



HAL
open science

The Benders by batch algorithm: design and stabilization of an enhanced algorithm to solve multicut Benders reformulation of two-stage stochastic programs

Xavier Blanchot, François Clautiaux, Boris Detienne, Aurélien Froger, Manuel Ruiz

► **To cite this version:**

Xavier Blanchot, François Clautiaux, Boris Detienne, Aurélien Froger, Manuel Ruiz. The Benders by batch algorithm: design and stabilization of an enhanced algorithm to solve multicut Benders reformulation of two-stage stochastic programs. 2022. hal-03286135v2

HAL Id: hal-03286135

<https://hal.science/hal-03286135v2>

Preprint submitted on 6 Jan 2022 (v2), last revised 15 Dec 2022 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The Benders by batch algorithm: design and stabilization of an enhanced algorithm to solve multicut Benders reformulation of two-stage stochastic programs

Xavier Blanchot^{1,2} François Clautiaux¹ Boris Detienne¹ Aurélien Froger¹
Manuel Ruiz²

January 6, 2022

¹ Université de Bordeaux, UMR CNRS 5251, Inria Bordeaux Sud-Ouest, Talence, France

² RTE, Paris La Défense, France

Abstract

This paper introduces a new exact algorithm to solve two-stage stochastic linear programs. Based on the multicut Benders reformulation of such problems, with one subproblem for each scenario, this method relies on a partition of the subproblems into batches. By detecting as soon as possible the non-optimality of a first-stage candidate, it solves only a small proportion of the subproblems at most iterations. We also propose a general framework to stabilize our algorithm, and show its finite convergence and exact behavior. We report an extensive computational study on large-scale instances of stochastic optimization literature that shows the efficiency of the proposed algorithm compared to six alternative algorithms from the literature. We also obtain significant additional computational time savings using the primal stabilization schemes.

Keywords— Large-scale optimization, Benders Decomposition, Stochastic programming, Cut aggregation

1 Introduction

Large-scale two-stage stochastic linear programs arise in many applications such as network design, telecommunications network planning, air freight scheduling, power generation planning. In such problems, first-stage decisions (also called here-and-know decisions) are to be made before knowing the value taken by random parameters, then second-stage decisions (also called wait and see decisions) are made after observing the value taken by each random parameter. In practice, many approaches introduced to solve those problems are based on decomposition techniques (Ruszczynski, 1997).

In this paper, we study two-stage stochastic linear programs. We assume that the probability distribution is given by a finite set of scenarios and focus on problems with a large number of scenarios.

Email addresses: xavier.blanchot@rte-france.com, xavier.blanchot@u-bordeaux.fr (Xavier Blanchot), francois.clautiaux@math.u-bordeaux.fr (François Clautiaux), boris.detienne@math.u-bordeaux.fr (Boris Detienne), aurelien.froger@u-bordeaux.fr (Aurélien Froger), manuel.ruiz@rte-france.com (Manuel Ruiz)

We consider the following linear program with a scenario block structure:

$$\begin{cases} \min c^\top x + \sum_{s \in S} p_s g_s^\top y_s \\ s.t. : W_s y_s = d_s - T_s x, \forall s \in S \\ y_s \in \mathbb{R}_+^{n_2}, \forall s \in S \\ x \in X \end{cases} \quad (1)$$

where $x \in \mathbb{R}^{n_1}$, $c \in \mathbb{R}^{n_1}$, S is a finite set of scenarios, $p_s \in \mathbb{R}^+$ is a positive weight associated with a scenario $s \in S$ (e.g., a probability), $g_s \in \mathbb{R}^{n_2}$, $W_s \in \mathbb{R}^{m \times n_2}$, $T_s \in \mathbb{R}^{m \times n_1}$, $d_s \in \mathbb{R}^m$, $X \subset \mathbb{R}^{n_1}$ is a polyhedral set. Variables x are called *first-stage variables* and variables y_s are called *second-stage variables* or *recourse variables*. Problem (1) is called the *extensive formulation* of a two-stage stochastic problem.

When the number of scenarios is large, problem (1) becomes intractable for MIP solvers. Its reformulation as

$$\begin{cases} \min c^\top x + \sum_{s \in S} p_s \phi(x, s) \\ s.t. x \in X \end{cases} \quad (2)$$

where for every $s \in S$ and every $x \in X$,

$$\phi(x, s) = \begin{cases} \min_y g_s^\top y \\ s.t. W_s y = d_s - T_s x \\ y \in \mathbb{R}_+^{n_2} \end{cases} \quad (3)$$

makes the use of decomposition methods attractive. If we fix the first-stage variables to $\hat{x} \in X$, then the resulting problem becomes separable according to the scenarios. We denote by $(SP(\hat{x}, s))$ the subproblem associated with a scenario $s \in S$ and by $\phi(\hat{x}, s)$ its value.

Let $\Pi_s = \{\pi \in \mathbb{R}^m \mid W_s^\top \pi \leq g_s\}$ be the polyhedron associated with the dual of $(SP(\hat{x}, s))$, which does not depend on first-stage variables x . We denote by $\text{Rays}(\Pi_s)$ the set of extreme rays of Π_s , and by $\text{Vert}(\Pi_s)$ the set of extreme points of Π_s . By Farkas' Lemma, we can write an expression of the domain of $\phi(\cdot, s)$ as $\text{dom}(\phi(\cdot, s)) = \{x \in \mathbb{R}^{n_1} \mid r_s^\top (d_s - T_s x) \leq 0, \forall r_s \in \text{Rays}(\Pi_s)\}$. Then we can replace in formulation (2) the polyhedral application $x \mapsto \phi(x, s)$ by its outer linearization on its domain. Using an epigraph variable θ_s for every $s \in S$, we obtain the multicut Benders reformulation (Birge and Louveaux, 1988) of problem (1):

$$\begin{cases} \min_{x, \theta} c^\top x + \sum_{s \in S} p_s \theta_s \\ s.t. : \theta_s \geq \pi_s^\top (d_s - T_s x), \forall s \in S, \forall \pi_s \in \text{Vert}(\Pi_s) \quad (i) \\ 0 \geq r_s^\top (d_s - T_s x), \forall s \in S, \forall r_s \in \text{Rays}(\Pi_s) \quad (ii) \\ x \in X, \theta \in \mathbb{R}^{\text{Card}(S)} \end{cases} \quad (4)$$

Constraints (i) are called optimality cuts, and constraints (ii), feasibility cuts.

The classical version of the multicut Benders decomposition algorithm (see Algorithm 1) consists in the relaxation of constraints (i) and (ii) and an iterative scheme to add them until convergence is proven. As the number of extreme rays and vertices of polyhedra Π_s is finite, for every $s \in S$, the

total number of optimality and feasibility cuts is finite. Then, this algorithm converges in a finite number of iterations. The relaxation of (4) at iteration k of the algorithm is called the *relaxed master program*, denoted by $(RMP)^{(k)}$ and its solution is denoted by $(\tilde{x}^{(k)}, (\tilde{\theta}_s^{(k)})_{s \in S})$.

Algorithm 1: Classical version of the multicut Benders decomposition algorithm

Parameters: $\epsilon > 0$ the selected optimality gap

- 1 **Initialization:** $k \leftarrow 0, UB^{(0)} \leftarrow +\infty, LB^{(0)} \leftarrow -\infty$
- 2 **while** $UB^{(k)} > LB^{(k)} + \epsilon$ **do**
- 3 $k \leftarrow k + 1$
- 4 Solve $(RMP)^{(k)}$ and retrieve $(\tilde{x}^{(k)}, (\tilde{\theta}_s^{(k)})_{s \in S})$
- 5 $LB^{(k)} \leftarrow c^\top \tilde{x}^{(k)} + \sum_{s \in S} p_s \tilde{\theta}_s^{(k)}$
- 6 **for** $s \in S$ **do**
- 7 Solve $(SP(\tilde{x}^{(k)}, s))$ and retrieve $\pi_s \in \text{Vert}(\Pi_s)$ or $\pi_s \in \text{Rays}(\Pi_s)$
- 8 **if** $(SP(\tilde{x}^{(k)}, s))$ *is feasible* **then**
- 9 Add $\theta_s \geq \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
- 10 **else**
- 11 Add $0 \geq \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
- 12 **if** $(SP(\tilde{x}^{(k)}, s))$ *is feasible* $\forall s \in S$ **then**
- 13 $UB^{(k)} \leftarrow \min\left(UB^{(k-1)}, c^\top \tilde{x}^{(k)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s \tilde{x}^{(k)})\right)$
- 14 $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$
- 15 **Return** $\tilde{x}^{(k)}$

When the total number of subproblems is large, solving all the subproblems at each iteration, like in Algorithm 1, can be time-consuming. To overcome this issue, we introduce a new exact algorithm to solve problem (1), referred to as the *Benders by batch* algorithm. The term *batch* refers to a given fixed partition of all subproblems into separate batches. We propose a new stopping criterion that allows us to identify a non-optimal solution without necessarily having to solve all the subproblems. As a result, only few subproblems are generally solved at a first-stage candidate solution. To prevent introducing too many cuts in the relaxed master program, the algorithm can use cut aggregation, thus generating a single cut from all subproblems that belong to an identical batch. If the number of batches is equal to one, the Benders by batch algorithm is equivalent to the classical version of the Benders decomposition algorithm (multicut or monocut, depending on the use of cut aggregation). Unlike several existing methods based on similar ideas (Wets, 1983; Dantzig and Infanger, 1991; Higle and Sen, 1991; Oliveira et al., 2011), our algorithm is valid for any stochastic linear problem formulated with scenarios, is exact, has finite convergence, and only requires solving linear programs. We develop a generic framework to stabilize the Benders by batch algorithm and prove the finite convergence and exact behavior of the stabilized algorithm. We propose two primal stabilization schemes for the algorithm based on in-out separation strategies (Ben-Ameur and Neto, 2007). Our method can benefit from advanced cut aggregation schemes and is compatible with classical dual stabilization techniques (Magnanti and Wong, 1981; Papadakos, 2008; Sherali and Lunday, 2013). The general concept of oracles with on-demand accuracy (de Oliveira and Sagastizábal, 2014) might embed the core idea of the Benders by batch algorithm. However, it requires that the oracle gives a subgradient which belongs to an approximate subdifferential of the objective function at each iteration which is not required in the Benders by batch algorithm, and may not be satisfied in the general case.

The contributions of the paper can be summarized as follows:

- We propose a new exact algorithm to solve the Benders reformulation of two-stage linear stochas-

tic programs with finite probability distributions, without any other assumption on the uncertainty. This algorithm is based on a sequential stopping criterion relying on a partition of the subproblems. This stopping criterion allows the algorithm to solve only a few subproblems at most iterations by detecting the non-optimality of a first-stage candidate solution early in the subproblems resolution process.

- We develop a primal stabilization of the Benders by batch algorithm. We state sufficient conditions for the stabilized algorithm to be exact and have a finite convergence and provide two effective primal stabilization schemes based on the in-out separation approach.
- We perform an extensive numerical study showing the effectiveness of the developed algorithm on some classical stochastic instances of the literature compared to classical implementations of the moncut and multicut Benders decomposition algorithm, with and without in-out stabilization, and a level bundle method.

The paper is organized as follows. Section 2 reviews the literature on acceleration techniques for Benders decomposition, with a focus on the stochastic case, and on closely related methods. In section 3, we present the Benders by batch algorithm. Section 4 presents a general framework to stabilize our algorithm and two stabilization schemes: the first one based on the classical in-out separation scheme, and the second one based on exponential moving averages. Section 5 presents extensive computational experiments. Then, section 6 concludes and outlines perspectives.

2 Related work

The classical version of the Benders decomposition algorithm can be slow to converge. Researchers have proposed several techniques to accelerate its convergence. We first present classical primal and dual stabilization methods, which are the most widespread and general methods to accelerate the Benders decomposition algorithm. We then present different methods specific to stochastic programming, with a focus on methods that avoid the systematic resolution to all the subproblems.

A well-known downside of cutting planes methods, and therefore of the Benders decomposition algorithm, is the oscillation of the first-stage variables. Because of the relaxation of all the constraints related to the subproblems, the solutions of the relaxed master programs might be far from the optimal solution to the initial problem. This might lead to a large amount of time spent in evaluating poor quality solutions in the early iterations. To our knowledge, successful methods proposed so far are either inspired by bundle methods (Zverovich et al., 2012; Linderoth and Wright, 2003; Wolf et al., 2014), or by in-out separation approaches (Ben-Ameur and Neto, 2007). Those methods try to restrict the search of an optimal solution close to a given first-stage solution. This solution is called *stability center* in the case of bundle methods, or *in point* in the case of in-out stabilization. On the one hand, many authors proposed quadratic stabilization techniques, such as Ruszczyński (1986), who added a quadratic proximal term in the objective function of the relaxed master program, or Zverovich et al. (2012), Wolf et al. (2014) and van Ackooij et al. (2017), who used quadratic level stabilizations. Linderoth and Wright (2003) used a trust-region bundle method and proposed to use the infinite norm with an effective asynchronous parallelized framework. On the other hand, the in-out separation scheme performs a linear search between the in point and the solution to the relaxed master program, and it can rely on the practical effectiveness of linear programming solvers.

The in-out separation approach has been applied successfully in a cutting plane algorithm to solve a survivable network design problem (Ben-Ameur and Neto, 2007), in column generation (Pessoa et al., 2013), in a branch-and-cut algorithm based on a Benders decomposition approach to solve facility location problems (Fischetti et al., 2016), and in a cutting plane algorithm applied to disjunctive optimization (Fischetti and Salvagnin, 2010).

Another family of acceleration techniques focuses on the quality of the optimality cuts. The polyhedral structure of the second-stage function implies a degeneracy of the dual subproblem. In the singular points of this function, many equivalent extreme dual solutions exist for the subproblem, each one defining a different optimality cut. The choice of a "good" dual solution can improve dramatically the convergence of the algorithm. Magnanti and Wong (1981) proposed to solve the dual of the subproblem twice in order to find the solution which maximizes the objective function at a fixed core point of the master problem. A different choice of the core point leads to a different cut. A cut derived in this framework is called a *Pareto-optimal cut*. Papadakos (2008) proposed a less restrictive way to choose the core point, and a practical framework to update it. Sherali and Lunday (2013) improved the method, bypassing the need to solve the subproblem twice.

In the case of stochastic programming, formulations rely either on an epigraph variable for every subproblem (see formulation (4)) or on a single epigraph variable for all the subproblems, also called *L-shaped method* (Van Slyke and Wets, 1969). The former formulation is referred to as the multicut Benders reformulation, whereas the latter is known as the monocut Benders reformulation. The multicut Benders reformulation was introduced by Birge and Louveaux (1988). You and Grossmann (2013) showed dramatic improvement both on computing time and number of iterations due to the multicut reformulation on two supply chain planning problems. The multicut version provides a tighter approximation of the second-stage function, and converges in less iterations than the monocut one. However the master problem might suffer from the large number of cuts added through the optimization process, and thus might become time consuming to solve. The question of using either the monocut or multicut version of the algorithm is not straightforward. As far as we know, one of the major improvements proposed to improve pure multicut Benders decomposition was to use massive parallelization (Linderoth and Wright, 2003). Trukhanov et al. (2010) proposed a framework to aggregate some optimality cuts with the aim of finding a compromise between the monocut and pure multicut versions of the algorithm. Wolf et al. (2014) proposed to maintain both a multicut and a monocut models. Using the general principle of *oracles with on-demand accuracy* (de Oliveira and Sagastizábal, 2014), when both models have a large difference between their objective values, they add in the monocut model an aggregation of some current active cuts in the multicut one to improve the relaxation without having to solve the subproblems.

One of the major bottlenecks faced to solve two-stage stochastic programs is the large number of subproblems to solve at each iteration to compute Benders cuts. Researchers proposed some methods to avoid the resolution to all the subproblems at each iteration of the Benders decomposition algorithm. In the case of stochastic problems with fixed recourse (i.e, $W_s = W$ for every $s \in S$ in problem (1)) where the second-stage objective function does not depend on the uncertainty (i.e, $g_s = g$ for every $s \in S$ in problem (1)), some authors, such as (Wets, 1983; Higle and Sen, 1991; Dantzig and Infanger, 1991; Infanger, 1992), used the fact that the duals of all the subproblems share the same constraint polyhedron: $\Pi_s = \Pi$, for every $s \in S$. Given an optimal dual solution π_{s_0} to a subproblem $s_0 \in S$, *bunching* (Wets, 1983) consists in checking the primal feasibility of this solution for the other subproblems. This solution is optimal for all the subproblems for which this solution is

primal feasible, and there is no need to solve them. Dantzig and Infanger (1991) and Infanger (1992) proposed to use *importance sampling* to compute a good approximation of the expected cut in the monocut formulation with only a few scenarios. Although the resulting algorithm is not exact, they report results with small confidence intervals on the objective value. Hige and Sen (1991) introduced *stochastic decomposition*. The method only solves a few subproblems at each iteration and computes cuts with all the dual solutions obtained at previous iterations. Finally, Oliveira et al. (2011) proposed an algorithm which does not restrict to the hypothesis $g_s = g$, $\forall s \in S$. It adapts the dual solutions of a subset of subproblems to generate inexact cuts to the remaining subproblems. The methods of Oliveira et al. (2011), Dantzig and Infanger (1991) and Hige and Sen (1991) are designed for a monocut algorithm, but the method of Oliveira et al. (2011) could be adapted to a multicut algorithm.

Finally, among other techniques used to accelerate the resolution to two-stage stochastic programs, Crainic et al. (2020) proposed the so-called *Partial Benders decomposition*. Under the hypothesis $g_s = g$, $\forall s \in S$ and fixed recourse, they add one of the scenarios, or an artificial scenario computed as the expectation of the others, to the master problem. They showed major improvements on some instances, both in computing time and number of iterations, even if the master problem becomes way larger than the original one, and might be harder to solve at each iteration. Under the same assumptions ($g_s = g$, $W_s = W$, $\forall s \in S$), Song and Luedtke (2015) proposed an adaptative partition-based approach, which does not rely on Benders reformulation. Given a partition of the subproblems, they compute a relaxation of the initial deterministic reformulation by summing the matrices and right-hand-sides of the subproblems of each element of the partition. They showed the existence of a partition with the same optimal value as the initial problem and an iterative algorithm to find it. van Ackooij et al. (2017) proposed to use level stabilization with the adaptative partition-based approach and showed numerical experiments where the resulting algorithms largely outperform classical level bundle or Benders decomposition methods. Table 1 classifies the different methods discussed in this section.

Paper	Randomness hypothesis	Solve all SPs	Monocut or multicut	Exact method	Finite convergence	Stabilization
(Crainic et al., 2020)	$g_s = g, W_s = W \forall s \in S$	Yes	Both	Yes	Yes	No
(Song and Luedtke, 2015)	$g_s = g, W_s = W \forall s \in S$	Yes	Not applicable	Yes	Yes	No
(van Ackooij et al., 2017)	$g_s = g, W_s = W \forall s \in S$	No	Both	Yes	Yes	Level
(Wets, 1983)	$g_s = g, W_s = W \forall s \in S$	No	Both	Yes	Yes	No
(Dantzig and Infanger, 1991)	$g_s = g, W_s = W \forall s \in S$	No	Monocut	No	Yes	No
(Hige and Sen, 1991)	$g_s = g, W_s = W \forall s \in S$	No	Monocut	Yes	No	No
(Trukhanov et al., 2010)	No	Yes	Multicut	Yes	Yes	No
(Linderth and Wright, 2003)	No	Yes	Multicut	Yes	Yes	Trust-region
(Wolf et al., 2014)	No	All or none	Monocut and Multicut	Yes	Yes	Level
(Oliveira et al., 2011)	No	No	Monocut	Inexact	Yes	Proximal bundle
This work	No	No	Multicut	Yes	Yes	In-out

Table 1: Comparison of stochastic methods to accelerate Benders decomposition. (SPs: subproblems)

3 The Benders by batch algorithm

We propose a new algorithm, hereafter referred to as the Benders by batch algorithm, to solve exactly the multicut Benders reformulation (4) of a two-stage stochastic linear program. The algorithm consists in solving the subproblems by batch and stopping solving subproblems at an iteration as soon as we identify that the current first-stage solution is non-optimal.

Without loss of generality, we assume that the problem has *relatively complete recourse* (i.e., $X \subset \text{dom}(\phi(\cdot, s))$ for every scenario $s \in S$), meaning that every subproblem is feasible for every

$x \in X$. As a result, only optimality cuts are required in the Benders decomposition algorithm, and every $x \in X$ defines an upper bound on the optimal value of the problem. Every two-stage linear stochastic program can be reformulated in a problem satisfying this hypothesis by introducing slack variables with large enough coefficients in the objective function.

We first present some notations necessary to formally describe the algorithm. We consider an ordered set of scenarios $S = \{s_1, s_2, \dots, s_{Card(S)}\}$ and a given batch size $1 \leq \eta \leq Card(S)$. We define $\kappa = \lceil Card(S)/\eta \rceil$ as the number of batches of subproblems. For every $i \in \llbracket 1, \kappa \rrbracket$, the i^{th} batch of subproblems S_i is defined as $S_i = \{s_{(i-1)\eta+1}, \dots, s_{(i-1)\eta+\eta_i}\}$, where η_i is the size of batch i , $\eta_1 = \dots = \eta_{\kappa-1} = \eta$ and $\eta_\kappa = (Card(S) \bmod \eta)$. Family $(S_i)_{i \in \llbracket 1, \kappa \rrbracket}$ defines a partition of S . We restrict ourselves to batches of the same size, but the method remains valid for any partition of S . We denote by $\check{x}^{(k)}$ the optimal first-stage solution to $(RMP)^{(k)}$ at iteration k of the algorithm, and by $\check{\theta}_s^{(k)}$ the optimal value of the epigraph variable associated with a scenario $s \in S$. A lower bound on the optimal value of problem (1) is then computed as $LB^{(k)} = c^\top \check{x}^{(k)} + \sum_{s \in S} p_s \check{\theta}_s^{(k)}$. For a first-stage solution $x \in X$, we denote by $UB(x) = c^\top x + \sum_{s \in S} p_s \phi(x, s)$ an upper bound on the optimal value of problem (1).

Let $\epsilon > 0$ be the optimality gap of the algorithm. The classical stopping criterion $UB - LB \leq \epsilon$ of the Benders decomposition algorithm cannot be directly applied if all the subproblems are not solved. Specifically, an upper bound on the optimal value of the problem is only known after computing, for a first-stage solution $x \in X$, the optimal value $\phi(x, s)$ of every subproblem $(SP(x, s))$. We propose a new stopping criterion, which detects that a current first-stage solution is non-optimal without necessarily having to solve all the subproblems. This criterion is based on the concept of ϵ_i -approximation that we define below.

Definition 1 (ϵ_i -approximation). *Let $\epsilon > 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration and σ a permutation of $\llbracket 1, \kappa \rrbracket$. For every $i \in \llbracket 1, \kappa \rrbracket$, we say that batch $S_{\sigma(i)}$ is ϵ_i -approximated by $(RMP)^{(k)}$ if*

$$\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon_i \quad (5)$$

with $\epsilon_i = \epsilon - \sum_{t=1}^{i-1} \sum_{s \in S_{\sigma(t)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right)$.

We refer to ϵ_i as the remaining gap of batch $S_{\sigma(i)}$ according to the permutation σ and the optimality gap ϵ . For every index $i \in \llbracket 2, \kappa \rrbracket$, we have $\epsilon_i = \epsilon_{i-1} - \sum_{s \in S_{\sigma(i-1)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right)$, which means that computing the successive remaining gaps consists in filling the gap ϵ with the differences between the true values of the subproblems and their epigraph approximations in $(RMP)^{(k)}$.

The following proposition shows that ϵ_i -approximation can be used to derive a stopping criterion for the Benders by batch algorithm.

Proposition 1. *Let $\epsilon > 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration of the algorithm, and σ a permutation of $\llbracket 1, \kappa \rrbracket$. The first-stage solution $\check{x}^{(k)}$ is an optimal solution to problem (1) if and only if batch $S_{\sigma(i)}$ is ϵ_i -approximated by $(RMP)^{(k)}$ for every index $i \in \llbracket 1, \kappa \rrbracket$.*

Proof of proposition 1. (\Rightarrow) Assume that $\check{x}^{(k)}$ is an optimal solution to problem (1). We have:

$$UB(\check{x}^{(k)}) - LB^{(k)} \leq \epsilon$$

$$\begin{aligned}
&\iff c^\top \tilde{x}^{(k)} + \sum_{s \in S} p_s \phi(\tilde{x}^{(k)}, s) - \left(c^\top \tilde{x}^{(k)} + \sum_{s \in S} p_s \check{\theta}_s^{(k)} \right) \leq \epsilon \\
&\iff \sum_{s \in S} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon
\end{aligned}$$

As family $(S_{\sigma(1)}, S_{\sigma(2)}, \dots, S_{\sigma(\kappa)})$ defines a partition of S , previous equation gives:

$$\begin{aligned}
&\sum_{t=1}^{\kappa} \sum_{s \in S_{\sigma(t)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon \\
&\iff \sum_{t=i}^{\kappa} \sum_{s \in S_{\sigma(t)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon_i, \quad \forall i \in \{1, \dots, \kappa\}
\end{aligned}$$

As $p_s \geq 0$, $\forall s \in S$, and as $(RMP)^{(k)}$ is a relaxation of problem 1, by independence of the batches, we have: $\sum_{s \in S_{\sigma(t)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \geq 0$, $\forall t \in \{1, \dots, \kappa\}$. We therefore have:

$$\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon_i, \quad \forall i \in \{1, \dots, \kappa\}$$

which is the definition of batch $S_{\sigma(i)}$ being ϵ_i -approximated by $(RMP)^{(k)}$.

(\Leftarrow) Assume that for every index $i \in \llbracket 1, \kappa \rrbracket$, we have $\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon_i$ and therefore:

$$\sum_{s \in S_{\sigma(\kappa)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon_\kappa \tag{6}$$

By definition of ϵ_κ we have:

$$\begin{aligned}
\epsilon_\kappa &= \epsilon - \sum_{i=1}^{\kappa-1} \left[\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \right] \\
&\iff \epsilon_\kappa + \sum_{i=1}^{\kappa-1} \left[\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \right] = \epsilon
\end{aligned}$$

Then, using equation (6), we have:

$$\begin{aligned}
&\sum_{i=1}^{\kappa} \left[\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \right] \leq \epsilon \\
&\iff UB(\tilde{x}^{(k)}) - LB^{(k)} \leq \epsilon
\end{aligned}$$

which implies that $\tilde{x}^{(k)}$ is an optimal solution to problem (1). \square

Corollary 1. *Let ϵ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration, and σ a permutation of $\llbracket 1, \kappa \rrbracket$. If there exists an index $i \in \llbracket 1, \kappa \rrbracket$ such that $\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) > \epsilon_i$, then $\tilde{x}^{(k)}$ is not an optimal solution to problem (1).*

Remark 1. *The previous rule provides a criterion which ensures a solution with an absolute gap of ϵ . In order to compute solutions with a relative gap tolerance of δ , we set $\epsilon = \max\{10^{-10}, LB\delta\}$ with LB the current value of the relaxed master program.*

Algorithm 2: The Benders by batch algorithm

Parameters: $\epsilon > 0$, $\eta \in \llbracket 1, \text{Card}(S) \rrbracket$ the batch size, **aggregation** $\in \{\text{True}, \text{False}\}$

1 **Initialization:** $i \leftarrow 1$, $k \leftarrow 0$, **stay_at_x** \leftarrow **True**

2 Define a partition $(S_i)_{i \in \llbracket 1, \kappa \rrbracket}$ of the subproblems according to batch size η

3 **while** $i < \kappa + 1$ **do**

4 $k \leftarrow k + 1$

5 Solve $(RMP)^{(k)}$ and retrieve $\tilde{x}^{(k)}$, $(\check{\theta}_s^{(k)})_{s \in S}$

6 $i \leftarrow 1$, $\epsilon_1 \leftarrow \epsilon$, **stay_at_x** \leftarrow **True**

7 Choose a permutation σ of $\llbracket 1, \kappa \rrbracket$

8 **while** **stay_at_x** = **True** and $i < \kappa + 1$ **do**

9 **for** $s \in S_{\sigma(i)}$ **do**

10 Solve $(SP(\tilde{x}^{(k)}, s))$ and retrieve $\phi(\tilde{x}^{(k)}, s)$ and $\pi_s \in \text{Vert}(\Pi_s)$

11 **if** **aggregation** **then**

12 Add $\sum_{s \in S_{\sigma(i)}} p_s \theta_s \geq \sum_{s \in S_{\sigma(i)}} p_s (\pi_s^\top (d_s - T_s x))$ to $(RMP)^{(k)}$

13 **else**

14 **for** $s \in S_{\sigma(i)}$ **do**

15 Add $\theta_s \geq \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$

16 **if** $\sum_{s \in S_{\sigma(i)}} p_s (\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)}) \leq \epsilon_i$ **then**

17 $\epsilon_{i+1} \leftarrow \epsilon_i - \sum_{s \in S_{\sigma(i)}} p_s (\phi(\tilde{x}^{(k)}, s) - \check{\theta}_s^{(k)})$

18 $i \leftarrow i + 1$

19 **else** **stay_at_x** \leftarrow **False**

20 $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$

21 **Return** $\tilde{x}^{(k)}$

We now present the Benders by batch algorithm (Algorithm 2). The while loop from lines 3 to 20 will be referred hereafter as the *master loop*. Each pass of this loop corresponds to an iteration of the algorithm. At iteration k , the relaxed master program $(RMP)^{(k)}$ is solved to obtain a new first-stage solution $\tilde{x}^{(k)}$. A permutation σ of $\llbracket 1, \kappa \rrbracket$ is then chosen. This permutation defines the order in which the batches of subproblems $(S_1, S_2, \dots, S_\kappa)$ will be solved at the current first-stage solution. The while loop from lines 8 to 19 will be referred as the *optimality loop*. In each pass in this loop:

- the subproblems of the current batch $S_{\sigma(i)}$ are solved (lines 9 to 10). This part of the algorithm can be parallelized, as in the classical Benders decomposition algorithm to accelerate the procedure.
- the cuts defined by the solutions of the subproblems are added to the relaxed master program (lines 11 to 15). We add a parameter **aggregation** to the algorithm. If this parameter is set to **False**, the cuts of each subproblem are added independently to the relaxed master program, as it is the case in the classical multicut Benders decomposition algorithm. If this parameter is set to **True**, we add only one cut, computed as the weighted sum of all the cuts of the batch according to the probability distribution.
- the gap between the value of the subproblems and the value of their outer linearization is checked (line 16 to 19). If the batch is ϵ_i -approximated by $(RMP)^{(k)}$, then i is increased by one, and the boolean **stay_at_x** still equals **True**. The algorithm returns to line 8 and solves a new batch at the same first-stage solution, as i has been incremented. If it reaches $i = \kappa + 1$, then all batches are ϵ_i -approximated by $(RMP)^{(k)}$ according to permutation σ , and $\tilde{x}^{(k)}$ is an optimal solution

to problem (1). If one of the batches is not ϵ_i -approximated by $(RMP)^{(k)}$, then $\check{x}^{(k)}$ cannot be an optimal solution to the problem. Then there exists at least one of the cuts which excludes the solution $(\check{x}^{(k)}, \check{\theta}^{(k)})$ from the relaxed master program. The algorithm exits the optimality loop, and goes to line 3 to solve again the relaxed master program.

Remark 2 (Aggregation of the cuts). *One of the most important drawback of the multicut Benders decomposition algorithm is the large number of cuts added to the relaxed master program at each iteration. As this number of cuts increases, the time needed to solve the master program can increase dramatically. The Benders by batch algorithm might suffer from the same effect, even if this effect might be delayed by the method (it adds fewer cuts at each iteration). We propose to aggregate the cuts of a batch, and add only one cut computed as $\sum_{s \in S_{\sigma(i)}} p_s \theta_s \geq \sum_{s \in S_{\sigma(i)}} p_s (\pi_s^\top (d_s - T_s x))$. As the subproblems are linearly independent, this cut is the Benders cut associated with the problem created by concatenation of the subproblems of a batch. The Benders by batch algorithm could also benefit from the methods proposed by Trukhanov et al. (2010) to find more effective aggregation schemes.*

The following proposition is related to the finite convergence of the algorithm.

Proposition 2. *The Benders by batch algorithm converges in a finite number of iterations.*

Proof of proposition 2. We solve each subproblem at most once for every optimal solution to $(RMP)^{(k)}$ because $(S_1, S_2, \dots, S_\kappa)$ defines a partition of S . Then if there exists a cut violated by $(\check{x}^{(k)}, (\check{\theta}_s^{(k)}))$, we find it in at most $Card(S)$ iterations in the optimality loop. Then, as the total number of optimality cuts is finite and equal to $\sum_{s \in S} Card(\text{Vert}(\Pi_s))$, this algorithm converges in at most $Card(S) \times \sum_{s \in S} Card(\text{Vert}(\Pi_s))$ iterations. When the cuts are aggregated, if the cut of a subproblem separates the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)}))$, then the aggregated cut of the batch also separates it, and the result remains true. \square

We propose an *ordered strategy* to choose the permutation σ at each iteration. We assume that there exists an initial and arbitrary ordering of the batches $S_1, S_2, \dots, S_\kappa$ and $\sigma = id$ at the first iteration. When we choose a new permutation, at the beginning of a master loop, the *ordered strategy* consists in starting from the first batch of subproblems that has not been solved at the previous first-stage solution. We introduce the following cyclic permutation μ of the batches:

$$\mu = \begin{pmatrix} 1 & 2 & \dots & \kappa - 1 & \kappa \\ 2 & 3 & \dots & \kappa & 1 \end{pmatrix}$$

Let N be the number of batches solved at the previous first-stage solution. Then, the ordered strategy consists in defining the new permutation σ at line 7 of Algorithm (2) as $\sigma \leftarrow \mu^N \circ \sigma$.

This strategy has a deterministic behavior and maintains the same number of resolutions of all the subproblems during the optimization process. A pure random strategy, shuffling the set of batches at the beginning of each master loop, showed a high variance in the total number of iterations. In preliminary computational experiments, we observed factors up to two between the running times of the fastest and the longest run on the same instance. As such a behavior is not desirable, we did not pursue this path.

4 Stabilization of the Benders by batch algorithm

The Benders by batch algorithm introduced in the previous section (Algorithm 2) may suffer, as every cutting-plane algorithm, from strong oscillations of the first-stage variables, and thus show an erratic decrease in the value of the upper bound over the iterations. We propose in this section a general framework to stabilize our algorithm, and show a sufficient condition for the convergence of the stabilized algorithm.

4.1 The stabilized Benders by batch algorithm

Many effective stabilization methods for cutting-plane algorithms solve at each iteration a separation problem in a point $x^{(k)}$ that is different from the current optimal solution $\tilde{x}^{(k)}$ of the relaxed master program (Zverovich et al., 2012; Pessoa et al., 2013). We focus on stabilization procedures in which the separation point is computed from a set a parameters, itself updated at each iteration with a first-stage solution (e.g., the solution to the relaxed master program). We call such a procedure a *primal stabilization scheme* and we formally define it below.

Definition 2 (Primal stabilization scheme). *A primal stabilization scheme is characterized by a triplet $(\mathcal{D}, \psi_1, \psi_2)$ where \mathcal{D} is a parameter space and (ψ_1, ψ_2) is a pair of applications $\begin{cases} \psi_1 : X \times \mathcal{D} \rightarrow \mathcal{D} \\ \psi_2 : \mathcal{D} \rightarrow X \end{cases}$ such that ψ_2 is surjective.*

At each iteration k of the stabilized algorithm, application ψ_2 is responsible for generating a first-stage solution $x^{(k)}$ at which the subproblems are solved, and cuts are generated, from a vector of parameters $d \in \mathcal{D}$. The surjective hypothesis ensures that every first-stage solution can be separated. Application ψ_1 computes the new vector of parameters at each iteration from the precedent vector of parameters and a first-stage solution, generally the solution to the relaxed master program.

We now present how to adapt the Benders by batch algorithm (Algorithm 2) when such a primal stabilization scheme is used. We generalize Definition 1 and Proposition 1 taking into account that the first-stage solutions at which we compute the lower bound and the upper bound may be different.

Definition 3 ($\epsilon_i(x)$ -approximation at a first-stage solution x). *Let ϵ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration and σ a permutation of $\llbracket 1, \kappa \rrbracket$. For every $i \in \llbracket 1, \kappa \rrbracket$, we say that batch $S_{\sigma(i)}$ is $\epsilon_i(x)$ -approximated by $(RMP)^{(k)}$ at $x \in X$ if*

$$\left[\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \epsilon_i(x)$$

with $\epsilon_i(x) = \epsilon - c^\top(x - \tilde{x}^{(k)}) - \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma(t)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \right]^+$ and $\zeta^+ = \max\{\zeta, 0\}$ for any $\zeta \in \mathbb{R}$.

Remark 3. *Saying that a batch $S_{\sigma(i)}$ is $\epsilon_i(\tilde{x}^{(k)})$ -approximated by $(RMP)^{(k)}$ is equivalent to saying that $S_{\sigma(i)}$ is ϵ_i -approximated by $(RMP)^{(k)}$ in Algorithm 2.*

The following proposition introduces a valid stopping criterion for our stabilized version of the Benders by batch algorithm.

Proposition 3. Let ϵ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration of the algorithm, and σ a permutation of $[[1, \kappa]]$. If there exists a first-stage solution $x \in X$ such that batch $S_{\sigma(\kappa)}$ is $\epsilon_\kappa(x)$ -approximated by $(RMP)^{(k)}$, then x is an optimal solution to problem (1).

Proof of proposition 3. Let $x \in X$ be a first-stage solution such that batch $S_{\sigma(\kappa)}$ is $\epsilon_\kappa(x)$ -approximated by $(RMP)^{(k)}$. This means:

$$\begin{aligned} & \left[\sum_{s \in S_{\sigma(\kappa)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \epsilon - c^\top(x - \check{x}^{(k)}) - \sum_{t=1}^{\kappa-1} \left[\sum_{s \in S_{\sigma(t)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \right]^+ \\ \Rightarrow & \left[\sum_{s \in S_{\sigma(\kappa)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \right]^+ + \left[\sum_{t=1}^{\kappa-1} \sum_{s \in S_{\sigma(t)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \epsilon - c^\top(x - \check{x}^{(k)}) \end{aligned}$$

As $\zeta \leq \zeta^+$ for any $\zeta \in \mathbb{R}$, we have:

$$\begin{aligned} & \sum_{t=1}^{\kappa} \sum_{s \in S_{\sigma(t)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon - c^\top(x - \check{x}^{(k)}) \\ \Rightarrow & \sum_{s \in S} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \leq \epsilon - c^\top(x - \check{x}^{(k)}) \\ \Rightarrow & \left(c^\top x + \sum_{s \in S} p_s \phi(x, s) \right) - \left(c^\top \check{x}^{(k)} + \sum_{s \in S} p_s \check{\theta}_s^{(k)} \right) \leq \epsilon \\ \Rightarrow & UB(x) - LB^{(k)} \leq \epsilon \end{aligned}$$

and x is an optimal solution to problem (1). □

We now present the stabilized Benders by batch algorithm (Algorithm 3).

The algorithm is structured in three nested *while loops*. The *while loop* from line 3 to 31 is called the *master loop*. In this loop, the relaxed master program is solved in order to define a new first-stage solution $\check{x}^{(k)}$. The *while loop* from line 5 to 31 is called the *separation loop*. This loop updates the current separation point $x^{(k)}$ while the solution to the relaxed master program $\check{x}^{(k)}$ remains constant. We increment the iteration counter k each time a new separation point is calculated. The *while loop* from line 12 to 29 is called the *optimality loop*. In the optimality loop, the subproblems of current batch $S_{\sigma(i)}$ are solved. There are three possibilities at the end of this loop:

- **Case 1:** The current batch is $\epsilon_i(x^{(k)})$ -approximated by $(RMP)^{(k)}$. It satisfies the condition of line 20 of Algorithm 3. Then, `stay_at_x` still equals `True` at the end of the loop, and i is incremented by one. If the algorithm reaches $i = \kappa + 1$, then the algorithm stops, and $x^{(k)}$ is an optimal solution to the problem. Otherwise, the algorithm solves the next batch of subproblems at the same first-stage solution.
- **Case 2:** The current batch $S_{\sigma(i)}$ is not $\epsilon_i(x^{(k)})$ -approximated by $(RMP)^{(k)}$ and at least one of the cuts derived from this batch of subproblems separates the solution $(\check{x}^{(k)}, (\theta_s^{(k)})_{s \in S})$ to the relaxed master program [see FIG. 1]. This means that `misprice` is set to `False`. The variable `stay_at_x` is set to `False` and we exit the optimality loop. Since `misprice` equals `False`, we exit the separation loop. We then go to line 3, and solve again the relaxed master program.
- **Case 3:** The current batch $S_{\sigma(i)}$ is not $\epsilon_i(x^{(k)})$ -approximated by $(RMP)^{(k)}$ and there exists no cut derived from this batch of subproblems, or a previous batch, which separates the solution

Algorithm 3: The stabilized Benders by batch algorithm

Parameters: $\epsilon > 0$, $\eta \in \llbracket 1, \text{Card}(S) \rrbracket$ the batch size, **aggregation** $\in \{\text{True}, \text{False}\}$, a primal stabilization scheme $(\mathcal{D}, \psi_1, \psi_2)$ and an initial vector of parameters $d^{(0)} \in \mathcal{D}$.

- 1 **Initialization:** $i \leftarrow 1$, $k \leftarrow 0$, **misprice** $\leftarrow \text{False}$, **stay_at_x** $\leftarrow \text{True}$
- 2 Define a partition $(S_i)_{i \in \llbracket 1, \kappa \rrbracket}$ of the subproblems according to batch size η
- 3 **while** $i < \kappa + 1$ **do**
- 4 Solve $(RMP)^{(k+1)}$ and retrieve $\check{x}^{(k+1)}$, $(\check{\theta}_s^{(k+1)})_{s \in S}$
- 5 **do**
- 6 $k \leftarrow k + 1$
- 7 $d^{(k)} \leftarrow \psi_1(\check{x}^{(k)}, d^{(k-1)})$
- 8 $x^{(k)} \leftarrow \psi_2(d^{(k)})$
- 9 $i \leftarrow 1$, $\epsilon_i \leftarrow \epsilon - c^\top(x^{(k)} - \check{x}^{(k)})$, **stay_at_x** $\leftarrow \text{True}$
- 10 Choose a permutation σ of $\llbracket 1, \kappa \rrbracket$
- 11 **misprice** $\leftarrow \text{True}$
- 12 **while** **stay_at_x** = **True** and $i < \kappa + 1$ **do**
- 13 **for** $s \in S_{\sigma(i)}$ **do**
- 14 Solve $(SP(x^{(k)}, s))$ and retrieve $\phi(x^{(k)}, s)$ and $\pi_s \in \text{Vert}(\Pi_s)$
- 15 **if** **aggregation** **then**
- 16 Add $\sum_{s \in S_{\sigma(i)}} p_s \theta_s \geq \sum_{s \in S_{\sigma(i)}} p_s (\pi_s^\top (d_s - T_s x))$ to $(RMP)^{(k)}$
- 17 **else**
- 18 **for** $s \in S_{\sigma(i)}$ **do**
- 19 Add $\theta_s \geq \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
- 20 **if** $\sum_{s \in S_{\sigma(i)}} [p_s (\phi(x^{(k)}, s) - \check{\theta}_s^{(k)})]^+ \leq \epsilon_i$ **then**
- 21 $\epsilon_{i+1} \leftarrow \epsilon - c^\top(x^{(k)} - \check{x}^{(k)}) - \left[\sum_{t=1}^i \sum_{s \in S_{\sigma(t)}} p_s (\phi(x^{(k)}, s) - \check{\theta}_s^{(k)}) \right]^+$
- 22 $i \leftarrow i + 1$
- 23 **else**
- 24 **stay_at_x** $\leftarrow \text{False}$
- 25 **if** **aggregation** **then**
- 26 **if** $\sum_{s \in S_{\sigma(i)}} p_s \check{\theta}_s^{(k)} < \sum_{s \in S_{\sigma(i)}} p_s (\pi_s^\top (d_s - T_s \check{x}^{(k)}))$ **then** **misprice** $\leftarrow \text{False}$
- 27 **else**
- 28 **for** $s \in S_{\sigma(i)}$ **do**
- 29 **if** $\check{\theta}_s^{(k)} < \pi_s^\top (d_s - T_s \check{x}^{(k)})$ **then** **misprice** $\leftarrow \text{False}$
- 30 $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$, $\check{x}^{(k+1)} \leftarrow \check{x}^{(k)}$, $(\check{\theta}_s^{(k+1)})_{s \in S} \leftarrow (\check{\theta}_s^{(k)})_{s \in S}$
- 31 **while** **misprice**
- 32 **Return** $x^{(k)}$

$(\check{x}^{(k)}, (\theta_s^{(k)})_{s \in S})$ to the relaxed master program [see FIG. 2]. The variable **misprice** still equals **True**. This is called a mis-pricing (Pessoa et al., 2013). As the solution to the relaxed master program has not been cut, it is useless to solve the relaxed master program again, its solution remains the same. We exit the optimality loop, but stay in the separation loop. We define a new separation point $x^{(k)}$, a new permutation of $\llbracket 1, \kappa \rrbracket$, and begin a new optimality loop.

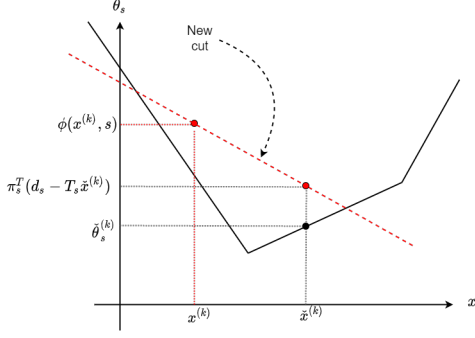


Figure 1: The cut derived from first-stage solution $x^{(k)}$ separates the solution to the relaxed master program $(\tilde{x}^{(k)}, (\tilde{\theta}_s^{(k)})_{s \in S})$.

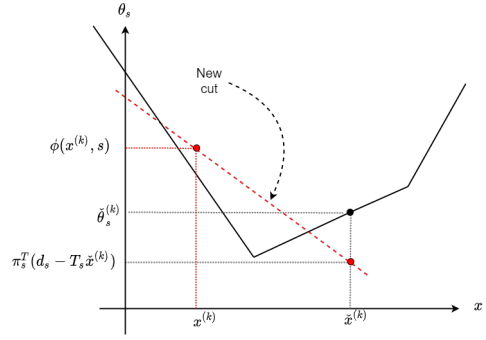


Figure 2: The cut derived from first-stage solution $x^{(k)}$ does not separate the solution to the relaxed master program $(\tilde{x}^{(k)}, (\tilde{\theta}_s^{(k)})_{s \in S})$. The solution to $(RMP)^{(k)}$ remains the same. The separation point $x^{(k)}$ induces a mis-pricing.

4.2 A sufficient condition for the convergence of the stabilized Benders by batch algorithm

In this section we prove that, if the sequence of separation points produced by the primal stabilization scheme converges to the solution to the relaxed master program when this latter solution remains constant over the iterations (i.e., during a mis-pricing sequence), then the stabilized Benders by batch algorithm (Algorithm 3) converges to an optimal solution to problem (1) in a finite number of iterations.

Definition 4 (Convergence property of a primal stabilization scheme). *Let $(\mathcal{D}, \psi_1, \psi_2)$ be a primal stabilization scheme. For every $(x, d) \in X \times \mathcal{D}$ we define $(d_x^\ell)_{\ell \in \mathbb{N}^*}$ as*

$$d_x^\ell = \begin{cases} \psi_1(x, d_x^{\ell-1}) & \ell > 1 \\ \psi_1(x, d) & \ell = 1 \end{cases} \quad \forall \ell \in \mathbb{N}^*$$

the parameters vector sequence obtained by application of ψ_1 on a constant first-stage solution $x \in X$. We say that a primal stabilization scheme $(\mathcal{D}, \psi_1, \psi_2)$ satisfies the convergence property if:

$$\forall (x, d) \in X \times \mathcal{D}, \lim_{\ell \rightarrow +\infty} \psi_2(d_x^\ell) = x$$

We first need to prove the following intermediate result.

Proposition 4. *Let $k \in \mathbb{Z}^+$ be an iteration of Algorithm 3, and $(\tilde{x}^{(k)}, (\tilde{\theta}_s^{(k)})_{s \in S})$ an optimal solution to $(RMP)^{(k)}$. Let $(x^{(k+r)})_{r \in \mathbb{N}}$ be a sequence of elements of X converging to $\tilde{x}^{(k)}$ and $(\sigma^{(k+r)})_{r \in \mathbb{N}}$ a sequence of permutations of $\llbracket 1, \kappa \rrbracket$. There exists $t \in \mathbb{N}$ such that one of the following assertions is true:*

1. *First-stage solution $x^{(k+t)}$ is proven optimal for problem (1) with an optimality gap of $\epsilon > 0$.*
2. *There exists a cut generated in $x^{(k+t)}$ which separates the solution to the relaxed master program $(\tilde{x}^{(k)}, (\tilde{\theta}_s^{(k)})_{s \in S})$.*

Proof of proposition 4. We focus on the solution $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s \in S})$ to the relaxed master program. There are two possible cases:

- **Case 1.** There exists $t_0 \in \mathbb{N}$ such that for all $l \geq t_0$ and for each index $i \in \llbracket 1, \kappa \rrbracket$, batch $S_{\sigma^{(k+l)}(i)}$ is $\epsilon_i(\check{x}^{(k)})$ -approximated by $(RMP)^{(k)}$ with an optimality gap of $\frac{\epsilon}{4}$
- **Case 2.** For all $t_0 \in \mathbb{N}$, there exists $l \geq t_0$ and an index $i \in \llbracket 1, \kappa \rrbracket$ such that batch $S_{\sigma^{(k+l)}(i)}$ is not $\epsilon_i(\check{x}^{(k)})$ -approximated by $(RMP)^{(k)}$ with an optimality gap of $\frac{\epsilon}{4}$

Case 1: Assume that there exists $t_0 \in \mathbb{N}$ such that for all $l \geq t_0$ and for each index $i \in \llbracket 1, \kappa \rrbracket$, batch $S_{\sigma^{(k+l)}(i)}$ is $\epsilon_i(\check{x}^{(k)})$ -approximated by $(RMP)^{(k)}$ with an initial gap of $\frac{\epsilon}{4}$. This means that for every $l \geq t_0$ and for every index $i \in \llbracket 1, \kappa \rrbracket$,

$$\left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \frac{\epsilon}{4} - \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \quad (7)$$

As the number of permutations of $\llbracket 1, \kappa \rrbracket$ is finite, as for every $l \geq t_0$ and for each index $i \in \llbracket 1, \kappa \rrbracket$, the application $x \mapsto \left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(x, s \right) - \check{\theta}_s^{(k)} \right) \right]^+$ is continuous, and as sequence $(x^{(k+r)})_{r \in \mathbb{N}}$ converges to $\check{x}^{(k)}$, there exists $t_1 \in \mathbb{N}$, $t_1 \geq t_0$ such that, for every $l \geq t_1$ and for every index $i \in \llbracket 1, \kappa \rrbracket$:

$$\left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ + \frac{\epsilon}{4} \quad (8)$$

Moreover, as for every $l \geq t_0$ and for every index $i \in \llbracket 1, \kappa \rrbracket$, the application $x \mapsto$

$\left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(x, s \right) - \check{\theta}_s^{(k)} \right) \right]^+$ is continuous, there exists $t_2 \in \mathbb{N}$, $t_2 \geq t_0$ such that, for every $l \geq t_2$ and for every index $i \in \llbracket 1, \kappa \rrbracket$:

$$\begin{aligned} & \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ - \frac{\epsilon}{4} \leq \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \\ \Rightarrow & - \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \leq - \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ + \frac{\epsilon}{4} \quad (9) \end{aligned}$$

And, as $(x^{(k+r)})_{r \in \mathbb{N}}$ converges to $\check{x}^{(k)}$, there exists $t_3 \in \mathbb{N}$ such that, $\forall l \geq t_3$, $0 \leq \frac{\epsilon}{4} - c^\top (x^{(k+l)} - \check{x}^{(k)})$.

Then, by setting $t_4 = \max\{t_1, t_2, t_3\}$, and jointly using (7), (8) and (9), we have, for every $l \geq t_4$ and for every index $i \in \llbracket 1, \kappa \rrbracket$:

$$\begin{aligned} & \left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \frac{\epsilon}{4} + \frac{\epsilon}{4} + \frac{\epsilon}{4} - \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \\ \Rightarrow & \left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \frac{3\epsilon}{4} - \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \\ \Rightarrow & \left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \leq \epsilon - c^\top (x^{(k+l)} - \check{x}^{(k)}) - \left[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(x^{(k+l)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \end{aligned}$$

And for every index $i \in \llbracket 1, \kappa \rrbracket$, batch $S_{\sigma^{(k+t_4)}(i)}$ is $\epsilon_i(x^{(k+t_4)})$ -approximated by $(RMP)^{(k)}$ with an optimality gap of ϵ , which implies, by Proposition 3, that $x^{(k+t_4)}$ is an optimal solution to problem (1).

Case 2: Now assume that for all $t_0 \in \mathbb{N}$, there exists $l \geq t_0$ and an index $i \in \llbracket 1, \kappa \rrbracket$ such that batch $S_{\sigma^{(k+l)}(i)}$ is not $\epsilon_i(\check{x}^{(k)})$ -approximated by $(RMP)^{(k)}$ with an initial optimality gap of $\frac{\epsilon}{4}$. This means, that for all $t_0 \in \mathbb{N}$, there exists $l \geq t_0$ and an index $i \in \llbracket 1, \kappa \rrbracket$ such that:

$$\left[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ > \frac{\epsilon}{4} - \left[\sum_{t=1}^{l-1} \sum_{s \in S_{\sigma^{(k+t)}(i)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \right]^+ \quad (10)$$

Then, there exists $\delta > 0$ such that, for all $t_0 \in \mathbb{N}$, there exists $l \geq t_0$ and an index $i \in \llbracket 1, \kappa \rrbracket$ (the first index such that (10) occurs) such that:

$$\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) > \delta \quad (11)$$

Let $g_i^{(k+\tau)} \in \mathbb{R}^{n_1}$ be a subgradient associated with the function $x \mapsto \sum_{s \in S_{\sigma^{(k+\tau)}(i)}} p_s \phi(x^{(k+\tau)}, s)$ at point $x^{(k+\tau)}$. The aggregated cut derived by resolution to batch $S_{\sigma^{(k+\tau)}(i)}$ can be written as follows:

$$g_i^{(k+\tau)\top} (x - x^{(k+\tau)}) + \sum_{s \in S_{\sigma^{(k+\tau)}(i)}} p_s \phi(x^{(k+\tau)}, s) \leq \sum_{s \in S_{\sigma^{(k+\tau)}(i)}} p_s \theta_s$$

By continuity of $\phi(\cdot, s)$ for all $s \in S$ and as the total number of cuts is finite, there exists $L > 0$ such that for every $l \in \mathbb{N}$ and for every $i \in \llbracket 1, \kappa \rrbracket$, $\|g_i^{(k+l)}\|_2 \leq L$. Then, as sequence $(x^{(k+r)})_{r \in \mathbb{N}}$ converges to $\check{x}^{(k)}$, there exists $t_1 \in \mathbb{N}$ such that for all $l \geq t_1$ and for all $i \in \llbracket 1, \kappa \rrbracket$, $|g_i^{(k+l)\top} (\check{x} - x^{(k+l)})| < \frac{\delta}{3}$. Moreover, as sequence $(x^{(k+r)})_{r \in \mathbb{N}}$ converges to $\check{x}^{(k)}$ and by continuity of $\phi(\cdot, s)$, there exists $t_2 \in \mathbb{N}$ such that for all $l \geq t_2$ and for each index $i \in \llbracket 1, \kappa \rrbracket$:

$$\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \phi(\check{x}^{(k)}, s) < \sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \phi(x^{(k+l)}, s) + \frac{\delta}{3}$$

Then, let $t_3 = \max\{t_1, t_2\}$. Let $i \in \llbracket 1, \kappa \rrbracket$ and $l_0 \geq t_3$ be the first indices such that (11) occurs. We have:

$$g_i^{(k+l_0)\top} (\check{x}^{(k)} - x^{(k+l_0)}) + \sum_{s \in S_{\sigma^{(k+l_0)}(i)}} p_s \phi(x^{(k+l_0)}, s) - \sum_{s \in S_{\sigma^{(k+l_0)}(i)}} p_s \check{\theta}_s^{(k)} > \frac{\delta}{3}$$

Then, at $x^{(k+l_0)}$, the aggregated cut of the batch $S_{\sigma^{(k+l_0)}(i)}$ separates the solution to the relaxed master program, as its value at $\check{x}^{(k)}$ is strictly larger than the outer linearization given by the relaxed master program. If `aggregation = False`, there exists at least one of the cuts associated with a subproblem of the batch which separates the solution to the relaxed master program. \square

Proposition 5. *If the primal stabilization scheme satisfies the convergence property, then the stabilized Benders by batch algorithm converges to an optimal solution to problem (1) in a finite number of iterations.*

Proof of proposition 5. Let $k \in \mathbb{Z}^+$ an iteration of the algorithm, σ a permutation of $\llbracket 1, \kappa \rrbracket$, and $x^{(k)} \in X$ the separation point. There are three possible cases:

1. $\forall i \in \llbracket 1, \kappa \rrbracket$, batch $S_{\sigma(i)}$ is $\epsilon_i(x^{(k)})$ -approximated by $(RMP)^{(k)}$. Then $x^{(k)}$ is an optimal solution to problem (1).
2. There exists an index $i \in \llbracket 1, \kappa \rrbracket$ such that solving the subproblems of batch $S_{\sigma(i)}$ generates a cut which separates the solution to $(RMP)^{(k)}$. As the total number of cuts is finite, we can only be a finite number of times in this situation.
3. There exists no cut derived at $x^{(k)}$ which separates the solution to $(RMP)^{(k)}$. Then, $x^{(k)}$ induces a mis-pricing. The solution to $(RMP)^{(k+1)}$ remains the same. Let suppose that this happens during an infinite number of consecutive iterations. Then, as the primal stabilization scheme satisfies the convergence property, the sequence of separation points converges to $\check{x}^{(k)}$. Prop. 4 states that in that case, we end up in a finite number of iterations in case 1 or case 2.

In conclusion, the stabilized Benders by batch algorithm ends in a finite number of iterations in case 1, and finds an optimal solution to problem (1). \square

Remark: The classical Benders decomposition algorithm is equivalent to the Benders by batch algorithm with a batch size $\eta = \text{Card}(S)$. Therefore, Algorithm 3 describes a valid way to add primal stabilizations to the classical Benders decomposition algorithm (providing that the primal separation scheme satisfies the convergence property).

4.3 Two primal stabilization schemes satisfying the convergence property

We introduce in this section two primal stabilization schemes satisfying the convergence property, based on the in-out stabilization approach (Ben-Ameur and Neto, 2007). In the in-out approach, the stability center $\hat{x}^{(k)}$ at iteration k is equal to the separation point (among those calculated so far) with the smallest objective function value: $\hat{x}^{(k)} = \arg \min_{j \in \llbracket 0, k-1 \rrbracket} \{c^\top x^{(j)} + \sum_{s \in S} p_s \phi(x^{(j)}, s)\}$. The separation point $x^{(k)}$ is then defined on the segment between $\hat{x}^{(k)}$ (in point) and $\check{x}^{(k)}$ (out point): $x^{(k)} = \alpha \check{x}^{(k)} + (1 - \alpha) \hat{x}^{(k)}$. The in-out approach creates a sequence of stability centers with decreasing objective values converging to an optimal solution to the problem. The definition of $\hat{x}^{(k)}$ requires computing the value $\phi(x^{(j)}, s)$ for every scenario $s \in S$, meaning that all the subproblems need to be solved at every separation point. As we generally do not solve all the subproblems at a given iteration, the in-out stabilization approach needs to be adapted for use in the Benders by batch algorithm.

We present below two primal stabilization schemes.

Scheme 1 - Basic stabilization: Let $\alpha \in (0, 1]$ be a stabilization parameter. The separation point at iteration k is computed as follows:

$$x^{(k)} = \alpha \check{x}^{(k)} + (1 - \alpha) x^{(k-1)}$$

for $k > 1$, and $x^{(1)} = \check{x}^{(1)}$.

This basically consists in doing $100\alpha\%$ of the way from the previous separation point to the solution to the master program. This could be seen as an *in-out stabilization*, updating the stability center to the last separation point at each iteration. By convexity of X , $x^{(k)}$ belongs to X for every $k \in \mathbb{N}$. This basic stabilization scheme satisfies the convergence property.

Scheme 2 - Solution memory stabilization: This stabilization uses an exponentially weighted average of the previous master solutions to compute the separation point. We choose a

stabilization parameter $\alpha \in (0, 1]$ and a memory parameter $\beta \in [0, 1)$. We also define the exponentially weighted averaged point $\bar{x}^{(k)}$ on master solutions. The separation point is computed as follows:

$$\begin{cases} \bar{x}^{(k)} &= \beta \bar{x}^{(k-1)} + (1 - \beta) \check{x}^{(k)} \\ x^{(k)} &= \alpha \bar{x}^{(k)} + (1 - \alpha) x^{(k-1)} \end{cases}$$

for $k > 1$, and $x^{(1)} = \bar{x}^{(1)} = \check{x}^{(1)}$. By convexity of X , $x^{(k)}$ belongs to X for every $k \in \mathbb{N}$. This stabilization takes inspiration from the stochastic gradient algorithm with momentum (Polyak, 1964) that has proven its efficiency in solving large-scale stochastic programs in the field of deep learning (Sutskever et al., 2013). We show in the following lemma that this solution memory stabilization scheme satisfies the convergence property.

Lemma 1. *Let $z, x^{(0)}, \bar{x}^{(0)}$ be three elements of X , $\alpha \in (0, 1]$, $\beta \in [0, 1)$. The sequence $(x^{(k)})_{k \in \mathbb{N}}$ defined by*

$$\begin{cases} \bar{x}^{(k+1)} &= \beta \bar{x}^{(k)} + (1 - \beta) z \\ x^{(k+1)} &= \alpha \bar{x}^{(k+1)} + (1 - \alpha) x^{(k)} \end{cases}$$

converges to z .

Proof of lemma 1. See Appendix A of the supplementary material. □

To limit the number of successive iterations which induce a mis-pricing, one can accelerate the convergence of the separation point to an unchanged solution to the relaxed master program for the two proposed primal stabilization schemes. Let $t \in \mathbb{N}$ be the number of consecutive mis-pricings that have occurred before starting iteration k of the algorithm. We compute $x^{(k)}$ replacing α by $\min\{1, \alpha(1 + t)\}$. It is clear that both stabilization schemes still satisfy the convergence property when this optional procedure is implemented.

5 Experimentations and numerical results

We want to estimate the numerical performance of the presented algorithms. We first present the benchmark we use, and our instance generation method. We then explain the different algorithms we compare to, and how we implemented them. Finally, we show and analyze the numerical results we obtained.

5.1 Instances

We use six well studied instances from the literature. The first five, 20term (Mak et al., 1999), gbd (Dantzig, 1963), LandS (Louveaux and Smeers, 1988), ssn (Sen et al., 1994) and storm (Mulvey and Ruszczyński, 1995), are available from the following link: www.cs.wisc.edu/~simswright/stochastic/sampling/. The problem 20term is taken from (Mak et al., 1999). It is a model of motor freight carrier's operations. The problem consists in choosing the position of some vehicles at the beginning of the day, the first-stage variables, and then to use those vehicles to satisfy some random demands on a network. Instance gbd has been created from chapter 28 of (Dantzig, 1963). It is an aircraft allocation problem. LandS has been created from an electrical investment planning problem described in (Louveaux and Smeers, 1988). In (Linderoth et al., 2006), the authors modified the problem to obtain an instance with 10^6 scenarios. Problem ssn is a problem of

telecommunication network design taken from (Sen et al., 1994) and storm is a cargo flight scheduling problem described by (Mulvey and Ruszczyński, 1995). The last instance, Fleet20_3, was found at <http://www.ie.tsinghua.edu.cn/lzhao/> which was itself taken from <https://people.orie.cornell.edu/huseyin/research/research.html>. It is a fleet-sizing problem, close to 20term, with a two-week horizon planning.

As those instances have a tremendous number of scenarios, see [FIG. 2], we generate instances by sampling scenarios from the initial ones. We generated instances with sample sizes 1000, 5000, 10000, and 20000. Three random instances have been generated for each problem and each sample size S , with random seeds $seed = S + k$, $k \in \{0, 1, 2\}$ so that two instances of different sample size should not share sub-samples. This leads to a benchmark of 72 different instances.

Table 2: Instances sizes, given in the format *lines* \times *columns*

instance	first-stage	second-stage	scenarios
LandS	2×4	7×12	10^6
gbd	4×17	5×10	$\sim 10^5$
20term	3×64	124×764	$\sim 10^{12}$
ssn	1×89	175×706	$\sim 10^{70}$
storm	185×121	528×1259	$\sim 10^{81}$
Fleet20_3	3×60	320×1920	$> 3^{200}$

5.2 Experimentations

In order to evaluate the numerical efficiency of our Benders by batch algorithm (**BbB**), we compare it to six different methods: IMB ILOG CPLEX 12.10 with its multicut Benders implementation (**CPLEX** hereafter); our implementation of the multicut Benders decomposition algorithm (**Classic multicut**); our implementation of the monocut Benders decomposition algorithm (**Classic monocut**); our implementation of the multicut Benders decomposition algorithm with an in-out stabilization (**In-out multicut**); our implementation of the monocut Benders decomposition algorithm with an in-out stabilization (**In-out monocut**); our implementation of the level bundle method (Lemaréchal et al., 1995) using aggregated cut as in the monocut Benders decomposition algorithm (**Level Bundle**). **In-out multicut**, **In-out monocut**, and **Level Bundle** are described in Appendix B of the supplementary material.

CPLEX is run with the following parameter values: **benders strategy 2** (an annotation file contains the first-stage variables, and CPLEX decomposes automatically the subproblems), **threads 1** (to run CPLEX using one core, as the other methods), **timelimit 43200** (time limit of twelve hours).

Classic multicut follows Algorithm 1. The first-stage variables appear as variables in all the subproblems, and are fixed to the desired values during the optimization process. The coefficients of the cuts are computed as the reduced cost of those variables in an optimal solution to the subproblems. We set the lower bound on the epigraph variables of the subproblems to 0 as it is a valid lower bound for every studied problem.

We use a dynamic strategy to update the stabilization parameter α in **In-out monocut** and **In-out multicut**. If $c^\top x^{(k)} + \sum_{s \in S} p_s \phi(s, x^{(k)}) < c^\top \hat{x}^{(k)} + \sum_{s \in S} p_s \phi(s, \hat{x}^{(k)})$, then the separation point has a lower cost than the current stability center. If we had separated farther, we could have found an

even better point, so we increase α with the rule $\alpha \leftarrow \min\{1.0, 1.2\alpha\}$. If $c^\top x^{(k)} + \sum_{s \in S} p_s \phi(s, x^{(k)}) \geq c^\top \hat{x}^{(k)} + \sum_{s \in S} p_s \phi(s, \hat{x}^{(k)})$, we did not stabilize enough, and we therefore decrease the stabilization parameter α with the rule $\alpha \leftarrow \max\{0.1, 0.8\alpha\}$. We initialize α to 0.5.

In **Classic monocut** and **In-out monocut**, we compute the cuts as $\sum_{s \in S} p_s \theta_s \geq \sum_{s \in S} p_s \left(\pi_s^\top (d_s - T_s x) \right)$. **Level Bundle** is tested with a level parameter $\lambda = 0.5$ and a stability center tolerance $\kappa = 0.1$ as in (van Ackooij et al., 2017). The value of $LB^{(0)}$ is set equal to 0, as it is a valid bound for our test instances. In **Level Bundle**, **Classic monocut** and **In-out monocut**, the starting solution $x^{(0)}$ is obtained by solving the mean-value problem.

The subproblems are solved with the dual simplex algorithm for all methods.

We also evaluate different parameters of the Benders by batch algorithm **BbB**. We first run **BbB** without stabilization, and try different batch sizes with and without cut aggregation. Then, we evaluate the impact of the two proposed stabilizations, with different stabilization parameters.

The experimentations are run on one core (sequential mode), on an Intel[®] Xeon[®] Gold SKL-6130 processor at 2,1 GHz with 96 Go of RAM with the TURBO boost (up to 3.7 GHz). The time limit is fixed to twelve hours for every algorithm. The optimality gap is fixed to a relative gap of 10^{-6} for every algorithm.

5.3 Numerical results

This section shows the numerical results obtained on our 72 instances. When an algorithm is stopped at its time limit of 12 hours (43 200s), the computing time is denoted $+\infty$, and the ratio to the best time will be denoted $> \frac{43200}{\text{best time}}$ in the tables, which means that this algorithm is at least this ratio slower than the best algorithm present in the table. All the tables presented in this section show, for each method, the average computing time to solve the three instances of each size, and the time ratio with respect to the best time obtained in this table. Results instance by instance are presented in Appendix C of the supplementary material. We always present the average time on the three instances of each size for each problem, rounded to the second.

We present the results with the performance profiles introduced by Dolan and Moré (2002). Let \mathcal{P} be a set of problems, and \mathcal{M} a set of methods. For any problem $p \in \mathcal{P}$ and method $m \in \mathcal{M}$, we denote as $t_{p,m}$ the computing time of method m to solve problem p . We define the *performance ratio* of method $m \in \mathcal{M}$ on problem $p \in \mathcal{P}$ as:

$$r_{p,m} = \frac{t_{p,m}}{\min_{m' \in \mathcal{M}} \{t_{p,m'}\}}$$

The performance profile of a method $m \in \mathcal{M}$ is the cumulative distribution on the set of problems of the performance ratios according to the computing time. It is defined as $\rho_m(\tau) = \text{Card}(\{p \in \mathcal{P} : r_{p,m} \leq \tau\})$

We first present in Table 3 the results on the six methods we use to benchmark algorithm **BbB**. We notice that **Cplex** is the less efficient method among the five presented in Table 3, and does not scale well when the number of subproblems becomes large. It succeeds in solving only 57 out of 72 instances. Even if **Cplex** embeds an in-out stabilization scheme and a multicut strategy with the option **benders strategy 2**, it is not competitive with **In-out multicut** in our tests. We also remark that **In-out monocut** and **In-out multicut** perform almost always better in term of computing time than their classical counterpart **Classic monocut** and **Classic multicut**. It is also

Table 3: Results for the algorithms used as a comparison basis

Times (left columns) are in second, ratios (right columns) are computed as the time divided by the best time to solve the instance by methods present in the table.

instance	Best	CPLEX		Classic multicut		Classic monocut		In-out multicut		In-out monocut		Level Bundle	
	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	1	1	1.0	1	1.3	2	3.4	1	1.5	1	2.5	1	2.2
LandS-N5000	5	5	1.0	9	1.7	11	2.0	8	1.5	9	1.8	7	1.3
LandS-N10000	14	15	1.1	29	2.1	22	1.5	24	1.7	16	1.1	14	1.0
LandS-N20000	27	43	1.6	105	3.8	45	1.7	62	2.3	41	1.5	27	1.0
gbd-N1000	1	1	1.0	1	1.2	2	2.7	1	1.6	2	2.5	2	2.7
gbd-N5000	10	10	1.0	10	1.1	12	1.2	10	1.0	10	1.1	10	1.0
gbd-N10000	21	34	1.6	33	1.6	23	1.1	23	1.1	21	1.0	24	1.1
gbd-N20000	44	131	3.0	121	2.8	48	1.1	82	1.9	54	1.2	44	1.0
ssn-N1000	6	15	2.7	7	1.2	2408	435.2	6	1.0	137	24.7	90	16.2
ssn-N5000	31	83	2.7	57	1.8	13460	437.4	31	1.0	795	25.8	657	21.3
ssn-N10000	63	180	2.8	188	3.0	25901	408.7	63	1.0	1464	23.1	1501	23.7
ssn-N20000	243	485	2.0	488	2.0	+∞	>177.9	243	1.0	2861	11.8	3109	12.8
storm-N1000	9	28	3.1	11	1.2	24	2.6	9	1.0	14	1.6	15	1.6
storm-N5000	41	187	4.6	106	2.6	114	2.8	41	1.0	62	1.5	76	1.9
storm-N10000	125	508	4.1	496	4.0	224	1.8	125	1.0	201	1.6	145	1.2
storm-N20000	252	1396	5.5	2370	9.4	458	1.8	573	2.3	252	1.0	288	1.1
20term-N1000	36	780	21.6	757	21.0	577	16.0	36	1.0	114	3.2	217	6.0
20term-N5000	482	+∞	>89.6	24429	50.7	3506	7.3	482	1.0	681	1.4	1044	2.2
20term-N10000	1190	+∞	>36.3	+∞	>36.3	6901	5.8	2805	2.4	1190	1.0	2450	2.1
20term-N20000	1754	+∞	>24.6	+∞	>24.6	13687	7.8	10992	6.3	1754	1.0	4843	2.8
Fleet20-N1000	50	148	2.9	225	4.5	533	10.6	50	1.0	93	1.9	107	2.1
Fleet20-N5000	473	15720	33.3	5330	11.3	2757	5.8	719	1.5	473	1.0	500	1.1
Fleet20-N10000	1004	+∞	>43.0	28933	28.8	5710	5.7	3747	3.7	1029	1.0	1004	1.0
Fleet20-N20000	1780	+∞	>24.3	+∞	>24.3	11300	6.3	17000	9.6	1780	1.0	2730	1.5

interesting to remark that **Level Bundle** shows similar results to **In-out monocut**, but the latter performs better on the hardest instances on our benchmark. Finally, we observe that the multicut algorithms do not scale well when the number of subproblems becomes large (except for ssn instances), in particular for instances 20term and Fleet20_3. The relaxed master programs tend to be too long to solve because of the large amount of cuts added. In the remaining tables, we only report the results for **In-out monocut** and **In-out multicut**, since for almost all the instances, one of the two methods shows the best results.

We now present the results of **BbB** without stabilization. We analyze the impact of the batch size, both without (Table 4) and with cut aggregation (Table 5). Each column of Tables 4 and 5 contains the average time in second to solve the instances and the ratio to the best time. We analyze batch sizes from 1% to 20% of the total number of subproblems (respectively denoted by **BbB 1%**, **BbB 5%**, **BbB 10%** and **BbB 20%**). Their aggregated counterparts are respectively denoted by **BbB 1% Aggreg**, **BbB 5% Aggreg**, **BbB 10% Aggreg** and **BbB 20% Aggreg**).

We first notice in Table 4 that **BbB 1%** solves almost all the instances (except for Fleet20_3 with 20000 subproblems where it succeeds to solve only one out of three problems), where **Classic Multicut** fails. The saving of subproblem resolutions and cuts added to the relaxed master program allows to overcome the computing time issues in both the subproblems and the master problem resolutions. However, **BbB** is not competitive with **In-out monocut**, except for ssn instances, where it can be up to 34.7 times faster.

Table 5 shows that **BbB** with cut aggregation is almost all the time the best method, and is almost always faster than **In-out monocut** with batch sizes of 1% and 5% of the total number of subproblems (respectively **BbB 1% Aggreg** and **BbB 5% Aggreg**). As we aggregate the cuts over each batch, the size of the relaxed master program remains reasonable, and as the cuts are only computed on a sample of subproblems, the algorithm avoids many symmetries due to the sum of the

cuts over the subproblems.

Table 4: Results for the Benders by batch algorithm without cut aggregation (no stabilization)

instance	Best	In-out multicut		In-out monocut		BbB 1%		BbB 5%		BbB 10%		BbB 20%	
	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	1	1	1.4	1	2.2	2	2.7	1	1.3	1	1.1	1	1.0
LandS-N5000	6	8	1.3	9	1.5	13	2.2	8	1.3	7	1.1	6	1.0
LandS-N10000	16	24	1.5	16	1.0	38	2.4	25	1.6	21	1.4	20	1.3
LandS-N20000	41	62	1.5	41	1.0	130	3.1	89	2.2	80	1.9	72	1.7
gbd-N1000	1	1	1.9	2	3.1	2	3.6	1	1.0	1	1.3	1	1.5
gbd-N5000	6	10	1.6	10	1.7	16	2.5	6	1.0	7	1.1	8	1.3
gbd-N10000	19	23	1.2	21	1.1	47	2.5	19	1.0	22	1.2	25	1.3
gbd-N20000	54	82	1.5	54	1.0	96	1.8	61	1.1	71	1.3	87	1.6
ssn-N1000	4	6	1.4	137	34.7	6	1.5	4	1.0	4	1.1	5	1.2
ssn-N5000	24	31	1.3	795	33.8	32	1.4	24	1.0	28	1.2	32	1.4
ssn-N10000	59	63	1.1	1464	24.9	71	1.2	79	1.3	59	1.0	79	1.3
ssn-N20000	145	243	1.7	2861	19.8	145	1.0	274	1.9	624	4.3	2821	19.5
storm-N1000	6	9	1.4	14	2.2	21	3.2	8	1.3	6	1.0	8	1.3
storm-N5000	41	41	1.0	62	1.5	175	4.3	60	1.5	55	1.3	65	1.6
storm-N10000	125	125	1.0	201	1.6	492	3.9	156	1.3	159	1.3	189	1.5
storm-N20000	252	573	2.3	252	1.0	1390	5.5	580	2.3	672	2.7	588	2.3
20term-N1000	36	36	1.0	114	3.2	38	1.1	82	2.3	49	1.4	74	2.1
20term-N5000	482	482	1.0	681	1.4	634	1.3	2101	4.4	1335	2.8	2247	4.7
20term-N10000	1190	2805	2.4	1190	1.0	2270	1.9	10733	9.0	6199	5.2	10413	8.8
20term-N20000	1754	10992	6.3	1754	1.0	20625	11.8	$+\infty$	>24.6	$+\infty$	>24.6	$+\infty$	>24.6
Fleet20-N1000	50	50	1.0	93	1.9	145	2.9	95	1.9	102	2.0	74	1.5
Fleet20-N5000	473	719	1.5	473	1.0	2417	5.1	1950	4.1	1873	4.0	2097	4.4
Fleet20-N10000	1029	3747	3.6	1029	1.0	9903	9.6	19913	19.3	8537	8.3	21383	20.8
Fleet20-N20000	1780	17000	9.6	1780	1.0	34867	19.6	$+\infty$	>24.3	$+\infty$	>24.3	$+\infty$	>24.3

Table 5: Results for the Benders by batch algorithm with cut aggregation (no stabilization)

instance	Best	In-out multicut		In-out monocut		BbB 1% Aggreg		BbB 5% Aggreg		BbB 10% Aggreg		BbB 20% Aggreg	
		time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	1	1	1.1	1	1.8	2	2.1	1	1.1	1	1.0	1	1.1
LandS-N5000	4	8	2.0	9	2.3	9	2.3	5	1.1	4	1.0	4	1.1
LandS-N10000	8	24	2.9	16	1.9	16	1.9	8	1.0	8	1.0	9	1.1
LandS-N20000	17	62	3.6	41	2.4	44	2.5	17	1.0	18	1.0	20	1.1
gbd-N1000	1	1	2.1	2	3.3	2	2.7	1	1.0	1	1.3	1	1.5
gbd-N5000	3	10	3.1	10	3.2	9	2.7	3	1.0	4	1.1	4	1.3
gbd-N10000	6	23	3.7	21	3.3	15	2.3	6	1.0	8	1.3	9	1.5
gbd-N20000	14	82	6.0	54	3.9	41	3.0	14	1.0	15	1.1	19	1.4
ssn-N1000	6	6	1.0	137	24.7	14	2.5	61	11.1	134	24.2	242	43.8
ssn-N5000	31	31	1.0	795	25.8	89	2.9	322	10.5	659	21.4	1322	43.0
ssn-N10000	63	63	1.0	1464	23.1	185	2.9	707	11.2	1423	22.5	2914	46.0
ssn-N20000	243	243	1.0	2861	11.8	441	1.8	1615	6.7	3386	13.9	6757	27.8
storm-N1000	6	9	1.5	14	2.3	12	1.9	6	1.0	7	1.1	9	1.5
storm-N5000	34	41	1.2	62	1.8	52	1.5	34	1.0	36	1.0	55	1.6
storm-N10000	74	125	1.7	201	2.7	110	1.5	74	1.0	82	1.1	104	1.4
storm-N20000	163	573	3.5	252	1.5	226	1.4	163	1.0	169	1.0	238	1.5
20term-N1000	15	36	2.5	114	7.7	15	1.0	37	2.5	68	4.6	141	9.6
20term-N5000	70	482	6.9	681	9.7	70	1.0	193	2.7	395	5.6	839	12.0
20term-N10000	130	2805	21.6	1190	9.2	130	1.0	402	3.1	898	6.9	1978	15.2
20term-N20000	280	10992	39.3	1754	6.3	280	1.0	914	3.3	2051	7.3	18312	65.4
Fleet20-N1000	28	50	1.8	93	3.3	28	1.0	42	1.5	74	2.6	131	4.6
Fleet20-N5000	107	719	6.7	473	4.4	107	1.0	211	2.0	358	3.3	649	6.0
Fleet20-N10000	212	3747	17.7	1029	4.9	212	1.0	440	2.1	721	3.4	1310	6.2
Fleet20-N20000	419	17000	40.6	1780	4.3	419	1.0	876	2.1	1520	3.6	2777	6.6

We summarize the results of Tables 4 and 5 in Figure 3. Figure 3 confirms that **BbB 1% Aggreg** and **BbB 5% Aggreg** are the two best algorithms on our benchmark. They show the best computing times in respectively 24 out of 72 instances and 33 out of 72. Algorithm **BbB 1% Aggreg** scales better as its higher performance ratio is lower than 4.

We now present the results for the two stabilization schemes presented in §4.3. We performed the tests for the stabilized Benders by batch algorithm with batch sizes of 1% and 5%, and with cut

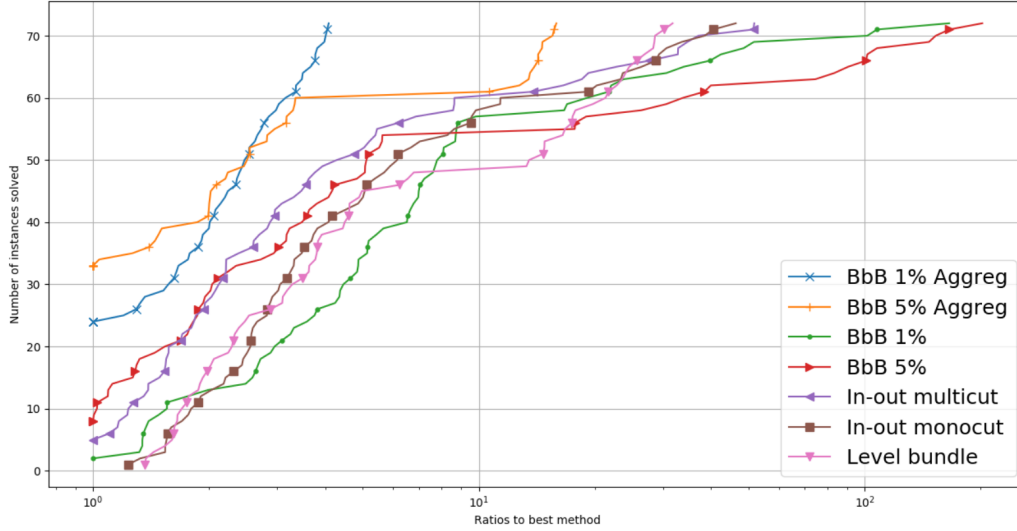


Figure 3: Performance profiles of Classical Benders decomposition with in-out stabilization and Benders by batch algorithm with and without cut aggregation

aggregation, as these parameters gave the most competitive results for the unstabilized version of the algorithm. Figures 4 and 5 show the performance profiles of **BbB** with and without stabilization. We present the results with basic stabilization for $\alpha \in \{0.1, 0.5, 0.9\}$ and with solution memory stabilization for $\alpha \in \{0.1, 0.5, 0.9\}$ and $\beta \in \{0.1, 0.5, 0.9\}$. Each stabilized method is denoted by **BbB 1% Aggreg** or **BbB 5% Aggreg** followed by the values for the parameters.

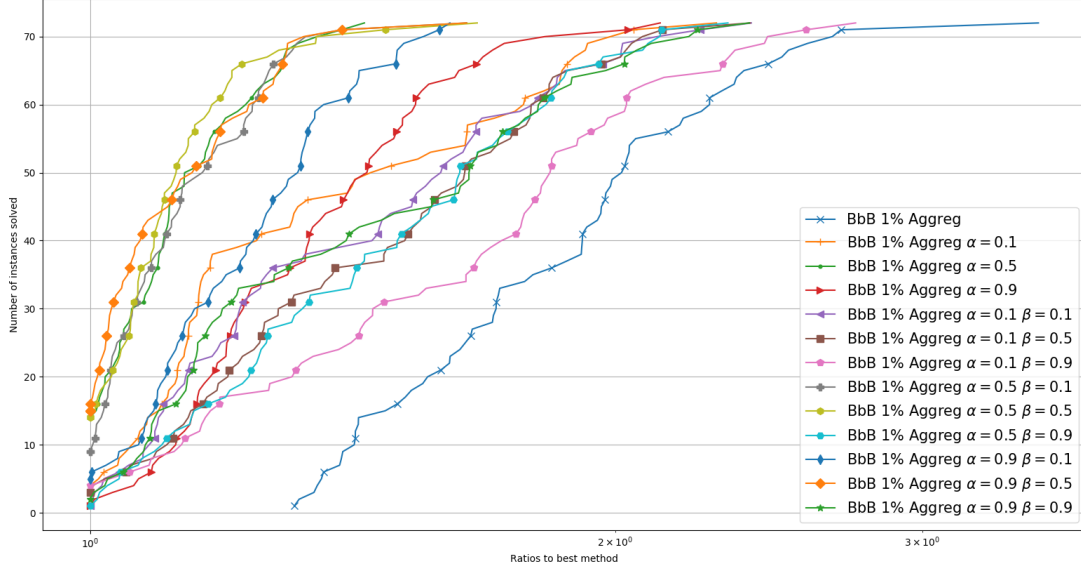


Figure 4: Performance profiles of the stabilized Benders by batch algorithm with batch size of 1% and cut aggregation.

Figure 4 shows that the proposed stabilization schemes accelerate **BbB 1% Aggreg**, and can be up to 70% faster than the unstabilized algorithm. Four stabilizations are more efficient on the tested instances and give similar results, namely the basic stabilization with $\alpha = 0.5$, and the solution memory stabilization with $(\alpha, \beta) \in \{(0.5, 0.1), (0.5, 0.5), (0.9, 0.5)\}$.

Figure 5 shows similar results for **BbB 5% Aggreg**. The same four methods are the most efficient and equivalent to each other. The algorithm with a solution memory stabilization parameterized by

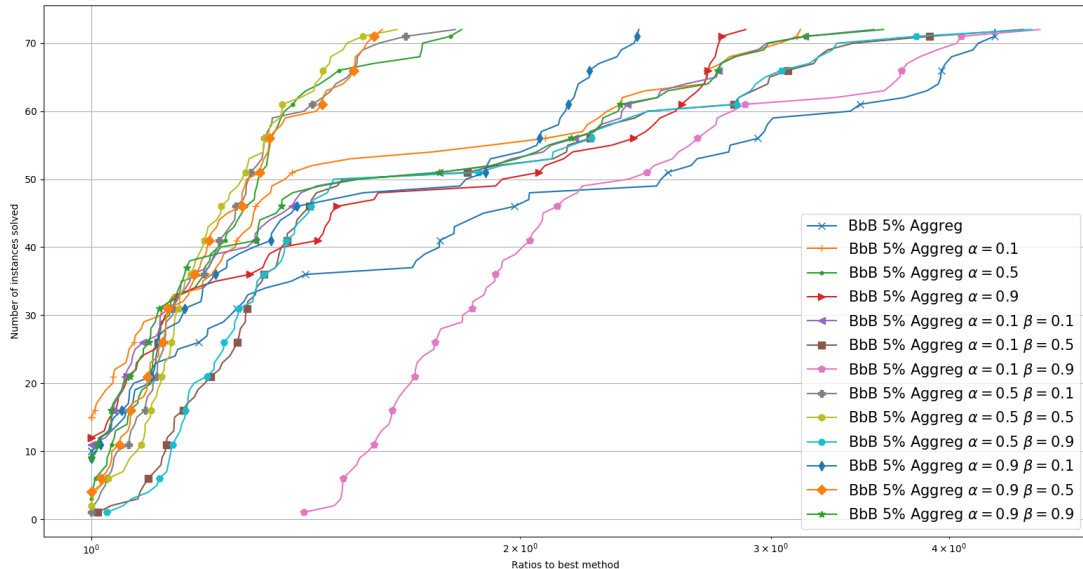


Figure 5: Performance profiles of the stabilized Benders by batch algorithm with batch size of 5% and cut aggregation.

$(\alpha, \beta) = (0.1, 0.9)$ is less efficient than **BbB 5% Aggreg**. In this case, a small step size ($\alpha = 0.1$) and a high memory parameter ($\beta = 0.9$) slow down the convergence. For all the other cases, the use of a primal stabilization scheme accelerates the algorithm.

Finally, results show no clear difference between the two proposed stabilization schemes. It seems that the solution memory stabilization does efficiently stabilize the algorithm, but the basic stabilization might be the method of choice as it is much simpler and provides quite similar computational results.

As final results, we show in Table 6 the times and ratios of the three best methods in our benchmark, **In-out monocut**, **In-out multicut**, and **Level bundle**, with the best versions of **BbB** (with and without stabilization). The table shows clearly that the stabilized Benders by batch algorithm outperforms the other methods on the tested instances, and can be up to more than 20 times faster than **Level Bundle** or 15 times faster than **In-out monocut**. We also show in Figure 6 the performance profiles of the first six methods we compare to, presented in Table 3 and **BbB 1% Aggreg** with $\alpha = 0.5$. The latter is the best algorithm for 62 out of 72 instances and shows up to two orders of magnitude of acceleration compared to **Classic monocut**, **Classic multicut** or **CPLEX**.

6 Conclusion

We proposed in this paper the Benders by batch algorithm to solve two-stage stochastic linear programs in which we solve only a few subproblems at most iterations. The algorithm is exact and does not need any assumptions on the structure of the problem. We showed that solving only a very few number of subproblems, 1% in our tests, allows us to significantly improve the solution time, and to solve large instances that classical Benders decomposition algorithms fail to solve in 12 hours.

To avoid strong oscillations of the first-stage variables, we also introduced a stabilized version of the algorithm. This algorithm is based on a primal stabilization scheme responsible for generating the points at which the subproblems are solved. We presented a sufficient condition for a primal stabilization scheme that ensures the convergence of the Benders by batch algorithm and proposed

Table 6: Comparison of the stabilized and unstabilized version of the Benders by batch algorithm with the best other implemented methods

instance	Best	In-out multicut		In-out monocut		Level Bundle		BBB 1% Aggreg		Bbb 1% Aggreg — $\alpha = 0.5$	
	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	1	1	1.0	1	1.6	1	1.4	2	1.9	1	1.1
LandS-N5000	5	8	1.6	9	1.9	7	1.3	9	1.8	5	1.0
LandS-N10000	11	24	2.1	16	1.4	14	1.3	16	1.4	11	1.0
LandS-N20000	22	62	2.8	41	1.9	27	1.2	44	2.0	22	1.0
gbd-N1000	1	1	1.5	2	2.3	2	2.5	2	1.9	1	1.0
gbd-N5000	4	10	2.3	10	2.3	10	2.2	9	2.0	4	1.0
gbd-N10000	9	23	2.7	21	2.4	24	2.7	15	1.7	9	1.0
gbd-N20000	17	82	4.9	54	3.2	44	2.6	41	2.4	17	1.0
ssn-N1000	6	6	1.0	137	24.7	90	16.2	14	2.5	8	1.5
ssn-N5000	31	31	1.0	795	25.8	657	21.3	89	2.9	50	1.6
ssn-N10000	63	63	1.0	1464	23.1	1501	23.7	185	2.9	94	1.5
ssn-N20000	195	243	1.2	2861	14.6	3109	15.9	441	2.3	195	1.0
storm-N1000	7	9	1.3	14	2.1	15	2.2	12	1.8	7	1.0
storm-N5000	33	41	1.2	62	1.9	76	2.3	52	1.6	33	1.0
storm-N10000	68	125	1.8	201	2.9	145	2.1	110	1.6	68	1.0
storm-N20000	129	573	4.4	252	1.9	288	2.2	226	1.8	129	1.0
20term-N1000	11	36	3.4	114	10.7	217	20.5	15	1.4	11	1.0
20term-N5000	53	482	9.1	681	12.8	1044	19.6	70	1.3	53	1.0
20term-N10000	105	2805	26.8	1190	11.4	2450	23.4	130	1.2	105	1.0
20term-N20000	231	10992	47.6	1754	7.6	4843	21.0	280	1.2	231	1.0
Fleet20-N1000	17	50	2.9	93	5.4	107	6.2	28	1.6	17	1.0
Fleet20-N5000	76	719	9.5	473	6.2	500	6.6	107	1.4	76	1.0
Fleet20-N10000	152	3747	24.7	1029	6.8	1004	6.6	212	1.4	152	1.0
Fleet20-N20000	311	17000	54.6	1780	5.7	2730	8.8	419	1.3	311	1.0

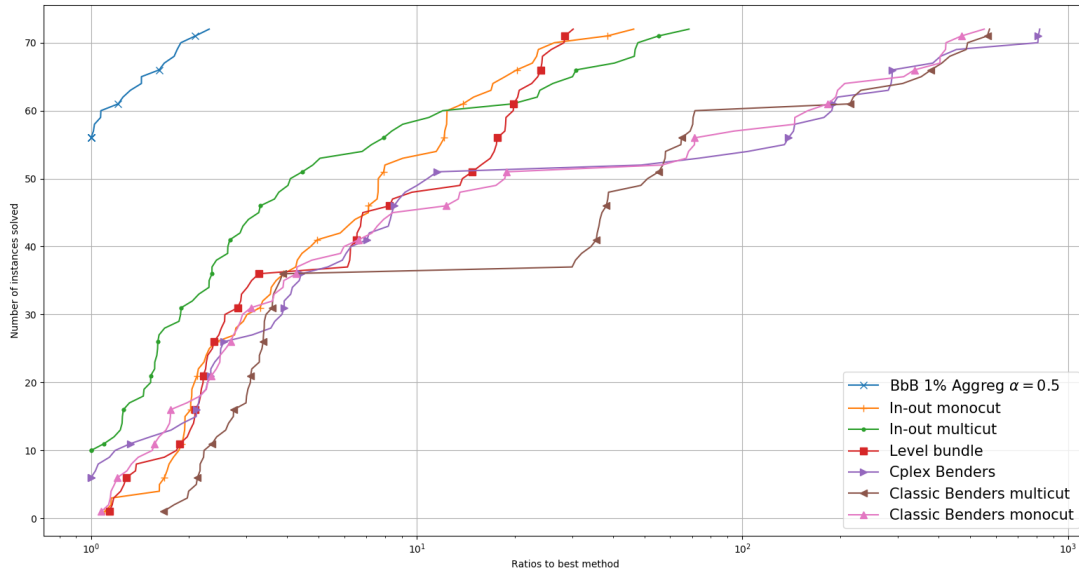


Figure 6: Performance profile of the best version of the stabilized Benders by batch algorithm and of the other implemented methods

two schemes satisfying it. The stabilized Benders by batch algorithm with cut aggregation solves in at most seven minutes some large instances of the literature which were not solved in 12 hours by the built-in Benders decomposition of CPLEX 12.10 or the Benders decomposition without stabilization. This algorithm showed speed-up factors of up to 15 over the best methods of the literature that we compared it to.

Applying dual stabilization (Magnanti and Wong, 1981; Sherali and Lunday, 2013) to the Benders by batch algorithm is straightforward and could improve the results. The algorithm can also be parallelized and may benefit from effective parallelized methods, such as the asynchronous method

of Linderoth and Wright (2003). Finally, an interesting perspective is to adapt the Benders by batch algorithm to solve mixed-integer master programs within a Branch&Cut framework.

Acknowledgements

This project has been funded by RTE (Réseau de Transport d'Electricité), french company in charge of the electricity network management, through the projects Antares and Antares Xpansion: <https://github.com/AntaresSimulatorTeam/antares-xpansion>, which are used for long-term adequacy studies. Computer time for this study was provided by the computing facilities MCIA (Mésocentre de Calcul Intensif Aquitain) of the Université de Bordeaux and of the Université de Pau et des Pays de l'Adour.

References

- Ben-Ameur, W. and Neto, J. (2007). Acceleration of cutting-plane and column generation algorithms: Applications to network design. *Networks*, 49(1):3–17.
- Birge, J. R. and Louveaux, F. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392.
- Crainic, T. G., Hewitt, M., Maggioni, F., and Rei, W. (2020). Partial Benders Decomposition: General Methodology and Application to Stochastic Network Design. *Transportation Science*, 55(2):414–435.
- Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton, New Jersey, princeton university press edition.
- Dantzig, G. B. and Infanger, G. (1991). Large-Scale Stochastic Linear Programs: Importance Sampling and Benders Decomposition:. Technical report, Defense Technical Information Center, Fort Belvoir, VA.
- de Oliveira, W. and Sagastizábal, C. (2014). Level bundle methods for oracles with on-demand accuracy. *Optimization Methods and Software*, 29(6):1180–1209.
- Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.
- Fischetti, M., Ljubić, I., and Sinnl, M. (2016). Redesigning Benders Decomposition for Large-Scale Facility Location. *Management Science*, 63(7):2146–2162.
- Fischetti, M. and Salvagnin, D. (2010). An In-Out Approach to Disjunctive Optimization. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140, pages 136–140. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Higle, J. L. and Sen, S. (1991). Stochastic Decomposition : An Algorithm for Two-Stage Linear Programms with Recours. *Mathematics of Operations Research*, 16(3):447–669.
- Infanger, G. (1992). Monte Carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research*, 39(1):69–95.

- Lemaréchal, C., Nemirovskii, A., and Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, 69(1-3):111–147.
- Linderoth, J., Shapiro, A., and Wright, S. (2006). The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241.
- Linderoth, J. and Wright, S. (2003). Decomposition Algorithms for Stochastic Programming on a Computational Grid. *Computational Optimization and Applications*, 24(2):207–250.
- Louveaux, F. and Smeers, Y. (1988). Optimal Investments for Electricity Generation: A Stochastic Model and a Test-Problem. In *Numerical Techniques for Stochastic Optimization*, Y. Ermoliev and R.J.-B. Wets (Eds.), pages 445–454, Berlin. Springer-Verlag,.
- Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. *Operations Research*, 29(3):464–484.
- Mak, W.-K., Morton, D. P., and Wood, R. (1999). Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1-2):47–56.
- Mulvey, J. M. and Ruszczyński, A. (1995). A New Scenario Decomposition Method for Large-Scale Stochastic Optimization. *Operations Research*, 43(3):477–490.
- Oliveira, W., Sagastizábal, C., and Scheimberg, S. (2011). Inexact Bundle Methods for Two-Stage Stochastic Programming. *SIAM Journal on Optimization*, 21(2):517–544.
- Papadakos, N. (2008). Practical enhancements to the Magnanti–Wong method. *Operations Research Letters*, 36(4):444–449.
- Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2013). In-Out Separation and Column Generation Stabilization by Dual Price Smoothing. In *Experimental Algorithms*, volume 7933, pages 354–365. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.
- Ruszczynski, A. (1986). A regularized decomposition method for minimizing a sum of polyhedral functions. *Mathematical Programming*, 35(3):309–333.
- Ruszczynski, A. (1997). Decomposition methods in stochastic programming. *Mathematical Programming*, 79(1):333–353.
- Sen, S., Doverspike, R. D., and Cosares, S. (1994). Network planning with random demand. *Telecommunication Systems*, 3(1):11–30.
- Sherali, H. D. and Lunday, B. J. (2013). On generating maximal nondominated Benders cuts. *Annals of Operations Research*, 210(1):57–72.
- Song, Y. and Luedtke, J. (2015). An Adaptive Partition-Based Approach for Solving Two-Stage Stochastic Programs with Fixed Recourse. *SIAM Journal on Optimization*, 25(3):1344–1367.

- Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.
- Trukhanov, S., Ntaimo, L., and Schaefer, A. (2010). Adaptive multicut aggregation for two-stage stochastic linear programs with recourse. *European Journal of Operational Research*, 206(2):395–406.
- van Ackooij, W., de Oliveira, W., and Song, Y. (2017). Adaptive Partition-Based Level Decomposition Methods for Solving Two-Stage Stochastic Programs with Fixed Recourse. *INFORMS Journal on Computing*, 30(1):57–70.
- Van Slyke, R. M. and Wets, R. (1969). L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.
- Wets, R. (1983). Stochastic Programming: Solution Techniques and Approximation Schemes. In *Mathematical Programming The State of the Art*, pages 566–603. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Wolf, C., Fábíán, C. I., Koberstein, A., and Suhl, L. (2014). Applying oracles of on-demand accuracy in two-stage stochastic programming – A computational study. *European Journal of Operational Research*, 239(2):437–448.
- You, F. and Grossmann, I. E. (2013). Multicut Benders decomposition algorithm for process supply chain planning under uncertainty. *Annals of Operations Research*, 210(1):191–211.
- Zverovich, V., Fábíán, C. I., Ellison, E. F. D., and Mitra, G. (2012). A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition. *Mathematical Programming Computation*, 4(3):211–238.

A Proof of Lemma 1

Proof of lemma 1. From the definition of $(x^{(k)})_{k \in \mathbb{N}}$,

$$\begin{cases} \bar{x}^{(k+1)} - z &= \beta(\bar{x}^{(k)} - z) \\ x^{(k+1)} - z &= \alpha\beta(\bar{x}^{(k)} - z) + (1 - \alpha)(x^{(k)} - z) \end{cases}$$

We define $u^{(k)} = x^{(k)} - z$ and $v^{(k)} = \bar{x}^{(k)} - z$, for every $k \in \mathbb{N}^*$.

$$\begin{cases} v^{(k+1)} &= \beta v^{(k)} \\ u^{(k+1)} &= \alpha\beta v^{(k)} + (1 - \alpha)u^{(k)} \end{cases}$$

We define

$$A = \begin{bmatrix} \beta & 0 \\ \alpha\beta & (1 - \alpha) \end{bmatrix}$$

Then we have:

$$\begin{pmatrix} v_1^{(k+1)} \\ u_1^{(k+1)} \\ \vdots \\ v_{n_1}^{(k+1)} \\ u_{n_1}^{(k+1)} \end{pmatrix} = \begin{bmatrix} A & & \\ & \ddots & \\ & & A \end{bmatrix} \cdot \begin{pmatrix} v_1^{(k)} \\ u_1^{(k)} \\ \vdots \\ v_{n_1}^{(k)} \\ u_{n_1}^{(k)} \end{pmatrix} = \begin{bmatrix} A^k & & \\ & \ddots & \\ & & A^k \end{bmatrix} \cdot \begin{pmatrix} v_1^{(1)} \\ u_1^{(1)} \\ \vdots \\ v_{n_1}^{(1)} \\ u_{n_1}^{(1)} \end{pmatrix}$$

If $(1 - \alpha) \neq \beta$, then the characteristic polynomial of A has two distinct roots, so A is diagonalizable and $Sp(A) = \{1 - \alpha, \beta\}$. Since $-1 < 1 - \alpha < 1$ and $-1 < \beta < 1$, the sequence $(A^k)_{k \in \mathbb{N}}$ converges to the null matrix. If $(1 - \alpha) = \beta$, then we have:

$$A^k = \begin{bmatrix} (1 - \alpha)^k & 0 \\ k\alpha(1 - \alpha)^k & (1 - \alpha)^k \end{bmatrix}$$

Since $k\alpha(1 - \alpha)^k$ converges to 0 for every $\alpha \in (0, 1]$, the sequence $(A^k)_{k \in \mathbb{N}}$ converges to the null matrix. This proves that the sequence $(u^{(k)}, v^{(k)})_{k \in \mathbb{N}}$ converges to $(0, 0)$. Then the sequence $(x^{(k)})_{k \in \mathbb{N}}$ converges to z . \square

B Detailed benchmark algorithms

Algorithm 4 describes our implementation of **In-out monocut** (`aggregation=True`) and **In-out multicut** (`aggregation=False`).

Algorithm 4: The Benders decomposition algorithm with in-out stabilization

Parameters: $\epsilon > 0$ the selected optimality gap, $x^{(0)} \in X$, `aggregation` $\in \{True, False\}$, $\alpha \in (0; 1]$

- 1 **Initialization:** $k \leftarrow 0$, $\hat{x}^{(1)} \leftarrow x^{(0)}$, $UB^{(0)} \leftarrow c^\top x^{(0)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(0)})$, $LB^{(0)} \leftarrow -\infty$, $\alpha_1 \leftarrow \alpha$
- 2 **while** $UB^{(k)} > LB^{(k)} + \epsilon$ **do**
- 3 $k \leftarrow k + 1$
- 4 Solve $(RMP)^{(k)}$ and retrieve $\tilde{x}^{(k)}$, $(\check{\theta}_s^{(k)})_{s \in S}$
- 5 $LB^{(k)} \leftarrow c^\top \tilde{x}^{(k)} + \sum_{s \in S} p_s \check{\theta}_s^{(k)}$
- 6 $x^{(k)} \leftarrow \alpha_k \tilde{x}^{(k)} + (1 - \alpha_k) \hat{x}^{(k)}$
- 7 **for** $s \in S$ **do**
- 8 \lfloor Solve $(SP(x^{(k)}, s))$ and retrieve π_s an extreme point or an extreme ray of Π_s
- 9 **if** `aggregation` **then**
- 10 Add $\sum_{s \in S} p_s \theta_s \geq \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x)$
- 11 **else**
- 12 **for** $s \in S$ **do**
- 13 \lfloor Add $\theta_s \geq \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
- 14 **if** $UB^{(k-1)} > c^\top x^{(k)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$ **then**
- 15 $UB^{(k)} \leftarrow c^\top x^{(k)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$
- 16 $\hat{x}^{(k+1)} \leftarrow x^{(k)}$
- 17 $\alpha_{k+1} \leftarrow \min\{1.0, 1.2\alpha_k\}$
- 18 **else**
- 19 $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$, $UB^{(k)} \leftarrow UB^{(k-1)}$
- 20 $\alpha_{k+1} \leftarrow \max\{0.1, 0.8\alpha_k\}$
- 21 $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$
- 22 **Return** $\hat{x}^{(k+1)}$

We now describe the level bundle method. We first define the quadratic master program. Let $\lambda \in (0, 1)$ denote the level parameter, LB a lower bound on the optimal value of the problem, and UB an upper bound. We define $f_{lev} = (1 - \lambda)UB + \lambda LB$ and a stability center \hat{x} as in the in-out stabilization approach. The quadratic master program $(QMP)(\hat{x}, f_{lev})$ parametrized by \hat{x} and f_{lev} is the following:

$$\left\{ \begin{array}{l} \min_{x, \theta} \frac{1}{2} \|x - \hat{x}\|_2^2 \\ s.t. : \sum_{s \in S} p_s \theta_s \geq \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x), \quad \forall s \in S, \quad \forall \pi_s \in \text{Vert}(\Pi_s) \\ c^\top x + \sum_{s \in S} p_s \theta_s \leq f_{lev} \\ x \in X, \theta \in \mathbb{R}^{\text{Card}(S)} \end{array} \right.$$

We denote by $(RQMP)^{(k)}(\hat{x}, f_{lev})$ its relaxation at iteration k of the algorithm and by $\kappa \in (0, \lambda)$ a acceptance tolerance to update the stability center. Algorithm 5 describes our implementation of **Level bundle**.

Algorithm 5: Level bundle method

Parameters: $\epsilon > 0$ the selected optimality gap, $x^{(0)} \in X$, $\lambda \in [0, 1)$, $LB^{(0)}$ a valid lower bound on the objective value, $\kappa \in (0, \lambda)$

- 1 **Initialization:** $k \leftarrow 0$, $UB^{(0)} \leftarrow c^\top x^{(0)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s \hat{x}^{(0)})$, $\hat{x}^{(1)} \leftarrow x^{(0)}$
- 2 **while** $UB^{(k)} > LB^{(k)} + \epsilon$ **do**
- 3 $k \leftarrow k + 1$
- 4 $f_{lev}^{(k)} = (1 - \lambda)UB^{(k-1)} + \lambda LB^{(k-1)}$
- 5 Solve $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$
- 6 **if** $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$ *is infeasible* **then**
- 7 $LB^{(k)} \leftarrow f_{lev}^{(k)}$
- 8 $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$
- 9 $UB^{(k)} \leftarrow UB^{(k-1)}$
- 10 **else**
- 11 Retrieve $x^{(k)}$ solution to $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$
- 12 **for** $s \in S$ **do**
- 13 Solve $(SP(x^{(k)}, s))$ and retrieve π_s an extreme point or an extreme ray of Π_s
- 14 Add $\sum_{s \in S} p_s \theta_s \geq \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x)$
- 15 **if** $c^\top x^{(k)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(k)}) < (1 - \kappa)UB^{(k-1)} + \kappa f_{lev}^{(k)}$ **then**
- 16 $UB^{(k)} \leftarrow c^\top x^{(k)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$
- 17 $\hat{x}^{(k+1)} \leftarrow x^{(k)}$
- 18 **else**
- 19 $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$
- 20 $UB^{(k)} \leftarrow UB^{(k-1)}$
- 21 $LB^{(k)} \leftarrow LB^{(k-1)}$
- 22 $(RQMP)^{(k+1)} \leftarrow (RQMP)^{(k)}$
- 23 **Return** $\hat{x}^{(k+1)}$

C Detailed numerical results

This section shows the detailed numerical results.

instance	CPLEX			Classic multicut		Classic monocut		In-out multicut		In-out monocut		Level bundle	
	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	1	1	1.0	1	1.4	2	3.4	1	1.9	2	3.6	1	2.1
LandS-N1000-s1001	1	1	1.0	1	1.3	2	3.8	1	1.1	1	2.2	1	2.5
LandS-N1000-s1002	1	1	1.0	1	1.1	2	3.1	1	1.6	1	1.7	1	2.0
LandS-N5000-s5000	5	5	1.0	9	1.7	11	2.1	8	1.6	10	2.0	8	1.5
LandS-N5000-s5001	6	6	1.0	10	1.7	10	1.8	8	1.4	6	1.2	7	1.3
LandS-N5000-s5002	5	5	1.0	9	1.7	11	2.1	8	1.6	12	2.2	6	1.1
LandS-N10000-s10000	11	15	1.4	26	2.3	22	2.0	24	2.1	11	1.0	14	1.2
LandS-N10000-s10001	13	15	1.1	30	2.3	22	1.7	24	1.9	13	1.0	13	1.0
LandS-N10000-s10002	14	14	1.0	30	2.1	20	1.4	23	1.6	23	1.6	15	1.0
LandS-N20000-s20000	28	44	1.5	96	3.4	49	1.8	71	2.5	42	1.5	28	1.0
LandS-N20000-s20001	26	43	1.7	119	4.6	43	1.7	67	2.6	40	1.5	26	1.0
LandS-N20000-s20002	29	44	1.5	99	3.5	44	1.5	48	1.7	43	1.5	29	1.0
gbd-N1000-s1000	1	1	1.0	1	1.2	2	2.4	1	1.6	3	3.4	2	2.2
gbd-N1000-s1001	1	1	1.0	1	1.1	2	2.9	1	1.5	2	1.9	2	2.9
gbd-N1000-s1002	1	1	1.0	1	1.2	2	2.7	1	1.6	2	2.1	2	2.9
gbd-N5000-s5000	7	10	1.3	10	1.4	13	1.7	7	1.0	13	1.8	8	1.1
gbd-N5000-s5001	9	10	1.1	10	1.1	11	1.2	11	1.3	9	1.0	11	1.2
gbd-N5000-s5002	9	10	1.1	11	1.2	12	1.3	12	1.4	9	1.0	11	1.2
gbd-N10000-s10000	19	33	1.8	34	1.9	24	1.3	19	1.0	30	1.6	23	1.2
gbd-N10000-s10001	18	34	1.9	32	1.8	24	1.3	32	1.8	18	1.0	26	1.5
gbd-N10000-s10002	15	33	2.2	32	2.1	23	1.5	20	1.3	15	1.0	23	1.5
gbd-N20000-s20000	45	130	2.9	119	2.6	48	1.1	107	2.4	56	1.2	45	1.0
gbd-N20000-s20001	47	131	2.8	120	2.6	51	1.1	72	1.5	55	1.2	47	1.0
gbd-N20000-s20002	39	132	3.4	125	3.2	47	1.2	69	1.7	51	1.3	39	1.0
ssn-N1000-s1000	4	16	3.9	7	1.7	2279	565.1	4	1.0	187	46.4	97	24.0
ssn-N1000-s1001	6	16	2.5	7	1.1	2720	433.0	6	1.0	117	18.7	85	13.6
ssn-N1000-s1002	6	14	2.3	7	1.1	2226	354.0	6	1.0	106	16.9	87	13.8
ssn-N5000-s5000	35	82	2.3	62	1.8	13425	379.6	35	1.0	936	26.5	621	17.6
ssn-N5000-s5001	35	81	2.3	45	1.3	14260	409.5	35	1.0	597	17.1	719	20.6
ssn-N5000-s5002	22	85	3.9	64	2.9	12695	573.7	22	1.0	852	38.5	631	28.5
ssn-N10000-s10000	86	163	1.9	185	2.2	26559	310.5	86	1.0	1937	22.6	1440	16.8
ssn-N10000-s10001	53	190	3.6	193	3.6	26228	491.6	53	1.0	1261	23.6	1613	30.2
ssn-N10000-s10002	51	188	3.7	187	3.6	24916	486.5	51	1.0	1195	23.3	1451	28.3
ssn-N20000-s20000	245	478	1.9	512	2.1	+∞	>176.1	245	1.0	3791	15.5	3232	13.2
ssn-N20000-s20001	237	484	2.0	503	2.1	+∞	>182.0	237	1.0	2460	10.4	2986	12.6
ssn-N20000-s20002	246	494	2.0	450	1.8	+∞	>175.8	246	1.0	2332	9.5	3108	12.6
storm-N1000-s1000	10	28	2.9	10	1.1	23	2.4	10	1.0	11	1.1	14	1.5
storm-N1000-s1001	7	28	4.1	11	1.5	24	3.4	7	1.0	21	3.0	16	2.2
storm-N1000-s1002	11	28	2.6	11	1.0	24	2.2	11	1.0	12	1.1	15	1.4
storm-N5000-s5000	41	191	4.7	100	2.5	110	2.7	41	1.0	63	1.5	74	1.8
storm-N5000-s5001	38	183	4.9	118	3.1	117	3.1	38	1.0	61	1.6	78	2.1
storm-N5000-s5002	43	188	4.3	99	2.3	116	2.7	43	1.0	63	1.5	76	1.8
storm-N10000-s10000	108	525	4.9	468	4.3	215	2.0	108	1.0	212	2.0	140	1.3
storm-N10000-s10001	105	516	4.9	479	4.6	225	2.1	105	1.0	201	1.9	149	1.4
storm-N10000-s10002	147	482	3.3	542	3.7	233	1.6	161	1.1	189	1.3	147	1.0
storm-N20000-s20000	259	1381	5.3	2240	8.6	465	1.8	515	2.0	259	1.0	316	1.2
storm-N20000-s20001	251	1524	6.1	2460	9.8	434	1.7	633	2.5	251	1.0	266	1.1
storm-N20000-s20002	246	1283	5.2	2410	9.8	476	1.9	570	2.3	246	1.0	283	1.2
20term-N1000-s1000	27	817	30.2	749	27.7	544	20.1	27	1.0	128	4.7	197	7.3
20term-N1000-s1001	43	559	13.0	646	15.0	584	13.6	43	1.0	74	1.7	214	5.0
20term-N1000-s1002	38	965	25.4	877	23.0	604	15.9	38	1.0	139	3.7	241	6.3
20term-N5000-s5000	581	+∞	>74.3	29455	50.7	3095	5.3	581	1.0	661	1.1	994	1.7
20term-N5000-s5001	423	+∞	>102.1	22490	53.2	3699	8.7	423	1.0	650	1.5	1059	2.5
20term-N5000-s5002	443	+∞	>97.6	21342	48.2	3725	8.4	443	1.0	732	1.7	1078	2.4
20term-N10000-s10000	863	+∞	>50.1	+∞	>50.1	6803	7.9	2491	2.9	863	1.0	2305	2.7
20term-N10000-s10001	1389	+∞	>31.1	+∞	>31.1	6404	4.6	3382	2.4	1389	1.0	2647	1.9
20term-N10000-s10002	1317	+∞	>32.8	+∞	>32.8	7494	5.7	2543	1.9	1317	1.0	2400	1.8
20term-N20000-s20000	1834	+∞	>23.6	+∞	>23.6	13429	7.3	13423	7.3	1834	1.0	4562	2.5
20term-N20000-s20001	1680	+∞	>25.7	+∞	>25.7	12763	7.6	10267	6.1	1680	1.0	4378	2.6
20term-N20000-s20002	1748	+∞	>24.7	+∞	>24.7	14868	8.5	9286	5.3	1748	1.0	5588	3.2
Fleet20-N1000-s1000	61	147	2.4	224	3.7	513	8.4	61	1.0	71	1.2	104	1.7
Fleet20-N1000-s1001	34	141	4.1	228	6.6	539	15.7	34	1.0	103	3.0	103	3.0
Fleet20-N1000-s1002	55	155	2.8	224	4.0	546	9.9	55	1.0	106	1.9	114	2.1
Fleet20-N5000-s5000	485	14769	30.5	5530	11.4	2780	5.7	933	1.9	552	1.1	485	1.0
Fleet20-N5000-s5001	331	21496	64.9	5090	15.4	2760	8.3	541	1.6	331	1.0	509	1.5
Fleet20-N5000-s5002	506	10894	21.5	5370	10.6	2730	5.4	682	1.3	535	1.1	506	1.0
Fleet20-N10000-s10000	988	+∞	>43.7	29600	30.0	5860	5.9	3540	3.6	1150	1.2	988	1.0
Fleet20-N10000-s10001	1040	+∞	>41.5	28200	27.1	5480	5.3	4750	4.6	1230	1.2	1040	1.0
Fleet20-N10000-s10002	708	+∞	>61.0	29000	41.0	5790	8.2	2950	4.2	708	1.0	984	1.4
Fleet20-N20000-s20000	2470	+∞	>17.5	+∞	>17.5	11400	4.6	14900	6.0	2470	1.0	2630	1.1
Fleet20-N20000-s20001	1490	+∞	>29.0	+∞	>29.0	11500	7.7	14100	9.5	1490	1.0	2910	2.0
Fleet20-N20000-s20002	1380	+∞	>31.3	+∞	>31.3	11000	8.0	22000	15.9	1380	1.0	2650	1.9

Table 7: Detailed results for the algorithms used as comparison basis

instance	BbB 1% Aggreg			BbB 1% Aggreg $\alpha = 0.1$		BbB 1% Aggreg $\alpha = 0.5$		BbB 1% Aggreg $\alpha = 0.9$	
	Best	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	1	2	1.8	2	1.7	1	1.0	1	1.3
LandS-N1000-s1001	1	2	1.5	1	1.4	1	1.0	1	1.1
LandS-N1000-s1002	1	2	1.8	1	1.0	1	1.1	1	1.3
LandS-N5000-s5000	4	10	2.3	7	1.7	4	1.0	8	1.8
LandS-N5000-s5001	5	9	2.0	5	1.0	6	1.2	7	1.5
LandS-N5000-s5002	4	9	1.9	7	1.6	5	1.1	4	1.0
LandS-N10000-s10000	10	17	1.7	14	1.4	10	1.0	19	1.8
LandS-N10000-s10001	9	14	1.5	9	1.0	11	1.2	10	1.1
LandS-N10000-s10002	9	17	2.0	9	1.0	12	1.4	10	1.1
LandS-N20000-s20000	24	45	1.9	30	1.2	25	1.0	24	1.0
LandS-N20000-s20001	18	42	2.4	18	1.0	20	1.1	21	1.2
LandS-N20000-s20002	20	45	2.3	30	1.5	21	1.1	20	1.0
gbd-N1000-s1000	1	2	1.6	2	1.6	1	1.0	1	1.2
gbd-N1000-s1001	1	2	2.0	1	1.9	1	1.0	1	1.2
gbd-N1000-s1002	1	2	2.1	2	1.8	1	1.0	1	1.3
gbd-N5000-s5000	5	10	2.2	8	1.6	5	1.0	5	1.1
gbd-N5000-s5001	4	8	2.2	7	1.9	4	1.2	4	1.0
gbd-N5000-s5002	5	9	2.0	7	1.6	5	1.0	7	1.6
gbd-N10000-s10000	8	18	2.2	14	1.8	8	1.0	12	1.5
gbd-N10000-s10001	9	13	1.4	15	1.7	11	1.2	9	1.0
gbd-N10000-s10002	8	14	1.9	16	2.1	8	1.0	8	1.1
gbd-N20000-s20000	16	50	3.2	16	1.0	16	1.0	20	1.3
gbd-N20000-s20001	14	31	2.2	27	1.9	14	1.0	24	1.7
gbd-N20000-s20002	16	43	2.7	30	1.9	21	1.3	16	1.0
ssn-N1000-s1000	8	14	1.6	9	1.0	8	1.0	11	1.3
ssn-N1000-s1001	8	15	1.8	9	1.0	8	1.0	12	1.4
ssn-N1000-s1002	8	13	1.6	9	1.1	8	1.0	11	1.4
ssn-N5000-s5000	51	88	1.7	51	1.0	51	1.0	76	1.5
ssn-N5000-s5001	50	90	1.8	51	1.0	50	1.0	72	1.4
ssn-N5000-s5002	51	90	1.8	52	1.0	51	1.0	74	1.5
ssn-N10000-s10000	92	175	1.9	106	1.2	92	1.0	135	1.5
ssn-N10000-s10001	96	187	1.9	105	1.1	96	1.0	141	1.5
ssn-N10000-s10002	94	193	2.0	100	1.1	94	1.0	133	1.4
ssn-N20000-s20000	187	457	2.4	212	1.1	187	1.0	297	1.6
ssn-N20000-s20001	202	458	2.3	221	1.1	202	1.0	280	1.4
ssn-N20000-s20002	197	407	2.1	213	1.1	197	1.0	316	1.6
storm-N1000-s1000	6	12	1.9	6	1.0	6	1.0	7	1.1
storm-N1000-s1001	7	12	1.8	7	1.0	7	1.1	8	1.2
storm-N1000-s1002	7	13	1.8	9	1.3	7	1.0	8	1.1
storm-N5000-s5000	32	44	1.3	33	1.0	32	1.0	44	1.4
storm-N5000-s5001	30	54	1.8	50	1.6	30	1.0	40	1.3
storm-N5000-s5002	34	58	1.7	34	1.0	35	1.1	36	1.1
storm-N10000-s10000	64	121	1.9	65	1.0	64	1.0	86	1.3
storm-N10000-s10001	66	90	1.4	66	1.0	72	1.1	69	1.0
storm-N10000-s10002	66	118	1.8	66	1.0	68	1.0	90	1.4
storm-N20000-s20000	128	216	1.7	139	1.1	128	1.0	172	1.3
storm-N20000-s20001	132	245	1.9	138	1.0	132	1.0	143	1.1
storm-N20000-s20002	128	218	1.7	133	1.0	128	1.0	150	1.2
20term-N1000-s1000	11	15	1.3	12	1.1	11	1.0	13	1.2
20term-N1000-s1001	11	15	1.3	11	1.0	11	1.0	13	1.2
20term-N1000-s1002	9	15	1.6	12	1.2	9	1.0	12	1.3
20term-N5000-s5000	53	67	1.3	72	1.4	53	1.0	61	1.1
20term-N5000-s5001	54	78	1.5	60	1.1	54	1.0	61	1.1
20term-N5000-s5002	49	64	1.3	49	1.0	53	1.1	60	1.2
20term-N10000-s10000	95	129	1.4	96	1.0	95	1.0	114	1.2
20term-N10000-s10001	112	122	1.1	165	1.5	112	1.0	127	1.1
20term-N10000-s10002	106	137	1.3	140	1.3	106	1.0	128	1.2
20term-N20000-s20000	227	261	1.1	227	1.0	243	1.1	239	1.1
20term-N20000-s20001	189	296	1.6	189	1.0	220	1.2	253	1.3
20term-N20000-s20002	230	283	1.2	265	1.2	230	1.0	239	1.0
Fleet20-N1000-s1000	17	28	1.7	21	1.2	17	1.0	20	1.2
Fleet20-N1000-s1001	17	27	1.6	18	1.1	17	1.0	20	1.2
Fleet20-N1000-s1002	18	30	1.6	20	1.1	18	1.0	21	1.1
Fleet20-N5000-s5000	78	108	1.4	87	1.1	78	1.0	88	1.1
Fleet20-N5000-s5001	74	104	1.4	98	1.3	74	1.0	85	1.1
Fleet20-N5000-s5002	75	110	1.5	86	1.1	75	1.0	85	1.1
Fleet20-N10000-s10000	151	214	1.4	177	1.2	151	1.0	164	1.1
Fleet20-N10000-s10001	154	209	1.4	175	1.1	154	1.0	171	1.1
Fleet20-N10000-s10002	150	213	1.4	178	1.2	150	1.0	164	1.1
Fleet20-N20000-s20000	312	402	1.3	409	1.3	312	1.0	336	1.1
Fleet20-N20000-s20001	301	429	1.4	396	1.3	301	1.0	337	1.1
Fleet20-N20000-s20002	321	425	1.3	431	1.3	321	1.0	367	1.1

Table 8: Detailed results for the Benders by batch algorithm with a batch size of 1%, cut aggregation, and basic stabilization or no stabilization

instance	Best	BbB 1% Aggreg		BbB 5% Aggreg		BbB 5% Aggreg $\alpha = 0.1$		BbB 5% Aggreg $\alpha = 0.5$		BbB 5% Aggreg $\alpha = 0.9$	
		time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	1	2	2.1	1	1.1	1	1.0	1	1.1	1	1.0
LandS-N1000-s1001	1	2	2.1	1	1.0	1	1.2	1	1.2	1	1.1
LandS-N1000-s1002	1	2	2.0	1	1.2	1	1.0	1	1.0	1	1.0
LandS-N5000-s5000	4	10	2.5	5	1.1	8	2.1	4	1.1	4	1.0
LandS-N5000-s5001	4	9	2.2	5	1.2	5	1.2	4	1.0	4	1.0
LandS-N5000-s5002	4	9	2.2	4	1.0	9	2.3	4	1.1	4	1.1
LandS-N10000-s10000	8	17	2.2	8	1.1	10	1.3	8	1.0	8	1.1
LandS-N10000-s10001	8	14	1.9	8	1.1	18	2.4	9	1.2	8	1.0
LandS-N10000-s10002	8	17	2.2	8	1.1	18	2.4	9	1.2	8	1.0
LandS-N20000-s20000	17	45	2.7	17	1.0	38	2.3	20	1.2	19	1.1
LandS-N20000-s20001	18	42	2.4	18	1.0	38	2.1	19	1.0	18	1.0
LandS-N20000-s20002	17	45	2.7	18	1.0	38	2.2	18	1.1	17	1.0
gbd-N1000-s1000	1	2	2.9	1	1.0	2	3.1	1	1.8	1	1.6
gbd-N1000-s1001	1	2	2.4	1	1.0	2	2.8	1	1.2	1	1.0
gbd-N1000-s1002	1	2	3.0	1	1.0	2	2.7	1	1.3	1	1.0
gbd-N5000-s5000	3	10	3.0	3	1.0	9	2.7	4	1.1	4	1.1
gbd-N5000-s5001	3	8	2.5	3	1.0	9	2.8	4	1.2	3	1.1
gbd-N5000-s5002	3	9	2.6	3	1.0	5	1.3	4	1.1	4	1.0
gbd-N10000-s10000	6	18	2.7	7	1.1	18	2.8	9	1.3	6	1.0
gbd-N10000-s10001	6	13	2.1	6	1.0	17	2.9	8	1.4	7	1.1
gbd-N10000-s10002	6	14	2.4	6	1.0	19	3.1	6	1.0	6	1.0
gbd-N20000-s20000	12	50	4.0	12	1.0	19	1.5	15	1.2	14	1.1
gbd-N20000-s20001	15	31	2.1	15	1.0	36	2.4	17	1.1	17	1.1
gbd-N20000-s20002	14	43	3.2	14	1.0	37	2.7	18	1.3	16	1.2
ssn-N1000-s1000	14	14	1.0	63	4.6	15	1.1	18	1.3	36	2.6
ssn-N1000-s1001	15	15	1.0	63	4.3	15	1.0	20	1.3	42	2.9
ssn-N1000-s1002	13	13	1.0	59	4.6	15	1.2	20	1.5	41	3.2
ssn-N5000-s5000	84	88	1.0	337	4.0	84	1.0	106	1.3	221	2.6
ssn-N5000-s5001	82	90	1.1	322	4.0	82	1.0	114	1.4	225	2.8
ssn-N5000-s5002	90	90	1.0	308	3.4	94	1.0	112	1.2	224	2.5
ssn-N10000-s10000	175	175	1.0	672	3.8	181	1.0	240	1.4	481	2.7
ssn-N10000-s10001	181	187	1.0	760	4.2	181	1.0	246	1.4	493	2.7
ssn-N10000-s10002	179	193	1.1	690	3.9	179	1.0	226	1.3	491	2.7
ssn-N20000-s20000	397	457	1.2	1651	4.2	397	1.0	528	1.3	1069	2.7
ssn-N20000-s20001	418	458	1.1	1651	3.9	418	1.0	559	1.3	1076	2.6
ssn-N20000-s20002	388	407	1.0	1543	4.0	388	1.0	561	1.4	1051	2.7
storm-N1000-s1000	6	12	1.9	6	1.0	7	1.2	6	1.0	6	1.0
storm-N1000-s1001	6	12	2.0	6	1.1	8	1.3	6	1.0	6	1.0
storm-N1000-s1002	6	13	2.2	6	1.1	10	1.7	6	1.0	6	1.0
storm-N5000-s5000	29	44	1.5	33	1.2	38	1.3	32	1.1	29	1.0
storm-N5000-s5001	27	54	2.0	33	1.2	36	1.3	36	1.3	27	1.0
storm-N5000-s5002	30	58	2.0	37	1.3	37	1.3	30	1.0	30	1.0
storm-N10000-s10000	62	121	2.0	73	1.2	78	1.3	66	1.1	62	1.0
storm-N10000-s10001	62	90	1.5	76	1.2	77	1.2	63	1.0	62	1.0
storm-N10000-s10002	62	118	1.9	73	1.2	112	1.8	62	1.0	62	1.0
storm-N20000-s20000	126	216	1.7	167	1.3	180	1.4	126	1.0	126	1.0
storm-N20000-s20001	125	245	2.0	161	1.3	152	1.2	127	1.0	125	1.0
storm-N20000-s20002	125	218	1.7	160	1.3	171	1.4	133	1.1	125	1.0
20term-N1000-s1000	15	15	1.0	36	2.5	15	1.0	21	1.4	28	1.9
20term-N1000-s1001	14	15	1.0	37	2.7	14	1.0	19	1.3	29	2.1
20term-N1000-s1002	14	15	1.0	37	2.6	14	1.0	21	1.5	32	2.3
20term-N5000-s5000	67	67	1.0	199	3.0	92	1.4	104	1.6	148	2.2
20term-N5000-s5001	78	78	1.0	197	2.5	81	1.0	99	1.3	154	2.0
20term-N5000-s5002	64	64	1.0	182	2.8	83	1.3	93	1.4	141	2.2
20term-N10000-s10000	129	129	1.0	411	3.2	148	1.1	238	1.8	305	2.4
20term-N10000-s10001	122	122	1.0	409	3.3	165	1.3	218	1.8	345	2.8
20term-N10000-s10002	137	137	1.0	388	2.8	176	1.3	204	1.5	353	2.6
20term-N20000-s20000	261	261	1.0	860	3.3	398	1.5	483	1.9	768	2.9
20term-N20000-s20001	296	296	1.0	985	3.3	302	1.0	517	1.8	780	2.6
20term-N20000-s20002	283	283	1.0	897	3.2	323	1.1	509	1.8	806	2.8
Fleet20-N1000-s1000	24	28	1.2	42	1.7	24	1.0	24	1.0	32	1.3
Fleet20-N1000-s1001	24	27	1.1	40	1.7	24	1.0	24	1.0	32	1.4
Fleet20-N1000-s1002	21	30	1.4	43	2.0	21	1.0	26	1.2	34	1.6
Fleet20-N5000-s5000	108	108	1.0	218	2.0	114	1.1	125	1.2	160	1.5
Fleet20-N5000-s5001	104	104	1.0	209	2.0	119	1.1	123	1.2	162	1.6
Fleet20-N5000-s5002	110	110	1.0	205	1.9	124	1.1	122	1.1	161	1.5
Fleet20-N10000-s10000	214	214	1.0	426	2.0	253	1.2	249	1.2	333	1.6
Fleet20-N10000-s10001	209	209	1.0	467	2.2	247	1.2	261	1.2	342	1.6
Fleet20-N10000-s10002	213	213	1.0	426	2.0	246	1.2	250	1.2	326	1.5
Fleet20-N20000-s20000	402	402	1.0	886	2.2	557	1.4	545	1.4	677	1.7
Fleet20-N20000-s20001	429	429	1.0	856	2.0	493	1.1	513	1.2	700	1.6
Fleet20-N20000-s20002	425	425	1.0	885	2.1	516	1.2	517	1.2	684	1.6

Table 9: Detailed results for the Benders by batch algorithm with a batch size of 5%, cut aggregation, and basic stabilization or no stabilization

instance	Best	BbB 1% Aggreg		BbB 1% Aggreg $\alpha = 0.1, \beta = 0.1$		BbB 1% Aggreg $\alpha = 0.1, \beta = 0.5$		BbB 1% Aggreg $\alpha = 0.1, \beta = 0.9$		BbB 1% Aggreg $\alpha = 0.5, \beta = 0.1$		BbB 1% Aggreg $\alpha = 0.5, \beta = 0.5$		BbB 1% Aggreg $\alpha = 0.5, \beta = 0.9$		BbB 1% Aggreg $\alpha = 0.9, \beta = 0.1$		BbB 1% Aggreg $\alpha = 0.9, \beta = 0.5$		BbB 1% Aggreg $\alpha = 0.9, \beta = 0.9$	
		time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	1	2	2.0	1	1.8	2	1.8	1	1.0	1	1.2	1	1.1	1	1.8	1	1.4	1	1.1	1	1.7
LandS-N1000-s1001	1	2	1.7	2	1.7	2	1.6	2	1.8	1	1.1	1	1.0	2	1.6	1	1.2	1	1.1	2	1.7
LandS-N1000-s1002	1	2	2.1	1	1.0	1	1.0	1	1.1	1	1.0	1	1.1	1	1.1	1	1.5	1	1.1	1	1.1
LandS-N5000-s5000	4	10	2.3	7	1.7	7	1.7	4	1.0	5	1.1	5	1.1	7	1.7	6	1.3	5	1.1	7	1.6
LandS-N5000-s5001	4	9	2.3	4	1.1	4	1.0	5	1.2	5	1.2	5	1.2	4	1.0	6	1.5	5	1.3	4	1.1
LandS-N5000-s5002	4	9	1.9	7	1.7	7	1.6	8	1.7	6	1.2	4	1.0	7	1.7	4	1.0	5	1.2	7	1.7
LandS-N10000-s10000	9	17	1.9	9	1.0	15	1.7	15	1.8	10	1.1	10	1.1	15	1.7	10	1.1	10	1.1	9	1.0
LandS-N10000-s10001	10	14	1.4	15	1.5	16	1.5	16	1.5	11	1.0	16	1.5	15	1.5	10	1.0	11	1.0	16	1.5
LandS-N10000-s10002	9	17	2.0	9	1.0	9	1.1	15	1.7	11	1.3	9	1.1	9	1.1	10	1.2	11	1.3	9	1.0
LandS-N20000-s20000	19	45	2.4	31	1.6	32	1.7	19	1.0	21	1.1	21	1.1	31	1.6	25	1.3	21	1.1	31	1.6
LandS-N20000-s20001	17	42	2.4	31	1.8	33	1.9	17	1.0	20	1.2	23	1.3	32	1.9	24	1.4	20	1.2	31	1.8
LandS-N20000-s20002	17	45	2.7	30	1.8	30	1.8	30	1.8	21	1.2	17	1.0	31	1.8	18	1.1	22	1.3	31	1.8
gbd-N1000-s1000	1	2	1.8	2	1.8	2	1.8	2	2.3	1	1.1	1	1.1	2	1.8	1	1.6	1	1.0	2	1.8
gbd-N1000-s1001	1	2	2.0	2	2.0	2	2.0	2	2.3	1	1.1	1	1.1	2	1.9	1	1.3	1	1.0	2	2.2
gbd-N1000-s1002	1	2	2.3	1	1.9	2	2.0	2	2.4	1	1.0	1	1.0	2	2.0	1	1.3	1	1.0	1	1.9
gbd-N5000-s5000	4	10	2.3	7	1.6	8	1.8	9	2.0	4	1.0	5	1.2	8	1.8	6	1.5	4	1.0	7	1.7
gbd-N5000-s5001	4	8	2.3	7	2.1	8	2.1	8	2.3	5	1.3	4	1.1	8	2.1	4	1.0	5	1.4	7	2.1
gbd-N5000-s5002	4	9	2.6	8	2.4	8	2.4	8	2.3	6	1.6	5	1.5	8	2.3	4	1.0	6	1.6	8	2.4
gbd-N10000-s10000	8	18	2.1	15	1.8	15	1.8	16	2.0	10	1.2	9	1.0	15	1.8	8	1.0	10	1.2	15	1.8
gbd-N10000-s10001	8	13	1.6	13	1.7	14	1.8	17	2.1	11	1.3	8	1.0	14	1.8	10	1.3	10	1.3	14	1.7
gbd-N10000-s10002	7	14	2.0	14	2.0	14	2.0	19	2.7	8	1.2	7	1.1	15	2.1	7	1.0	8	1.2	14	2.0
gbd-N20000-s20000	14	50	3.5	32	2.2	30	2.1	17	1.2	15	1.1	16	1.1	30	2.1	14	1.0	15	1.1	32	2.2
gbd-N20000-s20001	17	31	1.8	28	1.7	29	1.7	30	1.8	18	1.1	19	1.1	30	1.8	17	1.0	18	1.1	29	1.7
gbd-N20000-s20002	16	43	2.6	30	1.8	29	1.8	32	2.0	20	1.2	18	1.1	30	1.8	16	1.0	20	1.2	30	1.8
ssn-N1000-s1000	8	14	1.7	9	1.1	10	1.2	11	1.3	9	1.1	9	1.1	11	1.3	10	1.2	8	1.0	9	1.1
ssn-N1000-s1001	8	15	1.9	12	1.6	9	1.2	11	1.4	9	1.2	8	1.1	10	1.3	12	1.6	8	1.0	10	1.3
ssn-N1000-s1002	8	13	1.6	9	1.1	9	1.1	11	1.4	8	1.0	8	1.0	10	1.2	11	1.3	8	1.0	9	1.1
ssn-N5000-s5000	45	88	2.0	54	1.2	52	1.2	56	1.3	47	1.0	45	1.0	54	1.2	64	1.4	46	1.0	54	1.2
ssn-N5000-s5001	46	90	2.0	49	1.1	52	1.1	60	1.3	47	1.0	46	1.0	53	1.2	62	1.3	46	1.0	49	1.1
ssn-N5000-s5002	46	90	2.0	50	1.1	52	1.1	58	1.3	52	1.1	46	1.0	52	1.1	61	1.3	48	1.1	52	1.1
ssn-N10000-s10000	92	175	1.9	101	1.1	108	1.2	120	1.3	92	1.0	95	1.0	113	1.2	115	1.3	92	1.0	106	1.1
ssn-N10000-s10001	93	187	2.0	112	1.2	111	1.2	128	1.4	93	1.0	105	1.1	106	1.1	119	1.3	93	1.0	106	1.2
ssn-N10000-s10002	86	193	2.2	108	1.3	107	1.2	123	1.4	93	1.1	86	1.0	112	1.3	115	1.3	88	1.0	101	1.2
ssn-N20000-s20000	183	457	2.5	242	1.3	235	1.3	270	1.5	203	1.1	183	1.0	232	1.3	244	1.3	198	1.1	213	1.2
ssn-N20000-s20001	182	458	2.5	232	1.3	228	1.3	265	1.5	182	1.0	186	1.0	230	1.3	259	1.4	186	1.0	221	1.2
ssn-N20000-s20002	190	407	2.1	215	1.1	228	1.2	255	1.3	190	1.0	200	1.1	235	1.2	251	1.3	193	1.0	226	1.2
storm-N1000-s1000	6	12	1.9	10	1.5	10	1.6	7	1.1	8	1.3	6	1.0	10	1.6	7	1.1	7	1.0	10	1.6
storm-N1000-s1001	6	12	2.0	7	1.1	10	1.6	7	1.2	6	1.0	7	1.2	9	1.6	7	1.2	8	1.3	7	1.1
storm-N1000-s1002	6	13	2.0	10	1.5	10	1.5	7	1.0	7	1.1	8	1.2	10	1.5	7	1.1	6	1.0	10	1.5
storm-N5000-s5000	31	44	1.4	32	1.0	33	1.1	36	1.2	31	1.0	37	1.2	33	1.1	35	1.1	31	1.0	31	1.0
storm-N5000-s5001	32	54	1.7	47	1.5	34	1.1	35	1.1	42	1.3	32	1.0	33	1.0	33	1.0	32	1.0	48	1.5
storm-N5000-s5002	32	58	1.8	44	1.1	32	1.0	33	1.1	32	1.0	33	1.0	32	1.0	37	1.2	32	1.0	34	1.1
storm-N10000-s10000	59	121	2.0	67	1.1	67	1.1	109	1.8	64	1.1	68	1.1	68	1.1	62	1.0	59	1.0	67	1.1
storm-N10000-s10001	65	90	1.4	68	1.1	66	1.0	108	1.7	67	1.0	66	1.0	67	1.0	71	1.1	65	1.0	68	1.0
storm-N10000-s10002	62	118	1.9	67	1.1	101	1.6	70	1.1	69	1.1	64	1.0	100	1.6	62	1.0	66	1.1	67	1.1
storm-N20000-s20000	127	216	1.7	139	1.1	138	1.1	144	1.1	130	1.0	127	1.0	136	1.1	152	1.2	131	1.0	139	1.1
storm-N20000-s20001	123	245	2.0	140	1.1	129	1.0	146	1.2	130	1.1	123	1.0	128	1.0	137	1.1	126	1.0	141	1.1
storm-N20000-s20002	130	218	1.7	130	1.0	135	1.0	143	1.1	141	1.1	135	1.0	133	1.0	192	1.5	152	1.2	131	1.0
20term-N1000-s1000	9	15	1.7	14	1.6	12	1.4	16	1.8	9	1.0	10	1.2	10	1.1	12	1.4	11	1.3	15	1.7
20term-N1000-s1001	10	15	1.5	15	1.6	17	1.7	18	1.9	11	1.2	11	1.2	16	1.7	12	1.3	10	1.0	10	1.0
20term-N1000-s1002	9	15	1.6	18	2.0	12	1.4	22	2.4	11	1.2	11	1.2	14	1.5	12	1.3	9	1.0	16	1.8
20term-N5000-s5000	51	67	1.3	60	1.2	67	1.3	84	1.6	51	1.0	57	1.1	51	1.0	58	1.1	66	1.3	61	1.2
20term-N5000-s5001	46	78	1.7	67	1.5	67	1.5	84	1.8	51	1.1	46	1.0	74	1.6	58	1.3	48	1.1	74	1.6
20term-N5000-s5002	45	64	1.4	70	1.5	69	1.5	117	2.6	56	1.2	45	1.0	65	1.4	55	1.2	53	1.2	68	1.5
20term-N10000-s10000	101	129	1.3	118	1.2	147	1.5	188	1.9	101	1.0	101	1.0	135	1.3	101	1.0	102	1.0	113	1.1
20term-N10000-s10001	91	122	1.3	135	1.5	139	1.5	152	1.7	115	1.3	91	1.0	151	1.7	115	1.3	110	1.2	179	2.0
20term-N10000-s10002	85	137	1.6	126	1.5	126	1.5	169	2.0	87	1.0	85	1.0	102	1.2	136	1.6	101	1.2	176	2.1
20term-N20000-s20000	193	261	1.3	193	1.0	330	1.7	367	1.9	226	1.2	244	1.3	361	1.9	251	1.3	222	1.2	276	1.4
20term-N20000-s20001	224	296	1.3	289	1.3	337	1.5	326	1.5	241	1.1	243	1.1	272	1.2	236	1.1	224	1.0	267	1.2
20term-N20000-s20002	178	283	1.6	237	1.3	178	1.0	361	2.0	233	1.3	21									

