare the trees we obtained by applying the algorithms described above.

The nested distances of order 1 between the reduced trees and the initial master tree are in Table 2.1 along with the number of nodes at each stage of the reduced trees.

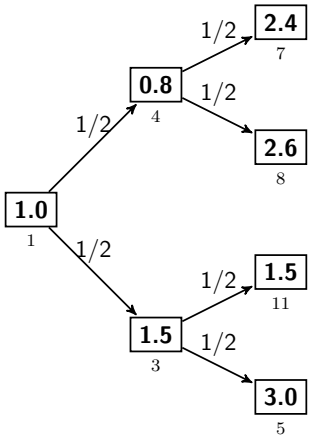
Algorithm	**The nested distance**	**Number of nodes per stage**
Nodal extraction	1.097	1-2-4
Nodal extraction improved	0.908	1-2-4
Scenario extraction	0.561	1-3-4
Single scenario reduction	0.278	1-3-4
Clustering alg.	0.467	1-2-4
Subtrees merging	1.175	1-1-3
Subtrees merging with degenerate subtrees	0.300	1-3-4

Table 2.1: Comparison of the reduced trees

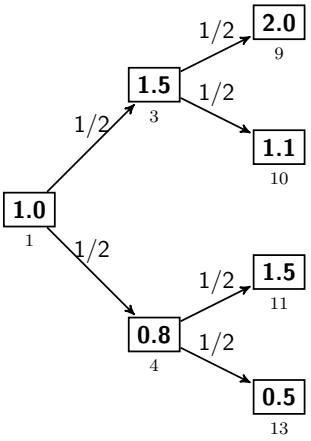
As we can see, the closest is the tree from the single scenario reduction, the furthest is the tree from the subtrees merging without degenerate subtrees, which has only three scenarios. Its second version, i.e., when the merging of the degenerate subtrees is allowed, is the second best.

But these reductions were just illustrative, so the distances are not very indicative. The algorithms are thoroughly compared in the next chapter, where more realistic (larger) master trees were generated.

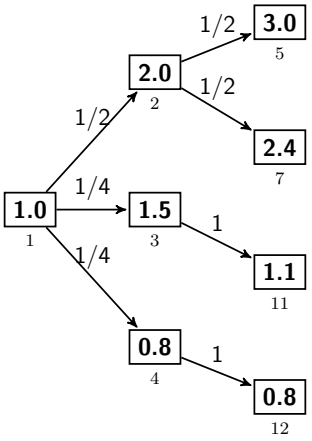
All reduced trees are shown in Figure 2.9.



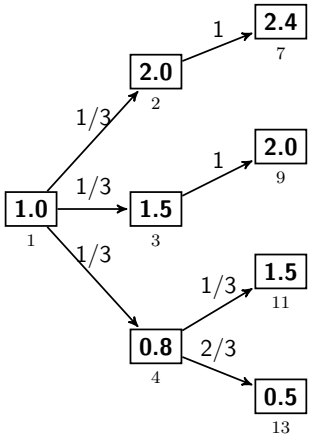
(a) Nodal extraction



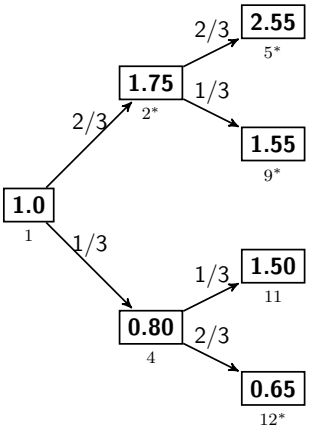
(b) Nodal extraction improved



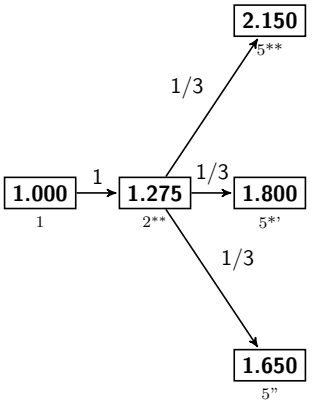
(c) Scenario extraction



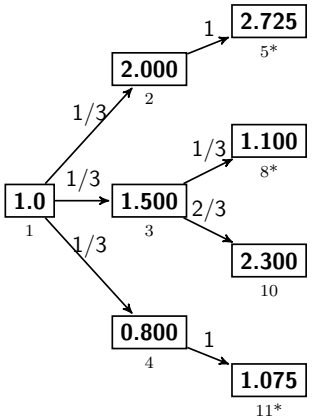
(d) Single scenario reduction



(e) Clustering algorithm



(f) Subtrees merging



(g) Subtrees merging with allowed degenerate trees

Figure 2.9: The final reduced trees from the example

3. Numerical study

In this chapter we apply the algorithms described in the preceding part to master trees. Then we compute the nested distances between the reduced trees and the master trees, and we compare the results. We also solve a simple portfolio selection problem, so we are able to compare the distances between the optimal objective values and solutions as well as the nested distances and the computational time.

Throughout the study, we used the nested distance of order one. We used the ℓ^1 norm to measure distances between points and scenarios, i.e.,

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

where T is the number of stages and N is the dimension of the values. In other words, N is the number of assets considered.

The subtrees merging algorithm is used with the reducing parameter p set to 0.5. Higher choices of p led to a significant increase in the number of nodes of the tree, which led to a significant increase in the computational time of the algorithm, and the algorithm eventually exceeded the time limit – one day.

3.1 Master tree generation

Based on real data, i.e., weekly returns of forty selected assets, we generated the following four-stage master trees:

(a) regularly branched 10-10-6-4 with 2,400 equiprobable scenarios,

(b) regularly branched 25-10-10-4 with 10,000 equiprobable scenarios.

We successively consider the cases using the weekly returns of 1, 5, 10, and 20 assets; we refer to these cases as dimensions 1, 5, 10, and 20 throughout the whole chapter, i.e., the dimension 1 means the case with 1-dimensional nodal values, the dimension 5 refers to the case with 5-dimensional nodal values (the case with 5 assets), etc.

For both master trees, the algorithms are reducing with the same ratio 100 to 1, i.e., the master tree with 10,000 scenarios is reduced to 100 scenarios, the master tree with 2,400 scenarios to 24.

We started by calculating the weekly returns of each asset over the last three years, i.e., from the beginning of 2015 to the beginning of 2018. For each asset we computed the sample mean and variance. We also computed the correlations between the assets. Based on the required structure and the number of assets (dimension of the nodal values) we then generated the master trees using the Monte Carlo sampling method assuming that the weekly returns of the assets distribute as a multivariate normal distribution.

Summary. To summarize the master trees:

- the number of stages: $T = 4$;
- the dimension of the nodal values (the number of assets): $N = 1, 5, 10, 20$;

- the number of nodes per stage:
  (a) 1 - 10 - 100 - 600 - 2,400;
  (b) 1 - 25 - 250 - 2,500 - 10,000;
- the target number of scenarios:
  (a) 24;
  (b) 100.

3.2 Distance of the trees

At first, we are going to compare the time needed to run each of the six algorithms from the preceding chapter and the nested distances between the reduced trees obtained by the algorithms and the master trees. All computation were done on a computer with Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz processor and 8GB memory. The time is measured in seconds.

3.2.1 2,400 to 24 reduction

Figure 3.1 shows the nested distances between the reduced trees and the master trees with 2,400 scenarios to the elapsed time of the algorithms, where we take the $\log_{10}$ of the time for better readability of the time differences between the algorithms.

As we can see, the simple algorithms generate a reduced tree in less than 0.1 second. The scenario extraction is slightly slower than the two nodal extractions, but, compared to the nodal extractions, it is closer to the master tree for the dimensions 5, 10, and 20 (i.e., to the master trees with 5-, 10-, and 20-dimensional nodal values).

The clustering algorithm needs 2 to 3 seconds to reduce the master tree. However, the reduced tree from this algorithm is the closest for all dimensions but the dimension 5, where it is beaten by the single scenario reduction.

The single scenario reduction takes over 900 seconds ($\sim$ 15 minutes) with comparable results to the clustering method, except for the dimension one, where it is significantly worse, but still usually better than the simple algorithms.

The subtrees merging algorithm is the slowest; it takes more than 19,500 seconds ($\sim$ 3 hours and 30 minutes) with a result comparable to the simple scenario extraction algorithm.

When we look at Table 3.1, which summarizes the nested distances, we see that the nested distance increases with the dimension of the nodal values of the master tree. Moreover, also the variance of the non-deterministic algorithms increases with the dimension. We see, however, that the variance for the scenario extraction is little smaller than for the nodal extractions.

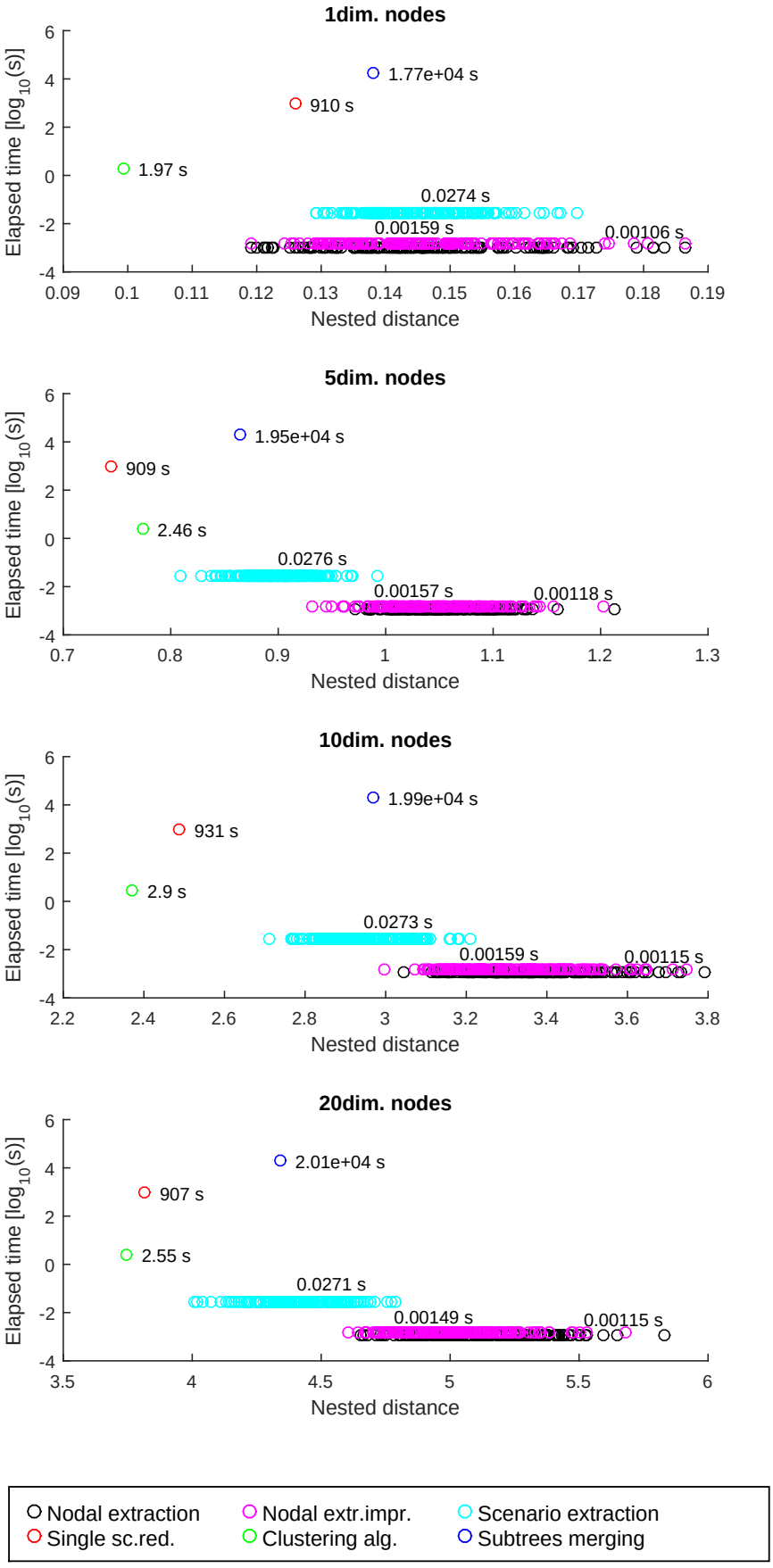


Figure 3.1: The nested distance of order 1 to the elapsed time for the 2,400 scenario master trees

The nested distance					
	Algorithm	Min.	Mean	Max.	SD
Dimension 1	Nodal extraction	0.119	0.146	0.186	0.014
	Nodal extraction improved	0.119	0.146	0.186	0.012
	Scenario extraction	0.129	0.146	0.170	0.008
	Single scenario reduction		0.126		
	Clustering alg.		0.099		
	Subtrees merging $p = 0.5$		0.138		
Dimension 5	Nodal extraction	0.972	1.054	1.213	0.042
	Nodal extraction improved	0.932	1.045	1.202	0.047
	Scenario extraction	0.808	0.900	0.992	0.030
	Single scenario reduction		0.745		
	Clustering alg.		0.774		
	Subtrees merging $p = 0.5$		0.864		
Dimension 10	Nodal extraction	3.043	3.367	3.792	0.146
	Nodal extraction improved	2.997	3.317	3.746	0.141
	Scenario extraction	2.710	2.945	3.209	0.099
	Single scenario reduction		2.488		
	Clustering alg.		2.372		
	Subtrees merging $p = 0.5$		2.969		
Dimension 20	Nodal extraction	4.655	5.111	5.830	0.224
	Nodal extraction improved	4.604	5.011	5.677	0.208
	Scenario extraction	4.010	4.406	4.785	0.155
	Single scenario reduction		3.816		
	Clustering alg.		3.746		
	Subtrees merging $p = 0.5$		4.342		

Table 3.1: The nested distance between the master trees with 2,400 scenarios and the reduced trees

3.2.2 10,000 to 100 reduction

For 10,000 to 100 reduction we adjusted the subtrees merging algorithm by determining the target number of nodes at each stage (in our case 5 for stage 1, 25 for stage 2, 50 for stage 3, and 100 for stage 4), and then started merging stage 1 until we reached the desired number of nodes at that stage. Then we moved to stage 2, and so on. Otherwise, the algorithm would have run longer than a week, which we think is a time period no one would want to wait. The resulted reduced trees are still non-regular.

The results are shown in Figure 3.2; the nested distances are summarized in Table 3.2.

By comparing it with the 2,400 to 24 reduction, the nodal extractions are still very fast, but here the distinction between them is more apparent – the improved nodal extraction generates trees which are slightly closer to the master trees than the trees generated by the unimproved nodal extraction. The scenario extraction takes little more than 0.2 seconds, and, except for the 1-dimensional case, its trees are closer to the master trees than are the trees of the two nodal extractions algorithms. Moreover, the variability of the nested distance for the nodal extractions is twice as big as the variability of the nested distance for the scenario extraction.

The clustering algorithm now takes 12 to 28 seconds, which is still a very good time, and its trees are the closest. The single scenario reduction is the slowest now; the time needed is 13 to 14.5 hours. For the dimensions 5 and 10 the results are comparable to the clustering algorithm, but it is not as good for the other dimensions, and it is even worse than the simple scenario extraction in the 20-dimensional case.

After the adjustment, the subtrees merging algorithm takes around 2.7 hours for the 1-dimensional nodes, 4.5 hours for the 5- and 10-dimensional nodes, and 7.5 hours in the 20-dimensional case. Its results are between the improved nodal extraction and the scenario extraction, which, when we look at the time needed, is not good.

If we use a different choice of the reducing parameter p in the adjusted subtrees merging algorithm, then even for the choice $p = 0.6$ the algorithm does not finish, as, although it reduces the number of nodes at stage 1, it enlarges the number of nodes at other stages so much that computing the nested distance between one pair of subtrees takes more than 5 hours and the time needed grows with each iteration of the algorithm. This would not happen in the original version of the algorithm with allowed degenerate subtrees, but the time needed would be more than a week.

The nested distances between the reduced trees and the master trees are comparable for both reductions, i.e., we obtained similar results for the 2,400 to 24 reduction and for the 10,000 to 100 reduction. The computational time of the algorithms, on the other hand, is higher for the 10,000 to 100 reduction.

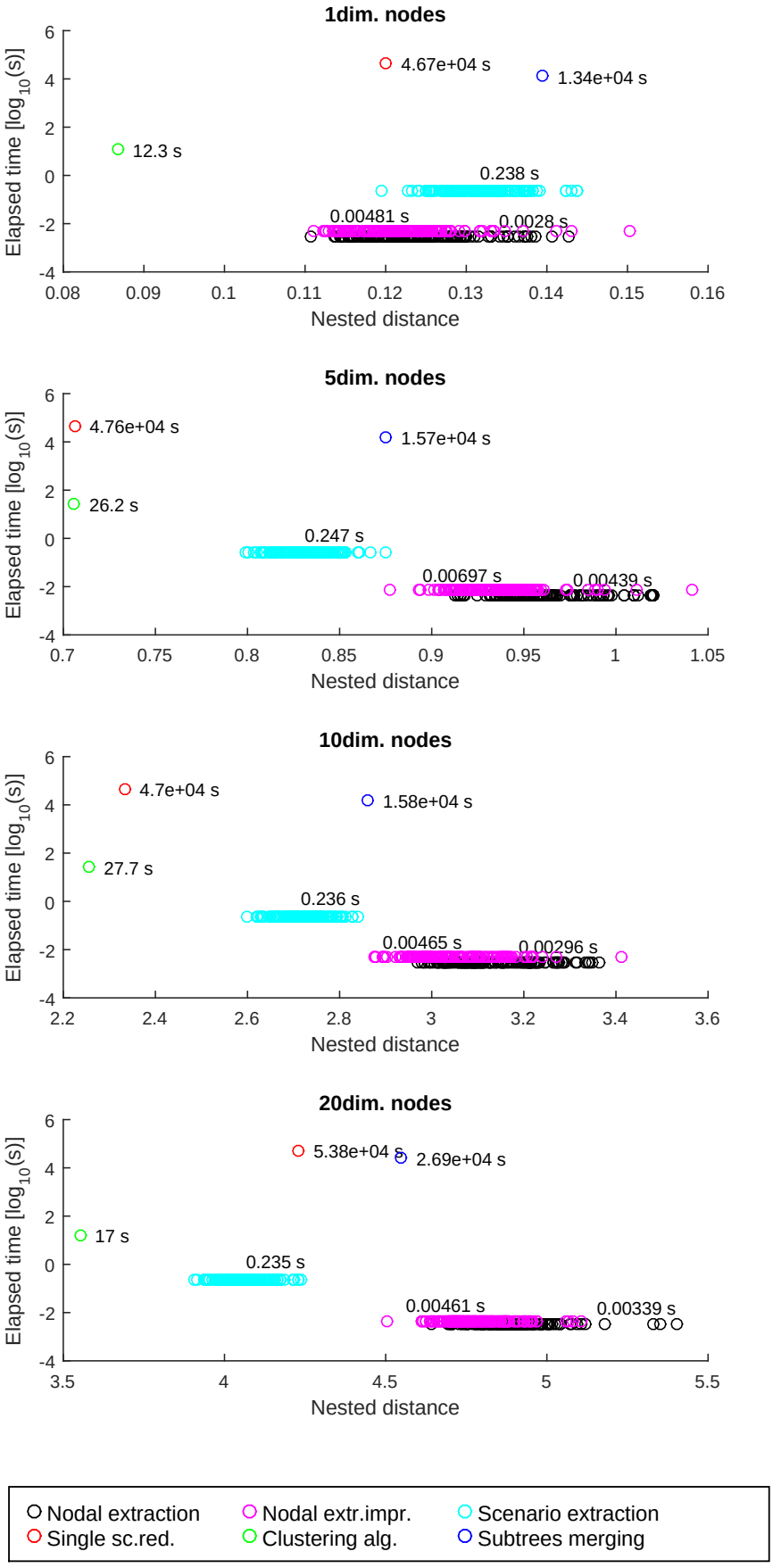


Figure 3.2: The nested distance of order 1 to the elapsed time for the 10,000 scenario master trees

The nested distance					
	Algorithm	Min.	Mean	Max.	SD
Dimension 1	Nodal extraction	0.111	0.124	0.143	0.006
	Nodal extraction improved	0.111	0.122	0.150	0.006
	Scenario extraction	0.119	0.132	0.144	0.004
	Single scenario reduction		0.120		
	Clustering alg.		0.087		
	Subtrees merging $p = 0.5$		0.139		
Dimension 5	Nodal extraction	0.913	0.961	1.020	0.027
	Nodal extraction improved	0.877	0.936	1.042	0.023
	Scenario extraction	0.799	0.831	0.875	0.015
	Single scenario reduction		0.706		
	Clustering alg.		0.706		
	Subtrees merging $p = 0.5$		0.875		
Dimension 10	Nodal extraction	2.969	3.134	3.364	0.091
	Nodal extraction improved	2.875	3.044	3.411	0.092
	Scenario extraction	2.599	2.716	2.841	0.045
	Single scenario reduction		2.333		
	Clustering alg.		2.256		
	Subtrees merging $p = 0.5$		2.862		
Dimension 20	Nodal extraction	4.641	4.895	5.405	0.113
	Nodal extraction improved	4.502	4.785	5.108	0.103
	Scenario extraction	3.906	4.067	4.238	0.064
	Single scenario reduction		4.230		
	Clustering alg.		3.552		
	Subtrees merging $p = 0.5$		4.549		

Table 3.2: The nested distance between the master trees with 10,000 scenarios and the reduced trees

3.3 Portfolio selection problem

Since we would like to have some other comparison between the scenario reduction algorithms described in Chapter 2 than just the nested distances and the computational time, we formulate and solve a simple portfolio selection problem, which allows us to compare also the distances between the optimal objective values and between the optimal solutions.

Formulation. Given the four-stage master trees, the initial wealth $W_0 = 100$, the number of assets $N = 5, 10$, and 20, the weekly returns ρ ($\rho_{i,k}$ is the weekly return of the asset i realized at node k from its ancestor $k-$), the scenario probabilities p_{k_4}, $k_4 \in \mathcal{K}_4$, the diversification coefficient $\theta = 0.65$, and the turnover coefficient $\gamma = 1$, the problem is formulated as follows, where $x_{k,i}$ is the amount of money invested in the asset i at the node k, W_{k_4} is the wealth at the end of the investment horizon, and $z_{k_t,i}$ represents how much the investment in the asset i changes from k_t- to k_t,

$$
\begin{aligned}
&\max_{\boldsymbol{x},\boldsymbol{W},\boldsymbol{z}} \sum_{k_4 \in \mathcal{K}_4} p_{k_4} W_{k_4} \\
\text{s.t.} \quad & \sum_{i=1}^{N} x_{k_0,i} = W_0, \\
& \sum_{i=1}^{N} (1+\rho_{i,k_t}) x_{k_{t-1},i} - \sum_{i=1}^{N} x_{k_t,i} = 0 \quad \forall k_{t-1} \in \mathcal{K}_{t-1},\ \forall k_t \in k_{t-1}+,\ t = 1,2,3, \\
& \sum_{i=1}^{N} (1+\rho_{i,k_4}) x_{k_3,i} = W_{k_4} \quad \forall k_3 \in \mathcal{K}_3,\ \forall k_4 \in k_3+, \\
& x_{k_t,i} \geq 0, \quad \forall i \in \{1,\dots,N\},\ \forall k_t \in \mathcal{K}_t,\ t = 0,1,2,3, \\
& \gamma \cdot \sum_{i=1}^{N} (1+\rho_{i,k_t}) x_{k_{t-1},i} - \sum_{i=1}^{N} z_{k_t,i} \geq 0 \quad \forall k_{t-1} \in \mathcal{K}_{t-1},\ \forall k_t \in k_{t-1}+,\ t = 1,2,3, \\
& (1+\rho_{i,k_t}) x_{k_{t-1},i} - x_{k_t,i} + z_{k_t,i} \geq 0 \quad \forall i,\ \forall k_{t-1} \in \mathcal{K}_{t-1},\ \forall k_t \in k_{t-1}+,\ t = 1,2,3, \\
& x_{k_t,i} - (1+\rho_{i,k_t}) x_{k_{t-1},i} + z_{k_t,i} \geq 0 \quad \forall i,\ \forall k_{t-1} \in \mathcal{K}_{t-1},\ \forall k_t \in k_{t-1}+,\ t = 1,2,3, \\
& \theta \cdot W_0 - x_{k_0,i} \geq 0, \quad \forall i, \\
& \theta \cdot \sum_{i=1}^{N} (1+\rho_{i,k_t}) x_{k_{t-1},i} - x_{k_t,i} \geq 0, \quad \forall i,\ \forall k_t \in \mathcal{K}_t,\ t = 1,2,3, \\
& W_{k_4} \in \mathbb{R}, \quad \forall k_4 \in \mathcal{K}_4\,.
\end{aligned}
$$

We are maximizing the average wealth with short-selling not allowed and an upper bound on how much we can invest in one asset. The upper bound depends on the wealth we have at that time and the diversification coefficient θ. For our choice $\gamma = 1$ the variables $\boldsymbol{z}$ and the constraints which contain these variables are omitted, i.e., portfolios can be completely changed at each stage.

This relatively simple problem had to be adjusted for the 10,000 scenario master trees with 10- and 20-dimensional nodal values in order to be solvable in the matlab implementation. The two next-to-last constraints, i.e., the upper bounds on the amount of money invested in one asset, were merged into one; they are of the form

$$\theta \cdot W_0 - x_{k_t,i} \geq 0, \quad \forall i,\ \forall k_t \in \mathcal{K}_t,\ t = 0,1,2,3.$$

After this adjustment, the upper bounds on the amount of wealth invested in one asset are the same for all assets at all stages and nodes. This way we avoid memory issues, which we face otherwise when trying to allocate the matrix of constraints.

3.4 Distance of the objective values

In this section, the nested distance to the distance between the optimal objective values is compared. The distance between the optimal objective values is measured using the ℓ^1 norm, i.e., for two optimal objective values obj_1^* and obj_2^* the distance is

$$\mathtt{d}(obj_1^*, obj_2^*) = |obj_1^* - obj_2^*| \,.$$

The objective value represents the average wealth after the investment, where we started with the initial wealth $W_0 = 100$.

3.4.1 2,400 to 24 reduction

The nested distance to the distance of the optimal objective values is shown in Figure 3.3; Table 3.3 summarizes the distances between the optimal objective values.

We can see that the optimal objective values of the reduced trees from the advanced algorithms are very close to the optimal objective values of the master trees. For the simple algorithms, on the other hand, it is very diverse with some optimal objective values very close and some very far from the optimal objective values of the master trees.

The optimal objective values from the scenario extraction never get close to the optimal objective values of the master trees, i.e., the minimum distance between them is much higher compared to the other algorithms. Moreover, also the average distance of the optimal objective values is higher for the scenario extraction than for the other algorithms; the variance of the distance and the maximum, however, are smaller for the scenario extraction than they are for the other two simple algorithms – both nodal extractions.

There is an apparent trend that the bigger the nested distance, the bigger the variance of the optimal objective distance. The variance also increases with the increase in the nodal dimension. The standard deviation is about 3 for the scenario extraction and about 5 for the nodal extractions in the 5-dimensional case. It increases to 12 for the scenario extraction and to about 15 for the nodal extractions in the 10-dimensional case, and it is 14 for the scenario extraction and over 24 for the nodal extractions in the 20-dimensional case. As well as the variance, also the average distance of the objectives increases with the increase in the dimension, which was expected.

What is not shown in the figure nor the table, is the fact that the optimal objective values are always higher than the optimal objective values of the master trees when using the reduced trees from the simple algorithms and the single scenario reduction. However, they are always lower for the clustering algorithm, and it varies for the subtrees merging algorithm.

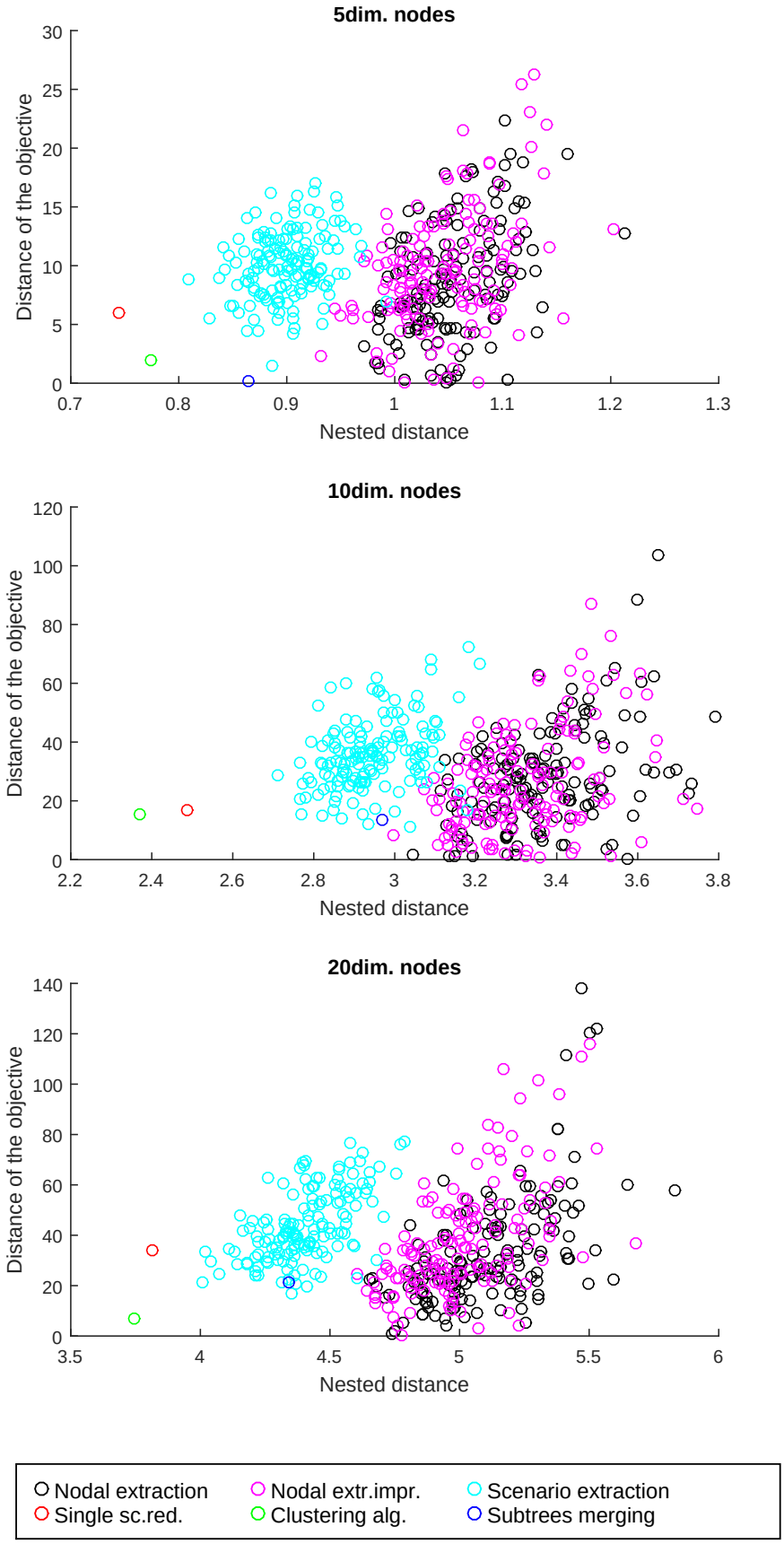


Figure 3.3: The distance between the optimal objective values for the 2,400 scenario master trees

The distance of the objectives					
	Algorithm	Min.	Mean	Max.	SD
Dimension 5	Nodal extraction	0.064	8.771	22.300	4.825
	Nodal extraction improved	0.050	9.695	26.226	4.899
	Scenario extraction	1.481	10.014	16.974	2.824
	Single scenario reduction		5.967		
	Clustering alg.		1.934		
	Subtrees merging $p = 0.5$		0.168		
Dimension 10	Nodal extraction	0.031	27.798	103.546	16.919
	Nodal extraction improved	0.718	25.943	86.962	17.130
	Scenario extraction	11.046	35.752	72.096	11.878
	Single scenario reduction		16.684		
	Clustering alg.		15.542		
	Subtrees merging $p = 0.5$		13.718		
Dimension 20	Nodal extraction	1.051	33.212	137.916	22.229
	Nodal extraction improved	0.399	37.890	115.704	22.431
	Scenario extraction	16.906	43.734	77.371	14.033
	Single scenario reduction		33.801		
	Clustering alg.		6.895		
	Subtrees merging $p = 0.5$		21.363		

Table 3.3: The distance between the objective values for the master trees with 2,400 scenarios

3.4.2 10,000 to 100 reduction

In Figure 3.4 and Table 3.4 is the distance between the optimal objective values for the master trees with 10,000 scenarios. Let us recall that the portfolio selection problem was adjusted for the dimensions 10 and 20, so the upper bound on the amount of money invested in one asset is for each asset, node, and stage the same constant, which is 65 % of the initial wealth $W_0 = 100$, i.e., the upper bound is equal to 65.

We see that the minimum distance between the optimal objective values is far from zero for the scenario extraction as it is in the case of the 2,400 to 24 reduction, but now it is more apparent. The average distance is again the highest for the scenario extraction; nevertheless, the variance and the maximum are still lower for the scenario extraction than for the nodal extractions. The standard deviations of the nodal extractions are even twice as high as the standard deviation of the scenario extraction, which is bigger difference than in the case of 2,400 to 24 reduction, where it is in the ratio 3:5.

Though the improved nodal extraction is closer (in the sense of the nested distance) to the master trees than the unimproved nodal extraction, the distances between the optimal objective values are about the same. Nevertheless, in general, it holds that the maximum distance between the optimal objective values decreases with the decrease in the nested distance.

The distance between the optimal objective values tends to increase with the increase in the dimension of the nodal values, however, the distinction between the 10-dimensional and 20-dimensional case is not as evident as in the 2,400 to 24 reduction; the results for these two dimensions in the 10,000 to 100 reduction are the same, more or less.

The optimal objective values for the advanced algorithms are relatively close to the optimal objective values of the master trees. However, the subtrees merging algorithm has slightly better results than the single scenario reduction and the clustering algorithm, for which the distances between the optimal objective values are practically the same although their nested distances are not (with the exception of the dimension 5).

Compared to the 2,400 to 24 reduction, the distances between the optimal objective values are lower for the 10,000 to 100 reduction for all dimensions. The difference is especially noticeable in the cases with 10- and 20-dimensional nodal values.

As it is in the case of the 2,400 to 24 reduction, the optimal objective values for the simple algorithms (the nodal extractions and the scenario extraction) and the single scenario reduction are always higher than the optimal objective values of the master trees, they are always lower for the clustering algorithm, and it varies for the subtrees merging algorithm.

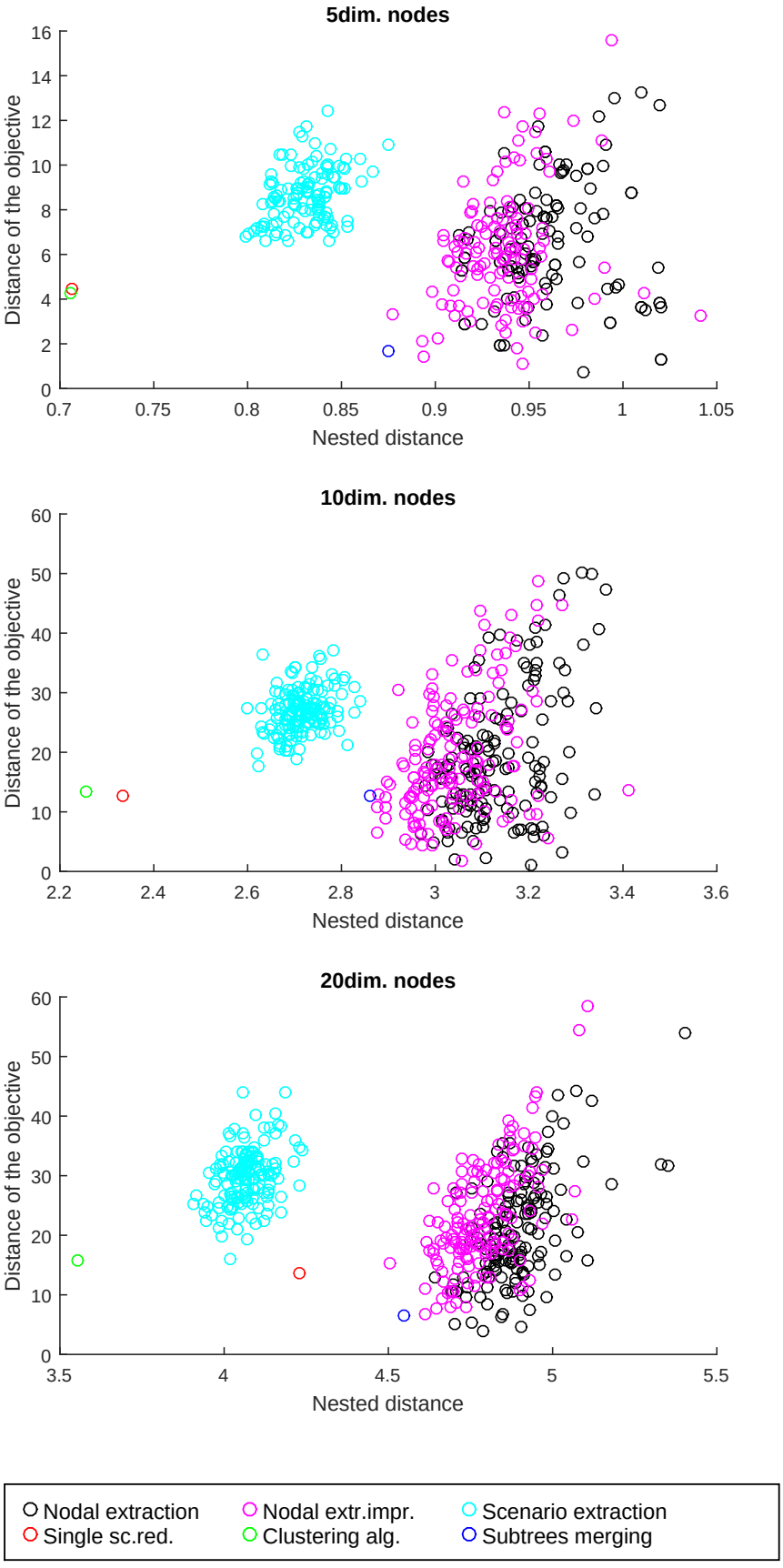


Figure 3.4: The distance between the optimal objective values for the 10,000 scenario master trees

The distance of the objectives					
	Algorithm	Min.	Mean	Max.	SD
Dimension 5	Nodal extraction	0.739	6.494	13.224	2.623
	Nodal extraction improved	1.107	6.157	15.609	2.618
	Scenario extraction	6.611	8.645	12.416	1.231
	Single scenario reduction		4.444		
	Clustering alg.		4.295		
	Subtrees merging $p = 0.5$		1.647		
Dimension 10	Nodal extraction	0.985	19.370	50.223	11.017
	Nodal extraction improved	1.718	19.218	48.727	10.055
	Scenario extraction	17.667	26.986	37.188	3.657
	Single scenario reduction		12.762		
	Clustering alg.		13.482		
	Subtrees merging $p = 0.5$		12.731		
Dimension 20	Nodal extraction	3.823	21.171	53.981	8.954
	Nodal extraction improved	6.663	22.984	58.459	8.857
	Scenario extraction	15.957	29.741	43.936	4.743
	Single scenario reduction		13.625		
	Clustering alg.		15.812		
	Subtrees merging $p = 0.5$		6.570		

Table 3.4: The distance between the objective values of the master trees with 10,000 scenarios

3.5 Distance of the solutions

By the distance between the optimal solutions we mean the ℓ^1 distance between the first decisions. That is, if $\boldsymbol{x}_1^1$ and $\boldsymbol{x}_1^2$ are two optimal solutions (decisions at the stage 0), the distance between them is then

$$\mathtt{d}(\boldsymbol{x}_1^1, \boldsymbol{x}_1^2) = \sum_{i=1}^{N} \left| x_{1,i}^1 - x_{1,i}^2 \right|,$$

where N is the dimension of the nodes, i.e., the number of assets, and $x_{1,i}^1$ and $x_{1,i}^2$, respectively, is the optimal amount of the initial wealth that should be invested in the asset i at time 0.

For our initial wealth 100, the distance's range is from 0 to 200, i.e., two completely different solutions have the distance equal to 200.

3.5.1 2,400 to 24 reduction

The distance between the optimal solutions to the nested distance is shown for all algorithms in Figure 3.5; the distance between the optimal solutions is summarized in Table 3.5.

We see that the distance between the optimal solutions tends to increase with the increase in the nodal dimension, i.e., for the dimension 20 almost every optimal solution is completely different from the optimal solution of the master tree, and there are only very few which are the same. For the dimension 5, on the other hand, a quarter of the optimal solutions from the nodal extractions and two thirds of the optimal solutions from the scenario extraction are the same as the optimal solution of the master tree, and only ten percent of the optimal solutions from the nodal extractions are completely different from the optimal solution of the master tree. There is a domination of the completely different optimal solutions in the case of the master with 10-dimensional nodal values, but not as great as for the dimension 20.

If we look at the dimension 5 (the top graph of Figure 3.5), we can see that for lower nested distances the solution distances are also, on average, lower than for the trees with higher nested distances. This is not so apparent for the other two dimensions (10 and 20), since there is a global increase in the distances between the optimal solutions.

When we compare the simple algorithms, the optimal solutions of the trees from the scenario extraction are clearly closer to the optimal solutions of the master trees than are the optimal solutions of the trees from the other two algorithms (the nodal extractions). This could be expected, since the scenario extraction tends to keep more nodes from the early stages than the nodal extractions.

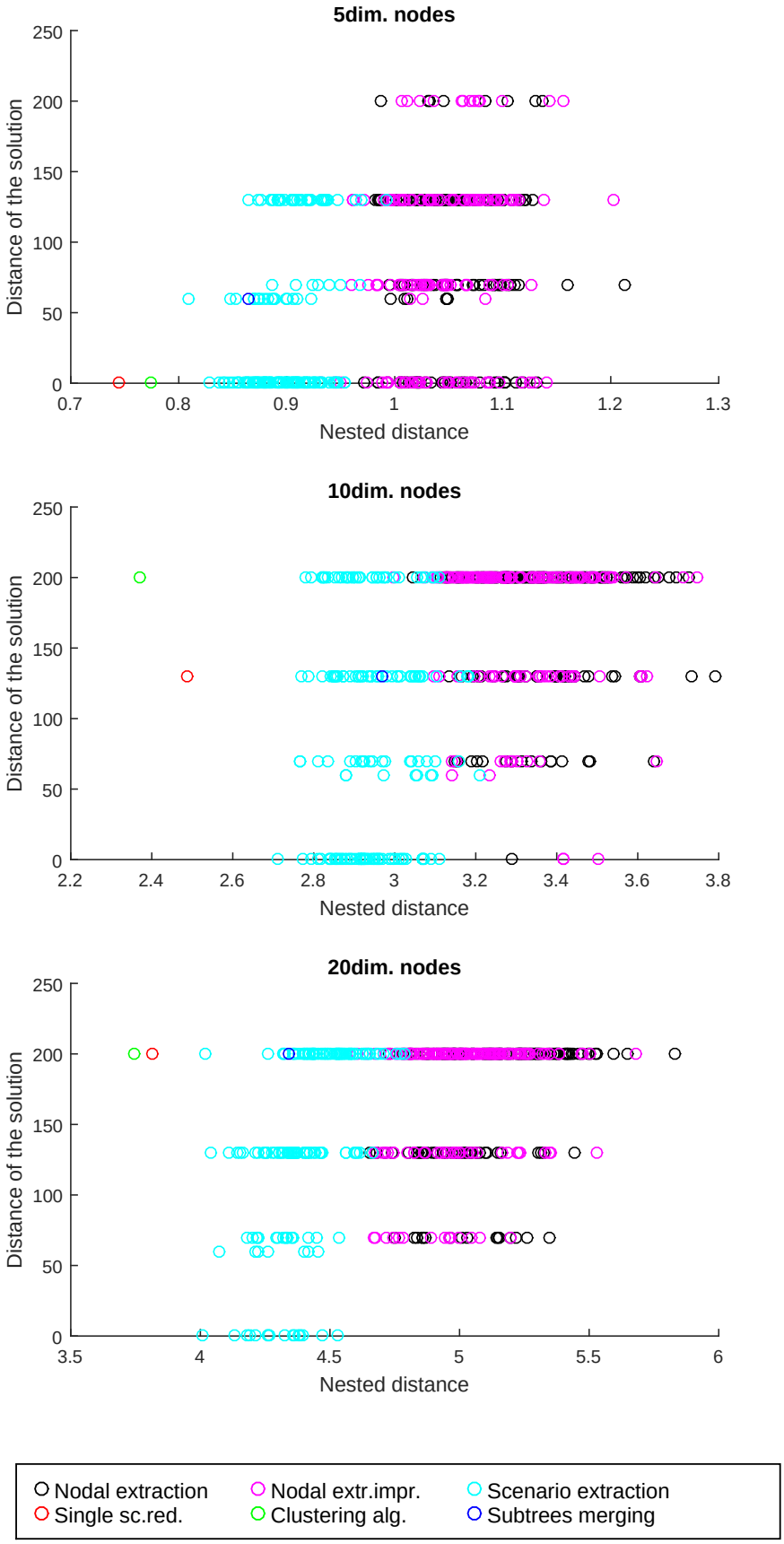


Figure 3.5: The distance between the optimal solutions for the 2,400 scenario master trees

The solution distance						
	Algorithm	0	60	70	130	200
Dimension 5	Nodal extraction	25 %	4 %	22 %	43 %	5 %
	Nodal extraction improved	25 %	2 %	23 %	41 %	9 %
	Scenario extraction	58 %	11 %	5 %	26 %	0 %
	Single scenario reduction	100 %	0 %	0 %	0 %	0 %
	Clustering alg.	100 %	0 %	0 %	0 %	0 %
	Subtrees merging $p = 0.5$	0 %	100 %	0 %	0 %	0 %
Dimension 10	Nodal extraction	1 %	0 %	11 %	24 %	64 %
	Nodal extraction improved	2 %	1 %	7 %	20 %	70 %
	Scenario extraction	33 %	5 %	13 %	27 %	22 %
	Single scenario reduction	0 %	0 %	0 %	100 %	0 %
	Clustering alg.	0 %	0 %	0 %	0 %	100 %
	Subtrees merging $p = 0.5$	0 %	0 %	0 %	100 %	0 %
Dimension 20	Nodal extraction	0 %	0 %	11 %	25 %	64 %
	Nodal extraction improved	0 %	0 %	9 %	22 %	69 %
	Scenario extraction	11 %	5 %	11 %	33 %	40 %
	Single scenario reduction	0 %	0 %	0 %	0 %	100 %
	Clustering alg.	0 %	0 %	0 %	0 %	100 %
	Subtrees merging $p = 0.5$	0 %	0 %	0 %	0 %	100 %

Table 3.5: Frequency of the solution distance for the master trees with 2,400 scenarios

3.5.2 10,000 to 100 reduction

Figure 3.6 shows the distance between the optimal solutions to the nested distance for the 10,000 scenarios master trees with 5-, 10-, and 20-dimensional nodal values. The frequencies of the distance between the optimal solutions are summarized in Table 3.6.

As for the dimension 5 in the case of 2,400 to 24 reduction, also in the case of 10,000 to 100 reduction we see that (on average) the distance of the optimal solutions decreases with the decrease in the nested distance. Unlike the case of the 2,400 to 24 reduction, in the case of the 10,000 to 100 reduction this is true for all dimensions, not just the dimension 5.

Unlike the 2,400 to 24 reduction, where there is an apparent increase in the distance between the optimal solutions with the increase in the dimension of the nodal values of the master trees, we do not see any noticeable increase in the case of the 10,000 to 100 reduction, on the contrary, there is an apparent decrease in the distance between the optimal solutions with the increase of the dimension of the nodal values of the master trees. More precisely, almost every optimal solution from the scenario extraction is the same as the optimal solution of the master tree with 20-dimensional nodal values, while only less than half of the optimal solutions is the same in the case of the master tree with 5-dimensional nodal values.

The reason why the optimal solutions from the scenario extraction are so close to the optimal solutions of the master trees is that the reduced trees from the scenario extraction keep most of the nodes from the first stage, whereas the other algorithms do not.

The distances between the optimal solutions are similar for both nodal extractions. Only a few optimal solutions from the nodal extractions are the same as the optimal solutions of the master trees; the majority of the optimal solutions from the nodal extractions has at least a part which is common with the optimal solution of the master tree.

We see that lot of optimal solutions, especially in the dimension 10, have a different distance from the optimal solution of the master tree than 0, 60, 70, 130, and 200. This is the consequence of the fact that these solutions do not have the upper bound constraint active, which the other optimal solutions do, i.e., in most optimal solutions we should invest in only two assets in the first investment – 65 in one and 35 in another.

Usually, the distance 60 corresponds to the situation where both optimal solutions invest in the same two assets, but the amount is switched, i.e., one optimal solution invests in the first common asset 65 and 35 in the other, the second optimal solution invests 35 in the first common asset and 65 in the other, the distances 70 and 130 correspond to the situation where both optimal solution invest in one common asset and one different; in the case of 60 the common asset is the major one, and it is the minor one in the case of 130.

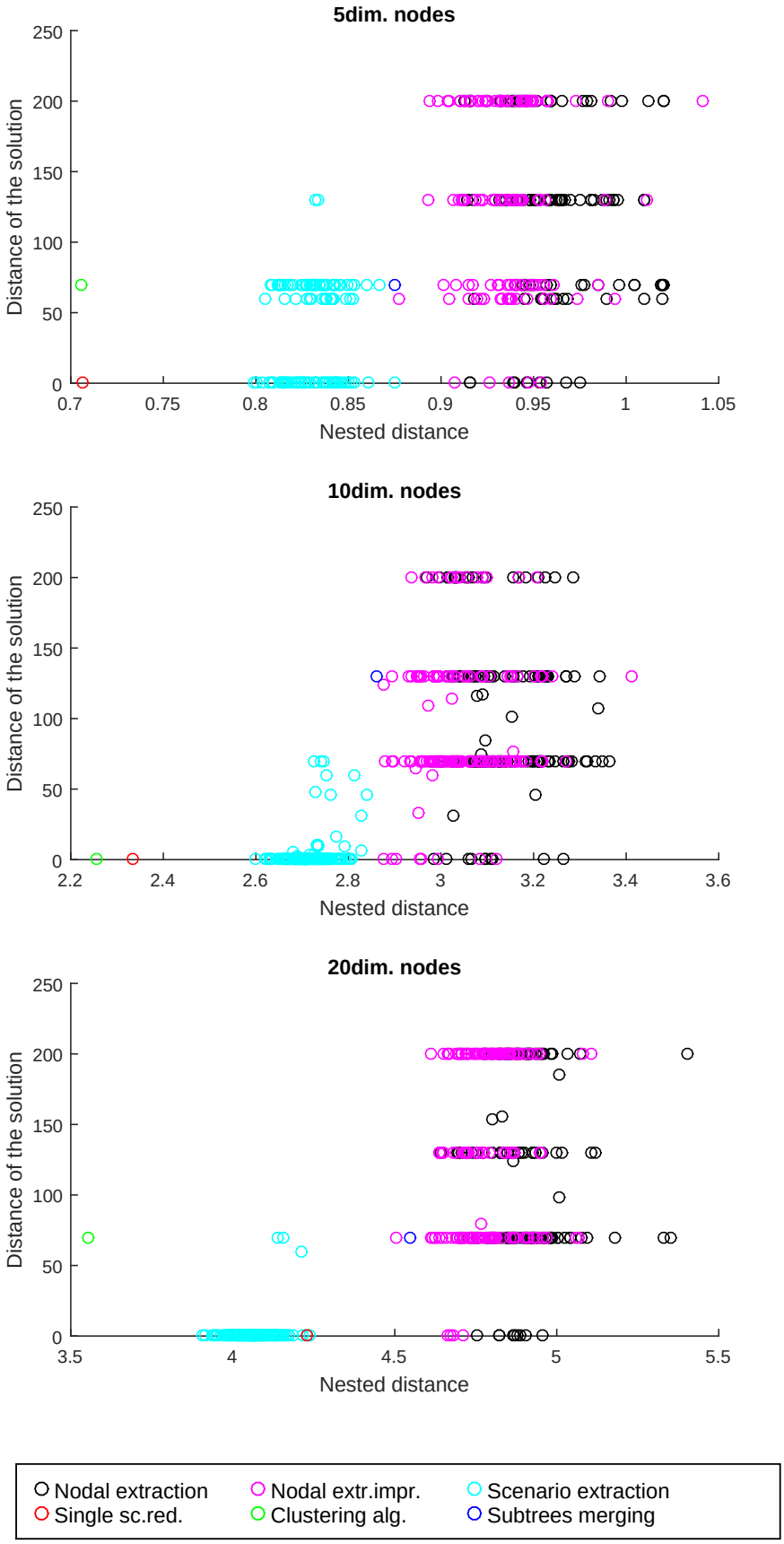


Figure 3.6: The distance between the optimal solutions for the 10,000 scenario master trees

The solution distance						
	Algorithm	0	60	70	130	200
Dimension 5	Nodal extraction	8 %	11 %	14 %	45 %	22 %
	Nodal extraction improved	4 %	15 %	17 %	28 %	36 %
	Scenario extraction	45 %	15 %	38 %	2 %	0 %
	Single scenario reduction	100 %	0 %	0 %	0 %	0 %
	Clustering alg.	0 %	0 %	100 %	0 %	0 %
	Subtrees merging $p = 0.5$	0 %	0 %	100 %	0 %	0 %
Dimension 10	Nodal extraction	7 %	0 %	48 %	26 %	14 %
	Nodal extraction improved	5 %	1 %	50 %	30 %	10 %
	Scenario extraction	85 %	1 %	2 %	0 %	0 %
	Single scenario reduction	100 %	0 %	0 %	0 %	0 %
	Clustering alg.	100 %	0 %	0 %	0 %	0 %
	Subtrees merging $p = 0.5$	0 %	0 %	0 %	100 %	0 %
Dimension 20	Nodal extraction	6 %	0 %	47 %	18 %	26 %
	Nodal extraction improved	3 %	0 %	39 %	17 %	41 %
	Scenario extraction	98 %	1 %	1 %	0 %	0 %
	Single scenario reduction	100 %	0 %	0 %	0 %	0 %
	Clustering alg.	0 %	0 %	100 %	0 %	0 %
	Subtrees merging $p = 0.5$	0 %	0 %	100 %	0 %	0 %

Table 3.6: Frequency of the solution distance for the master trees with 10,000 scenarios

Conclusion

In this thesis, we introduced several scenario reduction algorithms for scenario trees, and then we applied these algorithms on two types of master trees – one with 10,000 scenarios, the other with 2,400 scenarios. We compared the computational time of the algorithms and the nested distance between the master trees and the reduced trees. We also solved a simple portfolio selection problem with the master trees and the reduced trees and compared the optimal objective values and solutions.

The algorithm which uses clustering of nodal values with predetermined target structure seems to be the most suitable, since it produces a tree which is very close to the master tree in all the nested distance and the optimal objective value and solution in relatively short time (under 30 seconds for the 10,000 scenario master trees).

The single scenario reduction led to trees with comparable results to the clustering algorithm, but it takes much more time (around 14 hours for the 10,000 scenario master trees), which speaks in favor of the clustering algorithm. The generalization of this algorithm is impossible to apply on large trees for its computational demands.

All the simple algorithms generate a reduced tree within a second, but it depends on luck how far from the initial tree it is. It was shown that the resulted trees are many times very distant from the master tree and so is the optimal objective value and the optimal solution.

Concerning the subtrees merging algorithm, which is based on the nested distance, which we used as the measure between the master trees and the reduced trees, one would expect that the resulting tree would be the closest. However, it was not. The result strongly depends on the choice of the reducing parameter p, which, if chosen too high, might lead to a tree which is even bigger than the master tree, if chosen too small, we lose too much information and the approximation is not good enough. Nevertheless, despite its distance from the master tree, the optimal objective values and the optimal solutions are close to the optimal objective values and the optimal solutions of the master trees. There exists an improvement of this algorithm in the literature ([7]), which is even more computationally demanding, and therefore was not considered in this thesis.

All in all, we recommend the clustering algorithm for its good results.

Bibliography

[1] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, 1997.

[2] J. Dupačová, N. Gröwe-Kuska and W. Römisch. "Scenario reduction in stochastic programming". In: *Mathematical Programming* 95 (2003), 493–511.

[3] J. Dupačová, J. Hurt and J. Štěpán. *Stochastic Modeling in Economocis and Finance*. Kluwer Academic Publishers, 2002.

[4] J. Leskovec. *Clustering*. URL: `https://web.stanford.edu/class/cs246/slides/05-clustering.pdf`.

[5] K. Parthasarathy and R. Kalyanapuram. *Probability Measures on Metric Spaces*. Academic press, New York, 1972.

[6] G. Ch. Pflug and A. Pichler. "A Distance for Multistage Stochastic Optimization Models". In: *SIAM Journal on Optimization* 22.1 (2012), 1–23.

[7] G. Ch. Pflug and A. Pichler. *Multistage Stochastic Optimization*. Springer Series in Operations Research and Financial Engineering, 2014.

[8] W. B. Powell. "Clearing the Jungle of Stochastic Optimization". In: *Informs TutORials* (2014).

[9] S. T. Rachev. *Probability Metrics and the Stability of Stochastic Models*. Wiley, Chichester, 1991.

[10] S. T. Rachev, S. V. Stoyanov and F. J. Fabozzi. *A Probability Metrics Approach to Financial Risk Measures*. Wiley, London, 2011.

[11] A. Shapiro, D. Dentcheva and A. Ruszczyński. *Lectures on Stochastic Programming. Modeling and Theory*. SIAM and Mathemacital Programming Society, 2009.

[12] A. V. Timonina. "Multi-stage stochastic optimization: the distance between stochastic scenario processes". In: *Computational Management Science* 12.1 (2015), 171–195.