Noname manuscript No.
(will be inserted by the editor)

Benders Decomposition with Adaptive Oracles for Large Scale Optimization

Nicolò Mazzi · Andreas Grothey · Ken McKinnon · Nagisa Sugishita

Received: date / Accepted: date

Abstract This paper proposes an algorithm to efficiently solve large optimization problems which exhibit a column bounded block-diagonal structure, where subproblems differ in right-hand side and cost coefficients. Similar problems are often tackled using cutting-plane algorithms, which allow for an iterative and decomposed solution of the problem. When solving subproblems is computationally expensive and the set of subproblems is large, cutting-plane algorithms may slow down severely. In this context we propose two novel adaptive oracles that yield inexact information, potentially much faster than solving the subproblem. The first adaptive oracle is used to generate inexact but valid cutting planes, and the second adaptive oracle gives a valid upper bound of the true optimal objective. These two oracles progressively "adapt" towards the true exact oracle if provided with an increasing number of exact solutions, stored throughout the iterations. These adaptive oracles are embedded within a Benders-type algorithm able to handle inexact information. We compare the Benders with adaptive oracles against a standard Benders algorithm on a stochastic investment planning problem. The proposed algorithm shows the capability to substantially reduce the computational effort to ob-

Research is supported by the Engineering and Physical Sciences Research Council (EPSRC) through the CESI project (EP/P001173/1)

N. Mazzi[1]
E-mail: nicolo.mazzi@ed.ac.uk

A. Grothey[1]
E-mail: A.Grothey@ed.ac.uk

K. McKinnon[1]
E-mail: K.McKinnon@ed.ac.uk

N. Sugishita[1]
E-mail: s1576972@sms.ed.ac.uk

[1] School of Mathematics, University of Edinburgh,
James Clerk Maxwell Building, Edinburgh (UK), EH9 3FD

tain an ϵ-optimal solution: an illustrative case is 25.2 times faster for a 1.00% convergence tolerance and 11.6 times faster for a 0.01% tolerance.

Keywords Benders Decomposition · Inexact Oracles · Adaptive Oracles · Stochastic Investment Planning

1 Introduction

In this paper we consider problems that can be expressed in the form

$$\min_{\mathbf{x}\in\mathcal{X}} \quad f(\mathbf{x}) + \sum_{i\in\mathcal{I}} \pi_i\, g(x_i, c_i), \tag{1}$$

where $g(x_i, c_i)$ is the optimal solution of the LP subproblem

$$\mathbf{SP}: \qquad g(x_i, c_i) := \min_{y_i\in\mathcal{Y}} \{c_i^\top C y_i \mid Ay_i \le Bx_i\}. \tag{2}$$

The set of problems $\mathcal{I}$ is finite, the x_i are (possibly overlapping) subvectors of $\mathbf{x}$, $\mathcal{Y}$ is a convex polyhedron, and the π_i are non-negative constants. The coefficient matrices A, B, and C are the same in every subproblem so the subproblems differ only through the value of their parameters, x_i and c_i. The c_i are vectors of coefficients and the x_i are vectors of variables for (1) and parameters for (2). However, elements of x_i may be set to fixed values in $\mathcal{X}$, so become equivalent to coefficients. The inclusion of the matrices B and C allow the dimensions of x_i and c_i to differ from (and in the case we shall present be much smaller than) the number of explicit constraints and variables y_i in each subproblem.

The function $g(x_i, c_i)$ has properties that can be exploited in the solution procedure. It is a saddle function, convex w.r.t. x_i, and concave w.r.t. c_i. Moreover, we focus on problems where $g(x_i, c_i)$ is also a decreasing function of x_i and an increasing function of c_i. Monotonicity can be a natural property of the problem, e.g. when $C \ge 0$, $y \ge 0$, and $B \ge 0$, or the problem can be usually rearranged to have this property.

An example of a problem with the above structure is an investment planning problem. Here the $\mathbf{x}$ represent investment decisions with corresponding investment cost $f(\mathbf{x})$. The investments affect a set $\mathcal{I}$ of situations, and x_i is the subvector of $\mathbf{x}$ that represents the investments that affects the situation i, c_i specifies the operational costs, y_i is the operational decisions, and $g(x_i, c_i)$ gives the optimal operating cost. The test case we present in this paper is a stochastic problem, where the situations are the different possible scenarios that might occur in the future, each of which is weighted by the probability π_i of that situation occurring.

1.1 Literature Review

In the standard Benders decomposition algorithms [3,15] for problem (1) a sequence of relaxations is solved. At each iteration j the relaxed master problem is

$$\begin{aligned} \mathbf{RMP}: \quad & \min_{\mathbf{x}\in\mathcal{X}} \quad f(\mathbf{x}) + \sum_{i\in\mathcal{I}} \pi_i \beta_i \\ & \text{s.t.} \quad \beta_i \geq \theta + \lambda^\top (x_i - x), \quad \forall (x,\theta,\lambda) \in \Theta_i^{(j-1)},\ \forall i \in \mathcal{I}, \end{aligned} \tag{3}$$

and yields an optimal solution $\mathbf{x}^{(j)}$. To generate a new cutting plane $(x_i^{(j)}, \theta_i^{(j)}, \lambda_i^{(j)})$ at iteration j, we call an oracle that, for given $x_i^{(j)}$, returns the optimal objective value $\theta_i^{(j)} = g(x_i^{(j)}, c_i)$ of **SP** and a subgradient $\lambda_i^{(j)}$ w.r.t. $x_i^{(j)}$. Such an oracle that generates an exact value of g and its subgradient will be referred to as *exact*.

The standard Benders algorithm requires calling the exact oracle $|\mathcal{I}|$ times at each iteration j. Consequently, when the number of subproblems is large and the exact oracle is hard to solve (e.g., the problem (2) is very large), the computational efficiency of these methods is badly affected. This motivates the investigation of Benders-type algorithms able to exploit the information of *inexact* oracles. An inexact oracle only provides an estimate of $\theta_i^{(j)}$ and $\lambda_i^{(j)}$, but possibly much faster than an exact oracle.

Inexact oracles can have different characteristics. They may or may not guarantee to generate valid bounds and cutting planes. Additionally it may or may not be possible to control the accuracy of the oracle, and bounds on the approximation error may or may not be known. An example of inexact oracle is the one proposed in [11]. The authors suggest solving only a subset $\mathcal{I}_{ex} \subset \mathcal{I}$ of subproblems and using fast techniques to evaluate approximate values of $\theta_i^{(j)}$ and $\lambda_i^{(j)}$ for subproblems $i \in \mathcal{I} \setminus \mathcal{I}_{ex}$. Using a concept of collinearity the authors group similar subproblems, solve one subproblem per group, and derive an approximate solution for the remaining subproblems of each group. Similar inexact oracles that do not provide a guaranteed bound can be found in [7,14,8,9]. Zakeri et al. [17] propose solving each subproblem $i \in \mathcal{I}$ up to a predefined tolerance which is then tightened over time to ensure convergence. The generated cutting planes are all valid and asymptotically exact. Similar inexact but always valid oracles are proposed by [5,6]. The use of any of the above types of oracles leads to algorithms that are fast at the early iterations but are potentially almost as slow as exact oracles towards the end of the iterative procedure. As a matter of fact, in order to obtain high accuracy, the approach of [11] would need to solve almost all the set of subproblems (i.e., $\mathcal{I}_{ex} \approx \mathcal{I}$), and the inexact oracle of [17] would solve the subproblems up to a very tight tolerance.

An alternative approach to constructing inexact oracles is to exploit some known properties of the subproblems that allow valid cuts to be generated for subproblems from solution of different problems. This approach has been used

in stochastic dual dynamic programming (Pereira et al. [12]), for example by exploiting convexity of the recourse function w.r.t. the uncertain parameters. Stochastic dual dynamic programming is also used to solve minimax problems, which can arise from the inclusion of risk-aversion within the model. This leads to saddle functions similar to $g(x_i, c_i)$, i.e., convex in some directions (x_i) and concave in others (c_i). Philpott et al. [13] exploit this property and construct an upper bound on the recourse function using an inner approximation. However, an inner approximation is infeasible if the point of interest is not the convex hull of known points, so the authors propose to initially computing a solution for every extreme point of the uncertainty set. To avoid this extra work [2] proposes using penalties.

1.2 Approach and Contributions

This paper proposes two new inexact oracles, which we refer to as adaptive oracles. The first adaptive oracle approximates $g(x_i, c_i)$ from below and it is called to generate inexact but valid cuts of those subproblems that are not solved at an iteration j. The second adaptive oracle approximates $g(x_i, c_i)$ from above and it is called to obtain a valid upper bound when a subproblem is not solved. The adaptive oracles exploit properties of $g(x_i, c_i)$ such as convexity w.r.t. x_i and concavity w.r.t. c_i. In addition, they require $g(x_i, c_i)$ to be a monotonic function of both x_i and c_i. The adaptive oracles use the knowledge of m solutions of $g(x_i, c_i)$, known by having solved some subproblems in the previous iterations. Increasing the number m of known solutions, makes both oracles progressively "adapt" towards the true function $g(x_i, c_i)$.

The proposed adaptive oracles are asymptotically exact oracles, since they provide valid upper and lower bounds of $g(x_i, c_i)$, a valid subgradient, and both bounds tend toward $g(x_i, c_i)$ as the number of iterations grows. However, they have properties that combined distinguish them from the inexact oracles available in the literature [17,7,14,8,6,11,10,16,1,9]. First, the computational effort required to solve the adaptive oracles is independent of the size and complexity of the exact oracle (2), and only depends on the number m of known solutions. Second, a Benders-type algorithm that uses the adaptive oracles converges to an ϵ-optimal solution ($\epsilon \geq 0$) in a finite number of iterations even when only a single subproblem is solved at each iteration. As a consequence, when subproblems are expensive to solve and the set $\mathcal{I}$ of subproblems is large, each iteration is much faster than methods that solve every subproblem at each iteration, and we show this can lead to a significant reduction in total solution time. We test our new method on stochastic LP investment planning problems with up to 1.06×10^8 variables and 2.05×10^8 constraints. Compared to the standard Benders decomposition, the proposed algorithm is 25.2, 18.4, and 11.6 times faster at reaching an optimality tolerance ϵ of 1.00%, 0.10%, and 0.01%, respectively.

1.3 Paper Structure

The remainder of the paper is organized as follows. Section 2 introduces assumptions on problems (1) and (2), needed to apply our adaptive oracles. Section 3 briefly presents a standard formulation of the Benders decomposition algorithm. Section 4 illustrates the intuitions behind the proposed adaptive oracles and derives the associated mathematical formulation. The adaptive oracles are embedded within a Benders decomposition algorithm in Section 5. Section 6 tests the proposed Benders with adaptive oracles against a standard Benders algorithm on a stochastic investment planning problem. Finally, conclusions are drawn in Section 7.

2 Assumptions

We consider solving problems of the form of (1). The subproblem **SP** in (2) is assumed to be feasible and bounded for each decision $\mathbf{x}$ that belongs to $\mathcal{X}$ and for all c_i. Accordingly, we say that the function $g(x, c)$ can be computed for all $x \in \mathcal{K}^x$ and $c \in \mathcal{K}^c$, where the region $\mathcal{K}^x$ is obtained by collecting all the possible x_i such that $\mathbf{x} \in \mathcal{X}$, and the region $\mathcal{K}^c$ collecting all c_i for all $i \in \mathcal{I}$. Then, we assume that there exist two vectors $\underline{x}$ and $\underline{c}$ such that $\underline{x} \leq x$, $\forall x \in \mathcal{K}^x$, and $\underline{c} \leq c$, $\forall c \in \mathcal{K}^c$, and that subproblem **SP** is feasible and bounded at the special point $(\underline{x}, \underline{c})$. If the subproblem is not feasible for all $x \in \mathcal{K}^x$ and $c \in \mathcal{K}^c$, and/or at $(\underline{x}, \underline{c})$, we assume infeasibility is dealt with by penalties in the subproblem.

Function $g(x, c)$ is a convex function of x and a concave function of c (e.g., see [4]). In addition, we assume that $g(x, c)$ is a non-increasing function of x, and a non-decreasing function of c. If the property of monotonicity does not hold naturally, problem (1) can be usually modified so that the rearranged subproblem is a monotonic function of both x and c (see Appendix A).

3 Standard Benders

This section briefly describe how a Benders algorithm can exploit the block structure of problem (1) allowing it to be solved in a decomposed fashion.

3.1 Algorithm

In a standard Benders algorithm applied to problem (1), we iteratively approximate the subproblem cost function through a set of cutting planes. The formulation of the relaxed master problem **RMP** is given in (3) and the standard Benders decomposition is summarized in Algorithm 1.

Algorithm 1: `Stand_Bend` (Standard Benders)

choose $\epsilon \geq 0$ (convergence tolerance);
choose $\underline{\beta}$ (a priori lower bound for each β_i);
set $j := 0$ and $\Theta_i^{(0)} := \{(\underline{\beta}, 0, 0)\}$ for each $i \in \mathcal{I}$;
repeat
  set $j := j + 1$;
  solve **RMP** and obtain $\mathbf{x}^{(j)}$ and $\beta^{(j)}$;
  set $\mathrm{L}^{(j)} := f(\mathbf{x}^{(j)}) + \sum_{i \in \mathcal{I}} \pi_i \beta_i^{(j)}$;
  for $i \in \mathcal{I}$ **do**
    solve subproblem i at $(x_i^{(j)}, c_i)$ and obtain $\theta_i^{(j)}$ and $\lambda_i^{(j)}$;
  end
  set $\mathrm{U}^{(j)} := \min\Big(\mathrm{U}^{(j-1)}, f(\mathbf{x}^{(j)}) + \sum_{i \in \mathcal{I}} \pi_i \theta_i^{(j)}\Big)$;
  for $i \in \mathcal{I}$ **do**
    $\Theta_i^{(j)} := \Theta_i^{(j-1)} \cup \{(x_i^{(j)}, \theta_i^{(j)}, \lambda_i^{(j)})\}$;
  end
until $\mathrm{U}^{(j)} - \mathrm{L}^{(j)} \leq \epsilon$;

3.2 Convergence of Algorithm 1

Definition 1 Let $\mathcal{L}_i$ be the set of cutting planes $\{(x_i^{(\ell)}, \theta_i^{(\ell)}, \lambda_i^{(\ell)}), \forall \ell \in \mathcal{L}_i\}$ associated with subproblem i already added to the relaxed master problem. We say that a new exact cut (x, θ, λ) is *locally-improving* if and only if

$$\theta > \max_{\ell \in \mathcal{L}_i} \left\{ \theta_i^{(\ell)} + \lambda_i^{(\ell)\top} \Big(x - x_i^{(\ell)} \Big) \right\}.$$

Lemma 1 *At each iteration, either Algorithm 1 terminates with an ϵ-optimal solution, or the algorithm adds at least one locally-improving exact cut to the reduced master problem.*

Proof If none of the exact cuts $(x_i^{(j)}, \theta_i^{(j)}, \lambda_i^{(j)})$ generated at iteration j is locally-improving, it follows that

$$\theta_i^{(j)} \leq \max_{\ell < j} \left\{ \theta_i^{(\ell)} + \lambda_i^{(\ell)\top} \Big(x_i^{(j)} - x_i^{(\ell)} \Big) \right\}, \quad \forall i \in \mathcal{I}, \tag{4}$$

and given that the right side of (4) is equal to $\beta_i^{(j)}$, it follows that the lower bound and the upper bound have converged exactly. □

Lemma 2 *There exists a finite number of locally-improving exact cutting planes that can be added to the relaxed master problem.*

Proof Since each subproblem $g(x_i, c_i)$ is an LP, there exist a finite number of faces (vertices, edges, ..., facets), and each exact cutting plane gives an exact

representation of (at least) one of these faces. A new exact cut can be locally-improving if and only if it is associated with a face that has not been exactly represented yet. Then, given that the number $|\mathcal{I}|$ of subproblems is finite, it follows that there exists a finite number of locally-improving exact cuts that can be added to the relaxed master problem. □

Theorem 1 *For any $\epsilon \geq 0$, Algorithm 1 finds an ϵ-optimal solution to problem* (1) *in a finite number of iterations.*

Proof Lemma 2 proves that there exists a finite number of locally-improving exact cuts that can be added to the reduced master problem, and Lemma 1 proves that Algorithm 1 adds at least one locally-improving exact cut at each iteration (or it has converged). It follows that Algorithm 1 finds an ϵ-optimal solution to problem (1) in a finite number of iterations. □

Remark 1 Convergence proofs for Benders decomposition (e.g., see [17]) usually rely on the existence of a finite number of basis matrices for each subproblem. In contrast, Theorem 1 relies on the existence of a finite number of faces and does not require that each exact cut corresponds to a basis matrix.

4 Adaptive Oracles

This section presents the adaptive oracles used within the proposed novel Benders-type decomposition algorithm. Figure 1 illustrates a saddle function $g(x, c)$, convex and non-increasing w.r.t x, and concave and non-decreasing w.r.t. c. This function is used to illustrate the intuition behind the proposed adaptive oracles.

4.1 Graphical interpretation of the Adaptive Oracles

For a subproblem i that is not solved at iteration j, we call two adaptive oracles to retrieve the information needed to perform a Benders-type iteration. The first adaptive oracle provides $\underline{\theta}_i^{(j)}$ and $\underline{\lambda}_i^{(j)}$ such that $(x_i^{(j)}, \underline{\theta}_i^{(j)}, \underline{\lambda}_i^{(j)})$ generates

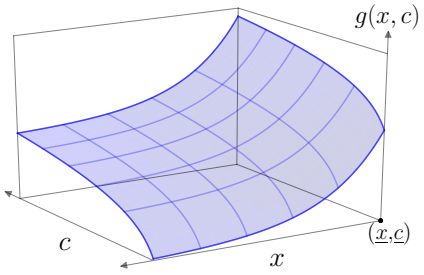

Fig. 1 Illustration of the saddle function $g(x, c)$, convex and non-increasing w.r.t x and concave and non-decreasing w.r.t. c.

a valid cutting plane. The second adaptive oracle yields an upper bound $\overline{\theta}_i^{(j)}$ on $g(x_i^{(j)}, c_i)$, which is then used to compute a valid upper bound on the optimal solution of the relaxed master problem at $\mathbf{x}^{(j)}$.

Before presenting the mathematical formulation of the two adaptive oracles, we give a graphic and explanatory example using the saddle function $g(x, c)$ shown in Figure 1. Figure 2 illustrates how to obtain a valid lower bound on $g(x, c)$, and Figure 3 shows how to obtain a valid upper bound on $g(x, c)$. Assume that we know the exact value $\theta_s = g(x_s, c_s)$ at points (x_s, c_s), shown with blue filled dots in Figures 2a and 3a, and we want to obtain a valid lower and upper bound on $g(x, c)$ at a new point (x_r, c_r), shown with a blue empty dot. Note that Figures 2a and 3a also include the special point $(\underline{x}, \underline{c})$, marked with a blue filled star, which is needed at the end of the examples.

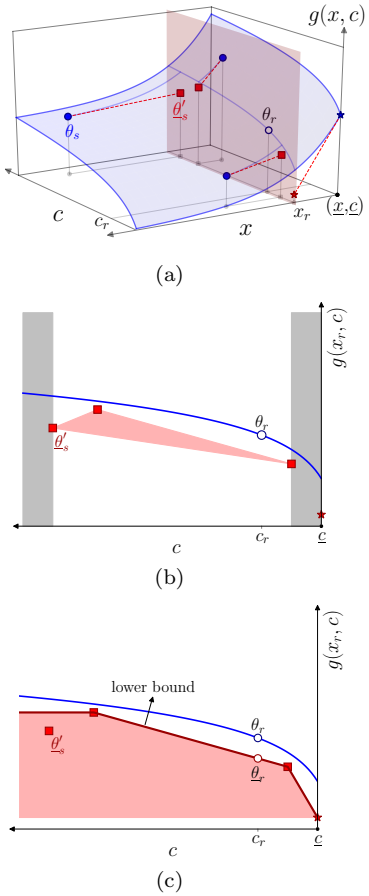

Fig. 2 Illustrative example of how to obtain a valid lower bound $\underline{\theta}_r$ on $\theta_r = g(x_r, c_r)$.

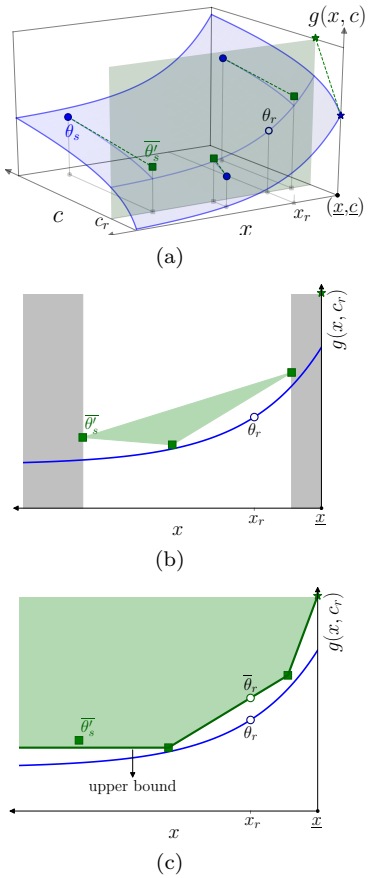

Fig. 3 Illustrative example of how to obtain a valid upper bound $\overline{\theta}_r$ on $\theta_r = g(x_r, c_r)$.

Each red dashed line in Figure 2a shows the tangent at a blue point (x_s, c_s) w.r.t. x (i.e., for fixed c_s). Since $g(x, c_s)$ is convex w.r.t. x, the tangent lies below $g(x, c_s)$. The red squares are the points $(\underline{\theta'_s}, x_r, c_s)$ on the tangent where $x = x_r$. If the gradients are λ_s, then $\underline{\theta'_s} = \theta_s + \lambda_s^\top (x_r - x_s)$. Since $g(x_r, c)$ is concave w.r.t. c and all the red square points lie in $x = x_r$ plane and below $g(x_r, c)$, the convex hull of these points also lies below $g(x_r, c)$. Figure 2b illustrates this convex hull as a red shaded area, and the gray shaded areas indicate values of c where a convex combination can not be found. Figure 2c shows that if the special point $(\underline{x}, \underline{c})$ is included among the known points (x_s, c_s), the non-decreasing property of $g(x, c)$ w.r.t. c can be exploited to extend the convex hull and obtain a valid lower bound for all $c \geq \underline{c}$. In particular, the upper envelope of the convex hull gives the tightest lower bound on $g(x_r, c)$.

The green dashed lines in Figure 3a are the tangents (with gradient ϕ_s) at each blue point (x_s, c_s) w.r.t. c, which lie above $g(x_s, c)$ since $g(x_s, c)$ is concave w.r.t. c. The green squares $(\overline{\theta'_s}, x_s, c_r)$ are the points on the tangents where $c = c_r$, i.e., $\overline{\theta'_s} = \theta_s + \phi_s^\top (c_r - c_s)$. The convex hull of these points lies above $g(x, c_r)$, given that $g(x, c_r)$ is convex w.r.t. x and the green square points lie above $g(x, c_r)$. Figure 3b shows this convex hull as a green shaded area, likewise two gray shaded areas where a convex combination can not be found. If the special point $(\underline{x}, \underline{c})$ is included we extend the convex hull and obtain a valid upper bound for all $x \geq \underline{x}$ by exploiting the non-increasing property of $g(x, c)$ w.r.t. x (see Figure 3c). Here, the lower envelope of the convex hull gives the tightest upper bound on $g(x, c_r)$.

Note that the methodology for obtaining a valid upper bound is the counterpart of the methodology for obtaining a valid lower bound (and vice versa). As an example, one could use the lower bounding oracle to get a valid lower bound on $g(x, c)$ and the same oracle to also obtain a valid upper bound. To do that one can compute a valid lower bound on $\hat{g}(c, x) = -g(x, c)$ and use it (with a change in sign) as a valid upper bound on $g(x, c)$.

4.2 Adaptive Oracles

This section gives a mathematical formulation of the intuitions illustrated in Section 4.1. Consider the saddle function $g(x, c)$, and suppose we have already found m optimal solutions at $\{(x_s, c_s), s = 1, .., m\}$. At each point s, we know the true value θ_s, a subgradient λ_s w.r.t. x, and a subgradient ϕ_s w.r.t. c. We are interested in building a valid cutting plane and obtaining a valid upper bound at a new point (x, c), without solving the associated subproblem.

Adaptive Oracle for valid cutting plane

Let $\mathcal{A}_m^{\mathrm{LB}}(x,c)$ be defined as

$$\begin{aligned} \mathcal{A}_m^{\mathrm{LB}}(x,c): \quad \underline{\theta} \;=\; \max_{\mu_s \geq 0} \quad & \theta = \sum_{s=1}^{m} \mu_s \left(\theta_s + \lambda_s^\top (x - x_s)\right) \\ \text{s.t.} \quad & \sum_{s=1}^{m} \mu_s c_s \leq c, \quad \sum_{s=1}^{m} \mu_s = 1. \end{aligned} \tag{5}$$

Lemma 3 *Assume the special point $(\underline{x},\underline{c})$ is one of the known points $\{(x_s,c_s), s = 1,...,m\}$, and assume μ is a feasible solution of (5) and that $\lambda = \sum_{s=1}^{m} \mu_s \lambda_s$. Then,*

a) $\mathcal{A}_m^{\mathrm{LB}}(x,c)$ is feasible for all $x \in \mathcal{K}^x$, $c \in \mathcal{K}^c$,
b) $\theta \leq \underline{\theta} \leq g(x,c)$, for all $x \in \mathcal{K}^x$, $c \in \mathcal{K}^c$,
c) $\underline{\theta} = g(x_r,c_r)$, for all $r = 1,..,m$,
d) $\theta + \lambda^\top(\hat{x} - x) \leq g(\hat{x},c)$, for all $\hat{x} \in \mathcal{K}^x$.

Proof
a) Set the variable μ_s associated with $(\underline{x},\underline{c})$ equal to 1, and all the others to 0. The first constraint of (5) becomes $\underline{c} \leq c$, which is true by definition, and the other constraints also hold.
b) The definition of θ leads to

$$\begin{aligned} \theta = \sum_{s=1}^{m} \mu_s \left(\theta_s + \lambda_s^\top (x - x_s)\right) \quad &\leq \quad \sum_{s=1}^{m} \mu_s \, g(x,c_s) \\ &\leq \quad g(x, \sum_{s=1}^{m} \mu_s c_s) \\ &\leq \quad g(x,c). \end{aligned}$$

The first inequality holds since $\mu \geq 0$ and $g(x,c)$ is convex w.r.t. x, the second inequality holds since the μ define a convex combination and $g(x,c)$ is concave w.r.t. c, and the third holds since $\sum_{s=1}^{m} \mu_s c_s \leq c$ and $g(x,c)$ is non-decreasing w.r.t. c.
c) Setting $\mu_r = 1$ gives a feasible solution with objective value $\theta_r + \lambda_r^\top (x_r - x_r) = \theta_r$. Since μ_r is feasible for $\mathcal{A}_m^{\mathrm{LB}}(x_r,c_r)$, it follows that $\underline{\theta} \geq \theta_r = g(x_r,c_r)$. This and *b*) imply $\underline{\theta} = g(x_r,c_r)$.
d) Since the weights μ are feasible for $\mathcal{A}_m^{\mathrm{LB}}(x,c)$, they are also feasible for $\mathcal{A}_m^{\mathrm{LB}}(\hat{x},c)$. Hence,

$$\begin{aligned} g(\hat{x},c) \geq \quad & \sum_{s=1}^{m} \mu_s \left(\theta_s + \lambda_s^\top (\hat{x} - x_s)\right) \\ = \quad & \sum_{s=1}^{m} \mu_s \left(\theta_s + \lambda_s^\top (x - x_s)\right) + \sum_{s=1}^{m} \mu_s (\lambda_s^\top (\hat{x} - x)) \\ = \quad & \theta + \lambda^\top (\hat{x} - x)\,. \end{aligned}$$

The first equality follows since

$$\lambda_s^\top(\hat{x} - x_s) = \lambda_s^\top(x - x_s) + \lambda_s^\top(\hat{x} - x)\,,$$

and the second from the definition of θ and λ. □

Adaptive Oracle for valid upper bound

Let $\mathcal{A}_m^{\mathrm{UB}}(x, c)$ be defined as

$$\begin{aligned} \mathcal{A}_m^{\mathrm{UB}}(x,c): \quad \overline{\theta} \;=\; \min_{\nu_s \geq 0} \quad & \theta = \sum_{s=1}^{m} \nu_s \left(\theta_s + \phi_s^\top (c - c_s)\right) \\ \text{s.t.} \quad & \sum_{s=1}^{m} \nu_s x_s \leq x, \quad \sum_{s=1}^{m} \nu_s = 1. \end{aligned} \tag{6}$$

Lemma 4 *Assume the special point $(\underline{x}, \underline{c})$ is one of the known points $\{(x_s, c_s), s = 1, ..., m\}$, and assume ν is a feasible solution of (6). Then,*

a) $\mathcal{A}_m^{\mathrm{UB}}(x, c)$ is feasible for all $x \in \mathcal{K}^x$, $c \in \mathcal{K}^c$,
b) $\theta \geq \overline{\theta} \geq g(x, c)$, for all $x \in \mathcal{K}^x$, $c \in \mathcal{K}^c$,
c) $\overline{\theta} = g(x_r, c_r)$, for all $r = 1, .., m$.

Proof
Lemma 4 can be proved similarly to Lemma 3 (parts a, b, and c) since obtaining a valid upper bound is the exact counterpart than obtaining a valid lower bound. As an example, it is possible to get an upper bound on $g(x, c)$ by calling the lower bound oracle on $\hat{g}(c, x) = -g(x, c)$. □

Notes on adaptive oracles

Note that every feasible solution μ of $\mathcal{A}_m^{\mathrm{LB}}(x, c)$ gives a valid cutting plane (x, θ, λ). Of these possible cuts, the one corresponding to the optimal solution, i.e., $(x, \underline{\theta}, \underline{\lambda})$, is the tightest at x. If $\mathcal{A}_m^{\mathrm{LB}}(x, c)$ and/or $\mathcal{A}_m^{\mathrm{UB}}(x, c)$ are terminated at a feasible but not optimal solution, the generated cutting plane and upper bound are still valid.

A graphical interpretation of $\underline{\theta}$ and θ of $\mathcal{A}_m^{\mathrm{LB}}(x, c)$ is shown in Figure 2c, where the set $(\underline{\theta}, c)$ is the red continuous curve, and the set (θ, c) is the red shaded area. Figure 3c gives an interpretation of $\overline{\theta}$ and θ of $\mathcal{A}_m^{\mathrm{UB}}(x, c)$, where the set $(\overline{\theta}, c)$ is the green continuous curve, and the set (θ, c) is the green shaded area.

5 Benders Decomposition with Adaptive Oracles

This section presents the Benders-type algorithm incorporating the inexact information of the adaptive oracles of Section 4.

5.1 Algorithm

Algorithm 2: `Adapt_Bend` (Benders with Adaptive Oracles)

choose ϵ (convergence tolerance);
choose $\underline{\beta}$ (lower bound for each β_i);
set $j := 0$ and $\Theta_i^{(0)} := \{(\underline{\beta}, 0, 0)\}$ for each $i \in \mathcal{I}$;
solve $g(\underline{x}, \underline{c})$ to obtain θ, λ, and ϕ;
set $\mathcal{S} := \{(\underline{x}, \underline{c}, \theta, \lambda, \phi)\}$;
repeat
  set $j := j + 1$;
  solve **RMP** and obtain $\mathbf{x}^{(j)}$ and $\beta^{(j)}$;
  set $\underline{\mathrm{L}}^{(j)} := f(\mathbf{x}^{(j)}) + \sum_{i \in \mathcal{I}} \beta_i^{(j)}$;
  set $\xi^{(j)} := 0$ and $\mathcal{E}^{(j)} := \mathcal{I}$;
  repeat
    set $\mathcal{I}_{ex} :=$ non-empty subset of $\mathcal{E}^{(j)}$;
    set $\mathcal{E}^{(j)} := \mathcal{E}^{(j)} \setminus \mathcal{I}_{ex}$;
    for $i \in \mathcal{I}_{ex}$ **do**
      solve subproblem i at $(x_i^{(j)}, c_i)$ and obtain $\theta_i^{(j)}$, $\lambda_i^{(j)}$, and $\phi_i^{(j)}$;
      if $(x_i^{(j)}, \theta_i^{(j)}, \lambda_i^{(j)})$ is locally-improving **then** $\xi^{(j)} := 1$;
      set $\mathcal{S} := \mathcal{S} \cup \{(x_i^{(j)}, c_i, \theta_i^{(j)}, \lambda_i^{(j)}, \phi_i^{(j)})\}$;
      set $\underline{\theta}_i^{(j)} := \theta_i^{(j)}$, $\underline{\lambda}_i^{(j)} := \lambda_i^{(j)}$, and $\overline{\theta}_i^{(j)} := \theta_i^{(j)}$;
    end
  until $\xi^{(j)} = 1$ **or** $\mathcal{E}^{(j)} = \varnothing$;
  for $i \in \mathcal{E}^{(j)}$ **do**
    solve $\mathcal{A}^{\mathrm{LB}}_{|\mathcal{S}|}(x_i^{(j)}, c_i)$ and obtain $\underline{\theta}_i^{(j)}$ and $\underline{\lambda}_i^{(j)}$;
    solve $\mathcal{A}^{\mathrm{UB}}_{|\mathcal{S}|}(x_i^{(j)}, c_i)$ and obtain $\overline{\theta}_i^{(j)}$;
  end
  set $\overline{\mathrm{U}}^{(j)} := \min\left(\overline{\mathrm{U}}^{(j-1)}, f(\mathbf{x}^{(j)}) + \sum_{i \in \mathcal{I}} \overline{\theta}_i^{(j)}\right)$;
  for $i \in \mathcal{I}$ **do**
    set $\Theta_i^{(j)} := \Theta_i^{(j-1)} \cup \{(x_i^{(j)}, \underline{\theta}_i^{(j)}, \underline{\lambda}_i^{(j)})\}$
  end
until $\overline{\mathrm{U}}^{(j)} - \underline{\mathrm{L}}^{(j)} \leq \epsilon$;

5.2 Convergence of Algorithm 2

Lemma 5 *At each iteration, either Algorithm 2 finds an ϵ-optimal solution, or the algorithm adds at least one locally-improving exact cut to the reduced master problem.*

Proof
At iteration j Algorithm 2 adds at least one locally-improving exact cut to the reduced master problem ($\xi^{(j)} = 1$) or all the subproblems are solved for the current solution ($\mathcal{E}^{(j)} = \varnothing$). If none of the exact cuts $(x_i^{(j)}, \theta_i^{(j)}, \lambda_i^{(j)})$ generated at iteration j is locally-improving ($\xi^{(j)} = 0$), it follows that

$$\theta_i^{(j)} \leq \max_{\ell<j} \left\{ \underline{\theta}_i^{(\ell)} + \underline{\lambda}_i^{(\ell)\top} \left(x_i^{(j)} - x_i^{(\ell)} \right) \right\}, \quad \forall i \in \mathcal{I}, \tag{7}$$

and given that the left and right side of (7) are equal to $\overline{\theta}_i^{(j)}$ and $\beta_i^{(j)}$, respectively, it follows that the lower bound and the upper bound have converged exactly. □

Theorem 2 *For any $\epsilon \geq 0$, Algorithm 2 finds an ϵ-optimal solution to problem* (1) *in a finite number of iterations.*

Proof
Lemma 2 proves that there exists a finite number of locally-improving exact cuts that can be added to the reduced master problem, and Lemma 5 proves that Algorithm 2 adds at least one locally-improving exact cut at each iteration (or it has converged). It follows that Algorithm 2 finds an ϵ-optimal solution to problem (1) in a finite number of iterations. □

6 Case Study

We test the proposed Benders algorithm with adaptive oracles on power system stochastic investment planning problems. The algorithms are implemented in Julia 1.0 on a Linux Desktop computer Intel i5 4-core processor clocking at 2.00 GHz and 16 GB of RAM. The optimization models are implemented in JuMP (Julia package) and solved with Gurobi 7.5. The Julia code implementing Algorithm 1 (`Stand_Bend`) and 2 (`Adapt_Bend`) for the proposed case study is provided at `https://github.com/nimazzi/Stand_and_Adapt_Bend`.

6.1 Investment Planning Model

We consider a power system investment planning problem with a time horizon of 15 years. The deterministic version of the problem has 3 decision nodes: one refers to decisions to be taken in the first stage, one to decisions in 5 years time, and one to decision in 10 years time. The stochastic version is obtained by modeling different possible scenarios for the future of the system in 5 and 10 years. At each node we also compute the cost of operating the system for the following 5 years for given installed capacity. We consider a construction time of 5 years, so new assets installed in the first stage will only be available in 5 and 10 years, and new capacity installed in 5 years will only be available in 10 years. We model a set $\mathcal{P}$ of technologies: 6 thermal units, 3 storage units, and

3 renewable generation units. The cost for operating the system is computed by solving an hourly economic dispatch for 1 year.

We formulate the stochastic investment planning problem as

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) + \sum_{i\in\mathcal{I}} \pi_i\, g(x_i, c_i), \tag{8}$$

where $\mathcal{I}$ is the set of stochastic decision nodes, each associated with a probability π_i. The function

$$g(x_i, c_i) = \min_{y_i\in\mathcal{Y}} \{c_i^\top C y_i \mid Ay_i \le Bx_i\}, \quad \forall i \in \mathcal{I}, \tag{9}$$

gives the cost of operating the system over 5 years. The vector of coefficients x_i is given by

$$x_i = \left(\left\{x_{pi}^{acc}, \forall p \in \mathcal{P}\right\}, -\nu_i^D, \nu_i^E\right), \quad \forall i \in \mathcal{I},$$

where x_{pi}^{acc} is the accumulated capacity of technology p at node i. Parameters ν_i^D and ν_i^E are the relative level of energy demand and the yearly CO_2 emission limit, respectively. Note that $g(x_i, c_i)$ is non-increasing w.r.t. x_i^{acc} and ν_i^E, and non-decreasing w.r.t. ν_i^D, so using $-\nu_i^D$ instead satisfies the non-increasing requirement. The vector of uncertain cost coefficients c_i is

$$c_i = (c_i^{nucl}, c_i^{\text{co}_2}), \quad \forall i \in \mathcal{I},$$

where c_i^{nucl} is the uranium fuel price and $c_i^{\text{co}_2}$ the CO_2 emission price, and $g(x_i, c_i)$ is non-decreasing w.r.t. both c_i^{nucl} and $c_i^{\text{co}_2}$. All the remaining parameters, e.g., A, B, and C, are the same for every node $i \in \mathcal{I}$. Finally, the function $f(\mathbf{x})$ yields the expected total investment and fixed cost, and it is computed as

$$f(\mathbf{x}) = \sum_{i\in\mathcal{I}} \pi_i \sum_{p\in\mathcal{P}} \left(c_{pi}^{inv} x_{pi}^{inst} + c_{pi}^{fix} x_{pi}^{acc}\right),$$

where the variable x_{pi}^{inst} is the newly installed capacity of technology p at node i. Parameters c_{pi}^{inv} and c_{pi}^{fix} are the unitary investment and fixed costs of technology p at node i. The accumulated capacity x_{pi}^{acc} at node i is computed as the sum of the historical capacity x_{pi}^{hist} and the newly installed capacity $x_{pi'}^{inst}$ in all nodes i' ancestors to i. Finally, the initial special point $(\underline{x}, \underline{c})$ is set to

$$\underline{x}, \underline{c} = \left(\left\{\min_i x_{pi}^{hist}, \forall p \in \mathcal{P}\right\}, -\max_i \nu_i^D, \min_i \nu_i^E\right), \left(\min_i c_i^{nucl}, \min_i c_i^{\text{co}_2}\right)$$

We consider three possible sources of uncertainty, i.e., ν_i^E, $c_i^{\text{co}_2}$, and c_i^{nucl}. Each uncertain parameters has 3 possible outcomes in 5 years, each of which is linked to 3 additional possible outcomes in 10 years. The result is 9 possible trajectories for each uncertainty, all with the same probability. We consider 4 different cases of the investment problem. Case 1 is the deterministic version, where ν_i^E, $c_i^{\text{co}_2}$, and c_i^{nucl} are deterministic parameters (weighted average of the scenarios). Then, case 2 has 1 uncertain parameter (ν_i^E), case 3 has 2 uncertain parameters (ν_i^E and $c_i^{\text{co}_2}$), and case 4 has 3 uncertain parameters (ν_i^E, $c_i^{\text{co}_2}$, and c_i^{nucl}). The number of decision nodes for the 4 versions of the problem is summarized in Table 1.

Table 1 Decision nodes $\mathcal{I}$ in different cases of the study.

case	number of decision nodes			$\lvert\mathcal{I}\rvert$
	present	in 5 years	in 10 years	
1	1	1	1	3
2	1	3	9	13
3	1	9	81	91
4	1	27	729	757

6.2 Results

We use Algorithm 1 (`Stand_Bend`) and 2 (`Adapt_Bend`) to solve the stochastic investment planning problem (8), whose operational subproblems (9) are LP problems with 2.72×10^5 constraints and 1.40×10^5 variables. For Algorithm 2 we use the following rule for choosing $\mathcal{I}_{ex}$ at each iteration:

$$\mathcal{I}_{ex} := \left\{ i \,\middle|\, i = \arg \max_{i \in \mathcal{E}^{(j)}} \ \pi_i \left(\overline{\theta}_i^{(j-1)} - \underline{\theta}_i^{(j-1)} \right) \right\}, \tag{10}$$

i.e., we solve a single subproblem for which the gap $\pi_i \overline{\theta}_i^{(j-1)} - \pi_i \underline{\theta}_i^{(j-1)}$ was the greatest at the previous iteration. On the other hand, Algorithm 1 solves $|\mathcal{I}| - 1$ subproblems at each iteration (the operational cost in the first stage is not affected by any investment decision and can be avoided).

Table 2 gives the optimal investment decisions to be taken in the first investment stage for cases 1-4. The solutions are obtained solving (8) with Algorithm 1 up to an 0.01%-optimal solution. Including a more accurate description of the stochastic processes involved in the decision-making process progressively modifies the optimal decisions of the system planner. Using the full stochastic model (case 4) the planner takes considerably different decisions compared to the ones yield by its deterministic equivalent (case 1). In case 1 the system planner builds 13.9 GW of CCGT, 1.2 GW of diesel, and 19 GW of onshore wind. In case 4 the investment in CCGT (11.7 GW) is less and there is a slight increase in the amount of diesel (1.9 GW). In addition, the planner invests in 1 GW of coal with CCS which is not used at all in case 1.

Note that the asymmetry between large investments in wind power generation (19 GW) in all cases and no investment in storage technologies is probably due to the significant amount of storage (8 GW in total) already available in the system. In addition, note that other technologies (e.g., nuclear and offshore wind) are chosen in many scenarios of the second investment stage (in 5 years) and that most of the technologies are explored during the iterative solution procedure, even if these are not used in the optimal solution.

Table 3 shows the effort in terms of iterations and computation time (the counter starts after the Julia code has been precompiled) to reach an ϵ-optimal solution using Algorithm 1 and Algorithm 2 for case studies 1-4. We report the results for values of the optimality tolerance ϵ of 1.00%, 0.10%, and 0.01%. For case 1 the computation efficiencies of Algorithm 1 and Algorithm 2 are similar. In this small case study Algorithm 1 solves 2 subproblems

Table 2 Optimal investments in the first investment stage for cases 1-4.

tech. $p \in \mathcal{P}$	type	new installed capacity (MW) case 1	case 2	case 3	case 4
coal	thermal	0	0	0	0
coal&CCS	thermal	0	0	901	1041
OCGT	thermal	0	0	0	0
CCGT	thermal	13941	13378	11689	11715
diesel	thermal	1162	1723	2065	1898
nuclear	thermal	0	0	0	0
pumpL	storage	0	0	0	0
pumpH	storage	0	0	0	0
lithium	storage	0	0	0	0
onshore Wind	renewable	19000	19000	19000	19000
offshore Wind	renewable	0	0	0	0
PV solar	renewable	0	0	0	0

each iteration and Algorithm 2 solves 1 subproblem. However, Algorithm 2 needs more iterations to reach the target tolerance, given that it also uses inexact information. As an example, to obtain a tolerance lower than 0.01% Algorithm 1 requires 35 iterations and Algorithm 2 needs 41. Increasing the size of the problem makes the comparison more interesting. For example, for case 3 Algorithm 1 only needs 14 and 24 iterations to reach tolerance of 1.00% and 0.01%, respectively. To reach the same tolerances Algorithm 2 performs 55 and 197 iterations, respectively. However, an iteration of Algorithm 1 solves 90 subproblems while Algorithm 2 solves only 1 and the results show that Algorithm 1 takes 90 minutes to yield an 0.01%-optimal solution compared to less than 11 minutes for Algorithm 2, i.e., 8.5 times faster. For case 4 this difference is even larger as Algorithm 1 calls the exact oracle 756 times at each iteration. Accordingly, even if it reaches a 1.00% tolerance in only 12 iterations, this requires 5 hours and 26 minutes. On the other hand, Algorithm 2 reaches the same tolerance in less than 13 minutes, 25.2 times faster than Algorithm 1. If a tighter tolerance is needed the difference of performance between Algorithm 1 and 2 progressively reduces but it is still significant for $\epsilon = 0.01\%$. Indeed, Algorithm 1 requires around 9 hours and 43 minutes while Algorithm 2 needs 50 minutes (11.6 times faster).

Table 4 shows the (relative) computation time of the main steps of Algorithm 2, i.e., solving the relaxed master problem, solving the subproblem, and solving adaptive oracles, to reach an ϵ-optimal solution for case studies 1-4. We report the results for value of the optimality tolerance ϵ of 1.00%, 0.10%, and 0.01%. In cases 1, 2, and 3 almost all the computation time is spent solving subproblems (one each iteration). However, in case 4 almost 18% of the computation time to obtain an 0.01%-optimal solution is spent solving the master problem, 21% solving the adaptive oracles, and only 61% solving subproblems. These results suggest that the proposed rule (10) of solving one subproblem at each iteration may not be the best when the number $|\mathcal{I}|$ of subproblems grows

Table 3 Comparative results for Algorithm 1 (`Stand_Bend`) and Algorithm 2 (`Adapt_Bend`).

		`Stand_Bend`		`Adapt_Bend`		
	ϵ-target (%)	iters	time (s)	iters	time (s)	time ratio
case 1	1.00	17	76	20	43	1.8
	0.10	26	121	26	66	1.8
	0.01	35	168	41	105	1.6
case 2	1.00	13	329	42	123	2.7
	0.10	16	416	48	139	3.0
	0.01	22	619	74	258	2.4
case 3	1.00	14	2853	55	151	18.9
	0.10	17	3569	96	289	12.3
	0.01	24	5395	197	638	8.5
case 4	1.00	12	19565	207	775	25.2
	0.10	16	27033	345	1469	18.4
	0.01	20	34986	596	3013	11.6

significantly and that a selection rule that dynamically decides the number of subproblems to solve at each iteration may then achieve a better balance.

Table 4 Analysis of Algorithm 2 (`Adapt_Bend`) computation time.

		Relative Computation Time		
	ϵ-target (%)	master problem (%)	exact oracle (%)	adaptive oracles (%)
case 1	1.00	0.02	99.94	0.04
	0.10	0.02	99.95	0.03
	0.01	0.02	99.94	0.04
case 2	1.00	0.05	99.83	0.12
	0.10	0.05	99.82	0.12
	0.01	0.05	99.83	0.12
case 3	1.00	0.57	98.56	0.84
	0.10	0.60	98.42	0.95
	0.01	0.87	97.65	1.45
case 4	1.00	8.72	79.43	11.67
	0.10	12.57	72.30	14.94
	0.01	17.74	61.09	21.02

7 Conclusions

This paper presents a novel concept of inexact oracles that can be used to speed up the computational time of Benders-type decomposition algorithms

for a class of large scale optimization problems. We propose two adaptive oracles that yield inexact and progressively more accurate information when subproblems are not solved. The first adaptive oracle builds valid cutting planes, and the second adaptive oracle provides a valid upper bound of the objective function. The two oracles exploit properties of the Benders subproblem such as convexity w.r.t. right-hand side coefficients and concavity w.r.t. cost coefficients, as well as monotonicity w.r.t. to both coefficients.

We use the novel adaptive oracles within a Benders-type decomposition algorithm to solve a stochastic investment planning problem. The results show substantial improvements in terms of computational efficiency (w.r.t. a standard Benders algorithm), especially if a large optimality gap is allowed. The largest problem we solve has 756 subproblems, each of which has 2.72×10^5 linear constraints and 1.40×10^5 linear variables. Our algorithm is 25.2 times faster than the standard Benders algorithm to reach a 1.00% optimal solution and 11.6 times faster if the optimality tolerance is tighten to 0.01%.

A Reformulation of achieve monotonicity w.r.t. x and c.

We consider the case when $g(x,c) := \min_{y \in \mathcal{Y}} \{c^\top C y \mid Ay \leq Bx\}$ is not a monotonic function of x and c. First, we formulate a modified version of $g(x,c)$ as

$$\begin{aligned} \tilde{g}(x,c) = \quad & \min_{y \in \mathcal{Y}} \quad c^\top C y + \gamma^\top x \\ & \text{s.t.} \quad Ay \leq Bx \end{aligned} \tag{11}$$

and choose a value of γ that makes the objective function $c^\top C y + \gamma^T x$ monotonic w.r.t. x. The master function objective is changed to $f(\mathbf{x}) - \sum_{i \in \mathcal{I}} \gamma^T x_i$ to cancel out this additional term. Then, we formulate a relaxed version of $\tilde{g}(x,c)$ as follows

$$\begin{aligned} \tilde{g}_{\mathrm{c}}(x, c^\alpha, c^\beta) = \quad & \min_{y \in \mathcal{Y}, z^\alpha, z^\beta} \quad c^{\alpha\top} z^\alpha + c^{\beta\top} z^\beta + \gamma^\top x \\ & \text{s.t.} \quad z^\alpha - z^\beta = Cy, \quad Ay \leq Bx \\ & \qquad\quad 0 \leq z^\alpha \leq Z^\alpha, \quad 0 \leq z^\beta \leq Z^\beta \end{aligned} \tag{12}$$

where parameters Z^α and Z^β ensures that the relaxed subproblem is not unbounded. The value of Z^α and Z^β is to be chosen so that they do not restrict the feasible region w.r.t. (11). Note that $\tilde{g}_{\mathrm{c}}(x, c^\alpha, c^\beta)$ is a concave and increasing function of both c^α and c^β, and that $\tilde{g}_{\mathrm{c}}(x, c^\alpha, c^\beta) = \tilde{g}(x,c)$ if $c^\alpha = c$ and $c^\beta = -c$. Computing $\tilde{g}_{\mathrm{c}}(\underline{x}, \underline{c}, \underline{c})$ gives a solution that can be used in the adaptive oracle for obtaining a valid lower bound on $\tilde{g}(x,c)$, $\forall x \in \mathcal{K}^x, \forall c \in \mathcal{K}^c$.

References

1. van Ackooij, W., Frangioni, A., de Oliveira, W.: Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support. Computational Optimization and Applications **65**(3), 637–669 (2016)
2. Baucke, R., Downward, A., Zakeri, G.: A deterministic algorithm for solving stochastic minimax dynamic programs. Optimization Online (2018). URL `http://www.optimization-online.org/DB_HTML/2018/02/6449.html`
3. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**(1), 238–252 (1962)

4. Birge, J.R., Louveaux, F.: Introduction to stochastic programming. Springer Science & Business Media (2011)
5. Fábián, C.I.: Bundle-type methods for inexact data. Central European Journal of Operations Research **8**(1), 35–55 (2000)
6. Fábián, C.I., Szőke, Z.: Solving two-stage stochastic programming problems with level decomposition. Computational Management Science **4**(4), 313–353 (2007)
7. Hintermüller, M.: A proximal bundle method based on approximate subgradients. Computational Optimization and Applications **20**(3), 245–266 (2001)
8. Kiwiel, K.C.: A proximal bundle method with approximate subgradient linearizations. SIAM Journal on Optimization **16**(4), 1007–1023 (2006)
9. Malick, J., de Oliveira, W., Zaourar, S.: Uncontrolled inexact information within bundle methods. EURO Journal on Computational Optimization **5**(1-2), 5–29 (2017)
10. Oliveira, W., Sagastizábal, C.: Level bundle methods for oracles with on-demand accuracy. Optimization Methods and Software **29**(6), 1180–1209 (2014)
11. Oliveira, W., Sagastizábal, C., Scheimberg, S.: Inexact bundle methods for two-stage stochastic programming. SIAM Journal on Optimization **21**(2), 517–544 (2011)
12. Pereira, M.V., Pinto, L.M.: Multi-stage stochastic optimization applied to energy planning. Mathematical Programming **52**(1-3), 359–375 (1991)
13. Philpott, A., de Matos, V., Finardi, E.: On solving multistage stochastic programs with coherent risk measures. Operations Research **61**(4), 957–970 (2013)
14. Solodov, M.V.: On approximations with finite precision in bundle methods for nonsmooth optimization. Journal of Optimization Theory and Applications **119**(1), 151–165 (2003)
15. Van Slyke, R.M., Wets, R.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM Journal on Applied Mathematics **17**(4), 638–663 (1969)
16. Wolf, C., Fábián, C.I., Koberstein, A., Suhl, L.: Applying oracles of on-demand accuracy in two-stage stochastic programming–a computational study. European Journal of Operational Research **239**(2), 437–448 (2014)
17. Zakeri, G., Philpott, A.B., Ryan, D.M.: Inexact cuts in Benders decomposition. SIAM Journal on Optimization **10**(3), 643–657 (2000)