KYBERNETIKA — VOLUME *53* (2017), NUMBER 6, PAGES 1012–1025

WARM-START CUTS FOR GENERALIZED BENDERS DECOMPOSITION

Jakub Kůdela and Pavel Popela

In this paper, we describe a decomposition algorithm suitable for two-stage convex stochastic programs known as Generalized Benders Decomposition. For this algorithm we propose a new reformulation that incorporates a lower bound cut that serves as a warm-start, decreasing the overall computation time. Additionally, we test the performance of the proposed reformulation on two modifications of the algorithm (bunching and multicut) using numerical examples. The numerical part is programmed in MATLAB and uses state-of-the-art conic solvers.

Keywords: stochastic programming, Generalized Benders Decomposition, L-shaped method, warm–start

Classification: 90C15, 90C25, 49M27

1. INTRODUCTION

In stochastic programming, we usually have to deal with problems that are large-scale but have a special structure [3]. Proper utilization of this special structure is the key part in the construction of any practically usable algorithm. One of the most widely used algorithms for two-stage stochastic linear programs is the L-shaped method developed by Van Slyke and Wets [12]. This method is based on (or, as the authors of the method wrote in the original paper: "is essentially the same as") the algorithm developed by Benders in [2] known as the Benders Decomposition. Over the years, numerous extensions for the L-shaped method have been proposed. A summary of the ones that are currently used can be found in [13] and [14].

A further generalization of the Benders decomposition for nonlinear convex problems ([1, 4]) was proposed by Geoffrion in [7] and was named the Generalized Benders Decomposition (GBD). The method found its main use as a solution technique for mixed-integer nonlinear problems, described in [5] and [6].

In this paper, we describe a formulation of the GBD that suits the particular structure of two-stage stochastic programming problems. After that, we introduce a reformulation that enables us to add a lower bound cut, which acts as a "warm-start" for the algorithm. As the lower bound cut, we decided to use the one that we can compute with the least effort. As there have been several lower bounds proposed for stochastic programs (for

DOI: 10.14736/kyb-2017-6-1012

example in [3] and [11]) the question of the appropriate one for our problem will be left open for future research.

2. GBD – MAIN IDEAS

In this section, we give a brief insight into the GBD, as it is not our intention to devote several pages to its thorough description. An interested reader can find an in-depth analysis of the method in the original paper [7] and in the works of Floudas in [5] and [6].

The problems GBD aims to solve are of the form:

$$\begin{array}{ll} \underset{x,y}{\text{minimize}} & f(x,y) \\ \text{subject to} & G(x,y) \le 0, \quad x \in X, y \in Y, \end{array} \tag{1}$$

where $x \in X \subseteq \Re^{n_1}$, $y \in Y \subseteq \Re^{n_2}$, $f : \Re^{n_1} \times \Re^{n_2} \longrightarrow \Re$ is a real-valued objective function and $G : \Re^{n_1} \times \Re^{n_2} \longrightarrow \Re^m$ is an m-vector of constraint functions. The variable x is called a complicating variable in the sense that (1) is a much easier optimization problem in y when x is temporarily held fixed. The following conditions are required:

C1: Y is a nonempty, convex set and the functions f and G are convex for each fixed $x \in X$.

C2: The set

$$Z_x = \{z \in \Re^m : G(x,y) \le z \text{ for some } y \in Y\} \tag{2}$$

is closed for each fixed $x \in X$.

C3: For each fixed $x \in X \cap V$, where

$$V = \{x : G(x,y) \le 0, \text{ for some } y \in Y\}, \tag{3}$$

one of the following conditions holds:

(i) the problem (1) has a finite solution and has an optimal multiplier vector for the inequalities.

(ii) the problem (1) is unbounded, that is, its objective function value goes to $-\infty$.

This covers quite a wide range of problems [5]. The particular situation we are interested in is when f and G are linearly separable in x and y, i. e.

$$\begin{array}{rcl} f(x,y) & = & f_1(x) + f_2(y), \\ G(x,y) & = & G_1(x) + G_2(y). \end{array} \tag{4}$$

The basic idea in GBD is the generation, at each iteration, of an upper bound and a lower bound on the optimal solution of (1). The upper bound results from a subproblem, while the lower bound results from a master problem. The subproblem corresponds to the problem (1) with fixed x-variable (i. e., it is in the y-space only), and its solution provides information about the upper bound and the Lagrange multipliers ([1, 4]) associated with the inequality constraints. The master problem is derived via nonlinear duality theory, makes use of the Lagrange multipliers obtained in the subproblem, and its solution provides information about the lower bound, as well as the next set of fixed x-variable to be used subsequently in the subproblem [5].

3. GBD FOR TWO-STAGE STOCHASTIC PROGRAMMING PROBLEMS

In stochastic programming linear separability of the objective function and constraints is a very common property. Especially the two-stage stochastic programming problems can be often linearly separated into the functions concerning only the first-stage and the second-stage decision variables – this is the raison d'etre of the following passages, and it is why we believe that the GBD (in its slightly modified form) is a well-suited algorithm for these kinds of problems.

3.1. Problem formulation

Let us consider the following problem:

$$\begin{aligned} \underset{x,y_1,\dots,y_K}{\text{minimize}} \quad & f_1(x) + \sum_{k=1}^{K} p(\xi_k) f_2(y_k,\xi_k) \\ \text{subject to} \quad & G_{11}(x) \le 0, \\ & G_{21}(\xi_k)x + G_{22}(y_k,\xi_k) \le 0, \ \xi_k \in \Xi, \end{aligned} \tag{5}$$

where $f_1 : \Re^{n_1} \longrightarrow \Re$ is a convex function, all m_1 constraint functions $G_{11} : \Re^{n_1} \longrightarrow \Re^{m_1}$ are convex, and for all $\xi_k \in \Xi$ with $|\Xi| = K$ finite, $G_{21}(\xi_k)$ is a $m_2 \times n_1$ matrix, $f_2(\cdot,\xi_k) : \Re^{n_2} \longrightarrow \Re$ is convex, all m_2 constraint functions $G_{22}(\cdot,\xi_k) : \Re^{n_2} \longrightarrow \Re^{m_2}$ are convex, $P(\xi = \xi_k) \equiv p(\xi_k) > 0, \sum_{k=1}^{K} p(\xi_k) = 1$.

The master problem corresponding to (5) has the following form:

$$\begin{aligned} \underset{x,\theta}{\text{minimize}} \quad & f_1(x) + \theta \\ \text{subject to} \quad & G_{11}(x) \le 0, \\ & D_i x \le d_i, \qquad i = 1,\dots,p, \\ & E_j x - \theta \le e_j, \quad j = 1,\dots,r, \end{aligned} \tag{6}$$

where $\theta \in \Re$ serves as the lower bound on the second stage objective value. The meaning of matrices D, E and vectors d, e will be fully discussed in the actual solution procedure. These matrices and vectors correspond to the feasibility and optimality cuts derived from the solutions of the subproblem.

Because of the structure of the two-stage stochastic programming problems, the subproblem separates into K independent subproblems (one for each scenario) in the form:

$$\begin{aligned} \underset{y_k}{\text{minimize}} \quad & f_2(y_k,\xi_k) \\ \text{subject to} \quad & G_{21}(\xi_k)x + G_{22}(y_k,\xi_k) \le 0. \end{aligned} \tag{7}$$

Remark 3.1. Regarding our notation – one could use k instead of ξ_k in the formulations above (and in the ones that will follow). The use of ξ is standard in the stochastic programming literature.

3.2. Solution procedure

The following algorithm is an implementation of the GBD inspired by [7] and [5]. The single difference (apart from the notation) is that the separability of the subproblem into K independent subproblems is taken into account. At the start of the procedure, the matrices D, E and vectors d, e are empty (they store the successive cuts as the iterations progress).

To our best knowledge, this is the first implementation of the GBD for two-stage stochastic convex programming problems of the form (5).

Step 0. Set $p = 0, r = 0$, and $\epsilon > 0$.

Step 1. Solve (6) and obtain $(\bar{x}, \bar{\theta})$. The optimal objective value of (6) gives us a lower bound on optimal objective value of (5).

Step 2. For fixed $x = \bar{x}$ solve all K subproblems (7). One of two possibilities can happen.

Step 2A. For some k the subproblem (7) is infeasible. Solve the following problem:

$$\begin{array}{ll} \underset{y_k, v \geq 0}{\text{minimize}} & ||v||_1 \\ \text{subject to} & G_{21}(\xi_k)\bar{x} + G_{22}(y_k, \xi_k) \leq v, \end{array} \tag{8}$$

where $v \in \Re^{m_2}$ is a decision vector representing "slacks" in the constraints. Get $(\bar{y}_k, \bar{v})$ and from its dual obtain the optimal Lagrange multipliers λ. Set $p = p+1$. Add a new row to the matrix D and vector d in (6):

$$D_p = \lambda^T G_{21}(\xi_k), d_p = \lambda^T(-G_{22}(\bar{y}_k, \xi_k)). \tag{9}$$

Return to Step 1.

Step 2B. All the subproblems have finite optimal values, we obtained $(\bar{y}_k, u_k)$, where u_k are optimal Lagrange multipliers. The evaluation of the objective of (5) at $(\bar{x}, \bar{y}_1, \dots, \bar{y}_K)$ gives us an upper bound on its optimal value. Check for optimality: if

$$\bar{\theta} + \epsilon \geq \sum_{k=1}^{K} p(\xi_k) f_2(\bar{y}_k, \xi_k), \tag{10}$$

terminate, $(\bar{x}, \bar{y}_1, \dots, \bar{y}_K)$ are ϵ-optimal [7]. Otherwise, set $r = r+1$ and add a new row to the matrix E and vector e in (6):

$$\begin{aligned} E_r &= \sum_{k=1}^{K} p(\xi_k)(u_k^T G_{21}(\xi_k)), \\ e_r &= -\sum_{k=1}^{K} p(\xi_k)(f_2(\bar{y}_k, \xi_k) + u_k^T(G_{22}(\bar{y}_k, \xi_k)). \end{aligned} \tag{11}$$

Return to Step 1.

Remark 3.2. In Step 1, before any optimality cut is added, $\bar{\theta}$ as well as the optimal objective value of (6) will be $-\infty$. For computational reasons it is advisable to include a lower bound on θ in the actual implementation of the algorithm.

Remark 3.3. If $X \subseteq V$ (i. e., in the case of complete or relatively complete recourse [3]), the Step 2A is never needed and for a given $\epsilon > 0$ the GBD terminates in a finite number of iterations. If however, $X \nsubseteq V$, then we may need to solve Step 2A infinitely many successive times. In such a case, to preserve finite ϵ-convergence, we can modify the procedure so as to finitely truncate any excessively long sequence of successive executions of Step 2A and go to Step 2B with $\hat{x}$ equal to the extrapolated limit point which is assumed to belong to $X \cap V$, see [5] or [6].

4. REFORMULATION WITH A BOUNDING CUT

In this section, we introduce a novel reformulation of the master problem (6) that includes bounds obtained from problems, that can be thought of as predecessors of the two-stage stochastic programming problem (5). The definitions of these problems, as well as their subsequent relations, are based on [9].

4.1. Bounds

Let us define

$$\begin{aligned} \underset{x_k, y_k}{\text{minimize}} \quad & f_1(x_k) + f_2(y_k, \xi_k) \\ \text{subject to} \quad & G_{11}(x_k) \le 0, \\ & G_{21}x_k + G_{22}(y_k, \xi_k) \le 0, \end{aligned} \tag{12}$$

as the optimization problem for one particular realization $\xi_k \in \Xi$ and denote its optimal objective function value as $z(\xi_k)$ The wait-and-see solution is the solution without nonanticipativity constraints (i. e. all scenarios are treated and optimized separately). We will denote the average of the optimal objective values of (12) (when treated separately) as:

$$\mathrm{WS} = \sum_{k=1}^{K} p(\xi_k) z(\xi_k). \tag{13}$$

Now we may compare this wait-and-see solution to the solution of (5). We will denote the optimal objective value of (5) as RP (the recourse problem [3]). The following inequality holds for any stochastic program:

$$\mathrm{WS} \le \mathrm{RP}. \tag{14}$$

From this, we can see that WS creates a valid lower bound on the harder problem we are aiming to solve. The idea behind the reformulation is to include such a valid lower bound to the algorithmic procedure to “jumpstart” it and by doing so save on iterations, and, as a result, save on the overall computational effort and time.

For practical purposes, many people would believe that finding the wait-and-see solution is still too much work. A natural temptation is to solve a much simpler problem:

the one obtained by replacing all random variables by their expected values. This is called the expected value problem, which is simply

$$\mathrm{EV} = z(\bar{\xi}), \tag{15}$$

where $\bar{\xi} = \sum_{k=1}^{K} p(\xi_k)\xi_k$.

4.2. Reformulation

Although WS is a valid bound, the computational effort for its enumeration is much higher compared to the effort to compute EV (if $|\Xi| = K$, then computing EV is K times faster). However, EV does not necessarily have to play the role of a lower bound on RP; there are instances, where RP $\leq$ EV. For the purpose of deriving the reformulation, we will, for now, suppose that EV is, in fact, a valid lower bound on RP. The discussion on what is going to occur when it is not will follow shortly after. Suppose

$$\mathrm{EV} \leq \mathrm{RP}, \tag{16}$$

holds, then

$$f_1(x) + \sum_{k=1}^{K} p(\xi_k) f_2(y_k, \xi_k) \geq \mathrm{EV}, \tag{17}$$

holds for the optimum of (5). This inequality cannot be added directly to (5) since it would cease to be a convex program. The reformulation we propose does not directly alter (5) but is instead aimed at the master problem (6). A new variable z is introduced to bound the first-stage objective from above (by minimizing this variable we effectively minimize the first-stage objective itself)

$$f_1(x) \leq z, \tag{18}$$

which is a convexity preserving inequality. Furthermore, this new variable z added to the variable representing the second stage θ form a lower bound on the overall objective function. Finally, the bound

$$z + \theta \geq \mathrm{EV}, \tag{19}$$

since it is affine, can be added to (6), and the reformulation of the problem is

$$\begin{array}{lll} \underset{z,x,\theta}{\text{minimize}} & z + \theta & \\ \text{subject to} & f_1(x) \leq z, & \\ & G_{11}(x) \leq 0, & \\ & z + \theta \geq \mathrm{EV}, & \\ & D_i x \leq d_i, & i = 1, \dots, p \\ & E_j x - \theta \leq e_j, & j = 1, \dots, r. \end{array} \tag{20}$$

After this reformulation, the algorithm continues as usual, arriving at an ϵ-optimal solution in, preferably, a shorter time than its original counterpart (we will see the results of some numerical examples in later sections).

Now, let us address what happens if (16) does not hold. One of two possibilities can occur, namely, that optimal objective function value (as determined by the algorithm) will be equal to EV, or that the problem will be infeasible. The price we pay for mistakenly using the cuts (16) is, in both cases, one iteration of the algorithm – i. e. after one iteration we can assess, if our algorithm will arrive at the desired solution, and, either restart it without (16) (possibly including WS instead), or continue.

However, certain situations can happen when we restart the algorithm without (16) and get the same result again. This occurs if the original problem is infeasible (in which case we have some serious model or data issues) or if EV = RP, in that case we would have to run the entire algorithm only to arrive at the same objective function value (which is a bit unfortunate, but unavoidable).

Another important question is if the cut (16) is worth having an additional variable. The numerical examples we provide in the later sections should supply us with some, although not definitive, insight into this issue.

Lastly, the question whether or not it is better to use the guaranteed lower bound in WS is also present. As we mentioned earlier, WS is computationally much more expensive than EV. In the examples that will follow we did not carry any examination of the WS bound, nor of any other possible bound. This is one of the areas that require further future investigation.

The solution procedure can be summarized in the following steps:

Step 0. Solve the expected value problem to get EV (15). Set $p = 0, r = 0$, and $\epsilon > 0$. Solve (20) and obtain $(\bar{z}, \bar{x}, \bar{\theta})$. If $\bar{z} + \bar{\theta} =$ EV, terminate (and use the original method without the EV cut, or use WS instead). Otherwise, go to Step 2.

Step 1. Solve (20) and obtain $(\bar{z}, \bar{x}, \bar{\theta})$.

Step 2., Step 2A., Step 2B. The same as in section 3.2.

5. BUNCHING AND MULTICUTS

Just as in the linear case with the L-shaped method, different implementations of the algorithm can be researched for improving its performance ([3, 13]). Two possible adjustments suitable for GBD – bunching and the multicut formulation will be discussed and brought into the numerical examination.

Bunching, as the name suggests, is a technique that instead of the full scenario decomposition uses "bunches" of scenarios and decomposes the original problem alongside these bunches. Having L bunches of scenarios and sets of indices $B_l \neq \emptyset, l = 1, \dots, L$, such that $B_i \cap B_j = \emptyset$ for $i \neq j$ and $\bigcup_{l=1}^{L} B_l = \{1, \dots, K\}$. The subproblems (7) for each bunch l have the form

$$\begin{aligned} \underset{y_k, k \in B_l}{\text{minimize}} \quad & \sum_{k \in B_l} p(\xi_k) f_2(y_k, \xi_k) \\ \text{subject to} \quad & G_{21}(\xi_k)x + G_{22}(y_k, \xi_k) \leq 0, k \in B_l. \end{aligned} \tag{21}$$

The feasibility and optimality cuts in Step 2A. and Step 2B. of the algorithm are changed accordingly. The feasibility cut in Step 2A. becomes

$$D_p = \sum_{k\in B_l} \lambda_k^T G_{21}(\xi_k), \quad d_p = -\sum_{k\in B_l} \lambda_k^T G_{22}(\bar{y}_k, \xi_k), \tag{22}$$

and the optimality cut in Step 2B. becomes

$$\begin{aligned} E_r &= \sum_{l=1}^{L}\sum_{k\in B_l} u_k^T G_{21}(\xi_k), \\ e_r &= -\sum_{l=1}^{L}\sum_{k\in B_l} p(\xi_k)(f_2(\bar{y}_k, \xi_k) + u_k^T G_{22}(\bar{y}_k, \xi_k), \end{aligned} \tag{23}$$

where λ_k and u_k are the Lagrange multipliers corresponding to the inequalities from scenario $k \in B_l$.

In the linear case, bunching comes from the idea that several second-stage problems might have the same optimal basis [3]. In the convex case, the justification is a bit different. Our argumentation is purely in the realm of the actual computation – it is sometimes faster (due to a non-zero initialization time, etc.) to compute a larger instance containing several separable problems than to solve these problems separately. The examples will show, up to a certain point, exactly this kind of behavior.

The multicut formulation comprises of developing one cut for every second-stage problem (i. e. for every scenario) instead of the aggregated cut introduced in (6). It results in adding a separate θ_k for each scenario and as a consequence in a much greater number of cuts which more accurately describe the recourse function [3]. The master problem for multicut formulation has the following form (without the additional cut developed in the previous section)

$$\begin{array}{lll} \underset{x,\theta_1,\dots,\theta_K}{\text{minimize}} & f_1(x) + \sum_{k=1}^{K} \theta_k & \\ \text{subject to} & G_{11}(x) \le 0, & \\ & D_i x \le d_i, & i = 1,\dots,p, \\ & E_{j(k)}x - \theta_k \le e_{j(k)}, & j(k) = 1,\dots,r(k), \\ & k = 1,\dots,K, & \end{array} \tag{24}$$

where $r(k)$ and $j(k)$ indices are related to the kth subproblem, see the steps below. In this case, the feasibility cuts remain the same, but the remaining steps require the following changes:

Step 0. – Multicut Set $p = 0, r(k) = 0$, for $k = 1,\dots,K$ and $\epsilon > 0$.

Step 1. – Multicut Solve (24) and obtain $(\bar{x}, \bar{\theta}_1, \dots, , \bar{\theta}_K)$.

Step 2. – Multicut For fixed $x = \bar{x}$ solve all K subproblems (7). One of two possibilities can happen.

Step 2A. – Multicut As before.

Step 2B. – Multicut All the subproblems have finite optimal values, we obtained $(\bar{y}_k, u_k)$, where u_k are optimal Lagrange multipliers. For $k = 1, \dots, K$ if

$$\bar{\theta}_k + \epsilon \le p(\xi_k) f_2(\bar{y}_k, \xi_k), \tag{25}$$

set $r(k) = r(k) + 1$ and add a new row to the matrix E and vector e in (24):

$$\begin{aligned} E_{r(k)} &= p(\xi_k)(u_k^T G_{21}(\xi_k)), \\ e_{r(k)} &= -p(\xi_k)(f_2(\bar{y}_k, \xi_k) + u_k^T G_{22}(\bar{y}_k, \xi_k)). \end{aligned} \tag{26}$$

If (25) does not hold for any k, terminate. Otherwise, return to Step 1.

Even though this formulation provides a more accurate description of the recourse function, its usefulness in the convex case is highly ambiguous. The number of variables in the master problem is much larger than in the original algorithm and the number of constraints (cuts) added in each iteration is also much higher.

6. NUMERICAL EXAMPLES

To test the above mentioned theoretical concepts, we designed two convex two-stage problems. On these problems, we compare the performance of different variations of the GBD as well as a formulation without any decomposition (denoted as full recourse problems).

The implementation was done in MATLAB using its embedded `fmincon` solver and the state-of-the-art conic solvers `SeDuMi` and `SDPT3` [10] (which are a part of the CVX modeling system [8]). Although the examples are not derived from any applied problems, they provide a valid insight into the advantages and disadvantages of the presented methods.

6.1. Example 1

The first example investigates the following problem

$$\begin{array}{ll} \underset{\boldsymbol{x}, \boldsymbol{y_1}, \dots, \boldsymbol{y_k}}{\text{minimize}} & (x_1 - 4)^4 + (x_2 - 3)^4 + \sum_{k=1}^{K} p_k (q_{k,1} e^{y_{k,1}} + q_{k,2} y_{k,2}^4) \\ \text{subject to} & x_2 - \ln(x_1 + 1) - 1 \le 0, \\ & x_2 + x_1^3 - 8 \le 0, \\ & x_1, x_2 \ge 0, \\ & x_1 + h_{k,1} - y_{k,1} \le 0, k = 1, \dots, K, \\ & x_2 + h_{k,2} - y_{k,2} \le 0, k = 1, \dots, K, \end{array}$$

where the random parameters q and h are $q \sim |N(0,3)|$, $h \sim 0.7 \cdot |N(0,1)| + 0.5$. The scenarios are then constructed using the usual Monte Carlo sampling, the number of scenarios will vary to demonstrate the performance of the different approaches. The methods and solvers used for solving the problem were:

- vanilla (original) version of GBD (master and subproblems solved by `fmincon`);

- reformulation with the EV cut (master and subproblems solved by `fmincon`);
- bunching of several scenarios (master and subproblems solved by `fmincon`);
- bunching of several scenarios with the EV cut (master and subproblems solved by `fmincon`);
- full recourse problem (FRP) solved by `fmincon`;
- FRP solved by `SDPT3` (as a part of the CVX modeling system);
- FRP solved by `SeDuMi` (as a part of the CVX modeling system).

The required precision for all the methods was set to $\epsilon = 10^{-5}$. The results are summarized in the tables that follow. The Time[s] value represents the computational time it took the procedure to terminate, given the same level of accuracy for all methods. The number of scenarios in the first instance is $K = 60$. The first two tables show, how the computational time of the GBD is affected by introducing the EV cut:

Method	Vanilla	EV cut
Time[s]	12.26	9.05

Tab. 1. Computational time [s] for Vanilla version and EV cut, $K = 60$.

and by bunching with different sizes of the bunch:

Bunch size	2	3	5	10	12	15	20	30
Time[s] – Without EV cut	7.04	5.08	3.61	2.76	2.64	2.65	2.75	3.22
Time[s] – With EV cut	5.19	3.79	2.75	2.07	2.02	1.99	2.13	2.44

Tab. 2. Computational time [s] for bunching with different sizes of the bunch, $K = 60$.

An identical structure is utilized in the case of $K = 240$ scenarios:

Method	Vanilla	EV cut
Time[s]	43.84	35.42

Bunch size	2	3	5	6	8	10	12	15
Time[s] – Without EV cut	24.8	17.7	12.4	11.4	9.9	9.2	8.8	8.7
Time[s] – With EV cut	19.9	14.3	10.2	9.2	8.0	7.4	7.2	7.0
Bunch size	16	20	24	30	40	60	80	120
Time[s] – Without EV cut	8.7	8.9	9.4	10.5	12.2	16.8	21.5	25.3
Time[s] – With EV cut	7.0	7.2	7.7	9.5	9.8	13.6	17.3	20.2

Tab. 3. Computational time [s] for Vanilla, EV cut and bunching, $K = 240$.

From these result, we see that the EV cut, as well as efficient bunching, can have a strong effect on the overall computation time. The experiments suggest that there exists an "optimal" bunch size that is independent of the number of scenarios. For this particular problem, it seemed that a bunch size between 12 and 16 was the one. For the subsequent computations, the bunch size 15 was chosen.

In the following table, we compare the computation times for growing number of scenarios using the methods and solvers mentioned above:

Number of scenarios	60	240	1,500	2,400	4,800	6,000
Vanilla	12.3	43.8	258.6	410.1	–	–
EV cut	9.1	35.4	208.9	332.7	–	–
Bunch 15	2.7	8.7	52.5	78.0	154.75	194.9
Bunch 15 with EV	1.9	7.0	40.1	62.6	124.6	156.4
FRP – `SDPT3`	6.1	22.3	139.9	241.7	–	–
FRP – `SeDuMi`	1.4	6.2	32.8	65.2	170.3	242.5
FRP – `fmincon`	0.5	12.2	3,000*	3,000*	–	–

Tab. 4. Computational time [s] for different methods, increasing number of scenarios.

The asterisk(*) denotes that the algorithm did not arrive at the desired precision (i. e. even after 3,000s the `fmincon` did not arrive sufficiently near the optimum). The dash(–) means that we did not pursue the analysis in this direction since we anticipated results incomparable with the more efficient methods.

These results show that for big enough problems, the efficient implementation GBD, even with simpler solvers, can outperform the state-of-the-art solvers. For smaller instances, however, these solvers are more efficient (as will be presented in the results of the second example).

6.2. Example 2

The second example included in our investigation, compared to the first one, adds some more first and second-stage variables and non-differentiable functions. These are the reason why, in the implementation, the more efficient solvers had to be utilized for the solution of the master problem (`fmincon` performed very poorly in this case). The problem in question is the following

$$\begin{aligned}\underset{\boldsymbol{x},\boldsymbol{y_1},\dots,\boldsymbol{y_k}}{\text{minimize}} \quad & (x_1-4)^4+(x_2-3)^4+(x_3-2.5)^2+3|x_1+x_4+4x_5-15| \\ & +\sum_{k=1}^{K} p_k(q_{k,1}e^{y_{k,1}}+q_{k,2}y_{k,2}^4+q_{k,3}(y_{k,3}-2)^2 \\ & +q_{k,4}|y_{k,4}+q_{k,5}y_{k,5}|)\end{aligned}$$

$$\begin{aligned}\text{subject to}\quad & x_2 - \ln(x_1+1) - \sqrt{x_3} + x_4 + x_5^2 - 10 \le 0,\\ & x_2 + x_1^3 + x_3^3 - 10 \le 0,\\ & -x_4 - \sqrt{x_5} + 5 <= 0,\\ & x_i \ge 0,\ i = 1,\dots,5\\ & T_k\boldsymbol{x} + W_k\boldsymbol{y_k} \le h_k,\ k = 1,\dots,K.\end{aligned}$$

The random parameters q, h, W and T are (using some MATLAB syntax), $q \sim |N(0,1)|$, $h \sim -0.7 \cdot |N(0,1)| - 1$, $W = -\mathrm{I}_5$, M = 5x5 matrix with 1 to 3 zeros in each column, the rest are 1, $T = \text{abs}(0.2.*\text{randn}(5)).*M$. The scenarios are, again, constructed using the Monte Carlo sampling. As before, we used several methods and solvers for solving the problem:

- vanilla version of GBD (master solved by `SeDuMi`, subproblems by `fmincon`);
- EV cut version (master solved by `SeDuMi`, subproblems by `fmincon`);
- bunching + EV cut (master solved by `SeDuMi`, subproblems by `by fmincon`);
- bunching + EV cut + multicut (master solved by `SeDuMi`, subproblems by `fmincon`);
- FRP solved by `SDPT3`;
- FRP solved by `SeDuMi`.

The required precision for all the methods was set to $\epsilon = 10^{-5}$. By computations similar to that of the first example, we found the appropriate bunching size to be 5. The comparison of the different methods for varying number of scenarios is summarized in the following table:

Number of scenarios	125	250	500	1,000	2,000	3,000	5,000	7,500
Vanilla	19	45	77	164	313	–	–	–
EV cut	15	34	61	123	320	–	–	–
Multicut	19	36	134	150	309	–	–	–
Bunch 5	12	32	44	84	170	255	452	650
Bunch 5 + EV	9	17	34	71	175	210	364	578
Bunch 5 + Multicut	11	20	32	74	250	452	–	–
Bunch 5 + Multicut + EV	9	15	34	99	256	463	–	–
FRP – `SDPT3`	13	30	64	130	334	–	–	–
FRP – `SeDuMi`	1	2	6	15	40	102	355	622

Tab. 5. Computational time [s] for different methods, increasing number of scenarios.

The results demonstrate the pros and cons of using the GBD algorithm. For smaller instances, it is much more efficient to use the appropriate state-of-the-art and free solver (`SeDuMi`) to attack the full recourse formulation. However, for larger problems, the bunching variation of the GBD was able to outperform all the rest. The multicut variation suffered from a growing size of the master problem and, in this setting, cannot be considered as an improvement (a similar behavior for linear problems was shown in [13]).

7. CONCLUSION

In this paper, we introduced a novel utilization and reformulation of the traditional Generalized Benders Decomposition. To support the utility of our reformulation (as well as the utility of the GBD itself), we presented our computational experience.

From the result of the numerical examples, it is apparent that the GBD and our modifications definitely have a place as solid techniques for solving medium-sized convex two-stage stochastic problems and that especially the bunching ideas and modifications produce fruitful results.

It must be acknowledged that further investigation (i. e. a wider variety of numerical test, preferably from applications) is needed to make the arguments more conclusive. Also, further research in terms of usable lower bound as the "warm-start" cuts is anticipated.

ACKNOWLEDGEMENT

This work was supported by NETME CENTRE PLUS (LO1202) created with financial support from the Ministry of Education, Youth and Sports under National Sustainability Programme I, by the research project "Moderní matematické metody pro modelování problémů technických a přírodních věd" (FSI-S-17-4464), and by the research project "Vývoj a aplikace moderních dekompozičních metod v oblasti stochastického programování" (FSI-FV-17-04).

(Received April 12, 2017)

REFERENCES

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty: Nonlinear Programming: Theory and Algorithms. Third edition. Wiley–Interscience, Hoboken, N. J. 2006. DOI:10.1002/0471787779

[2] J. F. Benders: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik *4* (1962), 1, 238–252. DOI:10.1007/bf01386316

[3] J. R. Birge and F. Louveaux: Introduction to Stochastic Programming. Springer, New York 2000.

[4] S. P. Boyd and L. Vandenberghe: Convex Optimization. Cambridge University Press, New York 2004. DOI:10.1017/cbo9780511804441

[5] C. A. Floudas: Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications. Oxford University Press, Oxford 1995.

[6] C. A. Floudas and P. M. Pardalos: Encyclopedia of Optimization. Springer, New York 2009. DOI:10.1007/978-0-387-74759-0

[7] A. M. Geoffrion: Generalized Benders decomposition. Journal of Optimization Theory and Applications *10* (1972), 4, 237–260. DOI:10.1007/bf00934810

[8] M. Grant and S. Boyd: Graph implementations for nonsmooth convex programs. In: Recent Advances in Learning and Control (V. Blondel, S. Boyd and H. Kimura, eds.), Springer–Verlag Limited, Berlin 2008, pp. 95–110. DOI:10.1007/978-1-84800-155-8_7

[9] A. Madansky: Inequalities for stochastic linear programming problems. Management Science *6* (1960), 197–204. DOI:10.1287/mnsc.6.2.197

[10] H. D. Mittelmann: The state-of-the-art in conic optimization software. In: Handbook on Semidefinite, Conic and Polynomial Optimization (M. F. Anjos and J. B. Lasserre, eds.), Springer US, New York 2012, pp. 671–686. DOI:10.1007/978-1-4614-0769-0_23

[11] G. Pflug and F. Maggioni: Bounds and approximations for multistage stochastic programs. SIAM J. Optim. *26* (2016), 1, 831–855. DOI:10.1137/140971889

[12] R. M. Van Slyke and R. Wets: L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. SIAM J. Appl. Math. *17* (1969), 4, 638–663. DOI:10.1137/0117061

[13] C. Wolf, C. I. Fabián, A. Koberstein, and L. Suhl: Applying oracles of on-demand accuracy in two-stage stochastic programming – A computational study. Europ. J. Oper. Res. *239* (2014), 2, 437–448. DOI:10.1016/j.ejor.2014.05.010

[14] V. Zverovich, C. I. Fabián, E. F. D. Ellison, and G. Mitra: A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition. Math. Program. Computation *4* (2012), 3, 211–238. DOI:10.1007/s12532-012-0038-z

Jakub Kůdela, FSI, Technická 2, Brno. Czech Republic.
e-mail: jakub.kudela89@gmail.com

Pavel Popela, FSI, Technická 2, Brno. Czech Republic.
e-mail: popela@fme.vutbr.cz