Adaptive discretization of stochastic programs

Diploma thesis

Stefan Vigerske

Humboldt-University to Berlin
Faculty of Mathematics and Natural Sciences II
Department of Mathematics

Tutor: Prof. Dr. W. Römisch

16th June 2005

Preface

Multistage stochastic programs build a powerful tool to model optimization problems where parts of the data form a stochastic process. In many cases, the associated probability distribution is not completely known or has an infinite support. To handle such optimization problems numerically, the distribution has to be discretized by selecting representative scenarios, i.e. realizations of the stochastic process, and supplying them with probability weights. Such a discretization leads to a finite-dimensional optimization problem, but it's size may still be too large to be tractable by today's optimizers. Therefore, the number of scenarios has to be reduced further, such that the problem becomes efficiently solvable, but does not deviate too much from it's origin.

Such a scenario reduction algorithm is the topic of this work. It integrates the reduction of scenario-trees into the solution process by starting with a rough approximation of the initial scenario tree and refining this approximation as long as necessary. To estimate the quality of a computed discretization, relaxation techniques are proposed.

The developed algorithm has been implemented for linear multistage stochastic programs. We present some numerical results for scenario reduction in the setting of power management problems with uncertain load.

Many thanks go to Prof. Werner Römisch and Dr. Ivo Nowak for their support and suggestions, to Holger Heitsch for supplying me with the power management model and associated scenario trees, and to many friends for their motivation and encouragement.

Contents

1 Introduction **7**

2 An abstract adaptive discretization algorithm **11**

2.1 The Problem 11

2.2 Aggregation 12

2.3 Optimal refinement 12

2.4 Updating the convex relaxation 15

2.4.1 Lagrangian decomposition 16

2.4.2 Column generation 17

2.4.3 Aggregated case 19

2.5 Adaptive discretization algorithm 21

3 An adaptive discretization algorithm for multistage stochastic programs **23**

3.1 Deterministic equivalent, stability, scenario reduction 23

3.1.1 Deterministic equivalent 23

3.1.2 Stability and probability metrics 24

3.1.3 Scenario reduction 26

3.2 Scenario tree formulation 28

3.3 Aggregations and their refinement 30

3.3.1 Choice of the metric c 31

3.3.2 Initial aggregation 32

3.3.3 Refining an aggregation 32

3.4 Decomposition 35

3.4.1 Scenario decomposition 35

3.4.2 Nodal decomposition 38

3.4.3 Geographical decomposition 39

3.5 Adaptive scenario reduction algorithm 41

3.6 Mixed-integer linear multistage stochastic programs 42

3.6.1 Discretization, Aggregation 43

3.6.2 Aggregation: 2. option 44

3.6.3 An example: Power Management in a hydro-thermal system under uncertainty 45

3.6.4 Nested Benders Decomposition 46

4 Numerical Results **49**

4.1 Software ADoSP 49

4.1.1 Input 49

4.2 Power management in a hydro-thermal system under uncertain load 53

5 Finish **59**

5.1 Further developments 59

5.2 Theses 61

Bibliography **63**

Chapter 1

Introduction

A stochastic program is an optimization problem where the data of the program underlies uncertainties. This uncertainty is modeled by a stochastic variable $\xi : \Omega \to \mathbb{R}^s$ over some probability space $(\Omega, \mathcal{F}, \mathbb{P})$. For a second stochastic variable $x : \Omega \to \mathbb{R}^r$ over $(\Omega, \mathcal{F}, \mathbb{P})$, a cost function $f : \mathbb{R}^r \times \mathbb{R}^s \to \mathbb{R}$ evaluates a realization $\xi(\omega)$ of the data-process and the corresponding realization $x(\omega)$ of x by $f(\xi(\omega), x(\omega))$. The aim of the optimizer is to find a decision-process x on Ω that minimizes $\mathbb{F}[f(x,\xi)]$ so that (s.t.) certain constraints on $x(\cdot)$ are fulfilled. $\mathbb{F}$ represents often the expected value of $f(x,\xi)$ over Ω. But also other functionals on the linear space of real-valued stochastic variables are thinkable. In economic settings, $\mathbb{F}$ is often a risk-measure (cf. [14, 9]). In this work, we will use the expected value, $\mathbb{F} := \mathbb{E}$, but the extension to more general functionals is conceivable. A stochastic program can formally be written as

$$\begin{array}{lll} \min\limits_{x} & \mathbb{E}[f(x,\xi)] & \\ \text{s.t.} & G(x,\xi) \leq 0 & \mathbb{P}\text{-a.s.} \\ & x \in X(\xi) & \mathbb{P}\text{-a.s.} \end{array} \tag{SP}$$

where $G : \mathbb{R}^r \times \mathbb{R}^m \to \mathbb{R}^d$ and $X : \mathbb{R}^m \rightrightarrows \mathbb{R}^r$ determines the *constraints* on x.

Multistage stochastic programs model a series of sequential stochastic optimization problems: Given a discrete-time stochastic data-process $\xi := \{\xi_t\}_{t=1}^T$ over $(\Omega, \mathcal{F}, \mathbb{P})$, $\xi_t : \Omega \to \mathbb{R}^{s_t}$, $t = 1, \ldots, T$, $\sum_{t=1}^T s_t = s$, denote by $\zeta_t := (\xi_1, \ldots, \xi_t)$ the stochastic variable that represents the events, that can be observed till time t, and by $\mathcal{F}_t := \sigma(\zeta_t)$ the sigma-algebra which is generated by ζ_t, $t = 1, \ldots, T$. We assume, that ξ_1 is deterministic, that is $\mathcal{F}_1 = \{\emptyset, \Omega\}$, and that at timestage T the complete information is known: $\mathcal{F}_T = \mathcal{F}$.

Let

$$\begin{aligned} f : \left(\bigtimes_{t=1}^{T} \mathbb{R}^{r_t}\right) \times \left(\bigtimes_{t=1}^{T} \mathbb{R}^{s_t}\right) &\to \mathbb{R}, \\ X_t : \bigtimes_{\tau=1}^{t} \mathbb{R}^{s_\tau} &\rightrightarrows \mathbb{R}^{r_t}, \qquad t = 1, \ldots, T, \\ g_t : \left(\bigtimes_{\tau=1}^{t} \mathbb{R}^{r_\tau}\right) \times \left(\bigtimes_{\tau=1}^{t} \mathbb{R}^{s_\tau}\right) &\to \mathbb{R}^{d_t}, \qquad t = 1, \ldots, T, \end{aligned}$$

be given functions. We search for a stochastic process $x := \{x_t\}_{t=1}^T$ over $(\Omega, \mathcal{F}, \mathbb{P})$, $x_t : \Omega \to \mathbb{R}^{r_t}$, $t = 1, \ldots, T$, $\sum_{t=1}^T r_t = r$, of sequential decisions that minimizes $\mathbb{E}[f(x,\xi)]$ such that the constraints $g_t(x_1, \ldots, x_t, \zeta_t) \leq 0$, $x_t \in X_t(\zeta_t)$ hold $\mathbb{P}$-almost surely for all $t = 1, \ldots, T$. Furthermore, we require from $x(\cdot)$ that the decision $x_t(\omega)$ on some timestage t depends only on the information which can be observed till time t, that is x is measurable w.r.t. $\mathcal{F}_t$. Equivalent formulations are $x_t = \mathbb{E}[x_t | \mathcal{F}_t]$, x_t depends only on ζ_t or x is non-anticipative. Thus a multistage stochastic program has the form

$$\begin{array}{llll} \min\limits_{x} & \mathbb{E}[f(x,\xi)] & & \\ & g_t(x_1,\dots,x_t,\zeta_t) \leq 0 & \mathbb{P}\text{-a.s.} & t=1,\dots,T \\ \text{s.t.} & x_t \in X_t(\zeta_t) & \mathbb{P}\text{-a.s.} & t=1,\dots,T \\ & x_t = \mathbb{E}[x_t | \mathcal{F}_t] & & t=1,\dots,T \end{array} \tag{MSP}$$

When any of the variables ξ is continuous, (MSP) is infinite dimensional. To solve it numerically it is then replaced by a finite dimensional approximation. Even if ξ has finite support, problem (MSP) is often computationally intractable and must be approximated by a problem of smaller dimension. The generation of such approximations has been a field of much research in the past decades. Often, the original probability-distribution $\mathbb{P}$ is replaced by an approximation $\hat{\mathbb{P}}$ which has finite support $\Omega_I = \{\omega_i\}_{i\in I}$. The realizations $\xi_i := \xi(\omega_i)$ of ξ, $i \in I$, represent a selected set of scenarios, and due to the assumption, that the information at timestage 1 is deterministic, they form a scenario-tree with root node $\xi_{i,1}$.

The challenge in scenario-tree generation lies now in the selection of scenarios such that the approximation (MSP[I]) is "close" to (MSP) *and* numerically solvable. Already the definition of a distance of stochastic programs with different underlying probability distributions is a difficult task.

Dupačová, Römisch, et al. ([7, 15]) reduced the evaluation of the distance between convex two-stage stochastic programs to a measure of the distance between the underlying probability distributions ($\mathbb{P}$ and $\hat{\mathbb{P}}$). Based on results from stability-theory the authors have shown that the distance between the optimal value of the original problem and it's approximation can be bounded by a product of a problem-dependent function and the distance of the probability-distributions, which is measured by some probability (semi-)metric. Therefore, the proposed method to approximate the original problem is the selection of a set of scenarios of a given size that still represent as much information of the original distribution as possible. Since the distance of $\mathbb{P}$ and $\hat{\mathbb{P}}$ is multiplied by a problem-dependent factor, it is difficult to give a quantitative measure on how good the approximation of $\mathbb{P}$ has to be so that the perturbation in the optimal value and optimal solution set is small. The algorithm we will develop in this work aims to deal with this problem. Therefore, a more detailed discussion of the scenario reduction approach from [7] will be given in section 3.1.

The scenario-tree generation algorithm from Casey and Sen for linear MSPs with random right-hand-side ([1]) constructs a sequence of approximations of the MSP which asymptotically lead to a solution of the original problem. For the solution x of such an approximated problem, the goodness of the decisions is measured in terms of a gap parameter δ_n and an infeasibility index η_n at each node n of the current scenario tree. If the gap or the infeasibility index exceeds some tolerance on some node, the tree is refined at this node. The gap parameter δ_n is defined as the gap between a lower and an upper bound on a salvage value function which is defined for each node and is the optimal value of a small linear program. The solutions of these problems can be used to construct decisions $\bar{x}_n$ for the original problem and it is shown that, provided that $\bar{x}_n$ is feasible for the original problem, $\delta_n = 0$ for all nodes n. The infeasibility index η_n is an upper bound on the probability, that a decision x_n is infeasible. The advantages of this algorithm are it's asymptotic convergence, it's measure of optimality, and the possibility to control the likelihood that a generated solution $\bar{x}_n$ for the reduced scenario tree is feasible for each node in the original tree. The computational challenge consists in the solution of the complete approximated problem in each iteration and the evaluation of the goodness of the obtained decisions in each node of the scenario tree. The latter has the advantage that it is seems to be perfectly suited for parallel computing environments.

Another scenario-tree generation algorithm which utilizes nodal information is the EVPI-based importance sampling algorithm from Dempster and Thompson ([4]). The expected value of perfect information (EVPI) for a stochastic program is calculated by first finding the optimal solution for each possible realization of the random variables and taking the expected value of them, that is problem (MSP) is solved, where nonanticipativity constraints are ignored. Then the optimal value of the true problem is subtracted. The EVPI, normalized by the optimal value of the problem, is used as a measure of the uncertainty in the model. The concept of the EVPI of a stochastic problem can be extended to the so-called local EVPI of the nodes in a scenario tree: For a node n of the scenario tree, the local EVPI is the difference between the optimal value of the problem with root node defined at n and the expected optimal value when having perfect information about the future, i.e. the average of the optimal values along each scenario that passes node n. The EVPI-sampling algorithm generates iteratively scenario trees by increasing the number of branches from nodes which have a high relative EVPI, resampling the branches for nodes with negligible relative EVPI and decreasing the number of branches for nodes which have small relative EVPIs repeatedly. This algorithm has been implemented for linear multistage stochastic programs in the software package MSLiP, which especially exploits the parallelisability of the used algorithms. The linear MSP which corresponds to a generated scenario tree is solved by a nested Benders decomposition algorithm ([5]) which is a cutting plane algorithm based on relaxing the time-coupling constraints. We will give a short introduction in this decomposition method in section 3.6.4.

Other approaches for solving MSPs are based on dual decomposition techniques. Edirisinghe's algorithm ([8]) for linear MSPs relaxes the nonanticipativity requirements of the decision-process up to a certain second-order aggregation and the computation of lower bounds. To form the aggregations, it is assumed that the support Ω of ξ can be enclosed in a compact simplex $\hat{\Omega}$ which is then partitioned into smaller subsimplices. At each iteration, the current aggregation is used to compute lower and upper bounds on the optimal value. Then a subsimplex which contributes most to the gap between the lower and upper bound is chosen and subdivided. The advantages of this approach are the computation of bounds on the value of the original problem and the possibility to exploit parallel computing facilities for this. But currently it is unclear, if this method provides any asymptotic guarantees.

Higle, Rayco and Sen presented in [16] a stochastic scenario decomposition algorithm which does not necessarily rely on a finite support for ξ or linearity of the MSP. The distribution $\mathbb{P}$ is approximated by an empirical distribution obtained by scenario sampling. The dual problem, obtained by dualizing the nonanticipativity constraints for this empirical distribution, is approximated by a stochastic cutting plane method, which generates the so-called master problem. In each iteration, solutions of the subproblems yield a new cut to update the master problem. The dual multipliers are then updated by solving the master problem and the empirical distribution is updated by sampling a new scenario. To avoid a too extensive grow of the master problem due to the amount of sampled scenarios and generated cuts, a variable aggregation scheme is proposed which aggregates columns that are subjected to similar data in the problem. We will give a more formal description of the algorithm at the end of section 3.4.1. Advantages of this approach are it's mild assumptions on the stochastic program and the ability to work with a high number of scenarios. On the other hand, no estimates on the quality of a solution after a finite number of iterations are known, i.e. it is not possible to work with an optimality tolerance, but almost sure asymptotically convergence can be shown.

In this work, we will develop an algorithm which integrates the construction of a reduced scenario-tree into the solution process of the MSP under only mild assumptions on the functions f, g_t and X_t. The origin of this algorithm is an idea of Nowak, cf. [17, chapter 9]. Starting with a rough aggregation of a given scenario-tree, this aggregation is refined as long as the estimated gap to the original problem is too large. The aggregations are formed by use of the fast-forward-selection algorithm proposed in [15]. Bounds on the optimal value of an approximated MSP are computed by solving convex relaxations. Further on, solving a convex relaxation yields an approximate solution point for the original problem. When the algorithm finished the scenario tree construction, a local search for the discretized MSP, starting from the last solution point of the convex relaxation, can be started. A column generation algorithm, based on Lagrangian decomposition, for updating the relaxation is sketched. Numerical results for the optimization of operation levels for a collection of coal fired thermal units and fill levels for pumped hydro units under uncertain load are presented.

In chapter 2 we formulate the algorithm in form of an adaptive discretization algorithm for infinite dimensional MINLPs. A lot of notation to describe the original problem, it's discretization and aggregation, the decomposition-, relaxation- and refinement-techniques is introduced there. Chapter 3 starts with a review on [7] and continues to adopt the abstract formulation from chapter 2 to multistage stochastic programs. The scenario-tree formulation of a MSP is formulated and algorithms for the aggregation and refinement of scenario trees are introduced. Further on, decomposition techniques that exploit the special structure of multistage stochastic programs are presented. As a special case, we introduce mixed-integer linear MSPs. The fourth chapter describes an implementation of the proposed algorithm for linear MSPs. Numerical results for the power management in a hydro-thermal system under uncertainty are presented. We end with some suggestions for further improvements and give some concluding remarks.

Chapter 2

An abstract adaptive discretization algorithm

This chapter aims to give an abstract formulation of an adaptive discretization algorithm for infinite-dimensional mixed integer nonlinear optimization problems. First we introduce the required notations, then we sketch an optimal refinement algorithm. Further on, we concentrate on algorithms to update a convex relaxation, which is used by the refinement algorithm. Finally, we state the adaptive discretization algorithm.

2.1 The Problem

We consider optimization problems of the form

$$\begin{array}{ll} \min\limits_{x\in E} & F(x) \\ \text{s.t.} & G(x) \leq 0 \\ & x \in X \end{array} \tag{P}$$

where E denotes some Banach space, maybe infinite-dimensional, $G : E \to \mathbb{R}$ and $X \subseteq E$.

For example, consider the stochastic program (SP), and let E be a Banach space of measurable functions $x : \Omega \to \mathbb{R}^r$, $F(x) := \mathbb{E}[f(x,\xi)]$, $X := \{x : \Omega \to \mathbb{R}^r \,|\, \mathbb{P}[x \in X(\xi)] = 1\}$, and $G(x) := \mathbb{P}[G(x,\xi) > 0]$.

A convex relaxation (C) of (P) has the form

$$\begin{array}{ll} \min\limits_{x} & \check{F}(x) \\ & \check{G}(x) \leq 0 \\ \text{s.t.} & x \in \check{X} \\ & c(x) \leq 0 \quad c \in \mathcal{C} \end{array} \tag{C}$$

where $\check{X}$ is a convex superset of X, $\check{F}$ and $\check{G}$ are convex underestimators of F and G over $\check{X}$ and $\mathcal{C} \subset \{c : E \to \mathbb{R} \text{ convex}\}$ is a set of cuts which are added to reduce the gap $|\mathrm{val}(P) - \mathrm{val}(C)|$ in the optimal values of (P) and (C) (if existing) or in the feasible sets, $\mathrm{feas}(C) \setminus \mathrm{feas}(P)$.

A discretization of (P) is based on a finite node set $\bar{N}$:

$$\begin{array}{ll} \min\limits_{x} & F_{\bar{N}}(x) \\ \text{s.t.} & G_{\bar{N}}(x) \leq 0 \\ & x \in X_{\bar{N}} \end{array} \tag{$P\left[\bar{N}\right]$}$$

where $x = (x_n)_{n\in\bar{N}} \in \mathbb{R}^{r_{\bar{N}}}$, $F_{\bar{N}} : \mathbb{R}^{r_{\bar{N}}} \to \mathbb{R}$, $G_{\bar{N}} : \mathbb{R}^{r_{\bar{N}}} \to \mathbb{R}^{d_{\bar{N}}}$, $X_{\bar{N}} \subseteq \mathbb{R}^{r_{\bar{N}}}$.

The convex relaxation of $(P[\bar{N}])$ is

$$\begin{aligned} \min_x \quad & \check{F}_{\bar{N}}(x) \\ \text{s.t.} \quad & \check{G}_{\bar{N}}(x) \le 0 \\ & x \in \check{X}_{\bar{N}} \\ & c(x) \le 0 \qquad c \in \mathcal{C}_{\bar{N}} \end{aligned} \tag{$C[\bar{N}]$}$$

where $\check{X}_{\bar{N}}$ is a convex superset of $X_{\bar{N}}$, $\check{F}_{\bar{N}}$ and $\check{G}_{\bar{N}}$ are convex underestimators of $F_{\bar{N}}$ and $G_{\bar{N}}$ over $\check{X}_{\bar{N}}$ and $\mathcal{C}_{\bar{N}} \subset \{c : \mathbb{R}^{r_{\bar{N}}} \to \mathbb{R} \text{ convex}\}$ is the set of cuts.

2.2 Aggregation

We consider now *partitions* $\mathcal{N}$ of the node set $\bar{N}$ (aggregations of nodes), i.e. $\mathcal{N}$ is a family of sets $N \subseteq \bar{N}$, s.t. $\bigcup\mathcal{N} := \bigcup_{N\in\mathcal{N}} N = \bar{N}$ and $N \cap M = \emptyset$ for all $N, M \in \mathcal{N}$, $N \neq M$. The aggregated problem is defined for a vector $x = (x_N)_{N\in\mathcal{N}}$. The aggregation can be constructed by averaging the nodes of a set N ($x_N := \sum_{n\in N} w_n x_n$, $w_n \ge 0$, $n \in N$, $\sum_{n\in N} w_n = 1$) or selecting a representative node from N ($x_N := x_n$ for a dedicated $n \in N$). Including the aggregation into $F_{\bar{N}}$, $G_{\bar{N}}$ and $X_{\bar{N}}$ gives functions $F_{\mathcal{N}} : \mathbb{R}^{r_{\mathcal{N}}} \to \mathbb{R}$ and $G_{\mathcal{N}} : \mathbb{R}^{r_{\mathcal{N}}} \to \mathbb{R}^{d_{\mathcal{N}}}$ and the set $X_{\mathcal{N}} \subset \mathbb{R}^{r_{\mathcal{N}}}$. Similar we assume to have the convex superset $\check{X}_{\mathcal{N}}$ of $X_{\mathcal{N}}$ and the convex underestimators $\check{F}_{\mathcal{N}}$ and $\check{G}_{\mathcal{N}}$ of $F_{\mathcal{N}}$ and $G_{\mathcal{N}}$ over $\check{X}_{\mathcal{N}}$.

Then the aggregated problem reads:

$$\begin{aligned} \min_x \quad & F_{\mathcal{N}}(x) \\ \text{s.t.} \quad & G_{\mathcal{N}}(x) \le 0 \\ & x \in X_{\mathcal{N}} \end{aligned} \tag{$P[\mathcal{N}]$}$$

The convex relaxation of $(P[\mathcal{N}])$ is

$$\begin{aligned} \min_x \quad & \check{F}_{\mathcal{N}}(x) \\ \text{s.t.} \quad & \check{G}_{\mathcal{N}}(x) \le 0 \\ & x \in \check{X}_{\mathcal{N}} \\ & c(x) \le 0 \qquad c \in \mathcal{C}_{\mathcal{N}} \end{aligned} \tag{$C[\mathcal{N}]$}$$

where $\mathcal{C}_{\mathcal{N}}$ is a set of valid cuts.

2.3 Optimal refinement

Notation. A *refinement* of a partition $\mathcal{N}$ of $\bar{N}$ is a partition $\mathcal{N}'$ of $\bar{N}$ s.t. for all $N' \in \mathcal{N}'$ exists an $N \in \mathcal{N}$ with $N' \subseteq N$. We will then write $\mathcal{N}' \preceq \mathcal{N}$. If additionally $|\mathcal{N}'| > |\mathcal{N}|$, we write $\mathcal{N}' \prec \mathcal{N}$.

For a refined partition $\mathcal{N}'$ of $\mathcal{N}$ and a set $N \in \mathcal{N}$ let $\mathcal{N}'|_N := \{N' \in \mathcal{N}' \,|\, N' \subseteq N\}$ be the restriction of $\mathcal{N}'$ to N. Observe, that $\bigcup\mathcal{N}'|_N = N$.

Given a large node set $\bar{N}$ and an initial (trivial) partition $\mathcal{N}_0 := \{\bar{N}\}$ of $\bar{N}$, we aim to generate a discretization $(P[\mathcal{N}])$ with $|\mathcal{N}| \ll |\bar{N}|$ that tries to keep the approximation error

$$\left|\operatorname{val}(P[\bar{N}]) - \operatorname{val}(P[\mathcal{N}])\right|$$

small. For this purpose, let $\mathcal{N}' \prec \mathcal{N}$ a refinement of $\mathcal{N}$. Then we separate the approximation error into the errors which are introduced by $\mathcal{N}$ as an approximation of $\mathcal{N}'$ and by $\mathcal{N}'$ (resp. $\bar{N}\,|_{\mathcal{N}'}$) as an approximation of $\bar{N}$. For the first error, we consider the distance between the optimal values of the convex relaxations w.r.t. $\mathcal{N}$ and $\mathcal{N}'$, for the second error, we assume firstly to have a measure for the fineness of the aggregation $\mathcal{N}'$. Since we work with convex relaxations, additional error terms due to the distance of the optimal values of a problem and it's relaxation appear. Altogether we estimate the error $\left|\operatorname{val}\left(P\left[\bar{N}\right]\right) - \operatorname{val}\left(P\left[\mathcal{N}\right]\right)\right|$ by

$$\begin{aligned}\left|\operatorname{val}\left(P\left[\bar{N}\right]\right) - \operatorname{val}\left(P\left[\mathcal{N}'\right]\right)\right| + \left|\operatorname{val}\left(P\left[\mathcal{N}'\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}'\right]\right)\right| \\ + \left|\operatorname{val}\left(C\left[\mathcal{N}'\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}\right]\right)\right| + \left|\operatorname{val}\left(C\left[\mathcal{N}\right]\right) - \operatorname{val}\left(P\left[\mathcal{N}\right]\right)\right|.\end{aligned}$$

To diminish the gap $\left|\operatorname{val}\left(C\left[\mathcal{N}\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}'\right]\right)\right|$ we check for the "importance" of a refinement $\mathcal{N}'\,|_N$ of $N \in \mathcal{N}$ with respect to this gap. The refinement of those $N \in \mathcal{N}$ which promise a high improvement are kept while the others are withdrawn. To reduce the gaps $\left|\operatorname{val}\left(P\left[\mathcal{N}'\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}'\right]\right)\right|$ and $\left|\operatorname{val}\left(P\left[\mathcal{N}\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}\right]\right)\right|$ a column generation algorithm for the improvement of the relaxation will be proposed in the next section, which allows a reduction up to the duality gap. The gap $\left|\operatorname{val}\left(P\left[\bar{N}\right]\right) - \operatorname{val}\left(P\left[\mathcal{N}'\right]\right)\right|$ is controlled by checking the distance between the aggregation $\mathcal{N}'$ and $\bar{N}$. Since this depends on the current application, we will here just assume to have a measure $\mathcal{D}\left(\mathcal{N}\right)$ on the distance between $\bar{N}\,|_{\mathcal{N}}$ and $\bar{N}$.

We continue to concentrate on the gap $\left|\operatorname{val}\left(C\left[\mathcal{N}\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}'\right]\right)\right|$: Let $\mathcal{N}$ be a partition of $\bar{N}$, $\mathcal{N}' \prec \mathcal{N}$ a refinement of $\mathcal{N}$, and let $\mathcal{M} \subseteq \mathcal{N}$ be the subsets of $\bar{N}$ which refinement we want to keep. Then denote by

$$\mathcal{N}\left(\mathcal{M}\right) := \left\{\mathcal{N}'\,|_M \mid M \in \mathcal{M}\right\} \cup \left(\mathcal{N} \setminus \mathcal{M}\right) \tag{$\mathcal{N}(\mathcal{M})$}$$

the refinement of $\mathcal{N}$ with respect to $\mathcal{M}$, i.e. the sets $M \in \mathcal{M}$ are replaced by their refinement $\mathcal{N}'\,|_M$ while the remaining sets $\mathcal{N} \setminus \mathcal{M}$ are not touched.

In each step of the method, a set $\mathcal{M} \subseteq \mathcal{N}$ with $|\mathcal{M}| \leq \bar{m}$ is computed, such that

$$\left|\operatorname{val}\left(C\left[\mathcal{N}\left(\mathcal{M}\right)\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}'\right]\right)\right|$$

is small. We compute the set $\mathcal{M}$ by use of the following strategy: Let σ_P be a given tolerance on $\left|\operatorname{val}\left(C\left[\mathcal{N}'\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}\right]\right)\right|$, relative to $\operatorname{val}\left(C\left[\mathcal{N}'\right]\right)$. After solving the problems $\left(C\left[\mathcal{N}\right]\right)$ and $\left(C\left[\mathcal{N}'\right]\right)$ either $\operatorname{val}\left(C\left[\mathcal{N}\right]\right)$ can be much smaller than $\operatorname{val}\left(C\left[\mathcal{N}'\right]\right)$, $\operatorname{val}\left(C\left[\mathcal{N}'\right]\right)$ can be much smaller than $\operatorname{val}\left(C\left[\mathcal{N}\right]\right)$ or both values are sufficient close (relative to the tolerance σ_P). We will proceed as follows:

1. If $\operatorname{val}\left(C\left[\mathcal{N}'\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}\right]\right) > \sigma_P \cdot \left|\operatorname{val}\left(C\left[\mathcal{N}\right]\right)\right|$, that is $\operatorname{val}\left(C\left[\mathcal{N}\right]\right)$ is much smaller than $\operatorname{val}\left(C\left[\mathcal{N}'\right]\right)$, then we choose $\mathcal{M}$ to be those $N \in \mathcal{N}$ which have the highest share in the difference $\operatorname{val}\left(C\left[\mathcal{N}'\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}\right]\right)$. Since we didn't set up any requirements on the structure of $F_{\mathcal{N}}$ resp. $\check{F}_{\mathcal{N}}$, it's difficult to give a formula on how to compute $\mathcal{M}$ in this case. But as an example, assume $\check{F}_{\mathcal{N}}$ can be decomposed into $\sum_{N \in \mathcal{N}} \check{F}_N$, i.e. for $x = \left(x_N\right)_{N \in \mathcal{N}}$ let

   $$\check{F}_{\mathcal{N}}\left(x\right) = \sum_{N \in \mathcal{N}} \check{F}_N\left(x_N\right).$$

   Then for $x \in \operatorname{Argmin}\left(C\left[\mathcal{N}\right]\right)$ and $x' \in \operatorname{Argmin}\left(C\left[\mathcal{N}'\right]\right)$, let

   $$\mathcal{M} := \operatorname{argmax}\left\{\sum_{M \in \mathcal{M}} \left(\sum_{M' \in \mathcal{N}'|_M} \check{F}_{M'}\left(x'\right)\right) - \check{F}_M\left(x\right) \,\middle|\, \mathcal{M} \subseteq \mathcal{N},\ |\mathcal{M}| \leq \bar{m}\right\}. \tag{2.1}$$

   Since $\operatorname{val}\left(C\left[\mathcal{N}\right]\right) - \operatorname{val}\left(C\left[\mathcal{N}'\right]\right) = \sum_{N \in \mathcal{N}} \left(\check{F}_N\left(x\right) - \sum_{N' \in \mathcal{N}'|_N} \check{F}_{N'}\left(x'\right)\right) < 0$ it holds $\mathcal{M} \neq \emptyset$.

2. If $\mathrm{val}(C[\mathcal{N}]) - \mathrm{val}(C[\mathcal{N}']) > \sigma_P \cdot |\mathrm{val}(C[\mathcal{N}'])|$, that is $\mathrm{val}(C[\mathcal{N}])$ is much higher than $\mathrm{val}(C[\mathcal{N}'])$, then we have to search for sets $N \in \mathcal{N}$ which refinement allows a high diminishment in $\mathrm{val}(C[\mathcal{N}])$. But instead of proceeding similar to case 1, which doesn't involve the constraints $G_\mathcal{N}$ and $X_\mathcal{N}$ directly, we use a less greedy method.

   Actually, we aim to reach $\mathrm{val}(C[\mathcal{N}]) - \mathrm{val}(C[\mathcal{N}']) \leq \sigma_P |\mathrm{val}(C[\mathcal{N}'])|$. This can be achieved by searching for a point $x = (x_N)_{N\in\mathcal{N}}$ which is feasible for $(C[\mathcal{N}'])$ and stays close to the optimal solution set of $(C[\mathcal{N}'])$. To find the sets $N \in \mathcal{N}$ which hamper this goal we compute a point $x' = (x'_{N'})_{N'\in\mathcal{N}'}$ which is feasible for $(C[\mathcal{N}'])$, it's objective value does not exceed $\mathrm{val}(C[\mathcal{N}'])$ by more than the fraction σ_P, and which is as "close" to the aggregation $\mathcal{N}$ as possible. We then choose the $N \in \mathcal{N}$ whose part in the distance between x' and the aggregation $\mathcal{N}$ is greatest.

   More formal, to express an artificial restriction from $\mathcal{N}'$ to $\mathcal{N}$ assume to have matrices W_N, $N \in \mathcal{N}$, such that a point $x' = (x'_{N'})_{N'\in\mathcal{N}'}$ which satisfies $W_N x'_N = 0$, $N \in \mathcal{N}$ ($x'_N := (x'_{N'})_{N'\in\mathcal{N}'|_N}$), has an unique representation as a point $x = (x_N)_{N\in\mathcal{N}}$. The matrices W_N might express some averaging- or equality-conditions on x'_N in accord with the definition of $F_\mathcal{N}$, $G_\mathcal{N}$ and $X_\mathcal{N}$. For example, the equality condition could be expressed by defining $W_N x'_N := (x'_{M'} - x'_{N'})_{N'\in\mathcal{N}'|_N}$ for a distinguished set $M' \in \mathcal{N}'|_N$, $N \in \mathcal{N}$.

   Now, we search for a point x' which is feasible for $(C[\mathcal{N}'])$, which has $\check{F}_{\mathcal{N}'}(x') \leq \mathrm{val}(C[\mathcal{N}']) + \sigma_P |\mathrm{val}(C[\mathcal{N}'])|$ and minimizes the violation of the $W_N x'_N = 0$ constraints, $N \in \mathcal{N}$. That is, we solve the convex problem

$$\begin{array}{lll} \min\limits_{x'} & \sum\limits_{N\in\mathcal{N}} \|W_N x'_N\|_\infty & \\ \text{s.t.} & \check{G}_{\mathcal{N}'}(x') \leq 0 & \\ & x' \in \check{X}_{\mathcal{N}'} & \\ & c(x') \leq 0 & c \in \mathcal{C}_{\mathcal{N}'} \\ & \check{F}_{\mathcal{N}'}(x') \leq \mathrm{val}(C[\mathcal{N}']) + \sigma_P |\mathrm{val}(C[\mathcal{N}'])| & \end{array} \tag{Proj[$\mathcal{N},\mathcal{N}'$]}$$

   In the case $\mathrm{val}(\mathrm{Proj}[\mathcal{N},\mathcal{N}']) = 0$, we are finished, since we found a point x' with $\check{F}_{\mathcal{N}'}(x') \leq \mathrm{val}(C[\mathcal{N}']) + \sigma_P |\mathrm{val}(C[\mathcal{N}'])|$ and $W_N x'_N = 0$ for all $N \in \mathcal{N}$. When we assume that the corresponding point $x = (x_N)_{N\in\mathcal{N}}$ is feasible for $(C[\mathcal{N}])$ (because $x' \in \mathrm{feas}(C[\mathcal{N}'])$ and $W_N x_N = 0$, $N \in \mathcal{N}$) and that $\check{F}_{\mathcal{N}'}(x') = \check{F}_\mathcal{N}(x)$, then this case cannot occur, because it indicates $\mathrm{val}(C[\mathcal{N}]) - \mathrm{val}(C[\mathcal{N}']) \leq \sigma_P |\mathrm{val}(C[\mathcal{N}'])|$, contradicting the assumption.

   If otherwise $\mathrm{val}(\mathrm{Proj}[\mathcal{N},\mathcal{N}']) > 0$, we add those $N \in \mathcal{N}$ with the highest violation of the $W_N x'_N = 0$ constraints to $\mathcal{M}$, i.e. for $x^* \in \mathrm{Argmin}(\mathrm{Proj}[\mathcal{N},\mathcal{N}'])$ let

$$\mathcal{M} := \mathrm{argmax}\left\{ \sum_{M\in\mathcal{M}} \|W_M x^*_M\|_\infty \,\middle|\, \mathcal{M} \subseteq \mathcal{N},\ |\mathcal{M}| \leq \bar{m},\ \|W_M x^*_M\|_\infty > 0\,\forall M \in \mathcal{M} \right\}. \tag{2.2}$$

3. Otherwise we have

$$\frac{\mathrm{val}(C[\mathcal{N}']) - \mathrm{val}(C[\mathcal{N}])}{|\mathrm{val}(C[\mathcal{N}])|} \leq \sigma_P \quad \text{and} \quad \frac{\mathrm{val}(C[\mathcal{N}]) - \mathrm{val}(C[\mathcal{N}'])}{|\mathrm{val}(C[\mathcal{N}'])|} \leq \sigma_P.$$

   Hence, the gap $|\mathrm{val}(C[\mathcal{N}]) - \mathrm{val}(C[\mathcal{N}'])|$ is small enough and we can proceed by checking $\mathcal{D}(\mathcal{N}')$, the distance of the aggregation $\mathcal{N}'$ to $\bar{N}$.

After inheriting parts of the refinement $\mathcal{N}'$ into $\mathcal{N}$ by means of formula ($\mathcal{N}(\mathcal{M})$), we need a new test-refinement $\mathcal{N}'$ in the next iteration. We compute this by partitioning those $M' \in \mathcal{N}'$ which have a $M \in \mathcal{M}$, s.t. $M' \subseteq M$. Thus we define

$$\mathcal{N}'(\mathcal{M}) := \left(\bigcup_{N\in\mathcal{N}\setminus\mathcal{M}} \mathcal{N}'|_N \right) \cup \left(\bigcup_{M\in\mathcal{M}} \bigcup_{M'\in\mathcal{N}'|_M} \mathrm{part}(M') \right). \tag{$\mathcal{N}'(\mathcal{M})$}$$

The computation of part (M') and $W_{M''}$, $M'' \in$ part (M'), is problem specific. One might define a metric $d : \bar{N} \times \bar{N} \to \mathbb{R}_+$ and compute a partition of some $M' \subseteq \bar{N}$ by a clustering algorithm. The size $\sum_{M' \in \mathcal{N}'|_M} |\text{part}(M')|$ for $M \in \mathcal{M}$ can be controlled by $\|W_M x_M\|_\infty$, i.e. the fraction of $\sum_{M' \in \mathcal{N}'|_M} |\text{part}(M')|$ on $\sum_{M \in \mathcal{M}} \sum_{M' \in \mathcal{N}'|_M} |\text{part}(M')|$ can be set (approximately) to

$$\frac{\|W_M x_M\|_\infty}{\sum_{M \in \mathcal{M}} \|W_M x_M\|_\infty}. \tag{2.3}$$

The next step is to point out, how the gap $|\text{val}(P[\mathcal{N}]) - \text{val}(C[\mathcal{N}])|$ can be controlled. This is done in the next section. Afterwards we can state the adaptive refinement algorithm in it's full form. For a summary of the above considerations, which build the core of the final discretization algorithm, we give here a first draft of the algorithm, not including the update of the convex relaxation:

Algorithm 2.1 (adaptive discretization (draft)).
Input: initial node set $\bar{N}$, parameters $\bar{m} \in \mathbb{N}$ and $\sigma_P \in [0,1]$, $\sigma_D \in [0,1]$
Compute:
compute initial rough aggregations $\mathcal{N}$ and $\mathcal{N}'$ of $\bar{N}$, $\mathcal{N}' \preceq \mathcal{N}$
$\mathcal{D}_0 := \mathcal{D}(\mathcal{N}')$
repeat
solve $(C[\mathcal{N}])$ and $(C[\mathcal{N}'])$
if $\text{val}(C[\mathcal{N}']) - \text{val}(C[\mathcal{N}]) > \sigma_P \cdot |\text{val}(C[\mathcal{N}])|$ **then**
compute $\mathcal{M}$ according to formula (2.1)
compute $\mathcal{N}(\mathcal{M})$ and $\mathcal{N}'(\mathcal{M})$
$\mathcal{N} := \mathcal{N}(\mathcal{M})$ and $\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
else if $\text{val}(C[\mathcal{N}]) - \text{val}(C[\mathcal{N}']) > \sigma_P \cdot |\text{val}(C[\mathcal{N}'])|$ **then**
solve (Proj$[\mathcal{N},\mathcal{N}']$)
compute $\mathcal{M}$ according to formula (2.2)
compute $\mathcal{N}(\mathcal{M})$ and $\mathcal{N}'(\mathcal{M})$
$\mathcal{N} := \mathcal{N}(\mathcal{M})$ and $\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
else if $\mathcal{D}(\mathcal{N}') > \sigma_D \mathcal{D}_0$ **then**
select $\mathcal{M} \subseteq \mathcal{N}$, s.t. $M' \in \mathcal{N}'|_M$, $M \in \mathcal{M}$, have a high share in $\mathcal{D}(\mathcal{N}')$
compute $\mathcal{N}'(\mathcal{M})$ (without change on $\mathcal{N}$)
$\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
end if
until $\mathcal{M} = \emptyset$
Output: the aggregated node set $\mathcal{N}$

2.4 Updating the convex relaxation

One important factor for the success of the proposed algorithm is a (primal) relaxation $(C[\mathcal{N}])$ of $(P[\mathcal{N}])$, that keeps the gap $|\text{val}(C[\mathcal{N}]) - \text{val}(P[\mathcal{N}])|$ small. Improvement of the relaxation is achieved by adding valid cuts such that the optimal value of $(C[\mathcal{N}])$ increases. We will here describe a cut-generation method which is founded on the observation that many (real-life) problems can be splitted into several smaller subproblems which are coupled through a master problem that can be solved efficiently also in high dimensions. Cuts are then used to incorporate the solution of subproblems into the master problem.

We start by introducing the concept of Lagrangian decomposition and Lagrangian cuts. Afterwards, a column-generation method for the approximate computation of the dual parameter, which is introduced by Lagrangian relaxation, is described. It has the advantage that it can be "warm-started" after refining the aggregations $\mathcal{N}$ or $\mathcal{N}'$. For an overview about Lagrangian decomposition techniques see [17], which describes also a column-generation method in the setting of branch-cut-and-price algorithms for mixed-integer nonlinear programs.

2.4.1 Lagrangian decomposition

Let problem (P) have a block-separable structure in the form that there exists a partition $\{I_1, \ldots, I_p\}$ such that

$$\begin{aligned} F(x) &= \sum_{k=1}^{p} F_k(x_{I_k}), \\ G(x) &= \sum_{k=1}^{p} G_k(x_{I_k}), \\ X &= X_1 \times \cdots \times X_p, \end{aligned}$$

i.e. problem (P) reads

$$\begin{array}{ll} \min\limits_{x} & \sum_{k=1}^{p} F_k(x_{I_k}) \\ \text{s.t.} & \sum_{k=1}^{p} G_k(x_{I_k}) \leq 0 \\ & \quad x_{I_k} \in X_k \quad k = 1, \ldots, p \end{array} \tag{P}$$

Then we can split (P) into p subproblems of smaller size by adding the coupling constraints G, provided by the dual parameter λ, to the objective function. The so-defined Lagrange function $L(x; \lambda)$ decomposes into a sum of partial Lagrange functions, one for each block:

$$\begin{aligned} L(x; \lambda) &:= \sum_{k=1}^{p} L_k(x_{I_k}; \lambda), \\ L_k(x_{I_k}; \lambda) &:= F_k(x_{I_k}) + \lambda^T G_k(x_{I_k}). \end{aligned}$$

And so the Lagrangian problem

$$\begin{array}{ll} \min\limits_{x} & L(x; \lambda) \\ \text{s.t.} & x \in X \end{array} \tag{D_λ}$$

decomposes into the partial Lagrangian problems

$$\begin{array}{ll} \min\limits_{x_{I_k}} & L_k(x_{I_k}; \lambda) \\ \text{s.t.} & x_{I_k} \in X_k \end{array} \qquad k = 1, \ldots, p, \tag{$D_{\lambda,k}$}$$

i.e. $\mathrm{val}(D_\lambda) = \sum_{k=1}^{p} \mathrm{val}(D_{\lambda,k})$.

The dual problem to (P) is now to find a dual parameter $\lambda \geq 0$ which maximizes (D_λ):

$$\max_{\lambda \geq 0} \quad \mathrm{val}(D_\lambda) \tag{D}$$

To see a relationship between problem (P) and it's dual (D), let (P) be feasible with finite optimal value and let x^* be a solution point of (P). Then $G(x^*) \leq 0$ and therefore, $L(x^*; \lambda) \leq F(x^*) = \mathrm{val}(P)$ for all $\lambda \geq 0$. Thus, $\mathrm{val}(D_\lambda) \leq L(x^*; \lambda)$ for all $\lambda \geq 0$ which implies $\mathrm{val}(D) \leq \mathrm{val}(P)$. Therefore, problems (D_λ) and (D) yield lower bounds on $\mathrm{val}(P)$. The gap $\mathrm{val}(P) - \mathrm{val}(D)$ is called *duality gap*.

To inherit the lower bounding property of (D_λ) into the convex (primal) relaxation (C), one can now use the separability of (D_λ) to generate cuts. Assume the functions F_k and G_k to be convex over $\operatorname{conv}(X_k)$ (so $\check{F} \equiv F$ and $\check{G} \equiv G$), $k = 1, \ldots, p$, so that the partial Lagrange functions $L_k(x_{I_k}; \lambda)$ become convex for all $x_{I_k} \in \operatorname{conv}(X_k)$ and all $\lambda \geq 0$, $k = 1, \ldots, p$. Then the constraint

$$L_k(x_{I_k}; \lambda) \geq \operatorname{val}(D_{\lambda,k})$$

yields a so-called ***Lagrangian Cut***, a convex function which can be added to $\mathcal{C}$. If we do so for all subproblems $k = 1, \ldots, p$ and solve the relaxation (C), obtaining a solution point $\bar{x}$, then $\operatorname{val}(C) \geq \operatorname{val}(D_\lambda)$ since

$$\begin{aligned}
\operatorname{val}(C) &= \check{F}(\bar{x}) = \sum_{k=1}^{p} F_k(\bar{x}_{I_k}) \\
&\geq \sum_{k=1}^{p} F_k(\bar{x}_{I_k}) + \lambda^T G_k(x_{I_k}) \\
&= \sum_{k=1}^{p} L_k(x_{I_k}; \lambda) \\
&\geq \sum_{k=1}^{p} \operatorname{val}(D_{\lambda,k}) \\
&= \operatorname{val}(D_\lambda).
\end{aligned}$$

The estimate $\operatorname{val}(P) \geq \operatorname{val}(D) \geq \operatorname{val}(D_\lambda)$ shows that a dual parameter $\lambda \in \operatorname{Argmin}(D)$ promises the best improvement for (C) that can be achieved by Lagrangian Cuts. Therefore, dual solution methods such as subgradient, dual cutting plane or proximal-bundle methods, which explore the concavity of the function $\lambda \mapsto \operatorname{val}(D_\lambda)$ and the convexity of $\operatorname{feas}(D)$, can be used to search for an optimal λ. An easier method is to use the dual solution point of (C) as an approximation of $\lambda \in \operatorname{Argmin}(D)$, yielding the so-called primal cutting plane methods. Unfortunately, this method does not provide convergence. For an example, see [17].

2.4.2 Column generation

A more elaborate proceeding to search for a "good" λ is the column generation method. Here, first an inner approximation of the convex hull of each X_k is build by computing extreme points of X_k. Denote by Z_k a discrete set of extreme points of X_k, then $\operatorname{conv}(Z_k) \subseteq \operatorname{conv}(X_k)$. Now we can state the restricted master problem

$$\begin{array}{ll}
\min\limits_{x} & F(x) \\
\text{s.t.} & \begin{array}{l} G(x) \leq 0 \\ \quad x_{I_k} \in \operatorname{conv}(Z_k) \quad k = 1, \ldots, p \end{array}
\end{array} \tag{RMP}$$

If (RMP) is feasible and $\times_{k=1}^{p} \operatorname{conv}(X_k) \subseteq \check{X}$, then $\operatorname{val}(C) \leq \operatorname{val}(\text{RMP})$. Observe further that for $\bar{x} \in \text{feas}\,(\text{RMP})$, i.e. $G(\bar{x}) \leq 0$ and $\bar{x}_{I_k} \in \text{conv}\,(Z_k)$, $k = 1, \ldots, p$, and for $\lambda \geq 0$ it holds that

$$\begin{aligned}
\text{val}\,(D_\lambda) &= \sum_{k=1}^{p} \text{val}\,(D_{\lambda,k}) \\
&= \sum_{k=1}^{p} \min\left\{ L_k\left(x_{I_k}, \lambda\right) | x_{I_k} \in \text{conv}\,(X_k) \right\} \\
&\leq \sum_{k=1}^{p} \min\left\{ L_k\left(x_{I_k}, \lambda\right) | x_{I_k} \in \text{conv}\,(Z_k) \right\} \\
&\leq \sum_{k=1}^{p} L_k\left(\bar{x}_{I_k}, \lambda\right) \\
&\leq \sum_{k=1}^{p} F_k\left(\bar{x}_{I_k}\right)
\end{aligned}$$

and maximizing the left-hand side over $\lambda \geq 0$ and minimizing the right-hand side over feas (RMP) yields

$$\text{val}\,(D) \leq \text{val}\,(\text{RMP})\,.$$

Increasing the size of Z_k by adding "good" extreme points will reduce val (RMP), while computation of a "good" dual parameter λ will increase val (D_λ). Column generation methods compute now extreme points for Z_k by solving the Lagrangian subproblems $(D_{\lambda,k})$, using the dual parameter obtained by solving (RMP). Solving a Lagrangian subproblem yields meanwhile Lagrangian Cuts for the primal relaxation (C). It can be shown that this method is equivalent to the dual cutting plane method, which computes a polyhedral approximation of the dual function $\lambda \mapsto \text{val}\,(D_\lambda)$ to solve the dual problem (D), and is convergent (cf. [17]).

Remark. In the case of linear functions F and G the problem (RMP) becomes an LP which can be be solved efficiently by a fast LP-solver. Solving a nonlinear nonconvex (RMP) will be much more difficult and is not always practicable. But if it is possible to "hide" the nonlinearity of the model in X_k, $k = 1, \ldots, p$, repeated solves of (RMP) are practicable, especially if solution information can be reused in successive solves with modifying sets of extreme points. In the case, that integrality constraints are included in X_k, it is otherwise imaginable to inherit these constraints into the (RMP) to improve the relaxation quality. In case of linear functions F and G, the master problem becomes a MIP. For such problems, efficient solution algorithms have been developed in the past years.

Initialization of Z_k

The initialization of (RMP) by computing initial extreme points Z_k, $k = 1, \ldots, p$, seems to be a difficult task. A first approach is to obtain a dual solution point λ of (C) and to solve the corresponding Lagrangian subproblems $(D_{\lambda,k})$, $k = 1, \ldots, p$. But this does not ensure the feasibility of (RMP) since the coupling constraints $G(\cdot) \leq 0$ might not be satisfied. So additional extreme points are needed to make problem (RMP) feasible. Nowak introduced a geometric motivated algorithm in [17] for the initialization of (RMP), but it does not guarantee to find enough extreme points in any case.

Since the sense of (RMP) is to compute an approximation of the dual parameter λ, we might drop the requirement that (RMP) is feasible and add the violation of the coupling constraints as penalty term to the objective function:

$$\begin{array}{lll} \min\limits_{x} & F(x) + \delta e^T v_G & \\ & G(x) \leq v_G & \\ \text{s.t.} & x_{I_k} \in \operatorname{conv}(Z_k) & k = 1, \ldots, p \\ & v_G \geq 0 & \end{array} \tag{RMP'}$$

A dual solution point λ of (RMP') can still be considered as an approximated solution point of (D) and we can proceed to formulate Lagrangian subproblems $(D_{\lambda,k})$ to compute new extreme points and Lagrangian cuts. Additional extreme points can reduce the infeasibility of (RMP), thus a new dual solution point of (RMP') yields a better approximation of a solution point of (D).

Another possible simplification is to diminish the requirement $Z_k \subseteq X_k$ to $Z_k \subseteq \operatorname{conv}(X_k)$. This will have no further influence on the quality of the inner approximation since $\operatorname{val}(D_{\lambda,k}) = \min\{L_k(x_{I_k}; \lambda) \,|\, x_{I_k} \in \operatorname{conv}(X_k)\}$, as we already used above.

Explored subproblems

When updating the convex relaxation by Lagrangian cuts, it is sufficient to solve only those Lagrangian subproblems $(D_{\lambda,k})$, $k = 1, \ldots, p$, which are not already "explored", that is the gap between the inner and outer approximation of X_k is still too high: The convex relaxation (outer approximation) of $(D_{\lambda,k})$ is

$$\begin{array}{lll} \min\limits_{x_{I_k}} & L_k(x_{I_k}; \lambda) & \\ \text{s.t.} & x_{I_k} \in \check{X}_k & \\ & c(x_{I_k}) \leq 0 & c \in \mathcal{C}_k \end{array} \tag{$C_{\lambda,k}$}$$

where $\check{X}_k$ is a convex superset of X_k and $\mathcal{C}_k$ denotes the cuts from $\mathcal{C}$ which apply to block k only. Further consider the inner approximation of $\operatorname{conv}(D_{\lambda,k}) := \min_{x_{I_k}} \{L_k(x_{I_k}; \lambda) \,|\, x_{I_k} \in \operatorname{conv}(X_k)\}$, obtained by replacing $\operatorname{conv}(X_k)$ by $\operatorname{conv}(Z_k)$, i.e.

$$\begin{array}{ll} \min\limits_{x_{I_k}} & L_k(x_{I_k}; \lambda) \\ \text{s.t.} & x_{I_k} \in \operatorname{conv}(Z_k) \end{array} \tag{$\mathrm{RMP}_{\lambda,k}$}$$

So the optimal value of $(D_{\lambda,k})$ is bounded from below and above by the inequalities

$$\operatorname{val}(C_{\lambda,k}) \leq \operatorname{val}(D_{\lambda,k}) \leq \operatorname{val}(\mathrm{RMP}_{\lambda,k}).$$

If the gap $\operatorname{val}(\mathrm{RMP}_{\lambda,k}) - \operatorname{val}(C_{\lambda,k})$ is small, the k-th Lagrangian subproblem does not need to be solved since an additional Lagrangian cut and an additional extreme point cannot improve the relaxations further.

2.4.3 Aggregated case

After introducing Lagrangian Decomposition and column generation for the general case above, we continue with the notation for the aggregation of the discretized case and have a look on how to warm-start column generation after a refinement of the aggregation.

Assume that $(P[\mathcal{N}])$ has the following block-separable structure:

$$\begin{aligned} F_{\mathcal{N}}(x) &= \sum_{k=1}^{p} F_{k,\mathcal{N}}(x_{I_k}), \\ G_{\mathcal{N}}(x) &= \sum_{k=1}^{p} G_{k,\mathcal{N}}(x_{I_k}), \\ X_{\mathcal{N}} &= X_{1,\mathcal{N}} \times \cdots \times X_{p,\mathcal{N}}. \end{aligned}$$

Then we obtain the aggregated problem

$$\begin{array}{ll} \min\limits_{x} & \sum\limits_{k=1}^{p} F_{k,\mathcal{N}}\left(x_{I_k}\right) \\ \text{s.t.} & \sum\limits_{k=1}^{p} G_{k,\mathcal{N}}\left(x_{I_k}\right) \leq 0 \\ & \qquad x_{I_k} \in X_{k,\mathcal{N}} \quad k = 1, \ldots, p \end{array} \tag{$P\left[\mathcal{N}\right]$}$$

where $x_{I_k} = \left(x_{I_k,N}\right)_{N \in \mathcal{N}}$.

The problem $(P\left[\mathcal{N}\right])$ splits now into p subproblems by dualizing the coupling constraints $G_{\mathcal{N}}$: Introduce the partial Lagrange function

$$L_{k,\mathcal{N}}\left(x_{I_k}; \lambda\right) := F_{k,\mathcal{N}}\left(x_{I_k}\right) + \lambda^T G_{k,\mathcal{N}}\left(x_{I_k}\right)$$

to obtain the partial Lagrangian problems

$$\begin{array}{ll} \min\limits_{x_{I_k}} & L_{k,\mathcal{N}}\left(x_{I_k}; \lambda\right) \\ \text{s.t.} & x_{I_k} \in X_{k,\mathcal{N}} \end{array} \tag{$D_{\lambda,k}\left[\mathcal{N}\right]$}$$

Joining the partial Lagrangian problems yields the aggregated dual problem

$$\max_{\lambda \geq 0} \quad \sum_{k=1}^{p} \mathrm{val}\left(D_{\lambda,k}\left[\mathcal{N}\right]\right) \tag{$D\left[\mathcal{N}\right]$}$$

Lagrangian cuts

Solving a partial Lagrangian problem for a dual parameter $\lambda \geq 0$ gives now the Lagrangian cut

$$L_{k,\mathcal{N}}\left(x_{I_k}; \lambda\right) \geq \mathrm{val}\left(D_{\lambda,k}\left[\mathcal{N}\right]\right)$$

which can be used to update the convex relaxation $(C\left[\mathcal{N}\right])$ of $(P\left[\mathcal{N}\right])$, that is the function $\mathrm{val}\left(D_{\lambda,k}\left[\mathcal{N}\right]\right) - L_{k,\mathcal{N}}\left(\cdot_{I_k}; \lambda\right)$ is added to $\mathcal{C}_{\mathcal{N}}$.

Solving all Lagrangian subproblems and adding the obtained cuts to $\mathcal{C}_{\mathcal{N}}$ yields the estimate $\mathrm{val}\left(C\left[\mathcal{N}\right]\right) \geq \sum_{k=1}^{p} \mathrm{val}\left(D_{\lambda,k}\left[\mathcal{N}\right]\right)$ so a "best" update of $(C\left[\mathcal{N}\right])$ is achieved by using a $\lambda \geq 0$ that solves $(D\left[\mathcal{N}\right])$. Since solving problem $(D\left[\mathcal{N}\right])$ can be expensive, we use the column generation method to find an approximate solution point of $(D\left[\mathcal{N}\right])$.

Column generation

Denote by $Z_{k,\mathcal{N}}$ a discrete set of extreme points of $X_{k,\mathcal{N}}$, so that $\mathrm{conv}\left(Z_{k,\mathcal{N}}\right) \subseteq \mathrm{conv}\left(X_{k,\mathcal{N}}\right)$. The aggregated restricted master problem is stated by

$$\begin{array}{ll} \min\limits_{x} & F_{\mathcal{N}}\left(x\right) \\ \text{s.t.} & G_{\mathcal{N}}\left(x\right) \leq 0 \\ & \quad x_{I_k} \in \mathrm{conv}\left(Z_{k,\mathcal{N}}\right) \quad k = 1, \ldots, p \end{array} \tag{RMP$[\mathcal{N}]$}$$

For (RMP$[\mathcal{N}]$) we have the estimate $\mathrm{val}\left(D\left[\mathcal{N}\right]\right) \leq \mathrm{val}\left(\mathrm{RMP}[\mathcal{N}]\right)$ and this gap is reduced by adding extreme points to $Z_{k,\mathcal{N}}$. Computation of extreme points by solving Lagrangian subproblems yield

Lagrangian cuts which are added to $\mathcal{C}_{\mathcal{N}}$. To reduce the number of subproblems to be solved in each iteration consider the outer approximation

$$\begin{array}{lll} \min\limits_{x_{I_k}} & L_{k,\mathcal{N}}\left(x_{I_k};\lambda\right) & \\ \text{s.t.} & x_{I_k} \in \check{X}_{k,\mathcal{N}} & \\ & c\left(x_{I_k}\right) \le 0 & c \in \mathcal{C}_{k,\mathcal{N}} \end{array} \tag{$C_{\lambda,k}\left[\mathcal{N}\right]$}$$

and inner approximation

$$\begin{array}{ll} \min\limits_{x_{I_k}} & L_{k,\mathcal{N}}\left(x_{I_k};\lambda\right) \\ \text{s.t.} & x_{I_k} \in \operatorname{conv}\left(Z_{k,\mathcal{N}}\right) \end{array} \tag{$\mathrm{RMP}_{\lambda,k}\left[\mathcal{N}\right]$}$$

of $\operatorname{conv}(D_{\lambda,k}\left[\mathcal{N}\right])$ with $\check{X}_{k,\mathcal{N}}$ a convex superset of $X_{k,\mathcal{N}}$ and $\mathcal{C}_{k,\mathcal{N}}$ the cuts from $\mathcal{C}_{\mathcal{N}}$ which apply to block k only, $k = 1, \ldots, p$.

Refinement operations

In the optimal refinement algorithm stated in section 2.3 an aggregation $\mathcal{N}$ is iteratively refined. After each refinement operation, the convex relaxations have to be evaluated. Since the problem $(P\left[\mathcal{N}'\right])$ for the refinement $\mathcal{N}'$ of $\mathcal{N}$ might be similar to $(P\left[\mathcal{N}\right])$, it is desirable to inherit the cuts $\mathcal{C}_{\mathcal{N}}$ from $(C\left[\mathcal{N}\right])$ to $\mathcal{C}_{\mathcal{N}'}$. Furthermore, having the column generation method and the difficulties with it's initialization in mind, it would be preferable to update the extreme points $Z_{k,\mathcal{N}}$ to extreme points $Z_{k,\mathcal{N}'}$ to form initial inner approximations of $\operatorname{conv}\left(X_{k,\mathcal{N}'}\right)$, $k = 1, \ldots, p$.

Therefore, we assume that a point $x_{I_k} = \left(x_{I_k,N}\right)_{N\in\mathcal{N}} \in X_{k,\mathcal{N}}$ can be extended to a point $x'_{I_k} = \left(x'_{I_k,N'}\right)_{N'\in\mathcal{N}'} \in X_{k,\mathcal{N}'}$, $k = 1, \ldots, p$. This operation will be expressed by operators $E_k : X_{k,\mathcal{N}} \to X_{k,\mathcal{N}'}$. Thus we get

$$E_k\left(X_{k,\mathcal{N}}\right) \subseteq X_{k,\mathcal{N}'}.$$

Consider now a Lagrangian cut $L_{k,\mathcal{N}}\left(\cdot_{I_k};\lambda\right) \ge \operatorname{val}\left(D_{\lambda,k}\left[\mathcal{N}\right]\right)$ from $\mathcal{C}_{\mathcal{N}}$ with $x^*_{I_k} \in \operatorname{Argmin}\left(D_{\lambda,k}\left[\mathcal{N}\right]\right)$. Since $E_k\left(x^*_{I_k}\right)$ does not need to be optimal for $(D_{\lambda,k}\left[\mathcal{N}'\right])$ we cannot just add $L_{k,\mathcal{N}'}\left(\cdot_{I_k};\lambda\right) \ge L_{k,\mathcal{N}'}\left(E_k\left(x^*_{I_k}\right);\lambda\right)$ to $\mathcal{C}_{\mathcal{N}'}$. But assuming that the problems $(D_{\lambda,k}\left[\mathcal{N}\right])$ and $(D_{\lambda,k}\left[\mathcal{N}'\right])$ are similar, solving $(D_{\lambda,k}\left[\mathcal{N}'\right])$ with a warm start from $E_k\left(x^*_{I_k}\right)$ might be maintainable. Then the cut $L_{k,\mathcal{N}'}\left(\cdot_{I_k};\lambda\right) \ge \operatorname{val}\left(D_{\lambda,k}\left[\mathcal{N}'\right]\right)$ can be added to $\mathcal{C}_{\mathcal{N}'}$.

For the extreme points $Z_{k,\mathcal{N}}$ observe that we did not require them to be optimal points of $(D_{\lambda,k}\left[\mathcal{N}\right])$ for some dual parameter $\lambda \ge 0$. So we might just set $Z_{k,\mathcal{N}'} := E_k\left(Z_{k,\mathcal{N}'}\right)$, so that $Z_{k,\mathcal{N}'} \subset X_{k,\mathcal{N}'}$ holds. But the points $Z_{k,\mathcal{N}'}$ can be improved similar to the update of Lagrangian cuts by solving Lagrangian subproblems, warm-started from the points in $Z_{k,\mathcal{N}'}$.

2.5 Adaptive discretization algorithm

We are now ready to give an abstract formulation of the adaptive discretization algorithm:

Algorithm 2.2 (adaptive discretization).
Input: initial node set $\bar{N}$, parameters $\bar{m} \in \mathbb{N}$ and $\sigma_P \in [0,1]$, $\sigma_D \in [0,1]$
Compute:
  compute initial rough aggregations $\mathcal{N}$ and $\mathcal{N}'$ of $\bar{N}$, $\mathcal{N}' \preceq \mathcal{N}$
  $\mathcal{D}_0 := \mathcal{D}\left(\mathcal{N}'\right)$
  initialize $(\mathrm{RMP}[\mathcal{N}])$ by computing an initial set of extreme points $Z_{k,\mathcal{N}} \subseteq X_{k,\mathcal{N}}$
  repeat
    update the convex relaxation $(C\left[\mathcal{N}\right])$ by cut generation, see algorithm 2.3

compute $\mathcal{C}_{\mathcal{N}'}$ from $\mathcal{C}_{\mathcal{N}}$
solve $(C[\mathcal{N}])$ and $(C[\mathcal{N}'])$
if $\mathrm{val}(C[\mathcal{N}']) - \mathrm{val}(C[\mathcal{N}]) > \sigma_P \cdot |\mathrm{val}(C[\mathcal{N}])|$ **then**
compute $\mathcal{M}$ according to formula (2.1)
compute $\mathcal{N}(\mathcal{M})$ and $\mathcal{N}'(\mathcal{M})$
$Z_{k,\mathcal{N}(\mathcal{M})} := E_k(Z_{k,\mathcal{N}})$, $k = 1, \ldots, p$.
$\mathcal{N} := \mathcal{N}(\mathcal{M})$ and $\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
else if $\mathrm{val}(C[\mathcal{N}]) - \mathrm{val}(C[\mathcal{N}']) > \sigma_P \cdot |\mathrm{val}(C[\mathcal{N}'])|$ **then**
solve $(\mathrm{Proj}[\mathcal{N}, \mathcal{N}'])$
compute $\mathcal{M}$ according to formula (2.2)
compute $\mathcal{N}(\mathcal{M})$ and $\mathcal{N}'(\mathcal{M})$
$Z_{k,\mathcal{N}(\mathcal{M})} := E_k(Z_{k,\mathcal{N}})$, $k = 1, \ldots, p$.
$\mathcal{N} := \mathcal{N}(\mathcal{M})$ and $\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
else if $\mathcal{D}(\mathcal{N}') > \sigma_D \mathcal{D}_0$ **then**
select $\mathcal{M} \subseteq \mathcal{N}$, s.t. $M' \in \mathcal{N}'|_M$, $M \in \mathcal{M}$, have a high share in $\mathcal{D}(\mathcal{N}')$
compute $\mathcal{N}'(\mathcal{M})$ (without change on $\mathcal{N}$)
$\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
end if
until $\mathcal{M} = \emptyset$
Output: the aggregated node set $\mathcal{N}$

The column generation method for the generation of cuts $\mathcal{C}_{\mathcal{N}}$ is stated in the following algorithm. A subproblem $(D_{\lambda,k}[\mathcal{N}])$ is considered as "explored" if the relative distance between the optimal values of the inner approximation $(\mathrm{RMP}_{\lambda,k}[\mathcal{N}])$ and the outer approximation $(C_{\lambda,k}[\mathcal{N}])$ of $\mathrm{conv}(D_{\lambda,k}[\mathcal{N}])$ does not exceed the parameter σ_E.

Algorithm 2.3 (Column Generation).
Input: extreme points $Z_{k,\mathcal{N}}$, $k = 1, \ldots, p$, and a tolerance $\sigma_E > 0$
Compute:
solve $(\mathrm{RMP}[\mathcal{N}])$ to obtain a dual solution point λ
repeat
for all $k = 1, \ldots, p$ **do**
solve $(C_{\lambda,k}[\mathcal{N}])$ and obtain a primal solution point $\hat{x}_{I_k}$
if $\frac{\mathrm{val}(\mathrm{RMP}_{\lambda,k}[\mathcal{N}]) - \mathrm{val}(C_{\lambda,k}[\mathcal{N}])}{|\mathrm{val}(C_{\lambda,k}[\mathcal{N}])|} > \sigma_E$ **then**
compute a solution point z_{I_k} of $(D_{\lambda,k}[\mathcal{N}])$, using $\hat{x}_{I_k}$ as startpoint
add z_{I_k} to $Z_{k,\mathcal{N}}$ and the cut $-L_{k,\mathcal{N}}(\cdot_{I_k}; \lambda) + \mathrm{val}(D_{\lambda,k}[\mathcal{N}])$ to $\mathcal{C}_{\mathcal{N}}$
end if
end for
solve $(\mathrm{RMP}[\mathcal{N}])$ to obtain new primal and dual solution points x^* and λ
prune extreme points from $Z_{k,\mathcal{N}}$ whose corresponding convex-combination-parameter for x^* is 0
until $\mathrm{val}(\mathrm{RMP}[\mathcal{N}])$ didn't change and $Z_{k,\mathcal{N}}$ did not vary for some $k \in \{1, \ldots, p\}$
Output: updated sets of extreme points $Z_{k,\mathcal{N}}$, $k = 1, \ldots, p$, and cuts $\mathcal{C}_{\mathcal{N}}$

Remark. Of course, many variations of algorithm 2.3 are imaginable. Varying between a feasible and infeasible master problem offers many possibilities on how to deal with these infeasibilities. Also the selection of subproblems to solve is a tricky task. For example, instead of solving all Lagrangian subproblem whose relative distance between $\mathrm{val}(C_{\lambda,k}[\mathcal{N}])$ and $\mathrm{val}(\mathrm{RMP}_{\lambda,k}[\mathcal{N}])$ exceeds σ_E, one may think about picking these subproblem where the relative distance is biggest, solve them, and then resolve $(\mathrm{RMP}[\mathcal{N}])$ to update the dual parameter.

Chapter 3

An adaptive discretization algorithm for multistage stochastic programs

This chapter aims to state the abstract algorithm from the last chapter for the discretization of multistage stochastic programs. First we introduce some theoretic background on stability of convex two-stage stochastic programs and derive scenario-reduction algorithms which are based on this results ([7, 15]). Further on we formulate the scenario-tree-form of a MSP and describe our approach to aggregations and refinements of the scenario tree. The next topic will be an overview about decomposition techniques for multistage stochastic programs. Then we are ready to state the adaptive scenario reduction algorithm in it's final form. This chapter ends by a consideration of the special case of mixed integer linear MSPs.

3.1 Deterministic equivalent, stability, scenario reduction

3.1.1 Deterministic equivalent

Recall the multistage stochastic program (MSP), page 8. We give here only a short description of the dynamic programming method, which will lead us to the deterministic equivalent of (MSP), for details see [19]. We start by defining the underlying recursion formulas:

For the last timestage, $t = T$, define the function $f_T : \Omega \times \mathbb{R}^r \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ by

$$f_T(\omega, x_1, \ldots, x_T) := \begin{cases} f(x_1, \ldots, x_T, \xi(\omega)) & x_t \in X_t(\zeta_t),\, g_t(x_1, \ldots, x_t, \zeta_t(\omega)) \leq 0,\ t = 1, \ldots, T \\ +\infty & \text{otherwise} \end{cases}$$

for all $\omega \in \Omega$, $x_t \in \mathbb{R}^{r_t}$, $t = 1, \ldots, T$. Furthermore let f, g_t and X_t, $t = 1, \ldots, T$, be such that there exists functions $\bar{f}_T : \Omega \times \mathbb{R}^r \to \bar{\mathbb{R}}$ and $q \in L^1(\Omega, \mathcal{F}, \mathbb{P})$ which satisfy the following properties:

- $\bar{f}_T$ is a normal integrand: The set-valued function $\omega \mapsto \mathrm{epi}\bar{f}_T(\omega, \cdot) := \{(x, \alpha) \in \mathbb{R}^r \times \mathbb{R} \,|\, \bar{f}_T(\omega, x) \leq \alpha\}$ is closed-valued and measurable ($\{\omega \in \Omega \,|\, \exists (x, \alpha) \in O : \bar{f}_T(x, \omega) \leq \alpha\} \in \mathcal{F}$ for all open sets $O \subseteq \mathbb{R}^r \times \mathbb{R}$).
- $\{x \in \mathbb{R}^r \,|\, \bar{f}_T(\omega, x) \leq \alpha\}$ is compact for all $\omega \in \Omega$ and all $\alpha \in \mathbb{R}$
- $\mathbb{P}\left[f_T(\omega, x) = \bar{f}_T(\omega, x)\ \forall x \in \mathbb{R}^r\right] = 1$
- $\forall x \in \mathbb{R}^r$: $f_T(\omega, x) \geq q(\omega)$ $\mathbb{P}$-a.s.

Now we can recursively define the following functions ("*Bellmann-recursion*"):

$$\varphi_t(\omega, x_1(\omega), \ldots, x_t(\omega)) := \mathbb{E}\left[f_t(\cdot, x_1(\cdot), \ldots, x_t(\cdot)) \,|\mathcal{F}_t\right](\omega) \qquad \mathbb{P}\text{-a.s.}, \quad t = T, \ldots, 1 \tag{3.1}$$

$$f_{t-1}(\omega, x_1, \ldots, x_{t-1}) := \inf_{y \in \mathbb{R}^{r_t}} \varphi_t(\omega, x_1, \ldots, x_{t-1}, y) \qquad \mathbb{P}\text{-a.s.}, \quad t = T, \ldots, 1 \tag{3.2}$$

The following theorem can be shown ([10, theorem 1 and 2]):

Theorem 3.1. *Let the above conditions on f_T hold and let there be a function $\tilde{x} : \Omega \to \mathbb{R}^r$ with $\tilde{x}_t$ $\mathcal{F}_t$-measurable and $\mathbb{E}\left[f_T(\omega, \tilde{x}_1(\omega), \ldots, \tilde{x}_T(\omega))\right] < \infty$. Then the mapping $\bar{x} : \Omega \to \mathbb{R}^r$ is optimal for (MSP) if and only if the Bellmann-equations*

$$\varphi_t(\omega, \bar{x}_1(\omega), \ldots, \bar{x}_t(\omega)) = f_{t-1}(\omega, \bar{x}_1(\omega), \ldots, \bar{x}_{t-1}(\omega)) \qquad \mathbb{P}\text{-a.s.},\ t = 1, \ldots, T \tag{3.3}$$

are satisfied. Furthermore there exists an optimal solution of (MSP).

For $t = 1$ the Bellmann-recursion formulas (3.1), (3.2) yield the deterministic optimization problem

$$f_0 := \inf_{x_1 \in \mathbb{R}^{r_1}} \mathbb{E}\left[f_1(\omega, x_1)\right] \tag{DA}$$

for the determination of an optimal first-stage decision. It is called the *deterministic equivalent* of (MSP). Theorem 3.1 gives requirements under which (DA) is solvable and a solution $\bar{x}_1$ of (DA) can be extended to a solution $\bar{x}(\cdot)$ of (MSP).

3.1.2 Stability and probability metrics

When solving multistage stochastic programs one is concerned with the underlying probability measure. The probability measure $\mathbb{P}$ does not need to be known precisely or does not have a finite support. Even in the case of a finite number of scenarios, problem (MSP) is often large-scale. On the other hand, solving the deterministic equivalent (DA) means to handle an objective function which is given implicitly only.

To deal with such problems, many approximation methods has been developed which replace $\mathbb{P}$ by a discrete probability measure $\mathbb{Q}$ with smaller support having the property that the corresponding optimal value is close to $\mathrm{val}(\mathrm{MSP}) = \mathrm{val}(\mathrm{DA})$. In this context, stability properties with respect to approximations of $\mathbb{P}$ become important. Unfortunately, such properties are currently unknown for the multistage case. We will here present some results from [7] about stability properties of convex two-stage stochastic programs. These considerations led to scenario-reduction algorithms we will present afterwards.

A two-stage stochastic program can be formulated similar to (DA) as

$$\inf_{x \in X} \mathbb{E}_{\mathbb{P}}\left[f(\cdot, x)\right] = \inf_{x \in X} \int_{\Omega} f(\omega, x)\, \mathbb{P}[d\omega] \tag{3.4}$$

where the convex set $X \subseteq \mathbb{R}^r$ describes the set of feasible first stage decisions and $f : \Omega \times \mathbb{R}^r \to \bar{\mathbb{R}}$ has a similar form as the function f_{T-1} in (3.2), i.e.

$$f(\omega, x) = \inf\left\{\tilde{f}(\omega, x, y) \,|y \in Y(\xi(\omega)), g(x, y, \xi(\omega)) \leq 0\right\}$$

for a function $\tilde{f} : \Omega \times \mathbb{R}^r \times \mathbb{R}^{r_2} \to \bar{\mathbb{R}}$. Further assume the function f to be measurable with respect to ω and lower semicontinuous and convex with respect to x. Let $\mathbb{E}_{\mathbb{P}}[\cdot]$ be the expected value with respect to the probability measure $\mathbb{P}$ on $(\Omega, \mathcal{F})$. Denote the set of probability measures on $(\Omega, \mathcal{F})$ by $\mathcal{P}(\Omega)$.

For $\mathbb{Q} \in \mathcal{P}(\Omega)$ let

$$v(\mathbb{Q}) := \inf_{x \in \mathbb{R}^r} \mathbb{E}_{\mathbb{Q}}[f(\cdot, x)]$$

be the optimal value of (3.4) w.r.t. the probability measure $\mathbb{Q}$. Denote the set of ε-approximate solutions by

$$S_\varepsilon(\mathbb{Q}) := \{x \in \mathbb{R}^r \,|\mathbb{E}_{\mathbb{Q}}[f(\cdot, x)] \leq v(\mathbb{Q}) + \varepsilon\}$$

for each $\varepsilon \geq 0$ and let $S(\mathbb{Q}) := S_0(\mathbb{Q})$ denote the set of solution points.

Consider now the set of probability measures

$$\mathcal{P}_f := \left\{ \mathbb{Q} \in \mathcal{P}(\Omega) \middle| -\infty < \int_\Omega \inf_{\|x\| \leq \rho} f(\omega, x)\, \mathbb{Q}[d\omega] \leq \sup_{\|x\| \leq \rho} \int_\Omega f(\omega, x)\, \mathbb{Q}[d\omega] < \infty \quad \forall \rho > 0 \right\}$$

and probability distances

$$d_{f,\rho}(\mathbb{P}, \mathbb{Q}) := \sup_{\|x\| \leq \rho} |\mathbb{E}_{\mathbb{P}}[f(\cdot, x)] - \mathbb{E}_{\mathbb{Q}}[f(\cdot, x)]|$$

for each $\rho > 0$ and $\mathbb{P}, \mathbb{Q} \in \mathcal{P}_f$. To estimate the variation in $v(\cdot)$ and $S_\varepsilon(\cdot)$ by perturbation in the probability-measure, we define the conditioning function ψ of (3.4) by

$$\psi(\tau) := \min_{x:\, d(x, S(\mathbb{P})) \geq \tau} \int_\Omega f(\omega, x)\, \mathbb{Q}[d\omega] - v(\mathbb{P})$$

for $\tau \geq 0$, with $d(x, C) := \inf_{y \in C} \|x - y\|$ the distance of $x \in \mathbb{R}^n$ to $C \subseteq \mathbb{R}^n$. Further let $\Psi(\eta) := \eta + \psi^{-1}(2\eta)$ for $\eta \geq 0$ and $\mathbb{B} := \{x \in \mathbb{R}^r \,|\|x\| \leq 1\}$ the closed unit ball in $\mathbb{R}^r$. Finally denote by

$$\mathbf{d}_\infty(C, D) := \sup_{x \in \mathbb{R}^n} |d(x, C) - d(x, D)|$$

the Pompeiu-Hausdorff distance of nonempty closed sets $C, D \subseteq \mathbb{R}^n$. Then the following stability result for optimal values and ε-approximate solution sets can be shown:

Theorem 3.2. *Let $\mathbb{P} \in \mathcal{P}_f$ and $S(\mathbb{P})$ be nonempty and bounded. Then there exist constants $\rho > 0$ and $\bar{\varepsilon} > 0$ such that*

$$|v(\mathbb{P}) - v(\mathbb{Q})| \leq d_{f,\rho}(\mathbb{P}, \mathbb{Q}) \quad \text{and} \quad \emptyset \neq S(\mathbb{Q}) \subseteq S(\mathbb{P}) + \Psi(d_{f,\rho}(\mathbb{P}, \mathbb{Q}))\,\mathbb{B}$$

whenever $\mathbb{Q} \in \mathcal{P}_f$ with $d_{f,\rho}(\mathbb{P}, \mathbb{Q}) < \bar{\varepsilon}$, and that it holds for any $\varepsilon \in (0, \bar{\varepsilon})$

$$\mathbf{d}(S_\varepsilon(\mathbb{P}), S_\varepsilon(\mathbb{Q})) \leq \frac{2\rho}{\varepsilon} d_{f,\rho+\varepsilon}(\mathbb{P}, \mathbb{Q})$$

whenever $\mathbb{Q} \in \mathcal{P}_f$ with $d_{f,\rho}(\mathbb{P}, \mathbb{Q}) < \varepsilon$.

While the probability distance $d_{f,\rho}$ is difficult to compute explicitly, it can be bounded above by a probability-metric which involves f only indirectly. For the probability-measure $\mathbb{Q}$ computed by a scenario-reduction-algorithm which aims to minimize the distance of $\mathbb{P}$ and $\mathbb{Q}$ w.r.t. this metric, the variation in the optimal value is controlled by theorem 3.2.

The integrand $f(\cdot, x)$ of (3.4) is often nondifferentiable but locally Lipschitz continuous on Ω. In many cases there exists a measurable continuous metric $c : \Omega \times \Omega \to \mathbb{R}^+$ and a nondecreasing function $g : \mathbb{R}^+ \to \mathbb{R}^+ \setminus \{0\}$ such that for each $x \in \mathbb{R}^r$ and $\omega, \tilde{\omega} \in \Omega$

$$|f(\omega, x) - f(\tilde{\omega}, x)| \leq g(\|x\|)\, c(\omega, \tilde{\omega}), \tag{3.5}$$

that is $f(\cdot, x) \in F_{g(\|x\|)c}$ where

$$F_c := \{f : \Omega \to \mathbb{R} \,| f(\omega) - f(\tilde{\omega}) \leq c(\omega, \tilde{\omega}) \quad \forall \omega, \tilde{\omega} \in \Omega\}.$$

For the metric c there exists (due to triangular-inequality) a measurable function $\lambda : \Omega \to \mathbb{R}^+$ that is bounded on bounded sets and has the property $c(\omega, \tilde{\omega}) \leq \lambda(\omega) + \lambda(\tilde{\omega})$ for all $\omega, \tilde{\omega} \in \Omega$.

Now we consider the probability (semi-)metric[1]

$$\zeta_c(\mathbb{P}, \mathbb{Q}) := \sup_{f \in F_c} \left| \int_\Omega f(\omega) \mathbb{P}[d\omega] - \int_\Omega f(\omega) \mathbb{Q}[d\omega] \right|$$

and the so-called *Kantorovich functional*

$$\hat{\mu}_c(\mathbb{P}, \mathbb{Q}) := \inf \left\{ \int_{\Omega \times \Omega} c(\omega, \tilde{\omega}) \eta(d(\omega, \tilde{\omega})) \middle| \eta \in \mathcal{P}(\Omega \times \Omega), \forall B \in \mathcal{F} : \begin{array}{c} \eta(B \times \Omega) = \mathbb{P}(B) \\ \eta(\Omega \times B) = \mathbb{Q}(B) \end{array} \right\}$$

for $\mathbb{P}, \mathbb{Q} \in \mathcal{P}(\Omega)$. The defining optimization problem is called *Monge-Kantorovich mass transportation problem*. It is known that ζ_c and $\hat{\mu}_c$ coincide for $\mathbb{P}, \mathbb{Q} \in \mathcal{P}_c(\Omega)$ with

$$\mathcal{P}_c(\Omega) = \left\{ \mathbb{Q} \in \mathcal{P}(\Omega) \middle| \int_\Omega \lambda(\omega) \mathbb{Q}[d\omega] < \infty \right\}.$$

Observe that $\mathcal{P}_c(\Omega) \subseteq \mathcal{P}_f(\Omega)$ since

$$|f(\omega, x)| \leq g(\|x\|) c(\omega, \omega_0) + |f(\omega_0, x)| \leq g(\rho)(\lambda(\omega) + \lambda(\omega_0)) + |f(\omega_0, x)|$$

for some fixed $\omega_0 \in \Omega$ and $\|x\| \leq \rho$.

Remark. If c has the form

$$c_h(\omega, \tilde{\omega}) := \|\omega - \tilde{\omega}\| \max\{1, h(\|\omega - \omega_0\|), h(\|\tilde{\omega} - \omega_0\|)\}, \qquad \omega, \tilde{\omega} \in \Omega,$$

where $\omega_0 \in \Omega$ is some fixed element and $h : \mathbb{R}^+ \to \mathbb{R}^+$ is continuous and nondecreasing, ζ_{c_h} is called *Fortet-Mourier (type) metric*. Note that we have only $\zeta_{c_h} \leq \hat{\mu}_{c_h}$, since c_h is not a metric in general. If $c(\omega, \tilde{\omega}) := \|\omega - \tilde{\omega}\|$, then ζ_c is also called *Kantorovich* or *L^1-Wasserstein metric*.

To connect Theorem 3.2 with ζ_c we observe $(g(\rho))^{-1} f(\cdot, x) \in F_c$ for all $x \in \mathbb{R}^r$ with $\|x\| \leq \rho$, so

$$d_{f,\rho}(\mathbb{P}, \mathbb{Q}) \leq g(\rho) \zeta_c(\mathbb{P}, \mathbb{Q}) \tag{3.6}$$

for $\mathbb{P}, \mathbb{Q} \in \mathcal{P}_f(\Omega)$ since

$$\begin{aligned}
d_{f,\rho}(\mathbb{P}, \mathbb{Q}) &= \sup_{\|x\| \leq \rho} \left| \int_\Omega f(\omega, x) \mathbb{P}[d\omega] - \int_\Omega f(\omega, x) \mathbb{Q}[d\omega] \right| \\
&\leq \sup_{\|x\| \leq \rho} \sup_{f \in F_{g(\|x\|)c}} \left| \int_\Omega f(\omega) \mathbb{P}[d\omega] - \int_\Omega f(\omega) \mathbb{Q}[d\omega] \right| \\
&= \sup_{\|x\| \leq \rho} \sup_{f \in F_c} \left| \int_\Omega g(\|x\|) f(\omega) \mathbb{P}[d\omega] - \int_\Omega g(\|x\|) f(\omega) \mathbb{Q}[d\omega] \right| \\
&\leq g(\rho) \sup_{f \in F_c} \left| \int_\Omega f(\omega) \mathbb{P}[d\omega] - \int_\Omega f(\omega) \mathbb{Q}[d\omega] \right| \\
&= g(\rho) \zeta_c(\mathbb{P}, \mathbb{Q}).
\end{aligned}$$

3.1.3 Scenario reduction

For the two-stage stochastic program (3.4), let us assume that f satisfies the condition (3.5) for some metric c and nondecreasing function g and that the original distribution $\mathbb{P}$ belongs to

[1] In general, $\zeta_c(\mathbb{P}, \mathbb{Q}) = 0$ does not imply $\mathbb{P} = \mathbb{Q}$ due to insufficiency of F_c.

$\mathcal{P}_c(\Omega)$. Let $S(\mathbb{P})$ be nonempty and bounded. Then from theorem 3.2 we obtain Lipschitz stability properties for the optimal value $v(\cdot)$ and the ε-approximate solution set $S_\varepsilon(\cdot)$ at $\mathbb{P}$ with respect to the probability (semi-)metric ζ_c. Thus we can use ζ_c as basis for approximating $\mathbb{P}$, i.e. the principle of *optimal scenario generation* for (3.4) may be formulated in the following way: Determine a discrete set $\Omega_I = \{\omega_i\}_{i\in I} \subset \Omega$ of scenarios in Ω of a given size $|I|$ and assign probabilities $\{q_i\}_{i\in I}$, $\sum_{i\in I} q_i = 1$, $q_i \geq 0$, $i \in I$, such that the distance to the original measure $\mathbb{P}$ in the sense of the metric ζ_c,

$$\zeta_c\left(\mathbb{P}, \sum_{i\in I} q_i\delta_{\omega_i}\right) \quad\rightarrow\quad \min. \tag{3.7}$$

Having a given discrete approximation $\mathbb{P} = \sum_{i\in I} p_i\delta_{\omega_i}$, $|I| < \infty$, the principle of *optimal scenario reduction* may be formulated as: Determine a set $J \subseteq I$ of given size $|J| = k$ and assign probabilities $\{q_j\}_{j\notin J}$, $\sum_{j\notin J} q_j = 1$, $q_j \geq 0$, $j \notin J$, to the remaining scenarios $\Omega_{I\setminus J}$ such that

$$\zeta_c\left(\mathbb{P}, \sum_{j\in I\setminus J} q_j\delta_{\omega_j}\right) \quad\rightarrow\quad \min. \tag{3.8}$$

While solving problem (3.7) is often hopeless in general settings, problem (3.8) represents a (large scale) linear problem, since $\zeta_c = \hat{\mu}_c$ over $\mathcal{P}_c(\Omega)$. We define the functional

$$\mathcal{D}(J;q) := \hat{\mu}_c\left(\sum_{i\in I} p_i\delta_{\omega_i}, \sum_{j\in I\setminus J} q_j\delta_{\omega_j}\right)$$

so that problem (3.8) takes the form

$$\min\left\{\mathcal{D}(J;q)\,\middle|\, J\subseteq I, |J| = k, \sum_{j\notin J} q_j = 1, q_j \geq 0 \forall j\right\}.$$

For a given set $J \subseteq I$ it can be shown that

$$\mathcal{D}(J) := \min\left\{\mathcal{D}(J;q)\,\middle|\,\sum_{j\notin J} q_j = 1, q_j \geq 0\forall j\right\} = \sum_{i\in J} p_i \min_{j\notin J} c(\omega_i,\omega_j) \tag{3.9}$$

and that the minimum is attained at

$$\bar{q}_j = p_j + \sum_{i\in J: j(i)=j} p_i,\ j\notin J, \quad \text{where } j(i) \in \operatorname*{Argmin}_{j\notin J} c(\omega_i,\omega_j),\ i\in J, \tag{3.10}$$

that is the probability p_i of a deleted scenario ω_i, $i \in J$, is assigned to one of the scenarios ω_j, $j \notin J$, which are closest to ω_i. (*optimal redistribution rule*)

It remains to find the set $J \subseteq I$ with $|J| = k$ which minimizes $\mathcal{D}(J)$. Since $\min_{J\subseteq I,|J|=k}\mathcal{D}(J)$ represents a SetCover problem, which is NP-hard, efficient solution algorithms are hardly available. But there are lower and upper bounds for the optimal $\mathcal{D}(J)$ known which will lead us to scenario reduction algorithms:

Theorem 3.3. *Let indices l_i and u_j are chosen recursively such that*

$$l_i = \operatorname*{argmin}_{l\in I\setminus\{l_1,\dots,l_{i-1}\}} p_l \min_{j\neq l} c(\omega_l,\omega_j), \qquad i = 1,\dots,k, \tag{3.11}$$

$$u_j = \operatorname*{argmin}_{u\in I\setminus\{u_1,\dots,u_{j-1}\}} \sum_{i\in I\setminus\{u_1,\dots,u_{j-1},u\}} p_i \min_{l\in\{u_1,\dots,u_{j-1},u\}} c(\omega_l,\omega_i), \qquad j = 1,\dots,|I|-k. \tag{3.12}$$

Then

$$\sum_{i=1}^{k} p_{l_i} \min_{j \neq l_i} c\left(\omega_{l_i}, \omega_j\right) \leq \min_{J \subseteq I, |J|=k} \mathcal{D}\left(J\right) \leq \sum_{i \in I \setminus \left\{u_1, \ldots, u_{|I|-k}\right\}} p_i \min_{j \in \left\{u_1, \ldots, u_{|I|-k}\right\}} c\left(\omega_i, \omega_j\right). \tag{3.13}$$

This theorem suggests two heuristic algorithms for the optimal reduction of $|I|$ scenarios to $|I|-k$ scenarios. While the first one determines k scenarios that have to be deleted, the second selects the $|I|-k$ scenarios to keep. The first one is called *backward reduction*, while the second one is called *forward selection*. One easily observes that the backward-reduction algorithm determines an optimal J in the case $k=1$ and the forward selection algorithm selects an optimal J in the case $k=|I|-1$. Algorithm (3.11) have been improved ([12]) to the so-called *simultaneous backward reduction* algorithm:

$$l_i = \operatorname*{argmin}_{l \in I \setminus \{l_1, \ldots, l_{j-1}\}} \sum_{k \in \{l_1, \ldots, l_{j-1}, l\}} p_k \min_{j \in I \setminus \{l_1, \ldots, l_{j-1}, l\}} c\left(\omega_k, \omega_l\right), \qquad i = 1, \ldots, k.$$

We are now arrived at a point, where we can reduce a given discrete approximation $\mathbb{P}$ by k scenarios to a rougher approximation $\mathbb{Q}$ and estimate the caused perturbation in the optimal value of the convex two-stage stochastic program by

$$\left|v\left(\mathbb{P}\right) - v\left(\mathbb{Q}\right)\right| \leq g\left(\rho\right) \min_{J \subseteq I, |J|=k} \mathcal{D}\left(J\right)$$

where the optimal $\mathcal{D}(J)$ is further bounded by (3.13). An open question is the optimal choice of the parameter k: How many scenarios are needed to keep the distance $|v(\mathbb{P}) - v(\mathbb{Q})|$ small? One possibility is to start with a small k and to increase it as long as the lower bound on the optimal $\mathcal{D}(J)$ given by the (simultaneous) backward reduction algorithm is "small enough", or, similarly, to start with a big k and decrease it's value as long as the upper bound on the optimal $\mathcal{D}(J)$ given by the forward selection algorithm is "too high". But since we do not know the parameter $g(\rho)$ it is difficult to give a quantitative statement for "small enough" or "too high". Observe, that the function $g(\cdot)$ depends primarily on the data of the stochastic program and only secondarily on the probability measure $\mathbb{P}$, whereas $\mathcal{D}(J)$ depends primarily on $\mathbb{P}$ and the function f intervenes only in the definition of the metric c.

This motivates to formulate an algorithm for scenario-reduction which utilizes a second measure on the gap $|v(\mathbb{P}) - v(\mathbb{Q})|$. Hence, we will proceed with a formulation of the abstract algorithm from section 2.5 for multistage stochastic programs. By starting with a rough approximation of the initial scenario-tree (i.e. starting with a big k) we refine the scenario-tree (i.e. diminish k) by use of the forward-selection algorithm as long as the achievable improvement in the optimal value seems to be high enough. To compute the achievable improvement, a test-refinement of the current aggregation is used. To ensure that this test-refinement is close enough to the initial tree, we bound the distance of the probability measures induced by the test-refinement and the initial tree by (3.13). But in contrast to the scenario-reduction algorithm by backward-reduction or forward-selection, we will add scenarios to the final approximation only in the case that a considerable improvement of the approximation of the stochastic program seems achievable. Thus a too large test-refinement due to an inappropriate choice of the upper bound on $g(\rho) \min_J \mathcal{D}(J)$ does not lead immediately to a blowup of the final approximation.

Since the adaptive scenario refinement algorithm will work with the multistage stochastic program (MSP) instead of it's deterministic equivalent (DA) we will proceed to formulate (MSP) for discrete probability distributions, scenario trees and aggregated scenario trees.

3.2 Scenario tree formulation

For the multistage stochastic program (MSP), assume ξ to have a finite support $\Omega_I = \{\omega_i\}_{i \in I} \subseteq \Omega$ for some finite set $I \subset \mathbb{N}$. Then $\mathbb{P} = \sum_{i \in I} p_i \delta_{\omega_i}$ with $p_i := \mathbb{P}(\omega_i) > 0$, $\sum_{i \in I} p_i = 1$. Define now the decision variables $x_{i,t} := x_t(\omega_i)$ and the process values $\xi_{i,t} := \xi_t(\omega_i)$, $\zeta_{i,t} := \zeta_t(\omega_i)$, $i \in I$. Let $\mathcal{C}_t$ be a finite generator of $\mathcal{F}_t$, i.e. $\mathcal{F}_t = \sigma(\mathcal{C}_t)$, $\mathcal{C}_t = \{E_{t,j}\}_{j\in J_t}$, s.t. $\bigcup_{j\in J_t} E_{t,j} = \Omega_I$, $E_{t,j} \cap E_{t,k} = \emptyset$, $j \neq k$. The nonanticipativity condition $x_t = \mathbb{E}[x_t | \mathcal{F}_t]$ reads now

$$x_{i,t} = \sum_{j\in J_t}\left(\chi_{E_{t,j}}(\omega_i)\cdot\frac{1}{\mathbb{P}(E_{t,j})}\sum_{\omega_{i'}\in E_{t,j}} p_{i'}x_{i',t}\right) = \sum_{j\in J_t}\left(\chi_{E_{t,j}}(\omega_i)\cdot\sum_{\omega_{i'}\in E_{t,j}} \bar{p}_{i'}x_{i',t}\right) \qquad \forall i \in I$$

with $\bar{p}_i = \frac{1}{\mathbb{P}(E_{t,j})}p_i$, where j is chosen such that $\omega_i \in E_{t,j}$.

The discretization of (MSP) w.r.t. the probability measure $\mathbb{P}$ is now defined by

$$\begin{array}{lllll}
\min\limits_x & \sum\limits_{i\in I} p_i f(x_{i,1},\dots,x_{i,T},\xi_{i,1},\dots,\xi_{i,T}) & & & \\
\text{s.t.} & g_t(x_{i,1},\dots,x_{i,t},\zeta_{i,t}) \le 0 & & i\in I & t=1,\dots,T \\
 & x_{i,t} \in X_t(\zeta_{i,t}) & & i\in I & t=1,\dots,T \\
 & \sum\limits_{\omega_{i'}\in E_{t,j}} \bar{p}_{i'}x_{i',t} = x_{i,t} & j \text{ s.t. } \omega_i\in E_{t,j} & i\in I & t=1,\dots,T
\end{array} \qquad (\text{MSP}[I])$$

where $x = \{x_{i,t}\}_{i\in I, t=1,\dots,T}$.

From the non-anticipativity-conditions we observe $x_{i,t} = x_{i',t}$ $\forall \omega_i, \omega_{i'} \in E_{t,j}$, $j \in J_t$, $t = 1,\dots,T$. Explicit consideration of these constraints allows for a reduction of the dimension of (MSP[I]): We construct a graph which nodes are the sets $E_{t,j}$, $j \in J_t$, $t = 1,\dots,T$ and with edges between $E_{t,j}$ and $E_{t+1,k}$, iff $E_{t+1,k} \subseteq E_{t,j}$. This graph has the structure of a rooted tree with root-node $E_{11} = \Omega$ and leafs $E_{T,i} = \{\omega_i\}$ (since $\mathcal{F}_T = \mathcal{F}$). So let $\bar{N} \subset \mathbb{N}$ be a numbering of the nodes of this graph, such that node $n = 1$ is the root-node. For the sake of simplicity we omit extra notation for the edges in that tree and will just write $\bar{N}$ for the *initial scenario tree*. Later we will consider subtrees of the initial scenario tree, but their structure will be induced by the original tree, wherefore omitting of edges will cause no obscurities.

Notation. For $n \in \bar{N}$ and $N \subseteq \bar{N}$ let

- n_- be the predecessor and $N_+(n)$ the set of successors
- $n_{-k} := \left(n_{-(k-1)}\right)_-$ with $n_{-1} := n_-$
- $N_{+k}(n) := N_{+(k-1)}(N_+(n))$ with $N_{+1}(\cdot) := N_+(\cdot)$
- $t(n)$ such that $n_{-t(n)+1} = 1$ (the depth of n)
- $N_t := \{n \in N \,| t(n) = t\}$ all nodes with depth (or timestage) t in N
- $p(n) := (1,\dots,n_-,n)$ the path from the root node to node n
- $i(n) \in I$ the scenario index which belongs to $n \in \bar{N}_T$.
- $p(i)$ the path which corresponds to scenario $i \in I$: $p(i) := p(n)$ with $n \in \bar{N}_T$ s.t. $i(n) = i$

We define recursively node probabilities $\pi_n := p_{i(n)} = \mathbb{P}\left[\{\omega_{i(n)}\}\right]$, $n \in \bar{N}_T$, and $\pi_n := \sum_{m\in\bar{N}_+(n)} \pi_m$, $n \in \bar{N} \setminus \bar{N}_T$, such that $\sum_{n\in\bar{N}_t} \pi_n = 1$, $t = 1,\dots,T$.

Let further $x^n := x_{i(n),t(n)}$, $\xi^n := \xi_{i(n),t(n)} := \xi_{t(n)}(\omega_i)$, $\zeta^n := \zeta_{i(n),t(n)} = \xi^{p(n)} := \left(\xi^1,\dots,\xi^{n_-},\xi^n\right)$. Then the scenario-tree-formulation of (MSP[I]) reads

$$\begin{array}{lll}
\min\limits_x & \sum\limits_{n\in\bar{N}_T} \pi_n f\left(x^{p(n)},\zeta^n\right) & \\
\text{s.t.} & g_{t(n)}\left(x^{p(n)},\zeta^n\right) \le 0 & n\in\bar{N} \\
 & x^n \in X_{t(n)}(\zeta^n) & n \in \bar{N}
\end{array} \qquad (\text{MSP}[\bar{N}])$$

where $x = \{x^n\}_{n\in\bar{N}}$.

3.3 Aggregations and their refinement

For multistage stochastic programs, we need to take the special structure of the node set $\bar{N}$ into account when defining a partition $\mathcal{N}$. Partitions are here considered as aggregation of scenarios: We select some scenarios $J \subseteq I$, consider the tree which is induced by J, i.e. the paths $\{p(i)\}_{i\in J}$, and define an assignment from the remaining scenarios $I \setminus J$ to the selected scenarios J. The partition $\mathcal{N}$ is now defined to be the partition of $\bar{N}_T$ which represents this assignment.

More precise, we define a mapping

$$\mathrm{agg} : \bar{N} \to \bigcup_{i\in J} p(i)$$

such that $\mathrm{agg}\left(\bar{N}_t\right) \subseteq \bar{N}_t$, $t = 1, \ldots, T$, $\mathrm{agg}\big|_{p(i)} \equiv \mathrm{id}_{p(i)}$, $i \in J$, and the predecessor-operation commute with agg:

$$\mathrm{agg}(n)_- = \mathrm{agg}(n_-) \quad \forall n \in \bar{N} \setminus \{1\}. \tag{3.14}$$

Now the partition $\mathcal{N}$ is defined as the set of preimages of agg w.r.t. $\bar{N}_T$, i.e.

$$\mathcal{N} = \left\{N \subseteq \bar{N}_T \,|N \in \mathcal{N}\right\} := \left\{\mathrm{agg}^{-1}(n) \,\middle|\, n \in \bar{N}_T, i(n) \in J\right\}.$$

The nodes which are invariant to an application of agg are called *representative*, because they "represent" the selected scenarios J in the initial scenario tree $\bar{N}$. Thus for $N \in \mathcal{N}$ we define $n_T(N)$ to be that node $n \in N$ with $\mathrm{agg}(n) = n$, and further on $n_t(N) := n_T(N)_{-T+t}$, $t = 1, \ldots, T-1$ (Observe that also $\mathrm{agg}(n_t(N)) = n_t(N)$, $t = 1, \ldots, T$.). Let $p(N) := p(n_T(N)) = (n_1(N), \ldots, n_T(N))$, $N \in \mathcal{N}$. The set of all representative nodes is denoted by $\bar{N}|_{\mathcal{N}}$, $\bar{N}|_{\mathcal{N}} = \bigcup_{i\in J} p(i)$. The tree induced by $\bar{N}|_{\mathcal{N}}$ is called *aggregated scenario tree* and will be denoted by $\bar{N}|_{\mathcal{N}}$ as well. The node probabilities for the leaves of $\bar{N}|_{\mathcal{N}}$ are defined by $\pi_N := \sum_{n\in N} \pi_n$, $N \in \mathcal{N}$.

Then we define the aggregated problem of $(\mathrm{MSP}[\bar{N}])$ by restricting the decision-process and constraints to the scenarios J or, in other words, by using the aggregated scenario tree for them. For the objective function, we average over aggregated scenarios, which simplifies the comparison with the objective functions for an refinement of $\mathcal{N}$. Thus the aggregated problem reads:

$$\begin{array}{lll} \min\limits_x & \sum\limits_{N\in\mathcal{N}} \sum\limits_{n\in N} \pi_n f\left(x^{p(N)}, \zeta^n\right) & \\ \text{s.t.} & g_{t(n)}\left(x^{p(n)}, \zeta^n\right) \le 0 & n \in \bar{N}|_{\mathcal{N}} \\ & x^n \in X_{t(n)}(\zeta^n) & n \in \bar{N}|_{\mathcal{N}} \end{array} \tag{MSP[$\mathcal{N}$]}$$

For the convex relaxation we assume $\check{X}_{t(n)}(\zeta^n)$ to be a convex superset of $X_{t(n)}(\zeta^n)$ for all $n \in \bar{N}$, $\check{f}(\cdot, \zeta^n)$ is a convex underestimator of $f(\cdot, \zeta^n)$ on $\times_{t=1}^T \check{X}_t\left(\zeta^{n_{-T+t}}\right)$ for all $n \in \bar{N}_T$ and $\check{g}_{t(n)}(\cdot, \zeta^n)$ is a convex underestimator of $g_{t(n)}(\cdot, \zeta^n)$ on $\check{X}_{t(n)}(\cdot, \zeta^n)$ for all $n \in \bar{N}|_{\mathcal{N}}$. Then we have

$$\begin{array}{lll} \min\limits_x & \sum\limits_{N\in\mathcal{N}} \sum\limits_{n\in N} \pi_n \check{f}\left(x^{p(N)}, \zeta^n\right) & \\ & \check{g}_{t(n)}\left(x^{p(n)}, \zeta^n\right) \le 0 & n \in \bar{N}|_{\mathcal{N}} \\ \text{s.t.} & x^n \in \check{X}_{t(n)}(\zeta^n) & n \in \bar{N}|_{\mathcal{N}} \\ & c(x) \le 0 & c \in \mathcal{C}_{\mathcal{N}} \end{array} \tag{CMSP[$\mathcal{N}$]}$$

For a refinement $J' \subseteq I$, $J \subseteq J'$, we define again an assignment $\mathrm{agg}' : \bar{N} \to \bigcup_{i\in J'} p(i)$ and a partition $\mathcal{N}'$ of $\bar{N}$ such that $\bigcup_{i\in J'} p(i) = \bar{N}|_{\mathcal{N}'}$ and $\mathcal{N}' \preceq \mathcal{N}$. To get $\mathcal{N}' \preceq \mathcal{N}$ we require

$$\mathrm{agg}(\mathrm{agg}'(n)) = \mathrm{agg}(n) \qquad \forall n \in \bar{N}. \tag{3.15}$$

Further on let $n'_t(N) := \{n_t(N') \,|N' \in \mathcal{N}'|_N\}$, $N \in \mathcal{N}$, be the set of representative nodes in the refinement of N (at timestage t), $t = 1, \ldots, T$. ($n'_t(N)$ is the set of invariant nodes of agg$'$ which map to $n_t(N)$ by agg.) The condition $J \subseteq J'$ provides also $\bar{N}|_{\mathcal{N}} \subseteq \bar{N}|_{\mathcal{N}'}$.

When formulating problem (Proj[$\mathcal{N}, \mathcal{N}'$]) for MSPs, we will need to restrict a decision process on $\bar{N}|_{\mathcal{N}'}$ to one on $\bar{N}|_{\mathcal{N}}$, i.e. the $W_N x_N = 0$ constraints will have the form $x^{n_t(N)} = x^{n_t(N')}$ for all $N' \in \mathcal{N}'|_N$, $t = 1, \ldots, T$. Additional bundling between a scenario $i(n_T(N'))$ and $i(n_T(M))$ with $N' \in \mathcal{N}'|_N$ and disjoint $N, M \in \mathcal{N}$ is thereby avoided due to the requirements (3.14) and (3.15):

Proposition 3.4. *Let $N, M \in \mathcal{N}$ and $t \in \{1, \ldots, T\}$. Then for all $N' \in \mathcal{N}'|_N$ with $n_t(N') = n_t(M)$ it holds that $n_t(N) = n_t(M)$.*

Proof. Let $N, M \in \mathcal{N}$, $N' \in \mathcal{N}'|_N$ and $t \in \{1, \ldots, T\}$ chosen such that $n_t(N') = n_t(M)$. Let $\tau := T - t$. Then

$$\begin{aligned} n_t(M) &= \operatorname{agg}(n_t(M)) \\ &= \operatorname{agg}(n_t(N')) \\ &= \operatorname{agg}(n_T(N'))_{-\tau} \qquad \text{by (3.14)} \\ &= (n_T(N))_{-\tau} \qquad \text{using } n_T(N') \in N, \text{ from (3.15) follows } \operatorname{agg}(n_T(N')) = n_T(N) \\ &= n_t(N) \end{aligned}$$

□

We will state here another property of the mapping agg which will lead us later to an algorithm that computes agg basing on leaves (i.e. scenarios), selected from $\bar{N}_T$. It is, that the distance of the nodes n to agg(n) is not longer than the distance to other representative nodes at the same timestage. The distance dist(n, m) for two nodes $n, m \in \bar{N}$ is here defined as the length of the shortest path between n and m in the initial tree. Observe, that dist(n, m) is even if $t(n) = t(m)$.

Proposition 3.5. *Let the mapping* $\operatorname{agg} : \bar{N} \to \bar{N}|_{\mathcal{N}}$ *be given as before. Then for all $n \in \bar{N}$*

$$\operatorname{dist}(n, \operatorname{agg}(n)) = \min_{m \in \bar{N}|_{\mathcal{N}} \,:\, t(m) = t(n)} \operatorname{dist}(n, m).$$

Proof. Let $n \in \bar{N}$ and $\tau \geq 0$ s.t. $2\tau = \operatorname{dist}(n, \operatorname{agg}(n))$, thus τ is chosen minimal such that $n_{-\tau} = \operatorname{agg}(n)_{-\tau}$. Let further $m \in \bar{N}|_{\mathcal{N}}$ with $t(m) = t(n)$ and $2\tau_* = \operatorname{dist}(n, m)$. Then we have to show, that $\tau \leq \tau_*$.

Since $n_{-\tau_*} = \operatorname{agg}(n)_{-\tau_*}$ we know $\operatorname{agg}(n_{-\tau_*}) = n_{-\tau_*}$ (all nodes on a path from the root to a representative node are representative). From (3.14) we get

$$\operatorname{agg}(n)_{-\tau_*} = \operatorname{agg}(n_{-\tau_*}) = n_{-\tau_*},$$

thus $\operatorname{dist}(\operatorname{agg}(n), n_{-\tau_*}) = \tau_*$ and therefore,

$$2\tau = \operatorname{dist}(\operatorname{agg}(n), n) \leq \operatorname{dist}(\operatorname{agg}(n), n_{-\tau_*}) + \operatorname{dist}(n_{-\tau_*}, n) = \tau_* + \tau_*.$$

□

3.3.1 Choice of the metric c

We will identify a scenario ω_i, $i \in I$, with its realization $\xi_i = \xi(\omega_i)$ and define the metric c as a Fortet-Mourier (type) metric c on $\times_{t=1}^T \mathbb{R}^{s_t}$ with $\xi(\omega_0) := 0$ and $h \equiv 0$, i.e. $c(\xi_i, \xi_j) := \|\xi_i - \xi_j\|$. The norm $\|\cdot\|$ on $\times_{t=1}^T \mathbb{R}^{s_t}$ is defined by $\|\xi_i\| := \|\xi_{i,1}\| + \ldots + \|\xi_{i,T}\|$ for given norms on $\mathbb{R}^{s_t}$, $t = 1, \ldots, T$.

Further on let $c_t(\zeta_{i,t}, \zeta_{j,t}) := \sum_{\tau=1}^t \|\xi_{i,\tau} - \xi_{j,\tau}\|$, $t = 1, \ldots, T$, so that $c \equiv c_T$.

3.3.2 Initial aggregation

A rough initial aggregation is defined by $\mathcal{N} := \left\{\bar{N}_T\right\}$ where the representing scenario is selected by forward selection (recall equation (3.12)): select $n \in \bar{N}_T$ s.t.

$$\sum_{m \in \bar{N}_T \setminus \{n\}} p_m c_T\left(\xi^m, \xi^n\right) \quad \rightarrow \quad \min$$

and set $\operatorname{agg}(m) := n_{-T+t(m)}$, $m \in \bar{N}$.

An initial refinement $\mathcal{N}'$ is defined by refining $\mathcal{N}$.

3.3.3 Refining an aggregation

Let $(\mathcal{N}, \operatorname{agg})$ be a partition of $\bar{N}_T$ and an aggregation of $\bar{N}$. Let $\mathcal{M} \subseteq \mathcal{N}$ be a set of leafsets, we aim to refine. A refinement $\mathcal{N}'$ of $\mathcal{N}$ is then constructed by selecting a given number of scenarios for each $M \in \mathcal{M}$ and updating the aggregation for the remaining scenarios (such that a mapping agg' that satisfies the conditions (3.14) and (3.15) is obtained). To remember, which representative nodes are added or at least touched during such a refinement operation, a flag is set (e.g. flag(m) :=true). To achieve a best possible reduction of the distance between the probability measures induced by the scenario trees $\bar{N}\,|_{\mathcal{N}}$ and $\bar{N}$ in terms of the probability metric $\hat{\mu}_c$ from section 3.1 we use the fast forward selection algorithm from [12], which is equivalent to the forward selection algorithm (3.12) but faster in running time, to select the scenarios. Following section 3.1 we measure by $\tilde{\mathcal{D}}(N)$,$N \in \mathcal{N}$, the distance between the scenario tree which is induced by all scenarios in N and the scenario tree which consists of scenario $i(N')$, $N' \in \mathcal{N}'\,|_N$, only:

$$\tilde{\mathcal{D}}(N) := \sum_{n \in N \setminus \{n_T(N') | N' \in \mathcal{N}'|_N\}} p_{i(n)} \min_{N' \in \mathcal{N}'|_N} c\left(\xi_{i(n_T(N'))}, \xi_{i(n)}\right).$$

Further let

$$\tilde{\mathcal{D}}(\mathcal{N}') := \sum_{N \in \mathcal{N}} \tilde{\mathcal{D}}(N)$$

be a measure for the distance between the probability measures, induced by the trees $\bar{N}\,|_{\mathcal{N}'}$ and $\bar{N}$. Observe that this measure is not fully consistent with the measure $\mathcal{D}(J)$ defined in (3.9) since the forward-selection algorithm will be restricted to subtrees of $\bar{N}$ when refining an aggregation $N \in \mathcal{N}$, i.e. the minimization in (3.12) is restricted to subsets of I. Furthermore we will not follow the optimal redistribution rule (3.10) for the sake of computational efficiency. But $\tilde{\mathcal{D}}(\mathcal{N}')$ can be seen as an upper bound on

$$\mathcal{D}\left(I \setminus \{i(N') | N' \in \mathcal{N}'\}\right) = \sum_{n \in \bar{N}_T \setminus \{n_T(N') | N' \in \mathcal{N}'\}} p_{i(n)} \min_{N' \in \mathcal{N}'} c\left(\xi_{i(n_T(N'))}, \xi_{i(n)}\right)$$

since $\bigcup_{N \in \mathcal{N}} \left(N \setminus \{n_T(N') | N' \in \mathcal{N}'|_N\}\right) = \bar{N}_T \setminus \{n_T(N') | N' \in \mathcal{N}'\}$.

Algorithm 3.6 (Scenario tree refinement).
Input: An aggregated scenario tree $(\mathcal{N}, \operatorname{agg})$, a set $\mathcal{M} \subseteq \mathcal{N}$ of leafsets to refine and limits $s_M \in \{1, \dots, |M|\}$, $M \in \mathcal{M}$.
Compute:
 $\operatorname{agg}' := \operatorname{agg}$, $\mathcal{N}' := \mathcal{N} \setminus \mathcal{M}$
 for all $M \in \mathcal{M}$ **do**
  $J := M \setminus \{n_T(M)\}$ {scenarios to delete, but keep $i(n_T(M))$ in any case}
  Define scenario distances $c_{n,k} := \min\left\{c_T\left(\zeta^n, \zeta^{n_T(M)}\right), c_T\left(\zeta^n, \zeta^k\right)\right\}$, $n, k \in J$.
  while $|J| \geq |M| - s_M$ **do**

Pick $m \in J$ s.t.

$$\mathcal{D} := \sum_{n \in J \setminus \{m\}} p_n c_{n,m}$$

is minimized and set $J := J \setminus \{m\}$, i.e. keep m.
Update the scenario distances: $c_{n,k} := \min \{c_{n,k}, c_{n,m}\}$, $n, k \in J$.
end while
for all $m \in M \setminus J$ **do**
agg$'(m) := m$
flag(m') :=true
end for
update agg$'$ by calling aggregate$(M \setminus J, (n_T(M)))$
$\mathcal{N}' := \mathcal{N}' \cup \left\{ \text{agg}'^{-1}(m) \,|m \in M \setminus J \right\}$
$\tilde{\mathcal{D}}(M) := \mathcal{D}$
end for
Output: $\mathcal{N}'$, agg$'$, $\tilde{\mathcal{D}}(\mathcal{N}')$

The algorithm aggregate(M^t, p) updates the aggregations of subtrees in $\bar{N}$ which root is a predecessor of a node in M^t, where M^t is a set of representative nodes on timestage t. The aggregations are computed such that the resulting mapping agg$'$ fulfills the conditions (3.14) and (3.15). The aggregation of a node $n \in M^t$ is updated as soon as either n_- is made representative (algorithm 3.7, line 8) or its aggregation is updated (algorithm 3.8, line 4), but only in the case, that agg$(n) \in p(M)$, i.e. we update only the aggregation for nodes which were aggregated to the path we refine currently. This and the condition, that a path is bundled only with paths which lead to a leaf in M (thats what the flag stands for) ensures condition (3.15). The idea to update an aggregation of n as soon as the aggregation of it's predecessor is updated originated from the shortest-distance-property shown in Proposition 3.5.

Algorithm 3.7 (Node aggregation). aggregate(M^t, p):
Input: a set $M^t \subseteq \bar{N}_t$ of representative nodes and a reference path $p = (p_t, \ldots, p_T)$
Compute:
1: **if** $t = 1$ **then**
2: **return**
3: **end if**
4: **for all** $m \in M^t$ **do**
5: define m_- to be representative: agg$'(m_-) := m_-$
6: flag(m_-) :=true
7: **for all** $n \in N_+(m_-)$ which are not representative and have agg$(n) = p_t$ **do**
8: update agg$'\,|_{N_*(n)}$ by calling bundle$(m_-, n, (p_{t+1}, \ldots, p_T))$ where $N_*(n) := \bigcup_{\tau=0}^{T-t(n)} N_{+\tau}(n)$ denotes the subtree of n in $\bar{N}$.
9: **end for**
10: **end for**
11: $M^{t-1} := \{m_- \,|m \in M^t\}$
12: proceed updating agg$'$ by calling aggregate$\big(M^{t-1}, \big((p_t)_-, p_t, \ldots, p_T\big)\big)$
Output: an updated aggregation

Algorithm 3.8 (Node bundling). bundle(m_-, n, p):
Input: a representative node m_-, a node n which aggregation has to be updated, and a reference path $\left(p_{t(n)+1}, \ldots, p_T\right)$
Compute:
1: select a representative successor $m \in N_+(m_-)$ with flag(m) =true, such that $c_{t(n)}(\zeta^n, \zeta^m)$ is minimized
2: bundle n with m: agg$'(n) := m$.
3: **for all** $n_+ \in N_+(n)$ with agg$(n_+) = p_{t(n)+1}$ **do**

4: bundle$\left(m, n_+, \left(p_{t(n)+2}, \ldots, p_T\right)\right)$
5: **end for**
Output: an updated aggregation for $N_*(n)$

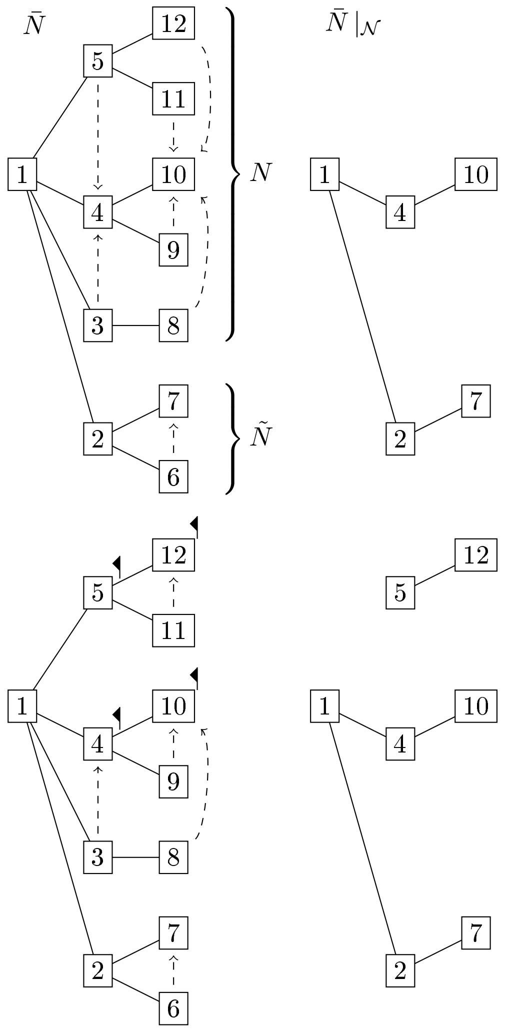
Proposition 3.9. *The mapping* agg′ *computed by algorithm 3.6 satisfies the requirements (3.14) and (3.15).*

Proof. agg′ is modified in line 5 of algorithm 3.7 and line 2 of algorithm 3.8.

To condition (3.14): Line 5 in algorithm 3.7 ensures the invariance of the nodes $\bar{N}\,|_{\mathcal{N}'}$ w.r.t. agg′. In line 2 of algorithm 3.8, n is aggregated with a node $m \in N_+(m_-)$ with flag(m) set. n_- was either aggregated in line 2 of 3.8 by a preceding call or by line 5 in algorithm 3.7. In both cases n_- was aggregated with m_-, i.e. $\mathrm{agg}'(n_-) = m_-$. Combining this with $\mathrm{agg}'(n) \in N_+(m_-)$ yields $\mathrm{agg}'(n)_- = m = \mathrm{agg}'(n_-)$.

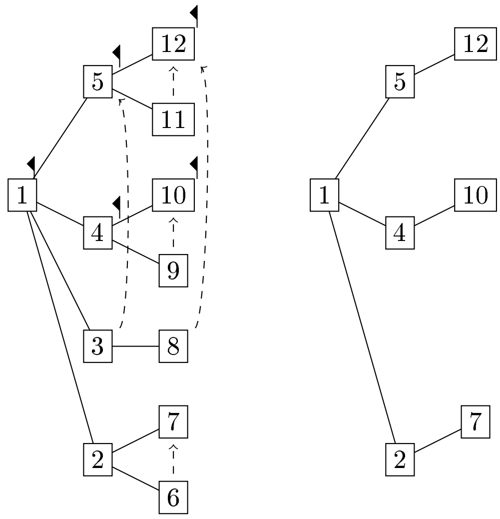
To condition (3.15): Line 7 of algorithm 3.7 and line 3 of algorithm 3.8 ensure, that only the aggregation for nodes n with $\mathrm{agg}(n) = n_{t(n)}(M)$ is updated (it holds $p_{t(n)} = n_{t(n)}(M)$ since $p_T := n_T(M)$ from algorithm 3.6). Due to the condition, that a node n is bundled only with a representative node which flag is set, it is ensured that $\mathrm{agg}(\mathrm{agg}'(n)) = n_{t(n)}(M)$. □

We finish this section by an illustrative example of the refinement algorithm 3.6:



Example. Consider an initial scenario tree $\bar{N}$ on 12 nodes and 3 timestages. Let $\mathcal{N} := \{N, \tilde{N}\}$ with $N := \{8, 9, 10, 11, 12\}$, $\tilde{N} := \{6, 7\}$, $n_3(\{6,7\}) = 7$ and $n_3(\{8,9,10,11,12\}) = 10$ be a given aggregation. The current aggregation mapping agg is indicated by the dashed arrows. We aim to refine $\mathcal{M} := \{N\}$ with $s_N = 1$. Then the algorithm 3.6 starts to select a node $m \in J := N \setminus \{10\}$ s.t. $\sum_{n \in J \setminus \{m\}} p_n c_{n,m}$ is minimized. Let the minimum be attained for $m = 12$, so $M \setminus J = \{10, 12\}$. For the aggregation mapping we set $\mathrm{agg}'(12) := 12$ and the flag is set for the nodes 10 and 12. Then algorithm 3.7 is called with $M^3 = \{10, 12\}$ and $p = (p_3) = (10)$.
Algorithm 3.7 adds the predecessor of node $m = 12$ (node $m_- = 5$) into the aggregated tree by setting $\mathrm{agg}'(5) := 5$ and sets the flag for node 5. Then for all $n_+ \in N_+(5) = \{11, 12\}$ which are not representative and are aggregated to node $p_3 = 10$, i.e. for $n_+ = 11$, algorithm 3.8 is called with $m_- = 5$, $n = 11$ and $p = ()$.
Here, a representative node $m \in N_+(5) = \{11, 12\}$ with flag set is selected that minimizes the distance $c_3\left(\zeta^{11}, \zeta^m\right)$. The only possibility is $m = 12$, so that $\mathrm{agg}'(11)$ is set to 12. $N_+(11)$ is empty, so algorithm 3.8 ends and we are back at line 4 of algorithm 3.7.
Now, the lines 4 to 10 are performed for $m = 10$: The flag is set for node 4 and the aggregation for node 9 is reconsidered, but evaluates to $\mathrm{agg}'(9) = 10$. Next, algorithm 3.7 calls itself with $M^2 := \{4, 5\}$ and $p = (5, 12)$.

For $m = 5$ (in line 4), the algorithm tries to add node 1 to the aggregated tree, setting $\text{agg}'(1) := 1$, and sets the flag for node 1. From line 7-8 we see, that algorithm 3.8 is called for $n = 3$ (with $m_- = 1$ and $p = (12)$).
Here, the representative nodes $m \in N_+(1) = \{2, 3, 4, 5\}$ with flag set are checked. This are the nodes 4 and 5. Assume $c_2(\zeta^3, \zeta^5) < c_2(\zeta^3, \zeta^4)$, so that $m = 5$ is selected. $\text{agg}'(3)$ is set to 5 and the bundle method is called for all $n_+ \in N_+(3) = \{8\}$ with $m_- = 5$ and empty $p = ()$.
Reconsidering the aggregation for node 8 results in $\text{agg}'(8) := 12$, the only possibility in this case. Back to algorithm 3.7, the lines $5 - 9$ for $m = 4$ (we had $M^2 = \{4, 5\}$) will not change the graph. Finally, algorithm 3.7 is called with $M^1 = \{1\}$ and $p = (1, 5, 12)$. This ends the refinement procedure.



3.4 Decomposition

Next to the aggregation and refinement algorithms, a second important component in the scenario reduction algorithm are convex relaxations and cut generation. In section 2.4.2 we proposed column generation based on Lagrangian decomposition to compute Lagrangian Cuts. Dual decomposition methods exploits a separable structure in the optimization problem. Constraints which interfere with this separable structure are added to the objective function, provided by dual variables. For multistage stochastic programs, three dual decomposition methods seem to be promising: *Scenario decomposition* dualizes the nonanticipativity constraints of (MSP), so that the discretized form (MSP$[I]$) decomposes in many smaller problems, one for each scenario. Decoupling the time-stages leads to the so-called *nodal decomposition* which dualizes the time-coupling constraints $g_t(x_1, \ldots, x_t, \zeta_t)$ of (MSP). In [6] it is shown, that scenario-decomposition provides a smaller gap than nodal decomposition. A third approach is based on a separable structure which is not given by the stochastic or time-discretization of the problem, but by the optimization model itself. Large models with network-structure often provide such separability (see section 3.6.3 for an example) and sometimes specialized solution algorithms for the subproblems are available that allow to speed up the solution process. This so-called *geographical decomposition* will be our favorite, but for the sake of completeness we give a formulation of the other two first. Comparison of the duality gaps between geographic decomposition and scenario and nodal decomposition, respective, can be found in [6].

Next to these dual decomposition methods, the so-called nested Benders decomposition has been successfully applied to linear stochastic programs ([5]). Similar to nodal decomposition, it relaxes the time-coupling constraints of (MSP$\left[\bar{N}\right]$) to obtain subproblems that are attached to each node of the scenario tree. We will give a short introduction into this decomposition method at the end of this chapter.

3.4.1 Scenario decomposition

Consider the multistage optimization problem (MSP). Let $\lambda_t \in L^1(\Omega, \mathcal{F}, \mathbb{P}; \mathbb{R}^{r_t})$, $t = 1, \ldots, T$, denote L^1-bounded $\mathcal{F}$-measurable functions. Then the scenario decomposition is defined by dualizing the nonanticipativity constraints,

$$\begin{array}{lll} \max\limits_{\lambda=(\lambda_t)_{t=1}^T} \min\limits_{x} & L_{\text{scen}}(x,\xi;\lambda) & \\ \text{s.t.} & g_t(x_1,\dots,x_t,\zeta_t) \le 0 & \mathbb{P}\text{-a.s.} \quad t=1,\dots,T \\ & x_t \in X_t(\zeta_t) & \mathbb{P}\text{-a.s.} \quad t=1,\dots,T \end{array}$$

with the Lagrange functions

$$L_{\text{scen}}(x,\xi;\lambda) := \mathbb{E}\left[f(x,\xi) + \sum_{t=1}^{T} \langle \lambda_t, x_t - \mathbb{E}[x_t \,|\mathcal{F}_t]\rangle\right].$$

Now observe

$$\begin{aligned} \mathbb{E}[\langle \lambda_t, \mathbb{E}[x_t\,|\mathcal{F}_t]\rangle] &= \mathbb{E}[\mathbb{E}[\langle \lambda_t, \mathbb{E}[x_t\,|\mathcal{F}_t]\rangle\,|\mathcal{F}_t]] \\ &= \mathbb{E}[\langle \mathbb{E}[\lambda_t\,|\mathcal{F}_t], \mathbb{E}[x_t\,|\mathcal{F}_t]\rangle] \\ &= \mathbb{E}[\mathbb{E}[\langle \mathbb{E}[\lambda_t\,|\mathcal{F}_t], x_t\rangle\,|\mathcal{F}_t]] \\ &= \mathbb{E}[\langle x_t, \mathbb{E}[\lambda_t\,|\mathcal{F}_t]\rangle] \end{aligned}$$

for $t = 1,\dots,T$, so that the Lagrange function can be written as

$$L_{\text{scen}}(x,\xi;\lambda) := \mathbb{E}\left[f(x,\xi) + \sum_{t=1}^{T} \langle x_t, \lambda_t - \mathbb{E}[\lambda_t\,|\mathcal{F}_t]\rangle\right].$$

Assuming now a discrete distribution $\mathbb{P}$, i.e. migrating from (MSP) to (MSP[I]) based on the scenarios Ω_I, results in the dual problem

$$\max_{\lambda} \quad \text{val}(D_\lambda[I]) \tag{3.16}$$

whereby the dual function $\lambda \mapsto \text{val}(D_\lambda[I])$ is defined by the Lagrangian problem

$$\begin{array}{lll} \min\limits_{x} & L_I(x,\xi;\lambda) & \\ \text{s.t.} & g_{t,i}(x_{1,i},\dots,x_{t,i},\zeta_{t,i}) \le 0 & i\in I, \quad t=1,\dots,T \\ & x_{t,i} \in X_t(\zeta_{t,i}) & i\in I, \quad t=1,\dots,T \end{array} \tag{$D_\lambda[I]$}$$

with Lagrange function

$$L_I(x,\xi;\lambda) := \sum_{i\in I} p_i \left(f(x_i,\xi_i) + \sum_{t=1}^{T} x_{i,t}\left(\lambda_{i,t} - \sum_{\omega_{i'}\in E_{t,j(i,t)}} \bar{p}_{i'}\lambda_{i',t}\right)\right),$$

$\lambda_{i,t} \in \mathbb{R}$ and $j(i,t) := j$ such that $\omega_i \in E_{t,j}$ (recall the nonanticipativity constraints in the discrete case from section 3.2). Then the partial Lagrangian subproblems read

$$\begin{array}{lll} \min\limits_{x_i} & f(x_i,\xi_i) + \sum\limits_{t=1}^{T} x_{i,t}\left(\lambda_{i,t} - \sum\limits_{\omega_{i'}\in E_{t,j(i,t)}} \bar{p}_{i'}\lambda_{i',t}\right) & \\ \text{s.t.} & g_t(x_{i,1},\dots,x_{i,t},\zeta_{i,t}) \le 0 \quad t=1,\dots,T & \\ & x_{i,t} \in X_t(\zeta_{i,t}) \quad t=1,\dots,T & \end{array} \quad i \in I. \tag{$D_{\lambda,i}$}$$

The Stochastic Scenario Decomposition Algorithm by Higle, Rayco and Sen

For the illustration of a possible application of scenario decomposition, we will give here a short description of the Stochastic Scenario Decomposition Algorithm proposed in [16]. It is based on the generation of cutting planes to develop a piecewise linear concave approximation of the value of the dual function $\lambda \mapsto \mathrm{val}\,(D_\lambda\,[I])$. Furthermore, the set of scenarios I and probabilities p_i, $i \in I$, is replaced by an empirical distribution that allows to start with a small number of scenarios in the beginning and addition of new scenarios (or updating the probability of the existing ones) during the process of approximating the dual function. To avoid unboundness of the solution set of the dual problem, the set of valid dual variables is restricted to those which satisfy the "zero-expectation-constraints" $\mathbb{E}\left[\lambda_t\,|\mathcal{F}_t\right] \equiv 0$, $t = 1, \ldots, T$, i.e. the dual problem (3.16) is replaced by

$$\begin{array}{ll} \max\limits_{\lambda} & \mathrm{val}\,(D_\lambda\,[I]) \\ \text{s.t.} & \sum\limits_{i' \in I :\, \omega_{i',t} \in E_{t,j}} \bar{p}_{i'} \lambda'_{i',t} = 0 \quad j \in J_t, \quad t = 1, \ldots, T \end{array}. \tag{3.17}$$

Observe that the additional constraints $\sum_{i' \in I :\, \omega_{i',t} \in E_{t,j}} \bar{p}_{i'} \lambda'_{i',t} = 0$, $j \in J_t$, $t = 1, \ldots, T$, does not influence val (3.17), since

$$\lambda_{i,t} - \sum_{\omega_{i'} \in E_{t,j(i,t)}} \bar{p}_{i'} \lambda_{i',t} = (\lambda_{i,t} - \kappa) - \sum_{\omega_{i'} \in E_{t,j(i,t)}} \bar{p}_{i'} \left(\lambda_{i',t} - \kappa\right)$$

in $L_I\,(x, \xi; \lambda)$, so that if $\sum_{i' \in I :\, \omega_{i',t} \in E_{t,j}} \bar{p}_{i'} \lambda'_{i',t} = \kappa$ for some $j \in J_t$ and $\kappa \in \mathbb{R}$, then with $\tilde{\lambda}_{i',t} := \lambda_{i',t} - \kappa$, $i' \in I$ s.t. $\omega_{i',t} \in E_{t,j}$, we obtain $\mathrm{val}\left(D_{\tilde{\lambda}}\,[I]\right) = \mathrm{val}\,(D_\lambda\,[I])$ and satisfy $\sum_{i' \in I :\, \omega_{i',t} \in E_{t,j}} \bar{p}_{i'} \tilde{\lambda}'_{i',t} = 0$ additionally.

We continue to describe one iteration of the algorithm. Let I be the current set of scenarios and p_i, $i \in I$, the corresponding probabilities. Describe the current set of cuts $\mathcal{C}$ by tuples $\left(\alpha_i, (\beta_{i,t})_{t=1}^T\right)_{i \in I} \in \times_{i \in I} \left(\mathbb{R} \times \mathbb{R}^r\right)$. Let λ be the last dual estimate and $\bar{\lambda}$ be an incumbent solution.

Algorithm 3.10 (Stochastic Scenario Decomposition [16]).

Compute:

Obtain solutions x_i, $i \in I$, of the partial Lagrangian problems $(D_{\lambda,i})$, $i \in I$.
Update the Master Problem:
  Generate a new cut by $\beta_{i,t} := p_i x_{i,t}^T$, $t = 1, \ldots, T$, and $\alpha_i := p_i f\,(x_i, \xi_i)$, $i \in I$.
  for all $i \in I$ **do** {Update all previously generated cuts}
    if i was previously added to I **then**
      for all $(\hat{\alpha}_j, (\hat{\beta}_{j,t})_{t=1}^T)_{j \in I \setminus \{i\}} \in \mathcal{C}$ **do**
        if $(\hat{\alpha}_j, (\hat{\beta}_{j,t})_{t=1}^T)_{j \in I \setminus \{i\}}$ was generated by using the current incumbent solution **then**
          obtain a solution point $\bar{x}_i$ of the corresponding subproblem $\left(D_{\bar{\lambda},i}\right)$
          $\hat{\alpha}_i := p_i f\,(\bar{x}_i, \xi_i)$, $\hat{\beta}_{i,t} := p_i \bar{x}_{i,t}^T$, $t = 1, \ldots, T$
        else
          $\hat{\alpha}_i := \alpha_i$, $\hat{\beta}_{i,t} := \beta_{i,t}$, $t = 1, \ldots, T$
        end if
      end for
    else {i was known so far}
      update all cuts in $\mathcal{C}$ to the current probability distribution $(p_i)_{i \in I}$.
    end if
  end for
  Test the incumbent solution:
    Let $\nu\,(\lambda) := \min_{(\alpha_i, (\beta_{i,t})) \in \mathcal{C}} \sum_{i \in I} \sum_{t=1}^T \alpha_{i,t} - \beta_{i,t} \lambda_{i,t}$.
    if the new cut improved the distance $\nu\,(\lambda) - \nu\left(\bar{\lambda}\right)$ by a sufficient amount **then**

make λ be the new incumbent solution.
end if
Obtain a new dual estimate by solving the Master Problem ($\sigma > 0$ is a given parameter)

$$\begin{array}{ll} \max\limits_{\lambda,\nu} & \nu - \frac{\sigma}{2} \left\|\lambda - \bar{\lambda}\right\| \\ \text{s.t.} & \sum\limits_{i\in I} \sum\limits_{t=1}^{T} \alpha_{i,t} - \beta_{i,t}\lambda_{i,t} \geq \nu \quad \left(\alpha_i, (\beta_{i,t})_{t=1}^{T}\right)_{i\in I} \in \mathcal{C} \\ & \sum\limits_{i'\in I:\, \omega_{i',t}\in E_{t,j}} \bar{p}_{i'}\lambda_{i',t} = 0 \quad j \in J_t, \quad t = 1,\ldots,T. \end{array}$$

Update the Empirical Distribution:
Generate a scenario randomly and independent of all previously generated scenarios and add it to I. Set the corresponding dual multiplier to 0 if it was unknown so far. Update p_i, $\bar{p}_i$, $i \in I$.
Project the incumbent solution $\bar{\lambda}$ to ensure that it satisfies the zero-expectation-constraints in the master program.

The following result on the asymptotic convergence of the Stochastic Scenario Decomposition algorithm has been shown ([16, Theorem 6]):

Theorem. *Assume that the primal problem* (MSP[I]) *is feasible and has relatively complete recourse (that is, the dual of* (MSP[I]) *is bounded for all feasible decisions* $(x_{i,1},\ldots,x_{i,t})$, $t = 1,\ldots,T$*). If* $\text{feas}\,(D_{\lambda,i})$ *is compact for all* $i \in I$*,* I *finite, and the sequence of dual estimates* λ *is bounded, then with probability one, there exists an accumulation point of the sequence of incumbent solutions* $\bar{\lambda}$ *that is an optimal solution to* (3.17)*.*

From a computational point of view, observe that in each iteration of the algorithm a new set of variables λ_i (if i is a "new" scenario) and a new cutting plane is introduced into the master problem. This makes the master problem expand significantly throughout the iteration process. Therefore, a procedure that aggregates variables based on the similarity of the data to which they are subjected in the master problem is proposed in [16]. The aggregation is varying from one iteration to the next, especially when the incumbent solution changes. It is done by creating a clustering vector c_j for each component j of $\lambda - \bar{\lambda}$. The components of c_j are

- the value of the corresponding component of the incumbent solution, $\bar{\lambda}_j$
- the variable's coefficient in the cut derived at the incumbent solution
- the variable's coefficient in the most recently derived cut
- the variable's aggregated coefficient in all other cutting planes present in the master problem

Then the smallest box which contains all vectors c_j is partitioned and all components of $\lambda - \bar{\lambda}$ whose clustering vectors are contained within the same element of the partition are aggregated within the master problem (by averaging). An estimate of the relative error due to this aggregation scheme is giving in [16] so a partition can be refined if the error estimate exceeds a given tolerance.

3.4.2 Nodal decomposition

Consider the multistage optimization problem (MSP) and assume the the objective function f and time-coupling constraints g_t separate into a sum of functions, each one depending only on one timestage,

$$\begin{aligned} f(x,\xi) &= \sum_{t=1}^{T} f_t(x_t,\xi), \\ g_t(x_1,\dots,x_t,\zeta_t) &= \sum_{\tau=1}^{t} g_{t,\tau}(x_\tau,\zeta_t), \end{aligned}$$

with $f_t : \mathbb{R}^{r_t} \times \mathbb{R}^s \to \mathbb{R}$ and $g_{t,\tau} : \mathbb{R}^{r_\tau} \times \mathbb{R}^{s_t} \to \mathbb{R}^{d_t}$, $\tau = 1,\dots,t$, $t = 1,\dots,T$. Let further $\lambda_t \in L^1\left(\Omega,\mathcal{F},\mathbb{P};\mathbb{R}^{d_t}\right)$, $t = 1,\dots,T$, denote L^1-bounded $\mathcal{F}$-measurable functions. Then the nodal decomposition of (MSP) is defined by

$$\begin{array}{lll} \max\limits_{\lambda_t:\,\lambda_t\geq 0\ \mathbb{P}\text{-a.s.}} \ \min\limits_{x} & L_{\text{nod}}(x,\xi;\lambda) & \\ \text{s.t.} & x_t \in X_t(\zeta_t) \quad \mathbb{P}\text{-a.s.} & t=1,\dots,T \\ & x_t = \mathbb{E}\left[x_t \,|\mathcal{F}_t\right] & t=1,\dots,T \end{array}$$

where

$$L_{\text{nod}}(x,\xi;\lambda) := \sum_{t=1}^{T} \mathbb{E}\left[f_t(x_t,\xi) + \sum_{\tau=1}^{t} \left\langle \lambda_\tau, g_{t,\tau}(x_\tau,\zeta_t)\right\rangle\right].$$

Thus we obtain the partial Lagrangian subproblems

$$\begin{array}{lll} \min\limits_{x_t} & \mathbb{E}\left[f_t(x_t,\xi) + \left\langle \lambda_t, \sum_{t'=1}^{T} g_{t',t}(x_t,\zeta_{t'})\right\rangle\right] & \\ \text{s.t.} & x_t \in X_t(\zeta_t) \quad \mathbb{P}\text{-a.s.} & t=1,\dots,T. \\ & x_t = \mathbb{E}\left[x_t \,|\mathcal{F}_t\right] & \end{array}$$

3.4.3 Geographical decomposition

Assume now, problem (MSP) has a block-separable structure in the form that there exists partitions $\{I_{t,1},\dots,I_{t,p}\}$ of $\{1,\dots,r_t\}$, $t = 1,\dots,T$, such that

$$\begin{aligned} f(x,\xi) &= \sum_{k=1}^{p} f_k\left(x_{I_k},\xi\right), \\ g_t(x_1,\dots,x_t,\zeta_t) &= \begin{pmatrix} \sum_{k=1}^{p} g_{t,k}\left(x_{I_{1,k}},\dots,x_{I_{t,k}},\zeta_t\right) \\ h_{t,1}\left(x_{I_{1,k}},\dots,x_{I_{t,k}},\zeta_t\right) \\ \vdots \\ h_{t,k}\left(x_{I_{1,k}},\dots,x_{I_{t,k}},\zeta_t\right) \end{pmatrix} \\ X_t(\zeta_t) &= X_{t,1}(\zeta_t) \times \dots \times X_{t,p}(\zeta_t), \end{aligned}$$

$I_k := \bigcup_{t=1}^{T} I_{t,k}$, $x_{I_k} := \left(x_{I_{1,k}},\dots,x_{I_{T,k}}\right)$, $f_k : \mathbb{R}^{|I_k|} \times \mathbb{R}^s \to \mathbb{R}$, $g_{t,k} : \mathbb{R}^{\sum_{\tau=1}^{t}|I_{\tau,k}|} \times \mathbb{R}^{\sum_{\tau=1}^{t} s_\tau} \to \mathbb{R}^{d_{t,0}}$, $h_{t,k} : \mathbb{R}^{\sum_{\tau=1}^{t}|I_{\tau,k}|} \times \mathbb{R}^{\sum_{\tau=1}^{t} s_\tau} \to \mathbb{R}^{d_{t,k}}$ and $X_{t,k} : \mathbb{R}^{\sum_{\tau=1}^{t} s_\tau} \rightrightarrows \mathbb{R}^{|I_{t,k}|}$, $k = 1,\dots,p$, $\sum_{k=0}^{p} d_{t,k} = d_t$, $t = 1,\dots,T$. When separating g_t we distinct here between the sum of functions $g_{t,k}$ over k and functions $h_{t,k}$, $k = 1,\dots,p$, to allow time-coupling constraints inside one block.

Introduce again dual parameters $\lambda_t \in L^1\left(\Omega,\mathcal{F},\mathbb{P};\mathbb{R}^{d_{t,0}}\right)$, $t = 1,\dots,T$. Then the geographical decomposition is defined by dualizing the $\sum_k g_{t,k}$-constraints,

$$\begin{array}{lll} \max\limits_{\lambda=(\lambda_t)_{t=1}^{T}} & \operatorname{val}(D_\lambda) & \\ \text{s.t.} & \lambda_t \geq 0 \quad \mathbb{P}\text{-a.s.} & t=1,\dots,T \end{array}$$

where the Lagrangian problem (D_λ) is

$$\begin{array}{llllll} \min\limits_{x} & L(x,\xi;\lambda) & & & & \\ & x_{I_{t,k}} \in X_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & k=1,\dots,p & t=1,\dots,T & \\ \text{s.t.} & h_{t,k}\left(x_{I_{1,k}},\dots,x_{I_{t,k}},\zeta_t\right) \le 0 & \mathbb{P}\text{-a.s.} & k=1,\dots,p & t=1,\dots,T & \\ & x_{I_{t,k}} = \mathbb{E}\left[x_{I_{t,k}} | \mathcal{F}_t\right] & & k=1,\dots,p & t=1,\dots,T & \end{array} \tag{D_λ}$$

with separable Lagrange function

$$\begin{aligned} L(x,\xi;\lambda) &:= \sum_{k=1}^{p} L_k\left(x_{I_k},\xi;\lambda\right), \\ L_k\left(x_{I_k},\xi;\lambda\right) &:= \mathbb{E}\left[f_k\left(x_{I_k},\xi\right) + \sum_{t=1}^{T} \lambda_t^T g_{t,k}\left(x_{I_{1,k}},\dots,x_{I_{t,k}},\zeta_t\right)\right], \qquad k=1,\dots,p, \end{aligned}$$

so that the Lagrangian problem (D_λ) decomposes into the partial Lagrangian problems

$$\begin{array}{llll} \min\limits_{x_{I_k}} & L_k\left(x_{I_k},\xi;\lambda\right) & & \\ & x_{I_{t,k}} \in X_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & t=1,\dots,T \\ \text{s.t.} & h_{t,k}\left(x_{I_{1,k}},\dots,x_{I_{t,k}},\zeta_t\right) \le 0 & \mathbb{P}\text{-a.s.} & t=1,\dots,T \\ & x_{I_{t,k}} = \mathbb{E}\left[x_{I_{t,k}} | \mathcal{F}_t\right] & & t=1,\dots,T \end{array} \tag{$D_{\lambda,k}$}$$

For the aggregated formulation (MSP[$\mathcal{N}$]) we define similarly the Lagrange function

$$\begin{aligned} L_{\mathcal{N}}(x,\xi;\lambda) &:= \sum_{k=1}^{p} L_{k,\mathcal{N}}\left(x_{I_k},\xi;\lambda\right), \\ L_{k,\mathcal{N}}\left(x_{I_k},\xi;\lambda\right) &:= \sum_{N\in\mathcal{N}} \left(\left(\sum_{n\in N} \pi_n f_k\left(x_{I_k}^{p(N)},\zeta^n\right)\right) + \pi_N \sum_{t=1}^{T} \lambda_{n_t(N)}^T g_{t,k}\left(x_{I_k}^{p(n_t(N))},\zeta^{n_t(N)}\right)\right) \end{aligned}$$

with dual parameters $(\lambda_n)_{n\in\bar{N}|_{\mathcal{N}}}$ to obtain the partial Lagrangian problems

$$\begin{array}{lll} \min\limits_{x_{I_k}} & L_{k,\mathcal{N}}\left(x_{I_k},\xi;\lambda\right) & \\ \text{s.t.} & x_{I_{t(n),k}}^{n} \in X_{t(n),k}(\zeta^n) & n\in\bar{N}|_{\mathcal{N}} \\ & h_{t(n),k}\left(x_{I_k}^{p(n)},\zeta^n\right) \le 0 & n\in\bar{N}|_{\mathcal{N}} \end{array} \tag{$D_{\lambda,k}[\mathcal{N}]$}$$

and the dual problem

$$\max_{\lambda\ge 0} \quad \sum_{k=1}^{p} \operatorname{val}\left(D_{\lambda,k}[\mathcal{N}]\right) \tag{$D[\mathcal{N}]$}$$

For a solution point $\hat{x}_{I_k}$ of $(D_{\lambda,k}[\mathcal{N}])$ $L_{k,\mathcal{N}}(\cdot;\lambda) \ge L_{k,\mathcal{N}}(\hat{x}_{I_k};\lambda)$ is a valid Lagrangian cut, which can be added to $\mathcal{C}_{\mathcal{N}}$.

Refinement operations

As already pointed out for the abstract formulation, it is desirable to inherit a cut from $\mathcal{C}_{\mathcal{N}}$ to $\mathcal{C}_{\mathcal{N}'}$ for a refinement $\mathcal{N}' \preceq \mathcal{N}$. This can be achieved by "transforming" $\hat{x}_{I_k}$ to a solution point of $(D_{\lambda',k}[\mathcal{N}'])$ where $\lambda'^n := \lambda^{\mathrm{agg}(n)}$ for $n\in\bar{N}|_{\mathcal{N}'}$ and $\mathrm{agg} : \bar{N} \to \bar{N}|_{\mathcal{N}}$ is the mapping that expresses the aggregation $\mathcal{N}$ (recall that $\bar{N}|_{\mathcal{N}'} \subseteq \bar{N}|_{\mathcal{N}}$). Hence, let the operator $E_k : X_{k,\mathcal{N}} \to X_{k,\mathcal{N}'}$ be defined by

$$E_k\left(\hat{x}_{I_k}\right) := \operatorname{argmin}\left(\left(D_{\lambda',k}[\mathcal{N}']\right);\hat{x}'_{I_k}\right)$$

whereby $\hat{x}'^{n}_{I_k} := \hat{x}^{\mathrm{agg}(n)}_{I_k}$, $n \in \bar{N}|_{\mathcal{N}'}$, denotes the starting point for solving $(D_{\lambda',k}[\mathcal{N}'])$. When the problems $(D_{\lambda,k}[\mathcal{N}])$ and $(D_{\lambda',k}[\mathcal{N}'])$ does not differ too much, the point $\hat{x}'_{I_k}$ might be already close to a solution point of $(D_{\lambda',k}[\mathcal{N}'])$ so that solving $(D_{\lambda',k}[\mathcal{N}'])$ does not become too expensive. The updated Lagrangian cut, that can be added to $\mathcal{C}_{\mathcal{N}'}$, is $L_{k,\mathcal{N}'}(\cdot_{I_k}, \xi; \lambda') \geq L_{k,\mathcal{N}'}(E_k(\hat{x}_{I_k}), \xi; \lambda')$.

Similar, updating of the extreme points $Z_{k,\mathcal{N}}$ can be achieved by $Z_{k,\mathcal{N}'} := E_k(Z_{k,\mathcal{N}})$, but since we do not depend on optimality for the extreme points $Z_{k,\mathcal{N}}$ the solution process of $(D_{\lambda',k}[\mathcal{N}'])$, started from a point from $Z_{k,\mathcal{N}}$, can be stopped as soon as a feasible point is reached.

3.5 Adaptive scenario reduction algorithm

We are now almost ready to formulate the adaptive discretization algorithm 2.2 for multistage stochastic programs. The last missing ingredients are an adapted formulation of formula (2.1) and the formulation of problem (Proj$[\mathcal{N},\mathcal{N}']$) for MSPs.

The function of (Proj$[\mathcal{N},\mathcal{N}']$) is to minimize the distance between a decision process on $\bar{N}|_{\mathcal{N}'}$ with a corresponding one on $\bar{N}|_{\mathcal{N}}$ if $\mathrm{val}(\mathrm{CMSP}[\mathcal{N}]) - \mathrm{val}(\mathrm{CMSP}[\mathcal{N}']) > \sigma_P |\mathrm{val}(\mathrm{CMSP}[\mathcal{N}'])|$. Hence, we define

$$\begin{array}{lll}
\min\limits_{x} & \displaystyle\sum_{N\in\mathcal{N}} \pi_N \max_{t=1,\dots,T} \max_{N'\in\mathcal{N}'|_N} \left\| x^{n_t(N')} - x^{n_t(N)} \right\|_\infty & \\
\text{s.t.} & \check{g}_{t(n)}\left(x^{p(n)}, \zeta^n\right) \leq 0 & n \in \bar{N}|_{\mathcal{N}'} \\
& x^n \in \check{X}_{t(n)}(\zeta^n) & n \in \bar{N}|_{\mathcal{N}'} \\
& c(x) \leq 0 & c \in \mathcal{C}_{\mathcal{N}'} \\
& \displaystyle\sum_{N'\in\mathcal{N}'} \sum_{n\in N'} \pi_n \check{f}\left(x^{p(N')}, \zeta^n\right) \leq \mathrm{val}(\mathrm{CMSP}[\mathcal{N}']) + \sigma_P \cdot |\mathrm{val}(\mathrm{CMSP}[\mathcal{N}'])| &
\end{array} \tag{ProjMSP$[\mathcal{N},\mathcal{N}']$}$$

to serve as formulation of (Proj$[\mathcal{N},\mathcal{N}']$) for MSPs.

In the case $\mathrm{val}(C[\mathcal{N}']) - \mathrm{val}(C[\mathcal{N}]) > \sigma_P |\mathrm{val}(C[\mathcal{N}])|$, algorithm 2.2 selected a set $\mathcal{M} \subseteq \mathcal{N}$ such that the aggregations $M \in \mathcal{M}$ have an high (positive) share in the difference $\mathrm{val}(C[\mathcal{N}']) - \mathrm{val}(C[\mathcal{N}])$. For MSPs we can adapt formula (2.1) by

$$\mathcal{M} := \operatorname{argmax}\left\{ \sum_{M\in\mathcal{M}} \sum_{M'\in\mathcal{N}'|_M} \pi_{M'} \sum_{n\in M'} \left(\check{f}\left(x'^{p(M')}, \zeta^n\right) - \check{f}\left(x^{p(M)}, \zeta^n\right)\right) \,\middle|\, \mathcal{M} \subseteq \mathcal{N}, |\mathcal{M}| \leq \bar{m} \right\} \tag{3.18}$$

for $x \in \mathrm{Argmin}(\mathrm{CMSP}[\mathcal{N}])$ and $x' \in \mathrm{Argmin}(\mathrm{CMSP}[\mathcal{N}'])$. Since $\mathrm{val}(\mathrm{CMSP}[\mathcal{N}']) > \mathrm{val}(\mathrm{CMSP}[\mathcal{N}])$, $\mathcal{M}$ will be nonempty.

We can now state algorithm 2.2 for multistage stochastic programs. The parameter σ_P controls how far $\frac{\mathrm{val}(\mathrm{CMSP}[\mathcal{N}])}{\mathrm{val}(\mathrm{CMSP}[\mathcal{N}'])}$ is allowed to deviate from 1. σ_D controls how close $\bar{N}|_{\mathcal{N}'}$ and $\bar{N}$ have to be in the end, relative to the distance between the initial aggregation and $\bar{N}$. The parameter $\bar{m}$ gives an upper bound on the size of $\mathcal{M}$ when applying definition (3.18) or (2.2) and s controls roughly the number of new scenarios, which are added to $\mathcal{N}'$ during a refinement.

Algorithm 3.11 (Adaptive scenario reduction).
Input: Initial scenario tree $\bar{N}$, Parameters $\sigma_P \geq 0$, $\sigma_D \in [0,1]$, $\bar{m} \in \mathbb{N}$, $s \in \mathbb{N}$, $s \geq \bar{m}$
Compute:

1: Compute an initial aggregation $\mathcal{N} := \{\bar{N}_T\}$ by selection of one scenario by forward selection (cf. section 3.3.2). Let $\mathcal{D}_0 := \zeta_c\left(\mathbb{P}, \delta_{\omega_{i(\bar{N}_T)}}\right) = p_{i(\bar{N}_T)} \min_{n\in\bar{N}_T\setminus\{n_T(\bar{N}_T)\}} c\left(\zeta^{n_T(\bar{N}_T)}, \zeta^n\right)$ be the distance between the initial scenario tree and the one consisting of the selected scenario only.

Refine $\mathcal{N}$ to an aggregation $\mathcal{N}'$ by algorithm 3.6 with $s_{\bar{N}_T} := s$.
Initialize $Z_{k,\mathcal{N}}$, $\mathcal{C}_{\mathcal{N}} := \emptyset$.
repeat
  Update $\mathcal{C}_{\mathcal{N}}$ and compute the cuts $\mathcal{C}_{\mathcal{N}'}$ from $\mathcal{C}_{\mathcal{N}}$.
  Solve (CMSP[$\mathcal{N}$]) and (CMSP [$\mathcal{N}'$]).
  if $\mathrm{val}\,(\mathrm{CMSP}\,[\mathcal{N}']) - \mathrm{val}\,(\mathrm{CMSP}[\mathcal{N}]) > \sigma_P\,|\mathrm{val}\,(\mathrm{CMSP}[\mathcal{N}])|$ **then**
    Compute $\mathcal{M}$ according to (3.18) and set $s_M := [s \cdot \rho_M]$ where ρ_M is the fraction from formula (2.3), $M \in \mathcal{M}$.
    Compute $\mathcal{N}(\mathcal{M})$ and $\mathcal{N}'(\mathcal{M})$ by algorithm 3.6.
    $Z_{k,\mathcal{N}(\mathcal{M})} := E_k\,(Z_{k,\mathcal{N}})$, $k = 1, \dots, p$.
    $\mathcal{N} := \mathcal{N}(\mathcal{M})$, $\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
  else if $\mathrm{val}\,(\mathrm{CMSP}[\mathcal{N}]) - \mathrm{val}\,(\mathrm{CMSP}\,[\mathcal{N}']) > \sigma_P\,|\mathrm{val}\,(\mathrm{CMSP}\,[\mathcal{N}'])|$ **then**
    solve (ProjMSP[$\mathcal{N}, \mathcal{N}'$])
    Compute $\mathcal{M}$ according to (2.2) and set $s_M := [s \cdot \rho_M]$ where ρ_M is the fraction from formula (2.3), $M \in \mathcal{M}$.
    Compute $\mathcal{N}(\mathcal{M})$ and $\mathcal{N}'(\mathcal{M})$ by algorithm 3.6.
    $Z_{k,\mathcal{N}(\mathcal{M})} := E_k\,(Z_{k,\mathcal{N}})$, $k = 1, \dots, p$.
    $\mathcal{N} := \mathcal{N}(\mathcal{M})$, $\mathcal{N}' := \mathcal{N}'(\mathcal{M})$
  else if $\tilde{\mathcal{D}}\,(\mathcal{N}') > \sigma_D \mathcal{D}_0$ **then**
    Let $\mathcal{M} := \mathrm{argmax}\left\{\sum_{M\in\mathcal{M}} \tilde{\mathcal{D}}\,(M) \,\middle|\, \mathcal{M} \subseteq \mathcal{N},\ |\mathcal{M}| \le \bar{m},\ \tilde{\mathcal{D}}\,(M) > 0, M \in \mathcal{M}\right\}$
    Compute $\mathcal{N}'(\mathcal{M})$ by algorithm 3.6, $s_M := s$ for $M \in \mathcal{M}$
    $\mathcal{N}' := \mathcal{N}'(\mathcal{M})$ ($\mathcal{N}$ remains unchanged.)
  end if
until $\mathcal{M} = \emptyset$
Output: Reduced scenario tree $\bar{N}\,|_{\mathcal{N}}$.

3.6 Mixed-integer linear multistage stochastic programs

The adaptive scenario reduction algorithm 3.11 has been implemented for linear multistage stochastic programs and will hopefully be extended to the case that involves integer variables. Therefore, we continue to introduce the notation for mixed integer linear multistage stochastic programs. Then we mention an alternative method for defining the aggregated problem. As an example for a linear MSP, we describe a model for the power management in a hydro-thermal system under uncertain load. Finally, we close this section by introducing a cutting-plane algorithm for the solution of linear MSPs which is based on relaxation of the time-coupling constraints.

The multistage stochastic program (MSP) is linear, if the functions f, g_t, $t = 1, \dots, T$, are linear in x and $X_t\,(\zeta_t)$ is polyhedral in $\mathbb{R}^{r_t}$ for all $\zeta_t \in \zeta_t\,(\Omega)$ and $t = 1, \dots, T$. Thus, keeping the geographical decomposition of (MSP) in mind, let us assume that

$$
\begin{aligned}
f\,(x,\xi) &= \sum_{k=1}^{p} \langle c_k\,(\xi)\,, x_{I_k}\rangle\,,\\
g_t\,(x_1,\dots,x_t;\zeta_t) &= \begin{pmatrix} \left(\displaystyle\sum_{\tau=1}^{t}\sum_{k=1}^{p} A_{t,\tau,k}\,(\zeta_t)\,x_{I_{\tau,k}}\right) - b_t\,(\zeta_t)\\ \displaystyle\sum_{\tau=1}^{t} \tilde{A}_{t,\tau,1}\,(\zeta_t)\,x_{I_{\tau,1}} - \tilde{b}_{t,1}\,(\zeta_t)\\ \vdots\\ \displaystyle\sum_{\tau=1}^{t} \tilde{A}_{t,\tau,p}\,(\zeta_t)\,x_{I_{\tau,p}} - \tilde{b}_{t,p}\,(\zeta_t) \end{pmatrix},\\
X_t\,(\zeta_t) &= X_{t,1}\,(\zeta_t) \times \cdots \times X_{t,k}\,(\zeta_t)
\end{aligned}
$$

where $c_k : \mathbb{R}^{\sum_{\tau=1}^{T} s_\tau} \to \mathbb{R}^{|I_k|}$, $A_{t,\tau,k} : \mathbb{R}^{\sum_{\tau'=1}^{t} s_{\tau'}} \to \mathbb{R}^{d_{t,0} \times |I_{\tau,k}|}$, $\tau = 1, \dots, t$, $b_t : \mathbb{R}^{\sum_{\tau=1}^{t} s_\tau} \to \mathbb{R}^{d_{t,0}}$, $\tilde{A}_{t,\tau,k} : \mathbb{R}^{\sum_{\tau'=1}^{t} s_{\tau'}} \to \mathbb{R}^{d_{t,k} \times |I_{\tau,k}|}$, $\tau = 1, \dots, t$, $\tilde{b}_{t,k} : \mathbb{R}^{\sum_{\tau'=1}^{t} s_{\tau'}} \to \mathbb{R}^{d_{t,k}}$ and $X_{t,k}(\zeta_t) \subseteq \mathbb{R}^{|I_{t,k}|}$ is polyhedral for all $\zeta_t \in \zeta_t(\Omega)$ and $t = 1, \dots, T$.

A mixed-integer linear (MSP) reads now

$$\begin{array}{lllll}
\min\limits_{x} & \mathbb{E}\left[\sum\limits_{k=1}^{p} \langle c_k(\xi), x_{I_k} \rangle\right] & & & \\
\text{s.t.} & \sum\limits_{\tau=1}^{t}\sum\limits_{k=1}^{p} A_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}} \le b_t(\zeta_t) & \mathbb{P}\text{-a.s.} & & t = 1, \dots, T \\
 & \sum\limits_{\tau=1}^{t} \tilde{A}_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}} \le \tilde{b}_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & k = 1, \dots, p & t = 1, \dots, T \\
 & x_{I_{t,k}} \in X_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & k = 1, \dots, p & t = 1, \dots, T \\
 & x_{B_{t,k}} \in \mathbb{Z}^{|B_{t,k}|} & \mathbb{P}\text{-a.s.} & k = 1, \dots, p & t = 1, \dots, T \\
 & x_{I_{t,k}} = \mathbb{E}\left[x_{I_{t,k}} \,|\mathcal{F}_t\right] & & k = 1, \dots, p & t = 1, \dots, T
\end{array} \tag{MILMSP}$$

where $B_{t,k} \subseteq I_{t,k}$ denotes the set of variables with integrality-constraints, $k = 1, \dots, p$, $t = 1, \dots, T$.

The Lagrange function, obtained by dualizing the coupling-constraints $\sum\sum A(\zeta)\, x \le b(\zeta)$, is

$$\begin{aligned}
L(x, \xi; \lambda) &= \sum_{k=1}^{p} L_k(x_{I_k}, \xi; \lambda) - \mathbb{E}\left[\sum_{t=1}^{T} \lambda_t^T b_t(\zeta_t)\right], \\
L_k(x_{I_k}, \xi; \lambda) &= \mathbb{E}\left[\langle c_k(\xi), x_{I_k} \rangle + \sum_{t=1}^{T} \lambda_t^T \sum_{\tau=1}^{t} A_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}}\right],
\end{aligned}$$

where $\lambda_t \in \mathbb{R}^{d_{t,0}}$. So the Lagrangian subproblems are

$$\begin{array}{llll}
\min\limits_{x_{I_k}} & L_k(x_{I_k}, \xi; \lambda) & & \\
\text{s.t.} & \sum\limits_{\tau=1}^{t} \tilde{A}_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}} \le \tilde{b}_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & t = 1, \dots, T \\
 & x_{I_{t,k}} \in X_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & t = 1, \dots, T \\
 & x_{B_{t,k}} \in \mathbb{Z}^{|B_{t,k}|} & \mathbb{P}\text{-a.s.} & t = 1, \dots, T \\
 & x_{I_{t,k}} = \mathbb{E}\left[x_{I_{t,k}} \,|\mathcal{F}_t\right] & & t = 1, \dots, T
\end{array}$$

3.6.1 Discretization, Aggregation

Similar to general MSPs, we obtain the scenario-tree-formulation

$$\begin{array}{llll}
\min\limits_{x} & \sum\limits_{k=1}^{p}\sum\limits_{n \in \bar{N}_T} \pi_n \left\langle c_k(\zeta^n), x_{I_k}^{p(n)} \right\rangle & & \\
\text{s.t.} & \sum\limits_{\tau=1}^{t(n)}\sum\limits_{k=1}^{p} A_{t(n),\tau,k}(\zeta^n)\, x_{I_{\tau,k}}^{n-t(n)+\tau} \le b_{t(n)}(\zeta^n) & & n \in \bar{N} \\
 & \sum\limits_{\tau=1}^{t(n)} \tilde{A}_{t(n),\tau,k}(\zeta^n)\, x_{I_{\tau,k}}^{n-t(n)+\tau} \le \tilde{b}_{t(n),k}(\zeta^n) & k = 1, \dots, p & n \in \bar{N} \\
 & x_{I_{t(n),k}}^{n} \in X_{t(n),k}(\zeta^n) & k = 1, \dots, p & n \in \bar{N} \\
 & x_{B_{t(n),k}}^{n} \in \mathbb{Z}^{|B_{t(n),k}|} & k = 1, \dots, p & n \in \bar{N}
\end{array} \tag{MILMSP$[\bar{N}]$}$$

and aggregated formulation

$$\begin{array}{llll}
\min_{x} & \sum_{k=1}^{p}\sum_{N\in\mathcal{N}}\sum_{n\in N}\pi_n\left\langle c_k\left(\zeta^n\right), x^{p(N)}_{I_k}\right\rangle & & \\
\text{s.t.} & \sum_{\tau=1}^{t(n)}\sum_{k=1}^{p} A_{t(n),\tau,k}\left(\zeta^n\right)x^{n-t(n)+\tau}_{I_{\tau,k}} \le b_{t(n)}\left(\zeta^n\right) & & n\in\bar{N}\mid_{\mathcal{N}} \\
& \sum_{\tau=1}^{t(n)}\tilde{A}_{t(n),\tau,k}\left(\zeta^n\right)x^{n-t(n)+\tau}_{I_{\tau,k}} \le \tilde{b}_{t(n),k}\left(\zeta^n\right) & k=1,\dots,p & n\in\bar{N}\mid_{\mathcal{N}} \\
& x^n_{I_{t(n),k}} \in X_{t(n),k}\left(\zeta^n\right) & k=1,\dots,p & n\in\bar{N}\mid_{\mathcal{N}} \\
& x^n_{B_{t(n),k}} \in \mathbb{Z}^{\left|B_{t(n),k}\right|} & k=1,\dots,p & n\in\bar{N}\mid_{\mathcal{N}}
\end{array} \qquad (\text{MILMSP}[\mathcal{N}])$$

The Lagrange function of (MILMSP[$\mathcal{N}$]) is

$$\begin{aligned}
L_{\mathcal{N}}\left(x,\xi;\lambda\right) &:= \sum_{k=1}^{p} L_{\mathcal{N},k}\left(x_{I_k},\xi;\lambda\right) - \sum_{N\in\mathcal{N}}\pi_N\sum_{t=1}^{T}\lambda^T_{n_t(N)}b_t\left(\zeta^{n_t(N)}\right), \\
L_{\mathcal{N},k}\left(x_{I_k},\xi;\lambda\right) &:= \sum_{N\in\mathcal{N}}\left\langle\sum_{n\in\mathcal{N}}\pi_n c_k\left(\zeta^n\right), x^{p(N)}_{I_k}\right\rangle + \sum_{N\in\mathcal{N}}\pi_N\sum_{t=1}^{T}\lambda^T_{n_t(N)}\sum_{\tau=1}^{t}A_{t,\tau,k}\left(\zeta^{n_t(N)}\right)x^{n_\tau(N)}_{I_{t,k}}
\end{aligned}$$

and the partial Lagrangian subproblems read

$$\begin{array}{llll}
\min_{x_{I_k}} & L_{\mathcal{N},k}\left(x_{I_k},\xi;\lambda\right) & & \\
\text{s.t.} & \sum_{\tau=1}^{t(n)}\tilde{A}_{t(n),\tau,k}\left(\zeta^n\right)x^{n-t(n)+\tau}_{I_{\tau,k}} \le \tilde{b}_{t(n),k}\left(\zeta^n\right) & n\in\bar{N}\mid_{\mathcal{N}} & \\
& x^n_{I_{t(n),k}} \in X_{t(n),k}\left(\zeta^n\right) & n\in\bar{N}\mid_{\mathcal{N}} & k=1,\dots,p. \\
& x^n_{B_{t(n),k}} \in \mathbb{Z}^{\left|B_{t(n),k}\right|} & n\in\bar{N}\mid_{\mathcal{N}} &
\end{array}$$

The convexification of (MILMSP) is defined by omitting the integrality constraints and adding linear cuts. We identify the set of cuts $\mathcal{C}_{\mathcal{N}}$ here by a set of pairs $\left(\{a^n\}_{n\in\bar{N}\mid_{\mathcal{N}}},\bar{a}\right)\in\left(\times_{n\in\bar{N}\mid_{\mathcal{N}}}\mathbb{R}^{r_{t(n)}}\right)\times\mathbb{R}$, such that the cut is defined by $\sum_{n\in\bar{N}\mid_{\mathcal{N}}}\langle a^n,x^n\rangle\le\bar{a}$. Then the convexification of (MILMSP[$\mathcal{N}$]) reads

$$\begin{array}{llll}
\min_{x} & \sum_{k=1}^{p}\sum_{N\in\mathcal{N}}\sum_{n\in N}\pi_n\left\langle c_k\left(\zeta^n\right), x^{p(N)}_{I_k}\right\rangle & & \\
\text{s.t.} & \sum_{\tau=1}^{t(n)}\sum_{k=1}^{p} A_{t(n),\tau,k}\left(\zeta^n\right)x^{n-t(n)+\tau}_{I_{\tau,k}} \le b_{t(n)}\left(\zeta^n\right) & & n\in\bar{N}\mid_{\mathcal{N}} \\
& \sum_{\tau=1}^{t(n)}\tilde{A}_{t(n),\tau,k}\left(\zeta^n\right)x^{n-t(n)+\tau}_{I_{\tau,k}} \le \tilde{b}_{t(n),k}\left(\zeta^n\right) & k=1,\dots,p & n\in\bar{N}\mid_{\mathcal{N}} \\
& x^n_{I_{t(n),k}} \in X_{t(n),k}\left(\zeta^n\right) & k=1,\dots,p & n\in\bar{N}\mid_{\mathcal{N}} \\
& \sum_{n\in\bar{N}\mid_{\mathcal{N}}}\langle a^n,x^n\rangle \le \bar{a} & & \left(\{a^n\}_{n\in\bar{N}\mid_{\mathcal{N}}},\bar{a}\right)\in\mathcal{C}_{\mathcal{N}}
\end{array} \qquad (\text{LMSP}[\mathcal{N}])$$

3.6.2 Aggregation: 2. option

In the works [1] and [8], another method for aggregating linear multistage stochastic programs is used. Instead of selecting a scenario for each $N\in\mathcal{N}$ and consider the MSP induced by $\bar{N}\mid_{\mathcal{N}}$, the authors averaged not only the objective function, as we did, but also the constraints. This allows the computation of lower and upper bounds on val $\left(\mathrm{MSP}\left[\bar{N}\right]\right)$ but also assumes additional requirements on the underlying stochastic process, since the $\mathbb{P}$-a.s.-structure of the constraints does not allow arbitrary averaging a priori. For completeness, we will here introduce the notation of this kind of aggregations shortly:

Assume, the polyhedron $X_{t,k}(\zeta_t)$ is given in the form

$$X_{t,k}(\zeta_t) = \left\{ x_{I_{t,k}} \in \mathbb{R}^{|I_{t,k}|} \,\middle|\, \bar{A}_{t,k}(\zeta_t)\, x_{I_{t,k}} \leq \bar{b}_{t,k}(\zeta_t) \right\}$$

such that (MILMSP) takes the form

$$\begin{array}{lrlll}
\min\limits_{x} & \mathbb{E}\left[\sum\limits_{k=1}^{p} \langle c_k(\xi), x_{I_k}\rangle\right] & & & \\
 & \sum\limits_{\tau=1}^{t}\sum\limits_{k=1}^{p} A_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}} \leq b_t(\zeta_t) & \mathbb{P}\text{-a.s.} & & t=1,\dots,T \\
\text{s.t.} & \sum\limits_{\tau=1}^{t} \tilde{A}_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}} \leq \tilde{b}_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & k=1,\dots,p & t=1,\dots,T \\
 & \bar{A}_{t,k}(\zeta_t)\, x_{I_{t,k}} \leq \bar{b}_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} & k=1,\dots,p & t=1,\dots,T \\
 & x_{B_{t,k}} \in \mathbb{Z}^{|B_{t,k}|} & \mathbb{P}\text{-a.s.} & k=1,\dots,p & t=1,\dots,T \\
 & x_{I_{t,k}} = \mathbb{E}\left[x_{I_{t,k}} \,|\mathcal{F}_t\right] & & k=1,\dots,p & t=1,\dots,T
\end{array}$$

Then another option to define the aggregated problem is

$$\begin{array}{ll}
\min\limits_{x} & \sum\limits_{k=1}^{p}\sum\limits_{N\in\mathcal{N}} \left\langle \sum\limits_{n\in N} \pi_n c_k(\zeta^n), x_{I_k}^{p(N)} \right\rangle \\
 & \sum\limits_{\tau=1}^{t(n)}\sum\limits_{k=1}^{p} \left(\sum\limits_{n'\in \mathrm{agg}^{-1}(n)} \pi_{n'} A_{t(n),\tau,k}\left(\zeta^{n'}\right)\right) x_{I_{\tau,k}}^{n-t(n)+\tau} \leq \sum\limits_{n'\in \mathrm{agg}^{-1}(n)} \pi_{n'} b_{t(n)}\left(\zeta^{n'}\right) \\
 & \hfill n\in \bar{N}\,|_{\mathcal{N}} \\
\text{s.t.} & \sum\limits_{\tau=1}^{t(n)} \left(\sum\limits_{n'\in \mathrm{agg}^{-1}(n)} \pi_{n'} \tilde{A}_{t(n),\tau,k}\left(\zeta^{n'}\right)\right) x_{I_{\tau,k}}^{n-t(n)+\tau} \leq \sum\limits_{n'\in \mathrm{agg}^{-1}(n)} \pi_{n'} \tilde{b}_{t(n)}\left(\zeta^{n'}\right) \\
 & \hfill k=1,\dots,p \quad n\in \bar{N}\,|_{\mathcal{N}} \\
 & \left(\sum\limits_{n'\in \mathrm{agg}^{-1}(n)} \pi_{n'} \bar{A}_{t(n),k}\left(\zeta^{n'}\right)\right) x_{I_{t(n),k}}^{n} \leq \sum\limits_{n'\in \mathrm{agg}^{-1}(n)} \pi_{n'} \bar{b}_{t(n),k}\left(\zeta^{n'}\right) \\
 & \hfill k=1,\dots,p \quad n\in \bar{N}\,|_{\mathcal{N}} \\
 & x_{B_{t(n),k}}^{n} \in \mathbb{Z}^{|B_{t(n),k}|} \\
 & \hfill k=1,\dots,p \quad n\in \bar{N}\,|_{\mathcal{N}}
\end{array}$$

3.6.3 An example: Power Management in a hydro-thermal system under uncertainty

After introducing the scenario reduction algorithm and all it's notation, it's now on the time to give a short example for a linear multistage stochastic program. The numerical results in the next chapter are based on this example.

We consider a power generation system consisting of a collection of coal fired thermal units I and pumped hydro units J. The aim is to find cost-optimal operation levels of the thermal and hydro units for one week under load. The electric load underlies thereby uncertainties.

Fixed parameters for the thermal units are fuel prices $b \geq 0$ and operation ranges $0 \leq \underline{p} \leq \bar{p}$. For the hydro units the parameters are the efficiency of the pumps $0 \leq \eta \leq 1$, the maximal operation levels of the pumps and turbines $\bar{w}, \bar{v} \geq 0$ and the capacity of the reservoirs $\bar{l} \geq 0$. For the beginning and end of the week, the fill levels l_{in} and l_{end} of the reservoirs is given.

The stochastic data consists of the load d and the required reserve r for one week. We search for a process of decisions on the operation levels of the thermal units p, pumps w and turbines v, and the fill levels l in the pump-reservoirs.

The decision process underlies the following constraints:

- conditions on the fill levels at the beginning and end of the week: $l_{j,1} = l_{\text{in},j} - (v_{j,1} - \eta_j w_{j,1})$, $l_{j,T} = l_{\text{end},j}$, $j \in J$
- time-coupling of the fill-levels: $l_{j,t} = l_{j,t-1} - (v_{j,t} - \eta_j w_{j,t})$, $j \in J$, $t = 2, \ldots, T$
- meet of the load: $\sum_{i \in I} p_{i,t} + \sum_{j \in J} (v_{j,t} - w_{j,t}) \geq d_t$, $t = 1, \ldots, T$
- reserve-requirements: $\sum_{i \in I} \bar{p}_i - p_{i,t} \geq r_t$, $t = 1, \ldots, T$
- several bound-constraints: $\underline{p}_t \leq p_{i,t} \leq \bar{p}_t$, $i \in I$, $0 \leq v_{j,t} \leq \bar{v}_j$, $0 \leq w_{j,t} \leq \bar{w}_j$, $0 \leq l_{j,t} \leq \bar{l}_j$, $j \in J$, $t = 1, \ldots, T$

The objective is to minimize the expected fuel costs $\mathbb{E}\left[\sum_{i \in I} \sum_{t=1,\ldots,T} b_i p_{i,t}\right]$.

Thus the complete model takes the form

$$
\begin{array}{lll}
\min\limits_{p,l,v,w} & \mathbb{E}\left[\sum\limits_{i \in I} \sum\limits_{t=1}^{T} b_i p_{i,t}\right] & \\
 & l_{j,1} + (v_{j,1} - \eta_j w_{j,1}) = l_{\text{in},j} & j \in J \\
 & l_{j,T} = l_{\text{end},j} & j \in J \\
 & l_{j,t} - l_{j,t-1} + (v_{j,t} - \eta_j w_{j,t}) = 0 & t = 2, \ldots, T, \quad j \in J \\
 & \sum\limits_{i \in I} p_{i,t} + \sum\limits_{j \in J} (v_{j,t} - w_{j,t}) \geq d_t & t = 1, \ldots, T \\
\text{s.t.} & \sum\limits_{i \in I} \bar{p}_i - p_{i,t} \geq r_t & t = 1, \ldots, T \\
 & \underline{p}_t \leq p_{i,t} \leq \bar{p}_t & t = 1, \ldots, T \quad i \in I \\
 & 0 \leq v_{j,t} \leq \bar{v}_j & t = 1, \ldots, T \quad j \in J \\
 & 0 \leq w_{j,t} \leq \bar{w}_j & t = 1, \ldots, T \quad j \in J \\
 & 0 \leq l_{j,t} \leq \bar{l}_j & t = 1, \ldots, T \quad j \in J
\end{array}
\tag{3.19}
$$

3.6.4 Nested Benders Decomposition

To solve a linear MSP, it is preferable to use an algorithm that exploits it's special structure. One option is the so-called nested Benders decomposition, which decomposes a linear MSP into subproblems, which are attached to the nodes of the scenario tree. The original problem is then solved by traversing the scenario tree and solving these subproblems. A node n passes thereby primal information to its children in the form of a decision vector $x^{p(n)}$, whereby the children pass dual information to it's parent in the form of cuts.

To simplify notation, assume a linear MSP of the form

$$\begin{aligned} \min_{x} \quad & \sum_{t=1}^{T} \mathbb{E}\left[\langle c_t(\zeta_t), x_t\rangle\right] \\ \text{s.t.} \quad & \sum_{\tau=1}^{t} A_{t,\tau}(\zeta_t)\, x_\tau \le b_t(\zeta_t) && \mathbb{P}\text{-a.s.} \quad t=1,\dots,T \\ & x_t \in X_t(\zeta_t) && \mathbb{P}\text{-a.s.} \quad t=1,\dots,T \\ & x_t = \mathbb{E}\left[x_t \,|\mathcal{F}_t\right] && t=1,\dots,T \end{aligned} \tag{3.20}$$

with $c_t : \mathbb{R}^{s_t} \to \mathbb{R}$, $A_{t,\tau} : \mathbb{R}^{\sum_{\tau'=1}^{t} s_{\tau'}} \to \mathbb{R}^{d_t \times r_\tau}$, $\tau = 1,\dots,t$, $b_t : \mathbb{R}^{\sum_{\tau=1}^{t} s_\tau} \to \mathbb{R}^{d_t}$ and $X_t(\zeta_t)$ is polyhedral for all $\zeta_t \in \zeta_t(\Omega)$, $t=1,\dots,T$. Then the scenario-tree formulation can be written as

$$\begin{aligned} \min_{x} \quad & \sum_{n\in\bar{N}} \pi_n \left\langle c_{t(n)}(\zeta^n), x^n\right\rangle \\ \text{s.t.} \quad & \sum_{\tau=1}^{t(n)} A_{t(n),\tau}(\zeta^n)\, x^{n-t(n)+\tau} \le b_{t(n)}(\zeta^n) && n\in\bar{N} \\ & x^n \in X_{t(n)}(\zeta^n) && n \in \bar{N} \end{aligned} \tag{3.21}$$

For a node $n \in \bar{N}$ and decision-vector $x^{p(n_-)}$ for the path from the root up to n_-, the following subproblem is considered:

$$\begin{aligned} \min_{x^n} \quad & \left\langle c_{t(n)}(\zeta^n), x^n\right\rangle + \theta^n \\ \text{s.t.} \quad & Q\left(x^{p(n_-)}, x^n\right) \le \theta^n \\ & A_{t(n),t}(\zeta^n)\, x^n \le b_{t(n)}(\zeta^n) - \sum_{\tau=1}^{t(n)-1} A_{t(n),\tau}(\zeta^n)\, x^{n-t(n)+\tau} \\ & x^n \in X_{t(n)}(\zeta^n) \end{aligned} \tag{3.22}$$

whereby

$$\begin{aligned} Q\left(x^{p(n)}\right) \quad := \quad & \min_{(x^{n_+})_{n_+\in N_+(n)}} \sum_{n_+\in N_+(n)} \pi_{n_+} \left\langle c_{t(n)+1}(\zeta^{n_+}), x^{n_+}\right\rangle \\ \text{s.t.} \quad & A_{t(n_+),t(n_+)}(\zeta^{n_+})\, x^{n_+} \le b_{t(n_+)}(\zeta^{n_+}) - \sum_{\tau=1}^{t(n)} A_{t(n_+),\tau}(\zeta^{n_+})\, x^{n-t(n)+\tau} && n_+ \in N_+(n) \\ & x^{n_+} \in X_{t(n_+)}(\zeta^{n_+}) && n_+ \in N_+(n) \end{aligned} \tag{3.23}$$

Since solving (3.22) is too expensive (evaluating $Q\left(x^{p(n_-)}, x^n\right)$ involves the solution of $|N_+(n)|$ subproblems) it is relaxed by replacing the constraint $Q\left(x^{p(n)}\right) \le \theta^n$ by a set of linear cuts $c(x^n, \theta^n) \le 0$, $c \in \mathcal{C}_n$. Therefore, the relaxed form of (3.22) is

$$\begin{aligned} \min_{x^n} \quad & \left\langle c_{t(n)}(\zeta^n), x^n\right\rangle + \theta^n \\ \text{s.t.} \quad & c(x^n, \theta^n) \le 0 && c \in \mathcal{C}_n \\ & A_{t(n),t}(\zeta^n)\, x^n \le b_{t(n)}(\zeta^n) - \sum_{\tau=1}^{t(n)-1} A_{t(n),\tau}(\zeta^n)\, x^{n-t(n)+\tau} \\ & x^n \in X_{t(n)}(\zeta^n) \end{aligned} \tag{3.24}$$

Such a cut is constructed by solving problem (3.23) or it's dual equivalent for a given decision vector $\hat{x}^{p(n)}$ and using the fact, that Q is convex. If (3.23) is infeasible, it's dual is unbounded, so a feasibility cut can be derived. If otherwise (3.23) is feasible, an optimality cut can be obtained. Or actually, instead of solving (3.23) for $n \in \bar{N}$, we can make use of the separability of (3.23) and solve problem (3.24) for all $n_+ \in N_+(n)$ and construct optimality or feasibility cuts from their solution. We formalize these thoughts in an algorithm, that is called for each node, when it is activated:

Algorithm 3.12 (subroutine of the nested Benders decomposition [20, Chapter 3, Section 5.2]). Let $n \in \bar{N}$.

1: **if** $t(n) > 1$ **then**
2:    Retrieve an approximate solution $x^{p(n_-)}$ from the predecessor nodes.
3: **end if**
4: **if** $t(n) < T$ **then**
5:    Retrieve optimality and feasibility cuts from the successor nodes $N_+(n)$.
6: **end if**
7: Formulate and solve problem (3.24).
8: **if** (3.24) is feasible **then**
9:    Store an optimal solution point x^n.
10:    **if** $t(n) > 1$ **and** the optimal value improved **then**
11:       Construct an optimality cut.
12:    **end if**
13: **else** {(3.24) is infeasible}
14:    **if** $t(n) > 1$ **then**
15:       Construct a feasibility cut.
16:    **else**
17:       **stop**: (3.21) is infeasible.
18:    **end if**
19: **end if**

The order, in which algorithm 3.12 is called for the nodes is determined by the *sequencing protocol*. In [4, 5], all nodes which share a common timestage t are treated at the same time. For those which predecessors have feasible subproblems, algorithm 3.12 can be called and optimality or feasibility cuts can be generated. If some of the subproblems (3.24) were feasible, it's possible to proceed with the next timestage $t + 1$. Otherwise, one has to return to timestage $t - 1$ and resolve the corresponding subproblems there.

The algorithm can be stopped when every subproblem is solved to optimality but does not provide new cuts. Then the complete problem (3.21) has been solved to optimality. The other stopping criteria is line 17 of algorithm 3.12, which detects infeasibility of (3.21).

Chapter 4

Numerical Results

4.1 Software ADoSP

The software package ADoSP (Adaptive Discretization of Stochastic Programs) in it's current state is an implementation of the algorithm 3.11 for linear multistage stochastic programs similar to (MILMSP), i.e. ADoSP is able to handle models of the form

$$\begin{array}{llr}
\min\limits_{x} & \mathbb{E}\left[\sum\limits_{k=1}^{p} \langle c_k(\xi), x_{I_k}\rangle\right] & \\
& \underline{b}_t(\zeta_t) \le \sum\limits_{\tau=1}^{t}\sum\limits_{k=1}^{p} A_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}} \le \bar{b}_t(\zeta_t) & \mathbb{P}\text{-a.s.} \quad t=1,\dots,T \\
\text{s.t.} & \underline{\tilde{b}}_{t,k}(\zeta_t) \le \sum\limits_{\tau=1}^{t} \tilde{A}_{t,\tau,k}(\zeta_t)\, x_{I_{\tau,k}} \le \bar{\tilde{b}}_{t,k}(\zeta_t) & \mathbb{P}\text{-a.s.} \quad k=1,\dots,p \quad t=1,\dots,T \\
& \underline{x} \le x \le \bar{x} & \mathbb{P}\text{-a.s.} \\
& x_{I_{t,k}} = \mathbb{E}\left[x_{I_{t,k}} \,|\mathcal{F}_t\right] & k=1,\dots,p \quad t=1,\dots,T
\end{array} \tag{4.1}$$

where some of the bounds $\underline{b}_t(\cdot)$, $\underline{\tilde{b}}_{t,k}(\cdot)$, $\underline{x}$, $\bar{b}_t(\cdot)$, $\bar{\tilde{b}}_{t,k}(\cdot)$, $\bar{x}$ can also attain infinite values.

The current version of ADoSP consists of ca. 6000 lines of C++ code. Models can be given in GAMS ([11]) format. To solve appearing LP's ADoSP utilizes the COIN-OSI-library ([2]), which themselves contains a CPLEX ([3]) interface.

4.1.1 Input

For the input of the model in ADoSP, a GAMS-interface ([11]) is implemented. Additional information about timestages and blockstructure of the variables must be provided separately. The corresponding information for the constraints is generated automatically from available sparsity information.

Model

Problem (4.1) is formulated as NLP in the variables x and ξ

$$\begin{array}{ll}
\min\limits_{x,\xi} & f(x;\xi) \\
\text{s.t.} & \underline{g} \le g(x;\xi) \le \bar{g} \\
& \underline{x} \le x \le \bar{x}
\end{array}$$

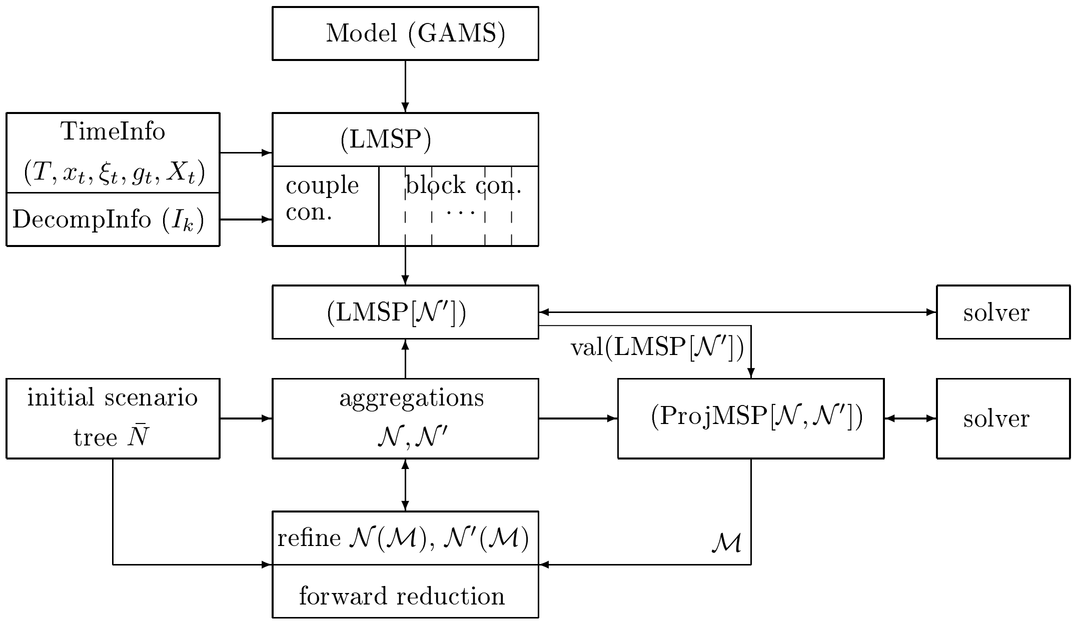


Figure 4.1: Design of the software package ADoSP.

and the following assumptions are made:

- The function $f(x, \xi)$ is given in the form $y = f(x, \xi)$ where y is a free variable. This variable is removed by ADoSP.
- $\nabla_x f(\cdot, \xi)$ and $\nabla_x g(\cdot, \xi)$ is constant, i.e. after substituting ξ by values from a finite set of scenarios, one obtains a linear program.
- The descriptive text for the decision variables x start with “desc”, the descriptive text for the data variables start with “data”. A descriptive text for y can be omitted.

Timestages

A file which gives information about the timestages needs to be provided separately.

The first (nonempty) line contains the number of timestages T. Then information about the variables which belong to each timestage is successively given. A timestage is introduced by the keyword “NEXT” (also for the first timestage). Then the names of the variables in x_t and ξ_t are given.

Constraints are assigned to timestage by checking the variables, which appear $(\exists y : \nabla_{y_i} g(y) \neq 0)$ in the constraint, i.e. a constraint is assigned to timestage t_*, if it depends on x_{t_*} or ξ_{t_*} but not on x_t or ξ_t for $t > t_*$.

Decomposition information

Further on, a file with information about the block-structure has to be provided.

The first (nonempty) line contains the number of blocks p. Then information about the variables which belong to each block is successively given. A block is introduced by the keyword “NEXT” (also for the first block). Then the names of the variables in x_{I_k} are given.

Constraints are assigned to blocks by checking the variables, which appear in the constraint, i.e. a constraint is assigned to block k, if it depends only on x_{I_k} (and ξ), but is independent from $x_{I_{k'}}$, $k' \neq k$. If a constraint cannot be assigned to any block, it is treated as coupling constraint in the following.

Initial scenario tree

Finally, the initial scenario tree $\bar{N}$ has to be provided.

The first (nonempty) line contains the number of nodes $|\bar{N}|$. Then information about each node is successively given. It is assumed, that the nodes are sequently numbered. Each line consists of the distance to the predecessor node $n - n_-$, the node-probability π_n and the scenario values ξ^n. For the root node $n = 0$, the distance to the predecessor node can be an arbitrary number, but cannot left out.

Example

For an example, consider the model (3.19) for power management under uncertain load and a time horizon of one week, which was discretized hourly.

The GAMS-model takes the following form:

```
SET i Kohlekraftwerke / Bk_Lau_1, Bk_Lau_3, GuD_Kue, GT_Vock / ;
SET j Pumpspeicherwerke / GOLD, MARK / ;
SET t Timestages / 1*168 /;
PARAMETERS b(i) Brennstoffkosten Kohlekraftwerk i / ...  / ;
PARAMETERS p_min(i) min.  Leistung Kohlekraftwerk i / ...  / ;
PARAMETERS p_max(i) max.  Leistung Kohlekraftwerk i / ...  / ;
PARAMETERS eta(j) Wirkungsgrad Pumpen PSW j / ...  / ;
PARAMETERS w_max(j) max.  Pumpleistung PSW j / ...  / ;
PARAMETERS v_max(j) max.  Turbinenleistung PSW j / ...  / ;
PARAMETERS l_max(j) max.  Fuellstand PSW j / ...  / ;
PARAMETERS l_in(j) Fuellstand zu Beginn PSW j; l_in(j) = 0.9 * l_max(j) ;
PARAMETERS l_end(j) Fuellstand am Ende PSW j; l_end(j) = l_in(j) ;
VARIABLES
d(t) data Lastdaten fuer eine Woche
r(t) data Reserveleistung
p(i,t) deci Leistungswerte der thermischen Bloecke i
v(j,t) deci Turbinenwerte der Pumpspeichereinheiten j
w(j,t) deci Pumpwerte der Pumpspeichereinheiten j
l(j,t) deci Fuellstaende der Pumpspeichereinheiten j
cost Gesamte Kosten (Zielfunktionswert) ;
POSITIVE VARIABLES p, v, w, l ;
EQUATIONS
ziel_funktion Zielfunktion
l_anf_res(j) Bedingung Anfangs-Fuellstand
l_kopp_res(j,t) Zeitkopplung der Fuellstaende
last_res(t) Bedingung Lastdeckung
reserve_res(t) Bedingung Reserveforderung ;
ziel_funktion ..  cost =e= sum((i,t),b(i)*p(i,t));
loop(t,
p.lo(i,t)=p_min(i);
p.up(i,t)=p_max(i);
v.up(j,t)=v_max(j);
```

```
w.up(j,t)=w_max(j);
l.up(j,t)=l_max(j);
);
l_anf_res(j) ..  l(j,'1') =e= l_in(j) - (v(j,'1')-eta(j)*w(j,'1')) ;
l.fx(j,'168')=l_end(j);
l_kopp_res(j,t)$(ord(t)>1) ..  l(j,t) =e= l(j,t-1) - (v(j,t)-eta(j)*w(j,t)) ;
last_res(t) ..  sum(i,p(i,t)) + sum(j,v(j,t)-w(j,t)) =g= d(t) ;
reserve_res(t) ..  sum(i,p_max(i)-p(i,t)) =g= r(t) ;
MODEL veag_small_stoch_last / all / ;
```

The information about timestages is given in the following file:

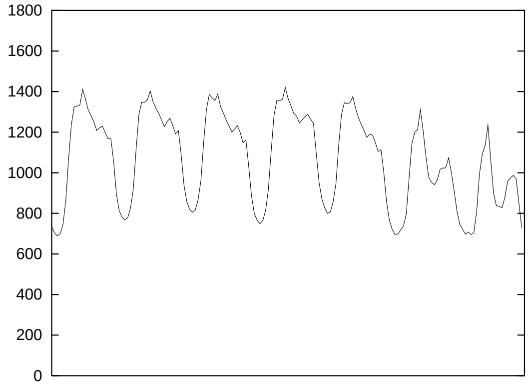
```
168
NEXT
d(1) r(1)
p(Bk_Lau_1,1) p(Bk_Lau_3,1) p(GuD_Kue,1) p(GT_Vock,1)
v(GOLD,1) w(GOLD,1) l(GOLD,1)
v(MARK,1) w(MARK,1) l(MARK,1)
NEXT
d(2) r(2)
p(Bk_Lau_1,2) p(Bk_Lau_3,2) p(GuD_Kue,2) p(GT_Vock,2)
v(GOLD,2) w(GOLD,2) l(GOLD,2)
v(MARK,2) w(MARK,2) l(MARK,2)
NEXT
...
NEXT
d(168) r(168)
p(Bk_Lau_1,168) p(Bk_Lau_3,168) p(GuD_Kue,168) p(GT_Vock,168)
v(GOLD,168) w(GOLD,168) l(GOLD,168)
v(MARK,168) w(MARK,168) l(MARK,168)
```

The blockstructure of this model is determined by the power stations and pump reservoirs. The file for the decomposition information has the form

```
6
NEXT
p(Bk_Lau_1,1) p(Bk_Lau_1,2) ...  p(Bk_Lau_1,168)
NEXT
p(Bk_Lau_3,1) p(Bk_Lau_3,2) ...  p(Bk_Lau_3,168)
NEXT
p(GuD_Kue,1) p(GuD_Kue,2) ...  p(GuD_Kue,168)
NEXT
p(GT_Vock,1) p(GT_Vock,2) ...  p(GT_Vock,168)
NEXT
v(GOLD,1) w(GOLD,1) l(GOLD,1)
v(GOLD,2) w(GOLD,2) l(GOLD,2)
...
v(GOLD,168) w(GOLD,168) l(GOLD,168)
NEXT
v(MARK,1) w(MARK,1) l(MARK,1)
v(MARK,2) w(MARK,2) l(MARK,2)
...
v(MARK,168) w(MARK,168) l(MARK,168)
```

4.2 Power management in a hydro-thermal system under uncertain load

ADoSP was run for the model (3.19) to find an optimal decision process for the operation of a collection of coal power stations and pump reservoirs under uncertain load, such that the expected fuel cost is minimized, see section 3.6.3. One scenario for the electric load of one week is displayed in the picture to the right. The seven days are good recognizable.
Two scenario trees for the modeling of the stochasticity in the electric load for one week were available. The first tree consists of 64 scenarios and 3048 nodes, the second one of 729 scenarios and 22623 nodes, see figures 4.2 and 4.3 for a visualization.



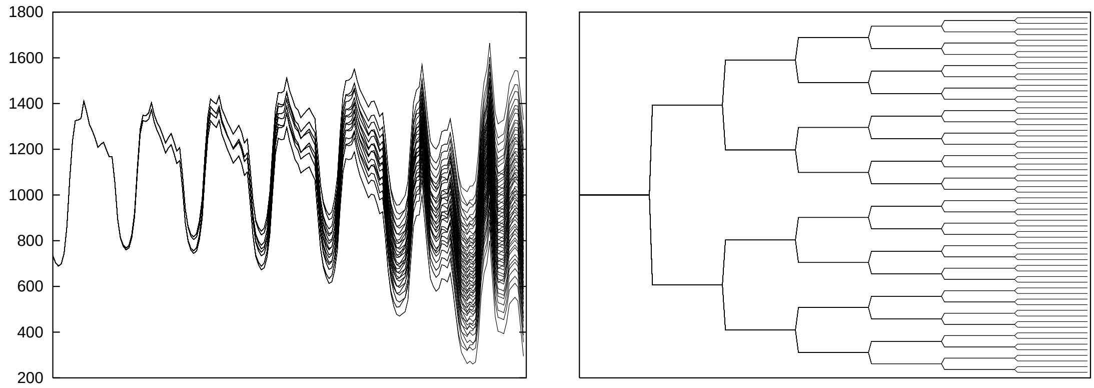


Figure 4.2: On the left, 64 scenarios of the electric load for one week. On the right, the corresponding scenario tree.

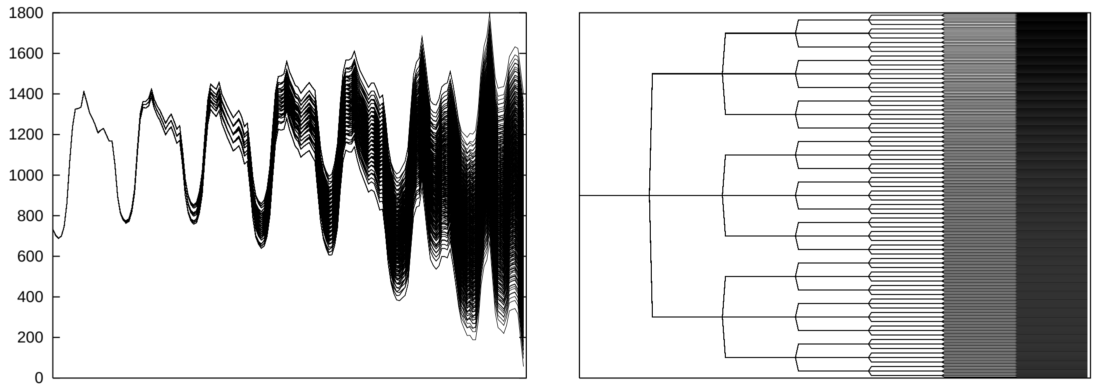


Figure 4.3: On the left, 729 scenarios of the electric load for one week. On the right, the corresponding scenario tree.

When running ADoSP for the tree on 64 scenarios and parameters $\sigma_P = 0.001$, $\sigma_D = 0.01$,

$\bar{m} = 2$, $s = 5$, it results in a tree on 25 scenarios, see figure 4.4. Figure 4.5 shows the progress of the optimal values $\mathrm{val}\left(\mathrm{LMSP}[\mathcal{N}]\right)$ and $\mathrm{val}\left(\mathrm{LMSP}\left[\mathcal{N}'\right]\right)$ and how the quantities $\frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])}$ and $\frac{\mathcal{D}(\mathcal{N}')}{\mathcal{D}_0}$ developed during the iteration progress. From the right picture in figure 4.5 one can observe in which iteration the condition $\frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])} < 1 + \sigma_P$ failed (when the curve falls below the marked interval, here iterations 3, 5, 11 and 14). In this case, ADoSP selects the set $\mathcal{M} \subseteq \mathcal{N}$ according to (3.18) so that $\mathrm{val}\left(\mathrm{LMSP}[\mathcal{N}]\right)$ increases. On the other side, when the condition $\frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])} < 1+\sigma_P$ fails (when the curve exceeds the marked interval), ADoSP solves problem $(\mathrm{ProjMSP}[\mathcal{N}, \mathcal{N}'])$ and selects $\mathcal{M}$ according to formula (2.2) to achieve a harmonization of $\mathrm{val}\left(\mathrm{LMSP}[\mathcal{N}]\right)$ and $\mathrm{val}\left(\mathrm{LMSP}\left[\mathcal{N}'\right]\right)$. If otherwise $\frac{1}{1+\sigma_P} \leq \frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])} \leq 1+\sigma_P$, but $\frac{\mathcal{D}(\mathcal{N}')}{\mathcal{D}_0} > \sigma_D$ then $\mathcal{N}'$ is refined, i.e. a set $\mathcal{M} \subseteq \mathcal{N}$ is selected such that the distance from $\bar{N}\left|_{\mathcal{N}'(\mathcal{M})}\right.$ to $\bar{N}$ is diminished. In the left picture, one can recognize these iterations because there is no change in $\mathrm{val}\left(\mathrm{LMSP}[\mathcal{N}]\right)$ in this case.

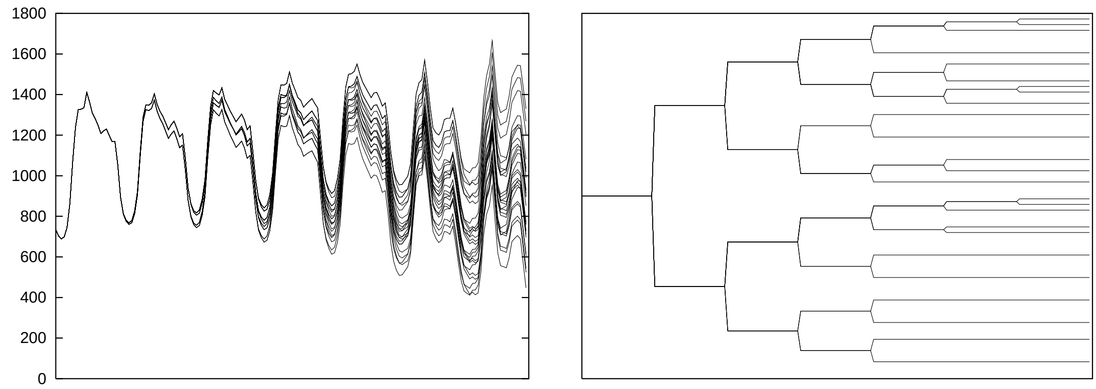


Figure 4.4: Reduced scenario tree on 25 scenarios. The initial scenario tree had 64 scenarios. Parameters were $\sigma_p = 0.001$, $\sigma_D = 0.01$

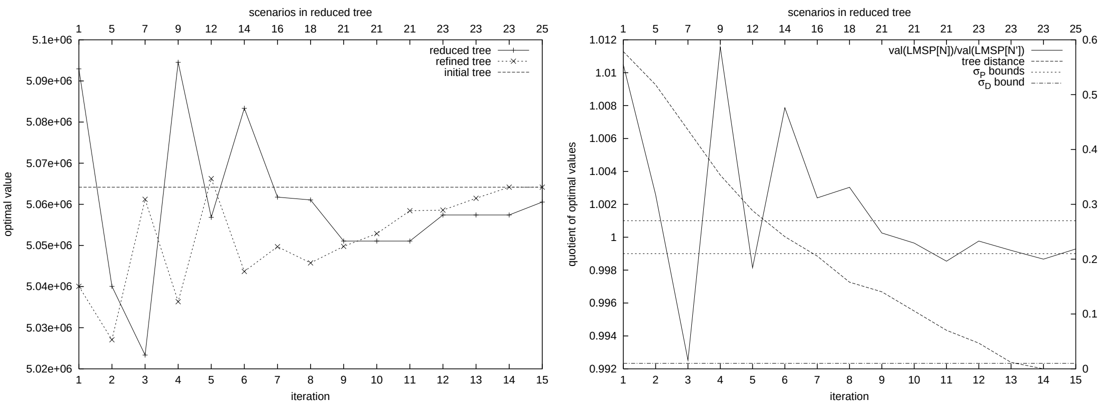


Figure 4.5: Iteration progress for an initial tree of 64 scenarios and $\sigma_p = 0.001$, $\sigma_D = 0.01$. The left picture shows the optimal values for the reduced tree $\bar{N}\left|_{\mathcal{N}}\right.$, the refined tree $\bar{N}\left|_{\mathcal{N}'}\right.$ and the initial tree $\bar{N}$ (straight line). The right picture shows the quotient $\frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])}$ and marks the interval $\left[\frac{1}{1+\sigma_P}, 1+\sigma_P\right]$ on the left ordinate and shows the quotient $\frac{\mathcal{D}(\mathcal{N}')}{\mathcal{D}_0}$ and marks the bound σ_D on the right ordinate.

To visualize the influence of the parameters σ_P and σ_D on the number of scenarios and nodes of the reduced tree, several runs with $\sigma_P \in \{0.1, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$ and $\sigma_D \in \{0.5, 0.1, 0.05, 0.01\}$ have been done. Figure 4.9 at the end of this section shows the iteration progress for selected parameter combinations, the tables 4.1 summarize the number of scenarios and nodes of the final reduced tree, respectively. While σ_P has obviously an high impact on the final number of scenarios, the distance $\mathcal{D}(\mathcal{N}')$, controlled by σ_D, seems to be of minor importance.

$\sigma_p \backslash \sigma_D$	0.5	0.1	0.05	0.01
0.1	1	1	1	1
0.01	5	5	5	5
0.005	6	6	6	6
0.001	21	23	23	25
0.0005	21	23	23	27
0.0001	39	39	39	41

$\sigma_p \backslash \sigma_D$	0.5	0.1	0.05	0.01
0.1	168	168	168	168
0.01	624	624	624	624
0.005	720	720	720	720
0.001	1728	1824	1824	1872
0.0005	1728	1824	1824	1920
0.0001	2424	2424	2424	2472

Table 4.1: Number of scenarios (left) and nodes (right) of the final reduced tree $\bar{N}|_{\mathcal{N}}$ for $\sigma_P \in \{0.1, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$ and $\sigma_D \in \{0.5, 0.1, 0.05, 0.01\}$.

We turn over to the reduction of the tree on 729 scenarios now. Due to the higher number of scenarios, we also increased the parameters $\bar{m}$ and s so that the number of scenarios in $\bar{N}|_{\mathcal{N}}$ and $\bar{N}|_{\mathcal{N}'}$ can increase faster during refinement operations. For $\sigma_P = 0.001$, $\sigma_D = 0.01$, $\bar{m} = 10$, $s = 25$, ADoSP ends in in a tree on 77 scenarios, see figure 4.6. Figure 4.7 shows the progress of the optimal values $\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])$ and $\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])$ and how the quantities $\frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])}$ and $\frac{\mathcal{D}(\mathcal{N}')}{\mathcal{D}_0}$ developed during the iteration progress. One observes that after the 6th iteration, the remaining iterations were used to push $\frac{\mathcal{D}(\mathcal{N}')}{\mathcal{D}_0}$ below the tolerance σ_D. When increasing σ_D to 0.1, one obtains the same tree, but the number of iterations decreases from 22 to 15, see figure 4.8. Already a value σ_D of 0.3 would have been enough to obtain the same result.

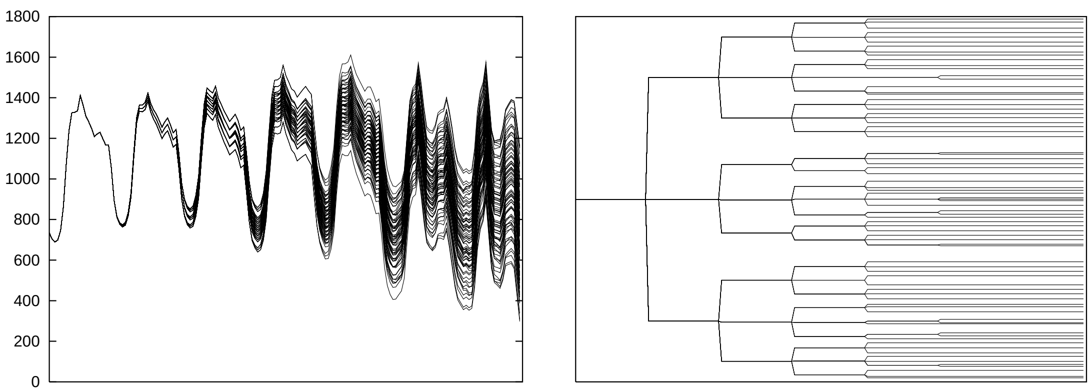


Figure 4.6: Reduced scenario tree on 77 scenarios. The initial scenario tree had 729 scenarios. Parameters were $\sigma_p = 0.001$, $\sigma_D = 0.01$

Again, we visualize the influence of the parameters σ_P and σ_D on the number of scenarios and nodes of the reduced tree. Several runs with $\sigma_P \in \{0.01, 0.001, 0.0001\}$ and $\sigma_D \in \{0.1, 0.01\}$ have been done. Figure 4.10 shows the iteration progress, the tables 4.2 summarize the number of scenarios and nodes of the final reduced tree, respectively.

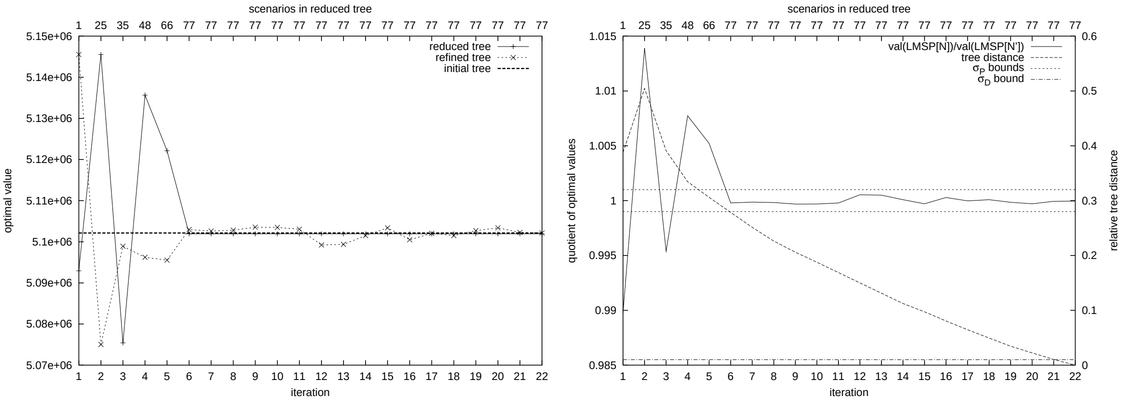


Figure 4.7: Iteration progress for an initial tree of 729 scenarios and $\sigma_p = 0.001$, $\sigma_D = 0.01$. The left picture shows the optimal values for the reduced tree $\bar{N}\,|_{\mathcal{N}}$, the refined tree $\bar{N}\,|_{\mathcal{N}'}$ and the initial tree $\bar{N}$ (straight line). The right picture shows the quotients $\frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])}$ and $\frac{\mathcal{D}(\mathcal{N}')}{\mathcal{D}_0}$.

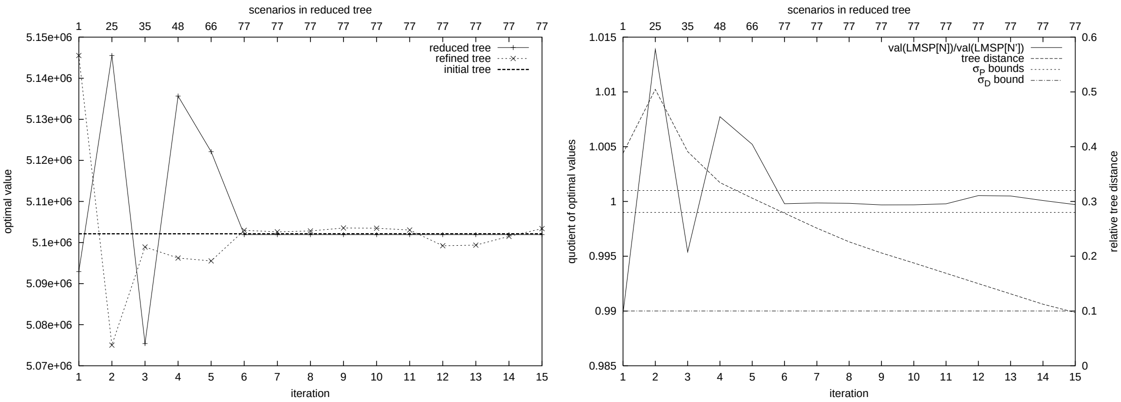


Figure 4.8: Iteration progress for an initial tree of 729 scenarios and $\sigma_p = 0.001$, $\sigma_D = 0.1$. The left picture shows the optimal values for the reduced tree $\bar{N}\,|_{\mathcal{N}}$, the refined tree $\bar{N}\,|_{\mathcal{N}'}$ and the initial tree $\bar{N}$ (straight line). The right picture shows the quotients $\frac{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}])}{\mathrm{val}(\mathrm{LMSP}[\mathcal{N}'])}$ and $\frac{\mathcal{D}(\mathcal{N}')}{\mathcal{D}_0}$.

$\sigma_p \backslash \sigma_D$	0.1	0.01
0.01	47	47
0.001	77	77
0.0001	294	630

$\sigma_p \backslash \sigma_D$	0.1	0.01
0.01	4152	4152
0.001	6240	6240
0.0001	15672	23856

Table 4.2: Number of scenarios (left) and nodes (right) of the final reduced tree $\bar{N}\,|_{\mathcal{N}}$ for $\sigma_P \in \{0.01, 0.001, 0.0001\}$ and $\sigma_D \in \{0.1, 0.01\}$.

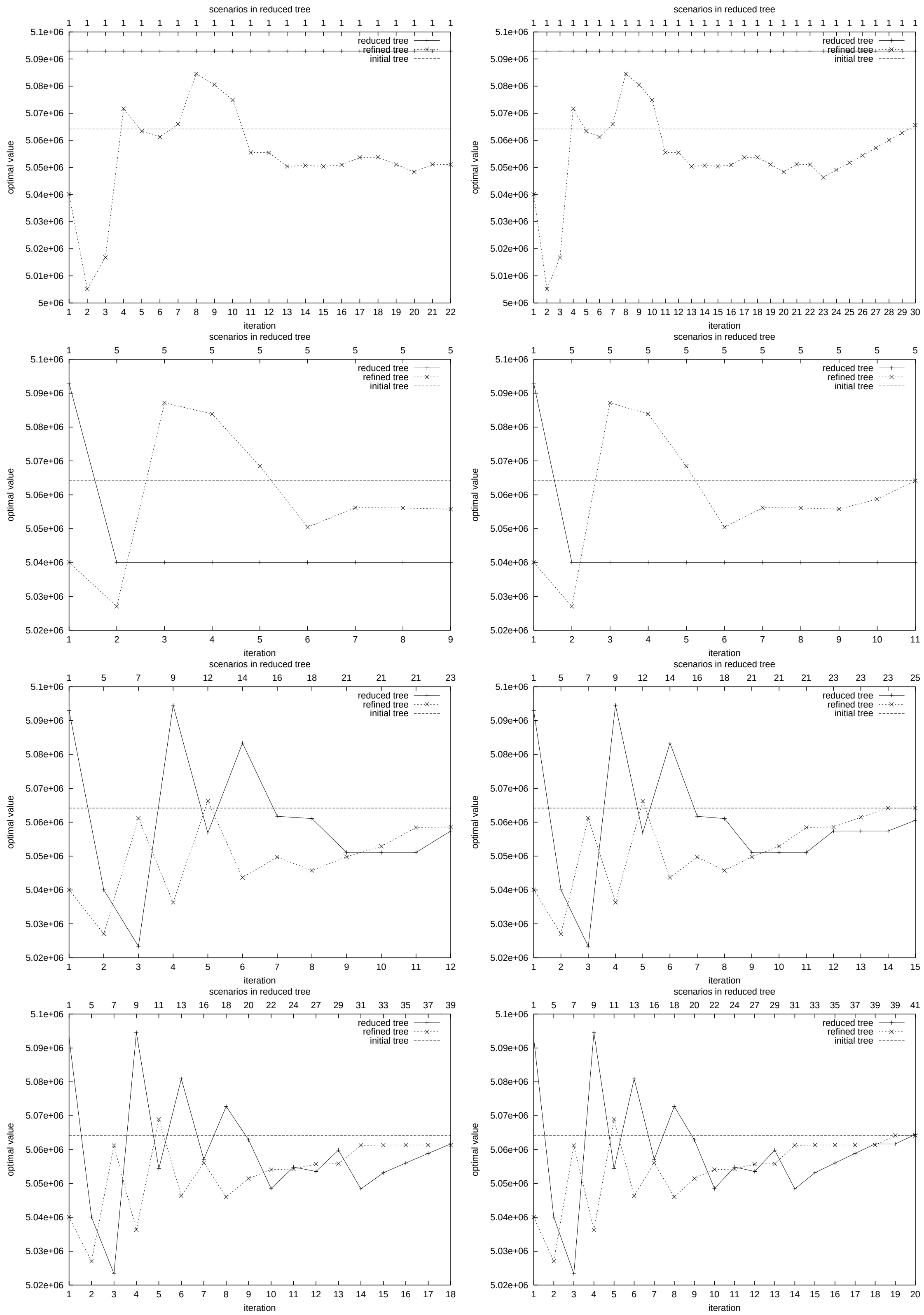


Figure 4.9: Iteration progress for $\sigma_P \in \{0.1, 0.01, 0.001, 0.0001\}$ (rowwise) and $\sigma_D \in \{0.1, 0.01\}$ (columnwise). The initial tree had 64 scenarios.

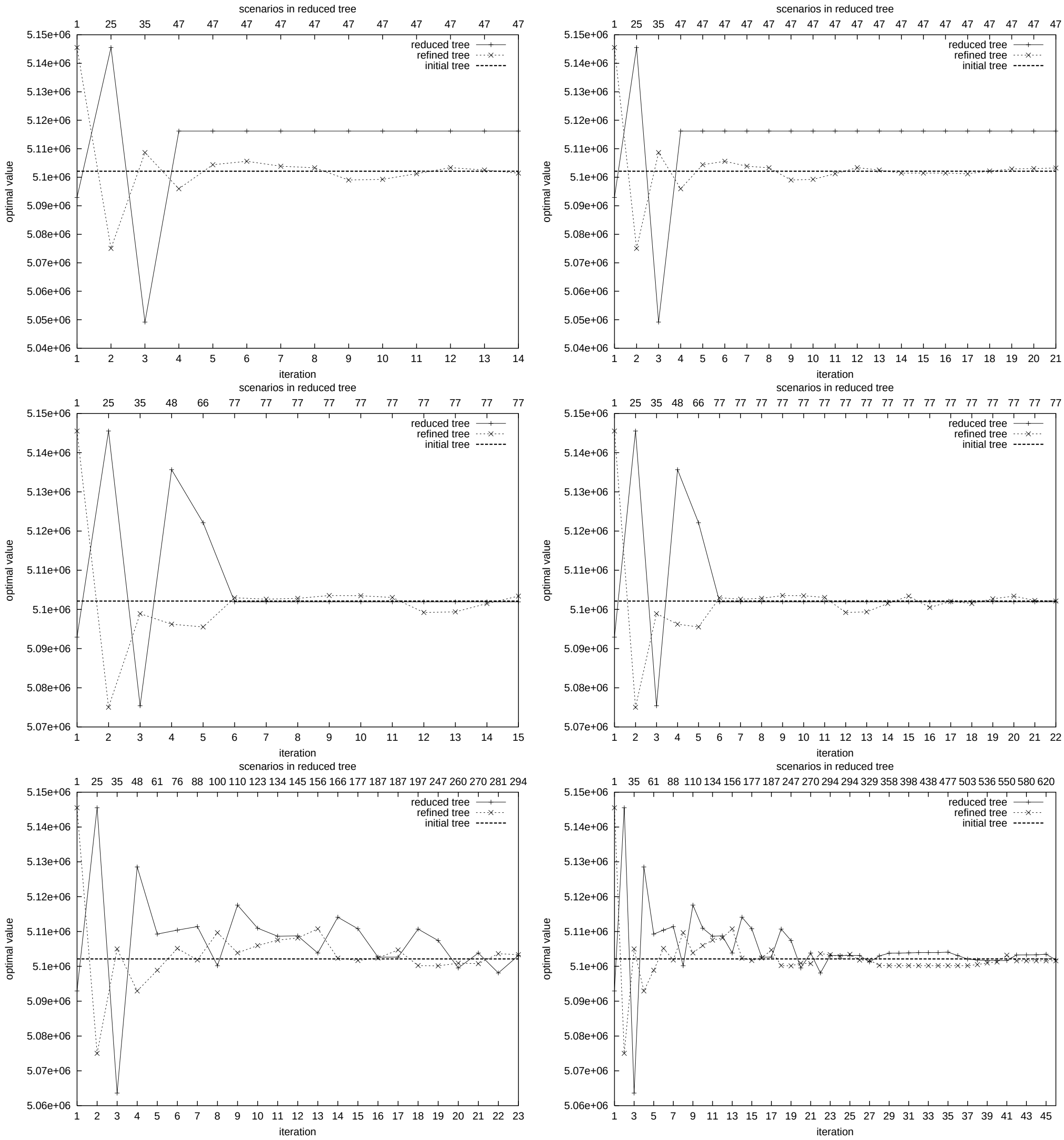


Figure 4.10: Iteration progress for $\sigma_P \in \{0.01, 0.001, 0.0001\}$ (rowwise) and $\sigma_D \in \{0.1, 0.01\}$ (columnwise) . The initial tree had 729 scenarios.

Chapter 5

Finish

Of course, at the current stage, the presented algorithm and it's implementation has more than a few weak points. We will discuss here some of the current disadvantages and give suggestions for further developments. Finally, this work is round up with some theses.

5.1 Further developments

Initial Scenario Tree $\bar{N}$

As one can already observe for the tree on 729 scenarios, the time to select scenarios by fast-forward selection becomes an important factor on the overall solution time when the number s of scenarios to select from the initial set N_T increases. From [15] we know that fast-forward selection takes time $O\left(s^3 + s\left|N_T\right|^2\right)$. Since we have to select only a small number of scenarios in each iteration, an option could be to start with a rather small tree $\bar{N}$ and increase it's size during the iteration progress. For a very large set $\mathfrak{N}$ of nodes, which doesn't need to be known explicitly, define the node set $\bar{N}$ as a (finite) aggregation of $\mathfrak{N}$. During the refinement of $\mathcal{N}'$ one could enlarge $\bar{N}$ by refining those $N' \in \mathcal{N}'$ with $|N'|$ small.

Furthermore, this approach allows a generalization to continuous distributions, so that the construction of scenarios, or the generation of the scenario-tree, is integrated in the solution process.

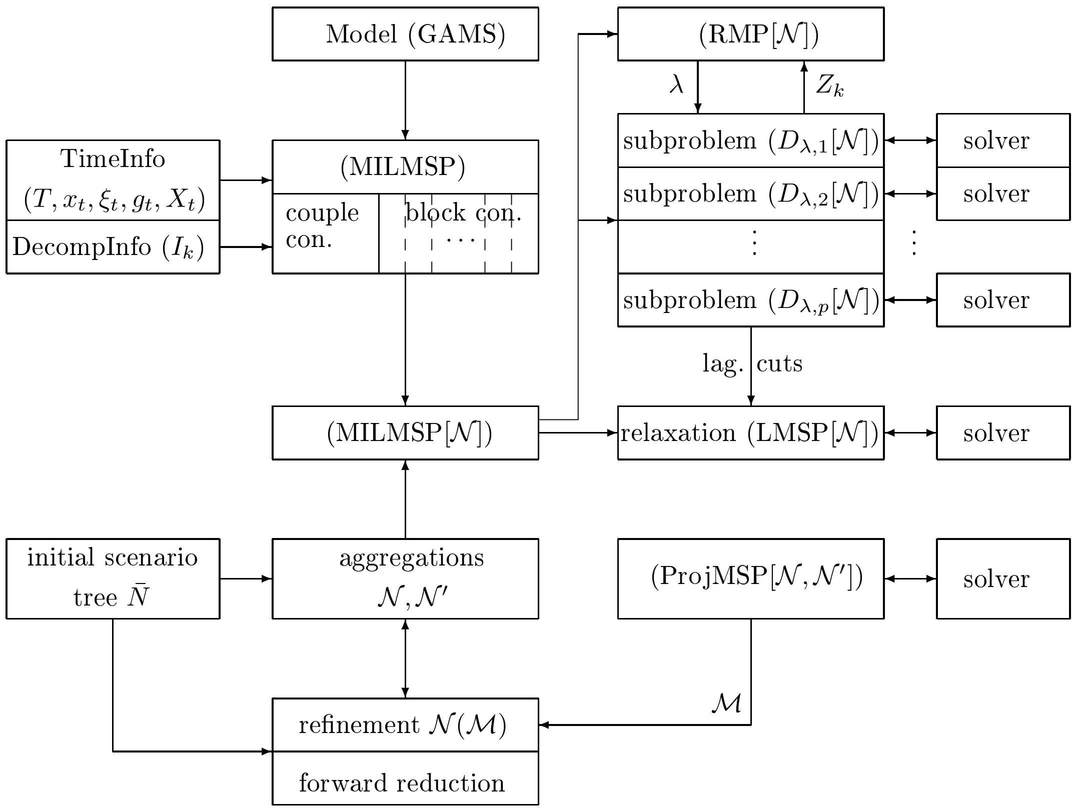
(Proj$[\mathcal{N}, \mathcal{N}']$)

An even higher impact on the solution time is the effort spend for solving (Proj$[\mathcal{N}, \mathcal{N}']$) (or (ProjMSP$[\mathcal{N}, \mathcal{N}']$) respectively). Since we are only interested to find $N \in \mathcal{N}$ which are of value for a refinement, it is possible to solve (Proj$[\mathcal{N}, \mathcal{N}']$) with less accuracy. Another disadvantage of (Proj$[\mathcal{N}, \mathcal{N}']$) is that due to the objective function $\sum_{N \in \mathcal{N}} \left\|W_N x_N\right\|_\infty$ a high interaction between many of the variables is introduced that is even enforced by the constraint $F_{\mathcal{N}'}(x) - \mathrm{val}\left(C\left[\mathcal{N}'\right]\right) \leq \sigma_P \left|\mathrm{val}\left(C\left[\mathcal{N}'\right]\right)\right|$. In case of multistage stochastic programs this prohibits to inherit the structure of (MSP$[\mathcal{N}']$) into (ProjMSP$[\mathcal{N}, \mathcal{N}']$). Therefore, a more elaborate but maybe heuristic method for selecting the set $\mathcal{M}$ in case that $\mathrm{val}\left(C\left[\mathcal{N}\right]\right) - \mathrm{val}\left(C\left[\mathcal{N}'\right]\right) > \sigma_P \left|\mathrm{val}\left(C\left[\mathcal{N}'\right]\right)\right|$ is desirable.

Implementation of column generation

The software package ADoSP has to be extended to mixed-integer linear multistage stochastic programs. For this, convex relaxations (like (LMSP$[\mathcal{N}]$)) have to be introduced and cut-generation must be implemented. The column generation method, as proposed in section 2.4.2, is a preferable candidate. Some of the best present-day solvers for mixed integer linear programs are successful implementations of the column generation method. For the implementation of this method, many aspect like the correct initialization of the (RMP) have to be concerned. Another important component is the fast solution of the high number of Lagrangian subproblems $(D_{\lambda,k})$, which appear after an update of the dual parameter. The concept of "explored" subproblems might be used to reduce this number, but this is only possible if one works with a feasible (RMP). Above this, specialized solvers which exploit the structure of the subproblems can effect in a high acceleration for the complete method.

A design of ADoSP that involves column generation might have the following form:



Lagrange Heuristic

Something not pointed out in this work, is the fact that the solution of $(C[\mathcal{N}])$ for the reduced tree $\mathcal{N}$ in the final iteration yields a solution point that might already be close to a solution point of $(P[\mathcal{N}])$, depending on the quality of the convex relaxation. In the case of mixed-integer linear MSPs this point will in general not fulfill some or all integrality constraints. Hence, a clever rounding heuristic could be used to find a feasible and near optimal solution point of $(P[\mathcal{N}])$.

When using column generation, observe that the extreme points $Z_{k,\mathcal{N}}$ already satisfy the block-constraints $X_{k,\mathcal{N}}$, $k = 1, \dots, p$. Therefore, a heuristic to find starting points for a local search in $(P[\mathcal{N}])$ by combining extreme points such that the violation of the coupling constraints is small, is another possible option to obtain a solution point of $(P[\mathcal{N}])$. Nowak proposed in [17] a simple Lagrange Heuristic that utilizes extreme points from a column generation method.

In [12], a Lagrange heuristics is used to drive a primal solution point, obtained from a proximal bundle method, to a near optimal feasible point of $(P[\mathcal{N}])$. This Lagrange heuristic makes use of the structure of the model when deciding which binary variables should be switched. Therefore, it has the disadvantage, that changes on the model require modifications on the heuristic, but on the other hand it might produce solution candidates in much fewer time than a general heuristic.

Nonlinearities

Finally, it is also imaginable to extend ADoSP to the case that the sets X_k are not polyhedral, i.e. described by nonlinear equations and integrality constraints. Computing a linear outer approximation of X_k yields a linear problem $(C\,[\mathcal{N}])$ that can still be solved efficiently in high dimensions. When updating this relaxation by (linear) Lagrangian cuts, the nonlinear Lagrangian subproblems $(D_{\lambda,k}\,[\mathcal{N}])$ have to be solved to (global) optimality. Again, this is tractable, if specialized solution algorithms are available. If this is not the case and the X_k are not convex, we have to start up with a general MINLP solver, what currently allows problems of small size only.

5.2 Theses

- The reduction of scenario trees is an important factor when stochastic programs have to be solved efficiently.
- Especially adaptive discretization algorithms, which integrate the generation or reduction of scenario trees into the solution process of multistage stochastic programs, are of high interest.
- The adaptive discretization algorithm proposed in this work yields an extension of the backward-reduction and forward-selection algorithms from [12, 15] in the sense that a better estimation of the the gap between the reduced problem and the initial one is possible.
- To allow an estimate for the deviation from the optimal value of the initial program, convex relaxations can be used. Cut generation is an important factor to reduce the gap that is introduced thereby.
- Column generation, based on Lagrangian Relaxation and Decomposition techniques, builds a promising approach to handle large size stochastic programs.
- For linear multistage stochastic programs, an adaptive discretization algorithm has successfully been implemented.
- Adaptive discretization of scenario trees for stochastic programs in power management environments allows the reduction of scenario trees up to 10 to 30% (in number of scenarios) while loosing at most 0.1% accuracy in the optimal value.

Bibliography

[1] M. S. Casey, S. Sen, The Scenario Generation Algorithm for Multistage Stochastic Linear Programming, to appear in Mathematics of Operations Research

[2] Computational Infrastructure for Operations Research, `http://www.coin-or.org`

[3] CPLEX, `http://www.ilog.com`

[4] M.A.H. Dempster, R.T. Thompson, EVPI-based importance sampling solution procedures for multistage stochastic programmes on parallel MIMD architectures, Annals of Operations Research, 90 (1999), 161-184

[5] M.A.H. Dempster, R.T. Thompson, Parallelization and Aggregation of Nested Benders Decomposition, Annals of Operations Research, 81, proceedings of the APMOD95 Conference, Brunel University

[6] D. Dentcheva, W. Römisch: Duality gaps in nonconvex stochastic optimization, Mathematical Programming Ser. A 101 (2004), 515-535

[7] J. Dupačová, N. Gröwe-Kuska, W. Römisch, Scenario Reduction in Stochastic Programming - An Approach using Probability Metrics, Mathematical Programming, Ser. A 95 (2003), 493-511

[8] N. C. P. Edirisinghe, Bound-based approximations in multistage stochastic programming: Nonanticipativity aggregation, Annals of Operations Research 85, 103-127 (1999)

[9] A. Eichhorn, N. Gröwe-Kuska, A. Liebscher, W. Römisch, G- Spangardt, I. Wegner, Mean-risk optimization of electricity portfolios, PAMM Proc. Appl. Math. Mech. 4 (2004), 3–6.

[10] I.V. Evstigneev, Measurable selections and dynamic programming, Math. of Operations Research, 1, 1976

[11] General Algebraic Modeling System, `http://www.gams.com`

[12] N. Gröwe-Kuska, H. Heitsch, W. Römisch, Scenario Reduction and Scenario Tree Construction for Power Management Problems, IEEE Bologna Power Tech Proceedings (A. Borghetti, C.A. Nucci, M. Paolone eds.), 2003 IEEE.

[13] N. Gröwe-Kuska, K. C. Kiwiel, M. P. Nowak, W. Römisch, I. Wegner, Power Management in a hydro-thermal system under uncertainty by Lagrangian Relaxation, in: Decision Making under Uncertainty: Energy and Power (C. Greengard, A. Ruszczynski eds.), IMA Volumes in Mathematics and its Applications Vol. 128, Springer-Verlag, New York 2002, 39-70.

[14] N. Gröwe-Kuska, A. Liebscher, M. Lucht, W. Römisch, G. Spangardt, I. Wegner, Mittelfristige risikoorientierte Optimierung von Strombeschaffungs-Portfolios kleinerer Marktteilnehmer, Preprint 03-11, Institut für Mathematik, Humboldt-Universität Berlin, 2003

[15] H. Heitsch, W. Römisch, Scenario Reduction Algorithms in Stochastic Programming, Computational Optimization and Applications 24 (2003), 187-206

[16] J. L. Higle, B. Rayco, S. Sen, Stochastic Scenario Decomposition for Multi-Stage Stochastic Programs, submitted to Annals of Operations Research

[17] I. Nowak, Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming, Habilitation thesis Humboldt University to Berlin, 2004

[18] S. T. Rachev, Probability metrics and the stability of stochastic models, Wiley, 1991

[19] W. Römisch, Lecture "Stochastische Optimierung", Humboldt University to Berlin, wintersemester 2002/03

[20] A. Ruszczyński, A. Shapiro, Stochastic Programming, Handbooks in Operations Research and Management Science, Volume 10, Elsevier Science, 2003