Generalized Partial Benders Decomposition of Two Stage Stochastic Programs

Pantuso, Giovanni

Publication date:
2019

Citation for published version (APA):
Generalized Partial Benders Decomposition of Two-Stage Stochastic Programs

Giovanni Pantuso
Department of Mathematical Sciences
University of Copenhagen
Universitetsparken 5, 2100, Copenhagen, Denmark
gp@math.ku.dk

Abstract

This paper introduces the concept of Generalized Partial Benders Decomposition (GPBD), a solution method for two-stage stochastic programs, possibly mixed-integer at both stages, which revisits the classical Benders decomposition algorithm. Richer master problems are obtained by retaining selected second-stage variables and constraints. The scope is that of providing the master problem additional structure to produce better solutions in shorter times, and thus to reduce the reliance on the cuts generated. A number of simple and general-purpose strategies for retaining second-stage variables and constraints in the master problem are proposed. GPBD is compared to classical Benders decomposition in an extensive computational study based on instances of the Capacitated Facility Location Problem with Stochastic Demand. The study illustrates that GPBD improves Benders decomposition on the great majority of the instances and that the improvements are stable across a set of heterogeneous instances. In addition, it illustrates that even simple retaining strategies may grant significant improvements and that the best improvements are obtained retaining a relatively small number of variables and constraints. Finally, results show that efficient retaining strategies can be discovered by training the method on small-scale instances.

Keywords: Stochastic Programming, Benders Decomposition, Facility Location.

1 Introduction

Two-stage Stochastic Programs (SPs) have, for long time, interested the operations research community. The formal theoretical foundation was laid by G.B. Dantzig in 1955 [Dantzig, 1955]. Essentially, some decisions are made here-and-now (so called first-stage decisions), then the value of some random parameters of the problem is realized and, finally, other decisions are made (so called, recourse or second-stage decisions). Their intuitive structure, and the potential benefits for decision makers dealing with uncertainty, attracted significant attention in areas such as vehicle routing [Gendreau et al., 2014, 2016], power production [Shiina and Birge, 2004, Papavasiliou and Oren, 2013], fleet planning [Pantuso et al., 2016, Mørch et al., 2017], and network design [Crainic et al., 2011, Bai et al., 2014]. Today SPs represent a well-established framework and research avenue.
Benders decomposition [Benders, 1962, BD], adapted to SPs by Van Slyke and Wets [1969], has been among the first and most used exact methods for solving SPs. In a nutshell, the second-stage components of the problem are dualized and projected onto the subspace generated by the first-stage variables only, generating the so called master problem [Geoffrion, 1970b]. The dualized components, initially relaxed, are iteratively reconstructed through the addition of optimality and feasibility cuts obtained by solving second-stage problems as subproblems.

Several improvements have been proposed to the original algorithm, see Rahmaniani et al. [2016]. Particularly, the authors report that less than 5% of the contributions focused on the decomposition strategy, that is on the partitioning of the problem in order to obtain a master and subproblems. Nevertheless, the formulation of the problem (and thus of the master and subproblems) has a crucial impact on the performances of BD [see, e.g., Geoffrion and Graves, 1974, Magnanti and Wong, 1981]. As Crainic et al. [2014] point out, the initial relaxation typically generates a rather weak formulation of the master problem, without any information about recourse decisions, and with a number of consequent computational challenges such as erratic bounds and slow convergence. In this paper we revisit the classical decomposition strategy applied in BD and thus propose improvements along the above-mentioned decomposition strategy.

As far as stochastic programs are concerned, alternative decomposition strategies have recently been proposed by Crainic et al. [2014] and Crainic et al. [2016]. The former introduce the idea of a partial Benders decomposition (PBD) obtained by retaining selected second-stage subproblems in the master problem, thus relaxing the original problem only partially, compared to BD. The intuition behind the method is to provide the master problem second-stage information from the beginning as to enable it to make better decisions in shorter times. Fewer cuts and iterations are consequently expected. The authors propose several strategies for selecting subproblems to retain in the master problem. Tests on instances of the Stochastic Fixed Charge Multicommodity Network Design Problem illustrated that the new decomposition approach improves plain BD. Crainic et al. [2016] extend the idea of Crainic et al. [2014] including scenario creation strategies. That is, artificial scenarios are generated with the scope of improving the bounds provided by the master, should the corresponding subproblems be retained. Also in this case, tests show a significant improvement compared to the original BD.

Motivated by the need of studies of alternative decomposition strategies [see Rahmaniani et al., 2016] and by the positive results of Crainic et al. [2014] and Crainic et al. [2016], we also pursue a partial decomposition of SPs. Particularly, we propose the idea of retaining in the master problem selected second-stage variables and constraints, thus projecting the original problem onto the subspace made of the first-stage variables and selected second-stage variables. We refer to this decomposition strategy as Generalized Partial Benders Decomposition (GPBD). With respect to BD, GPBD relaxes the second-stage components only partially, thus leaving the master problem additional structure to generate higher quality solutions already in the initial iterations of the algorithm. With respect to PBD, GPBD retains in the master problem only selected variables and constraints of the (selected) subproblems, as opposed to retaining entire subproblems. The aim is that of providing the master problem useful information without
excessively increasing its size by adding entire subproblems, and thus combining the benefits of an improved but light master problem formulation. GPBD can be seen as a generalization of PBD which allows the user to tailor the master problem based on the specific SP at hand without necessarily retaining entire subproblems. Thus, PBD represents an instance of GPBD where all the variables of selected subproblems are retained. The expected advantages of an augmented MP are essentially two: tighter bounds and early production of better first-stage candidate solutions, in turn leading to fewer iterations and required cuts. However, a trade-off between the amount of information kept in the master problem and its complexity is to be found. Therefore, an important exercise in GPBD is that of identifying the most relevant second-stage information (i.e., variables and constraints). While knowledge of the specific SP is a key element, in this paper we propose a number of simple general-purpose retaining strategies. Furthermore, we illustrate that effective retaining strategies can be identified by training the method on small instances of the problem.

The remainder of this paper is organized as follows. In Section 2 we briefly summarize the BD algorithms, in Section 3 we provide a more thorough description of GPBD, and in Section 4 we introduce general-purpose retaining strategies. In Section 5 we report our experience from an extensive computational study based on instances of the Capacitated Facility Location Problem with Stochastic Demand and, finally, we draw conclusions in Section 6.

2 Benders Decomposition

Consider the following two-stage stochastic program where we assume the uncertain parameters follow a discrete distribution with a set $S$ of possible outcomes:

$$\min_{x \in X} \{c^T x + Q(x) | Ax = b\}$$  \hspace{1cm} (1a)

where $Q(x) = \sum_{s \in S} p_s Q(x, \xi_s)$ is referred to as the recourse function, $p_s$ is the probability of scenario $s \in S$, and

$$Q(x, \xi_s) = \min_y \{q_s^T y | W_s y = h_s - T_s x, y_s \geq 0 \}.$$  \hspace{1cm} (1b)

Here $x \in X \subseteq \mathbb{R}^{n_1}$ are first-stage decision variables and $y \in \mathbb{R}^{n_2}$ are second-stage (recourse) decision variables. Parameters $c \in \mathbb{R}^{n_1}, b \in \mathbb{R}^{m_1}$, and $A \in \mathbb{R}^{m_1 \times n_1}$ are known and deterministic. However, parameters $q_s \in \mathbb{R}^{n_2}$, $h_s \in \mathbb{R}^{m_2}$, $W_s \in \mathbb{R}^{m_2 \times n_2}$ and $T_s \in \mathbb{R}^{m_2 \times n_1}$ are stochastic and depend on the scenario $s$ that eventually materializes. The collection of random parameters is referred to as $\xi_s^T = (q_s^T, W_1^s, \ldots, W_{m_2}^s, T_1^s, \ldots, T_{m_2}^s, h_s^T)$, where $W_i^s$ and $T_i^s$ are the $i$-th row of $W_s$ and $T_s$, respectively.

Benders decomposition first projects problem (1) onto the space of the $x$ variables only, then performs an outer-linearization of the recourse function, and finally relaxes and dynamically reconstructs the constraints generated by the projection and outer-linearization phase [see Geoffrion, 1970a,b]. More specifically, BD generates the following master problem (MP):

$$\begin{align*}
\min & \quad c^T x + \theta \\
\text{s.t.} & \quad Ax = b
\end{align*}$$  \hspace{1cm} (2a)

(2b)
\[ \theta \geq \sum_{s \in S} p_s (\pi^s)T (h_s - T_s x) \quad i = 1, \ldots, I \]  
\[ (\sigma^s)T (h_s - T_s x) \leq 0 \quad j = 1, \ldots, J_s, s \in S \]  
\[ x \in X \]

where \( \pi^s, i = 1, \ldots, I \) are extreme points and \( \sigma^j, j = 1, \ldots, J_s \) extreme rays of the dual to second-stage problem (1b). Constraints (2c) are referred to as \textit{optimality cuts} and represent the outer-linearization of the recourse function, while (2d) are referred to as \textit{feasibility cuts} and represent second-stage feasibility conditions. Constraints (2c)-(2d) are initially relaxed and iteratively reconstructed. At each iteration, given a candidate solution \((\hat{x}, \hat{\theta})\) to MP, second-stage problems are solved as subproblems for all \( s \in S \). If all second-stage problems are feasible and \( \hat{\theta} \geq Q(\hat{x}) \), then \( \hat{x} \) is optimal to (1). Otherwise, a violated constraint (2c) or (2d) is found and added to MP which is the re-solved. In the rest of this treatise, we assume that second-stage variables are continuous as this allows us a convenient representation of duality-based cuts. However, this is without loss of generality as alternative cuts for problems with integer second-stage variables can be found for example in the seminal works of Laporte and Louveaux [1993] and Carøe and Tind [1998]. Finally, note that when \( X \) imposes integrality restrictions on \( x \) the algorithm sketched in this section is embedded in a branch-and-cut framework [see Laporte and Louveaux, 1993].

3 Generalized Partial Benders Decomposition

Generalized Partial Benders Decomposition (GPBD) amounts to dualize and relax only partially the second-stage components of problem (1). Consider, for each \( s \in S \), an integer \( r_s \) such that \( 0 \leq r_s \leq n2 - n \) – we remind the reader that \( n2 \) is number of second-stage variables, see Section 2. Let \( y^R_s = (y_s)i=1,...,r_s \) be a selection of \( r_s \) second-stage variables from \( y_s \) we wish to retain in the master problem, and \( y^{NR}_s = (y_s)i=r_s+1,...,n2 \) the remaining variable in \( y_s \). The \( s \)-th second-stage problem can be reformulated, without loss of generality, as follows:

\[ \min q^R_s y^R_s + q^{NR}_s y^{NR}_s \]  
\[ \begin{bmatrix} W^R_s & 0 \\ W^{NR_1}_s & W^{NR_2}_s \end{bmatrix} \begin{bmatrix} y^R_s \\ y^{NR}_s \end{bmatrix} = \begin{bmatrix} h^R_s \\ h^{NR}_s \end{bmatrix} - \begin{bmatrix} T^R_s \\ T^{NR}_s \end{bmatrix} x \]  
\[ y^R_s, y^{NR}_s \geq 0 \]

where \( q^R_s \) and \( q^{NR}_s \) are an \( r_s \)-dimensional and an \((n2 - r_s)\)-dimensional sub-vector of \( q_s \), respectively, \( W^R_s \) is a \( v \times r_s \) sub-matrix of \( W_s \), \( 0 \) is a \( v \times (n2 - r_s) \) zero-matrix, \( W^{NR_1}_s \) is an \((m2 - v) \times r_s \), possibly null, sub-matrix of \( W_s \), and \( W^{NR_2}_s \) is an \((m2 - v) \times (n2 - r_s) \) non-null sub-matrix of \( W_s \). Furthermore, \( h^R_s \) and \( h^{NR}_s \) are a \( v \)-dimensional and an \((m2 - v) \)-dimensional sub-vector of \( h_s \), respectively, and finally \( T^R_s \) and \( T^{NR}_s \) are a \( v \times n1 \) and an \((m2 - v) \times n1 \) sub-matrix of \( T_s \), respectively. Thus, \( 0 \leq v \leq m2 \) is the number of second-stage constraints involving only \( y^R_s \) variables (i.e., the second-stage constraints where the coefficients of \( y^{NR}_s \) are null) and depends on the specific selection of \( y^R_s \) variables which is discussed in Section 4. For now we assume
that a selection of second-stage variables $y_s^R$ exists for each $s \in S$. GPBD consists of projecting the original problem onto the subspace defined by the $x$ and $(y_s^R)_{s \in S}$, decision variables, thus we can formulate the master problem (MP) as follows:

$$\min \ c^T x + \sum_{s \in S} p_s q_s^R y_s^R + \theta$$

s.t. $Ax = b$, 

$$W_s^R y_s^R = h_s^R - T_s^R x, \quad s \in S, \quad (4b)$$

$$\theta \geq \sum_{s \in S} p_s (\pi_i^s)^T (h_s^{NR} - T_s^{NR} x - W_s^{NR1} y_s^R), \quad i = 1, \ldots, I, \quad (4c)$$

$$(\sigma_j^s)^T (h_s^{NR} - T_s^{NR} x - W_s^{NR1} y_s^R) \leq 0, \quad j = 1, \ldots, J, s \in S, \quad (4d)$$

$$x \in X, \quad (4e)$$

$$y_s^R \geq 0, \quad s \in S \quad (4f)$$

Notice the following differences with respect to the master problem in BD (2). The objective function (4a) includes a term for the cost of the variables retained. Constraints (4c) are second-stage constraints which are added to MP as a consequence of retaining variables $(y_s^R)_{s \in S}$. The optimality and feasibility cuts, (4d) and (4d), now account for the fact that variables $y_s^R$ belong to the MP and are thus right-hand-side parameters in the subproblems. Finally, constraints (4g) set the range for the variables $y_s^R$ retained in MP. Thus, given a solution $(\hat{x}, \hat{y}_1^R, \ldots, \hat{y}_{|S|}^R, \hat{\theta})$ to MP, for each scenario $s$ we solve the following subproblem:

$$\min \ q_s^{NR} y_s^{NR}$$

$$W_s^{NR2} y_s^{NR} = h_s^{NR} - T_s^{NR} \hat{x} - W_s^{NR1} y_s^R$$

$$y_s^{NR} \geq 0$$

Notice that the constraints which involve $y_s^{NR}$ (and possibly also $y_s^R$) variables belong exclusively to the subproblems, while the constraints involving only the $y_s^R$ variables are in MP, see (4c). The algorithm is thus exactly the same as for BD (see Section 2) except for a preprocessing phase which consists of selecting the $y_s^R$ variables to keep in MP for each $s \in S$.

In general, GPBD amounts to solving slightly bigger MPs and slightly smaller subproblem compared with BD as the MP formulation is strengthened due to the addition of constraints (4c). The rationale behind GPBD is that the second-stage variables and, particularly, the constraints (4c) retained, might help MP provide higher quality solutions and better bounds already in the initial iterations of the algorithm. This in turn may correspond to shorter computation times and fewer cuts required. However, while the search space of MP is restricted, MP is heavier and in principle more difficult to solve. Thus, it is necessary to trade off the benefits of additional structure and the burden of a more difficult MP. Compared with PBD, the MP in GPBD is, in general, not augmented by full subproblems. Rather, GPBD provides the option of retaining (for some or all of the subproblems) only the structure which is thought more relevant for MP. This might in turn yield the advantages of an augmented formulation with a relatively little increase in the size and complexity of the MP. Finally, notice that BD is an instance of GPBD.
where \( r_s = 0 \) for all \( s \in S \), while PBD is an instance of GPBD where \( r_s = n^2 \) for selected \( s \in S \). Thus, GPBD can be thought of as a generalization of PBD.

## 4 Retaining Strategies

A crucial step in GPBD is that of selecting, for each scenario \( s \in S \), variables \( y_{Rs} \) to be retained in MP. In Section 3 we assumed, without loss of generality, that the first \( r_s \) variables of subproblem \( s \) were retained. More generally, this step amounts to selecting \( r_s \) out of \( n^2 \) variables, thus a subset \( R_s \subseteq \{1, \ldots, n^2\} \), such that \( |R_s| = r_s \). Let, thus, a Retaining Strategy (RS) be a criteria for selecting subsets \( R_s \) of \( \{1, \ldots, n^2\} \). While the selection of \( R_s \) can be arbitrary, a successful application of GPBD requires some intelligence. In fact, selecting variables \( y_{Rs} \) which are not tied by second-stage constraints (i.e., a \( W_{Rs} \) matrix – see (3b)) is arguably of little practical use. This would amount to adding unconstrained variables to MP (similar to \( \theta \)) whose value would have to be adjusted through the addition of cuts, and providing as such very little information. Instead, we advocate retaining second-stage variables generating (4c) constraints which are “representative” of selected groups of second-stage constraints. In what follows, we propose a number of general-purpose RSs following this line of reasoning. While any type of sophisticated rule can be applied, the RSs we propose are chosen following the criteria of “simplicity”, meaning that they are based on simple mnemonic rules. We proceed as follows: first, given a set of representative constraints, we illustrate how the variables to retain in the MP (i.e., the subset \( R_s \)) are found and, second, we clarify what a group of constraints is in our context and propose simple rules for identifying representative constraints for a given group.

Assume a set \( N_s \subseteq \{1, \ldots, m^2\} \) of representative constraints to retain for subproblem \( s \) has been identified. We define the set of variables retained as \( R_s = \{ i : \exists W_{sji} \neq 0, 1 \leq i \leq n^2, j \in N_s \} \), where \( W_{sji} \) is the \( i, j \)-th element of \( W_s \). That is, we retain the variables \( y_{Rs} \) which appear with a non-null \( W_{sji} \) coefficient at least once in the representative constraints. This corresponds to include in MP all the variables which are strictly necessary for generating the desired constraints (4c). As an example, consider the \( W_s \) matrix (6), and assume we want to represent the first block of constraints \( \{1, \ldots, 4\} \), i.e., \( N_s = \{1,3\} \). We can then retain variables \( R_s = \{1, 2, 3\} \) as there exist at least one \( W_{sji} \neq 0 \), for \( i = 1, \ldots, 3 \) and \( j \in N_s \).

\[
W_s = \begin{bmatrix}
4 & 5 & 5 & 0 & 0 & \ldots & 0 \\
3 & 0 & 5 & 3 & 0 & \ldots & 0 \\
0 & 1 & 2 & 0 & 0 & \ldots & 0 \\
1 & 0 & 1 & 0 & 0 & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots
\end{bmatrix}
\]

(6)

Generally, blocks of constraints can be easily identified in the structure \( W_s y_s = h_s - T_s x \) in (1b). A block of constraints typically represents the same requirements applied to a number of similar entities. As an example, in the Capacitated Facility Location Problem at least two blocks of constraints can be identified, one ensuring that demand is satisfied for each customer, the other ensuring that the capacity of each facility is respected [see, e.g., Fischetti et al., 2016a,b].
In the basic version of the Stochastic Unit Commitment Problem it is possible to identify one block of constraints ensuring the satisfaction of power demand, and another ensuring that each generating unit produces within the respective operating range [see, e.g., Shiina and Birge, 2004]. However, several additional blocks of restrictions, such as ramping limits, are typically included [see, e.g., Papavasiliou and Oren, 2013]. Finally, in the Stochastic Network Design Problem it is possible to find at least a block of flow-conservation constraint and a block of constraints which ensures that the capacity of individual network links is respected [see, e.g., Crainic et al., 2011, 2016]. Thus, it is generally straightforward to identify different blocks of constraints, each representing a different physical-world requirement.

Let \( m \) be the number of blocks of constraints in the second-stage problem and \( \mathcal{M}_1, \ldots, \mathcal{M}_m \) a partition of \( \{1, \ldots, m\} \), i.e., disjoint groups of second-stage constraints. We refer to the rows of \( W_s \) corresponding to the \( i \)-th block of constraints \( \mathcal{M}_i \) as \( W_s(i) = (W_s^j)_{j \in \mathcal{M}_i} \), where \( W_s^j \) is the \( j \)-th row of \( W_s \). Similarly, let \( T_s(i) = (T_s^j)_{j \in \mathcal{M}_i} \), and \( h_s(i) = (h_s^j)_{j \in \mathcal{M}_i} \), where \( h_s^j \) is the \( j \)-th element of vector \( h_s \). Thus, \( W_s(i), T_s(i) \) and \( h_s(i) \) is the data describing the \( i \)-th block of constraints. Such data is used by the RSs we propose.

We propose a number of general-purpose RSs based on simple but widely applicable rules for selecting representative constraints and consequently variables to retain. Clearly, more sophisticated method could be employed. However, for the scope of this paper we prefer to test GPBD only with simple selection rules in order to show its potential. The RSs we propose are summarized in Appendix A. Most of the RSs we propose are based on finding highest and lowest values, or on finding random samples. The most sophisticated RS we propose is based on a clustering algorithm which is easy implementable with standard programming languages. The rationale behind the RSs proposed is that once MP is aware of “extreme” constraints, such as ramping limits, it is possible to tailor its solutions to the second-stage information provided, and thus produce better first-stage solutions already in the initial phases of the algorithm, in turn improving convergence.

In RSs \( H^W(i,n), H^T(i,n) \) and \( H^h(i,n) \) the letter \( H \) stands for “highest”. These RSs have the scope of representing constraints block \( i \) by the \( n \) constraints of the block with the “highest” \( W_s^j \), \( j \in \mathcal{M}_i \), values (respectively, \( T_s^j \) and \( h_s^j \)). More precisely, let \( W_s^{(j)} \) be the \( j \)-th order statistic where \( \sum_{k=1}^{n_2} W_s^{k,(1)} \geq \sum_{k=1}^{n_2} W_s^{k,(2)} \geq \cdots \geq \sum_{k=1}^{n_2} W_s^{k,(|\mathcal{M}_i|)} \). Similarly, let \( T_s^{(j)} \) and \( h_s^{(j)} \) be order statistics such that \( \sum_{k=1}^{n_1} T_s^{k,(1)} \geq \sum_{k=1}^{n_1} T_s^{k,(2)} \geq \cdots \geq \sum_{k=1}^{n_1} T_s^{k,(|\mathcal{M}_i|)} \) and \( h_s^{(1)} \geq h_s^{(2)} \geq \cdots \geq h_s^{(|\mathcal{M}_i|)} \). For constraints block \( i \) let the representative constraints be \( N_s = \{ j : W_s^j \in \{W_s^{1,(1)}, \ldots, W_s^{1,(n_2)}\} \} \) (respectively \( N_s = \{ j : T_s^j \in \{T_s^{1,(1)}, \ldots, T_s^{1,|\mathcal{M}_i|}\} \}) and \( N_s = \{ j : h_s^j \in \{h_s^{1,(1)}, \ldots, h_s^{1,|\mathcal{M}_i|}\} \}). Thus, RSs \( H^W(i,n), H^T(i,n) \) and \( H^h(i,n) \) amount to representing the \( i \)-th block by means of the constraints with the \( n \) highest sums of \( W_s \) coefficients, with the \( n \) highest sums of \( T_s \) coefficients, and with the \( n \) highest \( h_s \) coefficients, respectively.

In RSs \( L^W(i,n), L^T(i,n) \) and \( L^h(i,n) \) the letter \( L \) stands for “lowest”. These RSs have the scope of representing constraints block \( i \) by the \( n \) constraints of the block with the “lowest” \( W_s^j \), \( j \in \mathcal{M}_i \), values (respectively, \( T_s^j \) and \( h_s^j \)). For constraints block \( i \) let the representative constraints be \( N_s = \{ j : W_s^j \in \{W_s^{(|\mathcal{M}_i|-n+1,1), \ldots, W_s^{(|\mathcal{M}_i|)}\} \} \) (respectively \( N_s = \{ j : T_s^j \in \{T_s^{(|\mathcal{M}_i|-n+1,1), \ldots, T_s^{(|\mathcal{M}_i|)}\} \}) and \( N_s = \{ j : h_s^j \in \{h_s^{(|\mathcal{M}_i|-n+1,1), \ldots, h_s^{(|\mathcal{M}_i|)}\} \}). Thus, RSs
\( L^W(i,n), L^T(i,n) \) and \( L^h(i,n) \) amount to representing the \( i \)-th block by means of the constraints with the \( n \) lowest sums of \( W_s \) coefficients, with the \( n \) lowest sums of \( T_s \) coefficients, and with the \( n \) lowest \( h_s \) coefficients, respectively.

In RSs \( EX^W(i,n), EX^T(i,n) \) and \( EX^h(i,n) \) the letters \( EX \) stand for “extremes”. These RSs represent a combination between the \( H \) and the \( L \)-type strategies. That is, they have the scope of representing constraints block \( i \) by means of the \( n \) constraints with the “lowest” and “highest” \( W_j \) values (respectively, \( T_j \) and \( h_j \) \( j \in M_i \)). Thus RSs \( EX^W(i,n), EX^T(i,n) \) and \( EX^h(i,n) \) retain twice as many constraints as the corresponding \( H \) and \( L \)-type strategies.

In RS \( S(i,n) \) the letter \( S \) stands for “sampling”. This RS amounts to representing constraints \( M_i \) by sampling \( n \) of its constraints. Any sampling technique can be adopted. In Section 5 we use random sampling.

Finally, in RSs \( C^W(i,n), C^T(i,n) \) and \( C^h(i,n) \) the letter \( C \) stands for “clustering”. These RSs cluster the constraints in block \( i \in \{1, \ldots, m\} \) into \( n \) subsets of “similar” constraints, and then, for each cluster, choose one constraint as representative of the cluster. In RS \( C^W(i,n) \) the population of constraints is clustered based on the similarity between the rows of \( W_s(i) \). Similarly, in \( C^T(i,n) \) and \( C^h(i,n) \), constraints \( M_i \) are clustered based on the similarity between the rows of \( T_s(i) \) and the elements of \( h_s(i) \), respectively. For each cluster \( c = 1, \ldots, n \), the point \( j \in M_i \) closest to the centroid of the cluster is selected as the representative for the cluster. Constraint \( j \) is added to \( N_s \) and the variables forming constraint \( j \) will be retained (as well as constraint \( j \) itself). Any measure of similarity and clustering algorithm can be applied (in Section 5 we use the Euclidean distance and the \( k\)-means++ algorithm of Arthur and Vassilvitskii [2007]).

Additional RSs can be thought of by combining the ones proposed in this section in such a way to represent different groups of constraints, possibly using a different strategy for each of them.

## 5 Computational Study

In this section we report on our computational experience based on several difficult instances of the Capacitated Facility Location Problem with Stochastic Demand (CFLPSD). The computational study comprises a training phase and test phase. In the training phase the GPBD algorithm is used to solve a set of small training instances with the scope of identifying the most promising RSs. In the test phase GPBD, with the selected RSs, is used to solve a set of test instances which comprise larger and numerically different problems.

GPBD and BD are implemented in Java using the Cplex 12.6.2 callable libraries. Cplex 12.6.2 is also used to implement and solve (when possible) the extensive problem. The implementation includes default solver settings (except for the tolerances necessary to ensure cuts are correctly enforced) and no additional algorithmic enhancement that could theoretically improve the results obtained. This allows us to isolate the effect of a revisited decomposition strategy. Possible algorithmic enhancements, such as stabilization, are discussed in Section 5.5. All tests
are performed on a cluster of machines equipped with 12 × 2.39 GHz CPU and 23.59 GB RAM. All training and test instances are solved to a target relative optimality gap of $10^{-6}$.

5.1 The Capacitated Facility Location Problem with Stochastic Demand

The Capacitated Facility Location Problem with Stochastic Demand (CFLPSD) consists of choosing which facilities to open, given a set of candidates, and how to assign uncertain customer demand to the open facilities. Facility location decisions must be made before the demand become known and are thus first-stage decisions, while the allocation of demand to facilities can be decided once the demand become known. The CFLPSD can be thus modeled as a two-stage stochastic program with recourse. The CFLPSD is NP-Hard as the deterministic Capacitated Facility Location Problem, which is a special case of the CFLPSD, is known to be NP-hard [Cornuejols et al., 1991].

An instance of the CFLPSD is made of a set of facilities $I$, a set of customers $J$, and a set of demand scenarios $S$, the probability $\pi_s$ for each $s \in S$, the opening cost $F_i$ and the capacity $Q_i$ for each facility $i \in I$, the cost $C_{ij}$ of allocating one unit of the demand of customer $j \in J$ to facility $i \in I$ and, finally, $D_{js}$ the demand of customer $j \in J$ under scenario $s \in S$. Let binary variable $x_i$ be equal 1 if facility $i \in I$ is open, 0 otherwise, and let continuous variable $y_{ijs}$ represent the amount of demand of customer $j \in J$ allocated to facility $i \in I$ under scenario $s \in S$. The CFLPSD is thus:

$$\min \left\{ \sum_{i \in I} F_i x_i + Q(x) \mid x_i \in \{0, 1\}, i \in I \right\}. \quad (7a)$$

where $x = (x_i)_{i \in I}$, $Q(x) = \sum_{s \in S} \pi_s Q(x, s)$ and

$$Q(x, s) = \min \sum_{i \in I} \sum_{j \in J} C_{ij} y_{ijs} \quad (7b)$$

s.t.

$$\sum_{i \in I} y_{ijs} = D_{js}, \quad j \in J, \quad (7c)$$

$$y_{ijs} \leq Q_i x_i, \quad i \in I, j \in J, \quad (7d)$$

$$\sum_{j \in J} y_{ijs} \leq Q_i x_i, \quad i \in I, \quad (7e)$$

$$y_{ijs} \geq 0 \quad i \in I, j \in J. \quad (7f)$$

The objective function in (7a) consists of the sum of opening costs and expected cost of fulfilling customer demands. First-stage decision are of a binary type, thus problem (7) is a two-stage stochastic program with integer first-stage. In the second-stage problem, objective function (7b) represents the cost of allocating demand to facilities given a scenario $s$ and a first-stage decision $x$. Constraints (7c) ensure that the demand of each customer is completely satisfied, constraints (7d) ensure that demand is allocated to a facility only if the facility is open and, finally, constraints (7e) ensure that the capacity of each facility is respected. Constraints (7f) set the range for the second-stage variables. Furthermore, we add constant $\sum_{i \in I} Q_i x_i \geq \max_{s \in S} \sum_{j \in J} D_{js}$ which ensures second-stage feasibility and is known to be useful in a BD approach [see, e.g., Fischetti et al., 2016a]. Finally, note that constraints (7d) are redundant.
in this formulation, but they are known to yield tighter linear programming relaxation bounds [Wentges, 1996].

To the best of our knowledge, no available set of benchmark instances exists for the CFLPSD. For this reason we use the well-known instances of the Capacitated Facility Location Problem generated by Beasley [1988], also available in the OR-library, and extend them by including stochastic demand. The instances in Beasley [1988] are randomly generated and have been used by several authors including, e.g., Wentges [1996], Guastaroba and Speranza [2012] and Fischetti et al. [2016a]. Consistently with Fischetti et al. [2016a], we use the instances sets A, B and C as they generate large and difficult problems. Each set contains four instances, and all instances have 100 facilities and 1000 customers. In order to obtain stochastic demand we follow the procedure adopted by Laporte et al. [1994] for a facility location problem. Particularly, customer demands are assumed normally distributed with mean equal to the deterministic value of the demand in Beasley [1988]. Different instances are obtained by setting different numbers of scenarios, values of demand standard deviation (SD – indicated as a percentage of the mean demand) and percentages of strongly correlated customer demands (C – a 0.8 correlation is used), as summarized in Table 1. Particularly, the set of test instances is made as follows: for each instance in Beasley [1988] and for each $|S| \in \{30, 60, 90\}$, there exists one instance for each $SD \in \{5, 10, 20\%\}$ and $C = 0\%$, and one instance for each $C \in \{20, 50, 80\%\}$ and $SD = 3\%$, for a total of 216 instances. The set of training instances contains, for each instance in Beasley [1988], one instances $|S| = 5$, $SD = 3\%$ and $C = 0\%$, for a total of 12 instances. Scenarios are obtained by sampling realizations from the underlying normal distribution.

Table 1: Instances of the CFLPSD.

| Instances | Type | $|S|$ | St. Deviation | Correlated Demands | # Instances |
|-----------|------|------|--------------|-------------------|-------------|
| Training  | Beasley [1988]'s A,B,C | 5 | 3\% | 0 | 12 |
| Test      | Beasley [1988]'s A,B,C | 30,60,90 | 3.5,10,20\% | 0, 20, 50, 80 \% | 216 |

5.2 Retaining Strategies for the CFLPSD

We use several of the general-purpose RSs introduced in Section 4 as well as additional problem-specific RSs. In problem (7) it is possible to identify two groups of constraints. The first group, (7c) - which we refer to as A - ensures satisfaction of the demand. The second group, (7e) - which we refer to as B - ensures the respect of the capacity of the facilities. The constraints in block A can be distinguished by their right-hand-side only, $h_A(A) = (D_{j,s})_{j \in J}$, while the constraints in block B can be distinguished by the rows of the $T_s(B)$ matrix only, which corresponds to an $|I| \times |I|$ diagonal matrix where the elements on the diagonal are $Q_i$ for $i \in I$. For this reason, we adopt only RSs $C^h(A,n)$, $H^h(A,n)$, $L^h(A,n)$, $EX^h(A,n)$, $C^T(B,n)$, $HT(B,n)$, $LT(B,n)$, $EXT(B,n)$, in addition to $S(A,n)$ and $S(B,n)$.

We also consider a number of problem-specific RSs. RS $C^F(B,n)$ consists of partitioning locations $i$ into $n$ clusters based on their opening costs. For each cluster, the point closest to the centroid is chosen as representative, and the constraint B for the corresponding locations are retained. RSs $H^F(B,n)$, $L^F(B,n)$, $EX^F(B,n)$ consist of selecting the constraints
B corresponding to the location with the highest, lowest, and extreme (highest and lowest) opening costs, respectively. Similarly, RS $C(A, n)$ consists of partitioning customers $j$ into $n$ clusters based on their allocation costs. For each cluster, the point closest to the centroid is chosen as representative, and the constraint $A$ for the corresponding customers are retained. RSs $H(A, n)$, $L(A, n)$, $EX(A, n)$ consist of selecting the constraints $A$ corresponding to the customers with the highest, lowest, and extreme (highest and lowest) allocation costs, respectively. Particularly, for each customer $j \in J$ the allocation costs considered is the maximum among all locations, that is, $\max_{i \in I} \{C_{ij}\}$.

Finally, we adopt a number of RSs which allow us to evaluate whether it is profitable to represent more than one group of constraints at a time. For example, RSs $EX_{h,F}(n)$ consists of the simultaneous adoption of RSs $EX_{h}(A, n)$ and $EX_{F}(B, n)$. A summary of all RSs can be found in Appendix A.

### 5.3 Training Phase

The scope of the training phase is that of identifying, by means of a set of small training instances, the RSs to use in the test phase. For the sake of our computational study, we perform an extensive training phase. All the training instances are solved with BD and GPBD (using all applicable RSs – see Section 5.2 – on all subproblems), with a 10 minute time limit. The RSs (if any) which provide better solutions than BD are then selected for the test phase. For the RSs used, values $n = 1$ and 2 are used. In fact, preliminary tests show that the performance of GPBD worsens with more constraints retained, providing a preliminary confirmation of the soundness of retaining only portions of subproblems.

Table 2 reports the results of the training phase. The statistics reported in Table 2 are the average gap obtained after reaching the time limit (Gap [%]), the percentage of instances solved to optimality (Sol %) and the average solution time in second (Time [s]). Consistently with Cplex statistics [IBM, 2015], the optimality gap is calculated as $100 \times (\text{BestInteger} - \text{BestBound})/\text{BestBound}$, where $\text{BestInteger}$ is the best objective value found with the specific RS and for the specific instance while $\text{BestBound}$ is the best available bound for the corresponding instance. This calculation is consistent with recent work such as Guastaroba and Speranza [2012] and Fischetti et al. [2016a]. The results in Table 2 are shown with increasing total average gap.

A number of elements can be pointed out. First, there exist several RSs for which GPBD outperforms BD but also a number of RSs for which GPBD does not improve BD and a number of RSs for which GPBD performs worse than solving the extensive problem. Second, the RSs based on representing constraints block $A$ are in general successful. All RSs applied to constraints block $A$ solve the same number of instances solved by BD, and all except $H_{h}(A, 2)$ reduce (in some cases significantly) the average optimality gap. On the contrary, the RSs based on representing constraints block $B$ lead to worse results than BD. It is worth noticing that constraints $A$ are affected by uncertainty in the right-hand-side $h$, while constraints $B$ are not affected by uncertainty. Third, when using RSs $EX_{h,T}(n)$, $EX_{h,F}(n)$, $EX_{C,T}(n)$ and $EX_{C,F}(n)$, GPBD is always outperformed by BD and in some cases also by Cplex. This is due
to the fact that such RSs retain more constraints (4n) than simple RSs. This finding illustrates that in this case generating a heavy MP is detrimental to a partial decomposition strategy, and confirms the need of methods to calibrate the size of the MP in the spirit of GPBD. As a result of the training phase, all the RSs applied to constraints block A will be used in the test phase.

The results are sorted with increasing global average gap across all instances. The results for BD and Cplex are shaded.

5.4 Test Phase

The scope the test phase is to assess whether i) GPBD yields any advantage compared to BD on large-scale problems, ii) the performance of GPBD is stable across heterogeneous instances, and iii) it is possible to rely on the training phase for selecting RSs. The test phase consists of solving the test instances by means of BD and GPBD using the RSs selected in the training phase, namely $H^b(A,n)$, $L^h(A,n)$, $EX^h(A,n)$, $C^b(A,n)$, $H^C(A,n)$, $L^C(A,n)$, $EX^C(A,n)$, $C^C(A,n)$ and $S(A,n)$, with $n = 1, 2$. The RSs are applied to an increasing percentage (REP)
of representative scenario subproblems, namely 20, 50, 80 and 100%. The subproblems are selected using the representation strategy described by Crainic et al. [2014]. All instances are solved with a 2400 second time limit.

Table 3, Table 4 and Table 5 report the results with different correlations settings for the test instances with $|S| = 30, 60$ and 90, respectively. Similarly, Table 6, Table 7 and Table 8 report the results with different standard deviation settings for the test instances with $|S| = 30, 60$ and 90 scenarios, respectively. The statistics reported are the average gap (Gap $[\%]$) with respect to the best known bound (see Section 5.3), the smallest and highest optimality gaps (Min$[\%]$ and Max$[\%]$, respectively), and the percentage of instances solved to optimality (Sol$[\%]$).

Tables 3 to 8 illustrate that GPBD has significant potential to improve BD and that, when the least number of subproblems are represented (i.e., REP=20%), GPBD outperforms BD in the great majority of the cases. In several cases the optimality gap reduction is significantly high. Furthermore, the gap reduction appears even more substantial as the size of the instances increases. As an example, in Table 5 RS $S(A,1)$ with REP = 20%, reduces the average BD optimality gap by almost ten percentage points for all correlations settings. RS $S(A,1)$ is highly effective also on the instances with $|S| = 90$ and different standard deviations (see Table 8).

Remarkably, RS $S(A,1)$ simply amounts to adding a random constraint from block A to MP, illustrating that very simple RSs can generate significant improvements. In Table 8 it is also possible to notice that RSs $L^h(A,n)$ nearly halves the optimality gap of BD for the instances with $SD = 5\%$. Its performances are confirmed also on the largest instances with different correlations settings (Table 5). Finally, the highest optimality gap (Max$[\%]$) is often much smaller than for BD, particularly on the largest instances. For the instances with $|S| = 30$ and 60, the optimality gap is also consistently smaller than for BD.

The results show also that, in general, the performances of GPBD worsen as the number of scenarios represented (REP) increases, particularly on the largest instances. On the one hand, this illustrates that increasing the size of MP excessively is detrimental, on the other hand, it certifies the need for tailoring the decomposition strategy to the specific problem in the spirit of GPBD. Finally, we can address the first scope of the computational study by concluding that GPBD significantly improves BD with all RSs considered, and particularly when a few scenario subproblems are represented. Remarkably, a very simple RS consisting of retaining a random constraint, provides a substantial reduction of the optimality gap, particularly on the largest instances.

To address the second question of the computational study, Tables 3 to 8 illustrate that the performance of GPBD is stable across all instances tested when the number of subproblems is represented is small. Consider the cases with REP $\leq 50\%$. It can be noticed that GPBD outperforms BD on the great majority of the instances independently on the correlations and standard deviation settings.

Finally, it emerges that training the GPBD on small instances is a suitable way for identifying the most effective RSs. All the RSs selected in the training phase outperform BD across a large set of instances with different number of scenarios, standard deviations and correlations. Furthermore, the training instances were solved with a time limit of 600 seconds. This shows that
a suitable way for identifying efficient RSs consists of solving a small-scale instance (possibly not to optimality) with a few well-thought RSs and selecting the RSs which yield the best results. Our computational experience shows that the RSs which perform well on small-scale instances maintain their performances also on large-scale instances.

Table 3: Minimum (Min), average (Gap) and maximum (Max) optimality gaps and percentage of instances solved (Sol) across the A, B and C instances with |S| = 30 and 3% standard deviation of the demand, for different percentages C of customer demands strongly correlated (0.8) and subproblems represented REP. Shaded cells correspond to average optimality gaps lower than for BD.

<table>
<thead>
<tr>
<th>RS</th>
<th>REP</th>
<th>C=20%</th>
<th>C=50%</th>
<th>C=80%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min [%]</td>
<td>Gap [%]</td>
<td>Max [%]</td>
</tr>
<tr>
<td>BD</td>
<td>0</td>
<td>0.000</td>
<td>10.434</td>
<td>20.149</td>
</tr>
<tr>
<td>H^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>10.648</td>
<td>21.055</td>
</tr>
<tr>
<td>H^(A,2)</td>
<td>20</td>
<td>0.000</td>
<td>9.506</td>
<td>21.047</td>
</tr>
<tr>
<td>L^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>10.143</td>
<td>19.896</td>
</tr>
<tr>
<td>L^(A,2)</td>
<td>20</td>
<td>0.000</td>
<td>9.837</td>
<td>22.174</td>
</tr>
<tr>
<td>H^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>10.772</td>
<td>21.911</td>
</tr>
<tr>
<td>H^(A,2)</td>
<td>20</td>
<td>3.310</td>
<td>11.093</td>
<td>17.128</td>
</tr>
<tr>
<td>S^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>10.039</td>
<td>15.137</td>
</tr>
<tr>
<td>S^(A,2)</td>
<td>20</td>
<td>0.000</td>
<td>9.739</td>
<td>23.438</td>
</tr>
<tr>
<td>E^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>10.185</td>
<td>24.455</td>
</tr>
<tr>
<td>E^(A,2)</td>
<td>20</td>
<td>0.000</td>
<td>8.769</td>
<td>16.995</td>
</tr>
<tr>
<td>C^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>8.948</td>
<td>17.795</td>
</tr>
<tr>
<td>C^(A,2)</td>
<td>20</td>
<td>0.000</td>
<td>18.874</td>
<td>15.851</td>
</tr>
<tr>
<td>E^(A,1)</td>
<td>20</td>
<td>0.681</td>
<td>10.631</td>
<td>18.229</td>
</tr>
<tr>
<td>E^(A,2)</td>
<td>20</td>
<td>0.681</td>
<td>9.942</td>
<td>16.277</td>
</tr>
<tr>
<td>C^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>11.648</td>
<td>20.869</td>
</tr>
<tr>
<td>C^(A,2)</td>
<td>20</td>
<td>2.791</td>
<td>10.765</td>
<td>16.341</td>
</tr>
<tr>
<td>L^(A,1)</td>
<td>20</td>
<td>0.000</td>
<td>10.591</td>
<td>20.064</td>
</tr>
<tr>
<td>L^(A,2)</td>
<td>20</td>
<td>2.187</td>
<td>10.320</td>
<td>20.739</td>
</tr>
</tbody>
</table>

- $H^{A,1}$
- $H^{A,2}$
- $L^{A,1}$
- $L^{A,2}$
- $S^{A,1}$
- $S^{A,2}$
- $E^{A,1}$
- $E^{A,2}$
- $C^{A,1}$
- $C^{A,2}$
- $EX^{A,1}$
- $EX^{A,2}$
- $CX^{A,1}$
- $CX^{A,2}$
- $EX^{C,1}$
- $EX^{C,2}$
- $C^{L,1}$
- $C^{L,2}$
- $L^{C,1}$
- $L^{C,2}$

14
Table 4: Minimum (Min), average (Gap) and maximum (Max) optimality gaps and percentage of instances solved (Sol) across the A, B and C instances with $|S| = 60$ and 30% standard deviation of the demand for different percentages $C$ of customer demands strongly correlated (0.8) and subproblems represented REP. Shaded cells correspond to average optimality gaps lower than for BD.

| RS | REP | $C=20\%$ | $C=50\%$ | $C=80\%$
|---|---|---|---|---
| BD | Min [%] | Gap [%] | Max [%] | Sol [%] | Min [%] | Gap [%] | Max [%] | Sol [%] | Min [%] | Gap [%] | Max [%] | Sol [%]
| $H^A(A, 2)$ | 100 | 0.000 | 9.301 | 14.800 | 8.3 | 0.418 | 10.575 | 20.497 | 0.0 | 0.000 | 9.315 | 16.218 | 8.3
| $S(A, 1)$ | 100 | 0.000 | 8.901 | 13.420 | 8.3 | 0.000 | 8.875 | 14.239 | 8.3 | 0.000 | 8.547 | 13.662 | 8.3
| $S(A, 2)$ | 100 | 0.000 | 9.433 | 16.482 | 8.3 | 0.418 | 10.129 | 17.072 | 0.0 | 0.000 | 10.080 | 17.471 | 8.3
| $EX^B(A, 1)$ | 100 | 0.000 | 9.931 | 18.786 | 8.3 | 0.000 | 10.200 | 19.086 | 8.3 | 0.000 | 10.288 | 16.664 | 8.3
| $EX^C(A, 2)$ | 100 | 0.681 | 13.359 | 23.808 | 0.0 | 0.418 | 11.946 | 24.966 | 0.0 | 0.000 | 13.480 | 27.485 | 8.3
| $C^A(A, 1)$ | 100 | 0.000 | 9.047 | 16.227 | 8.3 | 0.418 | 9.866 | 15.417 | 8.3 | 0.000 | 10.311 | 18.899 | 8.3
| $C^A(A, 2)$ | 100 | 0.000 | 11.269 | 18.408 | 8.3 | 0.000 | 10.082 | 17.846 | 0.0 | 0.000 | 10.060 | 15.662 | 8.3
| $EX^C(A, 1)$ | 100 | 0.412 | 9.806 | 18.105 | 0.0 | 0.000 | 8.785 | 19.135 | 0.0 | 0.000 | 9.316 | 17.755 | 8.3
| $EX^C(A, 2)$ | 100 | 1.666 | 11.171 | 21.604 | 0.0 | 1.667 | 12.171 | 27.153 | 0.0 | 0.000 | 11.197 | 19.744 | 8.3
| $C^C(A, 1)$ | 100 | 0.000 | 12.585 | 20.589 | 8.3 | 0.000 | 9.554 | 16.069 | 8.3 | 0.000 | 9.190 | 16.069 | 8.3
| $C^C(A, 2)$ | 100 | 0.412 | 9.739 | 16.730 | 0.0 | 0.418 | 9.706 | 16.153 | 0.0 | 0.000 | 9.626 | 14.077 | 8.3
| $L^C(A, 1)$ | 100 | 2.412 | 11.286 | 18.147 | 0.0 | 0.418 | 10.620 | 16.627 | 0.0 | 0.000 | 10.542 | 16.609 | 8.3
| $L^C(A, 2)$ | 100 | 0.000 | 9.486 | 15.324 | 8.3 | 0.000 | 8.675 | 14.969 | 8.3 | 0.000 | 9.140 | 13.931 | 8.3
| \( H^C(A, 1) \) | 100 | 6.143 | 19.209 | 34.886 | 0.0 | 4.434 | 17.034 | 36.748 | 0.0 | 4.110 | 17.491 | 30.328 | 0.0 |
| \( H^C(A, 2) \) | 100 | 11.869 | 29.979 | 41.893 | 0.0 | 5.124 | 23.590 | 40.407 | 0.0 | 10.857 | 21.987 | 29.855 | 0.0 |
| \( L^C(A, 1) \) | 100 | 3.989 | 14.556 | 23.726 | 0.0 | 3.318 | 13.557 | 23.959 | 0.0 | 4.373 | 17.742 | 19.832 | 0.0 |
| \( L^C(A, 2) \) | 100 | 9.038 | 24.144 | 38.328 | 0.0 | 11.629 | 24.500 | 44.297 | 0.0 | 5.324 | 21.157 | 35.832 | 0.0 |
| \( H^C(A, 1) \) | 100 | 8.431 | 16.950 | 25.922 | 0.0 | 8.893 | 16.532 | 24.890 | 0.0 | 7.809 | 16.120 | 24.007 | 0.0 |
| \( H^C(A, 2) \) | 100 | 11.135 | 20.680 | 41.655 | 0.0 | 5.040 | 19.936 | 47.245 | 0.0 | 6.675 | 24.237 | 37.794 | 0.0 |
| \( S(A, 1) \) | 100 | 5.925 | 18.597 | 40.516 | 0.0 | 6.030 | 19.366 | 34.556 | 0.0 | 7.234 | 20.731 | 38.669 | 0.0 |
| \( S(A, 2) \) | 100 | 7.641 | 24.375 | 47.608 | 0.0 | 12.776 | 25.242 | 42.090 | 0.0 | 7.696 | 21.337 | 36.038 | 0.0 |
| \( EX^C(A, 1) \) | 100 | 5.664 | 23.881 | 43.041 | 0.0 | 3.318 | 23.961 | 42.520 | 0.0 | 5.396 | 21.593 | 35.597 | 0.0 |
| \( EX^C(A, 2) \) | 100 | 8.603 | 30.123 | 70.133 | 0.0 | 10.684 | 30.213 | 70.413 | 0.0 | 4.373 | 24.427 | 37.794 | 0.0 |
| \( c^C(A, 1) \) | 100 | 4.418 | 17.835 | 31.634 | 0.0 | 3.749 | 16.673 | 28.829 | 0.0 | 6.665 | 17.211 | 32.765 | 0.0 |
| \( c^C(A, 2) \) | 100 | 4.697 | 23.892 | 44.926 | 0.0 | 6.174 | 24.672 | 55.305 | 0.0 | 12.969 | 22.897 | 32.947 | 0.0 |
| \( EX^C(A, 1) \) | 100 | 5.009 | 18.761 | 28.883 | 0.0 | 6.074 | 20.386 | 36.262 | 0.0 | 2.818 | 21.520 | 39.388 | 0.0 |
| \( EX^C(A, 2) \) | 100 | 7.871 | 20.713 | 36.012 | 0.0 | 8.820 | 22.217 | 42.827 | 0.0 | 9.257 | 21.988 | 34.954 | 0.0 |
| \( c^C(A, 1) \) | 100 | 11.628 | 16.311 | 24.318 | 0.0 | 4.562 | 14.248 | 20.734 | 0.0 | 6.833 | 13.948 | 19.332 | 0.0 |
| \( c^C(A, 2) \) | 100 | 8.986 | 22.806 | 48.982 | 0.0 | 5.948 | 22.710 | 60.662 | 0.0 | 6.309 | 23.554 | 47.268 | 0.0 |
| \( L^C(A, 1) \) | 100 | 5.072 | 14.903 | 23.179 | 0.0 | 10.213 | 15.065 | 21.435 | 0.0 | 8.862 | 14.706 | 21.961 | 0.0 |
| \( L^C(A, 2) \) | 100 | 10.918 | 25.734 | 55.465 | 0.0 | 8.558 | 21.813 | 37.054 | 0.0 | 6.290 | 24.206 | 46.511 | 0.0 |

Table 5: Minimum (Min), average (Gap) and maximum (Max) optimality gaps and percentage of instances solved (Sol) across the A, B and C instances with standard deviation of the demand for different percentages C of customer demands strongly correlated (0.8) and subproblems represented REP. Shaded cells correspond to average optimality gaps lower than for BD. |
<table>
<thead>
<tr>
<th>RS</th>
<th>REp</th>
<th>Min [%]</th>
<th>Gap [%]</th>
<th>Max [%]</th>
<th>Sol [%]</th>
<th>Min [%]</th>
<th>Gap [%]</th>
<th>Max [%]</th>
<th>Sol [%]</th>
<th>Min [%]</th>
<th>Gap [%]</th>
<th>Max [%]</th>
<th>Sol [%]</th>
<th>Min [%]</th>
<th>Gap [%]</th>
<th>Max [%]</th>
<th>Sol [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.406</td>
<td>11.067</td>
<td>19.039</td>
<td>0.000</td>
<td>10.818</td>
<td>18.974</td>
<td>8.3</td>
<td>0.681</td>
<td>10.451</td>
<td>17.708</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H^c (A, 1)</td>
<td>20</td>
<td>0.000</td>
<td>9.398</td>
<td>20.987</td>
<td>8.3</td>
<td>0.000</td>
<td>9.565</td>
<td>16.124</td>
<td>8.3</td>
<td>0.000</td>
<td>9.660</td>
<td>17.019</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H^c (A, 2)</td>
<td>20</td>
<td>0.000</td>
<td>9.318</td>
<td>20.879</td>
<td>8.3</td>
<td>0.000</td>
<td>9.280</td>
<td>18.343</td>
<td>8.3</td>
<td>0.000</td>
<td>9.278</td>
<td>19.188</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L^c (A, 1)</td>
<td>20</td>
<td>0.000</td>
<td>9.030</td>
<td>17.522</td>
<td>8.3</td>
<td>0.000</td>
<td>9.193</td>
<td>19.422</td>
<td>8.3</td>
<td>0.000</td>
<td>9.179</td>
<td>19.871</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L^c (A, 2)</td>
<td>20</td>
<td>0.000</td>
<td>9.494</td>
<td>17.922</td>
<td>8.3</td>
<td>0.000</td>
<td>10.034</td>
<td>12.771</td>
<td>8.3</td>
<td>0.000</td>
<td>10.061</td>
<td>21.279</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H^c (C, 1)</td>
<td>20</td>
<td>0.000</td>
<td>9.233</td>
<td>17.289</td>
<td>8.3</td>
<td>0.000</td>
<td>8.616</td>
<td>15.858</td>
<td>8.3</td>
<td>0.000</td>
<td>10.067</td>
<td>21.892</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H^c (C, 2)</td>
<td>20</td>
<td>0.000</td>
<td>8.194</td>
<td>13.348</td>
<td>8.3</td>
<td>0.000</td>
<td>9.306</td>
<td>17.931</td>
<td>8.3</td>
<td>0.000</td>
<td>8.739</td>
<td>15.535</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C^c (A, 1)</td>
<td>20</td>
<td>0.000</td>
<td>10.636</td>
<td>22.932</td>
<td>8.3</td>
<td>0.000</td>
<td>10.290</td>
<td>18.857</td>
<td>8.3</td>
<td>1.082</td>
<td>10.068</td>
<td>18.460</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C^c (A, 2)</td>
<td>20</td>
<td>0.000</td>
<td>8.356</td>
<td>17.333</td>
<td>8.3</td>
<td>0.000</td>
<td>10.066</td>
<td>20.340</td>
<td>8.3</td>
<td>0.000</td>
<td>10.060</td>
<td>21.279</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C^c (C, 1)</td>
<td>20</td>
<td>0.000</td>
<td>9.834</td>
<td>14.218</td>
<td>8.3</td>
<td>0.000</td>
<td>7.774</td>
<td>15.264</td>
<td>8.3</td>
<td>0.000</td>
<td>7.774</td>
<td>15.264</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C^c (C, 2)</td>
<td>20</td>
<td>0.000</td>
<td>9.834</td>
<td>14.218</td>
<td>8.3</td>
<td>0.000</td>
<td>7.774</td>
<td>15.264</td>
<td>8.3</td>
<td>0.000</td>
<td>7.774</td>
<td>15.264</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L^c (L, 1)</td>
<td>20</td>
<td>0.000</td>
<td>9.994</td>
<td>17.607</td>
<td>8.3</td>
<td>0.000</td>
<td>10.192</td>
<td>18.866</td>
<td>8.3</td>
<td>3.334</td>
<td>11.174</td>
<td>19.278</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L^c (L, 2)</td>
<td>20</td>
<td>0.000</td>
<td>9.919</td>
<td>20.187</td>
<td>8.3</td>
<td>1.611</td>
<td>9.327</td>
<td>14.419</td>
<td>8.3</td>
<td>0.000</td>
<td>9.282</td>
<td>15.985</td>
<td>8.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 6: Minimum (Min), average (Gap) and maximum (Max) optimality gaps and percentage of instances solved (Sol) across the A, B and C instances with |S| = 30 and uncorrelated customer demands, for different values of standard deviation |SD| of the demand and percentages of subproblems represented REp. Shaded cells correspond to average optimality gaps lower than for BD.**
Table 8: Minimum (Min), average (Gap) and maximum (Max) optimality gaps and percentage of instances solved (Sol) across the A, B and C instances with $|S| = 90$ and uncorrelated customer demands, for different values of standard deviation $SD$ of the demand and percentages of subproblems represented REP. Shaded cells correspond to average optimality gaps lower than for BD.

<table>
<thead>
<tr>
<th>SD = 5%</th>
<th>SD = 10%</th>
<th>SD = 20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS</td>
<td>Min [%]</td>
<td>Gap [%]</td>
</tr>
<tr>
<td>BD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.5 Other Observations

In this section we provide some additional insights drawing from our computational experience. We begin by examining the progression of the lower bound and of the incumbent for both BD and GPBD (with the RSs used in Section 5.4). For the sake of brevity, the progression is exemplarily illustrated for the instances with highest number of customer demands positively correlated \((C = 80\%)\), Figure 1, and highest standard deviation of the demand \((SD = 20\%)\), Figure 2. In both cases the instances with the highest number of scenarios \((|S| = 90)\) are considered. Both Figure 1 and Figure 2 show that, for all instances, the curves corresponding to BD almost “envelope” the curves corresponding to the GPBD RSs. This pattern illustrates that, during the entire solution process, the incumbent provided by GPBD is typically smaller (better in our case) than that provided by BD, and that the lower bound provided by GPBD is typically higher (tighter in our case) than that provided by BD, and that the lower bound provided by GPBD is considered. Both Figure 1 and Figure 2 show that, for all instances, the curves corresponding to BD almost “envelope” the curves corresponding to the GPBD RSs. This pattern illustrates that, during the entire solution process, the incumbent provided by GPBD is typically smaller (better in our case) than that provided by BD, and that the lower bound provided by GPBD is typically higher (tighter in our case) that that provided by BD. In turn, this shows that GPBD is
Figure 1: Progression of lower bound and incumbent in BD and GPBD (with all selected RSs, \( n = 1 \) and REP = 20\%) for all instance with \(|S| = 90 \) and \( C = 80\% \).

able to provide better candidate solutions and better bounds already in the initial phases of the algorithm. Notice particularly the steep descent of the incumbent during the first 500 seconds and the remarkably higher bound already from the beginning of the solution process.

The reasons of the improved bounds and candidate solutions are to be ascribed mostly to the improved formulation of MP. Figure 3 reports the Branch and Cut nodes explored using BD and GPBD for the instances with \(|S| = 90 \) and REP = 20\%. It can be noticed that, in general, GPBD explores fewer nodes than BD and, in some cases, significantly fewer. Thus, the improved incumbent and bound are not due to a wider exploration of the Branch and Cut tree, but rather to a better formulation of MP. The fact that GPBD explores fewer nodes can be justified by a heavier and more time-consuming LP relaxation with consequent tighter bounds leading to slimmer trees. In addition, we observed that both GPBD and BD generated approximately the same number of optimality cuts, confirming that the improved results are mainly due to the fact that a stronger MP formulation can provide better solutions already in the initial phases of the algorithm. Finally, we observed that when using GPBD, in general, the solver (i.e., Cplex 12.6.2 in this case) was able to add more valid inequalities to improve the LP relaxation.

Finally, in addition to the benefits provided by the revisited decomposition strategy illus-
Figure 2: Progression of lower bound and incumbent in BD and GPBD (with all selected RSs, \( n = 1 \) and REP= 20%) for all instance with \(|S| = 90\) and \( SD = 20\%\).
Figure 3: Number of B&C nodes explored by BD and GPBD (with the selected RSs and REP = 20%) for all instances with $|S| = 90$. 
trated in this article, GPBD is amenable to several of the enhancements suggested for the classical BD. The reader can find a survey of the possible enhancements in Rahmaniani et al. [2016]. Among these, we report the stabilization procedure which was recently pointed out by Fischetti et al. [2016b] as the most important ingredient in their BD approach for solving large-scale instances of the uncapacitated facility location problem. In a nutshell, stabilization implies generating cuts on a “stabilized” versions of the solution proposed by MP. The stabilized point is obtained by choosing a point in between the solution to MP and a stabilizer decided a priori, in this case $(1, \ldots, 1)^T$. The scope is that of avoiding the production of erratic solutions especially in the initial iterations. We tested the stabilization technique described by [Fischetti et al., 2016b] and observed that, in general, it grants a further noticeable reduction of the optimality gap (for some instances the optimality gap was remarkably reduced by approximately 80%). This offers significant opportunities to further improve the results in Section 5.3 which are obtained without additional efficiency measures in order to isolate the effect of a mere alternative decomposition strategy.

6 Conclusions

The proposed Generalized Partial Benders Decomposition of two-stage stochastic programs revisits the classical Benders decomposition by granting the possibility to generate more “informed” master problems. Selected portions of second-stage subproblems (variables and constraints) are retained in the master problem with the scope of providing additional information for generating better solutions in the early phases of the algorithm. The method provides enough flexibility for adapting the amount of second-stage information in MP on the specific case. A number of simple and general purpose strategies have been proposed for selecting second-stage variables and constraints to retain in the master problem.

The method has been applied to computationally difficult large-scale instances of the Capacitated Facility Location Problem with Stochastic Demand. An extensive computational study shows that GPBD improves BD on almost all instances tested and that the improvements are stable across a set of heterogeneous instances. The optimality gap reduction is particularly high on the largest instances (with 90 scenarios). In some cases the optimality gap is nearly halved. In addition, it emerges that GPBD performs better when the number of variables and constraints retained is small, meaning for example that they are selected from only a small number of scenario-subproblems (20% of the subproblems in our case). Conversely, the performances of GPBD worsen dramatically when the master problem becomes too heavy. The analysis of the solution process shows that GPBD provides high quality bounds and candidate solutions already in the early stages of the algorithm. Particularly, the initial bounds are significantly higher than the bounds provided by BD.

Our study also illustrates that efficient retaining strategies can be discovered simply by testing a number of candidate retaining strategies on small-scale instances. In fact, the corresponding retaining strategies remain effective also on larger and numerically different instances. Remarkable results on the largest instances were provided by retaining in the master-problem
only one, randomly selected, second-stage constraint (for each of the represented subproblems). This illustrates that even simple retaining strategies may grant satisfactory results and that retaining a small amount of second-stage information in the master problem may be extremely beneficial.

A Summary of the retaining strategies

Table 9: Retaining Strategies

<table>
<thead>
<tr>
<th>RS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C^W(i, n) )</td>
<td>Represents the ( i )-th block of constraints by partitioning the constraints in block ( i ) in ( n ) clusters based on the rows of ( W_s(i) ) and selecting, for each cluster, the constraint closest to its centroid.</td>
</tr>
<tr>
<td>( H^W(i, n) )</td>
<td>Represents the ( i )-th block of constraints by the ( n ) constraints of block ( i ) with highest values of ( W_s(i) ).</td>
</tr>
<tr>
<td>( L^W(i, n) )</td>
<td>Represents the ( i )-th block of constraints by the ( n ) constraints of block ( i ) with lowest values of ( W_s(i) ).</td>
</tr>
<tr>
<td>( EX^W(i, n) )</td>
<td>Applies simultaneously RSs ( LT^T(i, n) ) and ( HT^T(i, n) ).</td>
</tr>
<tr>
<td>( CT^T(i, n) )</td>
<td>Represents the ( i )-th block of constraints by partitioning the constraints in block ( i ) in ( n ) clusters based on the rows of ( T_s(i) ) and selecting, for each cluster, the constraint closest to its centroid.</td>
</tr>
<tr>
<td>( HT^T(i, n) )</td>
<td>Represents the ( i )-th block of constraints by the ( n ) constraints of block ( i ) with highest values of ( T_s(i) ).</td>
</tr>
<tr>
<td>( LT^T(i, n) )</td>
<td>Represents the ( i )-th block of constraints by the ( n ) constraints of block ( i ) with lowest values of ( T_s(i) ).</td>
</tr>
<tr>
<td>( EX^T(i, n) )</td>
<td>Applies simultaneously RSs ( LT^T(i, n) ) and ( HT^T(i, n) ).</td>
</tr>
<tr>
<td>( C^H(i, n) )</td>
<td>Represents the ( i )-th block of constraints by partitioning the constraints in block ( i ) in ( n ) clusters based on the coefficients ( h_s(i) ) and selecting, for each cluster, the constraint closest to its centroid.</td>
</tr>
<tr>
<td>( H^H(i, n) )</td>
<td>Represents the ( i )-th block of constraints by the ( n ) constraints of block ( i ) with highest values of ( h_s(i) ).</td>
</tr>
<tr>
<td>( L^H(i, n) )</td>
<td>Represents the ( i )-th block of constraints by the ( n ) constraints of block ( i ) with lowest values of ( h_s(i) ).</td>
</tr>
<tr>
<td>( EX^H(i, n) )</td>
<td>Applies simultaneously RSs ( LH^H(i, n) ) and ( HH^H(i, n) ).</td>
</tr>
<tr>
<td>( S(i, n) )</td>
<td>Represents the ( i )-th block of constraints by sampling ( n ) constraints from the subset ( M_i ).</td>
</tr>
</tbody>
</table>

Capacitated Facility Location Problem with Stochastic Demand

| RS \( B, n) \) | Represents constraints block \( B \) i.e., (7e), by partitioning facilities in \( n \) clusters based on their opening costs, and selecting, for each cluster, the constraint in \( B \) corresponding to the location closest to the centroid. |
| \( H^F(B, n) \) | Represents constraints block \( B \) i.e., (7e), by means of the \( n \) constraint corresponding to the \( n \) facilities with the highest opening cost. |
| \( L^F(B, n) \) | Represents constraints block \( B \) i.e., (7e), by means of the \( n \) constraint corresponding to the \( n \) facilities with the lowest opening cost. |
| \( EX^F(B, n) \) | Applies simultaneously RSs \( LF^F(B, n) \) and \( HF^F(B, n) \). |
| \( C^C(A, n) \) | Represents constraints block \( A \) i.e., (7c), by partitioning customers in \( n \) clusters based on their allocation costs, and selecting, for each cluster, the constraint in \( A \) corresponding to the customer closest to the centroid. |
| \( H^C(A, n) \) | Represents constraints block \( A \) i.e., (7c), by means of the \( n \) constraint corresponding to the \( n \) customers with the highest allocation cost. |
| \( L^C(A, n) \) | Represents constraints block \( A \) i.e., (7c), by means of the \( n \) constraint corresponding to the \( n \) customers with the lowest allocation cost. |
| \( EX^C(A, n) \) | Applies simultaneously RSs \( LC^C(A, n) \) and \( HC^C(A, n) \). |
| \( EX^h,F(n) \) | Applies simultaneously RSs \( EX^h(A, n) \) and \( EX^F(B, n) \). |
| \( EX^h,T(n) \) | Applies simultaneously RSs \( EX^h(A, n) \) and \( EX^T(B, n) \). |
| \( EX^T(n) \) | Applies simultaneously RSs \( EX^C(A, n) \) and \( EX^T(B, n) \). |

References


