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

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

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]

July 13, 2021

[1] Université de Bordeaux, UMR CNRS 5251, Inria Bordeaux Sud-Ouest, Talence,France
[2] RTE, Paris La Défense, France

Abstract

This paper introduces a new exact algorithm to solve two-stage stochastic linear programs. Based on the multicut Benders reformulation of such problems, with one subproblem for each scenario, this method relies on a partition of the subproblems into batches. By detecting as soon as possible the non-optimality of a first-stage candidate, it solves only a few subproblems at most iterations. We also propose two primal stabilization schemes for the algorithm. We report an extensive computational study on large-scale instances of stochastic optimization literature that shows the efficiency of the proposed algorithm compared to five classical alternative algorithms and significant computational time savings brought by the primal stabilization schemes.

Keywords— Benders Decomposition; Large-scale linear programming; Cut aggregation

1 Introduction

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

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

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

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

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

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

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

makes the use of decomposition methods attractive. A state-of-the art method to solve this problem is the Benders decomposition algorithm (Rahmaniani et al., 2017). 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 $\phi(., s)$ as:

$$dom\Big(\phi(., s)\Big) = \{x \in \mathbb{R}^{n_1} | r_s^\top (d_s - T_s x) \leqslant 0,\ \forall r_s \in \text{Rays}(\Pi_s)\}$$

Then we can replace in formulation (2) the polyhedral application $x \mapsto \phi(x, s)$ by its outer linearization on its domain. Using an epigraph variable θ_s for every $s \in S$, we obtain the multicut Benders reformulation (Birge and Louveaux, 1988) of problem (1):

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

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

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

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

Algorithm 1: Classical version of the multicut Benders decomposition algorithm

Parameters: $\epsilon > 0$ 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 $\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S}$
  $LB^{(k)} \leftarrow c^\top \check{x}^{(k)} + \sum_{s\in S} p_s \check{\theta}^{(k)}$
  for $s \in S$ **do**
    Solve $(SP(\check{x}^{(k)}, s))$ and retrieve π_s an extreme point or an extreme ray of Π_s
    if $(SP(\check{x}^{(k)}, s))$ *is feasible* **then**
      Add $\theta_s \geqslant \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
    else
      Add $0 \geqslant \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
  if $(SP(\check{x}^{(k)}, s))$ *is feasible* $\forall s \in S$ **then**
    $UB^{(k)} \leftarrow min\Big(UB^{(k-1)}, c^\top \check{x}^{(k)} + \sum_{s\in S} p_s \pi_s^\top (d_s - T_s \check{x}^{(k)})\Big)$
  $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$
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 a non-optimal solution without necessarily having to solve all the subproblems. As a result, only few subproblems are generally solved at a first-stage candidate solution. To prevent introducing too many cuts in the restricted master program, the algorithm can use cut aggregation, thus generating a single cut from all subproblems that belong to an identical batch. If the number of batches is equal to one, the Benders by batch algorithm is equivalent to the classical version of the Benders decomposition algorithm (multicut or monocut, depending on the use of cut aggregation). Unlike several existing methods based on similar ideas (Wets, 1983; Dantzig and Infanger, 1991; Higle and Sen, 1991; Oliveira et al., 2011), our algorithm requires no restrictive assumptions on the uncertainty (e.g., $\Pi_s = \Pi$ for every $s \in S$), is exact and has finite convergence. It is relatively easy to implement and only requires solving linear programs. Our method can benefit from advanced cut aggregation schemes and is compatible with classical dual stabilization techniques (Magnanti and Wong, 1981; Papadakos, 2008; Sherali and Lunday, 2013). We propose two primal stabilization schemes for the algorithm based on in-out separation strategies (Ben-Ameur and Neto, 2007) and prove the finite convergence and exact behavior of the stabilized algorithm.

The contributions of the paper can be summarized as follow :

- We propose a new exact algorithm to solve the Benders reformulation of two-stage linear stochastic programs based on a sequential stopping criterion relying on a partition of subproblems. The use of this stopping criterion allows the algorithm to solve only a few subproblems at most iterations by detecting the non-optimality of a first-stage candidate solution early in the subproblems resolution process.
- We also develop two primal stabilization schemes for our algorithm and provide a proof of finite convergence and exact behavior of the stabilized algorithm.
- We perform an extensive numerical study showing the effectiveness of the developed algorithm on some

classical stochastic instances of the literature compared to classical implementation of the monocut and multicut Benders decomposition algorithm, with and without in-out stabilization.

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 two primal stabilization methods: 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 works

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

A well-known downside of cutting planes methods, and therefore of the Benders decomposition algorithm, is the oscillation of the first-stage variables. Because of the relaxation of all the constraints related to the subproblems, the solutions of the relaxed master programs might be far from the optimal solution to the initial problem. This might lead to a large amount of time spent in evaluating poor quality solutions in the early iterations. To our knowledge, successful methods proposed so far are either inspired by bundle methods (Zaourar and Malick, 2014; Linderoth and Wright, 2003; Wolf et al., 2014), or by in-out separation approaches (Ben-Ameur and Neto, 2007). Those methods try to restrict the search of an optimal solution close to a given first-stage solution. This solution is called *stability center* in the case of bundle methods, or *in point* in the case of in-out stabilization. On the one hand, many authors proposed quadratic stabilization techniques, such as Ruszczyński (1986), who added a quadratic proximal term in the objective function of the relaxed master program, or Zaourar and Malick (2014) and Wolf et al. (2014), who used quadratic level stabilizations. Linderoth and Wright (2003) proposed to use the infinite norm and showed also relevant numerical results with an effective asynchronous parallelized framework. On the other hand, the in-out separation scheme performs a linear search between the in point and the solution to the relaxed master program, and it can rely on the practical effectiveness of linear programming solvers. The in-out separation approach has been applied successfully in a cutting plane algorithm to solve a survivable network design problem (Ben-Ameur and Neto, 2007), in column generation (Pessoa et al., 2013), in a branch-and-cut algorithm based on a Benders decomposition approach to solve facility location problems (Fischetti et al., 2016), and in a cutting plane algorithm applied to disjunctive optimization (Fischetti and Salvagnin, 2010).

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

In the case of stochastic programming, formulations rely either on an epigraph variable for every subproblem (see formulation (4)) or on a single epigraph variable for all the subproblems, also called *L-shaped method* (Van Slyke and Wets, 1969). The former formulation is referred to as the multicut Benders reformulation,

whereas the latter is known as the monocut Benders reformulation. The multicut Benders reformulation was introduced by Birge and Louveaux (1988). You and Grossmann (2013) showed dramatic improvement both on computing time and number of iterations due to the multicut reformulation on two supply chain planning problems. The multicut version provides a tighter approximation of the second-stage function, and converges in less iterations than the monocut one. However the master problem might suffer from the large number of cuts added through the optimization process, and thus might become time consuming to solve. The question of using either the monocut or multicut version of the algorithm is not straightforward. As far as we know, one of the major improvements proposed to improve pure multicut Benders decomposition was to use massive parallelization (Linderoth and Wright, 2003). Trukhanov et al. (2010) proposed a framework to aggregate some optimality cuts with the aim of finding a compromise between the monocut and pure multicut versions of the algorithm.

One of the major bottlenecks faced to solve two-stage stochastic programs is the large number of subproblems to solve at each iteration to compute Benders cuts. Researchers proposed some methods to avoid the resolution of all the subproblems at each iteration of the Benders decomposition algorithm. In the case of stochastic problems with fixed recourse where the second-stage objective function does not depend on the uncertainty (i.e, $g_s = g$ for every $s \in S$ in problem (3)), 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 for the objective value. Higle and Sen (1991) introduced *stochastic decomposition*. The standard version of the method only solves a few subproblems at each iteration and computes cuts with all the dual solutions obtained at previous iterations. Finally, Oliveira et al. (2011) proposed an algorithm which does not restrict to the hypothesis $g_s = g$, $\forall s \in S$. It adapts the dual solutions of a subset of subproblems to generate inexact cuts to the remaining subproblems. The methods of Oliveira et al. (2011), Dantzig and Glynn (1990) and Higle and Sen (1991) are designed for a monocut algorithm, but the method of Oliveira et al. (2011) could be adapted to a multicut algorithm.

Finally, among other techniques used to accelerate the resolution of two-stage stochastic programs, Crainic et al. (2020) proposed the so-called *Partial Benders decomposition*. Under the hypothesis $g_s = g$, $\forall s \in S$, 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, 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 competitive numerical results compared to a classical Benders decomposition implementation with level stabilization. Table 1 classifies the different methods discussed in this section.

Paper	Randomness hypothesis	Monocut or multicut	Exact method	Finite convergence	Solve all SPs	Stabilization
(Crainic et al., 2020)	$g_s = g \, \forall s \in S$	Both	Yes	Yes	Yes	No
(Song and Luedtke, 2015)	$g_s = g \, \forall s \in S$	Not applicable	Yes	Yes	Yes	Level*
(Trukhanov et al., 2010)	No	Multicut	Yes	Yes	Yes	No
(Linderoth and Wright, 2003)	No	Multicut	Yes	Yes	Yes	Level
(Wets, 1983)	$g_s = g \, \forall s \in S$	Both	Yes	Yes	No	No
(Dantzig and Infanger, 1991)	$g_s = g \, \forall s \in S$	Monocut	No	Yes	No	No
(Higle and Sen, 1991)	$g_s = g \, \forall s \in S$	Monocut	Yes	No	No	No
(Oliveira et al., 2011)	No	Monocut	Statistical	Yes	No	Bundle
This work	No	Multicut	Yes	Yes	No	In-out

Table 1: Comparison of stochastic methods to accelerate Benders decomposition.
SPs : subproblems
* : with the work of van Ackooij et al. (2017)

3 Benders by batch

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

Without loss of generality, we assume that the problem has *relatively complete recourse* (i.e., $X \subset dom\,(\phi(.,s))$ for every scenario $s \in S$), meaning that every subproblem is feasible for every $x \in X$. As a result, only optimality cuts are required in the Benders decomposition algorithm, and every $x \in X$ defines an upper bound on the optimal value of the problem. Every two-stage linear stochastic program can be reformulated in a problem satisfying this hypothesis by introducing slack variables with large enough coefficients in the objective function.

We first present some notations necessary to formally describe the algorithm. We consider an ordered set of scenarios $S = \{s_1, s_2, ..., s_{Card(S)}\}$ 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, $\eta_i = \min\{\eta, Card(S) - (i-1)\eta\}$. 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 $\check{x}^{(k)}$ the optimal first-stage solution to $(RMP)^{(k)}$ at iteration k of the algorithm, and by $\check{\theta}_s^{(k)}$ the optimal value of the epigraph variable associated with a scenario $s \in S$. A lower bound on the optimal value of problem (1) is then computed as $LB^{(k)} = c^\top \check{x}^{(k)} + \sum_{s \in S} p_s \check{\theta}_s^{(k)}$. For a first-stage solution $x \in X$, we denote by $UB(x) = c^\top x + \sum_{s \in S} p_s \phi(x, s)$ an upper bound on the optimal value of problem (1).

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

Definition 1. *ϵ_i-approximation: Let $\epsilon > 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration and σ a permutation of $[\![1, \kappa]\!]$. 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

$$\epsilon_i = \epsilon_{i-1} - \sum_{s \in S_{\sigma(i-1)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right), \tag{6}$$

which means that computing the successive remaining gaps consists in filling the gap ϵ with the differences between the true values of the subproblems and their epigraph approximations in $(RMP)^{(k)}$.

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

Proposition 1. *Let $\epsilon > 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration of the algorithm, and σ a permutation of $[\![1, \kappa]\!]$. 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. Proof of proposition 1

($\Rightarrow$) 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, previous equation gives:

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

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: $\sum_{s \in S_{\sigma(t)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \geqslant 0, \ \forall t \in \{1, \ldots, \kappa\}$. We therefore have:

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

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:

$$\sum_{s \in S_{\sigma(\kappa)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leqslant \epsilon_\kappa \tag{7}$$

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 (7), we have:

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

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

Corollary 1. *Let ϵ 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 $\check{x}^{(k)}$ is not an optimal solution to problem (1).*

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

Algorithm 2: The Benders by batch algorithm

Parameters: $\epsilon>0$, `aggregation` $\in\{$`True`, `False`$\}$, $\eta\in[\![1,Card(S)]\!]$ the batch size
1 **Initialization:** $k\leftarrow 0$, `stay_at_x` $\leftarrow$ `True`, $i\leftarrow 1$
2 Define a partition of the subproblems $\left(S_i\right)_{i\in[\![1,\kappa]\!]}$ according to batch size η
3 **while** $i<\kappa+1$ **do**
4 $k\leftarrow k+1$
5 Solve $(RMP)^{(k)}$ and retrieve $\check{x}^{(k)},(\theta_s^{(k)})_{s\in S}$
6 $i\leftarrow 1$, $\epsilon_1\leftarrow\epsilon$, `stay_at_x` $\leftarrow$ `True`
7 Choose a permutation σ of $[\![1,\kappa]\!]$
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 π_s (dual extreme solution)
11 **if** `aggregation` **then**
12 Add $\sum_{s\in S_{\sigma(i)}} p_s\theta_s\geqslant\sum_{s\in S_{\sigma(i)}} p_s\left(\pi_s^\top(d_s-T_sx)\right)$ to $(RMP)^{(k)}$
13 **else**
14 **for** $s\in S_{\sigma(i)}$ **do**
15 Add $\theta_s\geqslant\pi_s^\top(d_s-T_sx)$ to $(RMP)^{(k)}$
16 **if** $\sum_{s\in S_{\sigma(i)}} p_s\left(\phi(\check{x}^{(k)},s)-\theta_s^{(k)}\right)\leqslant\epsilon_i$ **then**
17 $\epsilon_{i+1}\leftarrow\epsilon_i-\sum_{s\in S_{\sigma(i)}} p_s\left(\phi\left(\check{x}^{(k)},s\right)-\theta_s^{(k)}\right)$
18 $i\leftarrow i+1$
19 **else** `stay_at_x` $\leftarrow$ `False`
20 $(RMP)^{(k+1)}\leftarrow(RMP)^{(k)}$
21 Return $\check{x}^{(k)}$

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 each 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:

1. the subproblems of the current batch $S_{\sigma(i)}$ are solved (lines 9 to 10). This part of the algorithm can be parallelized, as in the classical Benders decomposition algorithm to accelerate the procedure.

2. the cuts defined by the solutions of the subproblems are added to the relaxed master program (lines 11 to 15). We add a parameter `aggregation` to the algorithm. If this parameter is set to `False`, the cuts of each subproblem are added independently to the relaxed master program, as it is the case in the classical multicut Benders decomposition algorithm. If this parameter is set to `True`, we add only one cut, computed as the weighted sum of all the cuts of the batch according to the probability distribution.

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

Remark 2. ***Aggregation of the cuts :*** *One of the most important drawback of the multicut Benders decomposition algorithm is the large number of cuts added to the relaxed master program at each iteration. As this number of cuts increases, the time needed to solve the master program can increase dramatically. The Benders by batch algorithm might suffer from the same effect, even if this effect might be delayed by the method (it adds fewer cuts at each iteration). We propose to aggregate the cuts of a batch, and add only one cut computed as:*

$$\sum_{s \in S_{\sigma(i)}} p_s \theta_s \geqslant \sum_{s \in S_{\sigma(i)}} p_s \left(\pi_s^\top (d_s - T_s x)\right)$$

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. The Benders by batch algorithm could also benefit from the methods proposed by Trukhanov et al. (2010) to find more effective aggregation schemes.

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

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

Proof. Proof of proposition 2 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 $\left(\breve{x}^{(k)}, (\breve{\theta}_s)\right)$, 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 $\left(\breve{x}^{(k)}, (\breve{\theta}_s)\right)$, then the aggregated cut of the batch also separates it, and the result remains true. □

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 in starting from the first batch of subproblems that has not been solved at the previous first-stage solution. We introduce the following cyclic permutation μ of the batches:

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

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

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

4 Stabilization of the Benders by batch algorithm

The Benders by batch algorithm introduced in the previous section (Algorithm 2) may suffer, as every cutting-plane algorithm, from strong oscillations of the first-stage variables, and thus show an erratic decrease in the value of the upper bound over the iterations. We present two primal stabilization schemes of the algorithm based on the in-out separation approach (Ben-Ameur and Neto, 2007). We introduce an adaptation of the stopping criterion presented in section 3, and detail our stabilized Benders by batch algorithm (Algorithm 3). We finally prove the finite convergence and exact behavior of our stabilized algorithm.

We first recall the principle of the in-out separation approach. At an iteration k, this method solves the subproblems at a solution $x^{(k)}$, referred to as the separation point, that may not be the solution $\check{x}^{(k)}$ to relaxed master program $(RMP)^{(k)}$. We also define $\hat{x}^{(k)}$ the stability center at iteration k, set as the previous separation point with the lowest objective function value :

$$\hat{x}^{(k)} = \underset{j \in [\![0,k-1]\!]}{\operatorname{Arg\,min}} \{c^\top x^{(j)} + \sum_{s \in S} p_s \phi(x^{(j)}, s)\}$$

The separation point $x^{(k)}$ is defined on the segment between $\hat{x}^{(k)}$ (in point) to $\check{x}^{(k)}$ (out point) :

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

The in-out separation scheme 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. In the Benders by batch algorithm, we generally do not solve all the subproblems at a given iteration. We therefore need to adapt this approach.

We propose now two separation schemes to stabilize our algorithm. They satisfy the following property: If the solution to $(RMP)^{(k)}$ is constant over an infinite sequence of iterations, then $\lim_{k \to +\infty} x^{(k)} = \check{x}^{(k)}$. We will refer hereafter to this property as *Property P*.

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

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

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

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

Method 2 - Solution memory stabilization: This stabilization uses an exponentially weighted average of the previous master solutions to compute the separation point. We choose a stabilization parameter $\alpha \in [0, 1)$

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

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

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

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

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

converges to z.

Proof. Proof of lemma 1 See Appendix A □

When using those stabilization procedures, the subproblems are not solved at the solution to the relaxed master program. We therefore adapt the stopping criterion proposed in Proposition 1 for our stabilized algorithm. We rewrite the optimality condition taking into account that the first-stage solutions at which we compute the lower bound and the upper bound are not the same.

Definition 2. *$\epsilon_i(x)$-approximation at a first-stage solution x: Let ϵ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration and σ a permutation of $[\![1,\kappa]\!]$. 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 the unstabilized version of the algorithm (see Definition 1).*

The following proposition gives a valid stopping criterion for the stabilized Benders by batch algorithm.

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

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

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

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

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

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

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

Algorithm 3: Stabilized Benders by batch

Parameters: $\epsilon > 0$, `aggregation` $\in \{$`True`, `False`$\}$, $\eta \in [\![1, Card(S)]\!]$ the batch size, $\alpha \in (0,1]$, $\beta \in [0,1)$

Initialization: $k \leftarrow 0$, `stay_at_x` $\leftarrow$ `True`, $i \leftarrow 1$, $\alpha^{(k)} \leftarrow \alpha$, `misprice` $\leftarrow$ `False`

Define a partition of the subproblems $\left(S_i\right)_{i \in [\![1,\kappa]\!]}$ according to batch size η

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

  Solve $(RMP)^{(k+1)}$ and retrieve $\check{x}^{(k+1)}$, $(\check{\theta}_s^{(k+1)})_{s \in S}$

  `misprice_cnt` $\leftarrow 0$

  do

    $k \leftarrow k + 1$

    Compute $x^{(k)}$ according to a separation scheme statisfying Property P

    $i \leftarrow 1$, $\epsilon_i \leftarrow \epsilon - c^\top (x^{(k)} - \check{x}^{(k)})$, `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 π_s (dual extreme solution)

      if `aggregation` **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 `aggregation` **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`

    (Optional) $\alpha^{(k)} \leftarrow$ *mispricing_procedure*(`misprice`, $\alpha, \alpha^{(k)}$, `misprice_cnt`) (Algo. 4)

    $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$, $\check{x}^{(k+1)} \leftarrow \check{x}^{(k)}$, $(\check{\theta}_s^{(k+1)})_{s \in S} \leftarrow (\check{\theta}_s^{(k)})_{s \in S}$

  while `misprice`

Return $x^{(k)}$

We now present the stabilized Benders by batch algorithm (Algorithm 3). The algorithm is structured in three nested *while loops*. The *while loop* from line 3 to 32 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 6 to 32 is called the *separation loop*. This loop updates the current separation point $x^{(k)}$ with the solution to the relaxed master program $\check{x}^{(k)}$ constant. The *while loop* from line 12 to 29 is called the *optimality loop*. In the optimality loop, the subproblems of current batch $S_{\sigma(i)}$ are solved. There are three possibilities at the end of this loop:

- **Case 1:** The current batch is $\epsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$. It satisfies the condition of line 20 of Algorithm 3. Then, `stay_at_x` still equals `True` at the end of the loop, and i is incremented by one. If the algorithm reaches $i = \kappa + 1$, then the algorithm stops, and $x^{(k)}$ is an optimal solution to the problem. Otherwise, the algorithm solves the next batch of subproblems at the same first-stage solution.
- **Case 2:** The current batch $S_{\sigma(i)}$ is not $\epsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$ and at least one of the cuts derived from this batch of subproblems separates the solution $(\check{x}^{(k)}, (\theta_s^{(k)})_{s\in S})$ of the relaxed master program [see FIG. 1]. This means that `misprice` is set to `False`. The variable `stay_at_x` is set to `False` and we exit the optimality loop. Since `misprice` equals `False`, we exit the separation loop. We then go to line 3, and solve again the relaxed master program.
- **Case 3:** The current batch $S_{\sigma(i)}$ is not $\epsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$ and there exists no cuts derived from this batch of subproblems, or a previous batch, which separates the solution $(\check{x}^{(k)}, (\theta_s^{(k)})_{s\in S})$ of the relaxed master program [see FIG. 2]. The variable `misprice` still equals `True`. This is called a mis-pricing (Pessoa et al., 2013). As the solution to the relaxed master program has not been cut, it is useless to solve the relaxed master program again, its solution remains the same. We exit the optimality loop, but stay in the separation loop. We define a new separation point $x^{(k)}$, a new permutation of $[\![1, \kappa]\!]$, and begin a new optimality loop in this point.

If a mis-pricing occurs, we can increase parameter $\alpha^{(k)}$, such that, after t successive mis-pricings, $\alpha^{(k+t)} = \min\{1.0, t.\alpha^{(k)}\}$ (see Algorithm 4). This is an acceleration procedure to limit the number of successive iterations which induce a mis-pricing (Pessoa et al., 2013). This procedure is optional, as the sequence of separation points $x^{(k)}$ converges to the solution to the relaxed master program for the two proposed separation schemes during a sequence of mis-pricings, and no mis-pricing is possible at $\check{x}^{(k)}$.

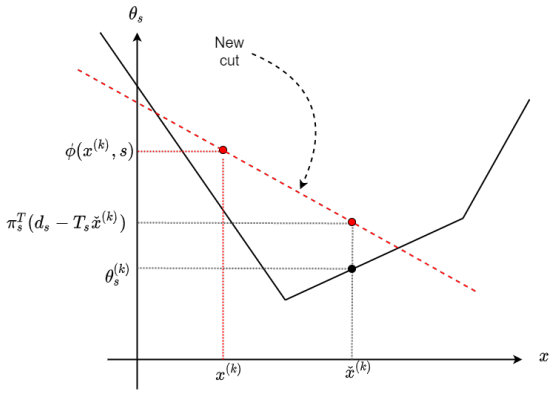

Figure 1: The cut derived from first-stage solution $x^{(k)}$ separates the solution to the relaxed master program $(\check{x}^{(k)}, (\theta_s^{(k)})_{s\in S})$.

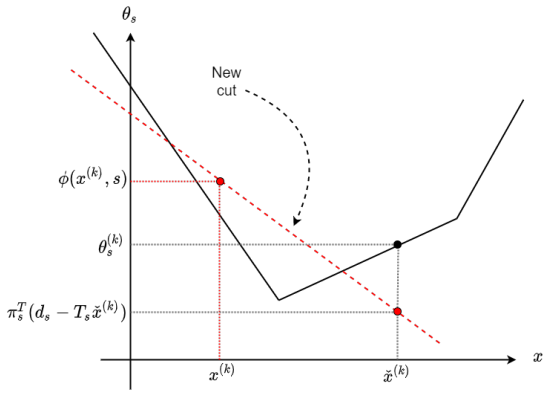

Figure 2: The cut derived from first-stage solution $x^{(k)}$ does not separate the solution to the relaxed master program $(\check{x}^{(k)}, (\theta_s^{(k)})_{s\in S})$. The solution to $(RMP)^{(k)}$ remains the same. The separation point $x^{(k)}$ induces a mispricing.

Algorithm 4: mispricing_procedure

Parameter: `misprice` $\in \{$`True`, `False`$\}, \alpha, \alpha^{(k)},$ `misprice_cnt`

if `misprice` **then**
  `misprice_cnt` $\leftarrow$ `misprice_cnt` $+ 1$
  $\alpha^{(k)} \leftarrow \min\{1.0, \left(1 + \frac{1}{\texttt{misprice\_cnt}}\right)\alpha^{(k-1)}\}$
else
  $\alpha^{(k)} \leftarrow \alpha$
Return $\alpha^{(k)}$

With the use of Proposition 4, we now prove in Proposition 5 that the stabilized Benders by batch algorithm converges to an optimal solution to problem (1) in a finite number of iterations.

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

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

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

- **Case 1.** There exists $t_0 \in \mathbb{N}$ such that for all $l \geqslant t_0$ 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 $\frac{\epsilon}{4}$
- **Case 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 $\frac{\epsilon}{4}$

Case 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 $i \in [\![1, \kappa]\!]$,

$$\Big[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \Big]^+ \leqslant \frac{\epsilon}{4} - \Big[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \Big]^+ \tag{8}$$

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 $i \in [\![1, \kappa]\!]$:

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

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 $i \in [\![1, \kappa]\!]$:

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

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

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 (8), (9) and (10), 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).

Case 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:

$$\Big[\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \Big]^+ > \frac{\epsilon}{4} - \Big[\sum_{t=1}^{i-1} \sum_{s \in S_{\sigma^{(k+l)}(t)}} p_s \left(\phi \left(\check{x}^{(k)}, s \right) - \check{\theta}_s^{(k)} \right) \Big]^+ \tag{11}$$

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 (11) occurs) such that:

$$\sum_{s \in S_{\sigma^{(k+l)}(i)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) > \delta \tag{12}$$

Let $g_i^{(k+\tau)} \in \mathbb{R}^{n_1}$ be a subgradient associated with the function $x \mapsto \sum_{s \in S_{\sigma^{(k+\tau)}(i)}} p_s \phi(x^{(k+\tau)}, s)$ at point $x^{(k+\tau)}$. The aggregated cut derived by resolution of 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 $i \in [\![1, \kappa]\!]$, $|g_i^{(k+l)\top}(\check{x} - x^{(k+l)})| < \frac{\delta}{3}$.
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 $i \in [\![1, \kappa]\!]$:

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

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 (12) occurs. We have:

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

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

Proposition 5. *The stabilized Benders by batch algorithm converges to an optimal solution to problem* (1) *in a finite number of iterations.*

Proof. 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).
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 a finite number of times in this situation.
3. There exists no cut derived at $x^{(k)}$ which separates the solution to $(RMP)^{(k)}$. Then, $x^{(k)}$ induces a mis-pricing. The solution to $(RMP)^{(k+1)}$ remains the same. Let suppose that this happens during an infinite number of consecutive iterations. Then, as the separation scheme satisfies Property P, the sequence of separation points converges to $\check{x}^{(k)}$. Prop. 4 states that in that case, we end 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: As the classical Benders decomposition algorithm can be seen as the Benders by batch algorithm with a batch size $\eta = Card(S)$, then Proposition 5 is still valid for the classical Benders decomposition algorithm. Every stabilization method producing a sequence of iterates converging asymptotically to the solution to the relaxed master program induces an algorithm converging in a finite number of iterations to an optimal solution to the problem.

5 Experimentations and numerical results

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

5.1 Instances

We use six well studied instances from the literature. The first five, 20term (Mak et al., 1999), gbd (Dantzig, 1963), LandS (Louveaux and Smeers, 1988), ssn (Sen et al., 1994) and storm (Mulvey and Ruszczyński, 1995), are available from the following link: `www.cs.wisc.edu/\simswright/stochastic/sampling/`. The problem 20term is taken from (Mak et al., 1999). It is a model of motor freight carrier's operations. The problem consists in choosing the position of some vehicles at the beginning of the day, the first-stage variables, and then to use those vehicles to satisfy some random demands on a network. Instance gbd has been created from chapter 28 of (Dantzig, 1963). It is an aircraft allocation problem. LandS has been created from an electrical investment planning problem described in (Louveaux and Smeers, 1988). In (Linderoth et al., 2006), the authors modified the problem to obtain an instance with 10^6 scenarios. Problem ssn is a problem of telecommunication network design taken from (Sen et al., 1994) and storm is a cargo flight scheduling problem described by (Mulvey and Ruszczyński, 1995). The last instance, Fleet20_3, was found at `http://www.ie.tsinghua.edu.cn/lzhao/` which was itself taken from `https://people.orie.cornell.edu/huseyin/research/research.html`. It is a fleet-sizing problem, close to 20term, with a two-week horizon planning.

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

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

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

5.2 Experimentations

In order to evaluate the numerical efficiency of our algorithm, we compare it to five different methods:

1. IMB ILOG CPLEX 12.10 with its multicut Benders implementation
2. Our implementation of the multicut Benders decomposition
3. Our implementation of the monocut Benders decomposition
4. Our implementation of the multicut Benders decomposition with an in-out stabilization
5. Our implementation of the monocut Benders decomposition with an in-out stabilization

The Benders decomposition algorithm of IMB ILOG CPLEX 12.10 is run with the following parameter values: benders strategy 2 (An annotation file contains the first-stage variables, and CPLEX decomposes automatically the subproblems), threads 1 (to run CPLEX using one core, as the other methods), timelimit 43200 (time limit of twelve hours).

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

$$\sum_{s\in S} p_s\theta_s \geqslant \sum_{s\in S} p_s\Big(\pi_s.(d_s - T_s x)\Big)$$

We also use a dynamic strategy to update the stabilization parameter in our implementation of *in-out* stabilization. We recall that we denote by $x^{(k)}$ the separation point at iteration k of the algorithm, and by $\hat{x}^{(k)}$ the stability center, which is computed as:

$$\hat{x}^{(k)} = \underset{0\leqslant i\leqslant k-1}{\text{Arg min}} \{c^\top x^{(i)} + \sum_{s\in S} p_s\phi(x^{(i)}, s)\}$$

We use the following rule to update parameter α. 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 our stabilization gave us a better first-stage solution according to the cost than the current stability center. If we had separated farther, we could have found an even better point, and we therefore 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 also evaluate different parameters of the Benders by Batch algorithm. We first run the Benders by Batch algorithm without stabilization, and try different batch sizes with and without aggregation. Then, we evaluate the impact of the two proposed stabilizations, with different stabilization parameters.

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

5.3 Numerical results

This section shows the numerical results obtained on our 72 instances. When an algorithm is stopped at its time limit of 12 hours (43 200s), the computing time is denoted $+inf$, 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 and in Appendix 1 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. The classical multicut Benders algorithm is denoted as *Classic multicut* and its monocut version as *Classic monocut*. Our Benders by batch algorithm is denoted as *BbB*. Our implementation of multicut in-out stabilization is denoted as *In-out multicut* and its monocut version as *In-out monocut*. The Benders decomposition implementation of IBM ILOG CPLEX 12.10 is denoted as *CPLEX*. We always present the average time on the three instances of each size for each problem, rounded to the second.

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

problem $p \in \mathcal{P}$ as:

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

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

Table 3: Classical Benders decomposition algorithms results
Times (left columns) are in second, ratios (right columns) are computed as the time divided by the best time to solve the instance by methods present in the table.

		Cplex Benders		Classic multicut		Classic monocut		In-out multicut		In-Out monocut	
instance	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	1	**1**	**1.0**	1	1.3	2	3.4	1	1.2	2	2.7
LandS-N5000	5	**5**	**1.0**	9	1.7	11	2.0	7	1.3	9	1.8
LandS-N10000	15	**15**	**1.0**	29	1.9	22	1.5	22	1.4	17	1.2
LandS-N20000	31	43	1.4	105	3.4	45	1.5	73	2.4	**31**	**1.0**
gbd-N1000	1	**1**	**1.0**	1	1.2	2	2.7	1	1.2	2	3.0
gbd-N5000	10	**10**	**1.0**	10	1.1	12	1.2	11	1.1	12	1.3
gbd-N10000	23	34	1.4	33	1.4	**23**	**1.0**	35	1.5	**23**	**1.0**
gbd-N20000	48	131	2.7	121	2.5	**48**	**1.0**	128	2.7	**48**	**1.0**
ssn-N1000	7	15	2.3	**7**	**1.0**	2408	357.7	**7**	**1.0**	131	19.4
ssn-N5000	57	83	1.5	**57**	**1.0**	13460	236.8	**58**	**1.0**	746	13.1
ssn-N10000	174	**180**	**1.0**	188	1.1	25901	148.6	**174**	**1.0**	1502	8.6
ssn-N20000	485	**485**	**1.0**	**488**	**1.0**	+inf	>89.0	531	1.1	2747	5.7
storm-N1000	11	28	2.6	**11**	**1.0**	24	2.2	**11**	**1.0**	18	1.7
storm-N5000	91	187	2.1	106	1.2	114	1.3	117	1.3	**91**	**1.0**
storm-N10000	190	508	2.7	496	2.6	224	1.2	455	2.4	**190**	**1.0**
storm-N20000	376	1396	3.7	2370	6.3	458	1.2	1753	4.7	**376**	**1.0**
20term-N1000	101	780	7.7	757	7.5	577	5.7	265	2.6	**101**	**1.0**
20term-N5000	556	+inf	>77.7	24429	43.9	3506	6.3	9240	16.6	**556**	**1.0**
20term-N10000	1116	+inf	>38.7	+inf	>38.7	6901	6.2	41461	37.2	**1116**	**1.0**
20term-N20000	2160	+inf	>20.0	+inf	>20.0	13687	6.3	+inf	>20.0	**2160**	**1.0**
Fleet20-N1000	70	148	2.1	225	3.2	533	7.6	**70**	**1.0**	105	1.5
Fleet20-N5000	501	15720	31.4	5330	10.6	2757	5.5	1843	3.7	**501**	**1.0**
Fleet20-N10000	1070	+inf	>40.4	28933	27.0	5710	5.3	7807	7.3	**1070**	**1.0**
Fleet20-N20000	2260	+inf	>19.1	+inf	>19.1	11300	5.0	+inf	>19.1	**2260**	**1.0**

We first present in Table 3 the results on the five methods we use to benchmark the Benders by batch algorithm. We notice that IBM ILOG CPLEX 12.10 implementation of Benders decomposition is the less efficient method among the 5 presented in Table 3, and does not scale well when the number of subproblems becomes large. It succeeds in solving only 57 out of 72 instances. Even if CPLEX implements an in-out stabilization, and a multicut strategy with the option *benders strategy 2*, it seems to be not competitive with our multicut in-out implementation. We also remark that the in-out stabilization performs almost always better in term of computing time than its classical counterpart (monocut versus monocut or multicut versus multicut). Finally, we observe that the multicut implementations do not scale well when the number of subproblems becomes large (except for ssn instances). The restricted master programs tend to be too long to solve because of the large amount of cuts added. In the remaining tables, we only report the In-out monocut and multicut results.

We now present the results of the Benders by batch algorithm without stabilization. We analyze the impact of the batch size, both without (Table 4) and with aggregation (Table 5). Each column of Tables 4 and 5 contains the average time in second to solve the instances and the ratio to the best time. We analyze batch sizes from 1% to 20% of the total number of subproblems.

We first notice in Table 4 that a batch size of one percent of the subproblems allows the Benders by batch algorithm to solve almost all the instances without aggregation (except for Fleet20_3 with 20000 subproblems where it succeeds to solve only one out of three problems), where the classical multicut Benders decomposition

fails. The saving of subproblem resolutions and cuts added to the relaxed master program allows to overcome the computing time issues in both the subproblems and the master problem resolutions. However, the Benders by batch algorithm without aggregation is not competitive with the in-out monocut Benders decomposition, except for ssn instances, where it can be up to 3.7 times faster.

Table 5 shows that the aggregated Benders by batch algorithm is almost all the time the best method, and, with batch sizes of 1% and 5% of the total number of subproblems, is almost always faster than the in-out monocut Benders decomposition. As we aggregate the cuts over each batch, the size of the relaxed master program remains reasonable, and as the cuts are only computed on a sample of subproblems, the algorithm avoids many symmetries due to the sum of the cuts over the subproblems.

Table 4: Benders by batch algorithm without aggregation

		In-out multicut		In-Out monocut		BbB 1%		BbB 5%		BbB 10%		BbB 20%	
instance	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	1	1	1.1	2	2.4	2	2.7	1	1.3	1	1.1	**1**	**1.0**
LandS-N5000	6	7	1.1	9	1.5	13	2.2	8	1.3	7	1.1	**6**	**1.0**
LandS-N10000	17	22	1.3	**17**	**1.0**	38	2.2	25	1.5	21	1.2	20	1.1
LandS-N20000	31	73	2.4	**31**	**1.0**	130	4.2	89	2.9	80	2.6	72	2.3
gbd-N1000	1	1	1.4	2	3.7	2	3.6	**1**	**1.0**	1	1.3	1	1.5
gbd-N5000	6	11	1.7	12	2.0	16	2.5	**6**	**1.0**	7	1.1	8	1.3
gbd-N10000	19	35	1.8	23	1.2	47	2.5	**19**	**1.0**	22	1.2	25	1.3
gbd-N20000	48	128	2.7	**48**	**1.0**	96	2.0	61	1.3	71	1.5	87	1.8
ssn-N1000	4	7	1.7	131	33.2	6	1.5	**4**	**1.0**	4	1.1	5	1.2
ssn-N5000	24	58	2.5	746	31.7	32	1.4	**24**	**1.0**	28	1.2	32	1.4
ssn-N10000	59	174	3.0	1502	25.5	71	1.2	79	1.3	**59**	**1.0**	79	1.3
ssn-N20000	145	531	3.7	2747	19.0	**145**	**1.0**	274	1.9	624	4.3	2821	19.5
storm-N1000	6	11	1.7	18	2.8	21	3.2	8	1.3	**6**	**1.0**	8	1.3
storm-N5000	55	117	2.1	91	1.7	175	3.2	60	1.1	**55**	**1.0**	65	1.2
storm-N10000	156	455	2.9	190	1.2	492	3.1	**156**	**1.0**	**159**	**1.0**	189	1.2
storm-N20000	376	1753	4.7	**376**	**1.0**	1390	3.7	580	1.5	672	1.8	588	1.6
20term-N1000	38	265	7.0	101	2.6	**38**	**1.0**	82	2.1	49	1.3	74	1.9
20term-N5000	556	9240	16.6	**556**	**1.0**	634	1.1	2101	3.8	1335	2.4	2247	4.0
20term-N10000	1116	41461	37.2	**1116**	**1.0**	2270	2.0	10733	9.6	6199	5.6	10413	9.3
20term-N20000	2160	+inf	>20.0	**2160**	**1.0**	20625	9.6	+inf	>20.0	+inf	>20.0	+inf	>20.0
Fleet20-N1000	70	**70**	**1.0**	105	1.5	145	2.1	95	1.4	102	1.5	74	1.1
Fleet20-N5000	501	1843	3.7	**501**	**1.0**	2417	4.8	1950	3.9	1873	3.7	2097	4.2
Fleet20-N10000	1070	7807	7.3	**1070**	**1.0**	9903	9.3	19913	18.6	8537	8.0	21383	20.0
Fleet20-N20000	2260	+inf	>19.1	**2260**	**1.0**	34867	15.4	+inf	>19.1	+inf	>19.1	+inf	>19.1

We summarize the results of Tables 4 and 5 in Figure 3. Figure 3 confirms that the Benders by batch algorithms with aggregation and batch sizes of 1% and 5% are the two best algorithms on our benchmark. They show the best computing times in respectively 24 out of 72 instances and 33 out of 72. The Benders by batch algorithm with aggregation and a batch size of 1% of the number of subproblems scales better as its higher performance ratio is lower than 4.

We now present the results for the two stabilization schemes presented in Section 4. We performed the computations for the algorithm with batch sizes of 1% and 5%, and with aggregation, as they were the most competitive ones in the unstabilized results. Figures 4 and 5 show the performance profiles of the stabilized methods with their unstabilized equivalent. We present the results of Benders by batch with basic stabilization, for values of $\alpha \in \{0.1, 0.5, 0.9\}$ and with solution memory stabilization with $\alpha \in \{0.1, 0.5, 0.9\}$ and $\beta \in \{0.1, 0.5, 0.9\}$.

Figure 4 shows that all the proposed stabilizations accelerate the Benders by batch algorithm with a batch size of 1% of the subproblems, and can be up to 70% faster than their unstabilized equivalent. Four stabilizations are more efficient on the tested instances and rather equivalent, 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 a batch size of 5% of the subproblems. The same four methods are the most efficient and equivalent to each other. One of the stabilizations, the solution memory stabilization with $(\alpha, \beta) = (0.1, 0.9)$ is less efficient than the unstabilized algorithm. In this case, a small step size ($\alpha = 0.1$)

Table 5: Benders by batch with aggregation

		In-out multicut		In-Out monocut		BbB 1% Aggreg		BbB 5% Aggreg		BbB 10% Aggreg		BbB 20% Aggreg	
instance	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	1	**1**	**1.0**	2	2.2	2	2.3	1	1.2	1	1.1	1	1.2
LandS-N5000	4	7	1.7	9	2.3	9	2.3	5	1.1	**4**	**1.0**	4	1.1
LandS-N10000	8	22	2.6	17	2.1	16	1.9	**8**	**1.0**	**8**	**1.0**	9	1.1
LandS-N20000	17	73	4.2	31	1.8	44	2.5	**17**	**1.0**	**18**	**1.0**	20	1.1
gbd-N1000	1	1	1.5	2	4.0	2	2.7	**1**	**1.0**	1	1.3	1	1.5
gbd-N5000	3	11	3.2	12	3.7	9	2.7	**3**	**1.0**	4	1.1	4	1.3
gbd-N10000	6	35	5.5	23	3.7	15	2.3	**6**	**1.0**	8	1.3	9	1.5
gbd-N20000	14	128	9.4	48	3.5	41	3.0	**14**	**1.0**	15	1.1	19	1.4
ssn-N1000	7	**7**	**1.0**	131	19.4	14	2.0	61	9.1	134	19.9	242	36.0
ssn-N5000	58	**58**	**1.0**	746	12.8	89	1.5	322	5.5	659	11.3	1322	22.8
ssn-N10000	174	**174**	**1.0**	1502	8.6	185	1.1	707	4.1	1423	8.2	2914	16.7
ssn-N20000	441	531	1.2	2747	6.2	**441**	**1.0**	1615	3.7	3386	7.7	6757	15.3
storm-N1000	6	11	1.7	18	2.9	12	1.9	**6**	**1.0**	7	1.1	9	1.5
storm-N5000	34	117	3.4	91	2.7	52	1.5	**34**	**1.0**	**36**	**1.0**	55	1.6
storm-N10000	74	455	6.2	190	2.6	110	1.5	**74**	**1.0**	82	1.1	104	1.4
storm-N20000	163	1753	10.8	376	2.3	226	1.4	**163**	**1.0**	**169**	**1.0**	238	1.5
20term-N1000	15	265	18.1	101	6.9	**15**	**1.0**	37	2.5	68	4.6	141	9.6
20term-N5000	70	9240	131.9	556	7.9	**70**	**1.0**	193	2.7	395	5.6	839	12.0
20term-N10000	130	41461	319.7	1116	8.6	**130**	**1.0**	402	3.1	898	6.9	1978	15.2
20term-N20000	280	+inf	>154.4	2160	7.7	**280**	**1.0**	914	3.3	2051	7.3	18312	65.4
Fleet20-N1000	28	70	2.5	105	3.7	**28**	**1.0**	42	1.5	74	2.6	131	4.6
Fleet20-N5000	107	1843	17.2	501	4.7	**107**	**1.0**	211	2.0	358	3.3	649	6.0
Fleet20-N10000	212	7807	36.8	1070	5.0	**212**	**1.0**	440	2.1	721	3.4	1310	6.2
Fleet20-N20000	419	+inf	>103.2	2260	5.4	**419**	**1.0**	876	2.1	1520	3.6	2777	6.6

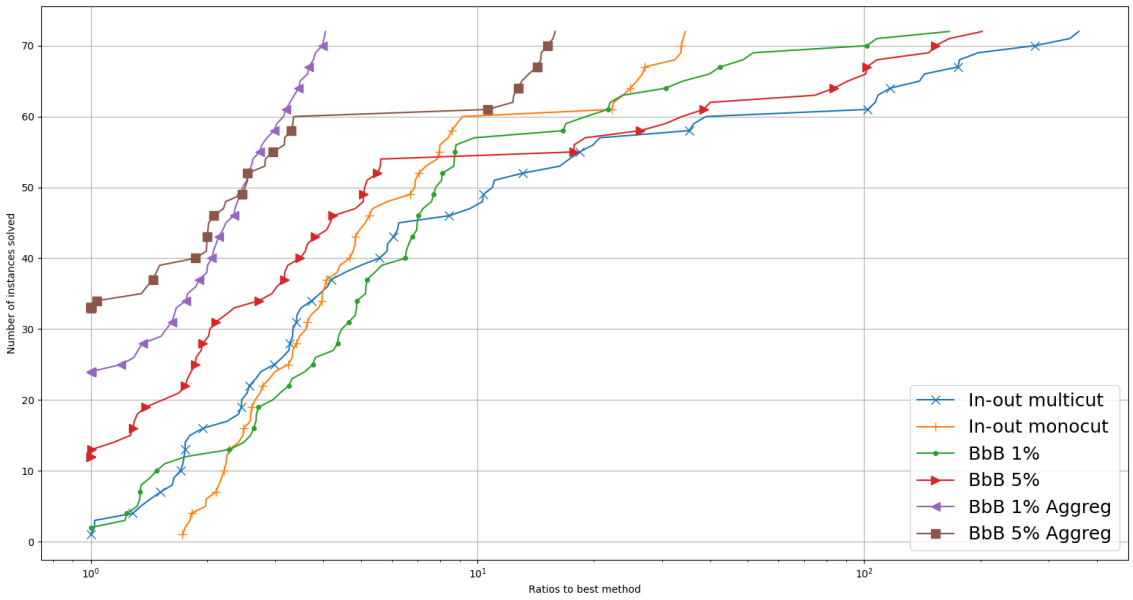

Figure 3: Performance profiles of Classical Benders decomposition with in-out stabilization and Benders by batch algorithm with and without aggregation

and a high memory parameter ($\beta = 0.9$) slow down the convergence. For all the other cases, the stabilization accelerates the algorithm.

Finally, there is no clear difference between the two stabilizations proposed, according to the results. It seems that the memory based 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.

As a final result, we show in Figure 6 the performance profiles of the first five methods we compare to, presented in Table 3 and the Benders by batch algorithm solving 1% of the subproblems at each iteration, and a basic stabilization with $\alpha = 0.5$. The stabilized Benders by batch is the best algorithm for 62 out of 72 instances. We show factors up to 800 times faster than the Benders decomposition of IMB ILOG Cplex 12.10,

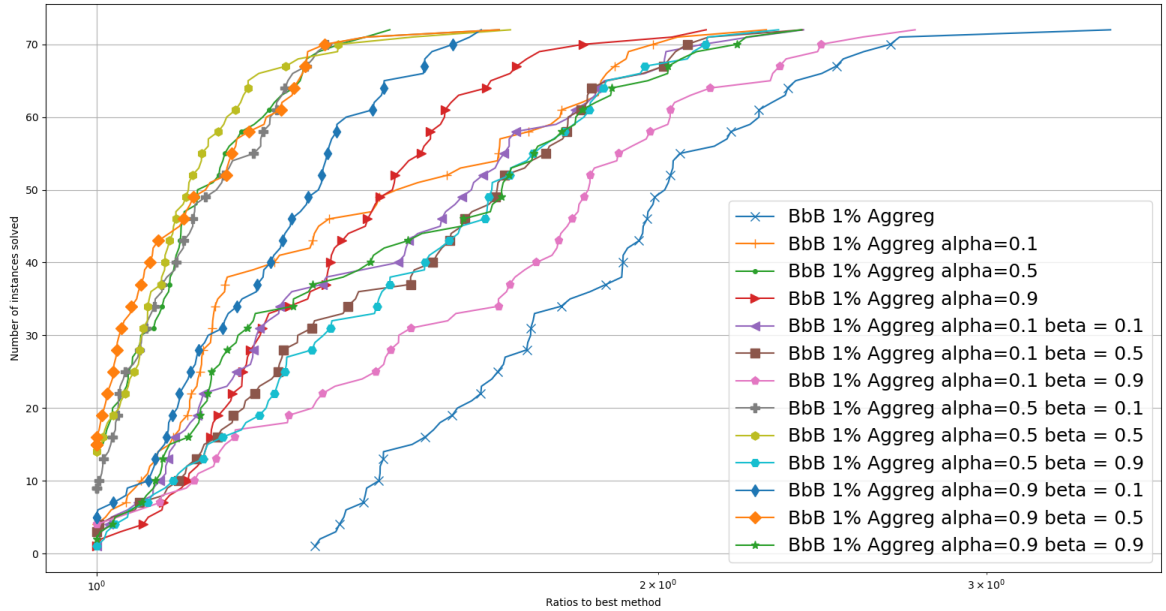

Figure 4: Performance profiles of Benders by batch with stabilizations and batch size of 1% and Benders by Batch with batch size of 1% and aggregation. *BbB* stands for Benders by batch.

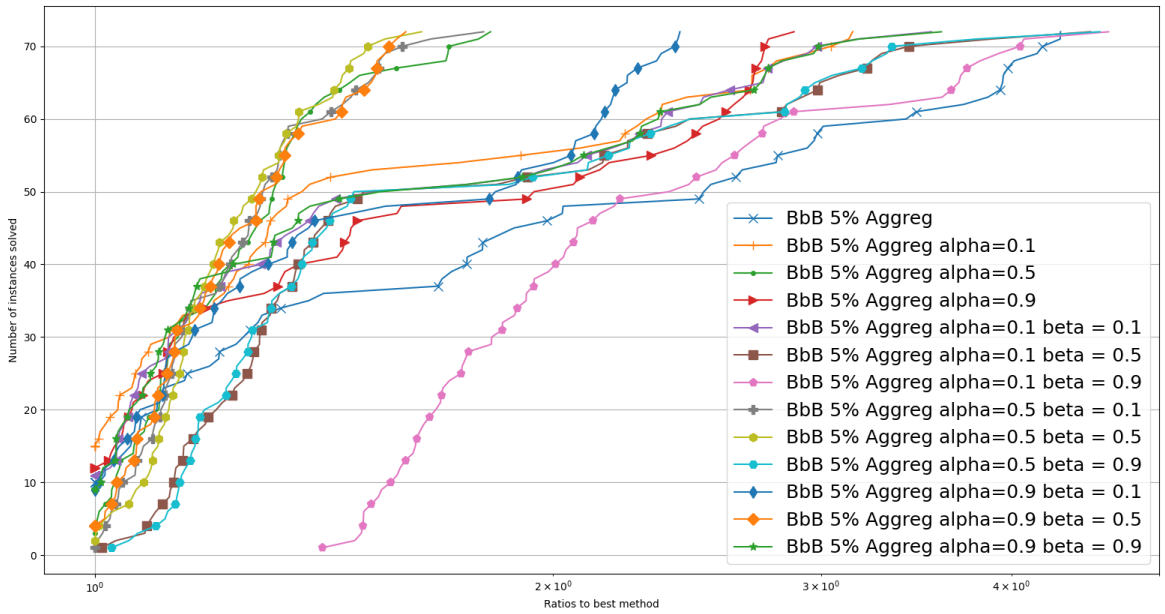

Figure 5: Performance profiles of Benders by batch with stabilizations and batch size of 5% and Benders by Batch with batch size of 5% and aggregation. *BbB* stands for Benders by batch.

and more than 300 times faster than the classical Benders decompositions without stabilization.

6 Conclusion

We proposed in this article the Benders by batch algorithm to solve two-stage stochastic linear programs. In this algorithm, we do not solve all the subproblems at each iteration. This idea has been explored in the literature in a monocut framework (Higle and Sen, 1991; Dantzig and Infanger, 1991), but these algorithms were not exact or needed some assumptions on the structure of the problem. We showed that solving only a very few number of subproblems, 1% in our tests, allows us to significantly improve the solution time, and to

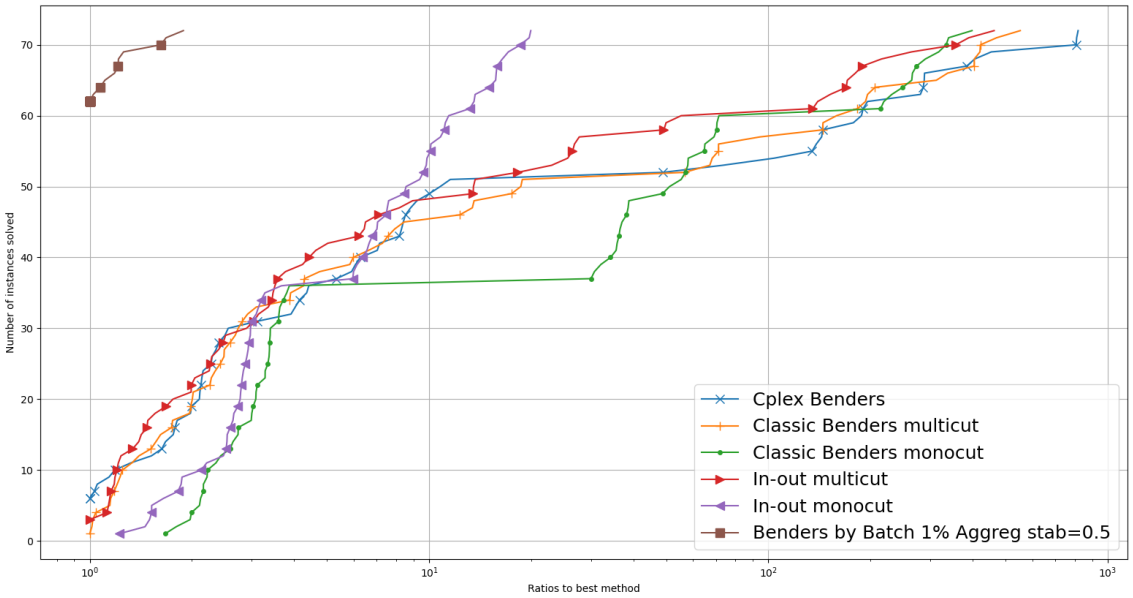

Figure 6: Performance profile of initial methods and best stabilized Benders by batch algorithm

solve large instances that classical Benders decomposition algorithms fails to solve in 12 hours.

We also proposed two methods to stabilize this algorithm. The monocut adaptation of the Benders by batch algorithm with those stabilizations allowed us to solve in at most seven minutes some large instances of the literature which where not solved in 12 hours by the built-in Benders decomposition of CPLEX 12.10, the Benders decomposition without stabilization, or the multicut Benders decomposition with in-out stabilization. This algorithm showed acceleration factors from 2 to more than 10 times compared to the best method of the literature we compared to.

Applying dual stabilization (Magnanti and Wong, 1981; Sherali and Lunday, 2013) to the Benders by batch algorithm is straightforward and could improve the results. The algorithm can be parallelized, as well as the classical Benders decomposition, and may benefit from the improvements of parallelized methods, such as the asynchronous method of Linderoth and Wright (2003). Finally, a major improvement should be to adapt the Benders by batch algorithm to mixed-integer master programs within a Branch&Cut framework. As there is no need to solve all the subproblems in a random node of the tree, one could apply this method at each node in which the solution to the relaxation is not integer. It could also be simply used to solve the Benders reformulation at the root node of the tree, to initialize the resolution.

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.

References

Ben-Ameur, W. and Neto, J. (2007). Acceleration of cutting-plane and column generation algorithms: Applications to network design. *Networks*, 49(1):3–17.

Birge, J. R. and Louveaux, F. (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34(3):384–392.

Crainic, T. G., Hewitt, M., Maggioni, F., and Rei, W. (2020). Partial Benders Decomposition: General Methodology and Application to Stochastic Network Design. *Transportation Science*, 55(2):414–435.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton university press edition.

Dantzig, G. B. and Glynn, P. W. (1990). Parallel processors for planning under uncertainty. *Annals of Operations Research*, (22):1–21.

Dantzig, G. B. and Infanger, G. (1991). Large-Scale Stochastic Linear Programs: Importance Sampling and Benders Decomposition:. Technical report, Defense Technical Information Center.

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.

Higle, J. L. and Sen, S. (1991). Stochastic Decomposition : An Algorithm for Two-Stage Linear Programms with Recourse. *Mathematics of Operations Research*, 16(3):650–669.

Infanger, G. (1992). Monte Carlo (importance) sampling within a benders decomposition algorithm for stochastic linear programs. *Annals of Operations Research*, 39(1):69–95.

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.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. *Operations Research*, 29(3):464–484.

Mak, W.-K., Morton, D. P., and Wood, R. (1999). Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1-2):47–56.

Mulvey, J. M. and Ruszczyński, A. (1995). A New Scenario Decomposition Method for Large-Scale Stochastic Optimization. *Operations Research*, 43(3):477–490.

Oliveira, W., Sagastizábal, C., and Scheimberg, S. (2011). Inexact Bundle Methods for Two-Stage Stochastic Programming. *SIAM Journal on Optimization*, 21(2):517–544.

Papadakos, N. (2008). Practical enhancements to the Magnanti–Wong method. *Operations Research Letters*, 36(4):444–449.

Pessoa, A., Sadykov, R., Uchoa, E., and Vanderbeck, F. (2013). In-Out Separation and Column Generation Stabilization by Dual Price Smoothing. In *Experimental Algorithms*, volume 7933, pages 354–365.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817.

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.

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.

Trukhanov, S., Ntaimo, L., and Schaefer, A. (2010). Adaptive multicut aggregation for two-stage stochastic linear programs with recourse. *European Journal of Operational Research*, 206(2):395–406.

van Ackooij, W., de Oliveira, W., and Song, Y. (2017). Adaptive Partition-Based Level Decomposition Methods for Solving Two-Stage Stochastic Programs with Fixed Recourse. *INFORMS Journal on Computing*, 30(1):57–70.

Van Slyke, R. M. and Wets, R. (1969). L-Shaped Linear Programs with Applications to Optimal Control and Stochastic Programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

Wets, R. (1983). Stochastic Programming: Solution Techniques and Approximation Schemes. In *Mathematical Programming The State of the Art*, pages 566–603.

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.

Zaourar, S. and Malick, J. (2014). Quadratic stabilization of Benders decomposition. https://hal.archives-ouvertes.fr/hal-01181273.

Appendix A: Proof of Lemma 1

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

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

We define $u^{(k)} = x^{(k+1)} - z$ and $v^{(k)} = \bar{x}^{(k+1)} - z$, for every $k \in \mathbb{N}^*$.

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

We define

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

Then we have:

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

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

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

Since $k\alpha^k(1-\alpha)$ converges to 0, $\forall\ 0 \leqslant \alpha < 1$, the sequence $(A^k)_{k\in\mathbb{N}}$ converges to the null matrix. This proves that the sequence $(u^{(k)}, v^{(k)})_{k\in\mathbb{N}}$ converges to $(0,0)$. Then the sequence $\left(x^{(k)}\right)_{k\in\mathbb{N}}$ converges to z. □

Appendix B: Detailed numerical results

This section shows the detailed numerical results.

		Cplex Benders		Classic multicut		Classic monocut		In-out multicut		In-Out monocut	
instance	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	1	**1**	**1.0**	1	1.4	2	3.4	1	1.1	1	2.5
LandS-N1000-s1001	1	**1**	**1.0**	1	1.3	2	3.8	1	1.4	2	3.0
LandS-N1000-s1002	1	**1**	**1.0**	1	1.1	2	3.1	1	1.2	2	2.5
LandS-N5000-s5000	5	**5**	**1.0**	9	1.7	11	2.1	6	1.2	8	1.6
LandS-N5000-s5001	6	**6**	**1.0**	10	1.7	10	1.8	7	1.2	10	1.9
LandS-N5000-s5002	5	**5**	**1.0**	9	1.7	11	2.1	7	1.4	9	1.8
LandS-N10000-s10000	15	**15**	**1.0**	26	1.7	22	1.5	21	1.3	17	1.1
LandS-N10000-s10001	15	**15**	**1.0**	30	2.0	22	1.5	22	1.5	17	1.2
LandS-N10000-s10002	14	**14**	**1.0**	30	2.1	20	1.4	21	1.5	18	1.2
LandS-N20000-s20000	30	44	1.4	96	3.2	49	1.6	75	2.5	**30**	**1.0**
LandS-N20000-s20001	31	43	1.4	119	3.8	43	1.4	70	2.3	**31**	**1.0**
LandS-N20000-s20002	31	44	1.4	99	3.2	44	1.4	74	2.4	**31**	**1.0**
gbd-N1000-s1000	1	**1**	**1.0**	1	1.2	2	2.4	1	1.2	2	3.1
gbd-N1000-s1001	1	**1**	**1.0**	1	1.1	2	2.9	1	1.1	2	2.6
gbd-N1000-s1002	1	**1**	**1.0**	1	1.2	2	2.7	1	1.1	3	3.3
gbd-N5000-s5000	10	**10**	**1.0**	10	1.1	13	1.3	11	1.1	13	1.3
gbd-N5000-s5001	10	**10**	**1.0**	10	1.1	11	1.2	11	1.1	13	1.3
gbd-N5000-s5002	10	**10**	**1.0**	11	1.1	12	1.2	10	1.1	11	1.2
gbd-N10000-s10000	23	33	1.4	34	1.5	**24**	**1.0**	36	1.5	**23**	**1.0**
gbd-N10000-s10001	23	34	1.5	32	1.4	**24**	**1.0**	34	1.5	**23**	**1.0**
gbd-N10000-s10002	23	33	1.5	32	1.4	**23**	**1.0**	35	1.6	24	1.1
gbd-N20000-s20000	46	130	2.8	119	2.6	**48**	**1.0**	128	2.8	**46**	**1.0**
gbd-N20000-s20001	51	131	2.6	120	2.3	**51**	**1.0**	127	2.5	**52**	**1.0**
gbd-N20000-s20002	46	132	2.8	125	2.7	**47**	**1.0**	130	2.8	**46**	**1.0**
ssn-N1000-s1000	7	16	2.4	**7**	**1.0**	2279	339.8	**7**	**1.0**	134	19.9
ssn-N1000-s1001	7	16	2.3	**7**	**1.0**	2720	398.2	**7**	**1.0**	135	19.7
ssn-N1000-s1002	7	14	2.1	**7**	**1.0**	2226	334.0	**7**	**1.0**	124	18.6
ssn-N5000-s5000	57	82	1.5	62	1.1	13425	236.5	**57**	**1.0**	796	14.0
ssn-N5000-s5001	45	81	1.8	**45**	**1.0**	14260	317.7	54	1.2	769	17.1
ssn-N5000-s5002	63	85	1.4	**64**	**1.0**	12695	201.4	**63**	**1.0**	672	10.7
ssn-N10000-s10000	163	**163**	**1.0**	185	1.1	26559	162.7	206	1.3	1505	9.2
ssn-N10000-s10001	190	**190**	**1.0**	**193**	**1.0**	26228	137.9	**191**	**1.0**	1508	7.9
ssn-N10000-s10002	126	188	1.5	187	1.5	24916	197.7	**126**	**1.0**	1492	11.8
ssn-N20000-s20000	462	**478**	**1.0**	512	1.1	+inf	>93.4	**462**	**1.0**	2807	6.1
ssn-N20000-s20001	484	**484**	**1.0**	**503**	**1.0**	+inf	>89.3	678	1.4	2736	5.7
ssn-N20000-s20002	450	494	1.1	**450**	**1.0**	+inf	>96.0	**452**	**1.0**	2697	6.0
storm-N1000-s1000	10	28	2.7	**10**	**1.0**	23	2.2	**11**	**1.0**	18	1.8
storm-N1000-s1001	11	28	2.7	**11**	**1.0**	24	2.2	**11**	**1.0**	18	1.7
storm-N1000-s1002	11	28	2.6	11	1.1	24	2.3	**11**	**1.0**	18	1.7
storm-N5000-s5000	92	191	2.1	100	1.1	110	1.2	112	1.2	**92**	**1.0**
storm-N5000-s5001	85	183	2.2	118	1.4	117	1.4	114	1.3	**85**	**1.0**
storm-N5000-s5002	96	188	1.9	**99**	**1.0**	116	1.2	124	1.3	**96**	**1.0**
storm-N10000-s10000	188	525	2.8	468	2.5	215	1.1	455	2.4	**188**	**1.0**
storm-N10000-s10001	201	516	2.6	479	2.4	225	1.1	469	2.3	**201**	**1.0**
storm-N10000-s10002	181	482	2.7	542	3.0	233	1.3	441	2.4	**181**	**1.0**
storm-N20000-s20000	381	1381	3.6	2240	5.9	465	1.2	1730	4.5	**381**	**1.0**
storm-N20000-s20001	351	1524	4.3	2460	7.0	434	1.2	1780	5.1	**351**	**1.0**
storm-N20000-s20002	396	1283	3.2	2410	6.1	476	1.2	1750	4.4	**396**	**1.0**
20term-N1000-s1000	105	817	7.8	749	7.2	544	5.2	256	2.4	**105**	**1.0**
20term-N1000-s1001	98	559	5.7	646	6.6	584	6.0	302	3.1	**98**	**1.0**
20term-N1000-s1002	100	965	9.6	877	8.7	604	6.0	238	2.4	**100**	**1.0**
20term-N5000-s5000	538	+inf	>80.3	29455	54.8	3095	5.8	9621	17.9	**538**	**1.0**
20term-N5000-s5001	541	+inf	>79.8	22490	41.5	3699	6.8	9140	16.9	**541**	**1.0**
20term-N5000-s5002	590	+inf	>73.3	21342	36.2	3725	6.3	8960	15.2	**590**	**1.0**
20term-N10000-s10000	1087	+inf	>39.7	+inf	>39.7	6803	6.3	+inf	>39.7	**1087**	**1.0**
20term-N10000-s10001	1081	+inf	>40.0	+inf	>40.0	6404	5.9	+inf	>40.0	**1081**	**1.0**
20term-N10000-s10002	1179	+inf	>36.6	+inf	>36.6	7494	6.4	37982	32.2	**1179**	**1.0**
20term-N20000-s20000	2051	+inf	>21.1	+inf	>21.1	13429	6.5	+inf	>21.1	**2051**	**1.0**
20term-N20000-s20001	2174	+inf	>19.9	+inf	>19.9	12763	5.9	+inf	>19.9	**2174**	**1.0**
20term-N20000-s20002	2254	+inf	>19.2	+inf	>19.2	14868	6.6	+inf	>19.2	**2254**	**1.0**
Fleet20-N1000-s1000	48	147	3.0	224	4.6	513	10.6	**48**	**1.0**	100	2.1
Fleet20-N1000-s1001	84	141	1.7	228	2.7	539	6.4	**84**	**1.0**	107	1.3
Fleet20-N1000-s1002	77	155	2.0	224	2.9	546	7.1	**77**	**1.0**	109	1.4
Fleet20-N5000-s5000	476	14769	31.0	5530	11.6	2780	5.8	2150	4.5	**476**	**1.0**
Fleet20-N5000-s5001	496	21496	43.3	5090	10.3	2760	5.6	1360	2.7	**496**	**1.0**
Fleet20-N5000-s5002	531	10894	20.5	5370	10.1	2730	5.1	2020	3.8	**531**	**1.0**
Fleet20-N10000-s10000	1060	+inf	>40.8	29600	27.9	5860	5.5	7540	7.1	**1060**	**1.0**
Fleet20-N10000-s10001	1010	+inf	>42.8	28200	27.9	5480	5.4	7580	7.5	**1010**	**1.0**
Fleet20-N10000-s10002	1140	+inf	>37.9	29000	25.4	5790	5.1	8300	7.3	**1140**	**1.0**
Fleet20-N20000-s20000	2350	+inf	>18.4	+inf	>18.4	11400	4.9	+inf	>18.4	**2350**	**1.0**
Fleet20-N20000-s20001	2250	+inf	>19.2	+inf	>19.2	11500	5.1	+inf	>19.2	**2250**	**1.0**
Fleet20-N20000-s20002	2180	+inf	>19.8	+inf	>19.8	11000	5.0	+inf	>19.8	**2180**	**1.0**

Table 6: Classical Benders decomposition algorithms

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

Table 7: basic stabilization with batch size of 1%

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

Table 8: basic stabilization with batch size of 5%

		BbB 1% Ag		BbB 1% $\alpha = 0.1$ $\beta = 0.1$		BbB 1% $\alpha = 0.1$ $\beta = 0.5$		BbB 1P $\alpha = 0.1$ $\beta = 0.9$		BbB 1P $\alpha = 0.5$ $\beta = 0.1$		BbB 1P $\alpha = 0.5$ $\beta = 0.5$		BbB 1P $\alpha = 0.5$ $\beta = 0.9$		BbB 1% $\alpha = 0.9$ $\beta = 0.1$		BbB 1% $\alpha = 0.9$ $\beta = 0.5$		BbB 1% $\alpha = 0.9$ $\beta = 0.9$	
instance	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	1	2	2.0	1	1.8	2	1.8	**1**	**1.0**	1	1.2	1	1.1	1	1.8	1	1.4	1	1.1	1	1.7
LandS-N1000-s1001	1	2	1.7	2	1.7	2	1.6	2	1.8	1	1.1	**1**	**1.0**	2	1.6	1	1.2	1	1.1	2	1.7
LandS-N1000-s1002	1	2	2.1	**1**	**1.0**	**1**	**1.0**	1	1.1	**1**	**1.0**	1	1.1	1	1.1	1	1.5	1	1.1	1	1.1
LandS-N5000-s5000	4	10	2.3	7	1.7	7	1.7	**4**	**1.0**	5	1.1	5	1.1	7	1.7	6	1.3	5	1.1	7	1.6
LandS-N5000-s5001	4	9	2.3	4	1.1	**4**	**1.0**	5	1.2	5	1.2	5	1.2	**4**	**1.0**	6	1.5	5	1.3	4	1.1
LandS-N5000-s5002	4	9	1.9	7	1.7	7	1.6	8	1.7	6	1.2	**4**	**1.0**	7	1.7	**4**	**1.0**	5	1.2	7	1.7
LandS-N10000-s10000	9	17	1.9	**9**	**1.0**	15	1.7	15	1.8	10	1.1	10	1.1	15	1.7	10	1.1	10	1.1	**9**	**1.0**
LandS-N10000-s10001	10	14	1.4	15	1.5	16	1.5	16	1.5	**11**	**1.0**	16	1.5	15	1.5	**10**	**1.0**	**11**	**1.0**	16	1.5
LandS-N10000-s10002	9	17	2.0	**9**	**1.0**	9	1.1	15	1.7	11	1.3	9	1.1	9	1.1	10	1.2	11	1.3	**9**	**1.0**
LandS-N20000-s20000	19	45	2.4	31	1.6	32	1.7	**19**	**1.0**	21	1.1	21	1.1	31	1.6	25	1.3	21	1.1	31	1.6
LandS-N20000-s20001	17	42	2.4	31	1.8	33	1.9	**17**	**1.0**	20	1.2	23	1.3	32	1.9	24	1.4	20	1.2	31	1.8
LandS-N20000-s20002	17	45	2.7	30	1.8	30	1.8	30	1.8	21	1.2	**17**	**1.0**	31	1.8	18	1.1	22	1.3	31	1.8
gbd-N1000-s1000	1	2	1.8	2	1.8	2	1.8	2	2.3	1	1.1	1	1.1	2	1.8	1	1.6	**1**	**1.0**	2	1.8
gbd-N1000-s1001	1	2	2.0	2	2.0	2	2.0	2	2.3	1	1.1	1	1.1	2	1.9	1	1.3	**1**	**1.0**	2	2.2
gbd-N1000-s1002	1	2	2.3	1	1.9	2	2.0	2	2.4	**1**	**1.0**	**1**	**1.0**	2	2.0	1	1.3	**1**	**1.0**	1	1.9
gbd-N5000-s5000	4	10	2.3	7	1.6	8	1.8	9	2.0	**4**	**1.0**	5	1.2	8	1.8	6	1.5	**4**	**1.0**	7	1.7
gbd-N5000-s5001	4	8	2.3	7	2.1	8	2.1	8	2.3	5	1.3	4	1.1	8	2.1	**4**	**1.0**	5	1.4	7	2.1
gbd-N5000-s5002	4	9	2.6	8	2.4	8	2.4	8	2.3	6	1.6	5	1.5	8	2.3	**4**	**1.0**	6	1.6	8	2.4
gbd-N10000-s10000	8	18	2.1	15	1.8	15	1.8	16	2.0	10	1.2	**9**	**1.0**	15	1.8	**8**	**1.0**	10	1.2	15	1.8
gbd-N10000-s10001	8	13	1.6	13	1.7	14	1.8	17	2.1	11	1.3	**8**	**1.0**	14	1.8	10	1.3	10	1.3	14	1.7
gbd-N10000-s10002	7	14	2.0	14	2.0	14	2.0	19	2.7	8	1.2	7	1.1	15	2.1	**7**	**1.0**	8	1.2	14	2.0
gbd-N20000-s20000	14	50	3.5	32	2.2	30	2.1	17	1.2	15	1.1	16	1.1	30	2.1	**14**	**1.0**	15	1.1	32	2.2
gbd-N20000-s20001	17	31	1.8	28	1.7	29	1.7	30	1.8	18	1.1	19	1.1	30	1.8	**17**	**1.0**	18	1.1	29	1.7
gbd-N20000-s20002	16	43	2.6	30	1.8	29	1.8	32	2.0	20	1.2	18	1.1	30	1.8	**16**	**1.0**	20	1.2	30	1.8
ssn-N1000-s1000	8	14	1.7	9	1.1	10	1.2	11	1.3	9	1.1	9	1.1	11	1.3	10	1.2	**8**	**1.0**	9	1.1
ssn-N1000-s1001	8	15	1.9	12	1.6	9	1.2	11	1.4	9	1.2	8	1.1	10	1.3	12	1.6	**8**	**1.0**	10	1.3
ssn-N1000-s1002	8	13	1.6	9	1.1	9	1.1	11	1.4	**8**	**1.0**	**8**	**1.0**	10	1.2	11	1.3	**8**	**1.0**	9	1.1
ssn-N5000-s5000	45	88	2.0	54	1.2	52	1.2	56	1.3	**47**	**1.0**	**45**	**1.0**	54	1.2	64	1.4	**46**	**1.0**	54	1.2
ssn-N5000-s5001	46	90	2.0	49	1.1	52	1.1	60	1.3	**47**	**1.0**	**46**	**1.0**	53	1.2	62	1.3	**46**	**1.0**	49	1.1
ssn-N5000-s5002	46	90	2.0	50	1.1	52	1.1	58	1.3	52	1.1	**46**	**1.0**	52	1.1	61	1.3	48	1.1	52	1.1
ssn-N10000-s10000	92	175	1.9	101	1.1	108	1.2	120	1.3	**92**	**1.0**	**95**	**1.0**	113	1.2	115	1.3	**92**	**1.0**	106	1.1
ssn-N10000-s10001	93	187	2.0	112	1.2	111	1.2	128	1.4	**93**	**1.0**	105	1.1	106	1.1	119	1.3	**93**	**1.0**	106	1.2
ssn-N10000-s10002	86	193	2.2	108	1.3	107	1.2	123	1.4	93	1.1	**86**	**1.0**	112	1.3	115	1.3	**88**	**1.0**	101	1.2
ssn-N20000-s20000	183	457	2.5	242	1.3	235	1.3	270	1.5	203	1.1	**183**	**1.0**	232	1.3	244	1.3	198	1.1	213	1.2
ssn-N20000-s20001	182	458	2.5	232	1.3	228	1.3	265	1.5	**182**	**1.0**	**186**	**1.0**	230	1.3	259	1.4	**186**	**1.0**	221	1.2
ssn-N20000-s20002	190	407	2.1	215	1.1	228	1.2	255	1.3	**190**	**1.0**	200	1.1	235	1.2	251	1. 3	**193**	**1.0**	226	1.2
storm-N1000-s1000	6	12	1.9	10	1.5	10	1.6	7	1.1	8	1.3	**6**	**1.0**	10	1.6	7	1.1	**7**	**1.0**	10	1.6
storm-N1000-s1001	6	12	2.0	7	1.1	10	1.6	7	1.2	**6**	**1.0**	7	1.2	9	1.6	7	1.2	8	1.3	7	1.1
storm-N1000-s1002	6	13	2.0	10	1.5	10	1.5	**7**	**1.0**	7	1.1	8	1.2	10	1.5	7	1.1	**6**	**1.0**	10	1.5
storm-N5000-s5000	31	44	1.4	**32**	**1.0**	33	1.1	36	1.2	**31**	**1.0**	37	1.2	33	1.1	35	1.1	**31**	**1.0**	**31**	**1.0**
storm-N5000-s5001	32	54	1.7	47	1.5	34	1.1	35	1.1	42	1.3	**32**	**1.0**	**33**	**1.0**	**33**	**1.0**	**32**	**1.0**	48	1.5
storm-N5000-s5002	32	58	1.8	34	1.1	**32**	**1.0**	33	1.1	**32**	**1.0**	**33**	**1.0**	**32**	**1.0**	37	1.2	**32**	**1.0**	34	1.1
storm-N10000-s10000	59	121	2.0	67	1.1	67	1.1	109	1.8	64	1.1	68	1.1	68	1.1	**62**	**1.0**	**59**	**1.0**	67	1.1
storm-N10000-s10001	65	90	1.4	68	1.1	**66**	**1.0**	108	1.7	**67**	**1.0**	**66**	**1.0**	**67**	**1.0**	71	1.1	**65**	**1.0**	**68**	**1.0**
storm-N10000-s10002	62	118	1.9	67	1.1	101	1.6	70	1.1	69	1.1	**64**	**1.0**	100	1.6	**62**	**1.0**	66	1.1	67	1.1
storm-N20000-s20000	127	216	1.7	139	1.1	138	1.1	144	1.1	**130**	**1.0**	**127**	**1.0**	136	1.1	152	1.2	**131**	**1.0**	139	1.1
storm-N20000-s20001	123	245	2.0	140	1.1	**129**	**1.0**	146	1.2	130	1.1	**123**	**1.0**	**128**	**1.0**	137	1.1	**126**	**1.0**	141	1.1
storm-N20000-s20002	130	218	1.7	**130**	**1.0**	**135**	**1.0**	143	1.1	141	1.1	**135**	**1.0**	**133**	**1.0**	192	1.5	152	1.2	**131**	**1.0**
20term-N1000-s1000	9	15	1.7	14	1.6	12	1.4	16	1.8	**9**	**1.0**	10	1.2	10	1.1	12	1.4	11	1.3	15	1.7
20term-N1000-s1001	10	15	1.5	15	1.6	17	1.7	18	1.9	11	1.2	11	1.2	16	1.7	12	1.3	**10**	**1.0**	**10**	**1.0**
20term-N1000-s1002	9	15	1.6	18	2.0	12	1.4	22	2.4	11	1.2	11	1.2	14	1.5	12	1.3	**9**	**1.0**	16	1.8
20term-N5000-s5000	51	67	1.3	60	1.2	67	1.3	84	1.6	**51**	**1.0**	57	1.1	**51**	**1.0**	58	1.1	66	1.3	61	1.2
20term-N5000-s5001	46	78	1.7	67	1.5	67	1.5	84	1.8	51	1.1	**46**	**1.0**	74	1.6	58	1.3	48	1.1	74	1.6
20term-N5000-s5002	45	64	1.4	70	1.5	69	1.5	117	2.6	56	1.2	**45**	**1.0**	65	1.4	55	1.2	53	1.2	68	1.5
20term-N10000-s10000	101	129	1.3	118	1.2	147	1.5	188	1.9	**101**	**1.0**	**101**	**1.0**	135	1.3	**101**	**1.0**	**102**	**1.0**	113	1.1
20term-N10000-s10001	91	122	1.3	135	1.5	139	1.5	152	1.7	115	1.3	**91**	**1.0**	151	1.7	115	1.3	110	1.2	179	2.0
20term-N10000-s10002	85	137	1.6	126	1.5	126	1.5	169	2.0	**87**	**1.0**	**85**	**1.0**	102	1.2	136	1.6	101	1.2	176	2.1
20term-N20000-s20000	193	261	1.3	**193**	**1.0**	330	1.7	367	1.9	226	1.2	244	1.3	361	1.9	251	1.3	222	1.2	276	1.4
20term-N20000-s20001	224	296	1.3	289	1.3	337	1.5	326	1.5	241	1.1	243	1.1	272	1.2	236	1.1	**224**	**1.0**	267	1.2
20term-N20000-s20002	178	283	1.6	237	1.3	**178**	**1.0**	361	2.0	233	1.3	212	1.2	288	1.6	254	1.4	230	1.3	337	1.9
Fleet20-N1000-s1000	17	28	1.6	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-N1000-s1001	17	27	1.6	18	1.1	20	1.1	24	1.4	**17**	**1.0**	**18**	**1.0**	22	1.3	19	1.1	**18**	**1.0**	18	1.1
Fleet20-N1000-s1002	18	30	1.7	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-N5000-s5000	76	108	1.4	89	1.2	96	1.3	125	1.6	**78**	**1.0**	84	1.1	95	1.2	83	1.1	**76**	**1.0**	89	1.2
Fleet20-N5000-s5001	76	104	1.4	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-N5000-s5002	74	110	1.5	93	1.3	101	1.4	137	1.8	**74**	**1.0**	79	1.1	107	1.4	82	1.1	**77**	**1.0**	101	1.4
Fleet20-N10000-s10000	152	214	1.4	184	1.2	197	1.3	270	1.8	**155**	**1.0**	**159**	**1.0**	191	1.3	163	1.1	**152**	**1.0**	174	1.1
Fleet20-N10000-s10001	155	209	1.3	183	1.2	193	1.2	240	1.5	**155**	**1.0**	164	1.1	193	1.2	180	1.2	**156**	**1.0**	175	1.1
Fleet20-N10000-s10002	142	213	1.5	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-N20000-s20000	307	402	1.3	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-N20000-s20001	301	429	1.4	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-N20000-s20002	305	425	1.4	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

Table 9: Solution memory stabilization with batch size of 1%

		BbB 5% Ag		BbB 5% $\alpha = 0.1$ $\beta = 0.1$		BbB 5% $\alpha = 0.1$ $\beta = 0.5$		BbB 5P $\alpha = 0.1$ $\beta = 0.9$		BbB 5P $\alpha = 0.5$ $\beta = 0.1$		BbB 5P $\alpha = 0.5$ $\beta = 0.5$		BbB 5P $\alpha = 0.5$ $\beta = 0.9$		BbB 5% $\alpha = 0.9$ $\beta = 0.1$		BbB 5% $\alpha = 0.9$ $\beta = 0.5$		BbB 5% $\alpha = 0.9$ $\beta = 0.9$	
instance	Best	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	1	1	1.2	1	1.3	1	1.1	2	2.8	1	1.2	**1**	**1.0**	1	1.1	**1**	**1.0**	1	1.2	1	1.1
LandS-N1000-s1001	1	**1**	**1.0**	2	2.4	1	1.3	2	2.6	1	1.1	1	1.2	1	1.2	1	1.2	1	1.1	2	2.3
LandS-N1000-s1002	1	1	1.4	1	1.4	1	1.3	1	1.7	1	1.1	**1**	**1.0**	1	1.3	1	1.1	**1**	**1.0**	1	1.3
LandS-N5000-s5000	4	5	1.1	8	2.1	9	2.1	10	2.5	5	1.1	**4**	**1.0**	9	2.2	**4**	**1.0**	4	1.1	8	2.0
LandS-N5000-s5001	4	5	1.4	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	**4**	**1.0**	9	2.3	9	2.2	10	2.7	4	1.1	5	1.2	9	2.2	**4**	**1.0**	4	1.1	9	2.3
LandS-N10000-s10000	8	**8**	**1.0**	10	1.2	10	1.3	14	1.7	9	1.1	**8**	**1.0**	11	1.3	**8**	**1.0**	9	1.1	10	1.2
LandS-N10000-s10001	8	**8**	**1.0**	18	2.2	18	2.3	15	1.8	9	1.1	10	1.2	18	2.2	9	1.1	**8**	**1.0**	18	2.2
LandS-N10000-s10002	8	**8**	**1.0**	18	2.2	18	2.3	22	2.7	**8**	**1.0**	**8**	**1.0**	18	2.3	**8**	**1.0**	**8**	**1.0**	18	2.2
LandS-N20000-s20000	17	**17**	**1.0**	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	17	18	1.1	36	2.1	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	**18**	**1.0**	37	2.1	38	2.1	48	2.7	20	1.1	20	1.1	38	2.1	19	1.1	20	1.1	37	2.1
gbd-N1000-s1000	1	**1**	**1.0**	2	3.5	2	3.9	2	4.6	1	1.5	1	1.6	2	3.8	1	1.6	1	1.5	2	3.6
gbd-N1000-s1001	1	**1**	**1.0**	2	2.6	2	3.4	2	3.6	1	1.3	1	1.3	2	3.3	1	1.1	1	1.2	2	2.8
gbd-N1000-s1002	1	**1**	**1.0**	2	2.8	2	3.0	2	3.7	1	1.5	1	1.5	2	2.9	1	1.1	1	1.5	2	2.8
gbd-N5000-s5000	3	**3**	**1.0**	9	2.8	10	2.9	12	3.7	4	1.1	4	1.3	10	2.9	4	1.1	**3**	**1.0**	9	2.7
gbd-N5000-s5001	3	3	1.1	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	3	1.1	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	2.5
gbd-N10000-s10000	7	**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	**6**	**1.0**	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	**6**	**1.0**	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	**12**	**1.0**	19	1.5	37	3.0	49	3.9	16	1.3	14	1.1	37	3.0	14	1.2	16	1.3	19	1.5
gbd-N20000-s20001	15	**15**	**1.0**	37	2.5	46	3.1	49	3.3	17	1.2	20	1.3	45	3.0	**15**	**1.0**	17	1.2	37	2.5
gbd-N20000-s20002	14	**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	15	63	4.1	**15**	**1.0**	**16**	**1.0**	22	1.4	18	1.2	17	1.1	16	1.1	32	2.1	19	1.2	**15**	**1.0**
ssn-N1000-s1001	15	63	4.2	**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	15	59	3.9	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	89	337	3.8	**89**	**1.0**	94	1.1	127	1.4	111	1.2	96	1.1	99	1.1	172	1.9	112	1.3	**93**	**1.0**
ssn-N5000-s5001	85	322	3.8	**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	90	308	3.4	95	1.1	99	1.1	141	1.6	113	1.3	100	1.1	99	1.1	172	1.9	116	1.3	**90**	**1.0**
ssn-N10000-s10000	185	672	3.6	**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	187	760	4.1	209	1.1	231	1.2	301	1.6	235	1.3	217	1.2	211	1.1	439	2.3	244	1.3	**187**	**1.0**
ssn-N10000-s10002	184	690	3.7	**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	432	1651	3.8	**432**	**1.0**	491	1.1	672	1.6	531	1.2	524	1.2	492	1.1	866	2.0	529	1.2	**432**	**1.0**
ssn-N20000-s20001	446	1651	3.7	474	1.1	485	1.1	728	1.6	561	1.3	516	1.2	475	1.1	893	2.0	551	1.2	**446**	**1.0**
ssn-N20000-s20002	434	1543	3.6	**434**	**1.0**	506	1.2	650	1.5	554	1.3	499	1.2	489	1.1	914	2.1	558	1.3	**450**	**1.0**
storm-N1000-s1000	6	**6**	**1.0**	10	1.8	12	1.9	11	1.8	**6**	**1.0**	7	1.1	12	1.9	6	1.1	**6**	**1.0**	10	1.8
storm-N1000-s1001	6	6	1.1	8	1.3	8	1.4	10	1.8	6	1.1	6	1.1	8	1.4	**6**	**1.0**	6	1.1	8	1.3
storm-N1000-s1002	6	6	1.1	7	1.2	11	1.8	10	1.7	6	1.1	**6**	**1.0**	11	1.9	**6**	**1.0**	6	1.1	7	1.2
storm-N5000-s5000	30	33	1.1	40	1.3	40	1.3	56	1.9	**30**	**1.0**	33	1.1	41	1.4	**30**	**1.0**	**30**	**1.0**	40	1.3
storm-N5000-s5001	30	33	1.1	37	1.2	39	1.3	52	1.7	**31**	**1.0**	34	1.1	39	1.3	**30**	**1.0**	**31**	**1.0**	37	1.2
storm-N5000-s5002	29	37	1.3	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	60	73	1.2	78	1.3	127	2.1	114	1.9	65	1.1	65	1.1	127	2.1	**60**	**1.0**	64	1.1	79	1.3
storm-N10000-s10001	60	76	1.3	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	59	73	1.2	116	2.0	82	1.4	118	2.0	67	1.1	70	1.2	82	1.4	**59**	**1.0**	66	1.1	117	2.0
storm-N20000-s20000	138	167	1.2	148	1.1	173	1.3	256	1.9	**138**	**1.0**	**140**	**1.0**	172	1.2	**139**	**1.0**	**138**	**1.0**	147	1.1
storm-N20000-s20001	125	161	1.3	180	1.4	186	1.5	243	1.9	**127**	**1.0**	143	1.1	185	1.5	**125**	**1.0**	**127**	**1.0**	181	1.4
storm-N20000-s20002	127	160	1.3	153	1.2	170	1.3	240	1.9	141	1.1	148	1.2	171	1.3	**127**	**1.0**	141	1.1	153	1.2
20term-N1000-s1000	15	36	2.4	**15**	**1.0**	21	1.4	30	2.0	22	1.4	20	1.3	18	1.2	27	1.8	20	1.3	16	1.1
20term-N1000-s1001	15	37	2.4	**16**	**1.0**	18	1.1	31	2.0	18	1.2	19	1.2	17	1.1	29	1.9	21	1.4	**15**	**1.0**
20term-N1000-s1002	12	37	3.0	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	71	199	2.8	**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	66	197	3.0	**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	73	182	2.5	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	140	411	2.9	**145**	**1.0**	192	1.4	285	2.0	208	1.5	201	1.4	158	1.1	302	2.2	217	1.6	**140**	**1.0**
20term-N10000-s10001	161	409	2.5	**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	147	388	2.6	**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	306	860	2.8	**306**	**1.0**	339	1.1	635	2.1	488	1.6	444	1.5	449	1.5	685	2.2	490	1.6	**309**	**1.0**
20term-N20000-s20001	284	985	3.5	**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	298	897	3.0	315	1.1	389	1.3	710	2.4	458	1.5	439	1.5	390	1.3	697	2.3	465	1.6	**298**	**1.0**
Fleet20-N1000-s1000	24	42	1.8	**24**	**1.0**	27	1.1	36	1.5	**24**	**1.0**	26	1.1	**24**	**1.0**	30	1.2	**24**	**1.0**	**24**	**1.0**
Fleet20-N1000-s1001	22	40	1.8	**22**	**1.0**	25	1.1	33	1.5	24	1.1	25	1.1	26	1.2	30	1.3	**23**	**1.0**	**22**	**1.0**
Fleet20-N1000-s1002	24	43	1.8	**25**	**1.0**	28	1.2	36	1.5	26	1.1	26	1.1	29	1.2	32	1.3	**25**	**1.0**	**24**	**1.0**
Fleet20-N5000-s5000	110	218	2.0	**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-N5000-s5001	122	209	1.7	**123**	**1.0**	143	1.2	188	1.5	**122**	**1.0**	**126**	**1.0**	**127**	**1.0**	151	1.2	**122**	**1.0**	**125**	**1.0**
Fleet20-N5000-s5002	122	205	1.7	**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-N10000-s10000	226	426	1.9	**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-N10000-s10001	230	467	2.0	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-N10000-s10002	226	426	1.9	**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-N20000-s20000	524	886	1.7	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-N20000-s20001	476	856	1.8	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-N20000-s20002	528	885	1.7	**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

Table 10: Solution memory stabilization with batch size of 5%