TWO-STAGE STOCHASTIC PROGRAMMING FOR ENGINEERING PROBLEMS

Pavel Popela*, Jan Novotný*, Jan Roupec**, Dušan Hrabec*, Asmund Olstad***

The purpose of the paper is to present existing and discuss modified optimization models and solution techniques which are suitable for engineering decision-making problems containing random elements with emphasis on two decision stages. The developed approach is called two-stage stochastic programming and the paper links motivation, applicability, theoretical remarks, transformations, input data generation techniques, and selected decomposition algorithms for generalized class of engineering problems. The considered techniques have been found applicable by the experience of the authors in various areas of engineering problems. They have been applied to engineering design problems involving constraints based on differential equations to achieve reliable solutions. They have served for technological process control e.g. in melting, casting, and sustainable energy production. They have been used for industrial production technologies involving related logistics, as e.g. fixed interval scheduling under uncertainty. The paper originally introduces several recent improvements in the linked parts and it focuses on the unified two-stage stochastic programming approach to engineering problems in general. It utilizes authentic experience and ideas obtained in certain application areas and advises their fruitful utilization for other cases. The paper follows the paper published in 2010 which deals with the applicability of static stochastic programs to engineering design problems. Therefore, it refers to the basic concepts and notation introduced there and reviews only the principal ideas in the beginning. Then, it focuses on motivation of recourse concepts and two decision stages from engineering point of view. The principal models are introduced and selected theoretical features are reviewed. They are also accompanied by the discussion about difficulties caused by real-world cases. Scenario-based approach is detailed as the important one for the solution of engineering problems, discussion in data input generation is added together with model transformation remarks. Robust algorithms suitable for engineering problems involving nonlinearities and integer variables are selected and scenario-based decomposition is preferred. An original experience with using heuristics is shared. Several postprocessing remarks are added at the end of the paper, which is followed by an extensive literature review.

Keywords: *two-stage stochastic programming, modelling, scenarios, decomposition, algorithms*

1. Introduction

Underlying mathematical program. The important property of problems considered in this paper is based on the fact that the decisions must be made under uncertainty. This

* RNDr. P. Popela, Ph.D., Ing. J. Novotný, Ing. D. Hrabec, Institute of Mathematics, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2896/2, 616 69 Brno, Czech Republic

** Ing. J. Roupec, Ph.D., Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2896/2, 616 69 Brno, Czech Republic

*** prof. A. Olstad, PhD, Department of Economics, Faculty of Economics, Informatics and Social Science, Molde University College, Britvegen 2, 6402 Molde, Norway

represents a case when traditional optimization models are limited in practical applications because their parameters are not completely known. Many real-world complex systems include these uncertainties, and they can be modelled in different ways. One of them is stochastic optimization. We will focus on the cases where the programs' parameters are influenced by random variables. In addition, we will assume that (two) subsequent decisions can be considered. We will discuss related modelling approach in this paper that will be noted as two-stage stochastic programming. Generalizing traditional mathematical program, we have

$$? \in \underset{\mathbf{x}}{\operatorname{argmin}}\{f(\mathbf{x};\boldsymbol{\xi}) \mid \mathbf{x} \in C(\boldsymbol{\xi}) = \{\mathbf{x} \in X \mid \mathbf{g}(\mathbf{x};\boldsymbol{\xi}) \circ \mathbf{0}\}\}\ , \tag{1}$$

where $\circ$ represents a column vector of symbols $\circ_i \in \{\leq, \geq, =\}$, and the feasible set $C(\boldsymbol{\xi})$ may be rewritten as $C(\boldsymbol{\xi}) = \bigcap_{i=1}^{m} C_i(\boldsymbol{\xi}) = \bigcap_{i=1}^{m}\{\mathbf{x} \in X \mid g_i(\mathbf{x},\boldsymbol{\xi}) \circ_i 0\}$, or $C(\boldsymbol{\xi}) = \{\mathbf{x} \in X \mid g_i(\mathbf{x},\boldsymbol{\xi}) \leq 0, 1 \leq i \leq l; g_i(\mathbf{x},\boldsymbol{\xi}) = 0, l+1 \leq i \leq m\}$. As the next step, we have to identify the meaning of the given program (1). It is clear when observation $\boldsymbol{\xi}^s$ substitutes for $\boldsymbol{\xi}$ but what happens to the program (1), when the realization of the randomness is not observed? So, it is understood as a syntactically correct description useful for modelling purposes, for which semantics will be given later by so called deterministic reformulations. Since the program (1) is based on a mathematical program resulting from the substitution of selected constants by random parameters, it is called an underlying mathematical program. There is $\boldsymbol{\xi}$, a random vector defined on the probability space $(\Xi, \Sigma, \mathcal{P})$, and $f : \mathbb{R}^n \times \Xi \longrightarrow \mathbb{R}$, $\mathbf{g} : \mathbb{R}^n \times \Xi \longrightarrow \mathbb{R}^m$ are measurable functions for every $\mathbf{x} \in \mathbb{R}^n$. We can see that the main components of a stochastic program are a decision vector $\mathbf{x}$ that must satisfy constraints $\mathbf{x} \in C(\boldsymbol{\xi})$, a decision criterion f and a probability distribution $\mathcal{P}$ of the random variable $\boldsymbol{\xi}$. Further details on static (one-stage) stochastic programs can be found in the previous paper [22]. The paper focuses on two-stage stochastic programs. More information and references on similar theme can be found in [19], [20] and in [21]. However, the emphasize is put on modelling and a nonlinear case that is suitable for engineering applications (see [12], [29], [37] and [18]) and cases with integer variables are also mentioned. It is important to notice that many introduced modelling and algorithmic ideas come from application areas. Surprisingly, various financial models involving utility functions may bring inspiration for nonlinear programs designed for engineering problems, see [21]. In addition, various applications in logistics (see [17] or scheduling problems as a fixed interval scheduling problem) involve integer variables and might be even nonlinear because of pricing related terms, and hence, they are inspiring design of engineering models where some structural optimization has to be considered.

Wait-and-see and here-and-now approaches. The main question that should be answered is when the decision will be made — before the random parameters $\boldsymbol{\xi}$ are observed or after the observations $\boldsymbol{\xi}^s$ are known. When the decision $\mathbf{x}$ is made after the observing the randomness $\boldsymbol{\xi}$, this case is called the *wait-and-see* (WS) approach. Decision makers must often make decisions before the observations of $\boldsymbol{\xi}$ are known. In this case, they are using a so called *here-and-now* (HN) approach as the decision $\mathbf{x}$ must be the same for any future realization of $\boldsymbol{\xi}$.

Penalty for infeasibility, recourse constraints, and randomness in penalty. The important modelling approach in stochastic programming we will further follow in the paper is based on the idea that constraints may be relaxed, and hence, the discrepancies will be penalized.

This idea leads to the program:

$$? \in \underset{\mathbf{x}}{\operatorname{argmin}}\{f^{\mathrm{RP}}(\mathbf{x}) \mid \mathbf{x} \in C_x\} \qquad \text{or} \qquad \min_{\mathbf{x}}\{f^{\mathrm{EO}}(\mathbf{x}) + \mathcal{Q}(\mathbf{x}) \mid \mathbf{x} \in C_x\}\ , \tag{2}$$

where RP denotes Recourse Program, EO means Expected Objective and C_x is a feasible set for the decision $\mathbf{x}$ and is defined only by deterministic constraints describing C. We left out explicit constraints depending on a random parameter $\boldsymbol{\xi}$. The chosen optimal decision $\mathbf{x}$ is based on the objective $f^{\mathrm{RP}}(\mathbf{x})$ composed of the expected cost part $f^{\mathrm{EO}}(\mathbf{x}) = E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\}$ and the additional cost denoted $\mathcal{Q} : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ where the infinite values of $\mathcal{Q}(\mathbf{x})$ are assigned for 'faraway decisions' $\mathbf{x}$. We suppose that the constraints $\mathbf{g}(\mathbf{x};\boldsymbol{\xi}) \circ \mathbf{0}$ have a soft form. It means that if a constraint is not satisfied then the WS recourse action $\mathbf{y}(\boldsymbol{\xi})$ can be taken to get the feasibility back. Such a recourse action usually depends on the random parameters $\boldsymbol{\xi}$, because it follows after the observation of the infeasibility caused by a random influence $C^{\mathrm{RP}} = \{\mathbf{x} \in X \mid \exists \mathbf{y}(\boldsymbol{\xi}) \in Y : \mathbf{h}(\mathbf{g}(\mathbf{x};\boldsymbol{\xi}), \mathbf{y}(\boldsymbol{\xi})) \circ \mathbf{0},\ \text{a.s.}\}$. The function $\mathbf{h}(.,.)$ describes how recourse actions enter the set of constraints. It seems reasonable to expect that the function $\mathcal{Q}(\mathbf{x})$ has to depend on the constraint functions that involve random parameters. This results in the fact that the penalty function $\mathcal{Q}(\mathbf{x})$ must either 'suppress' the influence of a random parameter $\boldsymbol{\xi}$ or its definition is not complete and its value changes with a varying value of $\boldsymbol{\xi}$. Therefore, we extend a concept of the penalty $\mathcal{Q}(\mathbf{x})$ taking into account $\boldsymbol{\xi}$, and we define $Q : \mathbb{R}^n \times \Xi \longrightarrow \mathbb{R} \cup \{+\infty\}$, so called a recourse cost function. It is assumed that this function is always measurable. This new penalty function $Q(\mathbf{x};\boldsymbol{\xi})$ changes, with the decision $\mathbf{x}$ and observation of $\boldsymbol{\xi}$, so we need to assign the deterministic reformulation to it. Then we return to $\mathcal{Q}(\mathbf{x})$ symbol and define an average recourse cost, as $\mathcal{Q}(\mathbf{x}) = E_\xi\{Q(\mathbf{x};\boldsymbol{\xi})\}$. This cost also represents an average penalty for repeated observations. If $Q(\mathbf{x};\boldsymbol{\xi}) = +\infty$ with strictly positive probability, then $\mathcal{Q}(\mathbf{x}) = +\infty$.

Recourse action and recourse program. The further extension is related to the idea that the recourse cost $Q(\mathbf{x};\boldsymbol{\xi})$ is based on an additional recourse action $\mathbf{y}^*$, correcting the result of a realization of random parameters in such a way that constraints with random parameters are satisfied after this action. The recourse action is made after the observations are known, then it depends on $\boldsymbol{\xi}$, and we have $\mathbf{y}^*(\boldsymbol{\xi})$ together with additional costs $q(\mathbf{x}, \mathbf{y}^*(\boldsymbol{\xi});\boldsymbol{\xi})$. It is reasonable to assume that the recourse action can be often realized in different ways, satisfying for the fixed $\mathbf{x} : \mathbf{y}^*(\boldsymbol{\xi}) \in C_y^{\mathrm{RP}}(\mathbf{x}) = C_y(\mathbf{x};\boldsymbol{\xi}) = \{\mathbf{y}(\boldsymbol{\xi}) \in Y \mid \mathbf{h}(\mathbf{g}(\mathbf{x};\boldsymbol{\xi}), \mathbf{y}(\boldsymbol{\xi})) \circ \mathbf{0}\ \text{a.s.}\}$. For various feasible recourse actions $\mathbf{y}(\boldsymbol{\xi})$, we may obtain different prices $q(\mathbf{x}, \mathbf{y}(\boldsymbol{\xi});\boldsymbol{\xi})$, and naturally, we will search the cheapest recourse action $\mathbf{y}_{\min}(\boldsymbol{\xi})$. Hence, we will obtain $Q(\mathbf{x};\boldsymbol{\xi}) = q(\mathbf{x}, \mathbf{y}_{\min}(\boldsymbol{\xi});\boldsymbol{\xi})$. For each realization $\boldsymbol{\xi}^s$ of $\boldsymbol{\xi}$ and decision $\mathbf{x}$, we have the following recourse program (RP):

$$Q(\mathbf{x};\boldsymbol{\xi}^s) = \min_{\mathbf{y}(\xi^s)}\{q(\mathbf{x}, \mathbf{y}(\boldsymbol{\xi}^s);\boldsymbol{\xi}^s) \mid \mathbf{h}(\mathbf{g}(\mathbf{x};\boldsymbol{\xi}^s), \mathbf{y}(\boldsymbol{\xi}^s)) \circ \mathbf{0}, \mathbf{y}(\boldsymbol{\xi}^s) \in Y\}\ . \tag{3}$$

This program (3) defines the recourse action $\mathbf{y}_{\min}(\boldsymbol{\xi})$ implicitly. Using inf instead of min in the recourse program assigns $+\infty$ value to $Q(\mathbf{x};\boldsymbol{\xi})$ for the infeasible solutions.

Two decision stages. Analysing ideas of previous paragraphs, we can see that we have turned from static programs discussed in previous paper [22] to the programs having two decision stages. So, the decision $\mathbf{x}$, obtained as the solution of the first-stage program (master program) (2), is followed by the decision $\mathbf{y}(\boldsymbol{\xi})$ that solves the second-stage program (sub

program) (3). Precisely, the first-stage decision $\mathbf{x} \in C_x$ is selected before the realization $\boldsymbol{\xi}^s$ of a random parameter $\boldsymbol{\xi}$ is observed. After this information becomes available, the decision process continues with the second-stage decision $\mathbf{y}(\boldsymbol{\xi}^s) \in C_y^{\mathrm{RP}}(\mathbf{x})$ that depends on the first-stage decision $\mathbf{x}$ and also uses the observed information $\boldsymbol{\xi}^s$. Such a decision process can be described by Figure 1.

Fig.1: The sequence decision — observation — decision

We must note here that it is not obvious to mark the dependence of $\mathbf{y}$ on the decision $\mathbf{x}$ in the notation $\mathbf{y}(\boldsymbol{\xi})$ explicitly, in opposite to the dependence of $\mathbf{y}$ on $\boldsymbol{\xi}$. Following Figure 1, we may put together the first-stage program (2) and the second-stage program (3) to have a complete mathematical description of the discussed decision situation. We may start with underlying mathematical programs at the background of (2) and (3), and we have:

$$? \in \operatorname*{argmin}_{\mathbf{x},\mathbf{y}} \{ f(\mathbf{x};\boldsymbol{\xi}) + q(\mathbf{x};\mathbf{y};\boldsymbol{\xi}) \mid \mathbf{x} \in C_x, \mathbf{h}(\mathbf{g}(\mathbf{x};\boldsymbol{\xi});\mathbf{y}) \circ \mathbf{0}, \mathbf{y} \in Y \} \ , \tag{4}$$

which is an underlying mathematical program for stochastic programs with recourse. As we have mentioned above it is only syntactical description, and hence, we denoted here the action $\mathbf{y}$ without mentioning the dependence on $\boldsymbol{\xi}$ explicitly. Subsequently, we can include (3) into (2), and this results in:

$$? \in \operatorname*{argmin}_{\mathbf{x}} \{ E_\xi \{ f(\mathbf{x};\boldsymbol{\xi}) + \min_{\mathbf{y}(\xi)} q(\mathbf{x};\mathbf{y}(\boldsymbol{\xi});\boldsymbol{\xi}) \} \mid \mathbf{x} \in C_x, \mathbf{h}(\mathbf{g}(\mathbf{x};\boldsymbol{\xi});\mathbf{y}(\boldsymbol{\xi})) \circ \mathbf{0}, \mathbf{y}(\boldsymbol{\xi}) \in Y, \text{ a.s.} \} \ . \tag{5}$$

We see that the second-stage objective was involved in the first-stage objective by substitution, and constraints were simply applied together on variables of both stages. Formulation (5), in comparison with (3), reflects the random nature of $\boldsymbol{\xi}$, and it accepts decisions that are almost surely feasible.

Two stages and recourse. The paper deals with two-stage programs that have been named in various ways by different authors (e.g., two-stage programs, programs with recourse, two-stage programs with recourse). We present our terminological viewpoint motivated by modelling needs that tries to bring more clarity and better understanding of these concepts. We have started with introducing the second-stage decision as a decision required only in the case of recourse necessity. Therefore, this decision is interpreted as an updating activity, which brings additional recourse costs. It may happen that for certain decision $\mathbf{x}$, constraints involving randomness are satisfied after getting an observation $\boldsymbol{\xi}^s$ of $\boldsymbol{\xi}$ without any need of the recourse action. When no recourse action $\mathbf{y}(\boldsymbol{\xi}^s)$ is required, no recourse cost $Q(\mathbf{x};\boldsymbol{\xi}^s)$ is paid. It seems that the suitable model in this case can mix one-stage and two-stage structures. Because we prefer a fixed decision structure, we model no recourse action as the second-stage decision $y(\boldsymbol{\xi}^s) = \mathbf{0}$, which is feasible ($\mathbf{h}(\mathbf{g}(\mathbf{x};\boldsymbol{\xi});\mathbf{0}) \circ \mathbf{0}$ a.s.), and we suppose that the nonnegative penalty function q satisfies $q(\mathbf{x};\mathbf{0};\boldsymbol{\xi}) = 0$. This modelling approach will be called stochastic programming with recourse. If the second-stage decision cannot be abandoned because it represents not only a recourse action related to the first-stage constraints including randomness, but it is also the action required by the second-stage

constraints themselves, we will talk about two-stage stochastic programming in general, and we will analyse it in the following paragraphs. There are practical examples when the second-stage decision is composed of two parts. The first one is a necessary stage-required action, the next one is an occasional randomness-related recourse. Such programs will be called two-stage stochastic programs with recourse.

2. General programs and basic concepts

Static and two-stage programs. At first, we return to formulas (2)–(5), and remembering previously introduced two-stage structure, we discuss two-stage underlying program, as follows:

$$? \in \underset{\mathbf{x}}{\operatorname{argmin}}\{f_1(\mathbf{x};\boldsymbol{\xi}) + Q_1(\mathbf{x};\boldsymbol{\xi}) \mid \mathbf{x} \in C_x(\boldsymbol{\xi})\} \,. \tag{6}$$

In addition to (4), we accept that $\boldsymbol{\xi}$ may influence both C_x and f_1. However, the independence of $\boldsymbol{\xi}$ on the first-stage decision $\mathbf{x}$ is assumed. In comparison with (1), we have $f(\mathbf{x};\boldsymbol{\xi}) = f_1(\mathbf{x};\boldsymbol{\xi}) + Q_1(\mathbf{x};\boldsymbol{\xi})$ (index 1 emphasizes the first-stage relation) and $C(\boldsymbol{\xi}) = C_x(\boldsymbol{\xi})$. Therefore, all choices of deterministic reformulations from the paper [22] published in 2010 may also be used for (6), together with introduced abbreviations. Specifically, HN-programs with EO-objective function will be considered (see $\min_x\{E_\xi\{f_1(\mathbf{x};\boldsymbol{\xi}) + Q_1(\mathbf{x};\boldsymbol{\xi}) \mid \mathbf{x} \in C_x(\boldsymbol{\xi})$ a.s.$\}$). We must add that the idea to use the expectation for the second stage is useful in the case when we deal with long series of similar decisions, and we want to use the two-stage model repeatedly. When we have to work with a unique decision, other modelling approaches used in paper [22] may be more suitable. We stress that $Q(\mathbf{x};\boldsymbol{\xi})$ values are defined implicitly as solutions of second-stage programs (3). Hence, the structure of (3) has to affect the properties of $Q(\mathbf{x};\boldsymbol{\xi})$, and this fact motivates further investigation of $Q(\mathbf{x};\boldsymbol{\xi})$ and $\mathcal{Q}(\mathbf{x})$ properties.

Deterministic reformulations and their properties. Answers to many theoretical questions may help in model building with formulation of a suitable deterministic reformulation and in model solving with a choice of a proper algorithm. We shortly revise several useful theoretical properties, and they will be utilized with algorithms later. At first, measurability is assumed for all considered functions depending on random elements, and questions about measurability of derived functions must be studied. Usually, the assumption of existence and finiteness of first and second moments of random elements is the initial step to analysis of existence and finiteness of the objective function value and optimal solution (see Wets [34] for overview). The existence and type of optimum often depend on continuity properties, and convexity properties either with respect to $\mathbf{x}$ or to $\boldsymbol{\xi}$. Differentiability is required when Karush-Kuhn-Tucker like optimality conditions are derived in [34], and it also helps with acceleration of algorithm convergence, because gradient-based computations may be involved. Program solvability is often tightly related to the properties as linearity, separability, data sparsity, and presence of special structures. Results in theory of stability are useful for program approximation and postprocessing. Therefore, the successful choice of deterministic reformulation is related to the truthful description of the studied problem, as well as, to discussed properties.

Generalized recourse. Two-stage stochastic programs may be formulated in a quite general manner. For general modelling purposes, we may introduce recourse that is more general than the additive recourse. In addition, a different characteristic in comparison with the

expectation may be also considered (cf. [34]):

$$\begin{gathered} ? \in \underset{\mathbf{x}}{\operatorname{argmin}}\{\mathcal{E}_\xi\{f(\mathbf{x};\boldsymbol{\xi})\} \mid f(\mathbf{x};\boldsymbol{\xi}) = f_1(f_{11}(\mathbf{x};\boldsymbol{\xi}), Q_1(\mathbf{x};\boldsymbol{\xi}))\ , \\ \mathbf{x} \in C_x(\boldsymbol{\xi}) = \{\mathbf{x} \in \mathbb{R}^{n_1} \mid \mathbf{g}_1(\mathbf{x};\boldsymbol{\xi}) \circ \mathbf{0}\} \text{ a.s.}\} \\ \text{where} \\ Q_1(\mathbf{x};\boldsymbol{\xi}) = \inf_{\mathbf{y}(\xi)}\{q(\mathbf{x},\mathbf{y}(\boldsymbol{\xi});\boldsymbol{\xi}) \mid \mathbf{y}(\boldsymbol{\xi}) \in C_y(\mathbf{x};\boldsymbol{\xi}) = \{\mathbf{y}(\boldsymbol{\xi}) \in \mathbb{R}^{n_2} \mid \mathbf{g}_2(\mathbf{x};\mathbf{y}(\boldsymbol{\xi});\boldsymbol{\xi}) \circ \mathbf{0} \text{ a.s.}\}\}\ . \end{gathered} \tag{7}$$

In (7), we use the rule that we add stage indicating indices (see f_{11}) everytime it is necessary. We accept that $f(\mathbf{x};\boldsymbol{\xi})$ and $\mathcal{E}_\xi\{f(\mathbf{x};\boldsymbol{\xi})\}$ may take the value $+\infty$ (see inf in the second stage). This assumes the presence of hidden (so called induced) constraints applied to $\mathbf{x}$. Similarly as in [34], we may denote the set of induced constraints as $K = \{\mathbf{x} \in C_x \mid \mathcal{Q}(\mathbf{x}) < +\infty\}$. If $K = \mathbb{R}^{n_1}$ then we talk about complete recourse, and $K \subset C_x$ identifies *relatively complete recourse* (see Wets [34]). *Simple recourse* refers to the case when constraints of the second-stage program uniquely determine the recourse decision $\mathbf{y}(\boldsymbol{\xi})$ for all $\mathbf{x}$ and $\boldsymbol{\xi}$ (see Wets [34]).

Deterministic reformulation with explicit nonlinear recourse. Kall and Wallace [11] discuss properties of the following deterministic reformulation with nonlinear recourse:

$$\begin{gathered} ? \in \underset{\mathbf{x}}{\operatorname{argmin}}\{E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\} \mid \mathbf{x} \in C_x\}\ , \\ f(\mathbf{x};\boldsymbol{\xi}) = f_1(\mathbf{x};\boldsymbol{\xi}) + \min_{\mathbf{y}(\xi)}\{q(\mathbf{y}(\boldsymbol{\xi})) \mid \mathbf{h}_2(\mathbf{y}(\boldsymbol{\xi})) \geq \mathbf{g}_2^+(\mathbf{x};\boldsymbol{\xi}), \mathbf{y}(\boldsymbol{\xi}) \in Y \text{ a.s.}\}\ , \end{gathered} \tag{8}$$

where $\mathbf{h}_2 : \mathbb{R}^{n_2} \longrightarrow \mathbb{R}^{m_2}$, $\mathbf{g}_2 : \mathbb{R}^{n_1} \times \Xi \longrightarrow \mathbb{R}^{m_2}$. If $\mathbf{g}_2(\mathbf{x};\boldsymbol{\xi}) \leq \mathbf{0}$ then $\mathbf{g}_2^+(\mathbf{x};\boldsymbol{\xi}) = 0$, otherwise $\mathbf{g}_2^+(\mathbf{x};\boldsymbol{\xi}) = \mathbf{g}_2(\mathbf{x};\boldsymbol{\xi})$. Convexity and differentiability conditions for (8) are given in [11]. Birge and Louveaux [1] summarize theoretical results for similar two-stage stochastic nonlinear programs with additive form of the recourse:

$$\begin{gathered} ? \in \underset{\mathbf{x}}{\operatorname{argmin}}\{f(\mathbf{x}) + \mathcal{Q}(\mathbf{x}) \mid \mathbf{x} \in C_x\}\ , \\ \mathcal{Q}(\mathbf{x}) = E_\xi\{Q(\mathbf{x};\boldsymbol{\xi})\} = E_\xi\left\{\min_{\mathbf{y}(\xi)}\{f_2(\mathbf{y}(\boldsymbol{\xi});\boldsymbol{\xi}) \mid \mathbf{h}_2(\mathbf{y}(\boldsymbol{\xi});\boldsymbol{\xi}) \circ \mathbf{g}_2(\mathbf{x};\boldsymbol{\xi}), \mathbf{y}(\boldsymbol{\xi}) \in Y \text{ a.s.}\}\right\}\ . \end{gathered} \tag{9}$$

They introduce measurability conditions for $Q(\mathbf{x};\boldsymbol{\xi})$ and discuss convexity of $Q(\mathbf{x};\boldsymbol{\xi})$ and $\mathcal{Q}(\mathbf{x})$, together with continuity and differentiability conditions.

3. Scenario-based programs

Discrete random variable. If $\boldsymbol{\xi}$ is a random element with a finite discrete distribution ($|\Xi| = S < \aleph_0$), we denote $\forall s \in \mathcal{S} : \mathbf{y}_s = \mathbf{y}(\boldsymbol{\xi}^s)$, $p_s = P(\boldsymbol{\xi} = \boldsymbol{\xi}^s) \geq 0$, $\sum_{s=1}^S p_s = 1$. In this case, the expectation E is expressed explicitly $E_\xi\{Q(\mathbf{x};\boldsymbol{\xi})\} = \sum_{s=1}^S p_s Q(\mathbf{x};\boldsymbol{\xi}^s)$, and we can easily delete scenarios with zero probability. So, by computational purposes, we prefer to deal with the case of discrete random variable $\boldsymbol{\xi}$ with a finite support. This case is also called a scenario-based (SB) approach to two-stage stochastic programs. For nonlinear underlying program (4), we receive:

$$\begin{gathered} ? \in \underset{\mathbf{x}}{\operatorname{argmin}}\left\{f(\mathbf{x}) + \sum_{s=1}^S p_s Q(\mathbf{x};\boldsymbol{\xi}^s) \mid \mathbf{x} \in C_x\right\}\ , \\ Q(\mathbf{x};\boldsymbol{\xi}^s) = \min_{\mathbf{y}_s}\{q(\mathbf{x};\mathbf{y}_s;\boldsymbol{\xi}^s) \mid \mathbf{y}_s \in C_y(\mathbf{x};\boldsymbol{\xi}^s)\}\ . \end{gathered} \tag{10}$$

Scenario tree. Structure of program (10) may be graphically represented with a so called scenario tree, as Figure 2 shows.

The node 1 in the top level represents the first-stage program, arcs denote realizations $\boldsymbol{\xi}^s$, and nodes in the bottom level correspond to the second-stage programs.

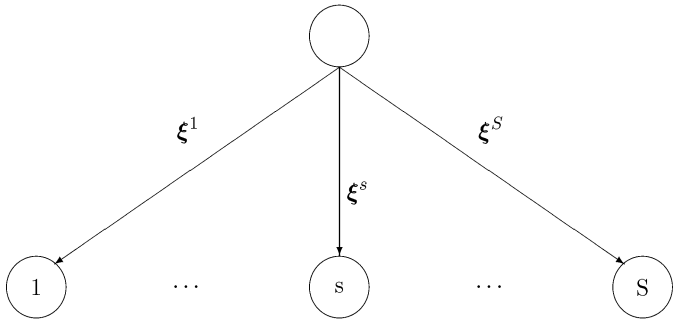

Fig.2: Scenario tree for two-stage program

Large-scale programs. We may write (10) as nested programs. Because of discrete distribution, we may exchange the order of E_ξ and $\min_{\mathbf{y}_s}$ in (10). Then, we have for nonlinear case:

$$? \in \underset{\mathbf{x},\mathbf{y}_s : s\in\mathcal{S}}{\operatorname{argmin}} \left\{ f(\mathbf{x}) + \sum_{s=1}^{S} p_s q(\mathbf{x};\mathbf{y}_s;\boldsymbol{\xi}^s) \mid \mathbf{x}\in C_x, \mathbf{y}_s \in C_y(\mathbf{x};\boldsymbol{\xi}^s), s\in\mathcal{S} \right\} . \qquad (11)$$

The programs like (11) are specifically structured large-scale deterministic programs. They are frequently employed, because they may also be obtained by discretization or sampling in the case of continuous random variable. The size of program (11) can be very large. It grows exponentially with the size of support. As usual, we denote the support set for $h_i(\boldsymbol{\xi})$ as Ξ_i, and m_2 is a number of second-stage constraints in the underlying mathematical program. The first-stage program has m_1 constraints, the second-stage problem has $m = \prod_{i=1}^{m_2} |\Xi_i|$ constraints for independent random components of $\mathbf{h}(\boldsymbol{\xi})$.

Scenarios. For solution of (11), we need to know scenarios $\boldsymbol{\xi}^s$ with their probabilities p_s. They may be found in different ways and we list several possibilities here: (1) If we know a distribution of $\boldsymbol{\xi}$: (a) and Ξ is finite and small, we take all $\boldsymbol{\xi}^s$ as scenarios for (11). (b) and Ξ is large, we choose several scenarios: (I) by an expert opinion about their importance, or (II) by a representative discretization, or (III) by sampling. (2) If we do not have complete information about distribution of $\boldsymbol{\xi}$: (a) then we may utilize the worst-case analysis; (b) then we may try to find out more information. There are following possibilities: (I) analysis of observations and historical data (usually rare events cannot be discovered), (II) hypothesis verification by experiments and statistical methods, (III) consultation of an expert's opinion. Solvability of scenario-based (SB) programs is related to their size, and so, to the number of scenarios. Such SB program is then either formulated directly in this SB form, or it is obtained by discretization, or discrete values are received by sampling. If the manageable size of scenario set was defined by Wets in 1988 (see [34]) as less then 10 000 scenarios (for a small program corresponding each considered scenario), during the last 25 years this number increased to tens of millions.

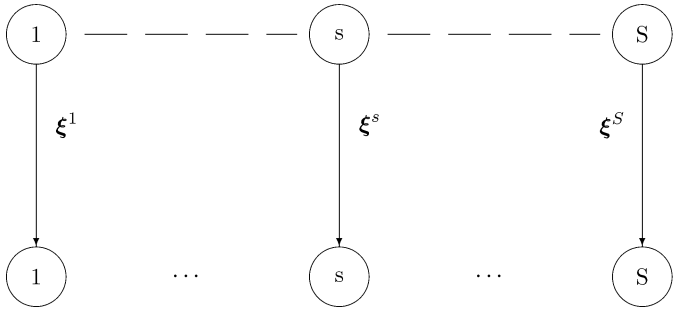

Fig.3: Scenarios with nonanticipativity constraints

Explicit nonanticipativity constraints. As we know, programs of type (10) also embrace the requirement of nonanticipativity. The following formulation is the reformulation to (10), and it expresses the requirement of nonanticipativity explicitly with nonanticipativity constraints. They are often used in algorithm development, see the PHA in the next sections. For nonlinear case, we have:

$$? \in \underset{\mathbf{x}_s, \mathbf{y}_s : s \in \mathcal{S}}{\operatorname{argmin}} \left\{ \sum_{s=1}^{S} p_s (f(\mathbf{x}_s) + q(\mathbf{x}_s; \mathbf{y}_s; \boldsymbol{\xi}^s)) \mid \mathbf{x}_s \in C_x, \mathbf{y}_s \in C_y(\mathbf{x}_s; \boldsymbol{\xi}^s), s \in \mathcal{S}, \forall r, u \in \mathcal{S} : \mathbf{x}_r = \mathbf{x}_u \right\}. \tag{12}$$

There are many redundant nonanticipativity constraints $\forall r, u \in \mathcal{S} : \mathbf{x}_r = \mathbf{x}_u$ in (12). We may reduce their number in different ways. Very often r is fixed to 1, and $u \in \mathcal{S}$ ('star form'). Another possibility is to use only circular references: $\forall s \in \mathcal{S} \setminus \{S\} : \mathbf{x}_s = \mathbf{x}_{s+1}$. General suggestions of how to use permutations and matrix $\mathbf{U}$ composed of zeroes and ones may be found, e.g., in [19]. This larger (instead of one $\mathbf{x}$ variable, we have S variables $\mathbf{x}_s$) but easy decomposable formulation is used in the algorithms based on the constraints relaxation. We may describe this situation graphically with Figure 3, where dash lines identify nonanticipativity constraints.

4. Special cases

The first-stage randomness. We already noticed in (6) that randomness may also influence the first-stage program. We may deal with this first-stage randomness in various ways. If its realization cannot be observed before taking decision $\mathbf{x}$, we add randomness to $\boldsymbol{\xi}$ and use HN approach. The situation is more complicated when random parameters are also involved in the first-stage constraints. Then, a suitable deterministic reformulation from paper published in 2010 [22] has to be selected. If realizations of the first-stage randomness are at least partly observed, then we may apply the combined objective. If our observations are complete, we use WS approach. It is typical for certain metallurgical problems that observed information is incomplete or available later.

Combined constraints and objectives. Although it is not typical, two-stage programs can also be enriched with probabilistic terms. First analytical attempts were realized by Charnes and Kirby. The influence of probabilistic constraints on two-stage stochastic program properties and solution techniques was analysed by Wets. An interesting idea was presented

by Prekopa [23] who discussed the possibilities of C_x modification with relaxed induced constraints: $\{\mathbf{x} \in \mathbb{R}^{n_1} \mid P(Q(\mathbf{x};\boldsymbol{\xi}) < +\infty) \geq \alpha\}$.

Quadratic recourse. Many program changes are algorithmically motivated. For example, Birge, Pollock, and Qi present a quadratic recourse representation of the two-stage stochastic linear program. They show how their differentiable (with respect to $\mathbf{x}$) quadratic recourse function approximates the linear recourse function, and they solve this approximation of the original program with high convergence rate algorithm.

Partial information available. Sometimes, only *partial information* about realized randomness is available. For instance, this situation occurs when randomness also influences a modelled problem after taking a second-stage decision. Working with melt control problems, we can expect that after the last alloying, a small influence of random losses may change the final composition of the melt. Among approaches presented in this paragraph, the discussed situation inspires the question of whether we may add another recourse action after this new observation. This question motivates further multistage stochastic program use. Various combinations of HN and WS approaches (or adaptive and anticipative models) can be utilized in this situation. Wets [34] developed a general approach. He represents the available information by a subfield $\mathcal{B}$ of field of all events Σ for static case as follows:

$$? \in \underset{\mathbf{x}(\xi)}{\operatorname{argmin}}\{E_\xi\{f(\mathbf{x}(\boldsymbol{\xi});\boldsymbol{\xi}) \mid \mathcal{B}\} \mid E_\xi\{\mathbf{g}(\mathbf{x}(\boldsymbol{\xi});\boldsymbol{\xi}) \mid \mathcal{B}\} \circ \mathbf{0}, \mathbf{x}(\boldsymbol{\xi}) \in X \text{ a.s.}\} \,. \tag{13}$$

Decisions $\mathbf{x}(\boldsymbol{\xi})$ are $\mathcal{B}$ measurable, and $E_\xi\{. \mid \mathcal{B}\}$ is a conditional expectation with respect to $\mathcal{B}$. The decision $\mathbf{x}(\boldsymbol{\xi})$ is WS for $\mathcal{B} = \Sigma$ and HN for $\mathcal{B} = \{\emptyset, \Xi\}$. The choice in between these extremal possibilities models partial information availability. Because computations are complicated even for simple $\mathcal{B}$, Wets suggests the parametric approach in [34]. Then, two-stage stochastic programs with only partial information about $\boldsymbol{\xi}$ after the first-stage decision may use the idea of (13) for the second-stage program formulation.

Decision dependent randomness. In the previous paragraph, we discussed the case of partial information available about realizations of $\boldsymbol{\xi}$. A more general situation arises when information about distribution is also incomplete. Related problems were studied, especially by Dupačová. In certain situations, information about distribution is incomplete, because distribution of $\boldsymbol{\xi}$ (or $\mathcal{B}$, see Varaiya and Wets [31]) may change with decision $\mathbf{x}$. This situation may occur, e.g., when the first-stage big investment decision changes market expectations, or when certain input alloy properties qualitatively influence melt losses. These examples show that the usual requirement to have independent $\mathbf{x}$ and $\boldsymbol{\xi}$ may be quite restrictive in practical applications. This decision dependent randomness can be described in many ways: by verbal description, by distribution selection rule, by special constraints, by mapping connecting $\mathbf{x}$ values with distributions $\mathcal{P} \in \Pi$, by change of distribution parameters with change of $\mathbf{x}$, and by functional dependence $\boldsymbol{\xi}$ on $\mathbf{x}$. A similar situation also arises in (13) when $\mathcal{B}$ depends nonlinearly on $\mathbf{x}$, as was discussed in [31]. Discussion about decision dependent randomness started with the Varaiya and Wets article [31] when they classified two main problems with two-stage programs (distribution and information availability dependence). Jonsbråten, Wets, and Woodruff [9] present new research results. They identify class of programs that are manageable by their approach, propose an algorithm, and discuss relations to stochastic integer programming. They concentrated on programs where distributions and variables controlling information availability are discrete. There is also area of stochastic

programs designed for logistics involving pricing mechanisms (cf. [17] and [8] for the recent development) that can inspire modelling of various structural features of engineering problems when the design variables influence the probability distribution.

The integer first stage. Two-stage stochastic linear programs *with integer variables* is usually much harder to solve. An overview of recent research results is presented by Klein Haneveld, Stougie, and van der Vlerk in [6]. Complexity of integer programs was initially studied by Stougie [30]. For fixed recourse and integer variables only in the first stage, we obtain an integer program having a convex objective term $\mathcal{Q}(\mathbf{x})$. In this case, a combination of the L-shaped method with branch-and-bound algorithm was developed. However, for the large scale problems (see [8] and [7]) based on real world data, heuristics can help, see [26] and [13].

The integer second stage. Even more difficult problems arise when *integer variables are contained in the second stage program*. Especially, $\mathcal{Q}(\mathbf{x})$ convexity is lost. Theoretical results about complete mixed-integer recourse were derived by Schultz, e.g., [28]. Carõe and Tind [2] consider the L-shaped decomposition for two-stage stochastic integer programs in general. A challenge is to find a substitute for the recourse problem that would not require the second stage variables to be integer. More specifically, we want to replace $\mathcal{Q}(\mathbf{x})$ by other function using a convex hull. There are promising results by Stougie, Klein Haneveld and Van der Vlerk for stochastic programs with simple recourse.

Network structure. The case of two-stage stochastic program involving a network structure has been studied by Wallace and his collaborators. Feasibility conditions are studied, together with bounds, and special algorithms. A reader will find the explanatory description and insightful discussion of related problems in the textbook written by Kall and Wallace [11].

5. Solution techniques

Models and solutions. Up to now, we have discussed modelling questions related to two-stage programs. We noticed classical theoretical results that were achieved in the initial research period when high performance computers could not help with algorithmic computations. They bring important insights and prepare conditions for an algorithm use. However, built models, often motivated by practical problems, are necessary to solve. Therefore, solution approaches are considered in the rest of this text. Anticipating possible questions related to solution approaches, we may list following problems:

- description of input/output data structures,
- development of the formal language for algorithm description,
- a choice of stopping criteria,
- analysis of algorithm convergence conditions (necessary and sufficient) and properties (degeneracy, cycling, stalling, etc.),
- derivation of theoretical convergence rate and its comparison with practical experience,
- estimation of memory and operational requirements subject to program size.

However, we will concentrate on ideas and common aspects, so all these questions will not be systematically presented. An interested reader will find complete information in references.

Transformations and algorithms. Algorithm use is often preceded by certain transformation of the original program. This transformation changes the original program into a reformulation transformed one, easily solvable. The program transformation is often based on syntactical manipulations. Hence, consideration of possible transformation errors is not usually needed. Different algorithms have been proposed for solving stochastic programs of various types. At first, we may obtain an explicit mathematical program when $\mathcal{Q}(\mathbf{x})$ is computed explicitly. The resulting program is either a nonlinear program having special properties or a large-scale nonlinear program. So, modified traditional algorithms are employed. Mathematical programming algorithms were successfully used in special cases (see Nazareth and Wets [16]). If dual block-angular structure of a linear program is detected then decomposition algorithms as the L-shaped method and regularized decomposition may be utilized. Other decomposition approaches allowing nonlinearities are related to the augmented Lagrangian use. As usual, there is a difference between algorithms developed for deterministic reformulations, which can be transformed in the explicit mathematical programs, and algorithms working with the original implicit formulation of deterministic reformulations. The latter method frequently calls for internal sampling procedures (see a stochastic quasigradient method).

Approximations and modifications. If any suitable transformation was not found and algorithms cannot be directly used, then a program approximation may help. Theoretical basement for approximation techniques is derived from the idea that under certain assumptions any approximation of the objective function $E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\}$ that is tight enough $\forall \mathbf{x} \in C_x$ guarantees the similar quality for the approximation of the optimal value $z_{\min}^{\mathrm{EO}}$ and the optimal solution $\mathbf{x}_{\min}^{\mathrm{EO}}$. We may consider approximations of two types. They are based, at first, on the external sampling of $\boldsymbol{\xi}$. A small-enough scenario tree is constructed and the approximating program is solved with one of the aforementioned algorithms. Error estimates can be obtained by statistical procedures. Deterministic approximation schemes were developed as the alternative to external sampling methods. They build a framework for iterative application of previous algorithms to an updated program. As with approximations in numerical mathematics, convergence is studied. Errors of approximations are constructed with special bounds. Sometimes, subjective (robusting) or additional (preprocessing, postprocessing) techniques may be utilized to obtain the original problem solution. Although they have various origins, contents, and purposes, we will call them together modifications.

Mathematical programming algorithms. Nazareth and Wets present a broad overview [16] of mathematical programming methods in stochastic programming. The main disadvantage of many mathematical programming algorithms is related to the assumption that gradients of all functions describing programs are easily available, because this is not the general case in stochastic programming. So, these algorithms are frequently used for the solution of specific programs (as aforementioned programs with simple recourse).

Basic Lagrangian-based algorithms. One unpleasant difficulty with discrete random parameters $\boldsymbol{\xi}$ is that $\mathcal{Q}(\mathbf{x})$ loses differentiability. Therefore, the L-shaped related methods are useful, because they require only cutting planes – the subgradient based information. However, these methods were motivated by linear programs, and they were extended to cover certain nonlinear problems. In contrast, Lagrangian-based algorithms are motivated by nonlinear programming theory and may be more flexible. The basic idea of these algorithms

is that scenario-related programs are linked only by nonanticipativity constraints. If these constraints are relaxed, then the solution algorithm is reduced to the solution of separate programs for each scenario. For the scenario-based nonlinear program (10), we may write the dual program:

$$? \operatorname*{argmax}_{\pi} \vartheta(\boldsymbol{\pi}) \tag{14}$$

$$\vartheta(\boldsymbol{\pi}) = \inf_{\mathbf{x}_s, \mathbf{y}_s : s \in \mathcal{S}} \left\{ \sum_{s=1}^{S} p_s \left(f(\mathbf{x}_s) + q(\mathbf{x}_s; \mathbf{y}_s; \boldsymbol{\xi}^s) + \boldsymbol{\pi}_s^{\mathrm{T}} \left(\mathbf{x}_s - \sum_{l=1}^{S} p_l \mathbf{x}_l \right) \right) \mid \right.$$
$$\left. \mathbf{x}_s \in C_x, \mathbf{y}_s \in C_y(\mathbf{x}_s; \boldsymbol{\xi}^s), s \in \mathcal{S} \right\} . \tag{15}$$

Gradient method for this dual (Lagrangian Dual Ascent Method (LDAM)) can be formed as follows (see [1]) :

Algorithm 1 (LDAM) :

1. Set $\boldsymbol{\pi}[0] := (\boldsymbol{\pi}_1[0]^{\mathrm{T}}, \ldots, \boldsymbol{\pi}_S[0]^{\mathrm{T}})^{\mathrm{T}}$ and $n := 0$.
2. Fix $\boldsymbol{\pi} := \boldsymbol{\pi}[n]$ in (15) and find the optimal solution $(\mathbf{x}[n]^{\mathrm{T}}, \mathbf{y}[n]^{\mathrm{T}})^{\mathrm{T}}$.
3. If $\forall s \in \mathcal{S} : \mathbf{x}_s[n] = \sum_{l=1}^{S} p_l \, \mathbf{x}_l[n]$, then STOP, and the overall optimal solution is found. Otherwise, set $\hat{\boldsymbol{\pi}}_s[n] := \mathbf{x}_s[n] - \sum_{l=1}^{S} \mathbf{x}_l[n]$ and form $\hat{\boldsymbol{\pi}}[n]$.
4. Let $\lambda[n] \in \operatorname{argmin}_{\lambda} \{ \vartheta(\boldsymbol{\pi}[n] + \lambda \, \hat{\boldsymbol{\pi}}[n]) \mid \boldsymbol{\pi}[n] + \lambda \, \hat{\boldsymbol{\pi}}[n] \geq \mathbf{0}, \lambda \geq 0 \}$. Then, $\boldsymbol{\pi}[n+1] := \boldsymbol{\pi}[n] + \lambda[n] \hat{\boldsymbol{\pi}}[n]$, $n := n + 1$, and GOTO 2 .

At first, we must notice that nonanticipativity constraints have the form of $\mathbf{x}(\boldsymbol{\xi}^s) = E_{\xi}\{\mathbf{x}(\boldsymbol{\xi})\}$. Under convexity assumptions, this algorithm converges to an optimal solution. It may require less iterations than the primal algorithm. Although nonanticipativity constraints are included as 'soft constraints', actual experience has shown that the practical convergence is slow. Other techniques to achieve faster convergence are based on adding a quadratic term to the objective. In this way, we obtain the augmented Lagrangian methods that utilize convexity properties and nonsingular Hessian to achieve superlinear convergence. There are reports on the computational experience with the reformulation of ϑ:

$$\vartheta(\boldsymbol{\pi}) = \inf_{\mathbf{x}, \mathbf{y}} \left\{ f(\mathbf{x}_0) + \sum_{s=1}^{S} p_s \left(q(\mathbf{x}_s, \mathbf{y}_s; \boldsymbol{\xi}^s) + \boldsymbol{\pi}_s^{\mathrm{T}} (\mathbf{x}_s - \mathbf{x}_0) + \frac{\rho}{2} \, ||\mathbf{x}_s - \mathbf{x}_0||^2 \right) \right\} .$$

The algorithm switches between solution spaces of $\mathbf{x}_0, \boldsymbol{\pi}$ and separate spaces $\mathbf{x}_s, \mathbf{y}_s$.

Progressive hedging algorithm. Wets presents [35] a comparison of the scenario aggregation with scenario analysis often used by practitioners (we recall that scenario analysis utilizes $E_{\xi}\{\mathbf{x}(\boldsymbol{\xi})\}$ as the HN-decision). The ideas of scenario aggregation are included into the algorithm. The *progressive hedging algorithm* (PHA) is proposed by Rockafellar and Wets [25]. It is aimed at solving scenario-based two-stage stochastic linear and nonlinear programs. This algorithm is also useful in cases when certain important scenarios must be taken into account and we need a hedging optimal solution. During the iteration, all scenario programs are solved, and obtained solutions are averaged to have the outer approximation of found solution. The objective of each scenario is updated using the input information about nonanticipativity constraints and new information about iterations' results. The sequence

of $\hat{\mathbf{x}}[n]$ converges to the optimal solution for the convex case, and its authors report algorithm convergence also for certain nonconvex cases. The convergence of this method is based on Rockafellar's proximal point method. The progressive hedging algorithm has the following form:

Algorithm 2 (PHA):

1. Initialization: $n := 1$, $\forall s \in \mathcal{S} : \mathbf{w}_s[0] := \mathbf{0}$, and choose $\hat{\mathbf{x}}[0]$, $\rho > 0$.

2. $\forall s \in \mathcal{S}$ find the solution $\mathbf{x}_s[n], \mathbf{y}_s[n]$ of the program:

$$? \in \underset{\mathbf{x}_s,\mathbf{y}_s}{\operatorname{argmin}} \Big\{ f(\mathbf{x}_s) + q(\mathbf{x}_s, \mathbf{y}_s; \boldsymbol{\xi}^s) + \mathbf{w}_s[n-1]^{\mathrm{T}} \mathbf{x}_s + \frac{\rho}{2} ||\mathbf{x}_s - \hat{\mathbf{x}}[n-1]||^2 \mid \mathbf{x}_s \in C_x, \mathbf{y}_s \in C(\mathbf{x}_s; \boldsymbol{\xi}^s) \Big\} . \tag{16}$$

3. Compute the new average solution: $\hat{\mathbf{x}}[n] := \sum_{s\in\mathcal{S}} p_s \, \mathbf{x}_s[n]$. Update perturbation terms $\forall s \in \mathcal{S} : \mathbf{w}_s[n] := \mathbf{w}_s[n-1] + \rho(\mathbf{x}_s[n] - \hat{\mathbf{x}}[n])$. If $\sum_{s\in\mathcal{S}} p_s \, \mathbf{w}_s[n] = \mathbf{0}$, then STOP, otherwise $n := n + 1$ and GOTO 2 .

The main advantage for programming purposes is that PHA uses locally convergent nonlinear programming algorithms having many available well-tested implementations. The updating step is then quite simple to be realized. In addition, solution averages guarantee robustness of computational process, but the convergence is slow. Helgason and Wallace [32] discuss PHA properties and conclude that it is not necessary to solve the scenario programs to optimality at early steps of the progressive hedging algorithm, because solving each scenario program to optimality can be very inefficient.

Diagonal quadratic approximation algorithm. Mulvey and Ruszczyński [15] have developed a variant of the augmented Lagrangian method called *Diagonal Quadratic Approximation* (DQA). Their approach is based on writing the nonanticipativity constraints with a permutation cyclic order (the bijection $\sigma : \mathcal{S} \longrightarrow \mathcal{S}$ defines a permutation, the inverse relation is denoted σ^{-1}). Then, they approximate the objective augmented terms $||\mathbf{x}_s - \mathbf{x}_{\sigma(s)}||^2$ with substitution of the current $\hat{\mathbf{x}}_{\sigma(s)}[n]$. Then the original augmented Lagrangian program is decomposed into S sub programs:

$$? \in \underset{\mathbf{x}_s,\mathbf{y}_s}{\operatorname{argmin}} \Big\{ f(\mathbf{x}_s) + q(\mathbf{x}_s, \mathbf{y}_s, \boldsymbol{\xi}^s) + (\boldsymbol{\pi}_s - \boldsymbol{\pi}_{\sigma^{-1}(s)})^{\mathrm{T}} \mathbf{x} + \frac{\rho}{2}(||\mathbf{x}_s - \hat{\mathbf{x}}_{\sigma(s)}||^2 + ||\mathbf{x}_s - \hat{\mathbf{x}}_{\sigma^{-1}(s)}||^2) \mid \mathbf{x}_s \in C_x, \mathbf{y}_s \in C_y(\mathbf{x}_s, \boldsymbol{\xi}^s) \Big\} . \tag{17}$$

The DQA method is completely generalizable to nonlinear problems, and it may be implemented in parallel.

6. Sampling algorithms

Motivation. Until now, presented algorithms were designed for scenario-based stochastic programs and for programs with explicitly computed expectations. If a distribution of $\boldsymbol{\xi}$ is continuous or discrete with a large support, then some approximation technique must be employed. At first, we may utilize an internal sampling. This is based on the random sampling within a deterministic optimization algorithm. A general purpose stochastic quasigradient

technique may be found in [4], [11] and [23]. However, there are also other stochastic algorithms that may take advantages from the special structure of stochastic program. We may involve sampling into various algorithms. At the beginning of this section, we mention the algorithm SSMO (Successive Sample Mean Optimization) that is a common ancestor of several proposed methods, because it optimizes a sample mean instead of the expectation.

Algorithm 3 (SSMO):

1. $n := 0$ and $\mathcal{Q}(\mathbf{x})[0] := 0$.
2. $n := n + 1$, randomly generate an observation $\boldsymbol{\xi}[n] := \boldsymbol{\xi}^n$ independent of any previously generated observations.
3. Update $\mathcal{Q}(\mathbf{x})[n] := \frac{n-1}{n}\mathcal{Q}(\mathbf{x})[n-1] + \frac{1}{n}Q(\mathbf{x}; \boldsymbol{\xi}[n])$.
4. Solve $? \in \operatorname{argmin}_{\mathbf{x}}\{\mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathcal{Q}(\mathbf{x})[n] \mid \mathbf{x} \in C_x\}$ to obtain $\mathbf{x}[n]$, and GOTO 2 .

We should notice that the termination criterion is not included.

Stochastic quasigradient. Stochastic quasigradient methods (SQG) may be also used for solution of two-stage programs (5). The SQG method is based on the repeated *choice of search directions*, so the observations of $Q(\mathbf{x}; \boldsymbol{\xi})$ are used in each iteration. Although SQG algorithms are useful for general convex programs, the choice of step-length and their termination is often solved problem dependent. The convergence of the algorithm is usually slow, the improvements were suggested by several authors. The whole class of these algorithms were developed by Ermoliev [3] and Gaivoronski [5].

Estimate. When we are faced with a large two-stage stochastic program, it is common to approximate it. We recall the notation that also covers two-stage programs (cf. (6)):

$$z_{\min}^{\mathrm{EO}} = E_\xi\{f(\mathbf{x}_{\min}^{\mathrm{EO}}; \boldsymbol{\xi})\} = \min_{\mathbf{x}}\{E_\xi\{f(\mathbf{x}; \boldsymbol{\xi})\} \mid \mathbf{x} \in C_x\} \ . \tag{18}$$

There are different ways to approximate two-stage program (18). The first one, discussed in this section is to reduce a program size with random sampling. We denote a random sample from $\boldsymbol{\xi}$ as $\boldsymbol{\xi}_{[.]} = (\boldsymbol{\xi}_{[1]}, \ldots, \boldsymbol{\xi}_{[\nu]})^{\mathrm{T}}$. There are $\boldsymbol{\xi}_{[s]}$ random variables identically distributed as $\boldsymbol{\xi}$ and they are independent. The realization of this random sample is usually denoted $\boldsymbol{\xi}^s_{[.]} = (\boldsymbol{\xi}^s_{[1]}, \ldots, \boldsymbol{\xi}^s_{[\nu]})^{\mathrm{T}}$. If it is contextually clear, then we simplify our notation as follows $\boldsymbol{\xi}^s_{[.]} = (\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^\nu)$. For computational purposes, we may easily replace $E_\xi\{f(\mathbf{x}; \boldsymbol{\xi})\}$ (and hence $E_\xi\{Q(\mathbf{x}; \boldsymbol{\xi})\}$ for two-stage programs) by the *realization of a sample mean* $\frac{1}{n}\sum_{s=1}^{\nu} f(\mathbf{x}; \boldsymbol{\xi}^s)$ (and $\frac{1}{n}\sum_{s=1}^{\nu} Q(\mathbf{x}; \boldsymbol{\xi}^s)$ for two-stage programs):

$$z_{\min}^{\nu} = \frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x}_{\min}^{\nu}; \boldsymbol{\xi}^s) = \min_{\mathbf{x}}\left\{\frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x}; \boldsymbol{\xi}^s) \mid \mathbf{x} \in C_x\right\} \ . \tag{19}$$

Randomly generated observations of $\boldsymbol{\xi}$ may then serve to compute the estimate of the objective (and recourse) function.

Estimate quality. Therefore, scenarios are not selected by any expert, but by a random procedure. Then, the scenarios $\boldsymbol{\xi}^s$ are used to build a scenario tree, and this reduced program is solved instead of the original one. However, its blindfold and exaggerated use can lead to *misleading results*. So, in addition, Monte Carlo techniques may be necessary to obtain an estimate of how good is such a simplification. Then, repeated computations for

different random samples of scenarios will inform us about the stability and sensitivity of the original results. More precisely, we see that the most obvious way how to approximate (18) with sampling is to use a sample mean as the estimate of the expected value. Then, we obtain for sample $\boldsymbol{\xi}_{[.]}$:

$$\begin{aligned}\zeta^{\nu}_{\min} = z^{\nu}_{\min}(\boldsymbol{\xi}_{[.]}) = \frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x}^{\nu}_{\min}(\boldsymbol{\xi}_{[.]}); \boldsymbol{\xi}_{[s]}) = \\ = \min_{\mathbf{x}(\boldsymbol{\xi}_{[.]})} \left\{ \frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x}(\boldsymbol{\xi}_{[.]}); \boldsymbol{\xi}_{[s]}) \mid \mathbf{x}(\boldsymbol{\xi}_{[.]}) \in C_x \text{ a.s.} \right\} ,\end{aligned} \tag{20}$$

and the quality of replacing the unknown $z^{\mathrm{EO}}_{\min}$ and $\mathbf{x}^{\mathrm{EO}}_{\min}$ (18) with $z^{\nu}_{\min}$ and $\mathbf{x}^{\nu}_{\min}$ may be evaluated by analysis of statistical properties of the random elements $\zeta^{\nu}_{\min} = z^{\nu}_{\min}(\boldsymbol{\xi}_{[.]})$ and $\mathbf{x}^{\nu}_{\min}(\boldsymbol{\xi}_{[.]})$.

Estimate consistency and asymptotic behaviour. The natural question rises of what we may say about the estimates (e.g., $\zeta^{\nu}_{\min}$) and their quality when $\nu \to \infty$. Basic description of this statistical framework for stochastic programs can show that under certain assumptions $\zeta^{\nu}_{\min}$ and $\mathbf{x}^{\nu}_{\min}(\boldsymbol{\xi}_{[.]})$ are *consistent estimates* of $z^{\mathrm{EO}}_{\min}$ and $\mathbf{x}^{\mathrm{EO}}_{\min}$. However, the asymptotic normality is achieved only under strong assumptions.

Confidence intervals and bounds. As in the previous paragraphs, we denote $\boldsymbol{\xi}_{[.]}$ a sample from $\boldsymbol{\xi}$ and we obtain the following $1-\alpha$ confidence interval for given $\mathbf{x} \in C_x$ and the HN-objective function value:

$$\begin{aligned}P\Bigg(\frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x}; \boldsymbol{\xi}_{[s]}) - \frac{t_{1-\alpha/2}\, s(\mathbf{x};\nu)}{\sqrt{\nu}} \leq \\ \leq E_{\xi}\{f(\mathbf{x};\boldsymbol{\xi})\} \leq \frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x};\boldsymbol{\xi}_{[s]}) + \frac{t_{1-\alpha/2}\, s(\mathbf{x};\nu)}{\sqrt{\nu}} \Bigg) \approx 1-\alpha ,\end{aligned} \tag{21}$$

where $t_{1-\alpha/2}$ denotes the $1-\alpha/2$ quantile of $N(0;1)$ (for small sample size ν, $1-\alpha/2$ quantile of Student's distribution with $\nu-1$ degrees of freedom may be utilized), and $s(\mathbf{x};\nu)$ denotes a usual estimate of the standard deviation $\sqrt{\operatorname{var} f(\mathbf{x};\boldsymbol{\xi})}$ using a sample $\boldsymbol{\xi}_{[.]}$. Similarly, the $1-\alpha$ confidence interval for the WS-program can be formulated:

$$\begin{aligned}P\Bigg(\frac{1}{\nu}\sum_{s=1}^{\nu} \min_{\mathbf{x}(\boldsymbol{\xi}_{[s]})} \{f(\mathbf{x}(\boldsymbol{\xi}_{[s]}); \boldsymbol{\xi}_{[s]}) \mid \mathbf{x}(\boldsymbol{\xi}_{[s]}) \in C_x \text{ a.s.}\} - \frac{t_{1-\alpha/2}\, s^{\mathrm{WS}}(\nu)}{\sqrt{\nu}} \leq \\ \leq E_{\xi}\left\{ \min_{\mathbf{x}(\xi)} \{f(\mathbf{x}(\boldsymbol{\xi});\boldsymbol{\xi}) \mid \mathbf{x}(\boldsymbol{\xi}) \in C_x \text{ a.s.}\} \right\} \leq \\ \leq \frac{1}{\nu}\sum_{s=1}^{\nu} \min_{\mathbf{x}(\boldsymbol{\xi}_{[s]})} \{f(\mathbf{x}(\boldsymbol{\xi}_{[s]}); \boldsymbol{\xi}_{[s]}) \mid \mathbf{x}(\boldsymbol{\xi}_{[s]}) \in C_x \text{ a.s.}\} + \frac{t_{1-\alpha/2}\, s^{\mathrm{WS}}(\nu)}{\sqrt{\nu}} \Bigg) \approx 1-\alpha ,\end{aligned} \tag{22}$$

where $s^{\mathrm{WS}}(\nu)$ denotes the usual estimate of the standard deviation $\sqrt{\operatorname{var} z^{\mathrm{WS}}_{\min}(\boldsymbol{\xi})}$. Then, the

following sample-based lower bound and approximate probabilistic upper bound are derived:

$$\frac{1}{\nu}\sum_{s=1}^{\nu}\min_{\mathbf{x}(\boldsymbol{\xi}_{[s]})}\{f(\mathbf{x}(\boldsymbol{\xi}_{[s]});\boldsymbol{\xi}_{[s]}) \mid \mathbf{x}(\boldsymbol{\xi}_{[s]}) \in C_x \text{ a.s.}\} - \frac{t_{1-\alpha/2}\, s^{\mathrm{WS}}(\nu)}{\sqrt{\nu}} \leq$$

$$\leq \min_{\mathbf{x}}\{E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\} \mid \mathbf{x} \in C_x\} \leq \tag{23}$$

$$\leq \min_{\mathbf{x}(\boldsymbol{\xi}_{[.]})}\left\{\frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x}(\boldsymbol{\xi}_{[.]});\boldsymbol{\xi}_{[s]}) \mid \mathbf{x}(\boldsymbol{\xi}_{[.]}) \in C_x\right\} + \frac{t_{1-\alpha/2}\, s(\mathbf{x}^{\nu}_{\min}(\boldsymbol{\xi}_{[.]});\nu)}{\sqrt{\nu}}\ .$$

Morton, Mak, and Wood prove under quite general assumptions [14] the following inequalities:

$$E_\xi\{\zeta^{\nu}_{\min}\} \leq E_\xi\{\zeta^{\nu+1}_{\min}\} \leq z^{\mathrm{EO}}_{\min} \leq z = E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\}\ , \tag{24}$$

where $\mathbf{x}$ is any feasible solution from C_x. They assume that a random sample from $\boldsymbol{\xi}$ denoted as $\boldsymbol{\xi}_{[.]} = (\boldsymbol{\xi}_{[1]}, \ldots, \boldsymbol{\xi}_{[\nu_u]})^{\mathrm{T}}$ is available. They have ν_l random samples, each having size ν, therefore $\forall i = 1, \ldots, \nu_l : \boldsymbol{\xi}_{[i.]} = (\boldsymbol{\xi}_{[i1]}, \ldots, \boldsymbol{\xi}_{[i\nu]})^{\mathrm{T}}$. They use inequalities (24) and the central limit theorem to derive the following *bounds*:

$$P\left(\frac{1}{\nu_l}\sum_{i=1}^{\nu_l}\min_{\mathbf{x}(\boldsymbol{\xi}_{[i.]})}\left\{\frac{1}{\nu}\sum_{s=1}^{\nu} f(\mathbf{x}(\boldsymbol{\xi}_{[i.]});\boldsymbol{\xi}_{[is]}) \mid \mathbf{x}(\boldsymbol{\xi}_{[i.]}) \in C_x \text{ a.s.}\right\} - \frac{t_{1-\alpha/2}\, s_l(\nu_l)}{\sqrt{\nu_l}} \leq\right.$$

$$\leq E_\xi\{\zeta^{\nu}_{\min}\} \leq z^{\mathrm{EO}}_{\min} \leq E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\} \leq \tag{25}$$

$$\left.\leq \frac{1}{\nu_u}\sum_{s=1}^{\nu_u} f(\mathbf{x};\boldsymbol{\xi}_{[s]}) + \frac{t_{1-\alpha/2}\, s_u(\nu_u)}{\sqrt{\nu_u}}\right) \approx 1-\alpha\ .$$

The symbol $t_{1-\alpha/2}$ denotes the $1-\alpha/2$ quantile of $N(0;1)$ distribution. Symbols $s_l(\nu_l)$ and $s_u(\nu_u)$ denote usual estimates of standard deviations $\sqrt{\operatorname{var}\zeta^{\nu}_{\min}}$ and $\sqrt{\operatorname{var} f(\mathbf{x};\boldsymbol{\xi})}$. Hence, we may set α, then substitute observations $\boldsymbol{\xi}^{s}_{[.]}$ and $\boldsymbol{\xi}^{s}_{[i.]}$ in the formula (25), and we obtain reliable bounds on the optimal value $z^{\mathrm{EO}}_{\min}$.

We can summarize advantages and disadvantages of sampling based approximations: the main advantages are that approximate confidence intervals are distribution free, and no assumptions about convexity of the objective function are needed. The main difficulties are that the precision of the bounds depends on the precision of the approximation by the central limit theorem and on the sample size.

7. Conclusions

Approximation schemes. Hitherto, we are able to solve two-stage stochastic programs, because we usually deal with one of the following situations:

- We are able to compute the expectation explicitly, and we transform a two-stage program into the compact solvable mathematical program.
- Having a finite support of the random variable $\boldsymbol{\xi}$, we use the effective algorithm designed for specially structured large-scale programs.
- In other cases, we may employ a random sample either internally or externally with respect to the optimization algorithm.

So, the important solution step is the program approximation motivated by the original program complexity, and solvability. There are other approximating techniques of how to

replace the original program to obtain a solution of newly introduced program that is good enough also for the approximated program. We may try to find other solution possibilities returning to the approximated program. Up to now, we have dealt with the approximation of E_ξ by the sample mean. There are also other types of the approximation in stochastic programming. We see that the program may be approximated by the approximation of: (1) the feasible set C_x, (2) the objective f, and (3) the distribution of $\boldsymbol{\xi}$.

Approximation of the feasible set. The idea of modifying the feasible set C_x is very simple. If we *relax certain constraints* (e.g., nonanticipativity constraints in PHA), we may solve the program more easily, however, we obtain a lower bound of $z_{\min}^{\mathrm{EO}}$. In an opposite way, we may *add several constraints*, and the optimal value of modified program gives the upper bound for $z_{\min}^{\mathrm{EO}}$.

Approximation of the objective. The approximation of the objective is often based on the replacement of $E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\}$ with $E_\xi\{\sum_i f_i(\mathbf{x};\xi_i)\}$. Then, this continuous approximation replaces the original high-dimensional integral with a combination of low-dimensional integrals to obtain the upper bound for the convex recourse program. Usefulness of this technique depends on the structure of $f(\mathbf{x};\boldsymbol{\xi})$. Mainly, the *interactions between various components* ξ_i of $\boldsymbol{\xi}$ may have only limited influence in the determination of the cost $E_\xi\{f(\mathbf{x};\boldsymbol{\xi})\}$ (cf. separable simple recourse).

Approximation of the distribution. The majority of approximation methods involve *discretization* of an underlying probability measure. The discretization may be chosen to provide upper or lower bounds on the objective function value. Both cases with a known probability measure or unknown distribution with certain known characteristics (such as mean and variance ranges) are considered. We may discretize $\boldsymbol{\xi}$, and we approximate the original distribution with a discrete one. It can be realized: (1) with numerical rules to achieve good approximation, (2) by the expert selection of scenarios, and (3) by random sampling. If the model size is also still quite large for discretized random parameters, then the next reduction step may be based on the deletion of some 'unimportant' scenarios.

Preprocessing. Preprocessing is any technique that is utilized before use of the algorithm. It often serves for model evaluation and simplification, see [11]. With preprocessing techniques, we will notice deleting and aggregation of matrix rows and constraints. We will add remarks about generating relatively complete recourse. The most important task for the modeller utilizing stochastic programs is the reduction of the model size. This may be realized in different ways.

Postprocessing. When a stochastic program is successfully solved, new questions usually appear. Traditional questions are inherited from deterministic programming (sensitivity analysis, stability with respect to constant parameters), and other questions are specific for stochastic programs (stability with respect to distribution change, scenario probability update, unimportant scenario deleting, and important scenario adding). All these techniques realized after the program solution belong to postprocessing.

Acknowledgements

The present work has been supported by the Molde University College's project NRF Power Up Project – WP3 and by European Regional Development Fund in the framework of the research project NETME Center under the Operational Program Research and Development for Innovation. Reg. No. CZ.1.05/2.1.00/01.0002, id code: ED0002/01/01.

References

[1] Birge J.R., Louveaux F.: Introduction to Stochastic Programming, Springer Series in Operations Research, Springer Verlag, Berlin, 1997
[2] Caroe C.C.: The *L*-shaped method in stochastic integer programming, 7th International Conference on Stochastic Programming, June 26–29, 1995
[3] Ermoliev Y.M.: Stochastic quasigradient methods and their application to systems optimization, Stochastics, 9:1–36, 1983
[4] Ermoliev Y.M., Wets R.J.-B., editors: Numerical Techniques for Stochastic Optimization Problems, Springer Series in Computational Mathematics, 10, Springer Verlag, Berlin, 1988
[5] Gaivoronski A.: Implementation of stochastic quasigradient methods, In Y.M. Ermoliev and R.J.-B. Wets, editors, Numerical Techniques for Stochastic Optimization, pages 313–352, Springer Verlag, Berlin, 1988, Chapter 16
[6] Haneveld W.K.K., Stougie L., van der Vlerk M.H.: Stochastic integer programming: State of the art, Technical report, SOM, May 1998
[7] Holešovský J., Popela P., Roupec J.: On a disruption in congested networks, In Proceedings of the 18th International Conference of Soft Computing MENDEL 2013, accepted
[8] Hrabec D., Popela P., Novotný J., Haugen K.K., Olstad A.: The stochastic network design problem with pricing, In Proceedings of the 18th International Conference of Soft Computing MENDEL 2012, pp. 416–421, 2012
[9] Jonsbråten T.W., Woodruff D.L., Wets R.J.-B.: A class of stochastic programs with decision dependent random elements, University of California Davis, April 14 1997
[10] Kall P.: Stochastic Linear Programming, Springer, Berlin, 1976
[11] Kall P., Wallace S.W.: Stochastic Programming, John Wiley and Sons, Chichester, 1994
[12] Láníková I., Štěpánek P., Šimůnek P.: The fully probabilistic design of concrete structures, In Proceedings of the 16th International Conference on Soft Computing MENDEL 2010, pp. 426–433, 2010
[13] Matoušek R., Žampachová E.: Promising GAHC and HC12 algorithms in global optimization tasks, Optimization Methods & Software, vol. 26, no. 3, pp. 405–419, 2011
[14] Mak W.K., Morton D.P., Wood R.K.: Monte carlo bounding techniques for deterministic solution quality in stochastic programs, The University of Texas at Austin, 1997
[15] Mulvey J.M., Ruszczyński A.: A diagonal quadratic approximation method for linear multistage stochastic programming problems, In P. Kall, editor, System Modelling and Optimization, pages 588–597, Springer, Berlin, 1992 Lecture Notes in Control and Information Sciences 180
[16] Nazareth J.L., Wets R.J.-B.: Nonlinear programming techniques applied to stochastic programs with recourse, In Y.M. Ermoliev and R.J.-B. Wets, editors, Numerical Techniques for Stochastic Optimization, pages 95–122, Springer Verlag, Berlin, 1988, Chapter 4
[17] Olstad A., Haugen K.K., et al.: Omya Hustadmarmor: Optimizing the supply chain of calcium carbonate slurry to the European paper making industry, INTERFACES fnalist in the Franz Edelemann competition, Vol. 37, No. 1 : 1–13
[18] Pavlas M., Touš M., Bébar L., Stehlík P.: Waste to energy? An evaluation of the environmental impact, Applied Thermal Engineering, vol. 30, no. 16, pp. 2326–2332, 2010
[19] Popela P.: An Objected-Oriented Approach to Multistage Stochastic Programming, PhD Thesis, Prague: Charles University, 1998
[20] Popela P.: Stochastic Programming Models and Methods for Technical Applications, Folia Facultatis Scientarum Naturalium Universitatis Masarykianae Brunensis, 11, pp. 181–206, (2002)

[21] Popela P.: Numerical Techniques and Available Software, Chapter 8 in Part II, In J. Dupacova, J. Hurt, J. Stepan: Stochastic Modeling in Economics and Finance, Applied Optimization, Dordrecht/Boston/London: Kluwer Academic Publishers, 2002, s. 206–227
[22] Popela P.: Stochastic Programming Models for Engineering Design Problems, Engineering Mechanics, Vol. 17, 2010, No. 5/6, p. 351–362
[23] Prékopa A.: Stochastic Programming, Kluwer Academic Publishers, 1995
[24] Rockafellar R.T., Wets R.J.-B.: A Lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming, Mathematical Programming Study, 28:63–93, 1986, Stochastic Programming 84, Part II, A. Prékopa and R.J.-B. Wets (eds.)
[25] Rockafellar R.T., Wets R.J.-B.: Scenarios and policy aggregation in optimization under uncertainty, Mathematics of Operations Research, 16(1):119–147, 1991
[26] Roupec J.: Advanced genetic algorithms for engineering design problems, Engineering Mechanics, vol. 17, no. 5-6, pp. 407–417, 2011
[27] Ruszczyński A.: Parallel decomposition of multistage stochastic problems, Mathematical Programming, 58(2):201–228, 1993
[28] Schultz R.: Continuity and stability in two-stage stochastic integer programming. In K. Marti, editor, Stochastic Optimization: Numerical Methods and Technical Applications, pages 81–92, Springer Verlag, Berlin, 1992, Lecture Notes in Economics and Mathematical Systems 379
[29] Štětina J., Klimeš L., Mauder T., Kavička F.: Final-structure prediction of continuously cast billets, Materiali in tehnologije, vol. 46, no. 2, pp. 155–160, 2012
[30] Stougie L.: Design and analysis of algorithms for stochastic integer programming, PhD thesis, Centrum for Wiskunde en Informatica, Amsterdam, 1985
[31] Varaiya P., Wets R.J.-B.: Stochastic dynamic optimization approaches and computation, In Mathematical Programming, Recent Developments and Applications, pages 309–332, Kluwer Academic Publishers, Dordrecht, 1989
[32] Wallace S.W., Helgason T.: Structural properties of the progressive hedging algorithm, Annals of Operations Research, 30–31:445–456, 1991, John R. Birge and Roger J.-B. Wets (eds.)
[33] Wallace S.W.: Decision making under uncertainty: Is sensitivity analysis of any use? Operations Research, To appear in, 1998
[34] Wets R.J.-B.: Stochastic programming, In G.L. Nemhauser et al., editors, Handbook on OR and MS, Vol. 1, pages 573–629, North-Holland, Amsterdam, 1989
[35] Wets R.J.-B.: The aggregation principle in scenario analysis and stochastic optimization, In S.W. Wallace, editor, Algorithms and Model Formulations in Mathematical Programming, pages 91–113, Springer, Berlin, 1989
[36] Wets R.J.-B.: Challenges in stochastic programming, Mathematical Programming, 75:115–135, 1996
[37] Žampachová E., Popela P., Mrázek M.: Optimum beam design via stochastic programming, Kybernetika, vol. 46, pp. 575–586, 2010

Received in editor's office: March 11, 2014
Approved for publishing: July 25, 2014