Quadratic stabilization of Benders decomposition

Sofia Zaourar, Jérôme Malick

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

Quadratic stabilization of Benders decomposition

Sofia Zaourar · **Jérôme Malick**

Received: date / Accepted: date

Abstract The foundational Benders decomposition, or variable decomposition, is known to have the inherent instability of cutting plane-based methods. Several techniques have been proposed to improve this method, which has become the state of the art for important problems in operations research. This paper presents a complementary improvement featuring quadratic stabilization of the Benders cutting-plane model. Inspired by the level-bundle methods of nonsmooth optimization, this algorithmic improvement is designed to reduce the number of iterations of the method. We illustrate the interest of the stabilization on two classical problems: network design problems and hub location problems. We also prove that the stabilized Benders method has the same theoretical convergence properties as the usual Benders method.

Keywords Benders decomposition · nonsmooth optimization · quadratic stabilization · bundle method · mixed-integer programming · network design problems · hub location problems · convex analysis

1 Introduction

Benders decomposition, or variable decomposition, is a fundamental method of operations research that is adapted to problems where fixing some "complicating" variables makes the problem much easier to solve. The approach consists in decomposing the initial optimization problem into a sequence of two problems: a master problem in the complicating variables producing the next iterate, and a subproblem in the easy variables generating a new constraint (or cut) for the master. Originally proposed by (Benders, 1962) for linear programming, the method was generalized to nonlinear

S. Zaourar
Université Joseph Fourier, INRIA, Grenoble, France
E-mail: sofia.zaourar@inria.fr

J. Malick
CNRS, Lab. J. Kuntzmann, Grenoble, France
E-mail: jerome.malick@inria.fr

programming by (Geoffrion, 1972) and then refined and adapted to various mixed-integer programs (see e.g. (Conejo et al, 2006)). Successful Real-world applications include network design (see the survey (Costa, 2005) and recent examples (Fortz and Poss, 2009; Contreras et al, 2011)), locomotive and car assignment (Cordeau et al, 2000), aircraft routing and crew scheduling (Mercier et al, 2005).

It is well-known, however, that the application of the classical Benders decomposition sometimes leads to excessively slow convergence and long computing times; see e.g. (Geoffrion and Graves, 1974; McDaniel and Devine, 1977; Magnanti and Wong, 1981; Saharidis and Ierapetritou, 2010; Naoum-Sawaya and Elhedhli, 2013). In particular, forty years ago, (Geoffrion and Graves, 1974) already observed that the problem formulation strongly affects the performance of Benders method. Similarly, (Magnanti and Wong, 1981) noted that a straightforward application of Benders method to some network design problems leads to poor performance.

Several techniques have been proposed to deal with this phenomenon and to accelerate the standard Benders method. These techniques mainly split into two categories: reducing the cost of each iteration, or reducing the number of iterations. First, one can achieve cheaper iterations by reducing the time spent solving the master problem or the subproblem. Among the first references that attempt to make the master problem easier to solve are: (McDaniel and Devine, 1977) which relaxes the master problem at most iterations, and (Côté and Laughton, 1984) which solves the master problem only approximately. The idea of solving the subproblems approximately (resulting in inexact cuts) has also been successfully used in e.g. (Zakeri et al, 2000; Oliveira et al, 2011). Second, the main idea to reduce the number of iterations is to generate more or "better" cuts. A standard technique, going back to (McDaniel and Devine, 1977), is to add an initial set of valid cuts to the master problem in order to restrict its feasible region. Most of the research done to accelerate Benders method focuses on generating more efficient cuts. In particular, (Magnanti and Wong, 1981) presented a multi-cut approach, introducing the so-called Pareto-optimal cuts; (Saharidis et al, 2010) proposed a strategy to generate a bundle of cuts involving most of the variables. In the case where a lot of feasibility cuts are needed, (Saharidis and Ierapetritou, 2010) constructs new optimality cuts from infeasible subproblems. Finally, another recent and fruitful idea is to combine Benders decomposition with branching strategies. For example, (Rei et al, 2009) proposed a local branching approach to be used within Benders and (Naoum-Sawaya and Elhedhli, 2013) exploits warm-starting using the Benders method within a branch-and-cut framework.

This paper brings a new tool to accelerate Benders decomposition: we propose an algorithmic improvement, complementary to existing techniques, to reduce the number of iterations. More specifically, looking at the Benders method from a nonsmooth optimization point of view (Hiriart-Urruty and Lemaréchal, 1993), we introduce a quadratic stabilization, inspired by bundle methods to stabilize and accelerate the Benders decomposition. Quadratic stabilizations of the Dantzig-Wolfe decomposition have already been studied: the review paper (Briant et al, 2008) shows the applicability and the interest of the approach on several mixed-integer problems. Quadratic stabilizations have also been introduced for Benders decomposition with no integer variables, as in particular stochastic two-stage problems appealing for scenario decomposition (see e.g. (Ruszczyński, 1986) and (Oliveira et al, 2011)). To the best of

our knowledge, quadratic stabilizations have not been adapted to the case of Benders decomposition for mixed-integer problems. In this case indeed, the situation is much different from Dantzig-Wolfe or continuous Benders, due to the presence of integer variables in the master problem.

This methodological paper is organized as follows. In Section 2, we revisit the classical Benders decomposition from a nonsmooth optimization perspective, laying the ground for our developments. We follow the abstract framework of the generalized Benders decomposition (Geoffrion, 1972). Section 3 introduces our main contribution: the stabilized Benders method, using the quadratic stabilization of level bundle methods (Lemaréchal et al, 1995). We prove in Section 4 that the convergence properties of the stabilized method are (almost) the same as those of the original one. Finally, in Section 5, we illustrate the relevance of the stabilization for two classical classes of problems: network design and hub location problems.

2 Benders decomposition from a nonsmooth optimization point of view

This section recalls classical Benders decomposition and introduces our notation. We revisit here the standard approach from a nonsmooth optimization point of view. Though this viewpoint is part of the folklore, it has never been precisely formalized. In particular, the underlying "oracle" is discussed in Section 2.1 and illustrated in Section 2.2. We recall in Section 2.3 the structure of the Benders cutting-plane algorithm.

2.1 Framework and convexity assumptions

Following the notation of (Geoffrion, 1972), we consider the optimization problem

$$\begin{array}{ll} \min & f(x,y) \\ \text{s.t.} & G(x,y) \leq 0 \\ & x \in X,\, y \in Y, \end{array} \tag{1}$$

where $X \subseteq \mathbb{R}^p$, $Y \subseteq \mathbb{R}^q$ and $G(\cdot,\cdot)$ is an $\mathbb{R}^m$ valued function. We do not make any assumptions on the constraints sets X and Y; in particular, Y can be discrete, of the form of the intersection of a polyhedral set and the integers $\mathbb{Z}^p$. We only assume that (1) is feasible, i.e. its optimal value is not $-\infty$.

Consider the situation where the y variables are the “complicating” variables in (1), in the sense that when temporarily fixed, the remaining subproblem is considerably more tractable. In this case, it is natural to decompose the problem, into two levels, by considering the function

$$\begin{array}{rll} v(y) := \inf & f(x,y) \\ \text{s.t.} & G(x,y) \leq 0 \\ & x \in X, \end{array} \tag{2}$$

and writing the problem (1) as

$$v^* := \min_{y \in Y} v(y) \qquad \text{or} \qquad v^* := \min_{y \in Y \cap V} v(y). \tag{3}$$

In the minimization problem above, V denotes the domain of v, i.e. the values y such that $v(y)$ is finite, namely:

$$V := \{y \in \mathbb{R}^q : (2) \text{ is feasible}\} = \{y \in \mathbb{R}^q : \exists x \in X, G(x,y) \leq 0\}.$$

Benders decomposition exploits this structure by considering a sequence of two simpler problems: a relaxation of (3) (called master problem) and a subproblem of the form (2). The resulting algorithm is detailed in Section 2.3. Let us emphasize here that the approach requires some "underlying convexity", formalized by the following assumptions.

Assumptions (Convexity assumptions)

(i) The function v is closed and convex (and as a result its domain V is a closed convex set).
(ii) For all $y \in V$, we can compute (approximately) the value and a subgradient of v at y.
(iii) For all $y \notin V$, we can compute a hyperplane separating y from V.

Before giving examples in the next section, note that assumption (i) is mandatory to ensure the consistency of Benders decomposition. Indeed, when v is not convex, (Sahinidis and Grossmann, 1991) provides an example where Benders decomposition may not even lead to a local optimum. Note also assumption (ii) has a theoretical aspect (existence of a subgradient), and a practical one (we can compute it).

In the nonsmooth optimization terminology, assumptions (ii) and (iii) mean that we have a procedure, called *oracle*, returning a linearization of v (the so-called optimality cuts) or a linearization of V (a feasibility cuts). More precisely, the oracle takes as an input a vector $y \in Y$ and returns a pair in $\mathbb{R} \times \mathbb{R}^q$ as follows.

– If $y \in V$: the oracle returns an approximate value of v at y

$$v_y \in \mathbb{R} \quad \text{such that} \quad v(y) - \eta \leq v_y \leq v(y)\,, \tag{4}$$

and an approximate subgradient of v at y

$$g_y \in \mathbb{R}^q \quad \text{such that} \quad \forall y' \in Y,\ v(y') \geq v_y + g_y^\top (y' - y), \tag{5}$$

where $\eta \geq 0$ is the accuracy of the oracle. When $\eta = 0$, we say that the oracle is exact. Note that by combining the inequalities (4) and (5), we see that g_y is an η-subgradient of v at y:

$$\forall y' \in Y, \qquad v(y') \geq v(y) + g_y^\top (y' - y) - \eta.$$

– If $y \notin V$: the oracle returns a hyperplane separating y from V, that is:

$$(\alpha, s) \in \mathbb{R} \times \mathbb{R}^q \quad \text{such that} \quad \forall y' \in V\,,\ s^\top y' \leq \alpha < s^\top y. \tag{6}$$

2.2 Examples where the convexity assumptions hold

In this section, we present a general situation and two classical examples of problems (linear problems and variable factor problems) that fit in the above framework. The problems of the numerical experiments of Section 5 are instances of linear problems.

The usual situation where the convexity assumptions hold is when the subproblem (2) is solved by duality. The next three lemmas study precisely this case. Their proofs rely on standard results of convex analysis and are provided in the Appendix. Following the notation of (Geoffrion, 1972), we consider the Lagrangian dual function

$$L^*(y,u) := \inf_{x\in X}\{f(x,y)+u^\top G(x,y)\} \tag{7}$$

for a dual multiplier $u \in \mathbb{R}^m_+$, and the associated dual problem

$$\sup_{u\geq 0} L^*(y,u). \tag{8}$$

We also introduce the Lagrangian dual function associated with the constraints of (2)

$$L_*(y,\lambda) := \inf_{x\in X} \lambda^\top G(x,y) \tag{9}$$

that is useful to build a separating hyperplane.

Lemma 1 (Convexity of v) *Assume that, for all $y \in Y \cap V$, there is no duality gap between* (2) *and* (8) *and that, for all $u \geq 0$, the function $L^*(\cdot,u)$ is convex. Then v is closed and convex, as written as:*

$$v(y) = \sup_{u\geq 0} L^*(y,u). \tag{10}$$

Lemma 2 (Subgradients of v) *Suppose that assumptions of Lemma 1 hold. Then, for given $y \in Y \cap V$, $\eta \geq 0$, and an η-optimal solution u of* (8)*, the subgradients of $L^*(\cdot,u)$ at y are η-subgradients of v at y.*

Lemma 3 (Separators of V) *Assume that X is compact and that, for all $\lambda \in \mathbb{R}^m_+$, the function $L_*(\cdot,\lambda)$ is convex. Then, for each $y \notin V$, there exists $\lambda \in \mathbb{R}^m_+$ such that $L_*(y,\lambda) > 0$, and for $s \in \partial_y L_*(y,\lambda)$ and $\alpha := s^\top y - L_*(y,\lambda)$, the hyperplane (s,α) separates y from V (that is, satisfies* (6) *).*

The previous lemmas explain how to build a subgradient or a separating hyperplane, under the assumption of convexity of L^* and L_* with respect to y. It is usually not difficult to check this convexity: we have it when f and G are convex with respect to the variables x and y jointly; the Example 2 below presents another case. The following two examples explicit the construction for classical general problems.

Example 1: Original problem. The seminal paper (Benders, 1962) focuses on problems, linear with respect to x, of the form:

$$\begin{array}{ll} \min & c^\top x + \varphi(y) \\ \text{s.t.} & Ax + \psi(y) \leq b \\ & x \in \mathbb{R}^p_+, y \in Y \subseteq \mathbb{R}^q, \end{array} \tag{11}$$

with $c \in \mathbb{R}^p$, $A \in \mathbb{R}^{m \times p}$ and $b \in \mathbb{R}^m$. We assume that $\varphi \colon \mathbb{R}^q \to \mathbb{R}$ and the m components ψ_i of $\psi \colon \mathbb{R}^q \to \mathbb{R}^m$ are convex. This formulation covers the two problems (network design and hub location) used for our numerical experiments. For a fixed vector $y \in Y$, the subproblem is thus linear:

$$\begin{array}{ll} \min & c^\top x + \varphi(y) \\ \text{s.t.} & Ax \leq b - \psi(y) \\ & x \in \mathbb{R}^p_+, \end{array} \tag{12}$$

and its linear dual is

$$\begin{array}{ll} \max & L^*(y,u) = (\psi(y) - b)^\top u + \varphi(y) \\ \text{s.t.} & A^\top u \geq -c \\ & u \in \mathbb{R}^m_+. \end{array} \tag{13}$$

For a given $y \in Y$, the oracle solves the pair of problems (12)-(13). Three cases are possible.

- If (13) has a finite optimal value, then the oracle returns the value and a subgradient:

$$\begin{aligned} v(y) &= (\psi(y) - b)^\top u(y) + \varphi(y), \\ g(y) &= \sum_i \beta_i u_i(y) + \gamma \quad \in \partial v(y), \end{aligned}$$

  where $u(y)$ is an optimal solution of (13), $\gamma \in \partial \phi(y)$ and $\beta_i \in \partial \psi_i(y)$.
- If (13) has an unbounded optimal value (i.e. (12) is infeasible), then any linear programming solver will return a certificate of unboundedness (or infeasibility). This latter takes the form of an unbounded dual vector: adding this vector to any feasible solution of (13) yields a feasible solution with a larger objective. Formally, the oracle returns $\lambda \in \mathbb{R}^m_+$ such that

$$A^\top \lambda = 0 \text{ and } (G(y) - b)^\top \lambda > 0. \tag{14}$$

  which implies that

$$\forall x \in \mathbb{R}^p_+, \qquad \lambda^\top A x = 0 > \lambda^\top (b - \psi(y)), \tag{15}$$

  and by taking the infimum over $x \in X$ we obtain that $\inf_{x \in \mathbb{R}^p_+} \lambda^\top A x = 0$. Thus, we see that λ is such that $L_*(y, \lambda) > 0$, and Lemma 3 gives the expression of a hyperplane separating y from V.

Example 2: Variable factor-type problems. Let us consider the problem:

$$\begin{array}{ll} \min & \displaystyle\sum_{i=1}^{n} f_i(x^i) y_i \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n} x^i y_i \le c \\ & x \in X \subseteq \mathbb{R}_+^{mn},\ y \in Y \subseteq \mathbb{R}_+^n \end{array} \tag{16}$$

where c is in $(\mathbb{R}_+^*)^m$, f_i are convex functions and X is a convex set. Several problems can be formulated as (16); for instance, the variable factor problem presented in (Geoffrion, 1972) and the unit-commitment of thermal systems problem presented in (Geromel and Belloni, 1986). In general, the problem is nonlinear and nonconvex, because of the constraint $\sum_{i=1}^n x^i y_i \le c$. However, we can prove that the function v is convex, as follows. For any fixed $y \in Y$, the subproblem in x is convex and the solution $x = 0$ is strictly feasible. Therefore Slater's condition holds, so that there is strong duality between the primal subproblem and its dual. Moreover, for given vectors $y \in Y$ and $u \in \mathbb{R}_+^m$, the Lagrangian (7) can be written as

$$L^*(y,u) = \sum_{i=1}^{n} y_i \inf_{x^i \in X_i} \{f_i(x^i) + u^\top x^i\} - u^\top c. \tag{17}$$

We see that the Lagrangian is linear in y for all u. Lemma 1 implies then that function v is convex; and because the subproblem is feasible for all $y \in Y$, v is finite everywhere, i.e. $V = \mathbb{R}^m$. Solutions of the n subproblems in x^i allow us to construct subgradients of v with Lemma 2.

2.3 Benders decomposition algorithm

The Benders algorithm proposed by (Geoffrion, 1972), generalizing the original algorithm of (Benders, 1962), is not readily implementable in general. It relies indeed the assumption (called "Property (P)" in (Geoffrion, 1972)) that for given $u \in \mathbb{R}_+^m$ and $\lambda \in \mathbb{R}_+^m$, one can compute explicitly the expressions of $L^*(y,u)$ and $L_*(y,\lambda)$ of (7) and (9). To cover the general case where this assumption may not hold, it is classical to use the linearizations $L^*(\cdot,u)$ and $L_*(\cdot,\lambda)$. From the nonsmooth perspective, this corresponds to linearizing v and V with the information given by the oracle (by lemmas 2 and 3), and the restrictive assumption (P) is compensated by the convexity assumptions (ii) and (iii).

More precisely, assuming that the oracle has provided information at k points $y_1, \ldots, y_k$, we denote by I_k the set of indices $i \le k$ such that $y_i \in V$ and J_k the set of indices $j \le k$ such that $y_j \notin V$. The cutting-plane models of the convex function v and the convex set V are:

$$\check{v}_k(y) := \max_{i \in I_k} \{v_{y_i} + g_{y_i}^\top (y - y_i)\} \quad \le v(y)\,, \tag{18}$$

$$\check{V}_k := \{y \in \mathbb{R}^q : s_j^\top y \le \alpha_j, j \in J_k\} \quad \supseteq V\,. \tag{19}$$

So we consider here the version of Benders algorithm (that we call cutting-plane Benders method) where the so-called master problem at iteration k is

$$\min \quad \check{v}_k(y) \quad \text{s.t.} \quad y \in \check{V}_k \cap Y, \tag{20}$$

written equivalently

$$\begin{array}{lll} \min & r & \\ \text{s.t.} & v_{y_i} + g_i^\top (y - y_i) \leq r & i \in I_k \\ & s_j^\top y \leq \alpha_j & j \in J_k \\ & y \in Y, r \in \mathbb{R}. & \end{array} \tag{21}$$

The (well-known) nonsmooth interpretation is then clear: the cutting-plane Benders method corresponds to the Kelley cutting-plane method (Kelley, 1960) using the oracle (typically with an error $\eta = 0$). This algorithm is presented schematically in Figure 1 and more precisely in Algorithm 1.

For a later use, let us describe further the notation of Algorithm 1 and its stopping test. The master problem (20) is a relaxation of the original problem (3), i.e. its optimal value $\check{v}_k(y_{k+1})$ provides a lower bound on v^*. We denote by v_k^{low} the best lower bound at iteration k, that is:

$$v_k^{\text{low}} := \max_{i=1,\dots,k} \check{v}_i(y_{i+1}) \quad \leq v^*. \tag{22}$$

Each (approximate) evaluation of v at $y_k \in V$ also gives an (approximate) upper bound. We define v_k^{up} as the best (approximate) upper bound:

$$v_k^{\text{up}} := \min_{i \in I_k} v_{y_i} \quad \geq v^* - \eta. \tag{23}$$

We can then define the optimality gap as

$$\Delta_k = v_k^{\text{up}} - v_k^{\text{low}}. \tag{24}$$

If the oracle is exact ($\eta = 0$), Δ_k is always nonnegative, and $\Delta_k = 0$ implies that the point y_i such that $v(y_i) = v_k^{\text{up}}$ is an optimal solution of (3). In the general case ($\eta \geq 0$), Δ_k is no longer always nonnegative, but still bounded from below: by definition of v_k^{up} and v_k^{low}, we have for all k, $\Delta_k \geq -\eta$. For a fixed tolerance $\varepsilon \geq 0$, the stopping test $\Delta_k \leq \varepsilon$ ensures the approximate convergence. Indeed, denoting y_i the iterate such that $v_{y_i} = v_k^{\text{up}}$, we would have

$$\varepsilon \geq \Delta_k = v_{y_i} - v_k^{\text{low}} \geq v_{y_i} - v^* \geq v(y_i) - \eta - v^*,$$

where the last inequality comes from the definition of the oracle. Hence, $\Delta_k \leq \varepsilon$ gives that y_i is an $(\eta + \varepsilon)$-solution of problem (3).

Algorithm 1 Cutting-plane Benders method

▷ **Initialization**

Choose $y_1 \in V$ and a stopping tolerance $\varepsilon > 0$
Call the oracle at y_1 and get v_{y_1}, $g_1 \in \partial_\eta v(y_1)$
Set $I_1 \leftarrow \{1\}$, $J_1 \leftarrow \emptyset$, $v_1^{\text{up}} \leftarrow v_{y_1}$, $v_1^{\text{low}} \leftarrow -\infty$

for $k = 1, 2, \ldots$ **do**

▷ **Test termination**

  if $\Delta_k \leq \varepsilon$ **then**
    return y^* and $v_{y^*} = v^{\text{up}}$
  end if

▷ **Solve master problem**

  Solve (20) and get $y_{k+1}, \check{v}_k(y_{k+1})$
  Update $v_{k+1}^{\text{low}} \leftarrow \max\{v_k^{\text{low}}, \check{v}_k(y_{k+1})\}$

▷ **Call oracle (solve subproblem)**

  if $y_{k+1} \in V$ **then**
    The oracle returns $(v_{y_{k+1}}, g_{k+1})$
    $I_{k+1} \leftarrow I_k \cup \{k+1\}$ ▷ Optimality cut
    if $v_{y_{k+1}} < v^{\text{up}}$ **then**
      Set $y^* \leftarrow y_{k+1}$, $v^{\text{up}} \leftarrow v_{y_{k+1}}$
    end if
  else $(y_{k+1} \notin V)$
    The oracle returns (α_{k+1}, s_{k+1})
    $J_{k+1} \leftarrow J_k \cup \{k+1\}$ ▷ Feasibility cut
  end if
end for

3 Quadratic stabilization of the algorithm

Cutting planes-based methods are known to suffer from an instability that can lead to a slow convergence, see for example (Bonnans et al, 2006, Example 8.7). The problem is twofold: the method can do very large steps, especially at the first iterations, and, even when the iterates become close to an optimal solution, the method can oscillate around it resulting in excessively slow convergence (the so-called tailing effect). It is the same for the Dantzig-Wolfe column generation which can be seen as dual to the Benders decomposition method.

In nonsmooth optimization, bundle methods have been proposed to stabilize cutting-planes method, see for e.g. the textbook (Hiriart-Urruty and Lemaréchal, 1993). The general idea of bundle methods is to encourage the next iterate to stay close to the best one, while decreasing the cutting-plane model objective. Along with the iterates $\{y_k\}$, the methods thus keep track of a sequence of so-called *stability centers* $\{\hat{y}_k\} \subseteq \{y_k\}$. For a given model $\check{v}_k$ and a stability center $\hat{y}_k$, three popular variants of bundle methods compute a next iterate by solving:

$$\min_{y \in Y} \{\, \check{v}_k(y) + \frac{1}{2t_k} \|y - \hat{y}_k\|^2 \,\} \qquad \text{(proximal bundle method)} \tag{25}$$

$$\min_{y \in Y} \{\, \check{v}_k(y) \;\text{ s.t. }\; \|y - \hat{y}_k\|^2 \leq R_k \,\} \qquad \text{(trust region bundle method)} \tag{26}$$

$$\min_{y \in Y} \{\, \frac{1}{2} \|y - \hat{y}_k\|^2 \;\text{ s.t. }\; \check{v}_k(y) \leq L_k \,\} \qquad \text{(level bundle method)} \tag{27}$$

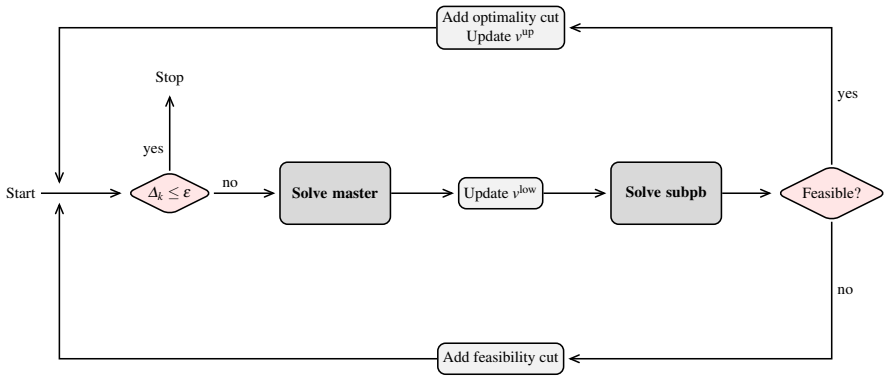

Fig. 1 Scheme of the standard Benders decomposition algorithm

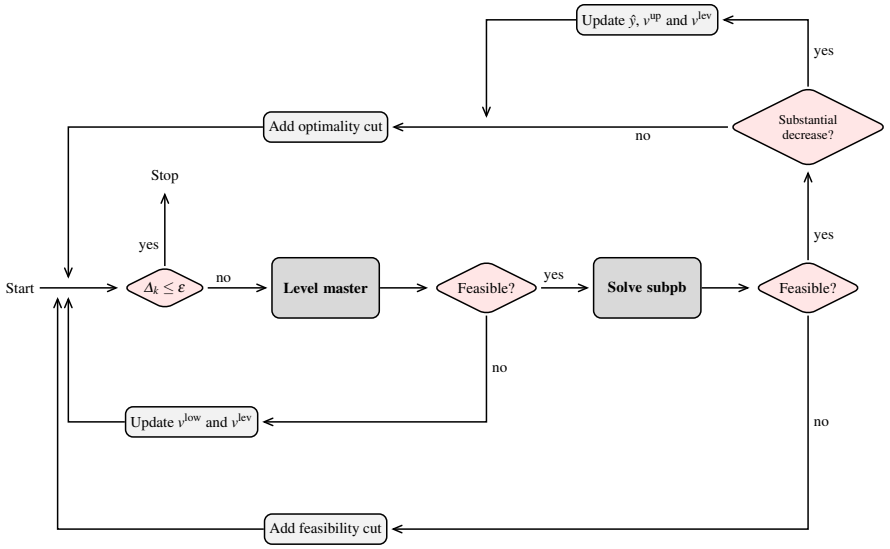

Fig. 2 Scheme of the stabilized Benders decomposition algorithm

with t_k, R_k, L_k real parameters whose role is to balance the minimization of the cutting-plane model and the distance to the stability center. There is a theoretical equivalence between the above three variants: for a fixed cutting-plane model $\check{v}_k$, there exists a choice of the parameters t_k, R_k, L_k such that the three variants generate the same next iterate, see for example (Bonnans et al, 2006, Theorem 9.7). In practice though, the algorithms differ in the way the parameters are managed during the iterations. Even for a given variant, the practical performance relies heavily on the update strategy of the parameters. For proximal bundle method, efficient heuristics have been proposed to manage t_k (Bonnans et al, 2006); it is one of the reasons why this type is usually preferred.

In the context of Benders decomposition for two-stage stochastic problems, quadratic regularizations, inspired from bundle methods, have been proposed by (Ruszczyński,

1986) for exact oracles and (Oliveira et al, 2011) for inexact oracles. In discrete linear optimization, the stabilization of Dantzig-Wolfe decomposition, in particular by a proximal bundle method, proves to accelerate the resolution (Briant et al, 2008) of several mixed-integer linear problems. In these applications however, the underlying nonsmooth problem is convex, i.e. v is convex *and* Y is a convex set. To the best of our knowledge, bundle-type methods have not been adapted to Benders decomposition for mixed-integer programming – or more generally to the minimization of a convex function over a discrete constraint set, as is our master problem in general.

Convexity is a crucial feature in proximal bundle methods: it is needed to define the stopping criteria, to efficiently manage the proximal parameter t_k, as well as to guarantee convergence. Level bundle methods (Lemaréchal et al, 1995) have the advantage of computing a lower bound during the algorithm which, together with an upper bound, allows us to define a simple stopping test. We propose here a stabilization of Benders decomposition based on level bundle methods. We call the resulting algorithm *stabilized Benders method*. Figure 2 provides an overview of the method (comparison of Figures 1 and 2 is instructive) and Algorithm 2 contains all the details. In the reminder of this section we present the main properties of the stabilized Benders method.

Level-master problem. Instead of computing the next iterate as the minimum of the cutting-plane model as for the usual Benders method, we compute the closest point to the current best iterate, within a certain level set of the cutting-plane model. More specifically, we introduce the following *level-master* problem:

$$\begin{array}{ll} \min & \frac{1}{2}\|y-\hat{y}_k\|^2 \\ \text{s.t.} & \check{v}_k(y) \le v_k^{\text{lev}} \\ & y \in \check{V}_k \cap Y\,, \end{array} \tag{28}$$

which is equivalent to the following problem

$$\begin{array}{lll} \min & \frac{1}{2}\|y-\hat{y}_k\|^2 & \\ \text{s.t.} & v_{y_i} + g_i^\top (y-y_i) \le v_k^{\text{lev}} & i \in I_k \\ & s_j^\top y \le \alpha_j & j \in J_k \\ & y \in Y\,. & \end{array}$$

The level-master problem (28) has a quadratic objective function, linear constraints and possibly integer variables. Compared to (21), note that the complexity of the master problem increases, from a mixed-integer linear program (MILP) to a mixed-integer convex quadratic program (MIQP). This is not really a limitation for the approach because solvers for convex MIQPs are efficient (for e.g. Cplex and Gurobi). Moreover, when the oracle is an expensive procedure, the additional time needed to solve the level-master is often negligible in the overall solving process. Thus, if the stabilization allows us to decrease the number of iterations (that is, the number of oracle calls), we reduce the total computing time, as observed in the numerical illustrations of Section 5.

Algorithm 2 Stabilized Benders method

▷ **Initialization**

1: Choose a stopping tolerance $\varepsilon \geq 0$ and parameters κ and $\lambda \in (0,1)$
2: Choose $y_1 \in V$ and call the oracle to get (v_{y_1}, g_1), choose $v_1^{\text{low}} \leq v^*$
3: Set $\hat{y}_1 \leftarrow y_1$, $v_1^{\text{up}} \leftarrow v_{y_1}$, $v_1^{\text{lev}} \leftarrow \lambda v_1^{\text{up}} + (1-\lambda) v_1^{\text{low}}$
4: Set $I_1 \leftarrow \{1\}$, $J_1 \leftarrow \emptyset$

5: **for** $k = 1, 2, \ldots$ **do**

▷ **Test termination**

6: **if** $\Delta_k \leq \varepsilon$ **then**
7: **return** $(\hat{y}_k, v_{\hat{y}_k} = v_k^{\text{up}})$
8: **end if**

▷ **Level-master problem**

9: Solve the (MIQP) level-master (28)
10: **if** Infeasible **then** ▷ Infeasible master iteration
11: Set $v_{k+1}^{\text{low}} \leftarrow v_k^{\text{lev}}$, $v_{k+1}^{\text{lev}} \leftarrow \lambda v_{k+1}^{\text{up}} + (1-\lambda) v_{k+1}^{\text{low}}$ ▷ Increase lower bound
12: Choose I_{k+1} such that $\{\hat{k}\} \subseteq I_{k+1} \subseteq I_k$
13: **continue** (go to line 5)
14: **else**
15: Get y_{k+1} solution of (28)
16: **end if**

▷ **Call oracle**

17: **if** $y_{k+1} \in V$ **then** ▷ Optimality cut
18: The oracle returns $(v_{y_{k+1}}, g_{k+1})$
19: **if** $v_{y_{k+1}} \leq v_k^{\text{up}} - \kappa \delta_k$ **then** ▷ Serious iteration
20: Set $\hat{y}_{k+1} \leftarrow y_{k+1}$, $v_{k+1}^{\text{up}} \leftarrow v_{y_{k+1}}$, $v_{k+1}^{\text{lev}} \leftarrow \lambda v_{k+1}^{\text{up}} + (1-\lambda) v_{k+1}^{\text{low}}$ ▷ Decrease (inexact) upper bound
21: Choose I_{k+1} such that $\{k+1\} \subseteq I_{k+1} \subseteq I_k$
22: **else** ▷ Null iteration
23: Set $I_{k+1} \leftarrow I_k \cup \{k+1\}$
24: **end if**
25: **else** $(y_{k+1} \notin V)$ ▷ Infeasible subproblem
26: The oracle returns (α_{k+1}, s_{k+1})
27: Set $J_{k+1} \leftarrow J_k \cup \{k+1\}$ ▷ Feasibility cut
28: **end if**
29: **end for**

Stability center and upper bound. As in the standard Benders decomposition, the evaluations of v at the solutions of the level-master problem provide approximate upper bounds on v^*. To avoid unnecessary moves, especially since the oracle can be inexact, the classical strategy in bundle methods is to update the stability center only when a "substantial" decrease of the objective is observed (see (Hiriart-Urruty and Lemaréchal, 1993, Chap.XV)). At iteration k, for an upper bound v_k^{up}, the observed decrease is considered substantial when it is at least a fixed fraction $\kappa \in (0,1)$ of the expected one, i.e.

$$v_{y_{k+1}} \leq v_k^{\text{up}} - \kappa \delta_k,$$

where the *expected decrease* δ_k is the decrease of the objective provided by the level

$$\delta_k := v_k^{\text{up}} - v_k^{\text{lev}}. \tag{29}$$

In this case, we update the stability center $\hat{y}_k = y_{k+1}$ and the upper bound $v_k^{\text{up}} := v_{\hat{y}_k}$. Following the standard terminology of bundle methods, an iteration is called *serious* or *null* (respectively), depending on whether there is a substantial decrease or not.

Level parameter and lower bound. In standard Benders decomposition, the master problem (20) is always feasible and its optimal value provides a lower bound. In contrast, the level-master (28) problem may not be feasible and its optimal value is obviously not a lower bound. In fact, it is precisely when the level-master problem is infeasible that we can access a lower bound, as the parameter v_k^{lev} then becomes a valid lower bound for problem (3). Indeed we have in this case:

$$\forall y \in Y \cap \check{V}_k, \qquad \check{v}_k(y) > v_k^{\text{lev}},$$

and, since by definition $\check{V}_k$ contains V and $\check{v}_k$ under-approximates v, this implies

$$\forall y \in Y \cap V, \qquad v(y) \geq v_k^{\text{lev}}.$$

We denote v_k^{low} the best (i.e. the largest) lower bound obtained at an iteration k. At each iteration we have the inequalities:

$$v_k^{\text{low}} \leq v^* \leq v_k^{\text{up}} + \eta. \tag{30}$$

As for the usual Benders method, we use the optimality gap Δ_k of (24) to define the stopping test. By construction the sequence $(v_k^{\text{low}})_k$ is nondecreasing and $(v_k^{\text{up}})_k$ nonincreasing. Thus, the sequence of optimality gaps $(\Delta_k)_k$ is nonincreasing.

Regarding the level parameter, one can observe that the larger v_k^{lev}, the smaller the steps from $\hat{y}_k$ (and conversely). In theory, convergence is ensured for any choice of v_k^{lev}:

$$v_k^{\text{lev}} := \lambda v_k^{\text{up}} + (1-\lambda) v_k^{\text{low}}, \quad \text{with } 0 < \lambda < 1. \tag{31}$$

In our numerical experiments, we use the standard value $\lambda = 0.5$.

Bundle reduction. As in standard Benders decomposition, the level-master problem contains two types of information on v: optimality cuts (indexed by I_k) and feasibility cuts (indexed by J_k). It turns out that, roughly speaking, the essence of optimality information is summarized in the quantities $\hat{y}_k$, v_k^{up}, v_k^{low} and v_k^{lev}. Thus we do not need to keep all the optimality cuts in the bundle to guarantee convergence. More precisely, each time v_k^{lev} is updated (as a consequence of the update of v_k^{up} or v_k^{low}), it is possible to get rid of all optimality cuts, except the one corresponding to the stability center. Let us denote by $\hat{k}$ the index such that $\hat{y}_k = y_{\hat{k}}$. We call *bundle reduction* the possibility of keeping only a subset of the optimality cuts, i.e. choosing any I_{k+1} such that:

$$\{\hat{k}\} \subseteq I_{k+1} \subseteq I_k,$$

at iterations k where v_k^{lev} is updated. In addition to limiting the memory space used by the algorithm, this feature prevents the level-master from growing too large. In particular in the case of multi-cuts approaches (see e.g. (Magnanti and Wong, 1981), (Saharidis et al, 2010) and references therein), bundle reduction would be a natural way to control the size of the level-master problem. In contrast, we emphasize that the bundle of feasibility cuts must be kept entirely, as there is no way to aggregate feasibility cuts information.

Note that the bundle reduction of our algorithm is different from classical bundle compression. Bundle compression is a key feature of bundle methods that aims

at keeping the memory space needed by the algorithm bounded while still ensuring convergence. It is based on the aggregation of the bundle information; basically, the removed cuts are replaced by some convex combination of them. This combination (and the proof that it is indeed enough to guarantee convergence) relies on the convexity of the master problem. Since our level-master problem is not convex in general, we cannot apply a similar bundle compression in our situation.

Bundle reduction however shares the same practical property as bundle compression: although theoretically only a few cuts are needed for convergence, in practice it is better to keep as many as possible for a faster convergence. As always, the user should find a trade-off between the memory space and the convergence speed.

Initialization. Note finally that all stability centers $(\hat{y}_k)_k$ are in V. Thus, the algorithm needs the initial y_1 to be in V. Finding such point can be straightforward (as in the problems used in our numerical experiments). If not, one can start by applying the classical Benders decomposition until a feasible point $y \in V$ if found, and then apply the stabilized version.

Moreover, the algorithm needs a lower bound on the optimal value. A lower bound is often available or easily computed (as in our numerical experiments). If it is not the case, minimizing the first cutting plane model gives one (when Y is bounded as in the analysis of the next section).

Thus the stabilized Benders algorithm needs a feasible solution (upper bound) and a lower bound. While Benders method uses them only for the stopping test, the stabilized Benders method also uses them to define the level parameter v_k^{lev} (see (31)). Thus it can take more advantage from good quality bounds.

4 Convergence analysis of the stabilized Benders method

4.1 Convergence result

In this section, we study the convergence properties of the stabilized Benders method (Algorithm 2). The main result is the following theorem, that we prove at the end of the section after getting some intermediate results. The theorem shows the consistency of our stabilization approach; its performance will be evaluated through numerical experiments in Section 5. We fix an oracle error $\eta \geq 0$ in the reminder of this section.

Theorem 1 (Convergence of the stabilized Benders method) *Assume that the convexity assumptions hold. Assume furthermore that*

- *either Y is finite,*
- *or Y is a compact subset of V, and the sequence of subgradients of v generated during the algorithm is bounded.*

If $\varepsilon = 0$, then the iterates $\hat{y}_k$ of Algorithm 2 generate an η-minimizing sequence, i.e.

$$v^* - \eta \leq \lim v_{\hat{y}_k} \leq v^* \leq \lim v(\hat{y}_k) \leq v^* + \eta\,.$$

If $\varepsilon > 0$, the algorithm terminates in a finite number of iterations with an $(\varepsilon+\eta)$-solution.

This theorem for the stabilized Benders algorithm is similar to those of the Benders algorithm of (Geoffrion, 1972). More precisely, when $\eta = 0$, the theorem corresponds for the stabilized Benders to Theorems 2.4 and 2.5 of (Geoffrion, 1972) for the Benders method.

Remark 1 **(When Y is finite and $\varepsilon = 0$)** In the case where Y is finite and $\varepsilon = 0$, the above convergence result is slightly weaker than the corresponding one of (Geoffrion, 1972) for the Benders decomposition. More precisely, Theorem 2.4 of (Geoffrion, 1972) proves that the Benders algorithm converges in a finite number of iterations (at most $|Y|$ iterations); this is not true anymore for the stabilized Benders algorithm. The stabilization makes us loose the finite convergence when $\varepsilon = 0$, as shown by the following example.

Consider the minimization of the convex quadratic function $v(y) = y^2$ over the singleton $Y = \{1\}$. Suppose that the lower bound 0 is known. Starting Algorithm 2 with $y_1 = 1$, the level parameter will be $v_1^{\text{lev}} = 1/2$. Then, the level-master problem (28) will be infeasible, leading to an update of the lower bound, and the level using expression (31) with $\lambda = 0.5$: $v^{\text{lev}} = 3/4$. Repeating this for the next iterations, we see that the level parameter takes the values $1/2, 3/4, ..., (2^k-1)/2^k$. Therefore, finite convergence of our algorithm is only ensured for a tolerance $\varepsilon > 0$.

Remark 2 **(On the assumptions)** Let us briefly discuss the role of the assumptions of the theorem. The convexity of v (convexity assumption (i)) implies existence of the subgradients on $\operatorname{int} V$, the interior of the domain (see (Hiriart-Urruty and Lemaréchal, 1993, Th XI.1.1.4)). The first part of assumption (ii) then brings the existence of subgradients on its boundary too. Again by (Hiriart-Urruty and Lemaréchal, 1993, Th XI.1.1.4), we have that the subdifferential of v is unbounded on the boundary of V; second part of assumption (ii) then says that we can still compute a finite subgradient on the boundary. Theorem 1 assumes furthermore that, in the case where Y is a compact subset of V, the subgradients of v visited during the algorithm are bounded. This assumption is closely related to the one of Theorem 2.5 of (Geoffrion, 1972), which states that the set of dual multipliers of (8) is bounded, for all $y \in Y$. Lemma 8 in the appendix studies a case where the boundedness of the dual multipliers implies the boundedness of the subgradients of v on all Y.

4.2 Convergence proof

On the schematic representation of the stabilized Benders algorithm of Figure 2, we see that there are several types of iterations, corresponding to different paths in the graph. Using Benders decomposition and bundle methods terminology, we categorize the iterations of Algorithm 2 as follows:

- infeasible master iterations,
- infeasible subproblem iterations,
- serious iterations (feasible master and subproblem + substantial decrease),

– null iterations (feasible master and subproblem but no substantial decrease).

We present a series of lemmas treating each case separately and leading to the proof of Theorem 1. The intuitive idea of the proof is to show that each type of iterations cannot be repeated infinitely many times without leading to convergence.

Lemma 4 (Infeasible master iterations) *If there is an infinite number of infeasible master iterations, then* $\lim \Delta_k \leq 0$.

Proof Let $(\phi(k))_k$ denote the indexes of iterations where the master problem is infeasible, and $(\psi(k))_k$ the indexes of the iterations directly preceding these iterations, i.e. $\psi(k) = \phi(k+1) - 1$. For all k, we have an update of the lower bound:

$$v^{\text{low}}_{\phi(k+1)} = v^{\text{lev}}_{\psi(k)} = \lambda v^{\text{up}}_{\psi(k)} + (1-\lambda) v^{\text{low}}_{\psi(k)} = v^{\text{low}}_{\psi(k)} + \lambda \Delta_{\psi(k)}\,.$$

By definition of ψ, we also have $v^{\text{low}}_{\psi(k)} = v^{\text{low}}_{\phi(k)}$, and therefore:

$$\Delta_{\psi(k)} = \frac{1}{\lambda}(v^{\text{low}}_{\phi(k+1)} - v^{\text{low}}_{\phi(k)})\,.$$

Summing over k, we obtain

$$\sum_{k=0}^{N} \Delta_{\psi(k)} = \frac{1}{\lambda}(v^{\text{low}}_{\phi(N+1)} - v^{\text{low}}_{\phi(0)}) \leq \frac{1}{\lambda}(v^* - v^{\text{low}}_{\phi(0)}) \qquad \text{for all } N. \tag{32}$$

Recall now that the sequence $(\Delta_k)_k$ is nonincreasing and bounded from below by $-\eta$ (by definition of v^{up}_k and v^{low}_k). As a result, the sequence $(\Delta_k)_k$ converges, and so does its subsequence $(\Delta_{\psi(k)})_k$. We conclude that the limit cannot be positive, in view of (32). □

Lemma 5 (Serious iterations) *If there is an infinite number of serious iterations, then we have* $\lim \Delta_k \leq 0$.

Proof Similarly to the proof of Lemma 4, let $(\phi(k))_k$ denote the indexes of the iterations where the descent test is satisfied, and $(\psi(k))_k$ the indexes of the iterations directly preceding these iterations. The condition of line 19 implies that for all k,

$$\delta_{\psi(k)} \leq \frac{1}{\kappa}(v_{\hat{y}_{\phi(k)}} - v_{\hat{y}_{\phi(k+1)}})\,.$$

Summing over k, we get for all N

$$\sum_{k=0}^{N} \delta_{\psi(k)} \leq \frac{1}{\kappa}(v_{\hat{y}_{\phi(0)}} - v_{\hat{y}_{\phi(N+1)}}).$$

From the oracle definition, we have $v_{\hat{y}_{\phi(N+1)}} \geq v(\hat{y}_{\phi(N+1)}) - \eta \geq v^* - \eta$. Therefore, for all N, the partial sum $\sum_{k=0}^{N} \delta_{\psi(k)}$ is bounded from above by the constant $\frac{1}{\kappa}(v_{\hat{y}_{\phi(0)}} - v^* + \eta)$. Observe now from the definitions of δ_k and v^{lev}_k in (29) and (31) that $\delta_k = (1-\lambda)\Delta_k$. This yields that the partial sums $\sum_{k=0}^{N} \delta_{\psi(k)}$ are bounded from above for all N. We can conclude by the same rationale as at the end of the proof of the previous lemma. □

Lemma 6 (Null iterations) *Assume that either Y is finite, or Y is compact and the sequence of η-subgradients of v visited during the algorithm is bounded. Then, there cannot be an infinite sequence of consecutive null iterations.*

Proof Suppose that after k iterations, we have an infinite sequence of consecutive null iterations. This yields that, for all $i \geq k$, $v_i^{\text{lev}} = v_k^{\text{lev}}$, and all the constraints are accumulated (no bundle reduction after iteration k). For all i and j such that $k < i < j$, iterate y_j satisfies the constraint:

$$v_{y_i} + g_i^\top (y_j - y_i) \leq v_j^{\text{lev}} = v_k^{\text{lev}} .$$

Using the Cauchy-Schwarz inequality, we get

$$\|g_i\| \left\|y_i - y_j\right\| \geq (v_{y_i} - v_k^{\text{lev}}) . \tag{33}$$

If Y is finite, we define Λ as the maximal norm of the η-subgradients of v on Y. If Y is compact, the additional assumption also gives that there exists $\Lambda > 0$ such that $\|g_i\| \leq \Lambda$. Then (33) implies that:

$$\left\|y_i - y_j\right\| \geq \frac{1}{\Lambda}(v_{y_i} - v_k^{\text{lev}}) .$$

Besides, since there are no serious iterations after k, it holds that $v_{y_i} > v_k^{\text{up}} - \kappa(v_k^{\text{up}} - v_k^{\text{lev}})$. As a conclusion, we have for all $k < i < j$:

$$\left\|y_i - y_j\right\| > \frac{(1-\kappa)}{\Lambda}(v_k^{\text{up}} - v_k^{\text{lev}}) > 0 .$$

This contradicts the existence of a convergent subsequence of the bounded sequence $(y_k)_k \subseteq Y$. □

Lemma 7 (Infeasible subproblem iterations) *Assume that Y is finite. There cannot be an infinite number of iterations where the subproblem is infeasible.*

Proof Suppose $y_k \notin V$ is a solution of the master problem at an iteration $k-1$. Then, the oracle returns a hyper-plane separating y_k from V, ie (α_k, s_k) such that $s_k^\top y_k > \alpha_k$. The constraint $s_k^\top y \leq \alpha_k$ is then added to the master problem and prevent y_k from being a solution in any future iteration. □

We are now in position to prove Theorem 1.

Proof (of Theorem 1) Suppose that Algorithm 2 performs an infinite number of iterations. Let us distinguish the two cases:

- If Y is a compact subset of V, then there are no infeasible subproblem iterations by definition, and no infinite sequence of null iterations by Lemma 6.
- If Y is finite, there is a finite number of infeasible subproblem iterations by Lemma 7, and again no infinite sequence of null iterations by Lemma 6.

Thus, in both cases we have an infinite number of either infeasible master iterations or serious iterations. These two situations are handled by Lemmas 5 and 4 respectively. Therefore in any case, we have that $\lim \Delta_k \leq 0$; let us show now that it is sufficient to conclude.

Recall the definition (24) of Δ_k. By construction, the sequence of upper bounds $\{v_k^{\text{up}}\} = \{v_{\hat{y}_k}\}$ is nonincreasing and bounded from below (see (30)) thus converges and $\lim v_{\hat{y}_k} \geq v^* - \eta$. Similarly the sequence of lower bounds $\{v_k^{\text{low}}\}$ is nondecreasing and bounded from above by v^*. Writing $\lim \Delta_k \leq 0$ as $\lim v_{\hat{y}_k} - \lim v_k^{\text{low}} \leq 0$, we obtain

$$v^* - \eta \leq \lim v_{\hat{y}_k} \leq v^* \tag{34}$$

Moreover the η-oracle properties imply that, for all k, $v^* \leq v(\hat{y}_{k+1}) \leq v_{\hat{y}_k} + \eta$ and passing to the limit-inf, we get

$$v^* \leq \liminf v(\hat{y}_{k+1}) \leq \lim v_{\hat{y}_k} + \eta. \tag{35}$$

Combining (34) and (35) we obtain the announced η-convergence. □

5 Numerical illustrations

In this section, we illustrate the efficiency of the stabilized Benders algorithm by comparing its performances to the standard Benders algorithm. We use two classical classes of mixed-integer linear problems for which Benders decomposition-based approaches are the state-of-the-art: network design problems and hub location problems. Regarding the problems themselves, we do not present any novel numerical result; here we just illustrate the interest of our quadratic stabilization. Each of the following subsections presents the problems and show a comparison between the stabilized and standard Benders algorithms.

For both problem classes, we have the following common experimental framework. We have implemented the standard and stabilized Benders algorithms in Python, using Gurobi callable libraries for solving the subproblems. All experiments have been done on a processor Intel(R) Xeon(R) CPU W3530, running at 2.8GHz, with 10GB of RAM in a Linux environment. The subproblems (2) are solved exactly (the oracle error is $\eta = 0$). We use the parameters $\kappa = 0.1$ and $\lambda = 0.5$, as well as a relative tolerance of 10^{-3} for the stopping tests. A lower bound on the optimal solution is computed by omitting the integrality constraints. For each problem, an initial feasible point ($y_1 \in V$) can be easily computed.

5.1 Experiments on network design problems

Network design problems consist in selecting arcs from a graph in order to satisfy some flow constraints, at minimal cost. These constraints concern the transportation of commodities from origin to destination nodes. The structure of the problem makes Benders decomposition a method of choice for tackling it; see the review (Costa, 2005).

We consider the fixed-charged uncapacitated network problem, where there are fixed costs associated with opening arcs, and no capacity limit on the amount of flow going through the network. We denote by N the set of nodes, A the set of arcs and K the set of commodities. We define the binary variables y_{ij} to express whether the link between nodes i and j is used or not in the solution, and the continuous variables x_{ijk} to represent the amount of flow of commodity $k \in K$ through arc (i, j). We can then formulate the problem as

$$\begin{aligned}
\min_{x,y} \ & \sum_{ijk} c_{ijk} x_{ijk} + \sum_{ij} f_{ij} y_{ij} \\
\text{s.t.} \ & \sum_{j} x_{ijk} - \sum_{j} x_{jik} = \begin{cases} d_k & i = O(k) \\ -d_k & i = D(k) \\ 0 & \text{otherwise} \end{cases} \quad \forall i,k \\
& x_{ijk} \leq d_k y_{ij} \quad \forall i,j,k \\
& x_{ijk} \geq 0, \quad \forall i,j,k \\
& y_{ij} \in \{0,1\} \quad \forall i,j,
\end{aligned}$$

where c_{ijk} denotes the unitary cost of routing commodity k through arc (i, j), f_{ij} the fixed cost of utilizing arc (i, j), d_k the demand in commodity k, $O(k)$ and $D(k)$ the origin and destination nodes of commodity k respectively.

Fixing the variable y means fixing the network; the remaining subproblem is the well-known easy (polynomial) problem of finding a flow of minimal cost. Moreover the subproblem is separable with respect to the commodities so that we can write the function v as:

$$v(y) = \sum_{k \in K} v_k(y) + \sum_{ij} f_{ij} y_{ij},$$

where $v_k(y)$ are the optimal values of the $|K|$ independent subproblems:

$$\begin{aligned}
v_k(y) := \min_{x} \ & \sum_{ij} c_{ijk} x_{ijk} \\
\text{s.t.} \ & \sum_{j} x_{ijk} - \sum_{j} x_{jik} = \begin{cases} d_k & i = O(k) \\ -d_k & i = D(k) \\ 0 & \text{otherwise} \end{cases} \quad \forall i \\
& x_{ijk} \leq d_k \bar{y}_{ij} \quad \forall i,j \\
& x_{ijk} \geq 0 \quad \forall i,j.
\end{aligned} \tag{36}$$

If the subproblem is infeasible for a fixed y, then similarly to Example 1 of Section 2.2, a certificate of infeasibility allows to build a feasibility cut.

We generate random instances of fixed charge uncapacitated network design problem, with a number of nodes $|N| \in \{5, 8, 10, 12, 15, 20\}$ and a number of commodities $|K| \in \{5, 10, 15, 20\}$. This leads to problems with up to 2400 variables (400 boolean and 2000 continuous) and reasonable computing times. For each problem size, we generate 3 different instances. We initialize the algorithms with the solution where all the arcs are opened, which is feasible for all instances.

Table 1 summarizes the results of the standard versus the stabilized Benders algorithms. Both algorithms converge to an optimal solution; for each, we report the CPU time in seconds, the total number of iterations and the number of iterations where the

subproblem (36) is infeasible ("f. cuts" column), which corresponds to the number of feasibility cuts added to the master problems (21) and (28). For each instance size, we present the average results over the 3 instances that we generated. We observe that the

Instances		Standard Benders algorithm			Stabilized Benders algorithm		
nodes	commodities	time (s)	iterations	f. cuts	time (s)	iterations	f. cuts
5	5	0.27	24	16	0.31	22	5.66
5	10	0.38	25.33	15	0.07	8	0.66
5	15	0.58	29.33	16	0.12	9	0.33
5	20	0.69	29.66	15.66	0.08	5	0
8	5	1.24	33.66	28	0.65	18.66	6
8	10	42.13	167.33	108.66	53.43	82	27
8	15	72.49	175.66	99.66	60.60	56.33	4.33
10	5	7.09	61.66	41	3.95	38.33	13
10	10	555.79	272	196.66	252.69	65.33	13.66
10	15	20099.7	671.66	426	20289.8	205.66	23.33
12	5	37.58	108.33	90.33	12.8	49.66	29
12	10	34267.4	893	698	10661.6	283.66	161.33
15	5	677.50	199	172.66	53.54	60	34.66
20	5	10796.2	324.33	252.33	1481.89	113.33	68.33
Average reduction					56 %	60 %	83 %

Table 1 Standard versus stabilized Benders algorithms on network design problems

stabilized version performs significantly better than the standard one. It is the fastest for 37 out of 42 instances with an average acceleration of 56 %. As expected, this speed-up is due to the reduction of the number of iterations needed for convergence (on average 60% less iterations).

More surprisingly, we observe a dramatic decrease in the number of feasibility cuts (at least 50 % and 83 % on average). Intuitively, by forcing the iterates to stay close to the best one, the stabilization prevents the iterates from staying for too long in infeasible regions. This is an interesting feature of our algorithm because numerous feasibility cuts are often an issue in Benders method (see for e.g. (Saharidis and Ierapetritou, 2010)).

Finally note that the stabilized Benders still requires a lot of iterations for some instances (especially the ones of size (10,15), (12,10)). A way to improve the performance could be a sophisticated management of the level parameter λ. For sake of simplicity, we keep this parameter fixed on our experiments; it is enough to outperform the standard Benders method. In practice anyway, an efficient solution of network design problems would require combining the stabilization with existing general acceleration techniques (mentioned in the introduction) and other strategies specific for this problem (see (Costa, 2005)).

5.2 Experiments on hub location problems

The hub location problem aims at locating hubs to rout commodities at minimal cost. Given a set of commodities to transport from origin to destination nodes, the problem consists in locating hubs and choosing routes for each commodity through one or

two hubs in order to minimize setup and transportation cost. Benders decomposition based approaches have been successfully applied to this problem, allowing to tackle large scale realistic instances (see (Contreras et al, 2011) and references therein).

We consider the uncapacitated hub location problem with multiple assignments. In this variant, the number of hubs is not fixed, there are no capacity limits on the amount of flow routed through the arcs and the hubs and each commodity can be routed through several paths. We use the path formulation proposed first by (Hamacher et al, 2004). For a given complete graph (N,A), we denote by $H \subseteq N$ the set of potential hubs. For each commodity $k \in K$, we denote by $o(k) \in N$ (resp. $d(k) \in N$) its origin (resp. destination) node and W_k the amount of commodity k to be routed. We denote by f_i the fixed cost associated with location a hub at node $i \in H$, d_{ij} the euclidean distance between nodes i and $j \in N$. The transportation cost of routing commodity k through two hubs i then $j \in H$, i.e. through the path $(o(k),i,j,d(k))$ is written as

$$F_{ijk} = W_k(\chi d_{o(k)i} + \tau d_{ij} + \delta d_{jd(k)})$$

where χ, τ, δ are the collection, transfer and distribution cost. The problem can be formulated as:

$$\begin{aligned}
\min_{x,y} \ & \sum_{i\in H} f_i y_i + \sum_{i\in H}\sum_{j\in H}\sum_{k\in K} F_{ijk}x_{ijk} & \\
& \sum_{i\in H}\sum_{j\in H} x_{ijk} = 1 & \forall k \in K \\
& \sum_{j\in H} x_{ijk} + \sum_{j\in H\setminus\{i\}} x_{jik} \leq y_i & \forall i \in H, \forall k \in K \\
& x_{ijk} \geq 0 & \forall i,j \in H, \forall k \in K \\
& y_i \in \{0,1\} & \forall i \in H.
\end{aligned}$$

When fixing the variable y (i.e. the locations of the hubs) the remaining subproblem is to assign each commodity to one or two hubs. A simple way to avoid infeasible subproblems is to add the constraint that at least one hub should be opened, namely

$$\sum_{i\in H} y_i \geq 1. \tag{37}$$

Again, the subproblem is separable with respect to the commodities leading to $|K|$ smaller independent subproblems. Note that we do not implement the sophisticated techniques of (Contreras et al, 2011) to exploit further the structure of these subproblems.

We use the Australian Post set of instances (available at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/phubinfo.html`), which is a classical data set in the hub location literature. It provides the postal flow between 200 cities (given by their coordinates), transportation costs (χ, τ and δ) and setup costs f_i. The data set also provides a procedure to generate smaller instances by grouping the cities. As postal flow is required between every pair of cities, the number of commodities is $|K| = |N|^2$, and all nodes are potential hubs so $|H| = |N|$. We consider instances with a number of nodes in $\{10,15,20,25,30,35,40\}$. We do not consider larger instances because the cost of solving subproblems (especially in terms of memory space). For each problem size, we generate 3 instances by taking different values for the transfer

cost $\tau \in \{0.1, 0.5, 1\}$. The different transfer costs leads to different numbers of open hubs in the optimal solutions, and thus various instances types.

Table 2 presents the results of applying the standard and stabilized Benders algorithms. Again, for all the instances, both algorithms converge to an optimal solution. Table 2 reports the computing time in seconds and the number of iterations. There is one less column than in Table 1: recall indeed that there is no feasibility issue here because of the additional constraint (37). We first see on Table 2 that the standard method already performs well, with at most 13 iterations to reach convergence. The stabilized version performs slightly better: it is faster for 16 out of 21 instances and give an average time reduction of 14 %. Thus, for this problem where standard Benders method works already fine, the stabilization is still able to improve the results.

Instance		Standard Benders algorithm		Stabilized Benders algorithm	
nodes	τ	time (s)	iterations	time (s)	iterations
10	0.1	1.06	7	0.91	6
	0.5	1.28	8	0.89	6
	1	1.07	7	0.75	5
15	0.1	5.31	7	3.01	4
	0.5	5.27	7	5.86	8
	1	4.52	6	5.83	8
20	0.1	21.72	9	16.61	7
	0.5	16.83	7	14.24	6
	1	14.3	6	13.94	6
25	0.1	58.66	10	35.18	6
	0.5	52.58	9	34.91	6
	1	46.31	8	28.7	5
30	0.1	112.08	9	144.47	12
	0.5	97.72	8	96.28	8
	1	97.11	8	96.11	8
35	0.1	296.61	13	182.69	8
	0.5	183.46	8	116.71	5
	1	177.94	8	110.59	5
40	0.1	467.17	12	498.91	13
	0.5	351.77	9	310.24	8
	1	306.04	8	336.76	9
Average reduction				14 %	13 %

Table 2 Comparison of standard et stabilized Benders algorithms on hub location problems

6 Conclusion and perspectives

In this methodological paper, we use a nonsmooth optimization perspective on Benders decomposition to introduce an algorithmic improvement of the method, inspired by level bundle methods and complementary to existing accelerating techniques. The idea is to add a quadratic stabilization in the master problem of the cutting-plane Benders algorithm, to reduce the number of iterations of the method. Convergence of the stabilized Benders algorithm is established under the usual assumptions. The interest of the approach is illustrated on two mixed-integer linear problems: for the network

design problem, the stabilized algorithm is able to drastically reduce the number of used cuts, especially the feasibility cuts; for the hub location problem, the stabilized algorithm is still able to improve the excellent performance of the standard method.

Quadratic stabilizations are known to be efficient in the context of constraint decomposition or scenario decomposition (see e.g. (Ruszczyński, 1986), (Lemaréchal, 2001), (Briant et al, 2008) and (Oliveira et al, 2011)); the contribution of this paper is to show that it is also the case for Benders decomposition of mixed-integer problems. It is of interest for future research to study similar stabilization techniques for other cutting plane-based algorithms for mixed-integer programming, as the extended cutting-plane method (Westerlund and Pettersson, 1995) and outer-approximation methods (see e.g. (Fletcher and Leyffer, 1994) and (Bonami et al, 2008)). The recent preprint (Oliveira, 2014) builds on this line of research.

Appendix: more on the nonsmooth optimization perspective

This appendix completes Section 2 about the nonsmooth optimization viewpoint on Benders decomposition by providing the proofs of the results together with an additional result.

Proof (of Lemma 1) The no duality gap assumption gives the expression (10). It follows that v is closed and convex, as the supremum of a family (indexed by $u \in \mathbb{R}^m_+$) of convex functions ($y \mapsto L^*(y,u)$). □

Proof (of Lemma 2) Suppose u is an η-optimal solution of (8) for a given $y \in Y$, then $\partial_y L^*(y,u)$ is nonempty (see for e.g. (Hiriart-Urruty and Lemaréchal, 1993, Chap VI)). Let g be in $\partial_y L^*(y,u)$, the expression (10) of v and the convexity of $y \mapsto L^*(y,u)$ imply that

$$\forall y' \in Y \cap V, \quad v(y') \geq L^*(y',u) \geq L^*(y,u) + g^\top (y'-y).$$

Besides, the definition of u gives that $L^*(y,u) \geq v(y) - \eta$, thus

$$\forall y' \in Y \cap V, \quad v(y') \geq v(y) + g^\top (y'-y) - \eta,$$

which ends the proof. □

Proof (of Lemma 3) The compactness of X implies that the two following systems of inequalities

$$x \in X \text{ s.t. } G(x,y) \leq 0 \quad \text{and} \quad \lambda \in \mathbb{R}^m_+ \text{ s.t. } L_*(y,\lambda) \leq 0,$$

satisfy the assumptions of the theorem of strong alternatives (see (Boyd and Vandenberghe, 2004, p 261)): for $y \notin V$ there exists a vector $\lambda \in \mathbb{R}^m_+$ such that $L_*(y,\lambda) > 0$. On the other hand, observe that by definition of V, for all $y' \in V$ and $\lambda \in \mathbb{R}^m_+$, we have $L_*(y,\lambda) \leq 0$. Thus we can write:

$$\forall y' \in V, \quad L_*(y',\lambda) \leq 0 < L_*(y,\lambda). \tag{38}$$

Besides for s in $\partial_y L_*(y,\lambda)$ (which is nonempty by definition (Hiriart-Urruty and Lemaréchal, 1993, Chap VI)), the convexity of L_* with respect to y implies that

$$\forall y' \in V, \quad L_*(y,\lambda) + s^\top (y'-y) \leq L_*(y',\lambda)\,, \tag{39}$$

Combining (38) and (39) we deduce that:

$$\forall y' \in V, \quad s^\top y' \leq s^\top y - L_*(y,\lambda) < s^\top y.$$

Then, taking the supremum over $y' \in V$ and introducing $\alpha := s^\top y - L_*(y,\lambda)$, we obtain

$$\sup_{y' \in V} s^\top y' \leq \alpha < s^\top y,$$

which means that (s,α) is a hyperplane separating y from V. □

Finally, we go beyond Lemma 2 by giving explicitly the expression of the subgradients of v.

Lemma 8 (Subdifferential of v when the problem is convex) *Assume that the set X is convex, and that the functions f and the G are convex. Then v is convex. For given $y \in Y \cap V$, assume moreover that there exists x an optimal solution of* (2) *and that Slater assumption holds, i.e.*

$$\exists x \in \mathit{int}\, X \quad G_i(x,y) < 0 \quad \textit{for all } i = 1,\dots,m.$$

Then the subdifferential of v at y consists in all the vectors of

$$\partial_y f(x,y) + \sum_{i=1}^m u_i \partial_y G_i(x,y), \tag{40}$$

where $u \in \mathbb{R}^m_+$ are the optimal solutions of (8)*, that is, such that $u^\top G(x,y) = 0$ and*

$$0 \in \partial_x f(x,y) + \sum_{i=1}^m u_i \partial_x G_i(x,y) + N_X(x). \tag{41}$$

Proof We provide a different proof recasting v explicitly as a marginal function obtained by partial minimization. Consider δ_X the indicator function of X, and $\delta_{]-\infty,0]}$ the indicator function of the interval $]-\infty,0]$. Thus we write

$$v(y) = \inf_{x \in \mathbb{R}^p} h(x,y) \quad \text{with} \quad h(x,y) := f(x,y) + \sum_{i=1}^m \delta_{]-\infty,0]} \circ G_i(x,y) + \delta_X(x). \tag{42}$$

Applying the calculus rule for the subdifferential of a marginal function (Hiriart-Urruty and Lemaréchal, 1993, X.3.3.2), we get that $\partial v(y)$ for given y is exactly the set $s \in \mathbb{R}^q$ such that

$$\begin{pmatrix} 0 \\ s \end{pmatrix} \in \begin{pmatrix} \partial_x h(x,y) \\ \partial_y h(x,y) \end{pmatrix}. \tag{43}$$

where x is an optimal solution (42) (that is (2)), which exists by assumption. We now explicit the two above partial subdifferentials. The Slater assumption is a sufficient

condition to be able to write the subdifferential of the sum of the subdifferentials (see (Hiriart-Urruty and Lemaréchal, 1993, XI.3.1.2))

$$\partial_x h(x,y) = \partial_x f(x,y) + \sum_{i=1}^{m} \partial_x(\delta_{]-\infty,0]} \circ G_i)(x,y) + \partial \delta_X(x).$$

Recall that the subdifferential of an indicator function is the normal cone of the underlying set:

$$\partial \delta_X(x) = N_X(x) \qquad \text{and} \qquad \partial \delta_{]-\infty,0]}(\alpha) = N_{]-\infty,0]}(\alpha) = \begin{cases} \emptyset & \text{if } \alpha > 0 \\ 0 & \text{if } \alpha < 0 \\ \mathbb{R}_+ & \text{if } \alpha = 0 \end{cases}$$

The subdifferential calculus rule for the post-composition (Hiriart-Urruty and Lemaréchal, 1993, Corollary VI.4.3.1) also gives that for the convex functions G_i's

$$\partial_x(\delta_{]-\infty,0]} \circ G_i)(x,y) = \begin{cases} \emptyset & \text{if } G_i(x,y) > 0 \\ 0 & \text{if } G_i(x,y) < 0 \\ \mathbb{R}_+ \partial_x G_i(x,y) & \text{if } G_i(x,y) = 0 \end{cases}$$

Thus the two parts of (43) gives respectively (40) and (41). □

This lemma shows that there is a direct link between the subgradients of v and the dual multipliers of (8). This result allows us to establish precisely the correspondance between the assumptions of Theorem 1 and the assumptions of its counterpart for Benders decomposition (Theorem 2.5 of (Geoffrion, 1972)). Since the subdifferentials of the finite-valued functions $y \mapsto f(x,y)$ and $y \mapsto G(x,y)$ are compact on Y, the expression (40) indeed shows that the boundedness of the dual multipliers leads to the boundedness of the subgradients of v on all Y.

References

Benders J (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4(1):238–252

Bonami P, Biegler L, Conn A, CornuéJols G, Grossmann I, Laird C, Lee J, Lodi A, Margot F, Sawaya N, WäChter A (2008) An algorithmic framework for convex mixed integer nonlinear programs. Discret Optim 5(2):186–204

Bonnans J, Gilbert J, Lemaréchal C, Sagastizábal C (2006) Numerical Optimization. Theoretical and Practical Aspects. Universitext, Springer-Verlag, Berlin, second edition

Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press, New York, NY, USA

Briant O, Lemaréchal C, Meurdesoif P, Michel S, Perrot N, Vanderbeck F (2008) Comparison of bundle and classical column generation. Mathematical Programming 113(2):299–344

Conejo A, Castillo E, Mìnguez R, Garcìa-Bertrand R (2006) Decomposition techniques in mathematical programming. Engineering and science applications, Springer

Contreras I, Cordeau JF, Laporte G (2011) Benders decomposition for large-scale uncapacitated hub location. Operations Research 59(6):1477–1490

Cordeau JF, Soumis F, Desrosiers J (2000) A Benders decomposition approach for the locomotive and car assignment problem. Transportation Science 34(2):133–149

Costa AM (2005) A survey on benders decomposition applied to fixed-charge network design problems. Computers & Operations Research 32(6):1429 – 1450

Côté G, Laughton MA (1984) Large-scale mixed integer programming: Benders-type heuristics. European Journal of Operational Research 16(3):327 – 333

Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming 66(1-3):327–349

Fortz B, Poss M (2009) An improved Benders decomposition applied to a multi-layer network design problem. Operations Research Letters 37(5):359 – 364

Geoffrion A (1972) Generalized Benders decomposition. Journal of Optimization Theory and Applications 10(4):237–260

Geoffrion AM, Graves GW (1974) Multicommodity distribution system design by Benders decomposition. Management Science 20(5):822–844

Geromel J, Belloni MR (1986) Nonlinear programs with complicating variables: Theoretical analysis and numerical experience. Systems, Man and Cybernetics, IEEE Transactions on 16(2):231–239

Hamacher HW, Labbé M, Nickel S, Sonneborn T (2004) Adapting polyhedral properties from facility to hub location problems. Discrete Applied Mathematics 145(1):104 – 116

Hiriart-Urruty JB, Lemaréchal C (1993) Convex Analysis and Minimization Algorithms. No. 305-306 in Grund. der math. Wiss, Springer-Verlag, (two volumes)

Kelley JE (1960) The cutting plane method for solving convex programs. J Soc Indust Appl Math 8:703–712

Lemaréchal C (2001) Lagrangian relaxation. In: Jünger M, Naddef D (eds) Computational Combinatorial Optimization, Springer Verlag, Heidelberg, pp 112–156

Lemaréchal C, Nemirovskii A, Nesterov Y (1995) New variants of bundle methods. Math Program 69(1):111–147

Magnanti TL, Wong RT (1981) Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. Operations Research 29(3):464–484

McDaniel D, Devine M (1977) A modified Benders partitioning algorithm for mixed integer programming. Management Science 24(3):312–319

Mercier A, Cordeau JF, Soumis F (2005) A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. Computers & Operations Research 32(6):1451 – 1476

Naoum-Sawaya J, Elhedhli S (2013) An interior-point Benders based branch-and-cut algorithm for mixed integer programs. Annals of Operations Research 210(1):33–55

Oliveira W (2014) Regularized nonsmooth optimization methods for convex MINLP problems. Tech. Rep. Submitted for publication, preprint available on `http://www.oliveira.mat.br/publications`

Oliveira W, Sagastizábal C, Scheimberg S (2011) Inexact bundle methods for two-stage stochastic programming. SIAM Journal on Optimization 21(2):517–544

Rei W, Cordeau JF, Gendreau M, Soriano P (2009) Accelerating Benders decomposition by local branching. INFORMS Journal on Computing 21(2):333–345

Ruszczyński A (1986) A regularized decomposition method for minimizing a sum of polyhedral functions. Mathematical Programming 35(3):309–333

Saharidis GK, Ierapetritou MG (2010) Improving Benders decomposition using maximum feasible subsystem (mfs) cut generation strategy. Computers & Chemical Engineering 34(8):1237 – 1245

Saharidis GKD, Minoux M, Ierapetritou MG (2010) Accelerating Benders method using covering cut bundle generation. International Transactions in Operational Research 17(2):221–237

Sahinidis N, Grossmann I (1991) Convergence properties of generalized benders decomposition. Computers & Chemical Engineering 15(7):481 – 491

Westerlund T, Pettersson F (1995) An extended cutting plane method for solving convex MINLP problems. Computers and Chemical Engineering 19, Supplement 1(0):131 – 136

Zakeri G, Philpott A, Ryan D (2000) Inexact cuts in Benders decomposition. SIAM Journal on Optimization 10(3):643–657