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, Aurelien 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

Xavier Blanchot[1,2], François Clautiaux[1], Boris Detienne[1], Aurélien Froger[1]
Manuel Ruiz[2]

[1] Université de Bordeaux, CNRS, INRIA, Bordeaux INP, IMB, UMR 5251, 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. The key idea is to solve at most iterations only a small proportion of the subproblems by detecting as soon as possible that a first-stage candidate solution cannot be proven optimal. 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 nine 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, telecommunication 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 such problems are based on decomposition techniques (Ruszczyński, 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. We consider the following linear program with a scenario block structure:

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

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), $g_s \in \mathbb{R}^{n_2}$, $W_s \in \mathbb{R}^{m\times n_2}$, $T_s \in \mathbb{R}^{m\times n_1}$, $d_s \in \mathbb{R}^m$, and $X \subset \mathbb{R}^{n_1}$ 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.

Email addresses: 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@math.u-bordeaux.fr (Aurélien Froger), manuel.ruiz@rte-france.com (Manuel Ruiz)

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

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

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

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

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 $\Pi_s = \{\pi \in \mathbb{R}^m | W_s^\top \pi \leqslant g_s\}$ 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 $dom\Big(\phi(\cdot,s)\Big) = \{x \in \mathbb{R}^{n_1} | r_s^\top (d_s - T_s x) \leqslant 0,\ \forall r_s \in \text{Rays}(\Pi_s)\}$. Then we can replace in formulation (2) the polyhedral mapping $x \mapsto \phi(x,s)$ 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):

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

Constraints (i) are called optimality cuts, and constraints (ii), feasibility cuts. Without loss of generality, we assume that the problem has *relatively complete recourse* (i.e., $X \subset dom\,(\phi(\cdot,s))$ 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 to a problem satisfying this hypothesis by introducing slack variables with large enough coefficients in the objective function (see e.g. (Bodur and Luedtke, 2022) or (Shapiro and Nemirovski, 2005)).

The classic multicut Benders decomposition algorithm (see Algorithm 1 in the case of relatively complete recourse) consists of 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 $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$.

Algorithm 1: Classic multicut Benders decomposition algorithm

Parameters: $\epsilon \geqslant 0$ the selected optimality gap
Initialization: $k \leftarrow 0$, $UB^{(0)} \leftarrow +\infty$, $LB^{(0)} \leftarrow -\infty$
while $UB^{(k)} > LB^{(k)} + \epsilon$ **do**
    $k \leftarrow k+1$
    Solve $(RMP)^{(k)}$ and retrieve $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$
    $LB^{(k)} \leftarrow c^\top \check{x}^{(k)} + \sum_{s\in S} p_s \check{\theta}_s^{(k)}$
    for $s \in S$ **do**
        Solve $(SP(\check{x}^{(k)}, s))$ and retrieve $\pi_s \in \text{Vert}(\Pi_s)$
        Add $\theta_s \geqslant \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
    $UB^{(k)} \leftarrow min\Big(UB^{(k-1)}, c^\top \check{x}^{(k)} + \sum_{s\in S} p_s \pi_s^\top (d_s - T_s \check{x}^{(k)})\Big)$
    $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$
Return $\check{x}^{(k)}$

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 that a solution cannot be proven optimal at the current iteration without necessarily having to solve all the subproblems. As a result, only a 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 partial 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 classic Benders decomposition algorithm (multicut or monocut, depending on the use of cut aggregation). Several existing methods based on similar ideas require fixed recourse ($W_s = W,\ \forall s \in S$ in problem (1)) (Oliveira et al., 2011) and deterministic second-stage objective function ($g_s = g,\ \forall s \in s$ in problem (1)) (Wets, 1983; Dantzig and Infanger, 1991; Higle and Sen, 1991). Moreover, some of them do not have finite convergence (Higle and Sen, 1991), or are not exact (Dantzig and Infanger, 1991). The method proposed in this work is exact, has finite convergence, and does not require any assumption on the value of the random parameters g_s, W_s, d_s, T_s in problem (1).

We also show how to stabilize the proposed algorithm. As the classical primal stabilization methods of the literature (Ben-Ameur and Neto, 2007; Lemaréchal et al., 1995) are designed for algorithms which solve all the subproblems at each iteration, it is not possible to apply them directly. They require the actual value of the recourse function at each iteration, at least to evaluate their stopping criterion. We therefore propose a generic framework to stabilize the Benders by batch algorithm and prove the finite convergence and exact behavior of the stabilized algorithm. Our algorithm is also compatible with classical dual stabilization techniques (Magnanti and Wong, 1981; Papadakos, 2008; Sherali and Lunday, 2013).

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 stochastic programs with finite probability distribution. 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 that a first-stage candidate solution cannot be proven optimal early in the subproblems' solution process.
- We develop a general framework to apply primal stabilization to the Benders by batch algorithm, as classical primal stabilization methods cannot be applied if all the subproblems are not solved at each iteration. We state sufficient conditions for the stabilized algorithm to be exact and have finite convergence and provide two effective primal stabilization schemes.
- We perform an extensive numerical study showing the efficiency of the developed algorithm on some classical stochastic instances from the literature compared to implementations of the classic monocut and multicut Benders decomposition algorithm, with and without in-out stabilization, the static multicut aggregation approach of Trukhanov et al. (2010), 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 classic 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 systematically solving all the subproblems.

A well-known downside of cutting-plane methods, and therefore of the Benders decomposition algorithm, is the oscillation of the first-stage variables (Nesterov, 2004; Pessoa et al., 2013). 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 to avoid the presented drawbacks of cutting-plane methods 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 to points 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 infinity 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 efficiency 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 one 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 decision between using either the monocut or the 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 model and a monocut model. When, for a given first-stage solution x, they observe that the monocut approximation of the recourse function is substantially lower than the multicut approximation, they aggregate the active cuts from the multicut model to generate a cut in the monocut one. As this cut has, at x, the value given by the multicut model, this cut improves the monocut approximation, without having to solve any subproblem. They embed their algorithm in the general concept of *oracles with on-demand accuracy* (de Oliveira and Sagastizábal, 2014). The concept of oracles with on-demand accuracy might embed the core idea of the Benders by batch algorithm presented in this work. 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.

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 solving 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. Higle 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 only requires the fixed recourse hypothesis ($W_s = W$, $\forall s \in S$). 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 Higle and Sen (1991) are designed for a monocut algorithm, but the method of Oliveira et al. (2011) can be adapted to a multicut algorithm.

Finally, among other techniques used to accelerate the solution time of 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 classic 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	Cut aggregation	Stabilization
(Crainic et al., 2020)	$g_s = g, W_s = W \; \forall s \in S$	Yes	Both	Yes	Yes	No	No
(Song and Luedtke, 2015)	$g_s = g, W_s = W \; \forall s \in S$	Yes	Not applicable	Yes	Yes	No	No
(van Ackooij et al., 2017)	$g_s = g, W_s = W \; \forall s \in S$	No	Both	Yes	Yes	No	Level
(Wets, 1983)	$g_s = g, W_s = W \; \forall s \in S$	No	Both	Yes	Yes	No	No
(Dantzig and Infanger, 1991)	$g_s = g, W_s = W \; \forall s \in S$	No	Monocut	No	Yes	No	No
(Higle and Sen, 1991)	$g_s = g, W_s = W \; \forall s \in S$	No	Monocut	Yes	No	No	No
(Trukhanov et al., 2010)	No	Yes	Multicut	Yes	Yes	Yes	No
(Linderoth and Wright, 2003)	No	Yes	Multicut	Yes	Yes	No	Trust-region
(Wolf et al., 2014)	No	All or none	Monocut and Multicut	Yes	Yes	No	Level
(Oliveira et al., 2011)	$W_s = W \; \forall s \in S$	No	Monocut	Inexact	Yes	No	Proximal bundle
This work	No	No	Multicut	Yes	Yes	Yes	In-out

* in addition to random parameters having a discrete finite probability distribution

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 of solving the subproblems by batch and stopping solving subproblems at an iteration as soon as we identify that the current first-stage solution cannot be proven optimal. This is made possible by checking, after solving of a subset of subproblems, if the gap between their optimal values and their epigraph approximations in the relaxed master program already exceeds the optimality gap.

We first present some notations necessary to formally describe the algorithm. We consider an ordered set of scenarios $S = \{s_1, s_2, ..., s_{card(S)}\}$ and a given batch size $1 \leqslant \eta \leqslant card(S)$. We define $\kappa = \lceil card(S)/\eta \rceil$ as the number of batches of subproblems. For every $i \in [\![1, \kappa]\!]$, the i^{th} batch of subproblems S_i is defined as $S_i = \{s_{(i-1)\eta+1}, ..., s_{(i-1)\eta+\eta_i}\}$, where η_i is the size of batch i, $\eta_1 = \cdots = \eta_{\kappa-1} = \eta$ and $\eta_\kappa = (card(S) \mod \eta)$. Family $(S_i)_{i \in [\![1,\kappa]\!]}$ 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 $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$ the optimal solution to $(RMP)^{(k)}$ at iteration k of the algorithm, where $\check{x}^{(k)}$ denotes the optimal value to the first-stage variables and $\check{\theta}_s^{(k)}$ the optimal value to the epigraph variable associated with 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 $\epsilon \geqslant 0$ be the optimality gap of the algorithm. We first define the notion of *provable optimality* in cutting-planes methods.

Definition 1. *Let $\epsilon \geqslant 0$ be the optimality gap of the algorithm and k an iteration of the algorithm. We say that a first-stage solution $x \in X$ cannot be proven optimal at an iteration k of the algorithm iff $UB(x) - LB^{(k)} > \epsilon$.*

Saying that a first-stage solution x cannot be proven optimal at an iteration k of the algorithm means that, either x is not an optimal solution to problem (1), or the current lower bound given by $(RMP)^{(k)}$ is too low to prove the optimality of an optimal solution. The classical stopping criterion $UB - LB \leqslant \epsilon$ of the Benders decomposition algorithm is based on such an optimality proof, but cannot be directly applied if not all the subproblems are 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 hereafter a new stopping criterion, which detects, when it occurs, that the current first-stage solution $\check{x}^{(k)}$ to $(RMP)^{(k)}$ cannot be proven optimal without necessarily having to solve all the subproblems. If after having solved some batches of subproblems, the sum of the differences between their value and their epigraph approximation in $(RMP)^{(k)}$ already exceeds the optimality gap ϵ, the algorithm does not solve the remaining batches of subproblems, as we already know that $\check{x}^{(k)}$ cannot be proven optimal (see Proposition 1). In this way, the Benders by batch algorithm is likely to explore more first-stage solutions than classic Benders decomposition algorithms as it tends to solve only a few subproblems at most iterations. The proposed stopping criterion is based on the concept of ϵ_i-approximation that we define below.

Definition 2 (ϵ_i-approximation)**.** *Let $\epsilon \geqslant 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration and σ a permutation of $[\![1, \kappa]\!]$. For every $i \in [\![1, \kappa]\!]$, we say that batch $S_{\sigma(i)}$ is ϵ_i-approximated by $(RMP)^{(k)}$ if*

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

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

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 $i \in [\![2, \kappa]\!]$, we have $\epsilon_i = \epsilon_{i-1} - \sum_{s \in S_{\sigma(i-1)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right)$, 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 $\epsilon \geqslant 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration of the algorithm, and σ a permutation of $[\![1, \kappa]\!]$. 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 $i \in [\![1, \kappa]\!]$.*

Proof of proposition 1. See Appendix A.1 □

Corollary 1. *Let $\epsilon \geqslant 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration, and σ a permutation of $[\![1, \kappa]\!]$. If there exists an index $i \in [\![1, \kappa]\!]$ such that $\sum_{s \in S_{\sigma(i)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) > \epsilon_i$, then $\check{x}^{(k)}$ cannot be proven optimal.*

Remark 1. *As stated in Proposition 1, the proposed stopping criterion is equivalent to the classical stopping criterion $UB - LB \leqslant \epsilon$. This means that, given a relaxed master program with some Benders cuts, and a first-stage solution $\check{x}$, either $\check{x}$ can be proven optimal by both stopping criteria, or both will reject it and let the algorithm continue.*

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 $\check{x}^{(k)}$. A permutation σ of $[\![1, \kappa]\!]$ is then chosen. This permutation defines the order in which the batches of subproblems $(S_1, S_2, ..., S_\kappa)$ 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 classic 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 `cutAggr` 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 classic 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 $\check{x}^{(k)}$ is

Algorithm 2: The Benders by batch algorithm

Parameters: $\epsilon \geqslant 0$, $\eta \in [\![1, card(S)]\!]$ the batch size, `cutAggr` $\in \{$`True`, `False`$\}$

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

2 Define a partition $\left(S_i\right)_{i\in[\![1,\kappa]\!]}$ 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 $\left(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S}\right)$

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

7   Choose a permutation σ of $[\![1, \kappa]\!]$

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

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

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

11   **if** `cutAggr` **then**

12   Add $\sum\limits_{s\in S_{\sigma(i)}} p_s\theta_s \geqslant \sum\limits_{s\in S_{\sigma(i)}} p_s\left(\pi_s^\top(d_s - T_s x)\right)$ to $(RMP)^{(k)}$

13   **else**

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

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

16   **if** $\sum\limits_{s\in S_{\sigma(i)}} p_s\left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)}\right) \leqslant \epsilon_i$ **then**

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

18     $i \leftarrow i + 1$

19   **else** `stay_at_x` $\leftarrow$ `False`

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

21 Return $\check{x}^{(k)}$

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 proven optimal. Then there exists at least one of the cuts which excludes the solution $\left(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S}\right)$ 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 (Partial cut aggregation)**.** *One of the most important drawbacks 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* $\sum\limits_{s\in S_{\sigma(i)}} p_s\theta_s \geqslant \sum\limits_{s\in S_{\sigma(i)}} p_s\left(\pi_s^\top(d_s - T_s x)\right)$*. 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. As the partition of the subproblems into batches is done prior to the algorithm, the cuts of the same subproblems are always aggregated together. This can be seen as the static cut aggregation strategy used in (Trukhanov et al., 2010).*

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

Proposition 2. *Let $\epsilon \geqslant 0$ be the optimality gap. The Benders by batch algorithm converges to an optimal solution to problem 1 in a finite number of iterations.*

Proof of proposition 2. See Appendix A.2. ☐

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 $S_1, S_2, ..., S_\kappa$ and $\sigma = id$ at the first iteration. When we choose a new permutation, at the beginning of a master loop, the *ordered strategy* consists of 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 of 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 implies solving all the subproblems the same number of times 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 may compute, in the early iterations, cuts that exclude solutions that are far away from the optimal solution (see e.g. (Vanderbeck, 2005) section 7). However, the classical primal stabilization procedures presented in Section 2 do not apply directly if we do not solve all the subproblems at each iteration as they require the value of the recourse function for the current first-stage solution. 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 primal stabilization methods for cutting-plane algorithms solve, at each iteration, a separation problem in a point $x^{(k)}$ (hereafter referred to as the separation point) that is different from the current optimal first-stage solution $\check{x}^{(k)}$ to the relaxed master program (Zverovich et al., 2012; Pessoa et al., 2013). We define hereafter formally a primal stabilization scheme, in which the separation point is computed as the image by a given mapping of a vector defining the state of the stabilization. Such a scheme must also incorporate a way to update this state vector.

Definition 3 (Primal stabilization scheme)**.** *A primal stabilization scheme is characterized by a triplet $(\mathcal{D}, \psi_1, \psi_2)$ where $\mathcal{D}$ is a stabilization state space and (ψ_1, ψ_2) is a pair of mappings* $\begin{cases} \psi_1 : X \times \mathcal{D} \to \mathcal{D} \\ \psi_2 : \mathcal{D} \to X \end{cases}$ *such that ψ_2 is surjective.*

At an iteration k of the stabilized algorithm, mapping ψ_1 computes the state vector of the stabilization to be used at the current iteration from the precedent state vector and the optimal solution to the current relaxed master program. This state vector may contain some elements of X, such as the last optimal solution to the relaxed master program. An initial stabilization state vector $d^0 \in \mathcal{D}$ is required when using the primal stabilization scheme in the first iteration of our algorithm. From the current stabilization state vector, mapping ψ_2 is then responsible for generating a first-stage solution $x^{(k)}$ at which the subproblems are solved and cuts are generated. Function ψ_2 is required to be surjective to ensure that every first-stage solution can be separated.

We now present how to adapt the Benders by batch algorithm (Algorithm 2) when such a primal stabilization scheme is used. We generalize Definition 2 and Proposition 1 to take into account that the lower bound at a given iteration k is computed based on the current optimal solution $\check{x}^{(k)}$ to RMP, while the subproblems are solved at a separation point x that is usually different from $\check{x}^{(k)}$. As this difference between the first-stage solutions induces a difference in the first-stage cost, we subtract in the definition of the remaining gap ϵ_i the difference $c^\top(x - \check{x}^{(k)})$. Because $\check{\theta}_s^{(k)}$ is a lower bound on $\phi\left(\check{x}^{(k)}, s\right)$, but not on $\phi(x, s)$, we also need to account for cases where $\phi(x, s) - \check{\theta}_s^{(k)} < 0$.

Definition 4 ($\epsilon_i(x)$-approximation at a first-stage solution x)**.** *Let $\epsilon \geqslant 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration and σ a permutation of $[\![1, \kappa]\!]$. For every $i \in [\![1, \kappa]\!]$, we say that batch $S_{\sigma(i)}$ is $\epsilon_i(x)$-approximated by $(RMP)^{(k)}$ at $x \in X$ if*

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

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

Remark 3. *Saying that a batch $S_{\sigma(i)}$ is $\epsilon_i(\check{x}^{(k)})$-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 $\epsilon \geqslant 0$ 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 $S_{\sigma(i)}$ is $\epsilon_i(x)$-approximated by $(RMP)^{(k)}$, for all $i \in [\![1, \kappa]\!]$, then x is an optimal solution to problem* (1).

Proof of proposition 3. See Appendix A.3 ☐

Algorithm 3: The stabilized Benders by batch algorithm

Parameters: $\epsilon \geqslant 0$, $\eta \in [\![1, card(S)]\!]$ the batch size, `cutAggr` $\in$ {`True`, `False`}, a primal stabilization scheme $(\mathcal{D}, \psi_1, \psi_2)$ and an initial stabilization state vector $d^{(0)} \in \mathcal{D}$.
Initialization: $i \leftarrow 1$, $k \leftarrow 0$, `misprice` $\leftarrow$ `False`, `stay_at_x` $\leftarrow$ `True`
Define a partition $\left(S_i\right)_{i \in [\![1,\kappa]\!]}$ of the subproblems according to batch size η
while $i < \kappa + 1$ **do**
  Solve $(RMP)^{(k+1)}$ and retrieve $\left(\check{x}^{(k+1)}, (\check{\theta}_s^{(k+1)})_{s \in S}\right)$
  do
    $k \leftarrow k + 1$
    $d^{(k)} \leftarrow \psi_1(\check{x}^{(k)}, d^{(k-1)})$
    $x^{(k)} \leftarrow \psi_2(d^{(k)})$
    $i \leftarrow 1$, $\epsilon_i \leftarrow \epsilon - c^\top(x^{(k)} - \check{x}^{(k)})$, `stay_at_x` $\leftarrow$ `True`
    Choose a permutation σ of $[\![1, \kappa]\!]$
    `misprice` $\leftarrow$ `True`
    while `stay_at_x` = `True` and $i < \kappa + 1$ **do**
      for $s \in S_{\sigma(i)}$ **do**
        Solve $(SP(x^{(k)}, s))$ and retrieve $\phi(x^{(k)}, s)$ and $\pi_s \in \text{Vert}(\Pi_s)$
      if `cutAggr` **then**
        Add $\sum_{s \in S_{\sigma(i)}} p_s \theta_s \geqslant \sum_{s \in S_{\sigma(i)}} p_s \left(\pi_s^\top (d_s - T_s x)\right)$ to $(RMP)^{(k)}$
      else
        for $s \in S_{\sigma(i)}$ **do**
          Add $\theta_s \geqslant \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
      if $\sum_{s \in S_{\sigma(i)}} \left[p_s \left(\phi(x^{(k)}, s) - \check{\theta}_s^{(k)}\right)\right]^+ \leqslant \epsilon_i$ **then**
        $\epsilon_{i+1} \leftarrow \epsilon - c^\top(x^{(k)} - \check{x}^{(k)}) - \left[\sum_{t=1}^{i} \sum_{s \in S_{\sigma(t)}} p_s \left(\phi(x^{(k)}, s) - \check{\theta}_s^{(k)}\right)\right]^+$
        $i \leftarrow i + 1$
      else
        `stay_at_x` $\leftarrow$ `False`
      if `cutAggr` **then**
        if $\sum_{s \in S_{\sigma(i)}} p_s \check{\theta}_s^{(k)} < \sum_{s \in S_{\sigma(i)}} p_s \left(\pi_s^\top (d_s - T_s \check{x}^{(k)})\right)$ **then** `misprice` $\leftarrow$ `False`
      else
        for $s \in S_{\sigma(i)}$ **do**
          if $\check{\theta}_s^{(k)} < \pi_s^\top (d_s - T_s \check{x}^{(k)})$ **then** `misprice` $\leftarrow$ `False`
    $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$, $\check{x}^{(k+1)} \leftarrow \check{x}^{(k)}$, $(\check{\theta}_s^{(k+1)})_{s \in S} \leftarrow (\check{\theta}_s^{(k)})_{s \in S}$
  while `misprice`
Return $x^{(k)}$

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

As, at each iteration, the cuts are now generated from a first-stage solution $x^{(k)}$ that may be different from the first-solution to $(RMP)^{(k)}$, there is no guarantee that the cuts added separate the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s \in S})$. When there is no cut, among added cuts, that separates the solution to the relaxed master program, we say that first-stage solution $x^{(k)}$ induces a **mis-pricing** (Pessoa et al., 2013). We represent such a case in Figure 1. Then, there is no need to solve again the relaxed master program as its solution remains the same. A boolean variable `misprice` appears in Algorithm 3 to handle such a case.

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 in $x^{(k)}$. 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 with an optimality gap $\epsilon \geqslant 0$. 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 there exists no cut derived from this batch of subproblems, or a previous batch, which separates the solution $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$ to the relaxed master program [see Figure 1]. The variable `misprice` still equals `True`. As the solution to the relaxed master program has not been cut, it is useless to solve the relaxed master program again. We exit the optimality loop, but stay in the separation loop. We define a new separation point $x^{(k)}$, a new permutation of $[\![1, \kappa]\!]$, and begin a new optimality loop.
- **Case 3:** 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 $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$ to the relaxed master program [see Figure 2]. 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.

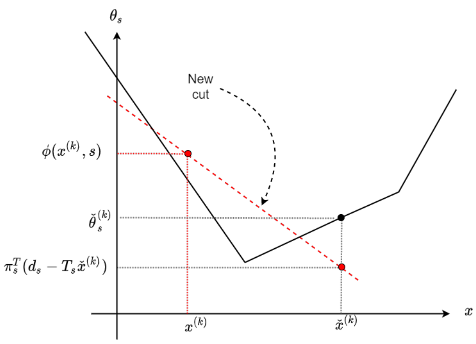


Figure 1: The cut derived from first-stage solution $x^{(k)}$ does not separate the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$. The solution to $(RMP)^{(k)}$ remains the same. The separation point $x^{(k)}$ induces a mis-pricing.

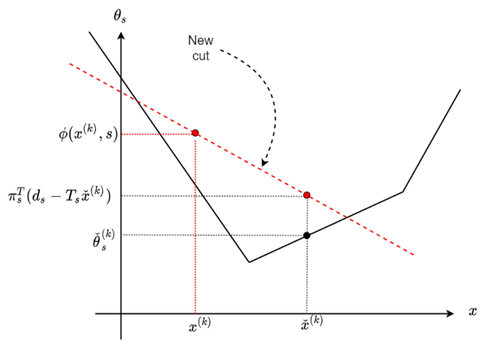


Figure 2: The cut derived from first-stage solution $x^{(k)}$ separates the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$.

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 5 (Convergence property and finite 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 sequence of stabilization state vectors obtained by successive applications 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:*

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

- *We say that a primal stabilization scheme $(\mathcal{D}, \psi_1, \psi_2)$ satisfies the **finite convergence property** if:*

$$\forall (x, d) \in X \times \mathcal{D},\ \exists \ell_0 \in \mathbb{N}^*,\ \psi_2\big(d_x^{\ell_0}\big) = x$$

We first need to prove the following intermediate results to show that the stabilized Benders by batch algorithm effectively converges to an optimal solution to problem (1).

Proposition 4. *Let $\epsilon > 0$ (resp. $\epsilon \geqslant 0$) be the optimality gap of Algorithm 3, $k \in \mathbb{Z}^+$ an iteration, and $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$ an optimal solution to $(RMP)^{(k)}$. If $\big(x^{(k+r)}\big)_{r\in\mathbb{N}}$ is a sequence of elements of X converging to $\check{x}^{(k)}$ (resp. converging to $\check{x}^{(k)}$ in a finite number of iterations) and $\big(\sigma^{(k+r)}\big)_{r\in\mathbb{N}}$ a sequence of permutations of $[\![1, \kappa]\!]$, then there exists $t \in \mathbb{N}$ such that one of the following assertions is true:*

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

Proof of proposition 4. See Appendix A.4. ☐

Proposition 5. *If the primal stabilization scheme satisfies the **convergence property** (resp. **finite convergence property**) of Definition 5, then the stabilized Benders by batch algorithm converges to an optimal solution to problem (1) in a finite number of iterations, for every $\epsilon > 0$ (resp. $\epsilon \geqslant 0$).*

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

1. $\forall i \in [\![1, \kappa]\!]$, batch $S_{\sigma(i)}$ is $\epsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$. Then $x^{(k)}$ is an optimal solution to problem (1) with an optimality gap of $\epsilon > 0$ (resp. $\epsilon \geqslant 0$).
2. There exists an index $i \in [\![1, \kappa]\!]$ 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 in this situation a finite number of times.
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 (resp. the finite convergence property), the sequence of separation points converges to $\check{x}^{(k)}$ (resp. in a finite number of iterations). 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). ☐

Remark 4. *The classic Benders decomposition algorithm is equivalent to the Benders by batch algorithm with a batch size $\eta = card(S)$. Therefore, Algorithm 3 describes a valid way to add primal stabilization to the classic 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: $\hat{x}^{(k)} = \arg\min_{j\in[\![0,k-1]\!]} \{c^\top x^{(j)} + \sum_{s\in S} p_s \phi(x^{(j)}, s)\}$. Then 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 $k \geqslant 1$, and $x^{(0)} \in X$ is a feasible first-stage solution. This basically consists in doing $100\alpha\%$ of the way from the previous separation point to the solution to the master program. This can 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 $k \in \mathbb{N}$.

The basic stabilization scheme can be expressed according to Definition 3 as:

$$\begin{array}{rcl} \mathcal{D} & = & X^2 \\ \psi_1 & : & \left\{ \begin{array}{lcl} X \times \mathcal{D} & \rightarrow & \mathcal{D} \\ x, (y, z) & \mapsto & (x, \alpha y + (1 - \alpha) z) \end{array} \right. \\ \psi_2 & : & \left\{ \begin{array}{lcl} \mathcal{D} & \rightarrow & X \\ (y, z) & \mapsto & \alpha y + (1 - \alpha) z \end{array} \right. \end{array}$$

with $d^0 = (x^{(0)}, x^{(0)})$ where $x^{(0)} \in X$ is a feasible first-stage solution. The vector of parameters $d^{(k)}$ computed at the iteration k is equal to $(\check{x}^{(k)}, x^{(k-1)})$.

Proposition 6. *The basic stabilization scheme satisfies the convergence property.*

Proof of proposition 6. See Appendix A.5. □

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:

$$\left\{ \begin{array}{rcl} \bar{x}^{(k)} & = & \beta \bar{x}^{(k-1)} + (1 - \beta) \check{x}^{(k)} \\ x^{(k)} & = & \alpha \bar{x}^{(k)} + (1 - \alpha) x^{(k-1)} \end{array} \right.$$

for $k \geqslant 1$, and $x^{(0)} = \bar{x}^{(0)} \in X$ is a feasible first-stage solution. 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).

The solution memory stabilization scheme can be expressed according to Definition 3 as:

$$\begin{array}{rcl} \mathcal{D} & = & X^2 \\ \psi_1 & : & \left\{ \begin{array}{lcl} X \times \mathcal{D} & \rightarrow & \mathcal{D} \\ x, (y, z) & \mapsto & (\beta y + (1 - \beta) x, \alpha y + (1 - \alpha) z) \end{array} \right. \\ \psi_2 & : & \left\{ \begin{array}{lcl} \mathcal{D} & \rightarrow & X \\ (y, z) & \mapsto & \alpha y + (1 - \alpha) z \end{array} \right. \end{array}$$

with $d^0 = (x^{(0)}, x^{(0)})$ where $x^{(0)} \in X$ is a feasible first-stage solution. The vector of parameters $d^{(k)}$ computed at the iteration k is equal to $(\bar{x}^{(k)}, x^{(k-1)})$.

Proposition 7. *The solution memory stabilization scheme satisfies the convergence property.*

Proof of proposition 7. See Appendix A.6. □

It is possible to adapt both schemes so that they satisfy the finite convergence property. Specifically, the separation point should become equal to the solution to the relaxed master program in a finite number of iterations when there are successive iterations which induce a mis-pricing. For the basic stabilization scheme, this implies that the value of α should increase to become equal to one in a finite number of iterations if successive mis-pricings occur. If $t \in \mathbb{N}$ denotes the number of consecutive mis-pricings that have occurred before starting iteration k of the algorithm, then computing $x^{(k)}$ replacing α by $\min\{1, \alpha(1 + t)\}$ works. For the solution memory stabilization scheme, in similar cases, the value of α should increase to become equal to one and the value of β should decrease to become equal to zero in a finite number of iterations.

5 Experimental design 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 that we used for comparison, and how we implemented them. Finally, we show and analyze the numerical results we obtained.

5.1 Instances

We use seven 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/~swright/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 two last instances come from `https://people.orie.cornell.edu/huseyin/research/research.html`. The first one, product, is the large instance of the product distribution problem available at `https://people.orie.cornell.edu/huseyin/research/sp_datasets/sp_datasets.html`. The second one, 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 planning horizon.

As those instances have a tremendous number of scenarios (see Table 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 $S + k,\ k \in \{0, 1, 2\}$ so that two instances of different sample size should not share sub-samples. This leads to a benchmark of 84 different instances. In the following, we will refer to the instances of problem *prob* with *#scenarios* scenarios as *prob-N#scenarios*.

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

problem	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	321×1921	$> 3^{200}$
product	75×1500	700×1450	3^{450}

5.2 Experimental Design

In order to evaluate the numerical efficiency of our Benders by batch algorithm (**BbB**), we compare it to nine different methods.

The experimentations are run on one core (sequential mode), on an Intel® Xeon® Gold SKL-6130 processor at 2,1 GHz with 96 GB 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. We set the lower bound on the epigraph variables associated with the subproblems to 0 as it is a valid lower bound on LandS, gbd, ssn, storm, Fleet20_3 and 20term instances and to -10^{10} on product instances as 0 is not a valid lower bound on those instances.

First, we run IBM ILOG CPLEX 12.10 (IBM, 2019) to solve the deterministic reformulation with the barrier algorithm (**CPLEX Barrier** hereafter) and with its multicut Benders implementation (**CPLEX Benders**) (Bonami et al., 2020). We also compare to our implementation of the multicut Benders decomposition algorithm (**Classic multicut**) and our implementation of the monocut Benders decomposition algorithm (**Classic monocut**).

In order to evaluate the effect of primal stabilization, we also run our implementations of the level bundle method (Lemaréchal et al., 1995) using aggregated cut as in the monocut Benders decomposition algorithm (**Level Bundle**), our implementation of the multicut Benders decomposition algorithm with an in-out stabilization (**In-out multicut**) and our implementation of the monocut Benders decomposition algorithm with an in-out stabilization (**In-out monocut**). We describe these algorithms in Appendix B.

As the partial cut aggregation proposed in the Benders by batch algorithm can be seen as the static cut aggregation scheme described by Trukhanov et al. (2010), which have already shown improvements compared to pure monocut or multicut Benders decomposition algorithms, we also implement the Benders decomposition algorithm with the same cut aggregation level as the one used in the Benders by batch algorithms (**Classic CutAggr**). Given $(S_i)_{i=1,..,\eta}$ the same partition of the subproblems into batches than the one used in the Benders by batch algorithm, we solve all the subproblems at each iteration and add the following cuts $\sum_{s \in S_i} p_s \theta_s \geqslant \sum_{s \in S_i} p_s \Big(\pi_s^\top (d_s - T_s x)\Big), \ \forall i \in [\![1, \eta]\!]$. Finally, we implement the Benders decomposition with static cut aggregation and in-out stabilization (**In-out CutAggr**).

CPLEX Benders is run with the following parameter values: `benders strategy 2` (an annotation file contains the first-stage variables, and CPLEX automatically decomposes 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. In **Classic monocut** and **In-out monocut**, we compute the cuts as $\sum_{s \in S} p_s \theta_s \geqslant \sum_{s \in S} p_s \Big(\pi_s^\top (d_s - T_s x)\Big)$.

The subproblems are solved with the dual simplex algorithm for all methods. In all our implementations, 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.

In **Level Bundle**, **In-out multicut**, **In-out monocut** and **In-out CutAggr** and **BbB** with stabilization, the starting solution $x^{(0)}$ is obtained by solving the mean-value problem. We use a dynamic strategy to update the stabilization parameter α in **In-out monocut**, **In-out multicut** and **In-out CutAggr**. 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 $c^\top x^{(k)} + \sum_{s \in S} p_s \phi(s, x^{(k)}) \geqslant c^\top \hat{x}^{(k)} + \sum_{s \in S} p_s \phi(s, \hat{x}^{(k)})$, 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. Such a procedure cannot be used in the stabilized Benders by batch algorithm as the actual value of the recourse function is required. **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).

We also evaluate different parameters of **BbB**. We first run **BbB** without stabilization, and try different batch sizes with and without partial cut aggregation. Then, we evaluate the impact of the two proposed stabilization schemes, with different values for the stabilization parameters.

We coded all the methods using C++ and compiled them with GCC 9.3.0. Every stochastic linear program to solve is given as input to our program in the SMPS format (Gassmann and Schweitzer, 2001). Our implementation and the instances are accessible from this link: `https://gitlab.inria.fr/edge/benders-by-batch`.

5.3 Numerical results

This section shows the numerical results obtained on the 84 instances of our benchmark. 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 $> \frac{43200}{best\ time}$ 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. Detailed results instance by instance are presented in Appendix E. We always present the average time on the three instances of each size for each problem, rounded to the second (when computing times are larger than one 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 function of its performance ratios computed over a set of problems $\mathcal{P}$. It is defined as $\rho_m(\tau) = card(\{p \in \mathcal{P} : r_{p,m} \leqslant \tau\})$

The ratios presented in the following tables are computed as the expectation of the performance ratios over the three instances of each problem with the same number of subproblems.

5.3.1 Performance of BbB without stabilization

We first present the results of **BbB** without stabilization. We analyze the impact of the batch size, both without (Table 3) and with partial cut aggregation (Table 4). Each column of Tables 3 and 4 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%**). The variants with cut aggregation are respectively designated by **BbB 1% CutAggr**, **BbB 5% CutAggr**, **BbB 10% CutAggr** and **BbB 20% CutAggr**.

In order to estimate only the effect of performing an optimality check after solving each batch of subproblems, we compare in Table 3 the Benders by batch algorithm without cut aggregation (**BbB**) to **Classic multicut**, which can be seen as the Benders by batch algorithm without cut aggregation with a batch size equal to the total number of subproblems. We compare in Table 4 the Benders by batch algorithm with cut aggregation (**BbB CutAggr**) to **Classic CutAggr**, which corresponds to the Benders by batch algorithm with partial cut aggregation, in which all subproblems are solved at each iteration. The same partition of subproblems is used in **BbB 1% CutAggr** and **Classic 1% CutAggr**, as well as in **BbB 5% CutAggr** and **Classic 5% CutAggr**. We also present the results of **Classic monocut**, as a classical alternative to **Classic multicut** in Table 3 and as a method where cuts are fully aggregated in Table 4.

Table 3: Results for the Benders by batch algorithm without partial cut aggregation, with batch sizes from 1% to 20% of the total number of subproblems.

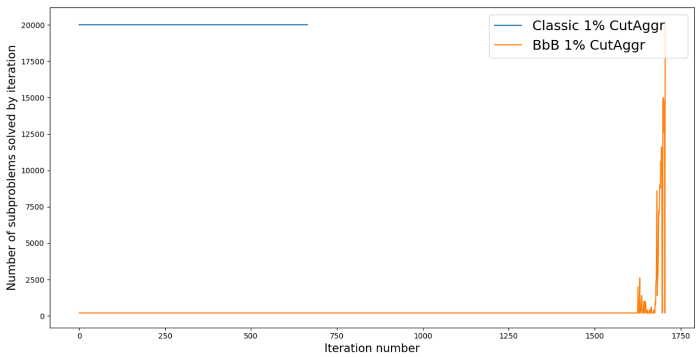
	Classic monocut		Classic multicut		BbB 1%		BbB 5%		BbB 10%		BbB 20%	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	2	3.0	0.75	1.1	2	2.7	0.83	1.3	0.72	1.1	**0.66**	**1.0**
LandS-N5000	11	1.7	9	1.5	13	2.2	8	1.3	7	1.1	**6**	**1.0**
LandS-N10000	22	1.1	29	1.5	38	2.0	25	1.3	21	1.1	**20**	**1.0**
LandS-N20000	**45**	**1.0**	105	2.3	130	2.9	89	2.0	80	1.8	72	1.6
gbd-N1000	2	3.3	0.94	1.4	2	3.6	**0.65**	**1.0**	0.84	1.3	0.96	1.5
gbd-N5000	12	1.9	10	1.7	16	2.5	**6**	**1.0**	7	1.1	8	1.3
gbd-N10000	23	1.2	33	1.7	47	2.5	**19**	**1.0**	22	1.2	25	1.3
gbd-N20000	**48**	**1.0**	121	2.5	96	2.0	61	1.3	71	1.5	87	1.8
ssn-N1000	2408	611.6	7	1.8	6	1.6	**4**	**1.0**	4	1.1	5	1.2
ssn-N5000	13460	590.1	57	2.5	32	1.4	**24**	**1.0**	28	1.2	32	1.4
ssn-N10000	25901	444.1	188	3.2	71	1.2	79	1.3	**59**	**1.0**	79	1.3
ssn-N20000	$+\infty$	>364.8	488	4.1	**145**	**1.2**	274	2.3	624	5.2	2821	24.9
storm-N1000	24	3.7	11	1.7	21	3.2	8	1.3	**6**	**1.0**	8	1.3
storm-N5000	114	2.1	106	1.9	175	3.2	60	1.1	**55**	**1.0**	65	1.2
storm-N10000	224	1.4	496	3.2	492	3.2	**156**	**1.0**	159	1.0	189	1.2
storm-N20000	**458**	**1.0**	2370	5.2	1390	3.0	580	1.3	672	1.5	588	1.3
20term-N1000	577	15.2	757	19.9	**38**	**1.0**	82	2.2	49	1.3	74	1.9
20term-N5000	3506	5.6	24429	38.6	**634**	**1.0**	2101	3.3	1335	2.1	2247	3.6
20term-N10000	6901	3.0	$+\infty$	>19.9	**2270**	**1.0**	10733	4.7	6199	2.7	10413	4.6
20term-N20000	**13687**	**1.3**	$+\infty$	>6.2	20625	1.7	$+\infty$	>4.2	$+\infty$	>4.2	$+\infty$	>4.2
Fleet20_3-N1000	533	9.1	225	3.9	145	2.5	95	1.7	102	1.7	**74**	**1.2**
Fleet20_3-N5000	2757	1.5	5330	2.9	2417	1.3	1950	1.0	**1873**	**1.0**	2097	1.1
Fleet20_3-N10000	**5710**	**1.0**	28933	5.1	9903	1.7	19913	3.4	8537	1.5	21383	3.7
Fleet20_3-N20000	**11300**	**1.0**	$+\infty$	>4.1	34900	3.1	$+\infty$	>3.8	$+\infty$	>3.9	$+\infty$	>3.9
product-N1000	1947	19.0	186	1.8	270	2.6	123	1.2	105	1.0	**103**	**1.0**
product-N5000	10467	7.6	3497	2.5	3730	2.7	1873	1.4	1483	1.1	**1377**	**1.0**
product-N10000	20200	3.7	15200	2.8	13300	2.5	6893	1.3	5583	1.0	**5397**	**1.0**
product-N20000	43000	1.9	$+\infty$	>2.0	$+\infty$	>1.9	29700	1.3	24733	1.1	**23067**	**1.0**

We first notice in Table 3 that **BbB 1%** solves all the instances, except Fleet20_3-N20000 where it only succeeds to solve one out of three problems, whereas **Classic Multicut** fails to solve optimally four groups of instances. As the algorithm avoids solving many subproblems and adding cuts in the relaxed master program, it overcomes the issue of the time spent in solving subproblems and delays the size growth of the relaxed master program. However, as we still add one cut for each solved subproblem in the Benders by batch algorithm, it still does not scale well when the number of subproblems becomes large. **Classic monocut** outperforms **BbB** on large-scale instances such as 20term with 20000 subproblems or Fleet20_3 with 20000 subproblems.

Table 4 shows that when partial cut aggregation is used, all the presented methods clearly outperform **Classic monocut**. 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 samples of subproblems, the algorithms avoid many symmetries due to the sum of the cuts over the subproblems. The table shows also that the best batch sizes are 1% and 5% (respectively **BbB 1% CutAggr** and **BbB 5% CutAggr**), except for two small instances. The two methods can be up to 25 times faster than **Classic 1% CutAggr** and more than 58 times faster than **Classic 5% CutAggr**.

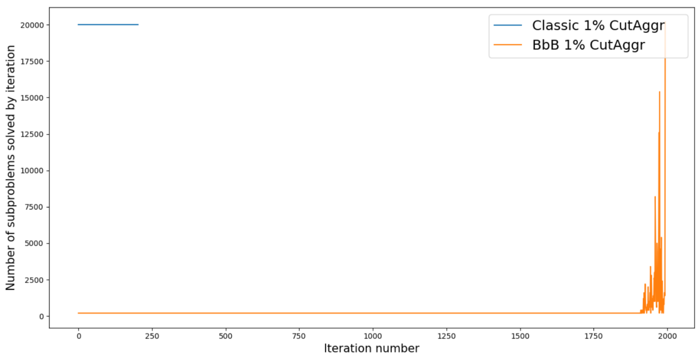
Table 4: Results for the Benders by batch algorithm with partial cut aggregation, with batch sizes from 1% to 20% of the total number of subproblems.

	Classic monocut		Classic 1% CutAggr		Classic 5% CutAggr		BbB 1% CutAggr		BbB 5% CutAggr		BbB 10% CutAggr		BbB 20% CutAggr	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	2	2.5	1	1.3	1	1.7	2	2.1	0.88	1.1	**0.78**	**1.0**	0.89	1.1
LandS-N5000	11	2.6	7	1.8	8	2.0	9	2.3	5	1.1	**4**	**1.0**	4	1.1
LandS-N10000	22	2.7	16	2.0	19	2.3	16	2.0	**8**	**1.0**	8	1.0	9	1.2
LandS-N20000	45	2.6	34	1.9	39	2.3	44	2.6	**17**	**1.0**	18	1.0	20	1.2
gbd-N1000	2	3.6	1	2.0	2	2.7	2	2.7	**0.61**	**1.0**	0.78	1.3	0.93	1.5
gbd-N5000	12	3.6	9	2.6	10	3.0	9	2.7	**3**	**1.0**	4	1.1	4	1.3
gbd-N10000	23	3.7	19	3.1	21	3.3	15	2.3	**6**	**1.0**	8	1.3	9	1.5
gbd-N20000	48	3.6	41	3.0	46	3.4	41	3.1	**14**	**1.0**	15	1.1	19	1.4
ssn-N1000	2408	175.8	24	1.8	142	10.5	**14**	**1.0**	61	4.5	134	9.8	242	17.7
ssn-N5000	13460	150.6	399	4.5	1582	17.7	**89**	**1.0**	322	3.6	659	7.4	1322	14.8
ssn-N10000	25901	140.4	1246	6.7	4858	26.1	**185**	**1.0**	707	3.8	1423	7.7	2914	15.8
ssn-N20000	$+\infty$	>98.4	8603	20.0	26122	58.9	**441**	**1.0**	1615	3.7	3386	7.7	6757	15.4
storm-N1000	24	3.8	12	2.0	15	2.4	12	1.9	**6**	**1.0**	7	1.1	9	1.5
storm-N5000	114	3.4	72	2.1	94	2.8	52	1.5	**34**	**1.0**	36	1.1	55	1.6
storm-N10000	224	3.0	164	2.2	198	2.7	110	1.5	**74**	**1.0**	82	1.1	104	1.4
storm-N20000	458	2.9	369	2.3	423	2.6	226	1.4	**163**	**1.0**	169	1.1	238	1.5
20term-N1000	577	39.4	272	18.5	313	21.4	**15**	**1.0**	37	2.5	68	4.6	141	9.6
20term-N5000	3506	50.3	1604	23.2	1945	28.0	**70**	**1.0**	193	2.8	395	5.7	839	12.1
20term-N10000	6901	53.2	3364	26.0	4840	37.4	**130**	**1.0**	402	3.1	898	6.9	1978	15.3
20term-N20000	13687	49.1	7032	25.2	16287	57.3	**280**	**1.0**	914	3.3	2051	7.3	18312	65.2
Fleet20_3-N1000	533	18.9	125	4.4	222	7.9	**28**	**1.0**	42	1.5	74	2.6	131	4.7
Fleet20_3-N5000	2757	25.7	903	8.4	1530	14.3	**107**	**1.0**	211	2.0	358	3.3	649	6.1
Fleet20_3-N10000	5710	26.9	2000	9.4	3460	16.3	**212**	**1.0**	440	2.1	721	3.4	1310	6.2
Fleet20_3-N20000	11300	27.0	5053	12.1	7860	18.8	**419**	**1.0**	876	2.1	1520	3.6	2777	6.6
product-N1000	1947	20.0	190	2.0	431	4.4	**98**	**1.0**	141	1.5	253	2.6	505	5.2
product-N5000	10467	28.9	1523	4.2	3323	9.2	**362**	**1.0**	773	2.1	1567	4.3	2873	7.9
product-N10000	20200	25.0	3827	4.8	7757	9.7	**823**	**1.0**	1523	1.9	3053	3.8	5530	6.9
product-N20000	43000	25.7	9963	6.0	19367	11.6	**1693**	**1.0**	3367	2.0	6320	3.8	12500	7.5



algorithm	total time	(RMP) time	# solved	(SP) time	# solved
Classic multicut	>43200	>43200	>20	>206	>400000
Classic monocut	13429	23	1732	12297	34640000
Classic 1% CutAggr	7375	1472	665	5610	13300000
BbB 1% CutAggr	261	26	1706	204	576000

(a) a 20term instance with 20000 subproblems (20term-N20000-s20000)



algorithm	total time	(RMP) time	# solved	(SP) time	# solved
Classic multicut	>43200	>43200	>17	>664	>340000
Classic monocut	>43200	>1697	>864	>25704	>17280000
Classic 1% CutAggr	9820	957	204	6186	4080000
BbB 1% CutAggr	1790	211	1994	863	547000

(b) a product instance with 20000 subproblems (product-N20000-s20000)

Figure 3: Number of subproblems solved at each iteration by **BbB 1% CutAggr** and **Classic 1% CutAggr** (left plots). For **Classic monocut**, **Classic multicut**, **BbB 1% CutAggr**, **Classic 1% CutAggr**, the total number of relaxed master programs and subproblems solved, as well as the associated solution time (right plots). Symbol ">" means that the time limit is reached without proven optimality. To the sum of the time needed to solve the relaxed master programs and the subproblems, one must add the time needed for the other operations (e.g., solving the mean-value problem to obtain $x^{(0)}$, cut computation and their addition to (RMP), configuration of the subproblems for each new first-stage solution).

The better performance of the Benders by batch algorithm with partial cut aggregation can be explained by Figure 3. We see that in most of the iterations, the algorithm solves only one batch of subproblems to show that the current first-stage solution cannot be proven optimal and to separate it. Despite the greater number of iterations performed by **BbB 1% CutAggr** due to its explorative nature, we observe that it needs to solve less subproblems than **Classic 1% CutAggr** to converge. Specifically, for a 20term instance with 20000 subproblems and a product instance with 20000 subproblems, **BbB 1% CutAggr** solves respectively 23 times less and 7 times less subproblems than **Classic 1% CutAggr** to converge. Although **Classic 1% CutAggr** evaluates almost three times less first-stage solutions for the 20term instance (and more than 10 times less for the product instance), it takes ultimately more time to converge than **BbB 1% CutAggr**: 7375 seconds for **Classic 1% CutAggr** compared to 261 seconds for **BbB 1% CutAggr** to solve the 20term instance, and 9820 seconds for **Classic 1% CutAggr** compared to 1790 seconds for **BbB 1% CutAggr** to solve the product instance. This can be explained by the fact that the relaxed master program contains fewer cuts at most iterations in **BbB 1% CutAggr** than in **Classic 1% CutAggr**. We observe that the time spent in solving the subproblems represents most of the computing time for the 20term instance and most of the computing time for the product instance. All of the above suggests that the smaller the first-stage problem is, the more efficient the Benders by batch algorithm is.

5.3.2 Impact of the stabilization on BbB

We now present the results obtained when the two stabilization schemes presented in §4.3 are applied to the most competitive versions of Bbb (batch sizes of 1% and 5%, and with partial cut aggregation). Figures 4 and 5 show the performance profiles of **BbB CutAggr** 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% CutAggr** or **BbB 5% CutAggr** followed by the values for the parameters.

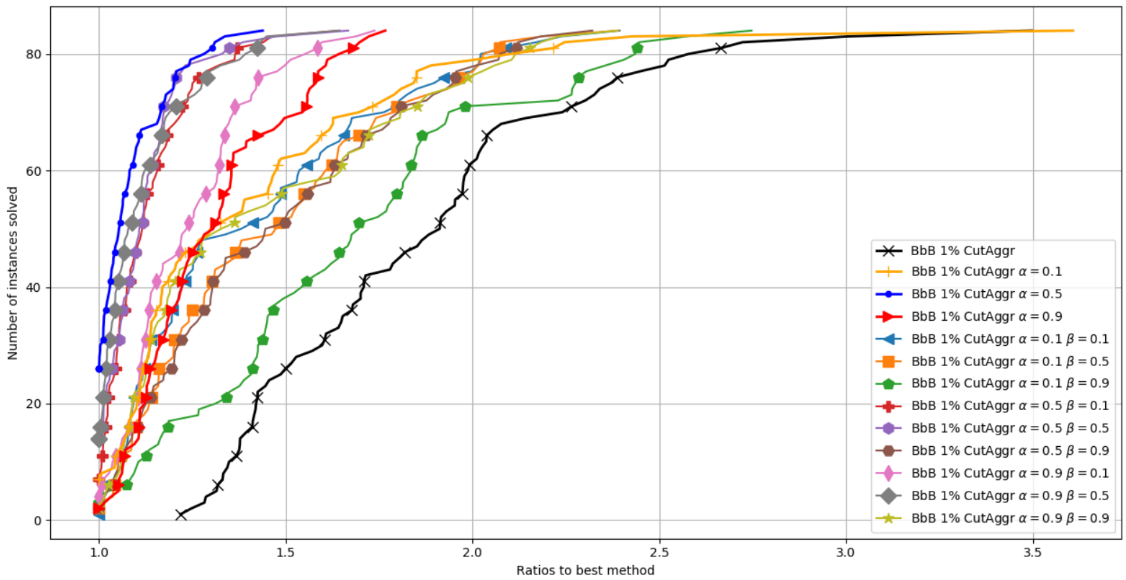


Figure 4: Performance profiles of the stabilized Benders by batch algorithm with batch size of 1% and cut aggregation.

Figure 4 shows that the proposed stabilization schemes accelerate **BbB 1% CutAggr**, 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% CutAggr**. The same four methods are the most efficient and equivalent to each other. The algorithm with a solution memory stabilization parameterized by $(\alpha, \beta) = (0.1, 0.9)$ is less efficient than **BbB 5% CutAggr**. 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.

To conclude, results show no clear difference between the two proposed stabilization schemes. 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 similar computational results for the tested instances.

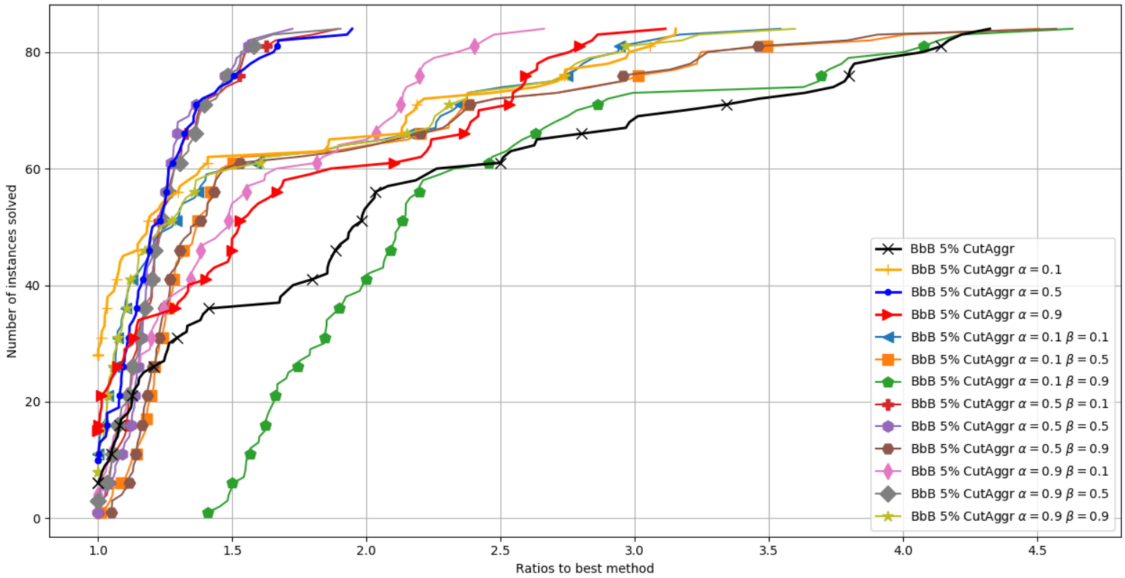


Figure 5: Performance profiles of the stabilized Benders by batch algorithm with batch size of 5% and cut aggregation.

Table 5: Final results, the best stabilized Benders by batch algorithm compared to all stabilized benchmark methods.

	CPLEX Barrier		Level Bundle		In-out multicut		In-out monocut		In-out 1% CutAggr		In-out 5% CutAggr		BbB 1% CutAggr $\alpha = 0.5$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	**0.07**	**1.0**	1	17.3	0.89	12.4	1	20.0	0.71	9.7	0.98	13.4	0.96	13.2
LandS-N5000	**1**	**1.0**	7	9.0	8	10.5	9	10.5	5	6.0	6	7.2	5	6.7
LandS-N10000	**1**	**1.0**	14	14.0	24	23.6	16	15.6	10	9.7	11	11.1	9	9.0
LandS-N20000	**5**	**1.0**	27	6.8	62	16.5	41	10.4	22	5.6	22	5.5	21	5.4
gbd-N1000	**0.04**	**1.0**	2	61.2	1	36.6	2	58.8	1	33.6	2	44.8	0.88	25.6
gbd-N5000	**0.17**	**1.0**	10	60.1	10	60.9	10	64.0	7	41.8	8	47.1	4	24.8
gbd-N10000	**0.35**	**1.0**	24	69.5	23	67.5	21	61.7	16	45.7	17	50.3	8	22.2
gbd-N20000	**0.91**	**1.0**	44	48.8	82	89.8	54	60.6	30	34.3	34	39.1	17	18.5
ssn-N1000	32	6.0	90	17.1	**6**	**1.0**	137	27.3	10	1.8	19	3.6	8	1.5
ssn-N5000	310	10.6	657	22.2	**31**	**1.0**	795	27.4	70	2.4	133	4.5	47	1.6
ssn-N10000	1223	20.3	1501	25.2	**63**	**1.0**	1464	23.3	171	2.9	312	5.2	91	1.5
ssn-N20000	2619	13.7	3109	16.3	243	1.3	2861	15.2	400	2.1	736	3.9	**191**	**1.0**
storm-N1000	41	5.8	15	2.1	9	1.3	14	2.1	8	1.1	9	1.4	**7**	**1.0**
storm-N5000	316	9.7	76	2.3	41	1.3	62	1.9	49	1.5	52	1.6	**33**	**1.0**
storm-N10000	764	11.8	145	2.3	125	1.9	201	3.1	99	1.5	110	1.7	**65**	**1.0**
storm-N20000	2390	17.4	288	2.1	573	4.2	252	1.8	211	1.5	232	1.7	**137**	**1.0**
20term-N1000	14	1.3	217	20.9	36	3.5	114	10.8	27	2.6	44	4.3	**10**	**1.0**
20term-N5000	82	1.7	1044	21.2	482	9.7	681	13.8	197	4.0	269	5.5	**50**	**1.0**
20term-N10000	199	2.0	2450	24.4	2805	27.9	1190	11.8	474	4.7	593	5.9	**100**	**1.0**
20term-N20000	455	2.3	4843	24.7	10992	56.0	1754	8.9	1010	5.1	1371	7.0	**197**	**1.0**
Fleet20_3-N1000	23	1.3	107	6.2	50	2.9	93	5.4	26	1.5	41	2.4	**17**	**1.0**
Fleet20_3-N5000	269	3.6	500	6.7	719	9.6	473	6.3	184	2.4	250	3.3	**75**	**1.0**
Fleet20_3-N10000	809	5.5	1004	6.9	3747	25.6	1029	7.0	435	3.0	590	4.0	**146**	**1.0**
Fleet20_3-N20000	2446	7.9	2730	8.8	17000	54.7	1780	5.8	1018	3.3	1313	4.2	**310**	**1.0**
product-N1000	179	2.3	625	8.2	81	1.1	513	6.7	113	1.5	183	2.4	**76**	**1.0**
product-N5000	2121	6.7	3200	10.3	1127	3.6	2690	8.7	787	2.5	1380	4.4	**312**	**1.0**
product-N10000	4397	8.0	7173	13.0	5357	9.8	5730	10.4	1970	3.6	3133	5.7	**552**	**1.0**
product-N20000	15463	13.6	14300	12.5	$+\infty$	>40.5	12333	10.8	4887	4.3	7983	7.0	**1140**	**1.0**

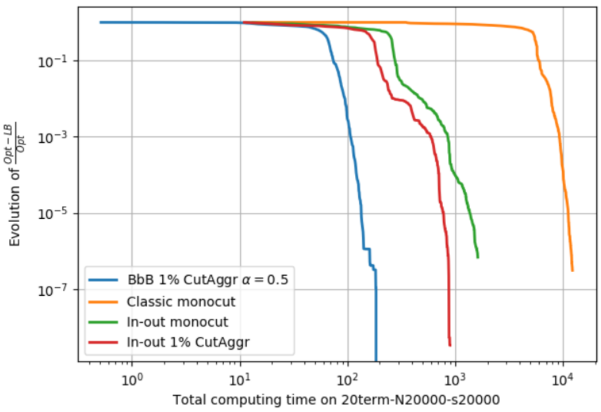
5.3.3 Comparison with state-of-the-art methods

We now compare the stabilized Benders by batch algorithm to classical methods of the literature. We show in Table 5 the times and ratios of **CPLEX Barrier** and all the stabilized methods of our benchmark, **In-out monocut**, **In-out multicut**, **Level bundle**, **In-out 1% CutAggr** and **In-out 5% CutAggr** with the best performing stabilized Benders by batch **BbB 1% CutAggr $\alpha = 0.5$**. We first observe that, on the small instances LandS and gbd, **CPLEX Barrier** converges faster than all the other methods. As those instances have very few variables both in first and second stages, they remain small even with 20000 subproblems, and are solved very efficiently by **CPLEX Barrier**. However, we can notice that even for these small instances, **BbB 1% CutAggr $\alpha = 0.5$** is the best method among all the cutting planes algorithms. Table 5 shows clearly that the stabilized Benders by batch algorithm outperforms all the other methods on the large instances, and can be up to more than 25 times faster than **Level Bundle** or 15 times faster than **In-out**

monocut. We also show that, even if **In-out 1% CutAggr** outperforms other classical stabilized methods from the literature, the stabilized Benders by batch algorithm can be up to 5 times faster. This shows that, firstly, using a static cut aggregation combined with primal stabilization allows to speed up classical methods used to benchmark algorithms from the literature, and secondly, that not solving systematically all the subproblems allows to further improve the computing times on the test instances.

As for the unstabilized case, we observe in our experiments that **BbB 1% CutAggr** $\boldsymbol{\alpha} = \mathbf{0.5}$ needs to solve way less subproblems than other methods to converge, and that the time spent in solving the subproblems represents almost all the computing time in all presented methods (see Appendix C).

Figure 6 shows the evolution of the relative gap between the lower bound and the optimal value, of twodifferent algorithms, on four different instances, according to the time. We see that adding only a few cuts at each iterations allows the lower bound to converge faster to the optimal value to the problem. Moreover, we observe that, on three of the four presented instances, **BbB 1% CutAggr** $\boldsymbol{\alpha} = \mathbf{0.5}$ reaches a relative gap of 10^{-6} while all the other algorithms still have a large relative gap (e.g. 10^{0} on ssn or 10^{-1} on Fleet). Although **BbB 1% CutAggr** $\boldsymbol{\alpha} = \mathbf{0.5}$ adds less cuts at each iteration, its lower bound value is usually larger than the one computed in the other algorithms, when compared for the same computing time, except for some very short time intervals early in the solution process where the lower bound in **In-Out 1% CutAggr** is better. This suggests that the cuts generated when the approximation of the subproblem value function is coarse, not only take time to be computed, but also do not help much to improve the value of the lower bound.



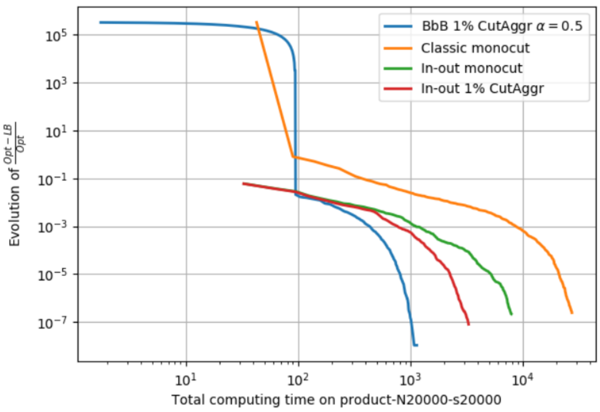


Figure 6: Evolution of the relative gap between the lower bound and the optimal value as a function of time, on a two instances with 20000 subproblems (20term-N20000-s20000 and product-N20000-s20000)

5.3.4 Sensitivity of BbB to the initial order of the subproblems

We performed several experiments testing different initial orders of the subproblems to assess the sensitivity of the computing time of our method to this choice. We ran **BbB 1% CutAggr** $\boldsymbol{\alpha} = \mathbf{0.5}$, for 500 different initial orders, on one instance with 5000 subproblems and one with 10000 subproblems for each tested problem. We report in Table 6 the minimum and maximum times observed, the median, and the first and ninth decile on computing times. We observe that the initial order has usually a limited impact on the efficiency of our algorithm. We also remark that the stabilized Benders by batch algorithm present lower computing times than **In-out 1% CutAggr**, the best performing method used as comparison in the numerical results, even for the maximum time observed. Although the impact is in general limited, we observe that the initial order can have an impact on the computing time for some instances, such as LandS or gbd. However, the computing times observed are almost always smaller than the computing times of **In-out 1% CutAggr**, the best performing method in the literature to which BbB is compared to in the paper.

We also evaluated the impact of the optimality gap on the convergence of the algorithm. We see expected results (see Appendix D), that is, a smaller optimality gap induces larger computing times on the largest instances of our test set, but this would also be the case with the other algorithms.

Table 6: Computing times for **BbB 1% CutAggr $\boldsymbol{\alpha} = \mathbf{0.5}$** on 500 different initial orders of the subproblems

	Min Time		10%		50%		90%		Max Time		In-out 1% CutAggr	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N5000	**4.1**	**1.0**	4.5	1.1	5.3	1.3	6.2	1.5	7.3	1.8	5.0	1.2
LandS-N10000	**8.3**	**1.0**	9.2	1.1	10.2	1.2	11.9	1.4	15.6	1.9	10.0	1.2
gbd-N5000	**3.1**	**1.0**	3.5	1.1	4.1	1.3	5.0	1.6	7.1	2.3	7.0	2.3
gbd-N10000	**6.0**	**1.0**	7.2	1.2	8.3	1.4	10.3	1.7	14.0	2.3	16.0	2.7
ssn-N5000	**40.2**	**1.0**	44.3	1.1	46.8	1.2	49.8	1.2	54.1	1.3	70.0	1.7
ssn-N10000	**82.5**	**1.0**	87.3	1.1	92.5	1.1	102.0	1.2	122.4	1.5	171.0	2.1
storm-N5000	**28.0**	**1.0**	29.8	1.1	31.4	1.1	34.5	1.2	43.5	1.6	49.0	1.8
storm-N10000	**58.0**	**1.0**	60.5	1.0	64.2	1.1	69.7	1.2	83.2	1.4	99.0	1.7
20term-N5000	**43.5**	**1.0**	47.8	1.1	54.1	1.2	61.6	1.4	77.2	1.8	197.0	4.5
20term-N10000	**82.0**	**1.0**	91.5	1.1	103.2	1.3	115.0	1.4	136.2	1.7	474.0	5.8
Fleet20_3-N5000	**72.5**	**1.0**	74.7	1.0	76.6	1.1	78.7	1.1	83.3	1.1	184.0	2.5
Fleet20_3-N10000	**142.0**	**1.0**	148.0	1.0	152.0	1.1	157.0	1.1	166.0	1.2	435.0	3.1
product-N5000	**268.0**	**1.0**	279.0	1.0	292.0	1.1	315.0	1.2	355.0	1.3	787.0	2.9
product-N10000	**528.0**	**1.0**	553.0	1.0	573.0	1.1	603.0	1.1	679.0	1.3	1970.0	3.7

6 Conclusion

We proposed in this paper the Benders by batch algorithm to solve two-stage stochastic linear programming problems with finite probability distribution. This algorithm solves only a few subproblems at most iterations. The algorithm is exact and does not need a fixed recourse or a deterministic objective function. We showed that performing an optimality check after the resolution of a very few subproblems, each 1% of the numbers of subproblems in our tests, allows to significantly improve the solution time.

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 two schemes satisfying it. The stabilized Benders by batch algorithm can be up to 25 times faster than the level bundle method, or 5 times faster than Benders decomposition with in-out stabilization and static partial cut aggregation of (Trukhanov et al., 2010).

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 be parallelized and may benefit from effective parallelized methods, such as the asynchronous method of Linderoth and Wright (2003). The use of more advanced cut aggregation strategies is also a path worth exploring. Finally, an interesting perspective is to adapt the Benders by batch algorithm to solve mixed-integer master programs within a Branch&Cut framework.

Acknowledgments

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.
We thank the anonymous referees, whose comments helped improve and clarify this paper.

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.

Bodur, M. and Luedtke, J. R. (2022). Two-stage linear decision rules for multi-stage stochastic programming. *Mathematical Programming*, 191(1):347–380.

Bonami, P., Salvagnin, D., and Tramontani, A. (2020). Implementing Automatic Benders Decomposition in a Modern MIP Solver. In *Integer Programming and Combinatorial Optimization*, volume 12125, pages 78–90. Springer International Publishing, Cham.

Crainic, T. G., Hewitt, M., Maggioni, F., and Rei, W. (2020). Partial Benders Decomposition: General Methodology and Application to Stochastic Network Design. *Transportation Science*.

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.

Gassmann, H. I. and Schweitzer, E. (2001). A comprehensive input format for stochastic linear programs. *Annals of Operations Research*, 104(1):89–125.

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.

IBM (2019). IBM ILOG CPLEX 12.10 User's Manual (IBM ILOG CPLEX Division, Incline Village, NV).

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 andR.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.

Nesterov, Y. (2004). Nonsmooth Convex Optimization. In Nesterov, Y., editor, *Introductory Lectures on Convex Optimization: A Basic Course*, pages 111–170. Springer US, Boston, MA.

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.

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

Ruszczyński, 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.

Shapiro, A. and Nemirovski, A. (2005). On Complexity of Stochastic Programming Problems. In Jeyakumar, V. and Rubinov, A., editors, *Continuous Optimization: Current Trends and Modern Applications*, Applied Optimization, pages 111–146. Springer US, Boston, MA.

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.

Vanderbeck, F. (2005). Implementing Mixed Integer Column Generation. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, pages 331–358. Springer US, Boston, MA.

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., Fábián, 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., Fábián, 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 Proofs

A.1 Proof of Proposition 1

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

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

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

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

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

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

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

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

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

By definition of ϵ_κ we have:

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

Then, using equation (6), we have:

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

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

A.2 Proof of Proposition 2

Proof. We solve each subproblem at most once for every optimal solution to $(RMP)^{(k)}$ because $(S_1, S_2, ..., S_\kappa)$ defines a partition of S. Then if there exists a cut violated by $\left(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S}\right)$, 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 $\left(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S}\right)$, then the aggregated cut of the batch also separates it, and the result remains true. □

A.3 Proof of Proposition 3

Proof. Let $x \in X$ be a first-stage solution such that batch $S_{\sigma(i)}$ is $\epsilon_i(x)$-approximated by $(RMP)^{(k)}$, for all $i \in [\![1,\kappa]\!]$. Then, $S_{\sigma(\kappa)}$ is $\epsilon_\kappa(x)$-approximated by $(RMP)^{(k)}$. This means:

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

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

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

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

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

A.4 Proof of Proposition 4

Proof. The proof consists of two cases:

1. $\epsilon > 0$ and $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ converges to $\check{x}^{(k)}$
2. $\epsilon \geqslant 0$ and $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ converges to $\check{x}^{(k)}$ in a finite number of iterations

- **Case 1:** Let $\epsilon > 0$ be the optimality gap and $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ be a sequence of elements of X converging to $\check{x}^{(k)}$. We focus on the solution $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S})$ to the relaxed master program. There are two possible sub-cases:
  - **Sub-case 1.1** There exists $t_0 \in \mathbb{N}$ such that for all $l \geqslant t_0$ and for each index $i \in [\![1,\kappa]\!]$, batch $S_{\sigma^{(k+l)}(i)}$ is $\epsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$ with an optimality gap of $\frac{\epsilon}{4}$
  - **Sub-case 1.2** For all $t_0 \in \mathbb{N}$, there exists $l \geqslant t_0$ and an index $i \in [\![1,\kappa]\!]$ such that batch $S_{\sigma^{(k+l)}(i)}$ is not $\epsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$ with an optimality gap of $\frac{\epsilon}{4}$

Sub-case 1.1: Assume that there exists $t_0 \in \mathbb{N}$ such that for all $l \geqslant t_0$ and for each index $i \in [\![1,\kappa]\!]$, 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 $l \geqslant t_0$ and for every index $i \in [\![1,\kappa]\!]$,

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

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

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

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

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

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

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

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

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

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

$$\Rightarrow \Big[\sum_{s\in S_{\sigma^{(k+l)}(i)}} p_s\left(\phi\left(x^{(k+l)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+ \leqslant \epsilon - c^\top(x^{(k+l)}-\check{x}^{(k)}) - \Big[\sum_{t=1}^{i-1}\sum_{s\in S_{\sigma^{(k+l)}(t)}} p_s\left(\phi\left(x^{(k+l)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+$$

And for every index $i \in [\![1,\kappa]\!]$, 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).

Sub-case 1.2: Now assume that for all $t_0 \in \mathbb{N}$, there exists $l \geqslant t_0$ and an index $i \in [\![1,\kappa]\!]$ 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 $l \geqslant t_0$ and an index $i \in [\![1,\kappa]\!]$ such that:

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

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

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

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 obtained after solving batch $S_{\sigma^{(k+\tau)}(i)}$ can be written as follows:

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

By continuity of $\phi(.,s)$ 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 $i \in [\![1,\kappa]\!]$, $||g_i^{(k+l)}||_2 \leqslant L$. Then, as sequence $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ converges to $\check{x}^{(k)}$, there exists $t_1 \in \mathbb{N}$ such that for all $l \geqslant t_1$ and for all $i \in [\![1,\kappa]\!]$,

$$|g_i^{(k+l)\top}(\check{x}-x^{(k+l)})| < \frac{\delta}{3} \tag{12}$$

Moreover, as sequence $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ converges to $\check{x}^{(k)}$ and by continuity of $\phi(.,s)$, there exists $t_2 \in \mathbb{N}$ such that for all $l \geqslant t_2$ and for each index $i \in [\![1,\kappa]\!]$:

$$\sum_{s\in S_{\sigma^{(k+l)}(i)}} p_s\phi(\check{x}^{(k)},s) < \sum_{s\in S_{\sigma^{(k+l)}(i)}} p_s\phi(x^{(k+l)},s) + \frac{\delta}{3} \tag{13}$$

Then, let $t_3 = \max\{t_1, t_2\}$. Let $i \in [\![1,\kappa]\!]$ and $l_0 \geqslant t_3$ be the first indices such that (11) occurs. By combining (11), (12) and (13), 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 `cutAggr` $= False$, there exists at least one of the cuts associated with a subproblem of the batch which separates the solution to the relaxed master program.

- **Case 2:** Let $\epsilon \geqslant 0$ be the optimality gap and $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ be a sequence of elements of X converging to $\check{x}^{(k)}$ in a finite number of iterations.

  As $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ converges to $\check{x}^{(k)}$, the proof of case 1 holds also in this case for every $\epsilon > 0$. We need to prove that the proposition is true if $\epsilon = 0$. Let t_0 be the first iteration such that $x^{(k+t_0)} = \check{x}^{(k)}$. Either, for each index $i \in [\![1, \kappa]\!]$, batch $S_{\sigma^{(k+t_0)}(i)}$ is $\epsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$ with an optimality gap of 0, and by proposition 3, $x^{(k+t_0)}$ is an optimal solution to problem (1) with an optimality gap $\epsilon = 0$, or there exists a batch which is not $\epsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$, and the aggregated cut derived from this batch separates the solution to the relaxed master program.

□

A.5 Proof of Proposition 6

Proof. Let $(x,(y,z)) \in X \times \mathcal{D}$. We have:

$$\begin{array}{rcl} d_x^1 & = & \left(x, \alpha y + (1-\alpha)z\right) \\ d_x^2 & = & \left(x, \alpha x + (1-\alpha)\alpha y + (1-\alpha)^2 z\right) \end{array}$$

Let $u = \alpha y + (1-\alpha)z - x$, we have $d_x^2 = \left(x, x + (1-\alpha)u\right)$. Then, by induction,

$$\forall \ell \geqslant 2,\ d_x^\ell = \left(x, x + (1-\alpha)^{\ell-1}u\right)$$

And $\forall \ell \geqslant 2,\ \psi_2(d_x^\ell) = x + (1-\alpha)^\ell u$. Finally, $\lim\limits_{\ell\to+\infty} \psi_2\left(d_x^\ell\right) = x$.

□

A.6 Proof of Proposition 7

Proof. Let $(x,(y,z)) \in X \times \mathcal{D}$. We have:

$$\begin{array}{rcl} d_x^1 & = & \left(x + \beta(y-x), \alpha y + (1-\alpha)z\right) \\ d_x^2 & = & \left(x + \beta^2(y-x), x - (1-\alpha)x + \alpha\beta(y-x) + (1-\alpha)\alpha y + (1-\alpha)^2 z\right) \end{array}$$

We define $u = y - x$ and $v = \alpha y + (1-\alpha)z - x$. Then

$$\begin{array}{rcl} d_x^2 & = & \left(x + \beta^2 u, x + \alpha\beta u + (1-\alpha)v\right) \\ d_x^3 & = & \left(x + \beta^3 u, x + \alpha(\beta^2 + \beta(1-\alpha))u + (1-\alpha)^2 v\right) \end{array}$$

By induction, we have

$$d_x^\ell \quad = \quad \left(x + \beta^\ell u, x + \alpha\left(\sum_{i=1}^{\ell-1} \beta^i (1-\alpha)^{\ell-i-1}\right)u + (1-\alpha)^{\ell-1}v\right),\ \forall l \geqslant 2$$

We define $\delta = \max\{\beta, (1-\alpha)\}$. For all $i \geqslant 0$ and for all $l \geqslant 2$, $\beta^i \leqslant \delta^i$ and $(1-\alpha)^{l-i-1} \leqslant \delta^{l-i-1}$. Then

$$\sum_{i=1}^{\ell-1} \beta^i (1-\alpha)^{\ell-i-1} \leqslant (\ell-1)\delta^{\ell-1}$$

Then, $\lim\limits_{\ell\to+\infty} \sum_{i=1}^{\ell-1} \beta^i (1-\alpha)^{\ell-i-1} = 0$ and $\lim\limits_{\ell\to+\infty} d_x^\ell = (x,x)$. Finally, $\lim\limits_{\ell\to+\infty} \psi_2\left(d_x^\ell\right) = x$. □

B Detailed benchmark algorithms

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

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

---

Parameters: $\epsilon \geqslant 0$, $x^{(0)} \in X$, `cutAggr` $\in \{True, False\}$, $\alpha \in (0;1]$

Initialization: $k \leftarrow 0$, $\hat{x}^{(1)} \leftarrow x^{(0)}$, $UB^{(0)} \leftarrow c^\top x^{(0)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(0)})$, $LB^{(0)} \leftarrow -\infty$, $\alpha_1 \leftarrow \alpha$

while $UB^{(k)} > LB^{(k)} + \epsilon$ **do**

  $k \leftarrow k+1$

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

  $LB^{(k)} \leftarrow c^\top \check{x}^{(k)} + \sum_{s \in S} p_s \check{\theta}^{(k)}$

  $x^{(k)} \leftarrow \alpha_k \check{x}^{(k)} + (1-\alpha_k)\hat{x}^{(k)}$

  for $s \in S$ **do**

    Solve $(SP(x^{(k)}, s))$ and retrieve π_s an extreme point of Π_s

  if `cutAggr` **then**

    Add $\sum_{s \in S} p_s \theta_s \geqslant \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x)$

  else

    for $s \in S$ **do**

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

  if $UB^{(k-1)} > c^\top x^{(k)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$ **then**

    $UB^{(k)} \leftarrow c^\top x^{(k)} + \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$

    $\hat{x}^{(k+1)} \leftarrow x^{(k)}$

    $\alpha_{k+1} \leftarrow \min\{1.0, 1.2\alpha_k\}$

  else

    $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$, $UB^{(k)} \leftarrow UB^{(k-1)}$

    $\alpha_{k+1} \leftarrow \max\{0.1, 0.8\alpha_k\}$

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

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:

$$\begin{cases} \min\limits_{x,\theta} \dfrac{1}{2}||x - \hat{x}||_2^2 \\ s.t. : \displaystyle\sum_{s \in S} p_s \theta_s \geqslant \sum_{s \in S} p_s \pi_s^\top (d_s - T_s x), \ \forall s \in S, \ \forall \pi_s \in \mathrm{Vert}(\Pi_s) \\ \qquad c^\top x + \displaystyle\sum_{s \in S} p_s \theta_s \leqslant f_{lev} \\ \qquad x \in X, \theta \in \mathbb{R}^{Card(S)} \end{cases}$$

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

Algorithm 5: Level bundle method

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

Initialization: $k \leftarrow 0$, $UB^{(0)} \leftarrow c^\top x^{(0)} + \sum_{s\in S} p_s \pi_s^\top (d_s - T_s \hat{x}^{(0)})$, $\hat{x}^{(1)} \leftarrow x^{(0)}$

while $UB^{(k)} > LB^{(k)} + \epsilon$ **do**

- $k \leftarrow k+1$
- $f_{lev}^{(k)} = (1-\lambda)UB^{(k-1)} + \lambda LB^{(k-1)}$
- Solve $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$
- **if** $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$ *is infeasible* **then**
  - $LB^{(k)} \leftarrow f_{lev}(k)$
  - $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$
  - $UB^{(k)} \leftarrow UB^{(k-1)}$
- **else**
  - Retrieve $x^{(k)}$ solution to $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$
  - **for** $s \in S$ **do**
    - Solve $(SP(x^{(k)}, s))$ and retrieve π_s an extreme point of Π_s
  - Add $\sum_{s\in S} p_s \theta_s \geqslant \sum_{s\in S} p_s \pi_s^\top (d_s - T_s x)$
  - **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**
    - $UB^{(k)} \leftarrow c^\top x^{(k)} + \sum_{s\in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$
    - $\hat{x}^{(k+1)} \leftarrow x^{(k)}$
  - **else**
    - $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$
    - $UB^{(k)} \leftarrow UB^{(k-1)}$
  - $LB^{(k)} \leftarrow LB^{(k-1)}$
- $(RQMP)^{(k+1)} \leftarrow (RQMP)^{(k)}$

Return $\hat{x}^{(k+1)}$

C Impact of the stabilization on BbB - additional analysis

For 8 different instances, we show the total time spent solving the relaxed master programs and the subproblems, as well as the total number of subproblems solved for each of the following methods: **Level bundle**, **In-out monocut**, **In-out 1% CutAggr** and **BbB 1% CutAggr** $\alpha = 0.5$.

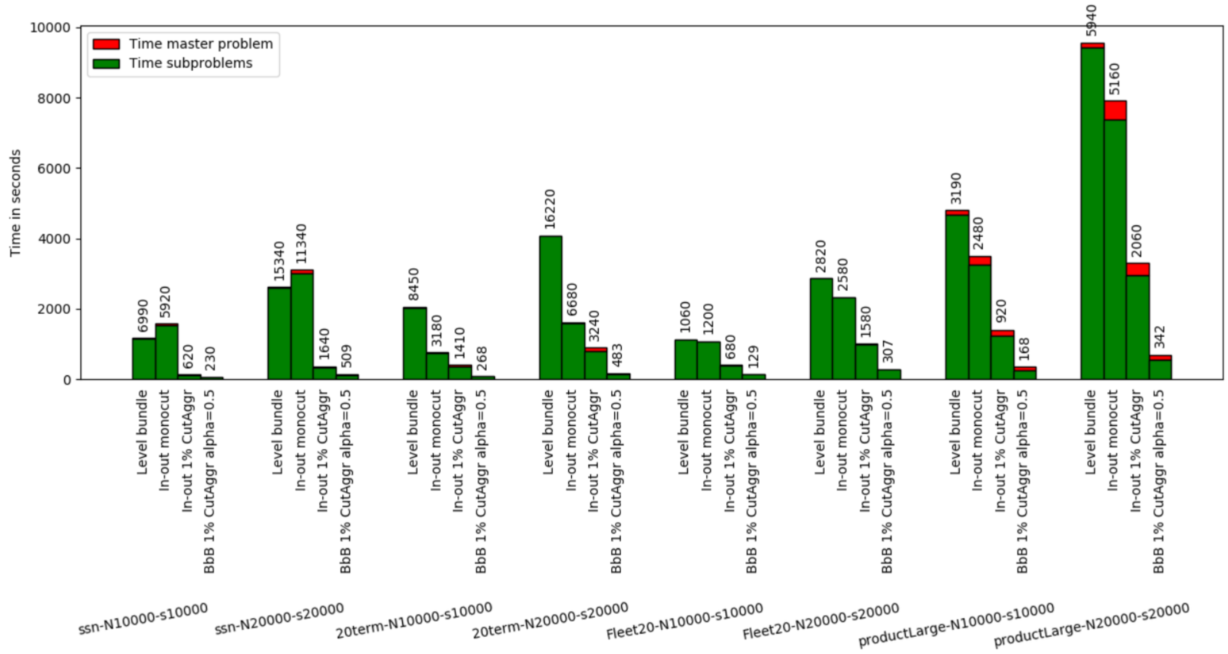


Figure 7: Time spent in solving the master program and the subproblems, for 8 different instances, solved by **Level bundle**, **In-out monocut**, **In-out 1% CutAggr** and **BbB 1% CutAggr** $\alpha = 0.5$. The total number of solved subproblems is written vertically on the top of each bar.

D Sensitivity of BbB to the optimality gap

We analyze the impact of the optimality gap on the convergence of the algorithm. The choice of a different optimality gap ϵ in the Benders by batch algorithm might have an impact on the number of batches that would be solved at each iteration. With a larger optimality gap, the algorithm tends to solve more batches at each iteration, and to add more cuts. As this might have an impact on the first-stage iterates, and then on the computing times, we show on Figure 8 the cumulative distribution of the computing times to solve our 84 instances with **BbB 1% CutAggr** $\alpha = 0.5$ with four different optimality gaps $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. The figure shows that different optimality gaps have a negligible impact on the computing times on most instances. A smaller optimality gap induces larger computing times on the largest instances of our test set, but this would also be the case with other classical algorithms.

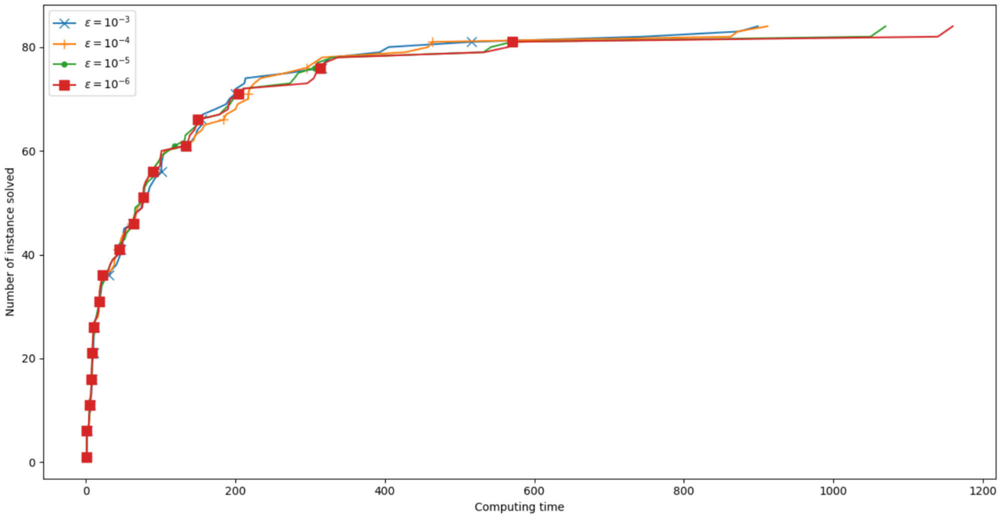


Figure 8: Cumulative distribution of the computing times on our 84 instances, for BbB with cut aggregation and base stabilization with $\alpha = 0.5$, and with optimality gaps in $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-5}\}$

E Detailed numerical results

This section gives the detailed numerical results of our experiments.

Table 7: Results for the Benders by batch algorithm without aggregation, with batch sizes from 1% to 20% of the total number of subproblems.

	Classic multicut		Classic monocut		BbB 1%		BbB 5%		BbB 10%		BbB 20%	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	2	3.2	0.81	1.3	2	2.8	0.91	1.5	0.75	1.2	**0.62**	**1.0**
LandS-N1000-s1001	2	2.9	0.72	1.0	2	2.4	0.86	1.2	0.74	1.1	**0.70**	**1.0**
LandS-N1000-s1002	2	3.0	0.72	1.1	2	2.9	0.71	1.1	**0.65**	**1.0**	0.66	1.0
LandS-N5000-s5000	11	1.6	9	1.3	12	1.9	8	1.2	7	1.1	**7**	**1.0**
LandS-N5000-s5001	10	1.6	10	1.6	15	2.5	8	1.3	6	1.1	**6**	**1.0**
LandS-N5000-s5002	11	1.9	9	1.5	13	2.2	8	1.3	7	1.2	**6**	**1.0**
LandS-N10000-s10000	22	1.1	26	1.3	41	2.0	25	1.2	**20**	**1.0**	21	1.0
LandS-N10000-s10001	22	1.1	30	1.5	36	1.8	25	1.2	22	1.1	**20**	**1.0**
LandS-N10000-s10002	20	1.1	30	1.7	37	2.0	25	1.4	22	1.2	**18**	**1.0**
LandS-N20000-s20000	**49**	**1.0**	96	1.9	134	2.7	86	1.7	78	1.6	71	1.4
LandS-N20000-s20001	**43**	**1.0**	119	2.8	130	3.0	92	2.1	77	1.8	71	1.7
LandS-N20000-s20002	**44**	**1.0**	99	2.2	125	2.8	90	2.0	85	1.9	73	1.7
gbd-N1000-s1000	2	2.7	0.95	1.4	2	3.3	**0.68**	**1.0**	0.78	1.1	0.95	1.4
gbd-N1000-s1001	2	3.7	0.90	1.4	2	3.8	**0.65**	**1.0**	0.90	1.4	0.94	1.5
gbd-N1000-s1002	2	3.6	0.96	1.6	2	3.7	**0.62**	**1.0**	0.83	1.3	0.99	1.6
gbd-N5000-s5000	13	2.0	10	1.7	18	2.9	**6**	**1.0**	7	1.2	8	1.4
gbd-N5000-s5001	11	1.9	10	1.7	14	2.3	**6**	**1.0**	7	1.1	8	1.3
gbd-N5000-s5002	12	1.8	11	1.6	15	2.4	**6**	**1.0**	7	1.1	9	1.3
gbd-N10000-s10000	24	1.2	34	1.8	54	2.8	**19**	**1.0**	21	1.1	26	1.4
gbd-N10000-s10001	24	1.3	32	1.7	41	2.2	**19**	**1.0**	24	1.3	26	1.4
gbd-N10000-s10002	23	1.2	32	1.7	46	2.4	**19**	**1.0**	22	1.1	24	1.2
gbd-N20000-s20000	**48**	**1.0**	119	2.5	97	2.0	63	1.3	71	1.5	86	1.8
gbd-N20000-s20001	**51**	**1.0**	120	2.3	100	2.0	64	1.2	73	1.4	90	1.8
gbd-N20000-s20002	**47**	**1.0**	125	2.7	92	2.0	57	1.2	70	1.5	85	1.8
ssn-N1000-s1000	2279	552.2	7	1.7	6	1.3	**4**	**1.0**	5	1.1	5	1.2
ssn-N1000-s1001	2720	679.7	7	1.8	6	1.6	**4**	**1.0**	4	1.0	5	1.2
ssn-N1000-s1002	2226	602.8	7	1.8	6	1.8	**4**	**1.0**	4	1.1	5	1.3
ssn-N5000-s5000	13425	580.9	62	2.7	31	1.3	**23**	**1.0**	33	1.4	33	1.4
ssn-N5000-s5001	14260	631.1	45	2.0	33	1.5	**23**	**1.0**	27	1.2	31	1.4
ssn-N5000-s5002	12695	558.4	64	2.8	31	1.4	25	1.1	**23**	**1.0**	31	1.4
ssn-N10000-s10000	26559	420.0	185	2.9	**63**	**1.0**	123	2.0	64	1.0	79	1.3
ssn-N10000-s10001	26228	449.1	193	3.3	72	1.2	**58**	**1.0**	59	1.0	78	1.3
ssn-N10000-s10002	24916	463.1	187	3.5	80	1.5	56	1.0	**54**	**1.0**	79	1.5
ssn-N20000-s20000	$+\infty$	>382.6	512	4.5	152	1.3	**113**	**1.0**	120	1.1	8143	72.1
ssn-N20000-s20001	$+\infty$	>355.0	503	4.1	**122**	**1.0**	588	4.8	128	1.1	167	1.4
ssn-N20000-s20002	$+\infty$	>356.6	450	3.7	160	1.3	**121**	**1.0**	1624	13.4	154	1.3
storm-N1000-s1000	23	3.6	10	1.6	19	3.0	8	1.3	**6**	**1.0**	8	1.3
storm-N1000-s1001	24	3.7	11	1.6	23	3.5	8	1.3	**7**	**1.0**	8	1.3
storm-N1000-s1002	24	3.8	11	1.7	21	3.3	8	1.3	**6**	**1.0**	8	1.3
storm-N5000-s5000	110	2.0	100	1.8	159	2.9	58	1.1	**54**	**1.0**	65	1.2
storm-N5000-s5001	117	2.2	118	2.2	184	3.4	59	1.1	**54**	**1.0**	65	1.2
storm-N5000-s5002	116	2.1	99	1.8	181	3.3	63	1.1	**55**	**1.0**	65	1.2
storm-N10000-s10000	215	1.4	468	3.0	508	3.2	162	1.0	**159**	**1.0**	191	1.2
storm-N10000-s10001	225	1.5	479	3.1	494	3.2	**154**	**1.0**	161	1.1	188	1.2
storm-N10000-s10002	233	1.5	542	3.5	474	3.1	**153**	**1.0**	157	1.0	189	1.2
storm-N20000-s20000	**465**	**1.0**	2240	4.8	1470	3.2	581	1.2	704	1.5	574	1.2
storm-N20000-s20001	**434**	**1.0**	2460	5.7	1300	3.0	585	1.3	669	1.5	603	1.4
storm-N20000-s20002	**476**	**1.0**	2410	5.1	1400	2.9	574	1.2	642	1.3	587	1.2
20term-N1000-s1000	544	13.5	749	18.6	**40**	**1.0**	82	2.0	46	1.1	74	1.8
20term-N1000-s1001	584	16.1	646	17.8	**36**	**1.0**	82	2.3	47	1.3	72	2.0
20term-N1000-s1002	604	16.0	877	23.2	**38**	**1.0**	82	2.2	53	1.4	76	2.0
20term-N5000-s5000	3095	4.7	29455	44.6	**660**	**1.0**	2059	3.1	1497	2.3	1951	3.0
20term-N5000-s5001	3699	5.4	22490	33.0	**681**	**1.0**	2066	3.0	1333	2.0	2302	3.4
20term-N5000-s5002	3725	6.6	21342	38.0	**561**	**1.0**	2178	3.9	1176	2.1	2486	4.4
20term-N10000-s10000	6803	3.1	$+\infty$	>20.4	**2193**	**1.0**	9654	4.4	5526	2.5	11592	5.3
20term-N10000-s10001	6404	2.7	$+\infty$	>19.5	**2330**	**1.0**	11062	4.7	7874	3.4	9436	4.1
20term-N10000-s10002	7494	3.3	$+\infty$	>19.6	**2288**	**1.0**	11483	5.0	5196	2.3	10212	4.5
20term-N20000-s20000	**13429**	**1.0**	$+\infty$	>5.7	$+\infty$	>3.2	$+\infty$	>3.2	$+\infty$	>3.2	$+\infty$	>3.2
20term-N20000-s20001	12763	1.4	$+\infty$	>5.0	**9062**	**1.0**	$+\infty$	>4.8	$+\infty$	>4.8	$+\infty$	>4.8
20term-N20000-s20002	14868	1.5	$+\infty$	>8.1	**9613**	**1.0**	$+\infty$	>4.5	$+\infty$	>4.6	$+\infty$	>4.6
Fleet20_3-N1000-s1000	513	9.4	224	4.1	143	2.6	105	1.9	102	1.9	**55**	**1.0**
Fleet20_3-N1000-s1001	539	10.1	228	4.3	139	2.6	110	2.1	100	1.9	**53**	**1.0**
Fleet20_3-N1000-s1002	546	7.7	224	3.2	154	2.2	**70**	**1.0**	103	1.5	115	1.6
Fleet20_3-N5000-s5000	2780	1.5	5530	2.9	2380	1.3	2050	1.1	**1880**	**1.0**	2110	1.1
Fleet20_3-N5000-s5001	2760	1.5	5090	2.8	2260	1.2	**1850**	**1.0**	1870	1.0	2070	1.1
Fleet20_3-N5000-s5002	2730	1.5	5370	2.9	2610	1.4	1950	1.0	**1870**	**1.0**	2110	1.1
Fleet20_3-N10000-s10000	**5860**	**1.0**	29600	5.1	10400	1.8	$+\infty$	>7.4	8780	1.5	11000	1.9
Fleet20_3-N10000-s10001	**5480**	**1.0**	28200	5.1	8310	1.5	8350	1.5	8560	1.6	9950	1.8
Fleet20_3-N10000-s10002	**5790**	**1.0**	29000	5.0	11000	1.9	8190	1.4	8270	1.4	$+\infty$	>7.5
Fleet20_3-N20000-s20000	**11400**	**1.0**	$+\infty$	>4.0	$+\infty$	>3.8	$+\infty$	>3.8	$+\infty$	>3.8	$+\infty$	>3.9
Fleet20_3-N20000-s20001	**11500**	**1.0**	$+\infty$	>3.8	18200	1.6	$+\infty$	>3.8	$+\infty$	>3.8	$+\infty$	>3.8
Fleet20_3-N20000-s20002	**11000**	**1.0**	$+\infty$	>4.6	$+\infty$	>3.9	$+\infty$	>3.9	$+\infty$	>3.9	$+\infty$	>4.0
product-N1000-s1000	1920	17.9	184	1.7	259	2.4	123	1.1	109	1.0	**107**	**1.0**
product-N1000-s1001	2070	19.9	197	1.9	302	2.9	125	1.2	109	1.0	**104**	**1.0**
product-N1000-s1002	1850	19.1	178	1.8	249	2.6	120	1.2	**97**	**1.0**	**97**	**1.0**
product-N5000-s5000	10500	8.0	3220	2.5	3630	2.8	1830	1.4	1390	1.1	**1310**	**1.0**
product-N5000-s5001	10100	7.4	3440	2.5	3830	2.8	1700	1.2	1480	1.1	**1360**	**1.0**
product-N5000-s5002	10800	7.4	3830	2.6	3730	2.6	2090	1.4	1580	1.1	**1460**	**1.0**
product-N10000-s10000	20200	3.6	15300	2.7	14000	2.5	7330	1.3	5820	1.0	**5580**	**1.0**
product-N10000-s10001	19100	3.7	13300	2.5	11800	2.3	6580	1.3	5560	1.1	**5230**	**1.0**
product-N10000-s10002	21300	4.0	17000	3.2	14100	2.6	6770	1.3	**5370**	**1.0**	5380	1.0
product-N20000-s20000	$+\infty$	>1.7	$+\infty$	>2.0	$+\infty$	>1.7	32700	1.3	26000	1.0	**25200**	**1.0**
product-N20000-s20001	42600	2.1	$+\infty$	>2.2	$+\infty$	>2.2	26600	1.3	24100	1.2	**20000**	**1.0**
product-N20000-s20002	$+\infty$	>1.8	$+\infty$	>1.8	$+\infty$	>1.8	29800	1.2	24100	1.0	**24000**	**1.0**

Table 8: Results for the Benders by batch algorithm with aggregation, with batch sizes from 1% to 20% of the total number of subproblems.

	Classic monocut		Classic 1% CutAggr		Classic 5% CutAggr		BbB 1% CutAggr		BbB 5% CutAggr		BbB 10% CutAggr		BbB 20% CutAggr	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	2	2.6	0.94	1.2	1	1.6	2	2.2	0.89	1.2	**0.77**	**1.0**	0.86	1.1
LandS-N1000-s1001	2	2.7	1.00	1.3	1	1.8	2	2.1	0.75	1.0	**0.75**	**1.0**	0.90	1.2
LandS-N1000-s1002	2	2.3	1	1.3	1	1.7	2	2.0	0.99	1.2	**0.84**	**1.0**	0.91	1.1
LandS-N5000-s5000	11	2.7	7	1.8	8	2.0	10	2.6	5	1.2	**4**	**1.0**	4	1.1
LandS-N5000-s5001	10	2.3	7	1.6	9	2.0	9	2.1	5	1.2	**4**	**1.0**	4	1.0
LandS-N5000-s5002	11	2.9	7	1.9	8	2.1	9	2.2	**4**	**1.0**	4	1.0	4	1.1
LandS-N10000-s10000	22	2.7	16	1.9	18	2.2	17	2.0	**8**	**1.0**	9	1.1	9	1.1
LandS-N10000-s10001	22	2.8	16	2.0	20	2.5	14	1.8	8	1.0	**8**	**1.0**	9	1.2
LandS-N10000-s10002	20	2.6	16	2.0	18	2.2	17	2.1	8	1.0	**8**	**1.0**	9	1.2
LandS-N20000-s20000	49	3.0	34	2.0	39	2.3	45	2.7	**17**	**1.0**	18	1.1	19	1.2
LandS-N20000-s20001	43	2.4	35	1.9	39	2.2	42	2.4	**18**	**1.0**	18	1.0	21	1.2
LandS-N20000-s20002	44	2.6	32	1.9	40	2.3	45	2.6	18	1.0	**17**	**1.0**	19	1.1
gbd-N1000-s1000	2	3.5	1	2.4	2	3.1	2	2.9	**0.53**	**1.0**	0.68	1.3	0.89	1.7
gbd-N1000-s1001	2	3.6	1	1.6	2	2.5	2	2.4	**0.67**	**1.0**	0.99	1.5	1	1.5
gbd-N1000-s1002	2	3.6	1	1.9	2	2.5	2	3.0	**0.61**	**1.0**	0.68	1.1	0.88	1.4
gbd-N5000-s5000	13	3.8	8	2.4	11	3.2	10	3.0	**3**	**1.0**	4	1.1	4	1.3
gbd-N5000-s5001	11	3.6	9	2.8	10	3.2	8	2.5	**3**	**1.0**	4	1.1	4	1.4
gbd-N5000-s5002	12	3.4	9	2.6	9	2.7	9	2.6	**3**	**1.0**	4	1.1	5	1.3
gbd-N10000-s10000	24	3.4	18	2.5	21	2.9	18	2.5	**7**	**1.0**	8	1.1	9	1.2
gbd-N10000-s10001	24	4.0	19	3.3	19	3.2	13	2.1	**6**	**1.0**	8	1.4	9	1.5
gbd-N10000-s10002	23	3.8	20	3.4	23	3.9	14	2.3	**6**	**1.0**	8	1.4	11	1.8
gbd-N20000-s20000	48	3.8	39	3.2	47	3.7	50	4.0	**12**	**1.0**	16	1.3	20	1.6
gbd-N20000-s20001	51	3.6	42	3.0	45	3.2	31	2.2	15	1.1	**14**	**1.0**	19	1.4
gbd-N20000-s20002	47	3.4	41	3.0	45	3.3	43	3.2	**14**	**1.0**	14	1.0	19	1.4
ssn-N1000-s1000	2279	168.5	25	1.9	146	10.8	**14**	**1.0**	63	4.6	129	9.5	235	17.4
ssn-N1000-s1001	2720	185.6	24	1.7	135	9.2	**15**	**1.0**	63	4.3	130	8.8	253	17.3
ssn-N1000-s1002	2226	173.3	23	1.8	146	11.4	**13**	**1.0**	59	4.6	144	11.2	238	18.5
ssn-N5000-s5000	13425	152.4	371	4.2	1685	19.1	**88**	**1.0**	337	3.8	630	7.2	1342	15.2
ssn-N5000-s5001	14260	158.7	411	4.6	1536	17.1	**90**	**1.0**	322	3.6	672	7.5	1343	15.0
ssn-N5000-s5002	12695	140.6	416	4.6	1524	16.9	**90**	**1.0**	308	3.4	674	7.5	1280	14.2
ssn-N10000-s10000	26559	151.5	1212	6.9	3343	19.1	**175**	**1.0**	672	3.8	1396	8.0	2771	15.8
ssn-N10000-s10001	26228	140.6	1378	7.4	6126	32.8	**187**	**1.0**	760	4.1	1477	7.9	3143	16.8
ssn-N10000-s10002	24916	129.1	1147	5.9	5105	26.4	**193**	**1.0**	690	3.6	1397	7.2	2827	14.6
ssn-N20000-s20000	$+\infty$	>94.6	7066	15.5	18068	39.6	**457**	**1.0**	1651	3.6	3463	7.6	6588	14.4
ssn-N20000-s20001	$+\infty$	>94.3	5558	12.1	40319	88.0	**458**	**1.0**	1651	3.6	3065	6.7	6749	14.7
ssn-N20000-s20002	$+\infty$	>106.2	13186	32.4	19979	49.1	**407**	**1.0**	1543	3.8	3630	8.9	6934	17.0
storm-N1000-s1000	23	3.7	12	2.0	15	2.4	12	1.9	**6**	**1.0**	7	1.1	10	1.6
storm-N1000-s1001	24	3.8	12	1.9	16	2.5	12	1.9	**6**	**1.0**	7	1.1	9	1.4
storm-N1000-s1002	24	3.7	13	2.0	15	2.3	13	2.0	**6**	**1.0**	7	1.1	9	1.4
storm-N5000-s5000	110	3.3	73	2.2	92	2.8	44	1.3	**33**	**1.0**	35	1.1	54	1.6
storm-N5000-s5001	117	3.6	72	2.2	97	3.0	54	1.6	**33**	**1.0**	36	1.1	56	1.7
storm-N5000-s5002	116	3.2	72	2.0	93	2.6	58	1.6	37	1.0	**36**	**1.0**	55	1.5
storm-N10000-s10000	215	3.0	157	2.2	202	2.8	121	1.7	**73**	**1.0**	82	1.1	105	1.4
storm-N10000-s10001	225	3.0	169	2.2	198	2.6	90	1.2	**76**	**1.0**	83	1.1	101	1.3
storm-N10000-s10002	233	3.2	166	2.3	194	2.7	118	1.6	**73**	**1.0**	80	1.1	107	1.5
storm-N20000-s20000	465	2.9	370	2.3	434	2.7	216	1.3	167	1.0	**161**	**1.0**	232	1.4
storm-N20000-s20001	434	2.7	380	2.4	413	2.6	245	1.5	**161**	**1.0**	179	1.1	246	1.5
storm-N20000-s20002	476	3.0	356	2.2	422	2.6	218	1.4	**160**	**1.0**	167	1.0	236	1.5
20term-N1000-s1000	544	36.7	272	18.4	310	20.9	**15**	**1.0**	36	2.5	71	4.8	140	9.5
20term-N1000-s1001	584	40.0	239	16.4	266	18.2	**15**	**1.0**	37	2.5	67	4.6	135	9.3
20term-N1000-s1002	604	41.4	305	20.9	364	25.0	**15**	**1.0**	37	2.5	65	4.5	148	10.2
20term-N5000-s5000	3095	46.0	1627	24.2	2026	30.1	**67**	**1.0**	199	3.0	401	6.0	830	12.4
20term-N5000-s5001	3699	47.2	1453	18.5	1911	24.4	**78**	**1.0**	197	2.5	381	4.9	794	10.1
20term-N5000-s5002	3725	57.8	1733	26.9	1898	29.5	**64**	**1.0**	182	2.8	404	6.3	893	13.9
20term-N10000-s10000	6803	52.5	3885	30.0	4741	36.6	**129**	**1.0**	411	3.2	892	6.9	1874	14.5
20term-N10000-s10001	6404	52.5	3193	26.2	4915	40.3	**122**	**1.0**	409	3.3	914	7.5	1970	16.1
20term-N10000-s10002	7494	54.5	3015	21.9	4864	35.4	**137**	**1.0**	388	2.8	886	6.4	2089	15.2
20term-N20000-s20000	13429	51.5	7375	28.3	10772	41.3	**261**	**1.0**	860	3.3	1913	7.3	7032	27.0
20term-N20000-s20001	12763	43.2	7433	25.1	26284	88.9	**296**	**1.0**	985	3.3	2139	7.2	4704	15.9
20term-N20000-s20002	14868	52.5	6287	22.2	11803	41.7	**283**	**1.0**	897	3.2	2101	7.4	$+\infty$	>152.6
Fleet20_3-N1000-s1000	513	18.6	123	4.5	221	8.0	**28**	**1.0**	42	1.5	71	2.6	127	4.6
Fleet20_3-N1000-s1001	539	20.0	126	4.7	219	8.1	**27**	**1.0**	40	1.5	73	2.7	131	4.9
Fleet20_3-N1000-s1002	546	18.2	126	4.2	225	7.5	**30**	**1.0**	43	1.4	77	2.6	135	4.5
Fleet20_3-N5000-s5000	2780	25.7	905	8.4	1570	14.5	**108**	**1.0**	218	2.0	354	3.3	675	6.2
Fleet20_3-N5000-s5001	2760	26.5	930	8.9	1500	14.4	**104**	**1.0**	209	2.0	363	3.5	645	6.2
Fleet20_3-N5000-s5002	2730	24.8	873	7.9	1520	13.8	**110**	**1.0**	205	1.9	356	3.2	628	5.7
Fleet20_3-N10000-s10000	5860	27.4	2030	9.5	3430	16.0	**214**	**1.0**	426	2.0	725	3.4	1290	6.0
Fleet20_3-N10000-s10001	5480	26.2	1960	9.4	3520	16.8	**209**	**1.0**	467	2.2	721	3.4	1290	6.2
Fleet20_3-N10000-s10002	5790	27.2	2010	9.4	3430	16.1	**213**	**1.0**	426	2.0	716	3.4	1350	6.3
Fleet20_3-N20000-s20000	11400	28.4	5200	12.9	8040	20.0	**402**	**1.0**	886	2.2	1510	3.8	2810	7.0
Fleet20_3-N20000-s20001	11500	26.8	4820	11.2	7690	17.9	**429**	**1.0**	856	2.0	1490	3.5	2750	6.4
Fleet20_3-N20000-s20002	11000	25.9	5140	12.1	7850	18.5	**425**	**1.0**	885	2.1	1560	3.7	2770	6.5
product-N1000-s1000	1920	18.5	191	1.8	415	4.0	**104**	**1.0**	140	1.3	246	2.4	471	4.5
product-N1000-s1001	2070	21.3	197	2.0	452	4.7	**97**	**1.0**	149	1.5	266	2.7	528	5.4
product-N1000-s1002	1850	20.2	182	2.0	425	4.6	**91**	**1.0**	135	1.5	247	2.7	515	5.6
product-N5000-s5000	10500	29.8	1530	4.3	3290	9.3	**352**	**1.0**	734	2.1	1550	4.4	3180	9.0
product-N5000-s5001	10100	29.3	1460	4.2	3250	9.4	**345**	**1.0**	787	2.3	1420	4.1	2580	7.5
product-N5000-s5002	10800	27.7	1580	4.1	3430	8.8	**390**	**1.0**	797	2.0	1730	4.4	2860	7.3
product-N10000-s10000	20200	28.7	3830	5.4	8170	11.6	**704**	**1.0**	1620	2.3	2980	4.2	5670	8.1
product-N10000-s10001	19100	25.3	3910	5.2	7480	9.9	**756**	**1.0**	1400	1.9	2980	3.9	5140	6.8
product-N10000-s10002	21300	21.1	3740	3.7	7620	7.5	**1010**	**1.0**	1550	1.5	3200	3.2	5780	5.7
product-N20000-s20000	$+\infty$	>24.1	9820	5.5	19300	10.8	**1790**	**1.0**	3330	1.9	6740	3.8	13500	7.5
product-N20000-s20001	42600	23.3	9670	5.3	19200	10.5	**1830**	**1.0**	3230	1.8	5950	3.3	11500	6.3
product-N20000-s20002	$+\infty$	>29.6	10400	7.1	19600	13.4	**1460**	**1.0**	3540	2.4	6270	4.3	12500	8.6

Table 9: Detailed results for the Benders by batch algorithm, with a batch size of 1%, cut aggregation, and stabilization (basic or solution memory) compared to without stabilization

	BbB 1% CutAggr		BbB 1% CutAggr $\alpha = 0.1$		BbB 1% CutAggr $\alpha = 0.5$		BbB 1% CutAggr $\alpha = 0.9$		BbB 1% CutAggr $\alpha = 0.1$ $\beta = 0.1$		BbB 1% CutAggr $\alpha = 0.1$ $\beta = 0.5$		BbB 1% CutAggr $\alpha = 0.1$ $\beta = 0.9$		BbB 1% CutAggr $\alpha = 0.5$ $\beta = 0.1$		BbB 1% CutAggr $\alpha = 0.5$ $\beta = 0.5$		BbB 1% CutAggr $\alpha = 0.5$ $\beta = 0.9$		BbB 1% CutAggr $\alpha = 0.9$ $\beta = 0.1$		BbB 1% CutAggr $\alpha = 0.9$ $\beta = 0.5$		BbB 1% CutAggr $\alpha = 0.9$ $\beta = 0.9$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	2	2.0	1	1.6	1.00	1.2	1	1.3	1	1.8	2	1.8	**0.85**	**1.0**	0.98	1.2	0.95	1.1	1	1.8	1	1.4	0.95	1.1	1	1.7
LandS-N1000-s1001	2	1.7	2	1.8	1	1.1	1	1.4	2	1.7	2	1.6	2	1.8	0.97	1.1	**0.92**	**1.0**	2	1.6	1	1.2	1	1.1	2	1.7
LandS-N1000-s1002	2	2.1	0.92	1.1	0.88	1.1	1	1.4	0.83	1.0	**0.83**	**1.0**	0.90	1.1	0.85	1.0	0.88	1.1	0.92	1.1	1	1.5	0.92	1.1	0.90	1.1
LandS-N5000-s5000	10	2.3	7	1.7	**4**	**1.0**	5	1.2	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	9	2.3	5	1.1	5	1.3	6	1.6	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	9	1.9	7	1.6	5	1.1	5	1.1	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	17	2.0	**9**	**1.0**	9	1.1	12	1.4	9	1.0	15	1.8	15	1.8	10	1.2	10	1.2	15	1.8	10	1.2	10	1.2	9	1.0
LandS-N10000-s10001	14	1.5	15	1.6	**9**	**1.0**	13	1.3	15	1.6	16	1.6	16	1.7	11	1.1	16	1.7	15	1.6	10	1.1	11	1.1	16	1.6
LandS-N10000-s10002	17	2.0	9	1.0	9	1.0	10	1.1	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	45	2.4	30	1.6	21	1.1	21	1.1	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	42	2.4	31	1.8	21	1.2	21	1.2	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	45	2.7	29	1.7	20	1.2	29	1.7	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	2	1.9	1	1.8	**0.81**	**1.0**	0.99	1.2	2	1.9	2	1.9	2	2.4	0.90	1.1	0.96	1.2	2	1.9	1	1.7	0.85	1.0	2	1.9
gbd-N1000-s1001	2	2.0	2	2.2	1	1.3	1	1.8	2	2.0	2	2.0	2	2.3	0.88	1.1	0.83	1.1	2	1.9	1	1.3	**0.79**	**1.0**	2	2.2
gbd-N1000-s1002	2	2.3	1	1.9	0.82	1.0	1	1.6	1	1.9	2	2.0	2	2.4	0.81	1.0	0.80	1.0	2	2.0	1	1.3	**0.79**	**1.0**	1	1.9
gbd-N5000-s5000	10	2.7	7	2.0	**4**	**1.0**	6	1.7	7	1.9	8	2.1	9	2.4	4	1.2	5	1.4	8	2.1	6	1.7	4	1.2	7	1.9
gbd-N5000-s5001	8	2.3	8	2.2	4	1.1	**3**	**1.0**	7	2.2	8	2.2	8	2.4	5	1.4	4	1.1	8	2.2	4	1.0	5	1.4	7	2.2
gbd-N5000-s5002	9	2.6	9	2.4	5	1.4	4	1.2	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	18	2.4	14	1.8	**7**	**1.0**	10	1.3	15	2.0	15	2.0	16	2.2	10	1.4	9	1.2	15	2.0	8	1.1	10	1.4	15	2.0
gbd-N10000-s10001	13	1.7	14	1.9	**7**	**1.0**	9	1.2	13	1.8	14	2.0	17	2.3	11	1.4	8	1.1	14	2.0	10	1.4	10	1.4	14	1.9
gbd-N10000-s10002	14	2.0	14	2.1	8	1.2	7	1.1	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	50	3.5	52	3.6	19	1.3	19	1.3	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	31	1.9	26	1.6	**17**	**1.0**	22	1.3	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	43	3.0	27	1.9	15	1.0	**14**	**1.0**	30	2.1	29	2.0	32	2.3	20	1.4	18	1.2	30	2.1	16	1.1	20	1.4	30	2.1
ssn-N1000-s1000	14	1.8	10	1.4	**8**	**1.0**	11	1.4	9	1.2	10	1.3	11	1.4	9	1.2	9	1.2	11	1.4	10	1.3	8	1.1	9	1.2
ssn-N1000-s1001	15	1.9	9	1.2	8	1.1	13	1.6	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	13	1.6	8	1.0	**8**	**1.0**	11	1.3	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	88	2.0	51	1.1	47	1.1	70	1.6	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	90	2.0	48	1.0	**46**	**1.0**	65	1.4	49	1.1	52	1.1	60	1.3	47	1.0	46	1.0	53	1.2	62	1.4	46	1.0	49	1.1
ssn-N5000-s5002	90	2.0	51	1.1	49	1.1	68	1.5	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	175	2.1	117	1.4	**84**	**1.0**	126	1.5	101	1.2	108	1.3	120	1.4	92	1.1	95	1.1	113	1.3	115	1.4	92	1.1	106	1.3
ssn-N10000-s10001	187	2.0	112	1.2	98	1.1	129	1.4	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	193	2.2	101	1.2	90	1.0	134	1.6	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	457	2.5	221	1.2	**181**	**1.0**	279	1.5	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	458	2.5	207	1.1	190	1.0	284	1.6	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	407	2.1	215	1.1	201	1.1	305	1.6	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	12	1.9	9	1.5	8	1.2	7	1.2	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	12	2.0	7	1.1	7	1.2	8	1.4	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	13	2.0	9	1.5	7	1.0	8	1.2	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	44	1.4	33	1.1	32	1.1	37	1.2	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	54	1.7	47	1.5	35	1.1	37	1.2	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	58	1.9	33	1.1	**30**	**1.0**	32	1.1	34	1.1	32	1.1	33	1.1	32	1.0	33	1.1	32	1.1	37	1.2	32	1.1	34	1.1
storm-N10000-s10000	121	2.0	65	1.1	64	1.1	81	1.4	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	90	1.4	68	1.1	**64**	**1.0**	68	1.1	68	1.1	66	1.0	108	1.7	67	1.1	66	1.0	67	1.0	71	1.1	65	1.0	68	1.1
storm-N10000-s10002	118	1.9	66	1.1	66	1.1	98	1.6	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	216	1.7	141	1.1	139	1.1	162	1.3	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	245	2.0	134	1.1	137	1.1	127	1.0	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	218	1.7	145	1.1	135	1.0	137	1.1	**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	15	1.7	13	1.5	11	1.3	14	1.6	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	15	1.5	10	1.1	10	1.0	13	1.4	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	15	1.6	11	1.3	10	1.2	14	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	67	1.3	60	1.2	52	1.0	64	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	78	1.8	64	1.5	**43**	**1.0**	57	1.3	67	1.5	67	1.5	84	1.9	51	1.2	46	1.1	74	1.7	58	1.3	48	1.1	74	1.7
20term-N5000-s5002	64	1.4	54	1.2	54	1.2	58	1.3	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	129	1.3	114	1.1	101	1.0	116	1.2	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	122	1.3	148	1.6	101	1.1	114	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	137	1.6	138	1.6	100	1.2	125	1.5	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	261	1.4	279	1.5	**191**	**1.0**	258	1.4	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	296	1.4	311	1.5	**210**	**1.0**	256	1.2	289	1.4	337	1.6	326	1.6	241	1.2	243	1.2	272	1.3	236	1.1	224	1.1	267	1.3
20term-N20000-s20002	283	1.8	**159**	**1.0**	191	1.2	270	1.7	237	1.5	178	1.1	361	2.3	233	1.5	212	1.3	288	1.8	254	1.6	230	1.4	337	2.1
Fleet20_3-N1000-s1000	28	1.7	20	1.2	**17**	**1.0**	19	1.1	18	1.1	21	1.3	24	1.4	17	1.0	18	1.1	22	1.3	19	1.1	17	1.0	19	1.1
Fleet20_3-N1000-s1001	27	1.6	**17**	**1.0**	17	1.0	20	1.2	18	1.1	20	1.2	24	1.4	17	1.0	18	1.0	22	1.3	19	1.1	18	1.1	18	1.1
Fleet20_3-N1000-s1002	30	1.7	20	1.2	18	1.0	21	1.2	21	1.2	21	1.2	27	1.5	19	1.1	18	1.0	22	1.2	20	1.2	**18**	**1.0**	18	1.1
Fleet20_3-N5000-s5000	108	1.4	86	1.2	**75**	**1.0**	83	1.1	89	1.2	96	1.3	125	1.7	78	1.0	84	1.1	95	1.3	83	1.1	76	1.0	89	1.2
Fleet20_3-N5000-s5001	104	1.4	95	1.3	77	1.0	84	1.1	90	1.2	94	1.2	135	1.8	**76**	**1.0**	81	1.1	105	1.4	80	1.1	78	1.0	103	1.4
Fleet20_3-N5000-s5002	110	1.5	96	1.3	**74**	**1.0**	84	1.1	93	1.3	101	1.4	137	1.9	74	1.0	79	1.1	107	1.4	82	1.1	77	1.0	101	1.4
Fleet20_3-N10000-s10000	214	1.5	172	1.2	**147**	**1.0**	163	1.1	184	1.3	197	1.3	270	1.8	155	1.1	159	1.1	191	1.3	163	1.1	152	1.0	174	1.2
Fleet20_3-N10000-s10001	209	1.4	196	1.3	**148**	**1.0**	167	1.1	183	1.2	193	1.3	240	1.6	155	1.0	164	1.1	193	1.3	180	1.2	156	1.1	175	1.2
Fleet20_3-N10000-s10002	213	1.5	192	1.4	144	1.0	161	1.1	206	1.5	220	1.5	275	1.9	154	1.1	163	1.1	213	1.5	163	1.1	**142**	**1.0**	181	1.3
Fleet20_3-N20000-s20000	402	1.3	358	1.2	**302**	**1.0**	347	1.1	426	1.4	482	1.6	557	1.8	307	1.0	327	1.1	434	1.4	340	1.1	315	1.0	401	1.3
Fleet20_3-N20000-s20001	429	1.4	391	1.3	310	1.0	333	1.1	364	1.2	416	1.4	534	1.8	322	1.1	332	1.1	460	1.5	340	1.1	**301**	**1.0**	422	1.4
Fleet20_3-N20000-s20002	425	1.4	424	1.4	317	1.0	355	1.2	385	1.3	462	1.5	564	1.8	311	1.0	333	1.1	460	1.5	337	1.1	**305**	**1.0**	389	1.3
product-N1000-s1000	104	1.4	87	1.2	76	1.0	89	1.2	96	1.3	91	1.2	124	1.6	85	1.1	**76**	**1.0**	92	1.2	88	1.2	80	1.1	95	1.3
product-N1000-s1001	97	1.2	88	1.1	**78**	**1.0**	79	1.0	83	1.1	81	1.0	145	1.9	89	1.1	82	1.1	79	1.0	85	1.1	88	1.1	83	1.1
product-N1000-s1002	91	1.2	84	1.1	75	1.0	83	1.1	82	1.1	75	1.0	110	1.5	76	1.0	77	1.0	**75**	**1.0**	79	1.1	76	1.0	81	1.1
product-N5000-s5000	352	1.3	**264**	**1.0**	305	1.2	312	1.2	266	1.0	306	1.2	382	1.4	290	1.1	296	1.1	294	1.1	331	1.3	291	1.1	288	1.1
product-N5000-s5001	345	1.3	**269**	**1.0**	335	1.2	303	1.1	290	1.1	293	1.1	396	1.5	303	1.1	286	1.1	293	1.1	315	1.2	281	1.0	288	1.1
product-N5000-s5002	390	1.4	285	1.0	295	1.0	301	1.1	298	1.1	308	1.1	400	1.4	287	1.0	**283**	**1.0**	305	1.1	306	1.1	286	1.0	298	1.1
product-N10000-s10000	704	1.3	**522**	**1.0**	565	1.1	630	1.2	549	1.1	636	1.2	824	1.6	579	1.1	590	1.1	638	1.2	594	1.1	581	1.1	544	1.0
product-N10000-s10001	756	1.4	563	1.1	**534**	**1.0**	567	1.1	540	1.0	620	1.2	735	1.4	554	1.0	543	1.0	626	1.2	579	1.1	553	1.0	542	1.0
product-N10000-s10002	1010	1.8	**555**	**1.0**	556	1.0	593	1.1	610	1.1	583	1.1	749	1.3	561	1.0	562	1.0	588	1.1	614	1.1	562	1.0	610	1.1
product-N20000-s20000	1790	1.6	1300	1.1	**1140**	**1.0**	1280	1.1	1300	1.1	1370	1.2	1830	1.6	1170	1.0	1330	1.2	1370	1.2	1240	1.1	1150	1.0	1300	1.1
product-N20000-s20001	1830	1.6	1260	1.1	1140	1.0	1360	1.2	1170	1.1	1160	1.0	1610	1.5	1120	1.0	1220	1.1	1230	1.1	1160	1.0	**1110**	**1.0**	1180	1.1
product-N20000-s20002	1460	1.3	1230	1.1	1140	1.0	1230	1.1	1260	1.1	1260	1.1	1690	1.5	1190	1.1	1300	1.2	1250	1.1	1240	1.1	**1110**	**1.0**	1250	1.1

Table 10: Detailed results for the Benders by batch algorithm, with a batch size of 5%, cut aggregation, and stabilization (basic or solution memory) compared to without stabilization

	BbB 5% CutAggr		BbB 5% CutAggr $\alpha = 0.1$		BbB 5% CutAggr $\alpha = 0.5$		BbB 5% CutAggr $\alpha = 0.9$		BbB 5% CutAggr $\alpha = 0.1$ $\beta = 0.1$		BbB 5% CutAggr $\alpha = 0.1$ $\beta = 0.5$		BbB 5% CutAggr $\alpha = 0.1$ $\beta = 0.9$		BbB 5% CutAggr $\alpha = 0.5$ $\beta = 0.1$		BbB 5% CutAggr $\alpha = 0.5$ $\beta = 0.5$		BbB 5% CutAggr $\alpha = 0.5$ $\beta = 0.9$		BbB 5% CutAggr $\alpha = 0.9$ $\beta = 0.1$		BbB 5% CutAggr $\alpha = 0.9$ $\beta = 0.5$		BbB 5% CutAggr $\alpha = 0.9$ $\beta = 0.9$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	0.89	1.2	0.79	1.1	0.80	1.1	0.81	1.1	0.96	1.3	0.84	1.1	2	2.8	0.91	1.2	**0.74**	**1.0**	0.84	1.1	0.75	1.0	0.86	1.2	0.85	1.1
LandS-N1000-s1001	**0.75**	**1.0**	0.87	1.2	0.93	1.2	0.97	1.3	2	2.4	0.96	1.3	2	2.6	0.84	1.1	0.90	1.2	0.88	1.2	0.89	1.2	0.84	1.1	2	2.3
LandS-N1000-s1002	0.99	1.4	0.99	1.4	0.71	1.0	0.75	1.1	0.99	1.4	0.89	1.3	1	1.7	0.78	1.1	0.72	1.0	0.89	1.3	0.78	1.1	**0.70**	**1.0**	0.92	1.3
LandS-N5000-s5000	5	1.2	8	2.1	**4**	**1.0**	4	1.1	8	2.3	9	2.4	10	2.8	5	1.2	4	1.1	9	2.4	4	1.1	4	1.2	8	2.2
LandS-N5000-s5001	5	1.4	4	1.2	5	1.2	4	1.2	5	1.4	5	1.4	7	1.9	4	1.0	4	1.1	5	1.3	4	1.2	**4**	**1.0**	5	1.4
LandS-N5000-s5002	4	1.0	8	2.2	5	1.3	**4**	**1.0**	9	2.3	9	2.3	10	2.9	4	1.2	5	1.3	9	2.3	4	1.1	4	1.2	9	2.4
LandS-N10000-s10000	8	1.1	16	2.1	**7**	**1.0**	9	1.1	10	1.3	10	1.4	14	1.8	9	1.2	8	1.1	11	1.4	8	1.1	9	1.2	10	1.3
LandS-N10000-s10001	8	1.0	17	2.1	10	1.2	**8**	**1.0**	18	2.3	18	2.3	15	1.8	9	1.1	10	1.3	18	2.3	9	1.1	8	1.1	18	2.2
LandS-N10000-s10002	8	1.1	9	1.2	**7**	**1.0**	8	1.0	18	2.4	18	2.5	22	2.9	8	1.1	8	1.1	18	2.5	8	1.1	8	1.1	18	2.4
LandS-N20000-s20000	**17**	**1.0**	36	2.1	18	1.1	17	1.1	38	2.3	38	2.3	44	2.7	20	1.2	19	1.2	38	2.3	19	1.2	20	1.2	38	2.3
LandS-N20000-s20001	18	1.1	36	2.2	18	1.1	**17**	**1.0**	36	2.2	20	1.2	42	2.5	19	1.1	17	1.0	20	1.2	18	1.1	19	1.1	35	2.1
LandS-N20000-s20002	18	1.1	35	2.2	18	1.1	**16**	**1.0**	37	2.3	38	2.4	48	3.0	20	1.2	20	1.2	38	2.4	19	1.2	20	1.2	37	2.3
gbd-N1000-s1000	**0.53**	**1.0**	2	3.2	1	1.9	0.61	1.1	2	3.5	2	3.9	2	4.6	0.81	1.5	0.87	1.6	2	3.8	0.82	1.6	0.81	1.5	2	3.6
gbd-N1000-s1001	0.67	1.2	2	3.1	0.89	1.5	**0.58**	**1.0**	2	3.1	2	4.0	2	4.2	0.89	1.5	0.90	1.6	2	3.9	0.75	1.3	0.80	1.4	2	3.2
gbd-N1000-s1002	0.61	1.0	2	2.7	0.61	1.0	**0.61**	**1.0**	2	2.8	2	3.0	2	3.7	0.93	1.5	0.92	1.5	2	3.0	0.65	1.1	0.89	1.5	2	2.8
gbd-N5000-s5000	3	1.0	9	2.6	4	1.1	**3**	**1.0**	9	2.8	10	3.0	12	3.8	4	1.1	4	1.3	10	2.9	4	1.1	3	1.1	9	2.8
gbd-N5000-s5001	3	1.1	9	3.1	3	1.0	3	1.0	9	3.2	13	4.5	7	2.5	5	1.5	3	1.1	13	4.6	**3**	**1.0**	5	1.5	9	3.2
gbd-N5000-s5002	3	1.1	10	3.2	4	1.2	**3**	**1.0**	8	2.5	11	3.2	12	3.7	4	1.3	4	1.3	10	3.2	3	1.0	4	1.2	8	2.6
gbd-N10000-s10000	7	1.0	17	2.5	7	1.1	7	1.0	18	2.7	18	2.7	25	3.6	8	1.2	9	1.3	18	2.7	**7**	**1.0**	9	1.3	18	2.7
gbd-N10000-s10001	**6**	**1.0**	17	2.9	9	1.5	7	1.2	17	2.9	19	3.2	24	4.1	7	1.3	9	1.5	19	3.2	7	1.2	7	1.3	17	3.0
gbd-N10000-s10002	**6**	**1.0**	18	3.0	7	1.2	6	1.1	18	2.9	19	3.2	24	4.0	7	1.2	8	1.3	20	3.3	7	1.2	7	1.2	18	3.0
gbd-N20000-s20000	12	1.1	35	3.0	13	1.1	**12**	**1.0**	19	1.6	37	3.2	49	4.1	16	1.4	14	1.2	37	3.1	14	1.2	16	1.4	19	1.6
gbd-N20000-s20001	15	1.2	36	2.7	14	1.1	**13**	**1.0**	37	2.8	46	3.5	49	3.8	17	1.3	20	1.5	45	3.5	15	1.1	17	1.3	37	2.8
gbd-N20000-s20002	**14**	**1.0**	37	2.7	16	1.2	14	1.0	37	2.7	38	2.8	50	3.7	17	1.2	19	1.4	39	2.8	15	1.1	17	1.2	37	2.7
ssn-N1000-s1000	63	4.3	**14**	**1.0**	17	1.2	38	2.6	15	1.1	16	1.1	22	1.5	18	1.3	17	1.2	16	1.1	32	2.2	19	1.3	15	1.1
ssn-N1000-s1001	63	4.2	**15**	**1.0**	19	1.3	41	2.8	16	1.0	16	1.1	22	1.5	18	1.2	16	1.1	17	1.1	35	2.3	19	1.2	15	1.0
ssn-N1000-s1002	59	3.9	15	1.0	19	1.3	43	2.9	17	1.1	17	1.1	22	1.5	18	1.2	17	1.2	17	1.1	31	2.1	18	1.2	**15**	**1.0**
ssn-N5000-s5000	337	4.1	**81**	**1.0**	111	1.4	197	2.4	89	1.1	94	1.2	127	1.6	111	1.4	96	1.2	99	1.2	172	2.1	112	1.4	93	1.1
ssn-N5000-s5001	322	3.8	**85**	**1.0**	106	1.3	205	2.4	85	1.0	100	1.2	126	1.5	108	1.3	100	1.2	101	1.2	182	2.1	112	1.3	85	1.0
ssn-N5000-s5002	308	3.8	**81**	**1.0**	110	1.4	209	2.6	95	1.2	99	1.2	141	1.8	113	1.4	100	1.2	99	1.2	172	2.1	116	1.4	90	1.1
ssn-N10000-s10000	672	3.6	196	1.1	233	1.3	471	2.5	**185**	**1.0**	204	1.1	277	1.5	232	1.3	213	1.2	212	1.1	389	2.1	222	1.2	194	1.1
ssn-N10000-s10001	760	4.2	**180**	**1.0**	236	1.3	512	2.8	209	1.2	231	1.3	301	1.7	235	1.3	217	1.2	211	1.2	439	2.4	244	1.4	187	1.0
ssn-N10000-s10002	690	3.7	185	1.0	231	1.3	469	2.5	186	1.0	193	1.0	289	1.6	222	1.2	195	1.1	218	1.2	406	2.2	228	1.2	**184**	**1.0**
ssn-N20000-s20000	1651	4.1	**405**	**1.0**	524	1.3	1027	2.5	432	1.1	491	1.2	672	1.7	531	1.3	524	1.3	492	1.2	866	2.1	529	1.3	432	1.1
ssn-N20000-s20001	1651	3.8	**436**	**1.0**	549	1.3	1043	2.4	474	1.1	485	1.1	728	1.7	561	1.3	516	1.2	475	1.1	893	2.0	551	1.3	446	1.0
ssn-N20000-s20002	1543	3.8	**406**	**1.0**	550	1.4	1052	2.6	434	1.1	506	1.2	650	1.6	554	1.4	499	1.2	489	1.2	914	2.3	558	1.4	450	1.1
storm-N1000-s1000	6	1.1	7	1.3	**6**	**1.0**	6	1.1	10	1.8	12	2.0	11	1.9	6	1.0	7	1.2	12	2.0	6	1.1	6	1.1	10	1.8
storm-N1000-s1001	6	1.2	8	1.4	**5**	**1.0**	6	1.1	8	1.4	8	1.4	10	1.8	6	1.1	6	1.2	8	1.4	6	1.0	6	1.1	8	1.4
storm-N1000-s1002	6	1.1	11	1.8	**6**	**1.0**	6	1.0	7	1.2	11	1.9	10	1.8	6	1.1	6	1.1	11	1.9	6	1.0	6	1.1	7	1.2
storm-N5000-s5000	33	1.1	54	1.9	**29**	**1.0**	32	1.1	40	1.4	40	1.4	56	1.9	30	1.1	33	1.1	41	1.4	30	1.0	30	1.1	40	1.4
storm-N5000-s5001	33	1.1	54	1.9	**29**	**1.0**	29	1.0	37	1.3	39	1.3	52	1.8	31	1.1	34	1.2	39	1.3	30	1.1	31	1.1	37	1.3
storm-N5000-s5002	37	1.3	38	1.3	30	1.0	30	1.0	56	1.9	41	1.4	47	1.6	30	1.0	37	1.3	41	1.4	**29**	**1.0**	30	1.0	56	1.9
storm-N10000-s10000	73	1.3	79	1.4	62	1.1	**58**	**1.0**	78	1.4	127	2.2	114	2.0	65	1.1	65	1.1	127	2.2	60	1.0	64	1.1	79	1.4
storm-N10000-s10001	76	1.3	75	1.2	60	1.0	**60**	**1.0**	79	1.3	130	2.2	163	2.7	63	1.0	64	1.1	127	2.1	**60**	**1.0**	63	1.0	79	1.3
storm-N10000-s10002	73	1.3	75	1.3	61	1.1	**56**	**1.0**	116	2.1	82	1.5	118	2.1	67	1.2	70	1.2	82	1.5	59	1.0	66	1.2	117	2.1
storm-N20000-s20000	167	1.4	147	1.2	132	1.1	**120**	**1.0**	148	1.2	173	1.4	256	2.1	138	1.1	140	1.2	172	1.4	139	1.2	138	1.1	147	1.2
storm-N20000-s20001	161	1.3	157	1.3	**121**	**1.0**	122	1.0	180	1.5	186	1.5	243	2.0	127	1.0	143	1.2	185	1.5	125	1.0	127	1.0	181	1.5
storm-N20000-s20002	160	1.3	169	1.4	130	1.1	**120**	**1.0**	153	1.3	170	1.4	240	2.0	141	1.2	148	1.2	171	1.4	127	1.1	141	1.2	153	1.3
20term-N1000-s1000	36	2.5	**15**	**1.0**	21	1.4	33	2.2	15	1.0	21	1.4	30	2.1	22	1.5	20	1.4	18	1.3	27	1.8	20	1.3	16	1.1
20term-N1000-s1001	37	2.6	**14**	**1.0**	19	1.3	32	2.2	16	1.1	18	1.2	31	2.2	18	1.3	19	1.3	17	1.2	29	2.0	21	1.5	15	1.1
20term-N1000-s1002	37	3.0	15	1.2	21	1.7	33	2.7	15	1.2	18	1.4	32	2.6	21	1.7	19	1.5	18	1.4	25	2.0	20	1.6	**12**	**1.0**
20term-N5000-s5000	199	2.8	81	1.1	106	1.5	156	2.2	**71**	**1.0**	89	1.3	157	2.2	94	1.3	91	1.3	94	1.3	134	1.9	104	1.5	87	1.2
20term-N5000-s5001	197	3.0	78	1.2	106	1.6	158	2.4	69	1.0	87	1.3	137	2.1	103	1.6	97	1.5	98	1.5	146	2.2	101	1.5	**66**	**1.0**
20term-N5000-s5002	182	2.5	83	1.1	101	1.4	161	2.2	88	1.2	92	1.3	144	2.0	102	1.4	83	1.1	92	1.3	132	1.8	99	1.4	**73**	**1.0**
20term-N10000-s10000	411	3.0	**136**	**1.0**	206	1.5	352	2.6	145	1.1	192	1.4	285	2.1	208	1.5	201	1.5	158	1.2	302	2.2	217	1.6	140	1.0
20term-N10000-s10001	409	2.5	166	1.0	218	1.4	381	2.4	**161**	**1.0**	163	1.0	342	2.1	213	1.3	204	1.3	210	1.3	305	1.9	197	1.2	173	1.1
20term-N10000-s10002	388	2.6	159	1.1	231	1.6	309	2.1	**147**	**1.0**	189	1.3	273	1.9	214	1.5	206	1.4	155	1.1	313	2.1	214	1.5	152	1.0
20term-N20000-s20000	860	3.3	**257**	**1.0**	501	1.9	802	3.1	306	1.2	339	1.3	635	2.5	488	1.9	444	1.7	449	1.7	685	2.7	490	1.9	309	1.2
20term-N20000-s20001	985	3.5	311	1.1	476	1.7	795	2.8	**284**	**1.0**	427	1.5	610	2.1	512	1.8	441	1.6	350	1.2	684	2.4	436	1.5	328	1.2
20term-N20000-s20002	897	3.2	**281**	**1.0**	466	1.7	743	2.6	315	1.1	389	1.4	710	2.5	458	1.6	439	1.6	390	1.4	697	2.5	465	1.7	298	1.1
Fleet20_3-N1000-s1000	42	1.9	**22**	**1.0**	25	1.1	32	1.5	24	1.1	27	1.3	36	1.7	24	1.1	26	1.2	24	1.1	30	1.4	24	1.1	24	1.1
Fleet20_3-N1000-s1001	40	1.9	**22**	**1.0**	25	1.2	32	1.5	22	1.0	25	1.2	33	1.5	24	1.1	25	1.2	26	1.2	30	1.4	23	1.1	22	1.0
Fleet20_3-N1000-s1002	43	1.9	**23**	**1.0**	25	1.1	33	1.4	25	1.1	28	1.2	36	1.5	26	1.1	26	1.1	29	1.2	32	1.3	25	1.1	24	1.0
Fleet20_3-N5000-s5000	218	2.0	114	1.0	126	1.1	165	1.5	**110**	**1.0**	130	1.2	179	1.6	126	1.1	127	1.2	128	1.2	152	1.4	129	1.2	122	1.1
Fleet20_3-N5000-s5001	209	1.7	**121**	**1.0**	125	1.0	160	1.3	123	1.0	143	1.2	188	1.6	122	1.0	126	1.0	127	1.0	151	1.2	122	1.0	125	1.0
Fleet20_3-N5000-s5002	205	1.7	125	1.0	126	1.0	163	1.3	**122**	**1.0**	126	1.0	172	1.4	126	1.0	124	1.0	140	1.1	152	1.2	**122**	**1.0**	126	1.0
Fleet20_3-N10000-s10000	426	1.9	232	1.0	264	1.2	343	1.5	**226**	**1.0**	253	1.1	367	1.6	254	1.1	273	1.2	261	1.2	315	1.4	255	1.1	248	1.1
Fleet20_3-N10000-s10001	467	2.0	249	1.1	250	1.1	345	1.5	264	1.1	279	1.2	376	1.6	256	1.1	259	1.1	269	1.2	310	1.3	258	1.1	**230**	**1.0**
Fleet20_3-N10000-s10002	426	1.9	228	1.0	258	1.1	324	1.4	**226**	**1.0**	290	1.3	394	1.7	254	1.1	259	1.1	294	1.3	307	1.4	252	1.1	265	1.2
Fleet20_3-N20000-s20000	886	1.7	556	1.1	**520**	**1.0**	695	1.3	586	1.1	612	1.2	885	1.7	543	1.0	525	1.0	639	1.2	635	1.2	524	1.0	577	1.1
Fleet20_3-N20000-s20001	856	1.8	488	1.0	523	1.1	671	1.4	509	1.1	544	1.1	783	1.6	530	1.1	541	1.1	539	1.1	637	1.3	518	1.1	**476**	**1.0**
Fleet20_3-N20000-s20002	885	1.7	545	1.0	546	1.0	681	1.3	554	1.0	662	1.3	819	1.6	**528**	**1.0**	562	1.1	670	1.3	671	1.3	551	1.0	576	1.1
product-N1000-s1000	140	1.9	80	1.1	83	1.1	115	1.5	**75**	**1.0**	79	1.1	122	1.6	85	1.1	86	1.1	79	1.1	108	1.4	84	1.1	75	1.0
product-N1000-s1001	149	2.0	**76**	**1.0**	92	1.2	119	1.6	79	1.0	92	1.2	168	2.2	92	1.2	90	1.2	92	1.2	114	1.5	92	1.2	79	1.0
product-N1000-s1002	135	2.0	**68**	**1.0**	82	1.2	127	1.9	108	1.6	84	1.2	146	2.1	93	1.4	80	1.2	85	1.2	106	1.6	93	1.4	109	1.6
product-N5000-s5000	734	1.9	**380**	**1.0**	454	1.2	607	1.6	382	1.0	458	1.2	835	2.2	432	1.1	459	1.2	448	1.2	554	1.5	439	1.2	382	1.0
product-N5000-s5001	787	2.0	**387**	**1.0**	438	1.1	580	1.5	421	1.1	468	1.2	880	2.3	449	1.2	448	1.2	488	1.3	574	1.5	449	1.2	392	1.0
product-N5000-s5002	797	2.1	**383**	**1.0**	474	1.2	627	1.6	411	1.1	454	1.2	737	1.9	463	1.2	483	1.3	437	1.1	585	1.5	478	1.2	405	1.1
product-N10000-s10000	1620	2.3	**716**	**1.0**	955	1.3	1280	1.8	740	1.0	926	1.3	1510	2.1	866	1.2	875	1.2	910	1.3	1160	1.6	863	1.2	740	1.0
product-N10000-s10001	1400	2.0	**696**	**1.0**	921	1.3	1160	1.7	771	1.1	889	1.3	1620	2.3	913	1.3	901	1.3	897	1.3	1130	1.6	894	1.3	772	1.1
product-N10000-s10002	1550	2.2	**697**	**1.0**	880	1.3	1180	1.7	743	1.1	948	1.4	1470	2.1	862	1.2	892	1.3	909	1.3	1160	1.7	868	1.2	802	1.2
product-N20000-s20000	3330	1.9	**1750**	**1.0**	2070	1.2	2670	1.5	1820	1.0	2120	1.2	3610	2.1	2120	1.2	2200	1.3	2130	1.2	2600	1.5	2150	1.2	1790	1.0
product-N20000-s20001	3230	2.0	**1620**	**1.0**	1880	1.2	2550	1.6	1640	1.0	1930	1.2	3040	1.9	1860	1.1	1840	1.1	1920	1.2	2430	1.5	1870	1.2	1740	1.1
product-N20000-s20002	3540	2.2	1640	1.0	1930	1.2	2730	1.7	**1620**	**1.0**	1990	1.2	3480	2.1	1960	1.2	2060	1.3	2110	1.3	2410	1.5	2060	1.3	**1620**	**1.0**

Table 11: Final results, the best stabilized Benders by batch algorithm compared to all stabilized benchmark methods.

	CPLEX Barrier		Level Bundle		In-out monocut		In-out multicut		In-out 1% CutAggr		In-out 5% CutAggr		BbB 1% CutAggr $\alpha = 0.5$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	**0.07**	**1.0**	1	17.3	1	15.6	2	29.4	0.71	10.1	1	14.4	1.00	14.2
LandS-N1000-s1001	**0.08**	**1.0**	1	17.0	0.59	7.4	1	15.0	0.74	9.3	1	12.7	1	12.8
LandS-N1000-s1002	**0.07**	**1.0**	1	17.8	0.99	14.1	1	15.6	0.69	9.9	0.91	13.0	0.88	12.5
LandS-N5000-s5000	**1**	**1.0**	8	5.7	8	6.3	10	7.6	5	3.5	5	3.9	4	3.3
LandS-N5000-s5001	**0.41**	**1.0**	7	17.2	8	19.4	6	15.5	5	11.2	6	13.5	5	13.2
LandS-N5000-s5002	**1**	**1.0**	6	4.2	8	5.8	12	8.4	4	3.2	6	4.3	5	3.5
LandS-N10000-s10000	**0.96**	**1.0**	14	14.5	24	24.8	11	11.6	9	9.4	12	12.4	9	9.4
LandS-N10000-s10001	**1**	**1.0**	13	12.1	24	22.1	13	11.9	10	9.4	10	9.3	9	8.7
LandS-N10000-s10002	**0.97**	**1.0**	15	15.5	23	23.8	23	23.4	10	10.3	11	11.7	9	9.0
LandS-N20000-s20000	**7**	**1.0**	28	4.1	71	10.4	42	6.1	22	3.2	26	3.8	21	3.1
LandS-N20000-s20001	**2**	**1.0**	26	12.4	67	32.4	40	19.0	22	10.5	21	9.9	21	10.3
LandS-N20000-s20002	**7**	**1.0**	29	4.0	48	6.7	43	6.0	22	3.1	21	2.9	20	2.7
gbd-N1000-s1000	**0.03**	**1.0**	2	58.1	1	42.2	3	88.7	0.97	32.2	2	57.0	0.81	26.9
gbd-N1000-s1001	**0.03**	**1.0**	2	78.4	1	42.0	2	53.0	1	46.8	2	50.9	1	33.6
gbd-N1000-s1002	**0.05**	**1.0**	2	46.9	1	25.4	2	34.6	1	21.8	1	26.5	0.82	16.4
gbd-N5000-s5000	**0.15**	**1.0**	8	55.7	7	48.5	13	89.3	7	48.3	9	58.5	4	24.4
gbd-N5000-s5001	**0.18**	**1.0**	11	61.4	11	63.7	9	50.5	7	37.2	7	41.3	4	20.1
gbd-N5000-s5002	**0.17**	**1.0**	11	63.1	12	70.5	9	52.0	7	39.8	7	41.5	5	29.8
gbd-N10000-s10000	**0.32**	**1.0**	23	70.9	19	57.9	30	93.1	17	54.5	18	54.8	7	23.0
gbd-N10000-s10001	**0.35**	**1.0**	26	74.3	32	91.1	18	50.5	14	39.2	17	47.6	7	21.0
gbd-N10000-s10002	**0.37**	**1.0**	23	63.4	20	53.5	15	41.5	16	43.4	18	48.6	8	22.4
gbd-N20000-s20000	**1**	**1.0**	45	40.1	107	94.6	56	49.7	30	26.5	34	30.1	19	16.5
gbd-N20000-s20001	**0.86**	**1.0**	47	54.1	72	83.4	55	64.5	30	34.7	31	35.9	17	19.4
gbd-N20000-s20002	**0.75**	**1.0**	39	52.3	69	91.4	51	67.6	31	41.8	38	51.3	15	19.6
ssn-N1000-s1000	32	7.9	97	24.0	**4**	**1.0**	187	46.4	9	2.3	19	4.8	8	1.9
ssn-N1000-s1001	32	5.2	85	13.6	**6**	**1.0**	117	18.7	10	1.5	19	3.1	8	1.3
ssn-N1000-s1002	31	4.9	87	13.8	**6**	**1.0**	106	16.9	10	1.6	19	3.0	8	1.3
ssn-N5000-s5000	293	8.3	621	17.6	**35**	**1.0**	936	26.5	67	1.9	139	3.9	47	1.3
ssn-N5000-s5001	327	9.4	719	20.6	**35**	**1.0**	597	17.1	69	2.0	128	3.7	46	1.3
ssn-N5000-s5002	311	14.1	631	28.5	**22**	**1.0**	852	38.5	74	3.4	133	6.0	49	2.2
ssn-N10000-s10000	1271	15.1	1440	17.1	86	1.0	1937	23.0	167	2.0	319	3.8	**84**	**1.0**
ssn-N10000-s10001	1332	25.0	1613	30.2	**53**	**1.0**	1261	23.6	185	3.5	318	6.0	98	1.8
ssn-N10000-s10002	1064	20.8	1451	28.3	**51**	**1.0**	1195	23.3	161	3.1	298	5.8	90	1.8
ssn-N20000-s20000	2592	14.3	3232	17.9	245	1.4	3791	21.0	441	2.4	729	4.0	**181**	**1.0**
ssn-N20000-s20001	2070	10.9	2986	15.7	237	1.2	2460	12.9	365	1.9	743	3.9	**190**	**1.0**
ssn-N20000-s20002	3195	15.9	3108	15.4	246	1.2	2332	11.6	395	2.0	735	3.6	**201**	**1.0**
storm-N1000-s1000	41	5.4	14	1.9	10	1.3	11	1.4	8	1.0	10	1.3	**8**	**1.0**
storm-N1000-s1001	41	6.0	16	2.2	**7**	**1.0**	21	3.0	7	1.0	10	1.4	7	1.1
storm-N1000-s1002	41	6.2	15	2.3	11	1.7	12	1.8	7	1.1	9	1.4	**7**	**1.0**
storm-N5000-s5000	348	10.7	74	2.3	41	1.3	63	1.9	52	1.6	53	1.6	**32**	**1.0**
storm-N5000-s5001	294	8.4	78	2.2	38	1.1	61	1.7	51	1.5	53	1.5	**35**	**1.0**
storm-N5000-s5002	305	10.1	76	2.5	43	1.4	63	2.1	45	1.5	51	1.7	**30**	**1.0**
storm-N10000-s10000	808	12.7	140	2.2	108	1.7	212	3.3	94	1.5	100	1.6	**64**	**1.0**
storm-N10000-s10001	732	11.5	149	2.3	105	1.6	201	3.2	104	1.6	117	1.8	**64**	**1.0**
storm-N10000-s10002	751	11.3	147	2.2	161	2.4	189	2.8	99	1.5	114	1.7	**66**	**1.0**
storm-N20000-s20000	2510	18.1	316	2.3	515	3.7	259	1.9	218	1.6	237	1.7	**139**	**1.0**
storm-N20000-s20001	2362	17.2	266	1.9	633	4.6	251	1.8	202	1.5	230	1.7	**137**	**1.0**
storm-N20000-s20002	2297	17.0	283	2.1	570	4.2	246	1.8	214	1.6	228	1.7	**135**	**1.0**
20term-N1000-s1000	14	1.2	197	17.3	27	2.4	128	11.3	24	2.1	41	3.6	**11**	**1.0**
20term-N1000-s1001	14	1.4	214	22.1	43	4.5	74	7.6	26	2.7	46	4.8	**10**	**1.0**
20term-N1000-s1002	14	1.3	241	23.2	38	3.7	139	13.4	31	3.0	45	4.4	**10**	**1.0**
20term-N5000-s5000	83	1.6	994	19.1	581	11.2	661	12.7	188	3.6	271	5.2	**52**	**1.0**
20term-N5000-s5001	80	1.8	1059	24.4	423	9.7	650	14.9	206	4.7	277	6.4	**43**	**1.0**
20term-N5000-s5002	84	1.6	1078	20.1	443	8.3	732	13.7	198	3.7	257	4.8	**54**	**1.0**
20term-N10000-s10000	205	2.0	2305	22.8	2491	24.7	863	8.5	465	4.6	649	6.4	**101**	**1.0**
20term-N10000-s10001	199	2.0	2647	26.3	3382	33.6	1389	13.8	491	4.9	560	5.6	**101**	**1.0**
20term-N10000-s10002	194	1.9	2400	24.1	2543	25.5	1317	13.2	467	4.7	569	5.7	**100**	**1.0**
20term-N20000-s20000	457	2.4	4562	23.9	13423	70.4	1834	9.6	1007	5.3	1412	7.4	**191**	**1.0**
20term-N20000-s20001	457	2.2	4378	20.9	10267	49.0	1680	8.0	980	4.7	1407	6.7	**210**	**1.0**
20term-N20000-s20002	451	2.4	5588	29.3	9286	48.7	1748	9.2	1043	5.5	1295	6.8	**191**	**1.0**
Fleet20_3-N1000-s1000	24	1.5	104	6.2	61	3.7	71	4.3	27	1.6	42	2.5	**17**	**1.0**
Fleet20_3-N1000-s1001	23	1.3	103	6.0	34	2.0	103	6.0	26	1.5	39	2.3	**17**	**1.0**
Fleet20_3-N1000-s1002	22	1.2	114	6.3	55	3.1	106	5.9	25	1.4	43	2.4	**18**	**1.0**
Fleet20_3-N5000-s5000	266	3.6	485	6.5	933	12.5	552	7.4	181	2.4	239	3.2	**75**	**1.0**
Fleet20_3-N5000-s5001	273	3.6	509	6.6	541	7.1	331	4.3	172	2.2	264	3.4	**77**	**1.0**
Fleet20_3-N5000-s5002	267	3.6	506	6.8	682	9.2	535	7.2	198	2.7	248	3.4	**74**	**1.0**
Fleet20_3-N10000-s10000	784	5.3	988	6.7	3540	24.1	1150	7.8	435	3.0	598	4.1	**147**	**1.0**
Fleet20_3-N10000-s10001	816	5.5	1040	7.0	4750	32.1	1230	8.3	422	2.9	550	3.7	**148**	**1.0**
Fleet20_3-N10000-s10002	826	5.7	984	6.8	2950	20.5	708	4.9	448	3.1	623	4.3	**144**	**1.0**
Fleet20_3-N20000-s20000	2488	8.2	2630	8.7	14900	49.3	2470	8.2	1070	3.5	1270	4.2	**302**	**1.0**
Fleet20_3-N20000-s20001	2469	8.0	2910	9.4	14100	45.5	1490	4.8	945	3.0	1240	4.0	**310**	**1.0**
Fleet20_3-N20000-s20002	2381	7.5	2650	8.4	22000	69.4	1380	4.4	1040	3.3	1430	4.5	**317**	**1.0**
product-N1000-s1000	185	2.5	479	6.4	**75**	**1.0**	480	6.4	108	1.4	180	2.4	76	1.0
product-N1000-s1001	186	2.4	718	9.2	83	1.1	539	6.9	124	1.6	179	2.3	**78**	**1.0**
product-N1000-s1002	165	2.2	677	9.0	84	1.1	519	6.9	108	1.4	189	2.5	**75**	**1.0**
product-N5000-s5000	1374	4.5	3290	10.8	1070	3.5	2840	9.3	820	2.7	1460	4.8	**305**	**1.0**
product-N5000-s5001	3073	9.2	3150	9.4	1100	3.3	2550	7.6	724	2.2	1330	4.0	**335**	**1.0**
product-N5000-s5002	1916	6.5	3160	10.7	1210	4.1	2680	9.1	817	2.8	1350	4.6	**295**	**1.0**
product-N10000-s10000	4991	8.8	6910	12.2	4940	8.7	5750	10.2	2030	3.6	3130	5.5	**565**	**1.0**
product-N10000-s10001	3850	7.2	6670	12.5	6860	12.8	5920	11.1	2000	3.7	2810	5.3	**534**	**1.0**
product-N10000-s10002	4351	7.8	7940	14.3	4270	7.7	5520	9.9	1880	3.4	3460	6.2	**556**	**1.0**
product-N20000-s20000	14757	12.9	13200	11.6	$+\infty$	>43.5	12700	11.1	4700	4.1	8300	7.3	**1140**	**1.0**
product-N20000-s20001	14346	12.6	13900	12.2	$+\infty$	>46.7	11700	10.3	4690	4.1	7580	6.6	**1140**	**1.0**
product-N20000-s20002	17287	15.2	15800	13.9	35600	31.2	12600	11.1	5270	4.6	8070	7.1	**1140**	**1.0**