

# Decomposition of Multistage Stochastic Programs with Recombining Scenario Trees

CHRISTIAN KÜCHLER\* and STEFAN VIGERSKE\*

Humboldt–Universität zu Berlin, Unter den Linden 6,  
D–10099 Berlin, Germany,  
ckuechler@math.hu-berlin.de, stefan@math.hu-berlin.de

## Abstract

This paper presents a decomposition approach for linear multistage stochastic programs, that is based on the concept of recombining scenario trees. The latter, widely applied in Mathematical Finance, may prevent the node number of the scenario tree to grow exponentially with the number of time stages. It is shown how this property may be exploited within a non-Markovian framework and under time-coupling constraints. Being close to the well-established Nested Benders Decomposition, our approach uses the special structure of recombining trees for simultaneous cutting plane approximations. Convergence is proved and stopping criteria are deduced. Techniques for the generation of suitable scenario trees and some numerical examples are presented.

**Keywords.** Multistage Stochastic Programming, Nested Benders Decomposition, Recombining Scenario Trees

**AMS subject classification.** 90C15, 90C39, 49M27, 65K05

## Introduction

In multistage stochastic optimization problems the underlying stochastic processes are usually represented through scenario trees. Unfortunately, even under a moderate branching structure, the number of scenarios can grow exponentially with the number of time stages. Consequently, many problems of practical interest are approximated by models

---

\*Support by the Wiener Wissenschafts-, Forschungs- und Technologiefonds (WWTF) and by the Bundesministerium für Bildung und Forschung (BMBF) under the grant 03SF0312E is gratefully acknowledged.

that include only few time stages or a small number of scenarios. An attempt to create such low-dimensional models relies on scenario reduction techniques, which pursue to present the stochastic processes as best as possible through a largely reduced set of scenarios, cf. [14]. An approach often used in practice, aiming to find acceptable decisions along a concrete observation process, is to optimize with rolling time horizon, cf. [26].

Under certain circumstances, a further approach to handle the dimensionality is to use recombining scenario trees. An illustrative example is the binomial model of stock price behaviour of Cox, Ross, and Rubinstein [5], whereby the number of nodes under  $T$  time stages diminishes from  $2^T - 1$  to  $T(T + 1)/2$  through recombination of scenarios. However, some information about the history of a node in the tree will be lost. This has no consequences if the represented stochastic process has the Markov property and the optimization problem contains no time-coupling constraints. But whenever one of these properties is not fulfilled, it is difficult to formulate a dynamic programming equation on a recombining scenario tree. Practical problems include often both non-Markovian stochastic input and time-coupling constraints. For such problems, recombining scenario trees seem to be inappropriate at first sight.

Motivated by the numerical possibilities of recombining scenario trees, we develop an approach to solve large-scale and long-term multistage stochastic programs. The basic idea can be described as follows. Although in many cases of practical interest the stochastic input is not Markovian, it has only a comparable short-term memory and can be displayed without great loss of precision by a scenario tree, where at certain time points scenarios with similar short-term history are recombined. However, in the presence of time-coupling constraints, the relevant past of both the stochastic process and the control variables has to be available at each node to allow a numerical solution based on the dynamic programming principle. A way out of this dilemma is to consider a reduced set of subtrees originated at one or multiple time stages. This approach does not result in a recombining tree, but it can be interpreted as the recombination of scenarios and offers the numerical advantages of recombining trees through the possibility to simultaneously approximate cost-to-go functions in nodes with corresponding subtrees.

Section 1 presents the problem structure and the concept of recombining scenarios. The solution algorithm is detailed in Section 2, as well as convergence results and stopping criteria. In Section 3 we sketch a method for generating recombining scenario trees. A small example and some numerical results that demonstrate the potentials of our method are presented in Section 4. Finally, we discuss possible extensions and further developments in Section 5.

**Acknowledgement.** We are grateful to Prof. Werner Römisch for his help and encouragement.

# 1 Problem Formulation

## 1.1 Linear multistage stochastic programs

On a probability space  $(\Omega, \mathcal{F}, \mathbb{P})$ , we consider an  $\mathbb{R}^s$ -valued discrete time stochastic process  $\boldsymbol{\xi} = (\boldsymbol{\xi}_t)_{t=1, \dots, T}$  with *time horizon*  $T \in \mathbb{N}$ . The filtration  $(\mathcal{F}_t)_{t=1, \dots, T}$  induced by  $\boldsymbol{\xi}$  is defined by

$$\mathcal{F}_t := \sigma(\boldsymbol{\xi}^t) \quad \text{with} \quad \boldsymbol{\xi}^t := (\boldsymbol{\xi}_s)_{s=1, \dots, t}, \quad t = 1, \dots, T.$$

Following the notation of [17], we consider the linear multistage stochastic program

$$(P) \quad \min \left\{ \mathbb{E} \left[ \sum_{t=1}^T \langle b_t(\boldsymbol{\xi}_t), x_t \rangle \right] : \begin{array}{l} x_t \in X_t, x_t \in \mathcal{F}_t, t = 1, \dots, T \\ A_{t,0}(\boldsymbol{\xi}_t)x_t + A_{t,1}(\boldsymbol{\xi}_t)x_{t-1} = h_t(\boldsymbol{\xi}_t), t = 2, \dots, T \end{array} \right\},$$

for closed polyhedral sets  $X_t \subset \mathbb{R}^r$  and matrix-valued mappings  $A_{t,0}(\cdot)$ ,  $A_{t,1}(\cdot)$ , and  $h_t(\cdot)$  of suitable dimensions. To render possible a numerical solution of (P),  $\boldsymbol{\xi}$  is assumed to take only finitely many values, i.e., the support of  $\boldsymbol{\xi}^t$ 's distribution can be written as

$$\Xi^t := \{\boldsymbol{\xi}_i^t : i = 1, \dots, n_t \in \mathbb{N}\} := \text{supp } \mathbb{P}[\boldsymbol{\xi}^t \in \cdot] \subset \mathbb{R}^{s \cdot t},$$

for  $t = 1, \dots, T$ . Considering certain time stages

$$0 = R_0 < R_1 < \dots < R_n < R_{n+1} = T,$$

the *cost-to-go function* at time  $R_j$  and state  $(x_{R_j}, \boldsymbol{\xi}_i^{R_j}) \in X_{R_j} \times \Xi^{R_j}$  is defined recursively by  $\mathcal{Q}_{R_{n+1}}(\cdot, \cdot) := 0$  and the Bellman Equation

$$\begin{aligned} (\mathcal{Q}_{R_j}) \quad \mathcal{Q}_{R_j}(x_{R_j}, \boldsymbol{\xi}_i^{R_j}) &:= \min \mathbb{E} \left[ \sum_{t=R_j+1}^{R_{j+1}} \langle b_t(\boldsymbol{\xi}_t), x_t \rangle + \mathcal{Q}_{R_{j+1}}(x_{R_{j+1}}, \boldsymbol{\xi}^{R_{j+1}}) \middle| \boldsymbol{\xi}^{R_j} = \boldsymbol{\xi}_i^{R_j} \right] \\ \text{s.t. } x_t &\in X_t, \quad x_t \in \mathcal{F}_t, \quad t = R_j + 1, \dots, R_{j+1}, \\ A_{t,0}(\boldsymbol{\xi}_t)x_t + A_{t,1}(\boldsymbol{\xi}_t)x_{t-1} &= h_t(\boldsymbol{\xi}_t), \quad t = R_j + 1, \dots, R_{j+1} \end{aligned}$$

for  $j = 1, \dots, n$ . Using this notation, (P) can be reformulated in terms of Dynamic Programming:

$$\begin{aligned} (\mathcal{Q}_0) \quad \min \quad &\mathbb{E} \left[ \sum_{t=1}^{R_1} \langle b_t(\boldsymbol{\xi}_t), x_t \rangle + \mathcal{Q}_{R_1}(x_{R_1}, \boldsymbol{\xi}^{R_1}) \right] \\ \text{s.t. } \quad &x_t \in X_t, \quad x_t \in \mathcal{F}_t, \quad t = 1, \dots, R_1, \\ &A_{t,0}(\boldsymbol{\xi}_t)x_t + A_{t,1}(\boldsymbol{\xi}_t)x_{t-1} = h_t(\boldsymbol{\xi}_t), \quad t = 2, \dots, R_1, \end{aligned}$$

and solved by, e.g., the *Nested Benders Decomposition method* [2, 21, 27]. Thereby, the piecewise-linear convex functions [3, Thm. 40]

$$x_{R_j} \mapsto \mathcal{Q}_{R_j}(x_{R_j}, \boldsymbol{\xi}_i^{R_j}), \quad j = 1, \dots, n, \quad i = 1, \dots, n_{R_j},$$

are approximated successively by a set of supporting hyperplanes and evaluated in a time- and node-dependent, adaptively chosen sequence of points  $x_{R_j}$ . Unfortunately, the number of nodes and functions that have to be approximated can grow exponentially with increasing number of time steps  $T$ .

## 1.2 Recombining scenario trees

The finiteness of  $\Xi^T$  allows to represent the process  $\boldsymbol{\xi}$  by a scenario tree, cf., e.g., [7]. We say that  $\boldsymbol{\xi}$  can be represented by a scenario tree within that the nodes  $\{\boldsymbol{\xi}^{R_j} = \xi_i^{R_j}\}$  and  $\{\boldsymbol{\xi}^{R_j} = \xi_k^{R_j}\}$  can be *recombined* at time  $R_j$ , if both share the same associated subtree, i.e., the corresponding conditional distributions of  $(\boldsymbol{\xi}_t)_{t=R_j+1,\dots,T}$  coincide:

$$(1) \quad \mathbb{P} \left[ (\boldsymbol{\xi}_t)_{t=R_j+1,\dots,T} \in \cdot \mid \boldsymbol{\xi}^{R_j} = \xi_i^{R_j} \right] = \mathbb{P} \left[ (\boldsymbol{\xi}_t)_{t=R_j+1,\dots,T} \in \cdot \mid \boldsymbol{\xi}^{R_j} = \xi_k^{R_j} \right].$$

The resulting scenario tree can then be displayed as a recombining tree with much less nodes than the original tree. Furthermore, recombining at several time stages may prevent the node number to grow exponentially with the number of time stages, cf. Figure 1. But, due to time-coupling constraints, the scenario-dependent control  $x_{R_j}(\boldsymbol{\xi}^{R_j})$  will not be equal on  $\{\boldsymbol{\xi}^{R_j} = \xi_i^{R_j}\}$  and  $\{\boldsymbol{\xi}^{R_j} = \xi_k^{R_j}\}$ , in general. Therefore, solution methods based on Dynamic Programming, like the Nested Benders Decomposition method, can not be applied on a recombining tree. Thus, we have to abstain from combining nodes.

Nevertheless, (1) can be useful, since it entails equality of the cost-to-go functions  $\mathcal{Q}_{R_j}(\cdot, \xi_i^{R_j})$  and  $\mathcal{Q}_{R_j}(\cdot, \xi_k^{R_j})$ . The benefit becomes apparent within the Nested Benders Decomposition Algorithm, where a cutting plane approximation of  $\mathcal{Q}_{R_j}(\cdot, \xi_i^{R_j})$  can be used to approximate  $\mathcal{Q}_{R_j}(\cdot, \xi_k^{R_j})$ , simultaneously for every  $\xi_k^{R_j}$ ,  $k = 1, \dots, n_{R_j}$ , that fulfils (1). In this way, a considerable reduction of the numerical complexity of the problem can be achieved.

The relation (1) divides every  $\Xi^{R_j}$ ,  $j = 1, \dots, n$ , into a family of equivalence classes. The latter can be represented by a set of *representative nodes at time  $R_j$*  for  $j = 1, \dots, n$ :

$$\Lambda^{R_j} := \left\{ \lambda_1^{R_j}, \dots, \lambda_{m_{R_j}}^{R_j} \right\} \subset \Xi^{R_j} \text{ with}$$

- (i) for every  $\xi_i^{R_j} \in \Xi^{R_j}$ , it exists  $\xi_k^{R_j} = \lambda_m^{R_j} \in \Lambda^{R_j}$  such that (1) holds for  $i$  and  $k$ ,
- (ii) for every  $\xi_i^{R_j}, \xi_k^{R_j} \in \Lambda^{R_j}$  with  $i \neq k$ , (1) does *not* hold for  $i$  and  $k$ .

Thereby,  $m_{R_j} \in \mathbb{N}$  denotes the number of different subtrees originated at time stage  $R_j$ ,  $j = 1, \dots, n$ . Given  $\Lambda^{R_j}$ , every node  $\xi_i^{R_j} \in \Xi^{R_j}$  is associated with a representative node by the well-defined mapping

$$\boldsymbol{\lambda}^{R_j} : \Xi_i^{R_j} \rightarrow \Lambda^{R_j}, \text{ such that } \xi_i^{R_j} \text{ and } \boldsymbol{\lambda}^{R_j}(\xi_i^{R_j}) \text{ fulfill (1).}$$

Although we do not use trees that are recombining in the strict sense, scenario trees with

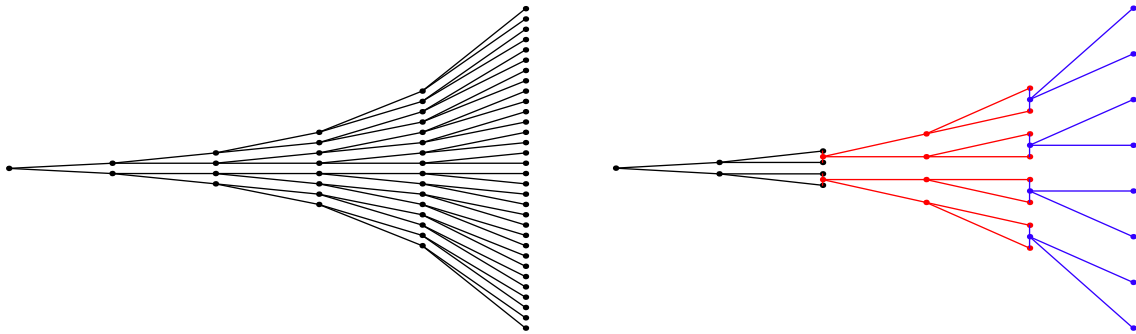


Figure 1: Non-recombining binary tree (left side) and binary tree with recombinations at stages  $R_1 = 3$  and  $R_2 = 5$  (right side).

coinciding subtrees in the sense of (1) are denoted as recombining, in the following.

While the number of different functions  $\mathcal{Q}_{R_j}(\cdot, \xi_i^{R_j})$  at time  $R_j$  has been reduced from  $n_{R_j}$  to  $m_{R_j}$ , the non-recombining nature of the control process  $(x_t)$  still causes an exponential growth (with increasing  $j$ ) of the number of control points that demand for an evaluation of cost-to-go functions  $\mathcal{Q}_{R_j}(\cdot, \lambda_m^{R_j})$ . Hence, to develop an algorithm that efficiently exploits property (1) for solving  $(\mathcal{Q}_0)$ , we have to find a way to deal with this difficulty.

## 2 Solving a linear multistage stochastic program with recombining scenarios

In this section we propose a modified Nested Benders Decomposition for the solution of (P). The basic algorithm, presented in Section 2.1, makes use of the identity (1) to simultaneously approximate the cost-to-go functions. Already leading to a complexity reduction, this approach forms the basis for further improvements. More precisely, it is shown in Sections 2.2 and 2.3 how the coincidence of subproblems and Lipschitz continuity and convexity of the cost-to-go functions allow to handle the exponential growth of control points. The full algorithm is stated in Section 2.4.

For simplicity, we avoid unboundedness of the subproblems  $(\mathcal{Q}_{R_j})$  by assuming boundedness of the sets  $X_t$ ,  $t = 1, \dots, T$ . However, in general, the method of Van Slyke and Wets [27] could be used to address this problem.

Whenever problem  $(\mathcal{Q}_{R_j})$  is infeasible for some  $x_{R_j}$ , we write  $\mathcal{Q}_{R_j}(x_{R_j}, \xi_i^{R_j}) = \infty$ , following the convention that the infimum over an empty set is equal to  $\infty$ .

## 2.1 A Nested Benders Decomposition approach

Consider the formulation  $(\mathcal{Q}_0)$  of problem (P). In a Nested Benders Decomposition approach [2, 21, 27], the cost-to-go functions  $\mathcal{Q}_{R_j}(\cdot, \lambda_i^{R_j})$  are replaced by an approximation using supporting hyperplanes (cutting planes). Due to the recombining nature of the stochastic process, the value functions  $\mathcal{Q}_{R_j}(\cdot, \xi_i^{R_j})$  and  $\mathcal{Q}_{R_j}(\cdot, \xi_k^{R_j})$  coincide whenever  $\xi_i^{R_j}$  and  $\xi_k^{R_j}$  fulfill (1), and, thus, they can be approximated simultaneously. Note that, in difference to the classical Nested Benders Decomposition [2], we do not decompose the problem at every time stage, but only on the stages  $R_j$ ,  $j = 1, \dots, n$ .

More formal, we define the following underestimating functions: For  $j = n, \dots, 0$ ,  $\bar{x}_{R_j} \in X_{R_j}$ , and  $\lambda_i^{R_j} \in \Lambda^{R_j}$  let

$$\begin{aligned}
 (\mathcal{Q}_{R_j}^L) \\
 \mathcal{Q}_{R_j}^L(\bar{x}_{R_j}, \lambda_i^{R_j}) &:= \min \mathbb{E} \left[ \sum_{t=R_j+1}^{R_{j+1}} \langle b_t(\boldsymbol{\xi}_t), x_t \rangle + \mathcal{Q}_{R_{j+1}}^{LC}(x_{R_{j+1}}, \boldsymbol{\lambda}^{R_{j+1}}(\boldsymbol{\xi}^{R_{j+1}})) \middle| \boldsymbol{\xi}^{R_j} = \lambda_i^{R_j} \right] \\
 \text{s.t. } x_t &\in X_t, \quad x_t \in \mathcal{F}_t, \quad t = R_j + 1, \dots, R_{j+1}, \\
 A_{t,0}(\boldsymbol{\xi}_t)x_t + A_{t,1}(\boldsymbol{\xi}_t)x_{t-1} &= h_t(\boldsymbol{\xi}_t), \quad t = R_j + 1, \dots, R_{j+1}, \\
 (2) \quad x_{R_j} &= \bar{x}_{R_j},
 \end{aligned}$$

where  $\mathcal{Q}_{R_{n+1}}^{LC}(\cdot, \cdot) := 0$  and  $\mathcal{Q}_{R_{j+1}}^{LC}(\cdot, \lambda_i^{R_{j+1}})$  is an approximation of  $\mathcal{Q}_{R_{j+1}}^L(\cdot, \lambda_i^{R_{j+1}})$  by supporting hyperplanes that is *easy* to evaluate and that will be properly defined in equation (4) below. Problem  $(\mathcal{Q}_{R_j}^L)$  is often referred to as *master problem* in the literature.

**Cutting plane approximation of  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$ .** The function  $\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda_i^{R_j})$  is used to induce a feasible solution at stage  $R_j$  and to approximate the value of  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  on its domain. For the latter purpose, let  $\bar{x} \in X_{R_j}$  with  $\mathcal{Q}_{R_j}^L(\bar{x}, \lambda_i^{R_j}) < \infty$ . An *optimality cut* supporting  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  is given by  $\mathcal{Q}_{R_j}^L(\bar{x}, \lambda_i^{R_j}) + \langle \pi, x_{R_j} - \bar{x} \rangle \leq 0$ , where  $\pi$  denotes the dual variables corresponding to the constraint (2) in an optimal solution of problem  $(\mathcal{Q}_{R_j}^L)$ .

To induce feasibility at time stage  $R_j$ , a point  $x_{R_j}$  that is infeasible for  $(\mathcal{Q}_{R_j}^L)$  is cut off using a feasibility cut: Let the function  $U(\cdot, \lambda_i^{R_j})$  measure the minimal  $L^1$ -distance of a point  $\bar{x} \in X_{R_j}$  from the feasible set of  $(\mathcal{Q}_{R_j}^L)$ ,

$$\begin{aligned}
 U(\bar{x}, \lambda_i^{R_j}) &:= \min_x \mathbb{E} \left[ \|x_{R_j} - \hat{x}_{R_j}\|_1 \right. \\
 &\quad \left. + \sum_{t=R_j+1}^{R_{j+1}} d_1(x_t, X_t) + \|A_{t,0}(\boldsymbol{\xi}_t)x_t + A_{t,1}(\boldsymbol{\xi}_t)x_{t-1} - h_t(\boldsymbol{\xi}_t)\|_1 \middle| \boldsymbol{\xi}^{R_j} = \lambda_i^{R_j} \right] \\
 \text{s.t. } x_t &\in \mathcal{F}_t, \quad t = R_j, \dots, R_{j+1}, \\
 (3) \quad \hat{x}_{R_j} &= \bar{x},
 \end{aligned}$$

where  $d_1(y, A) := \inf_{z \in A} \|y - z\|_1$  for  $y \in \mathbb{R}^r$ ,  $A \subseteq \mathbb{R}^r$ . Hence,  $U(\bar{x}, \lambda_i^{R_j}) > 0$  if and only if  $\mathcal{Q}_{R_j}^L(\bar{x}, \lambda_i^{R_j}) = \infty$ . Introducing slack variables, problem  $U(\bar{x}, \lambda_i^{R_j})$  can be formulated as a linear problem. Thus, analog to optimality cuts, a *feasibility cut* is given by a linearization of  $U(\cdot, \lambda_i^{R_j})$  at  $\bar{x}$ ,  $U(\bar{x}, \lambda_i^{R_j}) + \langle \pi, x_{R_j} - \bar{x} \rangle \leq 0$ , where  $\pi$  denotes the dual variables corresponding to the constraint (3) in an optimal solution of  $U(\bar{x}, \lambda_i^{R_j})$ .

To summarize, an approximation of  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  by means of optimality cuts  $C_{\text{opt}}(\lambda_i^{R_j})$  and feasibility cuts  $C_{\text{feas}}(\lambda_i^{R_j})$  is given by

$$(4) \quad \mathcal{Q}_{R_j}^{LC}(x_{R_j}, \lambda_i^{R_j}) := \max_{(\bar{x}, \bar{\pi}) \in C_{\text{opt}}(\lambda_i^{R_j})} \mathcal{Q}_{R_j}^L(\bar{x}, \lambda_i^{R_j}) + \langle \bar{\pi}, x_{R_j} - \bar{x} \rangle$$

$$\text{s.t.} \quad U(\bar{x}, \lambda_i^{R_j}) + \langle \bar{\pi}, x_{R_j} - \bar{x} \rangle \leq 0, \quad (\bar{x}, \bar{\pi}) \in C_{\text{feas}}(\lambda_i^{R_j}).$$

**Nested Benders Decomposition Algorithm.** The solution algorithm processes the master problems  $(\mathcal{Q}_{R_j}^L)$ ,  $j = 0, \dots, n$ , of the decomposed scenario tree in a forward or backward manner. At each time stage  $R_j$  the master problems  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$ ,  $\lambda_i^{R_j} \in \Lambda^{R_j}$ , are considered. The algorithm evaluates  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  for a set  $Z_j(\lambda_i^{R_j})$  of controls  $x_{R_j}$ . If  $\mathcal{Q}_{R_j}^{LC}(x_{R_j}, \lambda_i^{R_j}) < \mathcal{Q}_{R_j}^L(x_{R_j}, \lambda_i^{R_j})$ , new optimality or feasibility cuts are generated and added to the master problems at stage  $R_{j-1}$ . Further, in the forward mode, new control points  $x_{R_{j+1}}$  are generated from the solution of the master problem  $(\mathcal{Q}_{R_j}^L)$  to form the sets  $Z_{j+1}(\lambda_i^{R_{j+1}})$  for  $\lambda_i^{R_{j+1}} \in \Lambda^{R_{j+1}}$ . A special feasibility restoration mode ensures that traversing the tree in forward mode is only continued when all master problems at the current time stage could be solved. The algorithm stops when either the first timeperiod master problem  $(\mathcal{Q}_0^L)$  is infeasible, or all master problems could be solved to optimality and the generation of cuts has stopped. In the former case, also problem  $(\mathcal{Q}_0)$  is infeasible, in the latter, the problem has been solved to optimality. We refer to [2, 11, 21] for a more formal description of the algorithm.

**Proposition 1** (see, e.g., [21, Thm. 5]). *The Nested Benders Decomposition Algorithm stops in a finite number of steps either by reporting infeasibility of  $(\mathcal{Q}_0)$  or with a solution that is optimal for  $(\mathcal{Q}_0)$ .*

The finiteness of the algorithm follows from the polyhedrality of the functions  $U(\cdot, \lambda_i^{R_j})$  and  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$ , since only a finite number of cuts are needed to represent all facets of these functions.

## 2.2 Dealing with exponential growing control point sets

As mentioned above, a numerical challenge lies in the non-recombining nature of the control process. Since every function  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  is evaluated for all controls  $x_{R_j} \in Z_j(\lambda_i^{R_j})$  and each evaluation of  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  yields a set of new controls  $x_{R_{j+1}}(\xi^{R_{j+1}})$ , which are then

added to  $Z_{j+1}(\boldsymbol{\lambda}^{R_{j+1}}(\boldsymbol{\xi}^{R_{j+1}}))$ , the sets of control points  $Z_j(\lambda_i^{R_j})$  are growing exponentially with increasing  $j = 1, \dots, n$ .

We attempt to further improve the algorithm's efficiency by *thinning out* the sets  $Z_j(\lambda_i^{R_j})$ . Thus, before a master problem corresponding to scenario  $\lambda_i^{R_j}$  is processed, the set  $Z_j(\lambda_i^{R_j})$  is aggregated to a subset of representative points that have preferably many nonrepresentative points close to it. To this end, a graph  $G$  is constructed that has the points  $Z_j(\lambda_i^{R_j})$  as vertices. An edge exists between two points  $x$  and  $\hat{x}$  if and only if  $x$  and  $\hat{x}$  are considered as close, that is  $|x_i - \hat{x}_i| \leq \rho(x_i^{\text{up}} - x_i^{\text{low}})$  for  $i = 1, \dots, r$ , where  $\rho \in [0, 1]$  is a *thinning parameter* and  $x^{\text{low}}$  and  $x^{\text{up}}$  denote bounds on the values of  $x$ . The existence of  $x^{\text{low}}$  and  $x^{\text{up}}$  is ensured by the boundedness assumption on  $X_{R_j}$ .

An aggregation of  $Z_j(\lambda_i^{R_j})$  is computed by finding a minimum vertex cover in  $G$ , i.e., a subset of vertices  $S$  of minimal cardinality such that the union of  $S$  and its neighbourhood in  $G$  yields  $Z_j(\lambda_i^{R_j})$ . The grid-size parameter  $\rho$  guides the roughness of the thinning operation, that is  $\rho = 0$  corresponds to the removal of no points, and  $\rho = 1$  corresponds to the reduction of  $Z_j(\lambda_i^{R_j})$  to a singleton.

Note, that minimum vertex cover is known to be an NP problem [10], so that the computation of a *best* (smallest) aggregation of  $Z_j(\lambda_i^{R_j})$  can result in a considerable slowdown of the algorithm. Hence, we use a greedy heuristic to compute a *good* (small) aggregation of  $Z_j(\lambda_i^{R_j})$ . This heuristic selects representative points once at a time from  $Z_j(\lambda_i^{R_j})$  by searching for a vertex of maximum degree. After removing the neighbourhood of such a point from  $G$ , the algorithm proceeds with the remaining vertices until no edges are left in  $G$ .

**Proposition 2.** *Assume that relatively complete recourse [22] is fulfilled on the time stages  $R_j$ ,  $j = 0, \dots, n$ . Denote by  $v$  the optimal value of  $(\mathcal{Q}_0)$  and by  $v^L$  the optimal value of the lower bounding approximation  $(\mathcal{Q}_0^L)$ . The Nested Benders Decomposition Algorithm with thinning stops after a finite number of steps with a solution of  $(\mathcal{Q}_0^L)$  that is  $\rho$ -optimal for  $(\mathcal{Q}_0)$ , i.e.,  $|v - v^L| < C\rho$  holds true for some constant  $C \geq 0$ .*

*Proof.* Fix the sets  $Z_j(\lambda_i^{R_j})$ ,  $\lambda_i^{R_j} \in \Lambda^{R_j}$ , generated in the last forward pass of the Nested Benders Algorithm. To obtain the desired bound on  $|v - v^L|$ , it is sufficient to show that

$$(5) \quad |\mathcal{Q}_{R_j}(x, \lambda_i^{R_j}) - \mathcal{Q}_{R_j}^{LC}(x, \lambda_i^{R_j})| \leq C_j \rho \quad \forall x \in Z_j(\lambda_i^{R_j}), \quad i = 1, \dots, m_j,$$

for some constants  $C_j$ ,  $j = 0, \dots, n + 1$ .

For  $j = n + 1$  it is  $\mathcal{Q}_{R_{n+1}}(\cdot, \cdot) \equiv \mathcal{Q}_{R_{n+1}}^{LC}(\cdot, \cdot) \equiv 0$  by definition. Let  $C_{n+1} = 0$ . Assume now that (5) holds true for timeperiod  $R_{j+1}$ . Fix  $\lambda_i^{R_j} \in \Lambda^{R_j}$ . Let  $\hat{Z}_j(\lambda_i^{R_j})$  be the set of representative points that are selected from  $Z_j(\lambda_i^{R_j})$  by the thinning algorithm, and let  $\varphi : Z_j(\lambda_i^{R_j}) \rightarrow \hat{Z}_j(\lambda_i^{R_j})$  be the *thinning mapping* which maps every  $x_{R_j} \in Z_j(\lambda_i^{R_j})$  to a close representative point, i.e.,  $|x_i - \varphi(x)_i| < \rho(x_i^{\text{up}} - x_i^{\text{low}})$ . Thus,  $\|x - \varphi(x)\|_\infty < \rho C_\varphi$  holds with  $C_\varphi := \max_i |x_i^{\text{up}} - x_i^{\text{low}}|$ .

Fix  $x \in Z_j(\lambda_i^{R_j})$ . Due to relatively complete recourse,  $\mathcal{Q}_{R_j}(x, \lambda_i^{R_j})$  and  $\mathcal{Q}_{R_j}(\varphi(x), \lambda_i^{R_j})$  are finite and no feasibility cuts are generated. We estimate

$$(6) \quad \left| \mathcal{Q}_{R_j}(x, \lambda_i^{R_j}) - \mathcal{Q}_{R_j}^{LC}(x, \lambda_i^{R_j}) \right| \leq \left| \mathcal{Q}_{R_j}(\varphi(x), \lambda_i^{R_j}) - \mathcal{Q}_{R_j}^{LC}(\varphi(x), \lambda_i^{R_j}) \right| \\ + \left| \mathcal{Q}_{R_j}^{LC}(x, \lambda_i^{R_j}) - \mathcal{Q}_{R_j}^{LC}(\varphi(x), \lambda_i^{R_j}) \right| + \left| \mathcal{Q}_{R_j}(x, \lambda_i^{R_j}) - \mathcal{Q}_{R_j}(\varphi(x), \lambda_i^{R_j}) \right|$$

Since the Nested Benders Algorithm stopped, no optimality cuts have been generated after evaluating  $\mathcal{Q}_{R_j}^L(\varphi(x), \lambda_i^{R_j})$ . Hence,  $\mathcal{Q}_{R_j}^{LC}(\varphi(x), \lambda_i^{R_j}) = \mathcal{Q}_{R_j}^L(\varphi(x), \lambda_i^{R_j})$ . Let  $x^*$  be the optimal solution of problem  $(\mathcal{Q}_{R_j}^L)$  with  $x_{R_j} = \varphi(x)$  that was obtained when  $\mathcal{Q}_{R_j}^L(\varphi(x), \lambda_i^{R_j})$  was evaluated. Thus,  $x_{R_{j+1}}^*(\xi^{R_{j+1}}) \in Z_{j+1}(\lambda^{R_{j+1}}(\xi^{R_{j+1}}))$ . Then we can make use of (5) for timeperiod  $R_{j+1}$  and estimate the first term on the right hand side of (6) by

$$\left| \mathcal{Q}_{R_j}(\varphi(x), \lambda_i^{R_j}) - \mathcal{Q}_{R_j}^{LC}(\varphi(x), \lambda_i^{R_j}) \right| = \mathcal{Q}_{R_j}(\varphi(x), \lambda_i^{R_j}) - \mathcal{Q}_{R_j}^L(\varphi(x), \lambda_i^{R_j}) \\ \leq \mathbb{E} \left[ \mathcal{Q}_{R_{j+1}}(x_{R_{j+1}}^*, \lambda^{R_{j+1}}(\xi^{R_{j+1}})) - \mathcal{Q}_{R_{j+1}}^{LC}(x_{R_{j+1}}^*, \lambda^{R_{j+1}}(\xi^{R_{j+1}})) \mid \xi^{R_j} = \lambda_i^{R_j} \right] \leq C_{j+1}\rho.$$

For the last two terms of (6) we can utilize the Lipschitz continuity of the function  $\mathcal{Q}_{R_j}(\cdot, \lambda_i^{R_j})$  and its approximation  $\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda_i^{R_j})$  and estimate each term by  $L_j\|x - \varphi(x)\|_\infty$  for a Lipschitz constant  $L_j$ .

Thus, we can continue at (6) and obtain

$$\left| \mathcal{Q}_{R_j}^{LC}(x, \lambda_i^{R_j}) - \mathcal{Q}_{R_j}(x, \lambda_i^{R_j}) \right| \leq C_{j+1}\rho + 2L_j\|x - \varphi(x)\|_\infty \leq C_j\rho$$

with  $C_j := C_{j+1} + 2L_jC_\phi$ . □

**Remark 3.** *The relatively complete recourse assumption was needed to avoid that the thinning operation removes a point for which actually a feasibility cut had to be generated. Since the feasible set of  $(\mathcal{Q}_{R_j})$  is convex, this assumption can be dropped if the thinning operation is designed such that no extremal points of  $Z_j(\lambda_i^{R_j})$ 's convex hull are removed from  $Z_j(\lambda_i^{R_j})$ . Then all necessary feasibility cuts will be generated.*

Starting the Nested Benders Decomposition Algorithm with a large thinning parameter  $\rho$ , a rough approximation of the value functions  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  can be obtained. Empirical observations show that this preprocessing may lead to a significant speed-up, cf. Table 3 in Section 4.2. This is due to the fact that the rough approximation produces solution points that are already close to an optimal solution of the problem, and, hence, the generation of too many “useless” cuts during the first iterations of the algorithm can be avoided.

After having roughly solved the problem, the approximation can be improved by decreasing  $\rho$ . By Proposition 2, the algorithm converges to an optimal solution of  $(\mathcal{Q}_0)$  with  $\rho \rightarrow 0$ . Unfortunately, decreasing  $\rho$  until an error tolerance on the thinning error is satisfied leads to large sets  $Z_j(\lambda_i^{R_j})$  again. A reliable and better adapted stopping criterion is discussed in Section 2.3.

### 2.3 Upper bounding approximation

As discussed above, the thinning operation allows a rough approximation of the functions  $\mathcal{Q}_{R_j}^L(\cdot, \xi_i^{R_j})$ , but it is not clear how much the parameter  $\rho$  has to be decreased to ensure a solution within a given error tolerance. As an adapted stopping criterion we propose to compute a gap between lower and upper bounds on the functions  $\mathcal{Q}_{R_j}(\cdot, \xi_i^{R_j})$  and to stop when this gap is sufficiently small. While the lower bounds are given by the supporting hyperplane approximations  $\mathcal{Q}_{R_j}^{LC}(\cdot, \xi_i^{R_j})$  of  $\mathcal{Q}_{R_j}^L(\cdot, \xi_i^{R_j})$ , the upper bounds can be constructed from convex combinations of points where  $\mathcal{Q}_{R_j}(\cdot, \xi_i^{R_j})$  (or an upper bound on it) has been already evaluated.

Hence, similar to the underestimating functions  $\mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j})$  and  $\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda_i^{R_j})$ , we define overestimating functions by replacing the cost-to-go functions in the Bellman Equation ( $\mathcal{Q}_{R_j}$ ) by approximations using convex combinations of already evaluated points. More precisely, we define recursively the following overestimating functions: Let  $\mathcal{Q}_{R_{n+1}}^{UC}(\cdot, \cdot) := 0$  and for  $j = n, \dots, 0$ , let

$$\begin{aligned} \mathcal{Q}_{R_j}^U(x_{R_j}, \lambda_i^{R_j}) &:= \min \mathbb{E} \left[ \sum_{t=R_j+1}^{R_{j+1}} \langle b_t(\xi_t), x_t \rangle + \mathcal{Q}_{R_{j+1}}^{UC}(x_{R_{j+1}}, \lambda^{R_{j+1}}(\xi^{R_{j+1}})) \mid \xi^{R_j} = \lambda_i^{R_j} \right] \\ \text{s.t. } x_t &\in X_t, \quad x_t \in \mathcal{F}_t, \quad t = R_j + 1, \dots, R_{j+1}, \\ A_{t,0}(\xi_t)x_t + A_{t,1}(\xi_t)x_{t-1} &= h_t(\xi_t), \quad t = R_j + 1, \dots, R_{j+1}. \end{aligned}$$

For  $j = 1, \dots, n$ , let  $Y_j(\lambda_i^{R_j}) \subset X_{R_j}$ ,  $\lambda_i^{R_j} \in \Lambda^{R_j}$ , be finite sets of control points. Then the upper bounding approximation  $\mathcal{Q}_{R_j}^{UC}(\cdot, \lambda_i^{R_j})$  of  $\mathcal{Q}_{R_j}^U(\cdot, \lambda_i^{R_j})$  is defined by interpolating from the values of  $\mathcal{Q}_{R_j}^U(\cdot, \lambda_i^{R_j})$  in  $Y_j(\lambda_i^{R_j})$ :

$$\begin{aligned} \mathcal{Q}_{R_j}^{UC}(x_{R_j}, \lambda_i^{R_j}) &:= \min_{\alpha} \sum_{y \in Y_j(\lambda_i^{R_j})} \alpha_y \mathcal{Q}_{R_j}^U(y, \lambda_i^{R_j}) \\ \text{s.t. } x_{R_j} &= \sum_{y \in Y_j(\lambda_i^{R_j})} \alpha_y y \\ \alpha &\in S_{|Y_j(\lambda_i^{R_j})|}, \end{aligned}$$

where  $S_n$  is the  $(n - 1)$ -dimensional standard simplex in  $\mathbb{R}^n$  ( $n \in \mathbb{N}$ ). Note, that the domain of  $\mathcal{Q}_{R_j}^{UC}(\cdot, \lambda_i^{R_j})$  is the convex hull of  $Y_j(\lambda_i^{R_j})$ . Hence, to obtain meaningful bounds, the sets  $Y_j(\lambda_i^{R_j})$  have to be large enough. In the final algorithm, these sets are generated in an adaptive way similarly to the selection of points where optimality cuts are generated.

### 2.4 Full algorithm

We now incorporate the thinning operation and upper bound approximations in the nodal approximation algorithm. From the discussion above, we have the relation

$$\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda_i^{R_j}) \leq \mathcal{Q}_{R_j}^L(\cdot, \lambda_i^{R_j}) \leq \mathcal{Q}_{R_j}(\cdot, \lambda_i^{R_j}) \leq \mathcal{Q}_{R_j}^U(\cdot, \lambda_i^{R_j}) \leq \mathcal{Q}_{R_j}^{UC}(\cdot, \lambda_i^{R_j}).$$

Hence, for a point  $x_{R_j} \in X_{R_j}$ , the creation of control points for time stage  $R_{j+1}$  in forward mode can be omitted if the gap  $\mathcal{Q}_{R_j}^{UC}(x_{R_j}, \lambda_i^{R_j}) - \mathcal{Q}_{R_j}^{LC}(x_{R_j}, \lambda_i^{R_j})$  falls below a given tolerance  $\varepsilon > 0$ . Further, a small gap  $\mathcal{Q}_0^U - \mathcal{Q}_0^L$  ensures that the approximation ( $\mathcal{Q}_0^L$ ) is very close to the original problem ( $\mathcal{Q}_0$ ) and that the algorithm can be stopped before the creation of cuts has ended.

Algorithm 1 states the complete algorithm that is performed for a node  $\lambda_i^{R_j} \in \Lambda^{R_j}$  at time stage  $R_j$ . We write  $a \gg b$  if  $a - b > \varepsilon$ .

The order in which the master problems ( $\mathcal{Q}_{R_j}^L$ ) are handled is determined by the *sequencing protocol*. We implemented a fast-forward-fast-backward procedure [2, 11] to traverse the tree. In the forward mode, master problems are solved and control point sets  $Z_j(\lambda_i^{R_j})$  are generated. The traversing direction is changed into the backward mode when the end of the tree is reached (If a master problem is infeasible, the algorithm switches temporarily into a special feasibility restoration mode). In backward mode, optimality cuts are generated and upper bound approximations are updated.

**Algorithm 1** (*Nodal approximation algorithm*). Let  $\lambda := \lambda_i^{R_j} \in \Lambda^{R_j}$ .

If  $j > 0$  we perform the following algorithm:

Thin out  $Z_j(\lambda)$ .

**if** we are in forward mode **or**  $\mathcal{Q}_{R_j}^L(\cdot, \lambda)$  or  $\mathcal{Q}_{R_j}^U(\cdot, \lambda)$  have been updated **then**

**for each**  $x_{R_j} \in Z_j(\lambda)$  **do**

    Compute or update a lower bound  $l := \mathcal{Q}_{R_j}^{LC}(x_{R_j}, \lambda)$  and upper bound  $u := \mathcal{Q}_{R_j}^{UC}(x_{R_j}, \lambda)$  on  $\mathcal{Q}_{R_j}(x_{R_j}, \lambda)$ .

**if**  $u \gg l$  **then**

        Compute  $\mathcal{Q}_{R_j}^L(x_{R_j}, \lambda)$ .

**if**  $\mathcal{Q}_{R_j}^L(x_{R_j}, \lambda) = \infty$  **then**

            Construct a feasibility cut and update  $\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda)$ . **return.**

**else if**  $\mathcal{Q}_{R_j}^L(x_{R_j}, \lambda) \gg \mathcal{Q}_{R_j}^{LC}(x_{R_j}, \lambda)$  **then**

            Construct an optimality cut and update  $\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda)$ .

            Update  $l := \mathcal{Q}_{R_j}^L(x_{R_j}, \lambda)$ .

**end if**

**end if**

**if**  $u \gg l$  **then**

        Compute  $\mathcal{Q}_{R_j}^U(x_{R_j}, \lambda)$ .

**if**  $\mathcal{Q}_{R_j}^{UC}(x_{R_j}, \lambda) \gg \mathcal{Q}_{R_j}^U(x_{R_j}, \lambda)$  **then**

            Update  $\mathcal{Q}_{R_j}^{UC}(\cdot, \lambda)$  by adding  $x_{R_j}$  to  $Y_j(\lambda)$ .

            Update  $u := \mathcal{Q}_{R_j}^U(x_{R_j}, \lambda)$ .

**end if**

**if** forward mode **and**  $j < n$  **and**  $u \gg l$  **then**

        Add the points  $x_{R_{j+1}}(\xi^{R_{j+1}})$  from the solution of ( $\mathcal{Q}_{R_j}^L$ ) to  $Z_{j+1}(\lambda^{R_{j+1}}(\xi^{R_{j+1}}))$ .

```

    end if
  end if
end for
end if
if backward mode then
   $Z_j(\lambda) := \emptyset.$ 
end if

```

If  $j = 0$  everything simplifies to:

```

if forward mode then
  if ( $\mathcal{Q}_0^L$ ) has been updated then
    Solve ( $\mathcal{Q}_0^L$ ) and update the lower bound  $l$ .
    if ( $\mathcal{Q}_0^L$ ) is infeasible then
      ( $\mathcal{Q}_0$ ) is infeasible. return.
    end if
  end if
  if ( $\mathcal{Q}_0^U$ ) has been updated then
    Solve ( $\mathcal{Q}_0^U$ ) and update the upper bound  $u$ .
  end if
  if  $u \gg l$  then
    Add the points  $x_{R_1}(\boldsymbol{\xi}^{R_1})$  from the solution of ( $\mathcal{Q}_0^L$ ) to  $Z_1(\boldsymbol{\lambda}^{R_1}(\boldsymbol{\xi}^{R_1}))$ .
  end if
end if

```

We conclude this section with some notes on the main loop of the solution algorithm:

**Remark 4.** •  $\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda_i^{R_j})$  and  $\mathcal{Q}_{R_j}^{UC}(\cdot, \lambda_i^{R_j})$  are initialized by  $\mathcal{Q}_{R_j}^{LC}(\cdot, \lambda_i^{R_j}) := -M$  and  $Y_j(\lambda_i^{R_j}) := \{x^{\text{mid}}\}$  with  $x^{\text{mid}} := \frac{1}{2}(x^{\text{low}} + x^{\text{up}})$  the midpoint of  $X_{R_j}$  and  $\mathcal{Q}_{R_j}^U(x^{\text{mid}}, \lambda_i^{R_j}) := M$ , where the value  $M$  is sufficiently large.

- In the so-called rough phase we improve our approximations by traversing between the timeperiods with a very rough thinning ( $\rho = 0.1$ ) of the sets  $Z_j(\lambda_i^{R_j})$ . Finally, to assure that the approximation error does not exceed a specified tolerance, we continue the fast-forward-fast-backward traversing with a decreasing sequence of  $\rho$ .
- For small sets  $Y_j(\lambda_i^{R_j})$  as they appear in the first iterations of the algorithm, no or only weak upper bounds are available. However, tight upper bounds are not needed during the rough phase, and from our experimental results we can conclude that useful upper bounds are obtained after the rough phase already.

### 3 Generation of a recombining scenario tree

A variety of approaches for the generation of scenario trees have been developed, relying on different principles, e.g. in [1, 4, 6, 7, 8, 16, 18, 20]. The *forward tree construction* of [16] generates a scenario tree out of a given stochastic process  $\zeta$  by the iterative approximation of the conditional distributions

$$(7) \quad \mathbb{P} [\zeta_t \in \cdot \mid \zeta^{t-1} \in C_{t-1}^u],$$

given a finite partition  $(C_{t-1}^u)_{u \in \mathbb{N}^{t-1}}$  of  $\text{supp } \mathbb{P} [\zeta^{t-1} \in \cdot]$ . This is done by choosing points  $(v_t^{(u,i)}, i = 1, \dots, n_t^u)$  and a corresponding Voronoi partition  $(V_t^{(u,i)}, i = 1, \dots, n_t^u)$  of the support of (7). Thereby, the  $(t-1)$ -dimensional multiindex  $u$  describes the node of the tree at time  $t-1$ ,  $n_t^u$  is the branching degree of node  $u$ , and

$$C_t^{(u,i)} := C_{t-1}^u \times V_t^{(u,i)}.$$

Let  $\xi$  be a process whose distribution is given by the scenario tree, then the latter is defined by the conditional probabilities

$$(8) \quad \mathbb{P} [\xi_t = v_t^{(u,i)} \mid \xi^{t-1} = (v_1^{u_1}, \dots, v_{t-1}^{u_{t-1}})] := \mathbb{P} [\zeta_t \in V_t^{u,i} \mid \zeta^{t-1} \in C_{t-1}^u].$$

A recombining tree out of a Markov process can be constructed by approximating the marginal distributions  $\mathbb{P} [\zeta_t \in \cdot]$  instead of (7). More precisely, one selects the points  $v_t^{(u,i)}$  independently of  $u$  and the transition probabilities (8) depending only on  $u_{t-1}$ , cf. [1].

For constructing a recombining tree which approximates a non-Markovian stochastic process, we propose the following approach, that can be seen as a mixture of the proceedings in [1] and [16]. Basically, it consists of constructing non-recombining trees for every time interval  $[R_j + 1, R_{j+1}]$ . Recombining scenarios at time  $R_j$  means to assign the same subtree to several nodes at time  $R_j$ . The particular subtree has to approximate  $\zeta$ 's future distribution that can depend, in the non-Markovian case, on  $\zeta$ 's complete history. We want two nodes at time  $R_j$  to obtain the same subtree if  $\zeta$ 's values in these nodes are close during the time interval  $[R_j - k, \dots, R_j - 1]$ , for some value  $k \geq 1$ . This seems to be reasonable whenever  $\mathbb{P} [\zeta^T \in \cdot \mid \zeta^{t-1} = \zeta^{t-1}]$  depends continuously on  $(\zeta_{t-k}, \dots, \zeta_{t-1})$ , in some sense, see also Remark 6 below. The latter is fulfilled if  $\zeta$  has both the continuity assumed in [1] and a *short-term memory*:

**Assumption 1.** *There exists  $k \in \mathbb{N}$  with  $k < R_j - R_{j-1}$  for every  $j$ , such that we have for every  $R_j$  and for every  $B \in \mathcal{B}(\mathbb{R}^{(T-R_j) \cdot s})$*

$$\mathbb{P} [(\zeta_t)_{t=R_j+1, \dots, T} \in B \mid (\zeta_t)_{t=1, \dots, R_j}] = \mathbb{P} [(\zeta_t)_{t=R_j+1, \dots, T} \in B \mid (\zeta_t)_{t=R_j-k+1, \dots, R_j}].$$

**Remark 5.** *Short-term memory is a generalization of the Markov property. It is fulfilled by a variety of processes of practical interest and can be verified easily if  $\zeta$ 's distribution is given as a time series model, in general. It is easy to see that, by augmentation of its state space, every discrete-time process  $\zeta$  with short-term memory may be transformed into a Markov process.*

**Remark 6.** *As the approaches of [1] and [18], our tree generation method relies on the continuity of  $\zeta^{t-1} \mapsto \mathbb{P}[\zeta^T \in \cdot \mid \zeta^{t-1} = \zeta^{t-1}]$ . This is illustrated by Example 2.6. of [17]. Since a rigorous consistency analysis for recombining trees lies beyond the scope of this paper, we refrain from a proper definition of the claimed continuity.*

Our approach to construct a recombining scenario tree reads as follows:

**Algorithm 2** (*Generation of a recombining scenario tree*).

1. *Initialization:* Set  $C_1^1 = \{\zeta_1\}$ ,  $n_1^1 = 1$ .
2. *Tree generation until the first recombination:*  
 For  $t = 2, \dots, R_1$ : For every multiindex  $u = (u_1, \dots, u_{t-1})$  with  $u_s = 1, \dots, n_s^{(u_1, \dots, u_{s-1})}$  and  $s = 1, \dots, t-1$ : Approximate  $\mathbb{P}[\zeta_t \in \cdot \mid \zeta^{t-1} \in C_{t-1}^u]$  through the choice of cluster centers  $v_t^{(u,i)}$  and a corresponding Voronoi partition  $V_t^{(u,i)}$ ,  $i = 1, \dots, n_t^u$ . Set  $C_t^{(u,i)} := C_{t-1}^u \times V_t^{(u,i)}$  and define  $\xi$ 's transition probabilities via (8).
3. *Subtree generation:*  
 For  $t = R_1 + 1, \dots, T$ :
  - (a) Consider  $\underline{t} \in \{1, \dots, n\}$  such that  $R_{\underline{t}} < t \leq R_{\underline{t}+1}$ , i.e.,  $R_{\underline{t}}$  is the latest recombination time (strictly) before  $t$ .
  - (b) If  $t = R_{\underline{t}} + 1$ : (*short-term history clustering*)  
 Divide  $\text{supp } \mathbb{P}[(\zeta_{R_{\underline{t}}-k+1}, \dots, \zeta_{R_{\underline{t}}}) \in \cdot]$  into a Voronoi partition  $C_{R_{\underline{t}}}^i$ ,  $i = 1, \dots, m_{R_{\underline{t}}}$ .
  - (c) For every multiindex  $u = (u_{R_{\underline{t}}}, \dots, u_{t-1})$  with  $u_{R_{\underline{t}}} = 1, \dots, m_{R_{\underline{t}}}$  and  $u_s = 1, \dots, n_s^{(u_{R_{\underline{t}}}, \dots, u_{s-1})}$  for  $s = R_{\underline{t}} + 1, \dots, t-1$ :  
 Approximate  $\mathbb{P}[\zeta_t \in \cdot \mid (\zeta_{R_{\underline{t}}-k+1}, \dots, \zeta_{t-1}) \in C_{t-1}^u]$  by the choice of points  $v_t^{(u,i)}$  and a corresponding Voronoi partition  $V_t^{(u,i)}$ ,  $i = 1, \dots, n_t^u$ . Set  $C_t^{(u,i)} := C_{t-1}^u \times V_t^{(u,i)}$  and define  $\xi$ 's transition probabilities by

$$\begin{aligned} & \mathbb{P}[\xi_t = v_t^{(u,i)} \mid \xi^{t-1} = (v_1^{u_1}, \dots, v_{t-1}^{u_{t-1}})] \\ & := \mathbb{P}[\xi_t = v_t^{(u,i)} \mid (\xi_{R_{\underline{t}}-k+1}, \dots, \xi_{t-1}) \in C_{R_{\underline{t}}}^{u_{R_{\underline{t}}}} \times (v_{R_{\underline{t}}+1}^{u_{R_{\underline{t}}+1}}, \dots, v_{t-1}^{u_{t-1}})] \\ & := \mathbb{P}[\zeta_t \in V_t^{(u,i)} \mid (\zeta_{R_{\underline{t}}-k+1}, \dots, \zeta_{t-1}) \in C_{t-1}^u], \end{aligned}$$

whenever  $(v_{R_{\underline{t}}-k+1}^{u_{R_{\underline{t}}-k+1}}, \dots, v_{R_{\underline{t}}}^{u_{R_{\underline{t}}}}) \in C_{R_{\underline{t}}}^{u_{R_{\underline{t}}}}$ , and equal to 0 else.

**Remark 7.** *Algorithm 2 assigns two nodes at time  $R_j$  the same subtree, whenever their short term histories until  $R_j$  fall into the same cluster  $C_{R_t}^j$  determined in Step 3b. The numbers  $n_t^u$  determine the branching degree at node  $u$  and, using the notation of Section 1.2,  $m_{R_j}$  represents the number of different subtrees originated at time  $R_j$ . These values, determining the structure of the scenario tree, can either be predefined, or, as proposed by [16], determined within the discretization procedures to not exceed certain local error levels.*

We did not specify *how* to carry out the approximation within the Steps 2, 3b, and 3c. For this purpose, the above cited tree generation algorithms use a stochastic gradient method [1] and a heuristically motivated successive choice of  $v_t^u$  out of a set of simulated trajectories [16], respectively. Both can be applied in our framework, too. Furthermore, there is a huge amount of publications dealing with this question, cf. the citations at the beginning of this section, the monographies [12] and [13], and the references therein.

## 4 Numerical example

### 4.1 A simple power scheduling problem

We consider a power generating system consisting of several coal fired thermal units (index set  $I$ ), pumped hydro units (index set  $J$ ), and a wind power plant. The objective is to find cost-optimal operation levels of the thermal units and hydro units under uncertain production of electricity from wind.

Denote by  $p_{i,t}$  the operation level of the thermal unit  $i \in I$ , by  $l_{j,t}$  the fill level of the water reservoir  $j \in J$ , by  $w_{j,t}$  the operation level of the pump  $j \in J$ , and by  $v_{j,t}$  the operation level of the turbine  $j \in J$ . Deterministic parameters of the problem are operation ranges for the thermal units  $\underline{p}_i < \bar{p}_i$ ,  $i \in I$ , the pumps  $\bar{w}_j > 0$ , and the turbines  $\bar{v}_j > 0$ , the capacity of the water reservoirs  $\bar{l}_j > 0$ ,  $j \in J$ , the fill levels of the reservoirs at the beginning and end of the considered time horizon ( $l_{j,in}$  and  $l_{j,end}$ ,  $j \in J$ ), the efficiency of the pumps  $\eta_j$ ,  $j \in J$ , fuel costs  $b_i$ ,  $i \in I$ , and the energy demand  $d_t$ . As stochastic parameter we consider the wind power production  $\kappa_t$ .

The complete model has the form

$$\begin{aligned}
 & \min \quad \mathbb{E} \left[ \sum_{t=1}^T \sum_{i \in I} b_i p_{i,t} \right] \\
 (9) \quad & \text{s.t.} \quad l_{j,1} = l_{j,in} - (w_{j,1} - \eta_j v_{j,1}), \quad l_{j,T} = l_{j,end} \quad j \in J, \\
 (10) \quad & l_{j,t} = l_{j,t-1} - (w_{j,t} - \eta_j v_{j,t}), \quad t = 2, \dots, T, \quad j \in J, \\
 (11) \quad & |p_{i,t} - p_{i,t-1}| \leq \frac{1}{4}(\bar{p}_i - \underline{p}_i), \quad t = 2, \dots, T, \quad i \in I, \\
 (12) \quad & \sum_{i \in I} p_{i,t} + \sum_{j \in J} (w_{j,t} - v_{j,t}) + \kappa_t \geq d_t, \quad t = 1, \dots, T, \\
 (13) \quad & \sum_{i \in I} p_{i,t} \leq \sum_{i \in I} \bar{p}_i - \frac{1}{10}d_t, \quad t = 1, \dots, T,
 \end{aligned}$$

$$\underline{p}_i \leq p_{i,t} \leq \bar{p}_i, \quad 0 \leq v_{j,t} \leq \bar{v}_j, \quad 0 \leq w_{j,t} \leq \bar{w}_j, \quad 0 \leq l_{j,t} \leq \bar{l}_j, \quad i \in I, \quad j \in J, \quad t = 1, \dots, T.$$

Constraint (9) models the initial and final fill level of the reservoirs, (10) couples the fill levels of the reservoirs between successive time stages, (11) bounds the change in the operation of the thermal units between successive time stages, (12) ensures that the electricity demand is covered, and (13) is a reserve requirement.

For our numerical experiments we considered different time horizons between 2 days and several months in *hourly discretization*. The stochastic wind speed process was modeled by a GARCH-M time series model [9], that, in particular, exhibits the short-term memory claimed in Section 3. It was used to create a binary *wind speed scenario tree* with branching 3 times a day at 6, 12, and 18 o'clock. Recombination takes place once a day at 6 o'clock. The tree was generated as sketched in Section 3 and we used a  $k$ -mean clustering algorithm to approximate the (conditional) distributions. The resulting tree was transformed to a *wind energy scenario tree* by using an *aggregated power curve* [19].

Due to the form of the tree, for each master problem ( $\mathcal{Q}_{R_j}$ ) in the Nester Bender Decomposition a scenario tree consisting of 8 scenarios is used. Note, that a non-recombining scenario tree for 1 week with 3 branches per day consists of 39.546.276 nodes, while by recombination the number of nodes reduces to 924, 1716, 3300, or 6468 for 1, 2, 4, or 8 subtrees per timeperiod (day), respectively.

## 4.2 Numerical results

The following results have been realized on a Pentium IV 3 GHz machine with 1 GB RAM and Linux 2.6.11. All master problems have been solved with CPLEX 10.0.

First, we investigated how the efficiency of the Nested Benders Decomposition improves when it can use the coincidence of several subtrees, i.e., that relation (1) is fulfilled at one or several time stages  $R_j$ ,  $j = 1, \dots, n$ . To this end, we considered time horizons of 2, 3, and 4 days, respectively. We used once a Nested Benders Decomposition Algorithm

which decomposes the linear stochastic program every 24 hours, but without making use of the recombining nature of the scenario tree, i.e., equal cost-to-go are not approximated simultaneously. Thereon we run the decomposition algorithm on the recombining tree, i.e., we allowed for simultaneous approximation of coinciding cost-to-go functions.

Table 1 summarizes the results of this experiment. The first column gives the overall time horizon considered. Recall, that the deterministic equivalent is decomposed once per day. The second column is the number of different subtrees per timeperiod. The third and fifth column reports the total number of master problems that are considered after the decomposition. It can be seen that recombining scenarios avoids an exponential growth of the number of different subproblems, and, thus, significantly reduces the running time of the Nested Benders Decomposition Algorithm.

| time horizon | $ \Lambda^{R_j} $ | no recombination  |      | with recombination |      |
|--------------|-------------------|-------------------|------|--------------------|------|
|              |                   | # master problems | time | # master problems  | time |
| 2 days       | 1                 | 9                 | 10s  | 2                  | 3s   |
| 2 days       | 2                 | 9                 | 10s  | 3                  | 4s   |
| 2 days       | 4                 | 9                 | 12s  | 5                  | 7s   |
| 3 days       | 1                 | 73                | 91s  | 3                  | 3s   |
| 3 days       | 2                 | 73                | 99s  | 5                  | 5s   |
| 3 days       | 4                 | 73                | 94s  | 9                  | 9s   |
| 4 days       | 1                 | 585               | 762s | 4                  | 4s   |
| 4 days       | 2                 | 585               | 859s | 7                  | 6s   |
| 4 days       | 4                 | 585               | 789s | 13                 | 13s  |

Table 1: Performance of Nested Benders Decomposition Algorithm on non-recombining and recombining scenario trees

Next, we study the algorithm’s potential for optimization over longer time horizons and the performance of the control space aggregations from the Sections 2.2 and 2.3, see Table 2 for the results. The columns entitled with *rough* report the time spent for the rough phase only, i.e., using *thinning with*  $\rho = 0.1$ . The rough phase turns out to be very fast and its running time appears to depend linearly on the length of the considered time horizon as well as on the number of different subtrees assigned to every recombining time stage. In most cases, the rough phase provides solutions that are close to optimal solutions and that do not significantly change during the remaining optimization procedure. Thus, in practice, it may be reasonable to reduce the algorithm’s running time by putting the final  $\rho$  not too small.

The calculation of upper bounds, detailed in Section 2.3, gives the possibility of adaptively choosing  $\rho$  such that a certain error level is not exceeded. The column *rough phase gap* reports the relative gap between lower and upper bounds *after having completed the*

*rough phase*. The columns entitled with *complete* report the complete solution time, i.e., the time spent to decrease  $\rho$  to the value given in the column *final*  $\rho$ . It can be seen that with diminishing  $\rho$  the non-recombining nature of the decision process, which is due to time-coupling constraints, leads to an exponential growth of the number of control points and, consequently, of the running times.

| time horizon | $ \Lambda^{R_j} $ | final $\rho$ | no upper bounds |          | with upper bounds |                 |          |
|--------------|-------------------|--------------|-----------------|----------|-------------------|-----------------|----------|
|              |                   |              | rough           | complete | rough             | rough phase gap | complete |
| 2 days       | 1                 | 0.0001       | 2s              | 2s       | 3s                | < 0.01%         | 3s       |
| 2 days       | 2                 | 0.0001       | 4s              | 4s       | 4s                | < 0.01%         | 4s       |
| 2 days       | 4                 | 0.0001       | 6s              | 6s       | 6s                | < 0.01%         | 6s       |
| 1 week       | 1                 | 0.0001       | 3s              | 17s      | 7s                | 0.02%           | 10s      |
| 1 week       | 2                 | 0.0001       | 7s              | 17s      | 13s               | 0.05%           | 15s      |
| 1 week       | 4                 | 0.0001       | 13s             | 26s      | 26s               | 0.03%           | 34s      |
| 2 weeks      | 1                 | 0.001        | 5s              | 47s      | 16s               | 0.02%           | 33s      |
| 2 weeks      | 2                 | 0.001        | 11s             | 711s     | 27s               | 0.04%           | 536s     |
| 2 weeks      | 4                 | 0.001        | 21s             | 2512s    | 57s               | 0.03%           | 4535s    |
| 1 month      | 1                 | 0.001        | 7s              | 47s      | 27s               | 0.01%           | 28s      |
| 1 month      | 2                 | 0.001        | 21s             | >3h      | 59s               | 0.12%           | >3h      |
| 1 month      | 4                 | 0.001        | 35s             | >3h      | 151s              | 0.03%           | >3h      |
| 3 months     | 1                 | 0.001        | 19s             | >3h      | 68s               | 0.05%           | 507s     |
| 3 months     | 2                 | 0.001        | 60s             | >3h      | 195s              | 0.09%           | 829s     |
| 3 months     | 4                 | 0.001        | 60s             | >3h      | 868s              | 0.09%           | >3h      |
| 1 year       | 1                 | 0.01         | 37s             | >3h      | 149s              | 1.12%           | >3h      |

Table 2: Performance for different time horizons with and without the use of upper bounds

Finally, we have investigated the gain of starting the thinning in Section 2.2 with a large value of  $\rho$  and then decreasing it, instead of starting with the final value of  $\rho = 0.0001$  from the beginning on. Table 3 shows the results. The advantage of a successive decreasing of  $\rho$  is obvious, especially when the problem size increases. For the decreasing sequence, we start with  $\rho = 0.1$  and multiply  $\rho$  by 0.3 everytime that the approximations at the first timeperiod problem are not changing until  $\rho$  falls below 0.0001.

## 5 Further Developments

As the numerical results show, the adaption of a Nested Benders Decomposition Algorithm to recombining scenario trees allows to handle long time horizons with a reasonable representation of the stochastic process and enables a considerable reduction of the computing times. Of course, further improvements of the algorithm's performance seem to be

| time horizon | $ \Lambda^{R_j} $ | decreasing $\rho$ sequence | start with final $\rho$ |
|--------------|-------------------|----------------------------|-------------------------|
| 3 days       | 1                 | 3s                         | 3s                      |
| 3 days       | 2                 | 5s                         | 5s                      |
| 3 days       | 4                 | 9s                         | 10s                     |
| 5 days       | 1                 | 4s                         | 15s                     |
| 5 days       | 2                 | 9s                         | 22s                     |
| 5 days       | 4                 | 20s                        | 31s                     |
| 1 week       | 1                 | 10s                        | 51s                     |
| 1 week       | 2                 | 15s                        | 1685s                   |
| 1 week       | 4                 | 34s                        | 1960s                   |

Table 3: Performance for a decreasing sequence of thinning tolerances  $\rho$  and a stationary value of  $\rho$

possible. Note, that solving the master problem ( $\mathcal{Q}_{R_j}^L$ ) for all points from a set  $Z_j(\lambda_i^{R_j})$  means the solution of many linear programs that differ only in their right hand sides. Hence, a bunching procedure as introduced by Wets [28] and further developed by other authors [2, 11, 15] allows to find an ordering of the set  $Z_j(\lambda_i^{R_j})$  that can significantly decrease the running time. Also the choice of the sequencing protocol is not a trivial task [11] and can substantially influence the algorithmic performance. Furthermore, it would be desirable to investigate whether a more adaptive and time stage dependent choice of the thinning parameter  $\rho$  can decrease the number of control points when longer time horizons are considered.

A desirable extension, that would address many of today's applications where discrete decisions have to be modeled, would be the support of mixed-integer variables in the first and later timeperiod problems. While the integration into the first stage problem does not require a change in the methodology, the appearance of mixed-integer variables at later time stages changes the situation drastically since one loses the convexity of the cost-to-go functions  $\mathcal{Q}_{R_j}(\cdot, \lambda_i^{R_j})$ , which is a basic assumption for a Nested Benders Decomposition Algorithm. However, promising approaches that might allow to overcome this difficulty are, e.g., the work of Higle and Sen [24] (see also [23, Section 5] for a summary) and Sherali and Sen [25], but both deal only with the two-stage case yet. An extension of their approach to the multistage case, especially in connection with recombining scenario trees which rely on a decomposition between timesteps, would be very interesting.

## References

- [1] V. Bally, G. Pagès, and J. Printems. A quantization tree method for pricing and hedging multidimensional american options. *Mathematical Finance*, 15(1):119–168, 2005.

- [2] J.R. Birge. Decomposition and partitioning methods for multistage stochastic programming. *Operations Research*, 33(5):989–1007, 1985.
- [3] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research. Springer-Verlag, 1997.
- [4] M.S. Casey and S. Sen. The scenario generation algorithm for multistage stochastic linear programming. *Mathematics of Operations Research*, 30:615–631, 2005.
- [5] J. Cox, S. Ross, and M. Rubinstein. Option pricing: a simplified approach. *Journal on Financial Economics*, 7:229–263, 1979.
- [6] M.A.H. Dempster. Sequential importance sampling algorithms for dynamic stochastic programs. *Zapiski Nauchnykh Seminarov POMI*, 312:94–129, 2004.
- [7] J. Dupačová, G. Consigli, and S.W. Wallace. Scenarios for multistage stochastic programming. *Annals of Operations Research*, 100:25–53, 2000.
- [8] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. Scenarios reduction in stochastic programming: An approach using probability metrics. *Mathematical Programming*, 95(A):493–511, 2003.
- [9] B.T. Ewing, J.B. Kruse, and J.L. Schroeder. Time series analysis of wind speed with time-varying turbulence. Technical report, 2004. available at <http://www.ecu.edu/hazards/reports.htm>.
- [10] M.R. Garey and D.S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.
- [11] H.I. Gassmann. MSLiP: a computer code for the multistage stochastic linear programming problem. *Mathematical Programming*, 47:407–423, 1990.
- [12] A. Gersho and R.M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Press, 1992.
- [13] S. Graf and H. Luschgy. *Foundations of Quantization for Probability Distributions*, volume 1730 of *Lecture Notes in Mathematics*. Springer, New York, 2000.
- [14] N. Gröwe-Kuska, H. Heitsch, and W. Römisch. Scenario reduction and scenario tree construction for power management problems. In A. Borghetti, C.A. Nucci, and M. Paolone, editors, *IEEE Bologna Power Tech Proceedings*, 2003.
- [15] D. Haugland and S.W. Wallace. Solving many linear programs that differ only in the righthand side. *European Journal of Operational Research*, 37(3):318–324, 1988.
- [16] H. Heitsch and W. Römisch. Scenario tree modelling for multistage stochastic programs. Preprint 296, DFG Research Center Matheon "Mathematics for key technologies" and submitted, 2005.

- [17] H. Heitsch, W. Römisch, and C. Strugarek. Stability of multistage stochastic programs. *SIAM Journal on Optimization*, 17:511–525, 2006.
- [18] R. Mirkov and G.Ch. Pflug. Tree approximations of dynamic stochastic programs. *submitted*, 2006.
- [19] P. Nørgård (ed.). Fluctuations and predictability of wind and hydropower. Technical report, WILMAR, Risø National Laboratory, 2004. <http://www.wilmar.risoe.dk/Results.htm>.
- [20] G.Ch. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89:251–271, 2001.
- [21] A. Ruszczyński. *Decomposition Methods*, chapter 3, pages 141–211. In Ruszczyński and Shapiro [22], 2003.
- [22] A. Ruszczyński and A. Shapiro, editors. *Stochastic Programming*. Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, 2003.
- [23] R. Schultz. Stochastic programming with integer variables. *Mathematical Programming*, 97(1-2):285–309, 2003.
- [24] S. Sen and J.L. Hige. The  $C^3$  theorem and a  $D^2$  algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20, 2005.
- [25] S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(A):203–223, 2006.
- [26] S.P. Sethi and G. Sorger. A theory of rolling horizon decision making. *Annals of Operations Research*, 29:387–416, 1991.
- [27] R.M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal of Applied Mathematics*, 17(4):638–663, 1969.
- [28] R. Wets. Solving stochastic programs with simple recourse. *Stochastics*, 10:219–242, 1983.