Solving large-scale stochastic optimization programs : application to investment problems for power systems

Xavier Blanchot

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE MATHEMATIQUES ET INFORMATIQUE

SPECIALITE MATHEMATIQUES APPLIQUEES ET CALCUL SCIENTIFIQUE

Par **Xavier BLANCHOT**

Résolution de problèmes d'optimisation stochastique de grande taille : application à des problèmes d'investissement dans les réseaux électriques

Sous la direction de : **François CLAUTIAUX**

soutenue le 5 décembre 2022

Membres du jury :

Mme. Luce BROTCORNE	Directrice de Recherche	INRIA Lille Nord-Europe	Rapporteure
M. Pierre FOUILHOUX	Professeur	Université Sorbonne Paris Nord	Rapporteur
Mme. Céline GICQUEL	Maître de conférences	Université Paris Saclay	Examinatrice
M. Andrés RAMOS GALAN	Professeur	Universidad Pontificia Comillas	Examinateur
M. François CLAUTIAUX	Professeur	Université de Bordeaux	Directeur
M. Aurélien FROGER	Maître de conférence	Université de Bordeaux	Co-encadrant
M. Manuel RUIZ	Docteur	RTE	Invité

Président du jury :

M. Pierre FOUILHOUX Professeur Université Sorbonne Paris Nord

Résumé : Cette thèse s'intéresse à la modélisation et la résolution de problèmes d'optimisation stochastique de grande taille provenant de problèmes d'investissements dans les réseaux électriques. Ces travaux ont été motivés par la complexité numérique des problèmes d'investissement stochastiques résolus dans le cadre d'études prospectives menées à RTE (Réseau de Transport d'Electricité), le gestionnaire de réseau électrique français. A partir d'un réseau, ces problèmes modélisent des investissement sur des capacités de production sur les noeuds, ou des capacités de transport sur les arcs pour satisfaire des demandes en électricité sur un horizon de temps donné avec des pas de temps discrets. Les demandes en électricité, les prévision météorologiques ou les coûts opérationnels sont aléatoires et modélisés par un ensemble fini de scénarios d'aléas. On s'intéresse à deux problèmes particuliers.

Premièrement, nous présentons un algorithme amélioré basé sur une décomposition de Benders pour résoudres de grand problèmes linéaires stochastiques à deux étapes modélisés avec un ensemble fini de scénarios. De tels modèles sont utilisés dans les études prospectives menées à RTE mais les résolutions actuelles sont limitées dans le nombre de scénarios apparaissant dans les modèles pour conserver des temps de calculs raisonnables. L'algorithme proposé est basé sur une partition des sous-problèmes et sur une modification du critère d'arrêt. Le critère d'arrêt proposé permet de ne résoudre qu'un petit sous-ensemble de sous-problèmes à la plupart des itérations de l'algorithme. Nous présentons aussi une méthode générique pour stabiliser notre algorithme, et montrons son efficacité au travers d'une étude numérique étendue. Les résultats montrent que l'algorithme proposé peut être jusqu'à 25 fois plus rapide qu'un algorithme de faisceaux (level bundle) et jusqu'à 5 fois plus rapide qu'un algorithme basé sur la décomposition de Benders avec une stabilisation *in-out* et une méthode statique d'aggrégation de coupes.

Deuxièmement, nous étudions un modèle d'expansion de réseau stochastique prenant en compte une contrainte de fiabilité. Cette contrainte à pour but de limiter, en moyenne sur les scénarios d'aléas, le nombre de noeuds et de pas de temps auxquels la demande n'est pas entièrement satisfaite dans une solution du problème. Nous montrons que le fait d'ajouter une telle contrainte, qui prend en compte des variables provenant de différents scénarios dans sa formulation, implique de modéliser le problème comme un problème bi-niveaux. Les fonctions objectifs sont les mêmes dans les deux niveaux du problème. Malgré cette structure particulière, nous montrons que le problème n'est pas équivalent à sa *high-point relaxation*. Le problème obtenu est un très grand problème bi-niveaux en variables mixtes, et les algorithmes classiques pour résoudre ces problèmes ne sont pas en mesure de résoudre des problèmes de cette taille. Nous présentons une méthode heuristique basée sur une décomposition de Benders et une dichotomie sur les coûts d'investissements pour trouver des solutions réalisables de ce problème. Cette méthode est capable de trouver des solutions réalisables sur des problèmes générés aléatoirement avec plusieurs millions de variables et contraintes.

Mots-clés : Optimisation des systèmes de grande taille, Programmation stochastique, Décomposition de Benders, Optimisation bi-niveaux, Programmation linéaire, Programmation linéaire en nombres entiers

---

Abstract: This thesis addresses large-scale stochastic optimization programs arising from investment problems in power systems. This work has been motivated by the numerical complexity of stochastic investment problems that are solved in prospective studies conducted by RTE (Réseau de Transport d'Electricité), the French transmission system operator. Considering a network, these problems consist of investing on production capacities on the nodes or transmission capacities on the arcs in order to satisfy demands during a discrete time horizon. We model stochastic demands, weather forecasts or operational costs by a finite number of random scenarios. We focus on two particular numerically challenging optimization problems.

Firstly, we present an enhanced Benders decomposition algorithm to solve large-scale two-stage linear stochastic programs formulated with a large discrete set of scenarios. Such problems arise in prospective studies performed at RTE but models are currently limited in the number of scenarios to maintain reasonable computing times. The proposed algorithm relies on a partition of the subproblems into separated batches and a modified stopping criterion. This stopping criterion allows to solve only a few subproblems at most iterations. We also present a generic framework to stabilize the proposed algorithm, and show its efficiency on an extensive numerical study. We show, on the tested instances, that our algorithm can be up to 25 times faster than Benders decomposition with level bundle stabilization or 5 times faster than Benders decomposition with in-out stabilization and static cut aggregation.

Secondly, we study an extension of the proposed stochastic linear expansion planning problem, taking into account a network reliability constraint. This constraint aims at limiting the expected number of nodes and time steps at which the demand is not fully satisfied in an optimal solution. We show that adding such a constraint, involving variables from different scenarios, leads to a bilevel formulation of the problem, and show how a single-level linear formulation may lead to inconsistent solutions. As the resulting problem is a very large-scale mixed-integer bilevel program, we do not expect classical bilevel algorithms to solve real-life instances. We introduce a heuristic method based on Benders decomposition and on a binary search on the investment costs to find feasible solutions. We show that the proposed heuristic is able to find feasible solutions on randomly generated instances with up to several million variables and constraints.

Keywords: Large-scale optimization, Stochastic programming, Benders decomposition, Bilevel optimization, Linear programming, Integer linear programming

---

Unité de recherche
UMR CNRS 5251 Université de Bordeaux, 33400 Talence, France.

Résumé long [French]

Suite aux accords de Paris votés durant la COP21, le gouvernement français prévoit de ne plus produire d'électricité à partir d'énergies fossiles et d'atteindre la neutralité carbone à l'horizon 2050. Ces objectifs impliquent de modifier en profondeur les sources de production d'électricité et la structure du réseau électrique français (RTE, 2021). Les études prospectives évaluant les stratégies d'investissement concernant les moyens de production ou de transport d'électricité ont de ce fait une importance cruciale dans les prises de décisions publiques. Nous nous intéressons dans cette thèse à deux problèmes d'optimisation mathématique provenant de telles études d'investissement dans les réseaux électriques.

Contexte industriel

RTE (Réseau de Transport d'Electricité) est le gestionnaire du réseau de transport d'électricité français, et est responsable du réseau haute tension en France. L'entreprise est notamment garante du maintien de l'équilibre entre la consommation et la production d'électricité à tout instant. De par son rôle majeur dans l'équilibre offre-demande en France, la législation française impose à RTE de produire chaque année des études prospectives concernant l'évolution du réseau électrique à des horizons allant de 5 à 30 ans. Ces études ont pour but d'évaluer les risques de déséquilibres entre la demande et la production électriques en France, et d'analyser des stratégies d'investissement sur le réseau pour assurer la sécurité d'approvisionnement selon différents scénarios économiques, politiques ou environnementaux.

Dans les études auxquelles on s'intéresse ici, le réseau de transport est modélisé de manière grossière. La France est modélisée par une dizaine de noeuds, tandis que les pays voisins sont généralement modélisés par entre un et trois noeuds. A chaque noeud, on modélise une capacité de production agrégée pour chaque source d'énergie (solaire, éolien, nucléaire,...). Ces études ont pour objectif d'analyser des tendances globales et d'évaluer s'il serait intéressant de construire ou renforcer des capacités de transport d'électricité entre différentes régions, ou l'impact de différent mix énergétiques possibles. Ces études s'adressent avant tout aux décideurs politiques et n'ont pas pour but de décider d'investissements réels et précis sur le réseau. On s'intéresse dans cette thèse à la modélisation et à la résolution de problèmes d'optimisation mathématique utilisés dans les études prospectives de RTE appelés *problèmes d'expansion de réseau stochastiques*.

On considère un graphe $G = (\mathcal{N}, \mathcal{A})$. Chaque noeud du graphe correspond à une région géographique, et chaque arc à une capacité d'échange en électricité entre deux régions. Le but du problème est de choisir des investissements dans des capacités de production sur les noeuds ou dans des capacités de transport sur les arcs pour satisfaire des demandes en électricité sur un horizon de temps à pas de temps discrets $\mathcal{T}$ donné. Plusieurs paramètres peuvent être aléatoires, tels que la demande en électricité en un noeud du graphe, les prévisions météorologiques, la disponibilité des moyens de production ou de transport ou encore les coûts de production d'électricité. L'aléa est modélisé par un ensemble fini de scénarios $\mathcal{S}$. Pour chaque scénario, un problème opérationnel modélise un planning opérationnel (quantité produite à chaque noeud et chaque pas de temps et routage de l'énergie sur le réseau) dont le but est de satisfaire la demande

en électricité à coût minimum. L'objectif total du problème d'expansion de réseau stochastique est de minimiser la somme des coûts d'investissement et des coûts opérationnels, en moyenne sur l'ensemble des scénarios d'aléa. On considère des problèmes d'optimisation stochastiques à deux étapes. Les variables d'investissement et les variables opérationnelles sont traitées comme des variables continues, et toutes les contraintes sont linéaires. Ceci est rendu possible grâce à l'approximation du courant continu notamment (Roldán et al., 2018; Zhang and Conejo, 2018; Wu et al., 2018).

L'algorithme de Benders par paquets

Les problèmes d'expansion de réseau stochastique résolus chez RTE sont de très grande taille. Dans le but de maintenir des temps de résolution raisonnables, les instances résolues ne prennent en compte qu'un nombre très limité de scénarios d'aléas. Pour palier cette limite, on présente un algorithme basé sur la décomposition de Benders dans le but de résoudre efficacement des problèmes linéaires stochastiques à deux étapes modélisés avec un grand nombre de scénarios d'aléas. Ces problèmes peuvent être modélisés comme suit:

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

où $x \in \mathbb{R}^{n_1}$, $c \in \mathbb{R}^{n_1}$, $\mathcal{S}$ est un ensemble fini de scénarios, $p_s \in \mathbb{R}^+$ est un poids positif associé au scénario $s \in \mathcal{S}$ (e.g., une probabilité), $g_s \in \mathbb{R}^{n_2}$, $W_s \in \mathbb{R}^{m\times n_2}$, $T_s \in \mathbb{R}^{m\times n_1}$, $d_s \in \mathbb{R}^m$, et $\mathcal{X} \subset \mathbb{R}^{n_1}$ est un ensemble polyédral. Les variables x sont appelées *variables de première étape* et les variables y_s sont appelées *variables de seconde étape* ou *variables de recours*. Le problème (1) est appelé *formulation extensive* d'un problème d'optimisation stochastique à deux étapes.

L'algorithme proposé dans cette section repose sur la reformulation de Benders du problème (1) (Benders, 1962; Van Slyke and Wets, 1969), en particulier la reformulation multicut de Benders (Birge and Louveaux, 1988). Cette reformulation est basée sur l'observation suivante : si on fixe les variables x à une valeur donnée $\bar{x} \in \mathcal{X}$, le problème devient séparable selon les scénarios. De plus, la fonction de valeur du problème résultant associé à chaque scénario est polyédrale (convexe, continue et linéaire par morceaux). En remplaçant la fonction objectif par cette fonction polyédrale dans la formulation précédente, on obtient la reformulation de Benders du problème, qui contient un nombre exponentiel de contraintes par rapport à la taille du problème. On appelle ces contraintes les *coupes de Benders*.

L'idée des algorithmes classiques de résolution est le suivant. On résout le problème avec un sous-ensemble de coupes de Benders. Comme le problème ainsi formulé est une relaxation du problème initial, il fournit une borne inférieure sur la valeur optimale du problème (1). On choisit une solution de première étape $x \in \mathcal{X}$, et on évalue les problèmes associés à chaque scénario pour

cette valeur de x donnée. La résolution de ces problèmes nous permet de générer des coupes de Benders qui ne sont pas présentes dans la formulation actuelle. Plus on ajoute des coupes de Benders dans la formulation, plus la borne inférieure fournie est proche de la valeur optimale du problème. De plus, cela nous permet d'évaluer une solution réalisable pour le problème initial, et donc de fournir une borne supérieure sur la valeur du problème. Lorsque l'écart entre la borne inférieure et la borne supérieure devient inférieur à un certaint seuil d'optimalité, l'algorithme s'arrête. De nombreux algorithmes de la littérature recquièrent de résoudre l'ensemble des problèmes associés à tous les scénarios à chaque itération de l'algorithme (Song and Luedtke, 2015; van Ackooij et al., 2017; Trukhanov et al., 2010; Linderoth and Wright, 2003). Lorsque le nombre de scénarios d'aléas est grand, cela peut être très couteux en temps de calcul. Certains algorithmes proposés dans la littérature permettent de ne pas résoudre systématiquement tous les sous-problèmes. Cependant, ces algorithmes ne convergent pas nécessairement en un nombre fini d'itérations (Higle and Sen, 1991), ou recquièrent que les fonctions objectifs des sous-problèmes ne dépendent pas de l'aléa (Crainic et al., 2021; Wets, 1983; Dantzig and Infanger, 1991).

Nous proposons dans cette thèse un algorithme basé sur la décomposition de Benders dans lequel seulement un petit nombre de sous-problèmes sont résolus à chaque itération appelé l'*algorithme de Benders par paquets*. Cet algorithme ne requiert pas que les fonctions objectifs des sous-problèmes soient déterministes, est exact et converge en un nombre fini d'itérations. Nous introduisons un critère d'arrêt qui permet de déterminer, après la résolution d'un sous-ensemble de sous-problèmes, si la solution de première étape courante peut être prouvée optimale ou non à ce stade de la résolution. Si elle n'est pas optimale, nous pouvons arrêter la résolution des sous-problèmes et définir une nouvelle solution de première étape. De cette manière, l'algorithme proposé permet d'explorer un grand nombre de solutions de première étape en peu de temps, et ainsi de converger plus rapidement vers une solution optimale.

Nous proposons aussi une méthode pour stabiliser l'algorithme proposé. Un des problèmes majeurs associés à la convergence des algorithmes de Benders est le phénomène d'oscillation des variables de première étape. Les méthodes de stabilisation (Lemaréchal et al., 1995; Ben-Ameur and Neto, 2007) permettent de palier ce problème et ainsi d'accélérer de manière significative les algorithmes. Cependant, ces méthodes ne peuvent pas être appliquées directement si l'ensemble des sous-problèmes n'est pas résolu à chaque itération, comme c'est le cas dans l'algorithme de Benders par paquets. Nous présentons donc une méthode générique pour stabiliser notre algorithme. Sous certaines conditions sur la méthode de stabilisation, nous montrons que l'algorithme stabilisé reste exact et converge en un nombre fini d'itérations. Finalement, nous évaluons les performances de l'algorithme de Benders par paquets sur des instances de la littérature d'optimisation stochastique avec jusqu'a 20.000 scénarios d'aléas. Les expérimentations montrent que l'algorithme de Benders par paquet peut résoudre jusqu'à 23 fois moins de sous-problèmes qu'un algorithme équivalent qui résout tous les sous-problèmes à chaque itérations. Dans sa verison stabilisée, l'algorithme est jusqu'à 25 fois plus rapide que l'algorithme du *level bundle*, et jusqu'à 5 fois plus rapide que la décomposition de Benders avec stabilisation in-out et une méthode d'agrégation de coupes statique.

Une contrainte de fiabilité pour un problème d'expansion de réseau stochastique

On s'intéresse dans cette partie à la structure des solutions optimales du problème d'expansion de réseau stochastique. Dans les formulations utilisées chez RTE, il existe un coût associé à la quantité d'énergie non distribuée dans une solution. Ce coût, d'environ 20.000€/MWh, n'est généralement pas sufffisant pour que toutes les demandes en électricité soient entièrement satisfaites dans une solution optimale du problème. En effet, les coûts d'investissement étant très élevés, il est parfoit moins cher de payer ce coût et de ne pas satisfaire une demande plutôt que d'investir dans de nouveaux moyens de production ou de transport. Les solutions optimales des problèmes d'expansion de réseau stochastique présentent souvent une grande quantité de demandes non satisfaites.

La législation française impose à RTE qu'en moyenne le nombre d'heures de défaillance soit inférieur à 3h (Article D141-12-6 du code de l'Energie). De fait, les solutions optimales du problèmes d'expansion de réseau ne permettent pas de satisfaire cette contrainte légale et ne sont pas considérées comme valides par les économistes de RTE. Actuellement, ils modifient les solutions à la main après la phase d'optimisation pour obtenir des solutions qui satisfont ce critère de *3h de défaillance*.

RTE ne dispose pas de modèle mathématique prenant en compte cette contrainte pour le problème d'expansion de réseau stochastique. Nous présentons dans cette section un tel modèle, avec une contrainte qui limite, en espérance sur l'ensemble des scénarios, le nombre de noeuds et de pas de temps auxquels une demande n'est pas entièrement satisfaite dans une solution optimale. Nous montrons que ce problème peut être naturellement modélisé par un problème bi-niveaux. Un problème bi-niveaux est un problème dans lequel certaines contraintes doivent être satisfaites par les solution optimales d'un autre problème d'optimisation. Dans le cas qui nous intéresse, la structure bi-niveaux vient du fait que la contrainte sur la défaillance doit être satisfaite par les solutions optimales des problèmes opérationnels de satisfaction de la demande. On présente deux formulations équivalentes pour le problème. La première met en évidence le caractère stochastique à deux étapes de la formulation, tandis que la seconde permet de présenter le problème comme un problème bi-niveaux linéaire en variables mixtes. On montre finalement que la contrainte bi-niveaux est nécessaire pour maintenir la cohérence du modèle.

Le problème ainsi obtenu est de très grande taille, et présente un très grand nombre de variables binaires. Les algorithmes classiques de résolution des problèmes bi-niveaux ne permettent pas de résoudre des instances aussi grandes (Fortuny-Amat and McCarl, 1981; Siddiqui and Gabriel, 2013). Nous présentons une méthode heuristique pour trouver des solutions réalisables à ce problème. Cette méthode est basée sur une décomposition de Benders, et une dichotomie sur les coûts d'investissement. On génère ainsi une suite de problèmes d'optimisation à un niveau dont les solutions convergent en un nombre fini d'itérations vers une solution réalisable du problème à deux niveaux.

Nous montrons que l'heuristique proposée permet de trouver des solutions réalisables sur des

instances possédant plusieurs millions de variables et de contraintes en moins de six heures. La méthode SOS-1 pour résoudre la reformulation KKT du problème ne trouve aucune solution réalisable en six heures. Ces résultats, réalisés sur des instances générées aléatoirement, sont encourageant. Des travaux futurs, permettant d'appliquer cette heuristique sur des instances de RTE, permettront de valider la méthode.

Acknowledgments

First of all, I would like to sincerely thank my thesis supervisors, François Clautiaux and Aurélien Froger, for having accompanied and supported me during these last years. I would also like to thank Boris Detienne. All three of them have taught me a lot and I am very grateful to them. I would also like to thank Manuel Ruiz for his trust and his precious advices.

I also would like to thank Luce Brotcorne and Pierre Fouilhoux to have accepted to be my rapporteurs and Céline Gicquel and Andrés Ramos to be part of the jury.

I would also like to thank all the members of the REALOPT team in Bordeaux to have warmly welcomed me. I am grateful to Mathilde Françon and the Antares team at RTE for including me despite this difficult period.

I finally want to thank my friends for the unforgettable vacations we spent together, which were sometimes necessary, and to Clara for her unfailing support. Some special thoughts go to my family, and particularly to my grand-parents who will not be able to see this work finished.

Contents

Chapter 1

Introduction

Following the Paris agreements voted during the COP21, the French government plans to get out of fossil fuels electricity production and achieve carbon neutrality by 2050. This process will involve deep changes in the electricity production and the structure of the power system (RTE, 2021). In this context, studies evaluating different investment strategies on the electricity production mix and on the transmission system are crucial. We study in this thesis optimization problems arising from such investment studies on power systems.

1.1 Industrial context

RTE is the French transmission system operator, and is in charge of the management of the high-voltage transmission system in France. One of the major roles of RTE is to guarantee, at all times, that the electricity demand is satisfied in France. Everyday, the company forecasts the next day's electricity consumption, and provides an electricity production schedule. The company is committed to the French government by a public service contract. RTE is also part of the ENTSO-E (European Network of Transmission System Operators for Electricity), which represents 39 European transmission system operators from 36 different countries. RTE is in charge of the electricity exchanges with the neighbor countries in order to maintain an equilibrium in the interconnected European electricity network. We present for example in Figure 1.1 some data representing the exchanges between France and its neighbors on the 1^{st} of January, 2022 and how electricity was produced in France at this time.

Because of its major role in the equilibrium between the electricity demand and production in France, RTE is asked by French legislation to produce every year a prospective report on the evolution of the transmission system to horizons from 5 to 30 years. These reports aim at measuring the risk of imbalance between electricity production and demand, and analyze investment strategies on the network in order to ensure the security of supply according to different technical, economical, environmental and political situations called *systemic scenarios*. These systemic scenarios represent evolutions that cannot be associated to probabilities. For each presented systemic scenario, an independent study is performed. The notion of systemic scenario in this industrial context should not be confused with the notion of scenario in stochastic programming that we will present later, where a scenario represents a given sample of a random

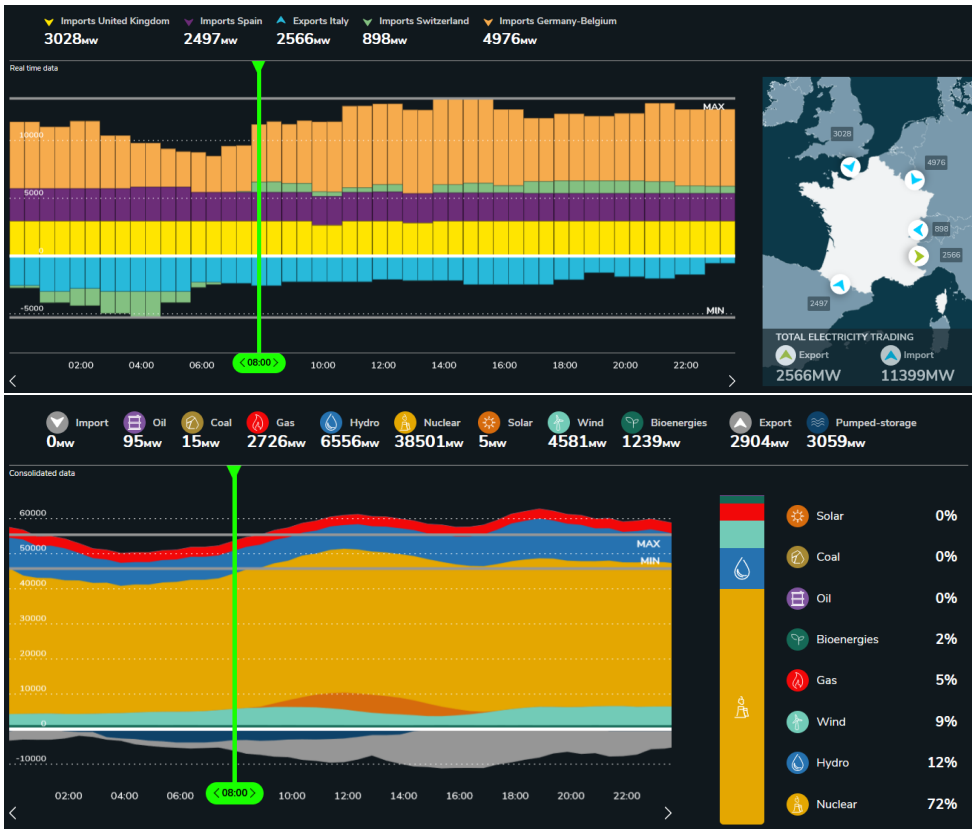


Figure 1.1: Sharing of the exchanges of electricity between France and its neighbors and sharing of the different sources of electricity production in France on January 1^{st}, 2022 at 8a.m.. Data taken from RTE Eco2Mix (https://www.rte-france.com/en/eco2mix) on September 12^{th}, 2022.

variable, associated to a probability distribution. We present in Figure 1.2 some of the systemic scenarios associated to energy consumption in France at horizon 2050 presented in (RTE, 2021), a prospective study with a 30 years horizon planning.

In prospective studies we are interested in, the transmission network is modeled in a very coarse way. France is generally modeled with around 10 nodes and neighbor countries are modeled with one or a few nodes. We show such a coarse representation in Figure 1.3. At each node, there is an aggregated production capacity for each source of energy (solar production, wind turbines, nuclear power plants, ...). Those studies aim at understanding global and structural trends, and to understand if it could be interesting to build or reinforce transmission capacities between large region or the possible electricity production mix according to different systemic scenarios. They are intended to inform decision-makers and not to model actual investment on the network. Other studies are performed at RTE in order to decide to build or reinforce specific transmission lines, in which there is a fine network representation, with only a fewer variables (many variables have been filtered with the results of the prospective studies). However, such studies are out of the scope of the present work.

Since RTE is a public service company entrusted by the government, the company must ensure the balance between electricity production and demand at all times. The French legislation allows a maximum of three hours of power system outage per year (Article D141-12-6 from the

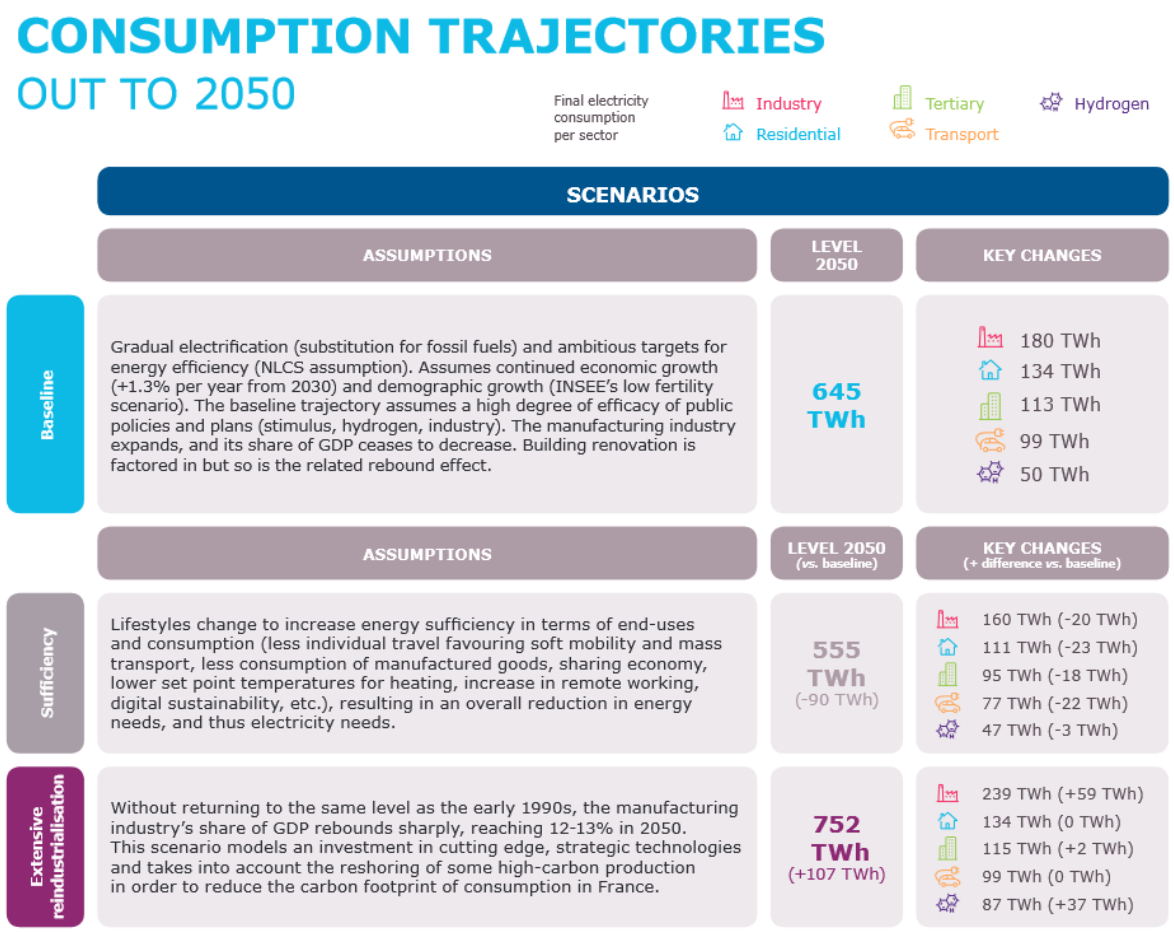


Figure 1.2: Some of the hypotheses leading to different prospective studies given in (RTE, 2021)

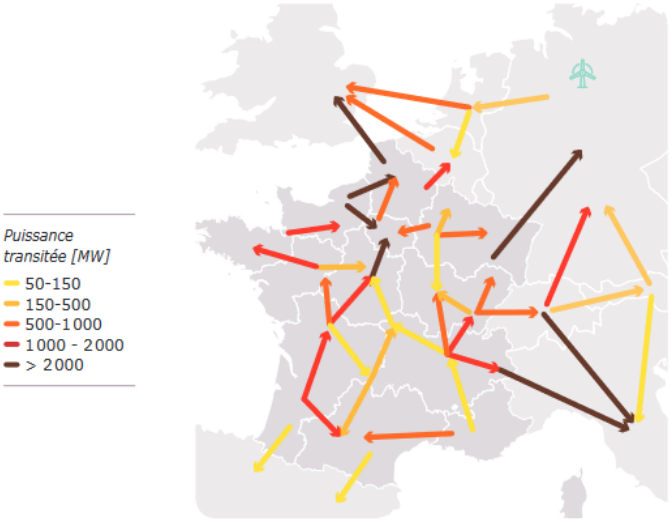


Figure 1.3: A representation of the French transmission network with only a few nodes, one for each region, and one to three node to represent other countries. Taken from (RTE, 2019).

French Energy Code). This legal criterion, called the *"3 hours criterion"*, ensures the reliability of the power system. In prospective studies conducted at RTE, this criterion have to be satisfied by the system with the proposed investment solutions.

1.2 Modeling hypotheses

The prospective studies presented in the previous section involve the resolution of mathematical programming problems the stochastic *expansion planning problems*. Considering a network, these problems aim at finding investments on transmission lines or production capacities in order to satisfy the electricity demand over a given time horizon modeled by a finite set of times steps $\mathcal{T}$. Many parameters are stochastic, such as the electricity demand, weather forecasts, operational costs, production and transmission lines availability or total capacity. We model the uncertainty with a finite set of scenarios. Each scenario represents one possible realization of the random variables (e.g., a given demand at each node and each time step in the given time horizon). For every scenario, we model an operational problem. This problem consists of finding a minimum cost operational planning (electricity productions and flows on the graph) to satisfy the electricity demands at every time steps, according to operational constraints. The objective of the stochastic epxansion planning problem is to minimize the sum of the investment costs and the expected operational costs over the scenarios (Alvarez Lopez et al., 2007; van der Weijde and Hobbs, 2012). We model it as a two-stage stochastic program.

In the general case, such two-stage expansion planning problems involve integer variables in order to model investment solutions, such as large power plants for which the invested capacity is either 0 or a large value, but nothing in between. We will consider hereafter only continuous investment variables. This hypothesis is reasonable in a first approximation of the problem as we consider a large aggregated system. Some investments can also be treated as continuous variables as they are made of large numbers of small units, such as solar panels or wind turbines. Moreover, investment capacities with very large integer steps, such as the number of nuclear power plants, are generally taken as hypotheses in the studies and are not decision variables. The variables modeling operations of the network are also treated as continuous variables and power flows are treated with the direct current hypothesis (DC hypothesis). The DC hypothesis is used in many studies such as in (Roldán et al., 2018; Zhang and Conejo, 2018; Wu et al., 2018), as it allows to model the problem with linear programs. It is a reasonable hypothesis to model such strategic decisions at long term horizon planning. We therefore consider large-scale two-stage stochastic optimization problems in this thesis.

The model also allows that demands may not be satisfied at every node and every time step. If all the demands in every scenario had to be satisfied, there would be some investment capacities which were almost never used in optimal solutions, leading to very expensive and conservative solutions. A compromise have to be found between a guaranteed supply of electricity in all scenarios and reasonable investment costs. This is modeled with a price associated to an unsatisfied demand: the higher this price, the lower the probability that a demand is not satisfied in an optimal solution. In RTE studies, this price is generally set to 20000€/MWh, the estimated cost for the collectivity of one hour of power system outage.

At RTE, more precise simulations are performed in order to validate investment solutions proposed in prospective studies or to evaluate short-term investment decisions on the transmission

network. In such studied, investments decisions are integer and the models take into account the alternative current power flow (AC power flow). AC power flow models take into account voltage angles and Kirchhoff's laws with non-linear equations. The resulting problems are numerically and theoretically challenging non-convex problems. Such studies are out of the scope of this work. For further information about the different models used in transmission and generation expansion planning problems, the reader is referred to (Lumbreras and Ramos, 2016; Micheli and Vespucci, 2021) and the references therein.

1.3 Objectives and thesis organization

This thesis aims at satisfying different objectives from a mathematical programming point of view, given the modeling hypotheses presented in Section 1.2.

Firstly, randomness is modeled with random variables with discrete distribution and finite support. Each possible realization of the random variable is called scenario in stochastic programming. In the prospective studies performed at RTE, there should be a large number of scenarios in the models in order to take precisely into account the large number of random parameters that impact the value of the problem. However, this would lead to very large-scale stochastic programs which are numerically challenging to solve. Currently, problems with a limited number of scenarios are solved in order to maintain reasonable computing times. The first objective of this work is to develop an efficient algorithmic procedure to solve linear stochastic programs with a large number of scenarios.

Secondly, a reliability criterion is used at RTE to validate or reject an investment solution given by the optimization model. Reliability is a largely studied question in transmission expansion planning literature (Choi et al., 2005; Li et al., 2021). However, there is in general no clear formulation of what is a reliable system (Lei et al., 2018). Reliability criteria are often penalized in the objective function in order to find a compromise between investment cost and reliability. The constraint used at RTE can be expressed as a limit on the expected number of nodes and time steps at which the demand is not fully satisfied in an optimal solution. However, this constraint is currently not taken into account in the models run at RTE. After the optimization step, the reliability criterion is checked. If it is not satisfied, the solution is modified by hand, by adding production capacities at the nodes where demands are not satisfied, in order to obtain a solution that satisfies this reliability criterion. The second objective of this work is to add this reliability constraint in optimization programs used at RTE and to develop an algorithmic procedure to solve the resulting problem. This thesis is organized as follows.

We introduce in Chapter 2 the classical mathematical concepts used in the following chapters. We describe basic concepts of linear programming such as the decomposition theorem of polyhedrons and its link to the polyhedral structure of parametric problems, which arise in stochastic programming. We then introduce the concept of two-stage stochastic programming and the Benders decomposition algorithm, a classical algorithmic procedure to solve such programs.

We present in Chapter 3 a literature review on mathematical programming methods to solve large-scale stochastic linear programs. We show classical acceleration methods used with Benders decomposition algorithms such as primal and dual stabilization methods. We also present methods focusing on problems with a large number of scenarios. Then, we present classical methods to solve mixed-integer linear bilevel programs, as such problems arise in models presented in Chapter 5.

In Chapter 4, we present an efficient algorithm to solve two-stage stochastic linear programs with a large number of scenarios, based on their multicut Benders reformulation. This algorithm allows to solve only a few subproblems at most iterations of the Benders decomposition algorithm. The core idea is to detect early in the subproblems solution process if a first-stage solution cannot be proven optimal. As soon as the algorithm detects that the current first-stage solution cannot be proven optimal, it stops solving the subproblems. We show that the algorithm is exact and converges in a finite number of iterations, and, contrary to many proposed algorithms in the literature, it does not require additional hypotheses on the subproblems, such as fixed recourse or a deterministic cost function in the subproblems. We also propose a generic framework to stabilize the proposed algorithm and show its numerical efficiency compared to classical algorithms of the literature on some generic instances taken from the stochastic programming literature. We show acceleration rates up to 25 times faster than Benders decomposition with level bundle stabilization or 5 times faster than Benders decomposition with in-out stabilization and static cut aggregation, on large-scale instances with up to 20.000 subproblems.

Chapter 5 presents an extension of stochastic expansion planning problems, taking into account a reliability constraint used in RTE prospective studies. This constraint aims at limiting the expected number of nodes and time steps at which there is an unsatisfied demand in a solution. We show that taking into account such a constraint in expectation in a two-stage framework leads to a bilevel formulation. We also present that a single-level linear formulation is not a two-stage stochastic program anymore, and may lead to invalid solutions. We present an example in which, in an optimal solution, some recourse variables have a cost as large as we want compared to the optimal cost of the recourse function of the associated scenario, which is inconsistent. We finally obtain a very large-scale mixed-integer bilevel linear program, which can be very challenging to solve even on very small instances. We then propose to use a heuristic method to find feasible solutions in a reasonable time. The heuristic is based on a binary search on the investment cost and a Benders decomposition algorithm to solve, at each iteration of the binary search, a modified single-level stochastic expansion planning problem. We show the existence of a bilevel feasible solution under some reasonable assumptions. We finally present on randomly generated instances that our solution method allows to produce feasible solutions when a classical bilevel algorithm fails to find any feasible solution in 6 hours.

We finally conclude in Chapter 6 and present some perspectives to the current work that focus on improving the efficiency of the proposed methods or on solving more challenging problems, such as problems with integer investment variables.

Chapter 2

Preliminaries

We present in this chapter some mathematical notions used in this thesis. We first recall some general linear programming results. Then, we present classical formulations of two-stage stochastic linear programs and the Benders decomposition algorithm, a classical algorithm to solve such problems. We refer the reader to (Schrijver, 1998) or (Conforti et al., 2014) for the proofs of the theorems.

2.1 Linear programming - Basic notions

A linear optimization program (P) is a program of the following form

$$(P): \begin{cases} \min\limits_{x} c^\top x \\ s.t.\ Ax \geq b \\ \quad\quad x \in \mathbb{R}^n_+ \end{cases}$$

where $n \in \mathbb{N}^*$ is the number of variables, $m \in \mathbb{N}$ the number of constraints, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. To every linear program (P), we can associate a dual program (D), which is defined as:

$$(D): \begin{cases} \max\limits_{\lambda} b^\top \lambda \\ s.t.\ A^\top \lambda \leq c \\ \quad\quad \lambda \in \mathbb{R}^m_+ \end{cases}$$

Both sets $\Pi_P = \{x \in \mathbb{R}^n_+ :\ Ax \geq b\}$ and $\Pi_D = \{\lambda \in \mathbb{R}^m_+ :\ A^\top \lambda \leq c\}$ are polyhedrons and we will refer to those respectively as the primal polyhedron for Π_P and the dual polyhedron for Π_D in the following. In the following, we say that a linear optimization problem (P) is infeasible if $\Pi_P = \varnothing$. We say that a linear optimization problem (P) is unbounded if there exists a sequence $(x_n)_{n\in\mathbb{N}}$ of solutions of Π_P such that $\lim\limits_{n\to+\infty} c^\top x_n = -\infty$ if (P) is a minimization problem or if $\lim\limits_{n\to+\infty} c^\top x_n = +\infty$ if (P) is a maximization problem.

Theorem 1. ***Weak duality theorem*** *Let (P) a linear optimization program and (D) its dual. If Π_P and Π_D are both non empty, then, for every $x \in \Pi_P$ and every $\lambda \in \Pi_D$, we have $b^\top \lambda \leq c^\top x$.*

This can be easily shown by noticing that if $Ax \geq b$ and $\lambda \geq 0$ then $\lambda^\top Ax \geq \lambda^\top b$, and if $\lambda^\top A \leq c^\top$ and $x \geq 0$ then $\lambda^\top Ax \leq c^\top x$.

Corollary 1. *If a linear program (P) is unbounded, then its dual is infeasible.*

Corollary 2. *If the primal polyhedron Π_P of a linear program (P) is non-empty and its dual is infeasible, then program (P) is unbounded.*

An important extension to the weak duality theorem is known as the *strong duality theorem*. It can be formulated as follows.

Theorem 2. ***Strong duality theorem*** *Let (P) be a linear optimization program and (D) its dual. Let Π_P and Π_D be their primal and dual polyhedrons. If Π_P and Π_D are both non empty, if x^* is an optimal solution to (P) and λ^* an optimal solution to (D), then:*

$$b^\top \lambda^* = c^\top x^*$$

This theorem allows to reformulate a linear optimization program with its dual when the latter have an exploitable structure.

Decomposition theorem of polyhedrons

We first present some definitions in order to formally write the decomposition theorem of polyhedrons.

Definition 1. *Let E be a vector space of dimension $n \in \mathbb{N}^*$. Let A and B be two sets included in E. Set C is called the Minkowski sum of A and B, denoted by $C = A + B$ if:*

$$C = \{a + b : a \in A, b \in B\}$$

Definition 2. *Let $n \in \mathbb{N}^*$ and $\ell \in \mathbb{N}$. A bounded polyhedron Q is a finitely generated convex set for which there exists a family $(u_i)_{1 \leq i \leq \ell}$ of vectors of $\mathbb{R}^n$ such that:*

$$Q = \{x \in \mathbb{R}^n : \ x = \sum_{i=1}^{\ell} \alpha_i u_i, \ \sum_{i=1}^{\ell} \alpha_i = 1, \ \alpha_i \geq 0, \ \forall i \in [\![1, \ell]\!]\}$$

Definition 3. *Let $n \in \mathbb{N}^*$ and $q \in \mathbb{N}$. A polyhedral cone C is a finitely generated convex set for which there exists a family $(v_j)_{1 \leq j \leq q}$ of vectors of $\mathbb{R}^n$ such that:*

$$C = \{x \in \mathbb{R}^n : \ x = \sum_{j=1}^{q} \beta_j v_j, \ \beta_j \geq 0, \ \forall j \in [\![1, q]\!]\}$$

Theorem 3. ***Decomposition theorem of polyhedrons*** *Let $n \in \mathbb{N}^*$ and $\Pi \subset \mathbb{R}^n$ a polyhedron. There exists a bounded polyhedron Q a polyhedral cone C such that Π is the Minkowski sum of Q and C, $\Pi = Q + C$.*

This theorem states that, for every polyhedron $\Pi \subset \mathbb{R}^n$, there exists two families of vectors

$(u_i)_{1\leq i\leq \ell}$ and $(v_j)_{1\leq j\leq q}$ such that:

$$\Pi = \left\{x \in \mathbb{R}^n : x = \sum_{i=1}^{\ell} \alpha_i u_i + \sum_{j=1}^{q} \beta_j v_j,\ \sum_{i=1}^{\ell} \alpha_i = 1,\ \alpha_i \geq 0,\ \forall i \in [\![1,\ \ell]\!],\ \beta_j \geq 0,\ \forall j \in [\![1, q]\!]\right\}$$

We show a geometrical representation of this theorem in Figure 2.1. We can easily notice that such a representation is not unique, for example by adding in the family of generators of the cone any convex combination of the currently present generators. However, there exists a unique minimal (in the sense of the cardinality) family of generators for the bounded polyhedron and the polyhedral cone. We will refer to those minimal families as, respectively the **extreme points** of the polyhedron for the generators of the bounded polyhedron, and as the **extreme rays** of the polyhedron for the generators of the polyhedral cone. This theorem allows to formally write some property of an optimization problem defined on a polyhedron Π. For example, a linear optimization program has a finite optimal value if $\Pi \neq \varnothing$ and its optimization direction c satisfies $c^\top v \leq 0$ for every extreme ray v of Π.

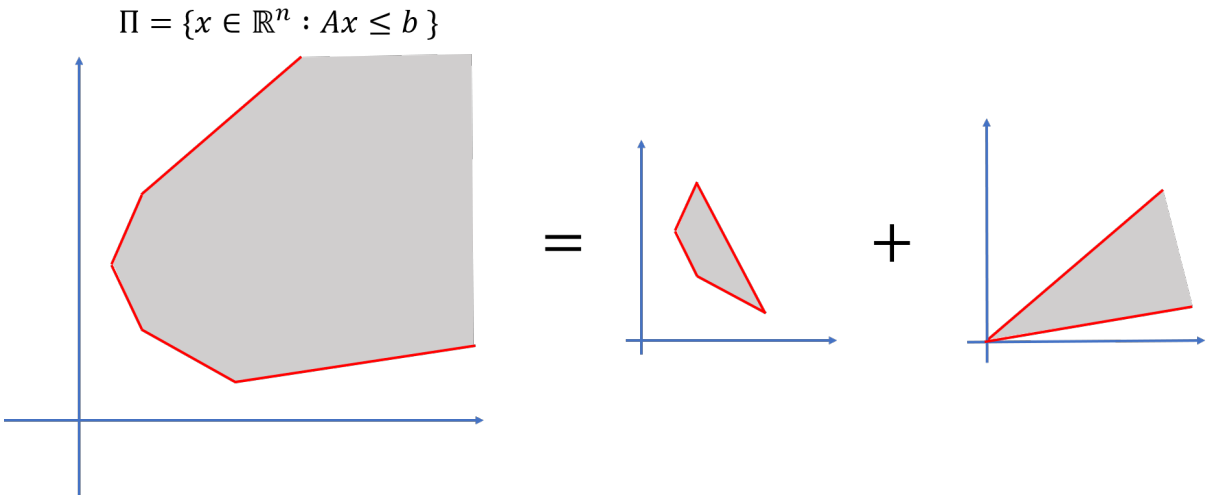


Figure 2.1: Geometrical representation of the decomposition theorem of polyhedrons

Polyhedral structure of parametric optimization programs

In this work, we are interested in parametric programs of the following form, for any $x \in \mathbb{R}^n$:

$$\phi(x) = \begin{cases} \min\limits_{y} g^\top y \\ s.t.\ Wy \geq d - Tx \\ \quad y \in \mathbb{R}^{n_2}_+ \end{cases} \tag{2.1}$$

where $\phi(x) \in \mathbb{R} \cup \{-\infty, +\infty\}$ denotes the optimal value of the problem, $g \in \mathbb{R}^{n_2}$, $d \in \mathbb{R}^m$, $W \in \mathbb{R}^{m\times n_2}$ and $T \in \mathbb{R}^{m\times n}$. We will refer to this program as $(P(x))$. Such problems arise naturally in stochastic programming. In this problem, the value of x modifies the constraints polyhedron, and thus, the optimal value. A way to study such problems is to analyze its dual polyhedron $\Pi_D = \{\lambda \in \mathbb{R}^m_+ :\ W^\top \lambda \leq g\}$. We denote by $\text{Vert}(\Pi_D)$ the set of extreme points of Π_D and by $\text{Rays}(\Pi_D)$ the set of extreme rays of Π_D. As we saw with the decomposition theorem

of polyhedrons, if there exists an extreme ray $v \in \text{Rays}(\Pi_D)$ such that $(d - Tx)^\top v > 0$, then the dual of $(P(x))$ is unbounded, and $(P(x))$ is infeasible.

Property 1. *Let Π_D define the dual polyhedron of a parametric problem (2.1), and $\text{Rays}(\Pi_D)$ the set of its extreme rays. We have:*

$$dom(\phi(\cdot)) = \{x \in \mathbb{R}^{n_1} : \ (d - Tx)^\top v \leq 0, \ v \in \text{Rays}(\Pi_D)\}$$

An other property arises when, for a given $x_0 \in \mathbb{R}^n$, the dual is feasible and bounded. Then, there exists an extreme point u of Π_D which is an optimal solution to the dual of $(P(x_0))$, and the optimal value of $(P(x_0))$ is $\phi(x_0) = (d - Tx_0)^\top u$, by the strong duality theorem. Moreover, the dual polyhedron $\Pi_D = \{\lambda \in \mathbb{R}^m_+ : \ W^\top \lambda \leq g\}$ does not depend on the value of x_0. Then, for any $x \in \mathbb{R}^n$, we know, as the dual is a maximization problem and as u_i is dual feasible, that $\phi(x) \geq (d - Tx)^\top u$. Finally, we have:

$$\begin{aligned} &\phi(x_0) = (d - Tx_0)^\top u \\ &\phi(x) \geq (d - Tx)^\top u, \ \forall x \in \mathbb{R}^n \end{aligned}$$

which means that $u^\top T$ is a subgradient of ϕ at x_0. This shows that the extreme points of the dual polyhedron define the subgradients of mapping ϕ.

The two previous results show that the extreme rays of the dual polyhedron allow to define the domain of ϕ, and that the extreme points allow to define the subgradients. As the number of extreme points is finite, the union set of the subgradients of ϕ on its domain is a finite set, which means that ϕ is a polyhedral application (continuous, convex and piecewise linear). An other way to see it is that, for any extreme point u of the dual polyhedron Π_D, there exists a cone, such that, for any optimization direction $\bar{d}$ belonging to this cone, u is an optimal solution to the problem of maximizing $\bar{d}^\top \lambda$ over Π_D. Let $x_0 \in \mathbb{R}^n$ be in the interior of $dom(\phi)$, and u an optimal extreme solution of the dual of $(P(x_0))$. Then, there exists a ball of radius $r > 0$ such that x_0 belongs to this ball, and for every x in this ball, u is also an optimal solution to the dual of $(P(x))$. Then, $\phi(x) = (d - Tx)^\top u$, for every x in this ball, and ϕ is linear on this set. As this is true for every feasible solution x_0, ϕ is polyhedral on its domain.

2.2 Two-stage stochastic programming

We now present the principle of two-stage stochastic programming, which is a core concept of the models we are interested in. Stochastic programs are optimization programs in which some parameters are not fully known. They are modeled with some random variables. Two-stage stochastic programs are special cases of stochastic programs in which optimization decisions are taken in two steps. First, some decisions are to be taken before knowing the realization of the random variables. Those decisions are called *first-stage variables*. Then, the realization of the random variables is revealed, and other decisions can be made, taking into account the realization of the random variables. Those decisions are called *second-stage variables*, or *recourse*

variables. We will focus in this thesis on two-stage stochastic linear programs, where both the objective function and constraints are linear functions. A two-stage stochastic linear program can be modeled as follows:

$$\begin{cases} \min\limits_{x} \; c^\top x + \mathbb{E}_{\boldsymbol{\omega}}[\phi(x, \boldsymbol{\omega})] \\ s.t. \; x \in \mathcal{X} \end{cases} \tag{2.2}$$

where $\mathcal{X} \subset \mathbb{R}^{n_1}$ is a polyhedral set, $x \in \mathbb{R}^{n_1}$ are the first-stage variables and $\boldsymbol{\omega}$ is a random variable. For any realization ω_s of $\boldsymbol{\omega}$, the value of $\phi(x, \omega_s)$ is computed as the value of the following parametric optimization program:

$$\phi(x, \omega_s) = \begin{cases} \min\limits_{y} \; g_{\omega_s}^\top y \\ s.t. \; W_{\omega_s} y \geq d_{\omega_s} - T_{\omega_s} x \\ \qquad y \in \mathbb{R}_+^{n_2} \end{cases} \tag{2.3}$$

with $g_{\omega_s} \in \mathbb{R}^{n_2}$, $W_{\omega_s} \in \mathbb{R}^{m \times n_2}$, $T_{\omega_s} \in \mathbb{R}^{m \times n_1}$ and $d_{\omega_s} \in \mathbb{R}^m$.

Depending on the probability distribution of the random variable $\boldsymbol{\omega}$, when it is known, it can be hard to compute the value of problem (2.2). Often, stochasticity is tackled by the introduction of scenarios, a finite sample of realizations of $\boldsymbol{\omega}$. The aim of a scenario representation is to approximate the random variable $\boldsymbol{\omega}$ by an other random variable $\boldsymbol{s}$ with a discrete and finite distribution. We denote by $\mathcal{S}$ the set of scenarios and by p_s the probability associated to the scenario $s \in \mathcal{S}$. When such a representation is used, there exists a reformulation of the two-stage stochastic program (2.2) as the following linear program:

$$\begin{cases} \min\limits_{x, (y_s)_{s \in \mathcal{S}}} \; c^\top x + \sum\limits_{s \in \mathcal{S}} p_s g_s^\top y_s \\ \quad s.t. : W_s y_s \geq d_s - T_s x, \; \forall s \in \mathcal{S} \\ \qquad\quad y_s \in \mathbb{R}_+^{n_2}, \; \forall s \in \mathcal{S} \\ \qquad\quad x \in \mathcal{X} \end{cases} \tag{2.4}$$

This reformulation is called *deterministic reformulation*, or *extensive formulation* of the stochastic program. By Monte-Carlo sampling scenarios from the initial probability distribution, and solving iteratively the resulting deterministic reformulations, one can solve problem (2.2) by *Sample Average Approximation*, see e.g. (Linderoth et al., 2006). Moreover, the single-stage structure of the deterministic reformulation allows the use of classical optimization techniques to solve the problem, such as the simplex algorithm when there are no integer variables, or decomposition techniques when the problem becomes larger. In the following, we only consider such discrete and finite probability distributions.

Benders decomposition

A successful decomposition method to solve such programs is the Benders decomposition (Benders, 1962), also called L-shaped method (Van Slyke and Wets, 1969) in the stochastic case. It is based on a reformulation of program (2.4). This reformulation can be obtained in three steps:

- A projection of the problem on the space of the first-stage variables

- A dualization of the projected problem
- A discretization of the resulting problem by enumeration of the extreme solutions of a polyhedron

The idea behind Benders decomposition is that, if we fix some variables, the resulting problem is easy to solve (decomposability, max flow on a network, ...). In our case, when we fix the first-stage variables x, the resulting problem becomes decomposable according to the scenarios. The resulting problem associated to a scenario $s \in \mathcal{S}$, called *subproblem associated to scenario s* and denoted by $(SP(x,s))$, is the following:

$$\begin{cases} \min\limits_{y} g_s^\top y \\ s.t.\ W_s y = d_s - T_s x \\ \qquad y \in \mathbb{R}_+^{n_2} \end{cases} \tag{2.5}$$

We denote, when it does exist, its value by $\phi(x,s)$. As we saw before, for every scenario $s \in \mathcal{S}$, the domain of $\phi(\cdot,s)$ can be expressed according to the extreme rays of the polyhedron of the dual of the subproblem $\Pi_D(s) = \{\lambda \in \mathbb{R}_+^m :\ W_s^\top \lambda \leq g_s\}$. We denote by $\text{Rays}(\Pi_D(s))$ the set of the extreme rays of $\Pi_D(s)$. We have:

$$dom(\phi(\cdot,s)) = \{x \in \mathbb{R}^{n_1} :\ (d_s - T_s x)^\top v \leq 0,\ v \in \text{Rays}(\Pi_D)\}$$

We can now write problem (2.4) with the value function of the subproblems $\phi(\cdot,s)$, by writing formally its domain:

$$\begin{cases} \min\limits_{x} c^\top x + \sum\limits_{s\in\mathcal{S}} p_s \phi(x,s) \\ s.t. : (d_s - T_s x)^\top v_s \leq 0,\ \forall s \in \mathcal{S},\ \forall v_s \in \text{Rays}(\Pi_D(s)) \\ \qquad x \in \mathcal{X} \end{cases} \tag{2.6}$$

Then, by dualizing the subproblems, we can express the value functions $\phi(\cdot,s)$ with the extreme points of the dual polyhedron $\Pi_D(s)$. Let us denote by ϕ the expected value mapping, such that, $\forall x \in \mathbb{R}^{n_1}$, $\phi(x) = \sum_{s\in\mathcal{S}} p_s \phi(x,s)$, and by Π_D the dual polyhedron associated with the problem generated by concatenation of all the subproblems. We have $\Pi_D = \Pi_D(1) \times \Pi_D(2) \times ... \times \Pi_D(\mathcal{S})$, and $\forall u \in \Pi_D$, $\exists\ (u_1, ..., u_{|\mathcal{S}|})$ such that $u = (u_1, ..., u_{|\mathcal{S}|})$. We also define $\text{Vert}(\Pi_D)$ the set of the

extreme points of polyhedron Π_D. We have:

$$\begin{aligned}
\phi(x) &= \sum_{s\in\mathcal{S}} p_s \phi(x,s) \\
&= \begin{cases} \min\limits_{(y_s)_{s\in\mathcal{S}}} \sum\limits_{s\in\mathcal{S}} p_s (g_s^\top y_s) \\ \quad s.t. : W_s y_s \geq d_s - T_s x, \ \forall s \in \mathcal{S} \\ \qquad y_s \in \mathbb{R}^{n_2}_+, \ \forall s \in \mathcal{S} \end{cases} \\
&= \begin{cases} \max\limits_{(\pi_s)_{s\in\mathcal{S}}} \sum\limits_{s\in\mathcal{S}} p_s (d_s - T_s x)^\top \pi_s \\ \quad s.t. : W_s^\top \pi_s \leq g_s, \ \forall s \in \mathcal{S} \\ \qquad \pi_s \in \mathbb{R}^{m_2}_+, \ \forall s \in \mathcal{S} \end{cases} \\
&= \max_{(u_1,\dots,u_{|\mathcal{S}|})\in \mathrm{Vert}(\Pi_D)} \Big\{ \sum_{s\in\mathcal{S}} p_s (d_s - T_s x)^\top u_s \Big\}
\end{aligned}$$

Then, by replacing ϕ with the last equation in formulation (2.6), and by introducing an epigraph variable θ, we obtain the *Benders reformulation* of problem (2.4):

$$\begin{cases} \min\limits_{x,\theta} \ c^\top x + \theta \\ s.t. : \theta \geq \sum\limits_{s\in\mathcal{S}} p_s (d_s - T_s x)^\top u_s, \ \forall (u_1,\dots,u_{|\mathcal{S}|}) \in \mathrm{Vert}(\Pi_D) \quad (i) \\ \qquad (d_s - T_s x)^\top v_s \leq 0, \ \forall s \in \mathcal{S}, \ \forall v_s \in \mathrm{Rays}(\Pi_D(s)) \quad (ii) \\ \qquad x \in \mathcal{X}, \theta \in \mathbb{R} \end{cases} \tag{2.7}$$

Constraints (i) are called *optimality cuts*, and constraints (ii), *feasibility cuts*. We obtain an equivalent formulation of problem (2.4) with an exponential number of constraints. This formulation is not directly usable, as a total enumeration of the extreme points and extreme rays of the polyhedrons of the duals of the subproblems would be way too expensive. However, this allows the use of constraints generation methods, such as Kelley's cutting planes (Kelley, 1960), or more advanced techniques, to efficiently solve the problem.

We we call hereafter **a Benders decomposition algorithm** an algorithm designed to solve the Benders reformulation of a problem. The idea behind Benders decomposition algorithms is first to relax both the optimality and feasibility cuts. The resulting program is called the **relaxed master program**. Then, the algorithms consist of defining iteratively a first-stage solution, called hereafter **separation point**, and to solve the subproblems at this first-stage solution in order to discover new dual extremal solutions and to add new Benders cuts in the relaxed master program until convergence is proven. We present in Algorithm 2.1 a classical algorithm based on Kelley's cutting-planes algorithm to solve Benders reformulation of problem (2.4). At line 4, the algorithm solves the relaxed master program and retrieves its optimal solution. The relaxed master program is used as an oracle to get the separation point. As it is a relaxation of the original problem, it also defines a lower bound on the optimal value of the problem, computed at line 5. From lines 6 to 9, the subproblems are solved and their dual extremal solutions are retrieved. At line 9, the algorithm add feasibility cuts if some subproblems are infeasible. If all

the subproblems are feasible, an optimality cut is added at line 11, and the algorithm computes the value of the objective function in the original problem at the separation point at line 12, which defines an uppper bound on the optimal value of the problem. Finally, at line 2, the optimality gap between the lower bound and the best upper bound is checked and the algorithms stops when it reaches a given optimality gap.

Algorithm 2.1: Kelley's classical version of the monocut Benders decomposition algorithm

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

1 **Initialization:** $k \leftarrow 0$, $UB^{(0)} \leftarrow +\infty$, $LB^{(0)} \leftarrow -\infty$

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

3 $\quad k \leftarrow k + 1$

4 $\quad$ Solve $(RMP)^{(k)}$ and retrieve $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in\mathcal{S}})$

5 $\quad LB^{(k)} \leftarrow c^\top \check{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \check{\theta}_s^{(k)}$

6 $\quad$ **for** $s \in \mathcal{S}$ **do**

7 $\quad\quad$ Solve $(SP(\check{x}^{(k)}, s))$ and retrieve $\pi_s \in \text{Vert}(\Pi_D(s))$ or $\pi_s \in \text{Rays}(\Pi_D(s))$

8 $\quad\quad$ **if** $(SP(\check{x}^{(k)}, s))$ *is infeasible* **then**

9 $\quad\quad\quad$ Add $0 \geq \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$

10 $\quad$ **if** $(SP(\check{x}^{(k)}, s))$ *is feasible* $\forall s \in \mathcal{S}$ **then**

11 $\quad\quad$ Add $\theta \geq \sum_{s\in\mathcal{S}} \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$

12 $\quad\quad UB^{(k)} \leftarrow min\Big(UB^{(k-1)}, c^\top \check{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \pi_s^\top (d_s - T_s \check{x}^{(k)})\Big)$

13 $\quad (RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$

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

Chapter 3

Literature review

We focus in this literature review on two fields of mathematical programming: two-stage linear stochastic programs and bilevel programs. As presented in Chapter 1, classical formulations of network expansion studies such as the ones conducted by RTE are based on two-stage stochastic programs. We present in Section 3.1 a review on methods to solve such two-stage stochastic linear programs, with a focus on problems formulated with a large number of scenarios. We then present in Section 3.2 a review on algorithms to solve bilevel programs with a linear lower level, as the reliability constraint presented in Chapter 5 relies on a bilevel formulation with linear lower levels.

3.1 Benders decomposition to solve two-stage stochastic programs

We focus in this section on the resolution of two-stage stochastic linear programs with Benders decomposition algorithms. We will focus here on problems with scenario-based uncertainty, although some researchers proposed methods to solve two-stage stochastic programs with general probability distributions, such as Ramirez-Pico and Moreno (2021) or Forcier and Leclère (2021). Then, we focus on problems with the following structure:

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s g_s^\top y_s \\ s.t. : W_s y_s \geq d_s - T_s x,\ \forall s \in \mathcal{S} \\ \qquad y_s \in \mathbb{R}_+^{n_2},\ \forall s \in \mathcal{S} \\ \qquad x \in \mathcal{X} \end{cases} \tag{3.1}$$

Those acceleration techniques can be partitioned in four types. First, we present methods working on the formulation, such as the so-called *mono-cut* and *multi-cut formulations* and their extensions. Then we present the primal stabilization techniques. Those techniques aim at improving the quality of the separation point in the algorithm. Next, we present the dual stabilization methods to improve the quality of the cuts. Finally, we show some methods with a focus on large-scale stochastic programs.

3.1.1 Monocut and multicut reformulations

We presented in Section 2.2 the Benders reformulation of a linear optimization problem as stated in the seminal paper (Benders, 1962). This reformulation is also called *monocut reformulation*, or *single cut reformulation*, as we approximate the recourse function with only one cut at each iteration. The reformulation is based on one unique epigraph reformulation for the whole recourse function. In the stochastic case, the recourse function is separable over the scenarios, and is then the sum of one polyhedral function for each scenario. Birge and Louveaux (1988) proposed a reformulation in which there is one independent epigraph variable for each scenario. This reformulation, that we present in problem (3.2), is called *the multicut Benders reformulation* of problem (3.1).

$$\begin{cases} \min\limits_{x,(\theta_s)_{s\in\mathcal{S}}} c^\top x + \sum\limits_{s\in\mathcal{S}} p_s\theta_s \\ \quad s.t. : \theta_s \geq (d_s - T_s x)^\top u_s, \ \forall s \in \mathcal{S}, \ \forall u_s \in \text{Vert}(\Pi_D(s)) \quad (i) \\ \qquad (d_s - T_s x)^\top v_s \leq 0, \ \forall s \in \mathcal{S}, \ \forall v_s \in \text{Rays}(\Pi_D(s)) \quad (ii) \\ \qquad x \in \mathcal{X}, \ \theta_s \in \mathbb{R}, \ \forall s \in \mathcal{S} \end{cases} \tag{3.2}$$

The cutting planes algorithm based on the multicut reformulation usually requires less iterations to converge. However, the large number of cuts added to the master program in a multicut framework at each iteration might become a bottleneck of the algorithm. As card($\mathcal{S}$) constraints are added to the master program at each iteration, its size grows quickly, and the time to solve the relaxed master program at each iteration can be too long to produce a competitive algorithm. You and Grossmann (2013) showed the numerical efficiency of the multicut reformulation on a supply chain planning problem. However, the choice between a monocut or a multicut reformulation may be in general problem dependent.

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. They show the numerical efficiency of the proposed methods, as local cut aggregations allows to reduce the number of iterations compared to pure monocut algorithm, while the size of the master program remains reasonable. Wolf et al. (2014) proposed a different way to use jointly the monocut and multicut formulations. They use a monocut master problem, but maintain also a multicut model during the solution process. For any given first-stage solution, they also evaluate the multicut model at the first-stage solution to the relaxed monocut master program, and check if it gives a larger value than the monocut one. If this occurs, they propose to generate a cut by aggregating the actives cuts from the multicut model, and add it in the monocut master program. In this way, they generate sub-optimal cuts improving the current relaxation of the master program, without the need to solve any subproblem. As evaluating the multicut model at a given first-stage solution involves only to find the maximum of the cuts for each scenario, this can be done in a linear time in the number of cuts and avoids the costly resolution of a multicut master program.

Finally, the multicut formulation allows the development of asynchronous parallel methods, such as in (Linderoth and Wright, 2003). They propose to solve the master problem when a given percentage of the subproblems have been solved, from 50% to 85% in their numerical study,

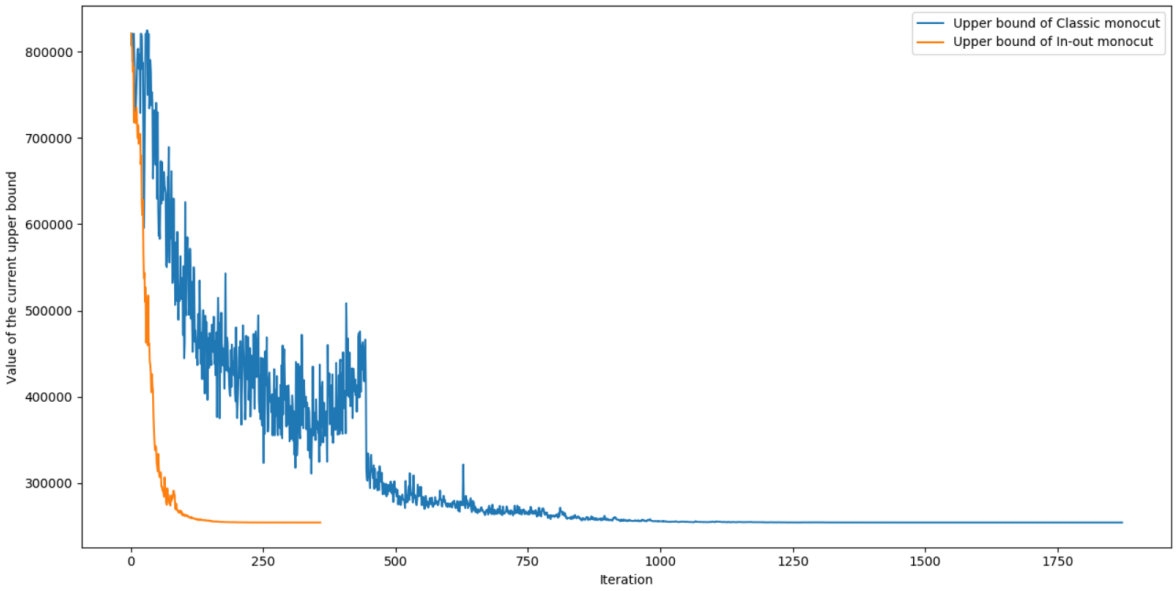


Figure 3.1: Evolution of the upper bound over the iterations of the monocut Benders decomposition algorithm on a stochastic problem with and without primal stabilization (in-out stabilization).

so that the core containing the master program does not wait too long that all the subproblems have been solved.

3.1.2 Primal stabilization

A classical drawback to the Benders decomposition algorithm, and more generally to every cutting plane method (Kelley, 1960), is the so-called *bang-bang effect* of first-stage variables (Vanderbeck, 2005; Pessoa et al., 2013) leading both to poor quality cuts, and a slow and erratic decrease in the upper bound (see e.g. Fig 3.1). Many successful methods have been proposed to alleviate this effect which we refer to as primal stabilization techniques as they aim at finding better sequences of primal solutions during the solution process. A large class of acceleration techniques are the *bundle methods* (Lemaréchal et al., 1995). We present here the bundle methods as primal stabilization methods. Those methods try to restrict the search of an optimal solution to points close to a given first-stage solution called *stability center*. This stability center is generally defined as the separation point with the smallest objective function value among those evaluated so far: $\hat{x}^{(k)} = \arg\min_{j \in [\![0,k-1]\!]} \{c^\top x^{(j)} + \sum_{s \in \mathcal{S}} p_s \phi(x^{(j)}, s)\}$. Bundle methods are often implemented as quadratic stabilization techniques. They can be seen, in the case of Benders decomposition, as stabilization methods based on a modified relaxed master program, in which the solution have to stay close to the stability center. Among the most used methods, one can cite the proximal bundle, the level bundle or the trust-region. Proximal bundle methods penalize the distance to the stability center in the objective function, as proposed also by (Ruszczyński, 1986), whereas trust-region methods restrict the search to an optimal solution in a ball around the stability center according to a given norm. In the level bundle method, the relaxed master program returns a separation point as the projection of the stability center on a level set of the objective function, according to a value $f_{lev} \in \mathbb{R}$. This value is computed as $f_{lev} = (1 - \lambda)UB(\hat{x}) + \lambda LB$, where λ is

the level parameter, $UB(\hat{x})$ the objective value of the stability center in the original problem, and LB a given lower bound on the objective function of the relaxed master program.The modified master program is the following:

$$\begin{cases} \min\limits_{x,\theta} \ \dfrac{1}{2}||x-\hat{x}||_2^2 \\ s.t.: \theta \geq \sum\limits_{s\in\mathcal{S}} p_s u_s^\top (d_s - T_s x), \ \forall s \in \mathcal{S}, \ \forall u_s \in \text{Vert}(\Pi_s) \\ \qquad c^\top x + \theta \leq f_{lev} \\ \qquad x \in \mathcal{X}, \ \theta \in \mathbb{R} \end{cases}$$

We can remark that, if the value f_{lev} is too low, this program can be infeasible. This shows that f_{lev}, which is greater than LB by construction, is a valid lower bound on the model, and LB can be updated to f_{lev}.

(Zverovich et al., 2012) presented a computational study of bundle methods. Their results show, on some instances of the literature, that the level bundle scales better than proximal or trust-region method when the number of scenarios increases. This justifies the choice made in Section 4.4 to compare the presented algorithm to the level bundle algorithm as a state-of-the-art primal stabilization method. Bundle methods have been successfully applied on some stochastic programs, as the level bundle in (van Ackooij et al., 2017), the proximal bundle in (Oliveira et al., 2011) or the trust-region, with an infinite norm in (Linderoth and Wright, 2003).

An other successful method to stabilize the algorithm is the so-called *in-out stabilization method* (Ben-Ameur and Neto, 2007). It has been initially developed for a general framework of cutting-planes methods, in order to generate cuts closer to a given domain $\mathcal{D}$. Let $x_{in} \in \mathcal{D}$ a known feasible solution, and x_{out} the solution to a relaxation of $\mathcal{D}$ at a given iteration of the algorithm. Let $\alpha \in [0,1)$ a given parameter. The in-out separation scheme consists in separating a solution $x_{sep} = \alpha x_{in} + (1-\alpha)x_{out}$ instead of separating solution x_{out} (see Fig 3.2). If $x_{sep} \in \mathcal{D}$, then x_{sep} has a lower cost in the objective function than x_{in}, and one improves the upper bound. Else, the cut generated to separate solution x_{sep} is deeper than the one computed at x_{out}. However, in the case of Benders decomposition, the domain $\mathcal{D}$ that we want to approximate is the epigraph of the recourse function. The cuts are always tangents to the recourse function. Moreover, in order to generate a cut, the subproblem requires only the information about the first-stage solution x, regardless of its current epigraph solution θ. We show in Fig 3.3 the difference between a textbook application of in-out stabilization, and how it is done in Benders decomposition in practice. The *in-point* is then comparable to the stability center of bundle methods. We denote $\hat{x}^{(k)}$ the in-point at iteration k of the algorithm to unify the notations with bundle methods. It is equal to the separation point (among those calculated so far) with the smallest objective function value: $\hat{x}^{(k)} = \arg\min_{j\in[\![0,k-1]\!]} \{c\top x^{(j)} + \sum_{s\in\mathcal{S}} p_s \phi(x^{(j)}, s)\}$. As bundle methods, the in-out stabilization aims at generating cuts close to the anchored solution *in-point* in order to alleviate the *bang-bang effect* of primal variables. The separation point $x^{(k)}$ is then defined on the segment between $\hat{x}^{(k)}$ (in point) and $\check{x}^{(k)}$ (out-point):

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

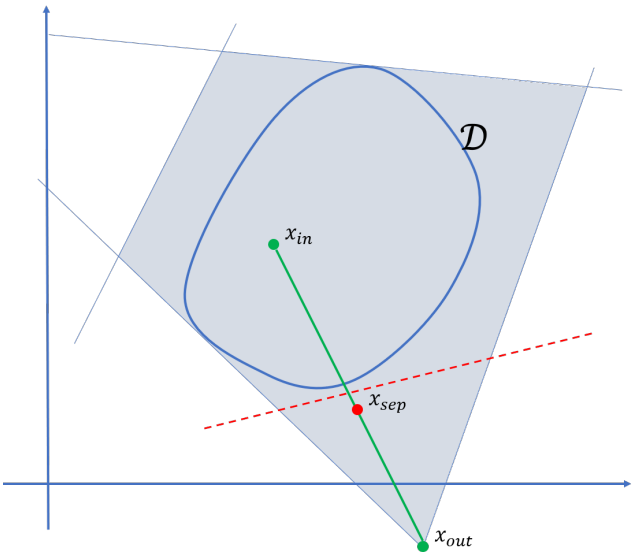


Figure 3.2: In-out scheme in a general framework, optimizing over an approximation of a given domain $\mathcal{D}$

The in-out approach creates a sequence of stability centers with decreasing objective values converging to an optimal solution to the problem. As it performs a linear search between the in-point and the solution to the relaxed master program, it can rely on the numerical efficiency of linear programming solvers. The in-out separation approach has been applied successfully in a cutting plane algorithm to solve a survivable network design problem (Ben-Ameur and Neto, 2007), in column generation (Pessoa et al., 2013), in a branch-and-cut algorithm based on a Benders decomposition approach to solve facility location problems (Fischetti et al., 2016), and in a cutting plane algorithm applied to disjunctive optimization (Fischetti and Salvagnin, 2010).

Primal stabilization methods for cutting plane based algorithms are essential to develop competitive algorithms (Vanderbeck, 2005). We show in Figure 3.4 how stabilization (with an in-out method) affects the evolution of the space in which the algorithm seeks an optimal solution, given the level set of the best first-stage solution evaluated so far.

3.1.3 Dual stabilization

Another family of acceleration techniques focuses on the quality of the optimality cuts. We refer to them as dual stabilization methods. The polyhedral structure of the recourse function implies a degeneracy of the dual subproblem. In the singular points of the recourse function, many equivalent extreme dual solutions exist for the subproblem, each one defining a different optimality cut. The subdifferential of the second-stage function in a singular point is a cone, and the extremal solutions of the dual subproblem define the generators of this cone. Among all of those cuts, Magnanti and Wong (1981) proposed a method to find what they call a *pareto-optimal cut*. We first define the notion of *dominance* between cuts. We say that a cut $\theta \geq g^1 x + \nu^1$ *dominates* or *is stronger than* $\theta \geq g^2 x + \nu^2$ if

1. $g^1 x + \nu^1 \geq g^2 x + \nu^2, \forall x \in \mathcal{X}$
2. $\exists\, \bar{x} \in \mathcal{X}, g^1 \bar{x} + \nu^1 > g^2 \bar{x} + \nu^2$

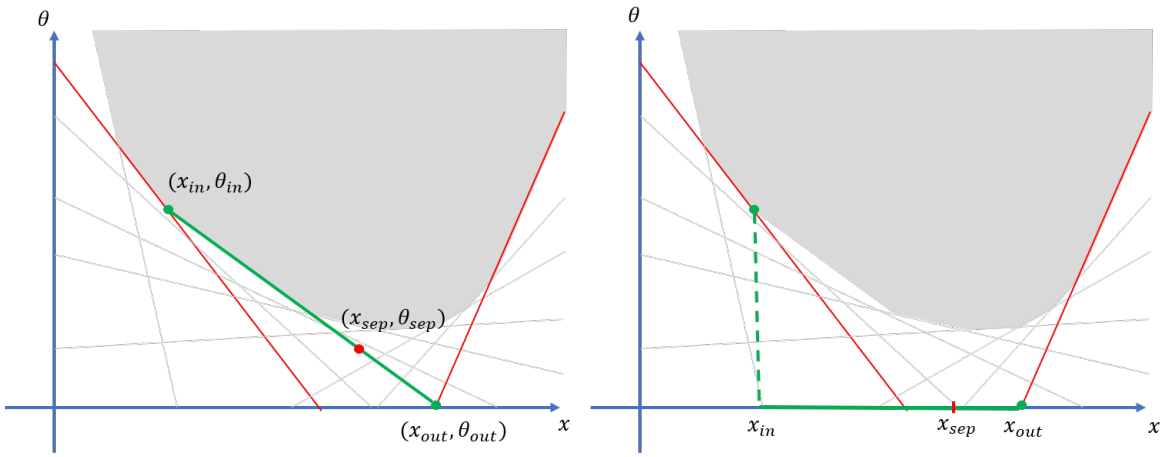


Figure 3.3: Classical in-out stabilization scheme (on the left) and the way it is done in Benders decomposition, as the separation problem does not depend on the epigraph variable θ. The grey set is the set we want to approximate. Red lines are the cuts generated so far, and the green segment represents the segment on which the new separation point is chosen.

Then, a cut is said to be pareto-optimal if no cut dominates it. Let $\mathring{x} \in int(Conv(X))$, the relative interior of the convex hull of $\mathcal{X}$, and $\check{x} \in \mathcal{X}$ the solution to the relaxed master program at a given iteration of the Benders decomposition algorithm. Let $U(\check{x}, s) = \text{Vert}(\text{argmax}_{\pi \in \mathbb{R}^m}\{(d_s - T_s\check{x})\pi : W_s^\top \pi \geq g_s\})$, the set of extreme points of $(DSP(\check{x}, s))$. The authors propose to find a pareto-optimal cut in a two-step framework. At each iteration, they first solve the subproblems to retrieve a dual optimal solution, π_0. Then, they solve an auxiliary problem :

$$\begin{cases} \max\limits_{\pi \in \mathbb{R}^m} \ (d_s - T_s\mathring{x})^\top \pi \\ \quad s.t. \ W_s^\top \pi \leq g_s \\ \qquad (d_s - T_s\check{x})^\top \pi \geq (d_s - T_s\check{x})^\top \pi_0 \\ \qquad \pi \in \mathbb{R}^m_+ \end{cases} \tag{3.3}$$

in order to find, among the optimal solutions of the dual subproblems, one which maximizes the cost at a solution $\mathring{x} \in \text{int}(\text{Conv}(\mathcal{X}))$. The importance of using a first-stage solution in the relative interior of $\mathcal{X}$ is illustrated in Figure 3.5. In this example, choosing a point $\mathring{x} < x_1$ would have led to generate the blue cut $\mathcal{C}_1$ which is strictly dominated by the cut $\mathcal{C}_2$ on the first-stage feasible subset $\mathcal{X}\backslash\{x_1\}$. We can however remark that this problem does not occur at first-stage solution x_2, as the subdifferential of the recourse function at this point is a singleton, or at every non-singular first-stage solution. Then, solving the classical Benders subproblem once at those points is already sufficient to get a pareto-optimal cut. Moreover, we see that if the solution to the relaxed master program $\check{x}$ is already in $\text{int}(\text{Conv}(\mathcal{X}))$, then all the possible cuts generated at $\check{x}$ are pareto-optimal. Then, it seems that the practical application of the Magnanti-Wong method is restricted to first-stage solutions on the edge of the feasible space. However, the methods allows to discriminate dual solutions in practice, even between different pareto-optimal cuts. With a wisely chosen core point, close to an optimal solution to the problem, this method allows to generate deeper cuts, accelerating the convergence of the cutting-plane algorithm.

Papadakos (2008) propose then to extend the algorithm of Magnanti and Wong (1981),

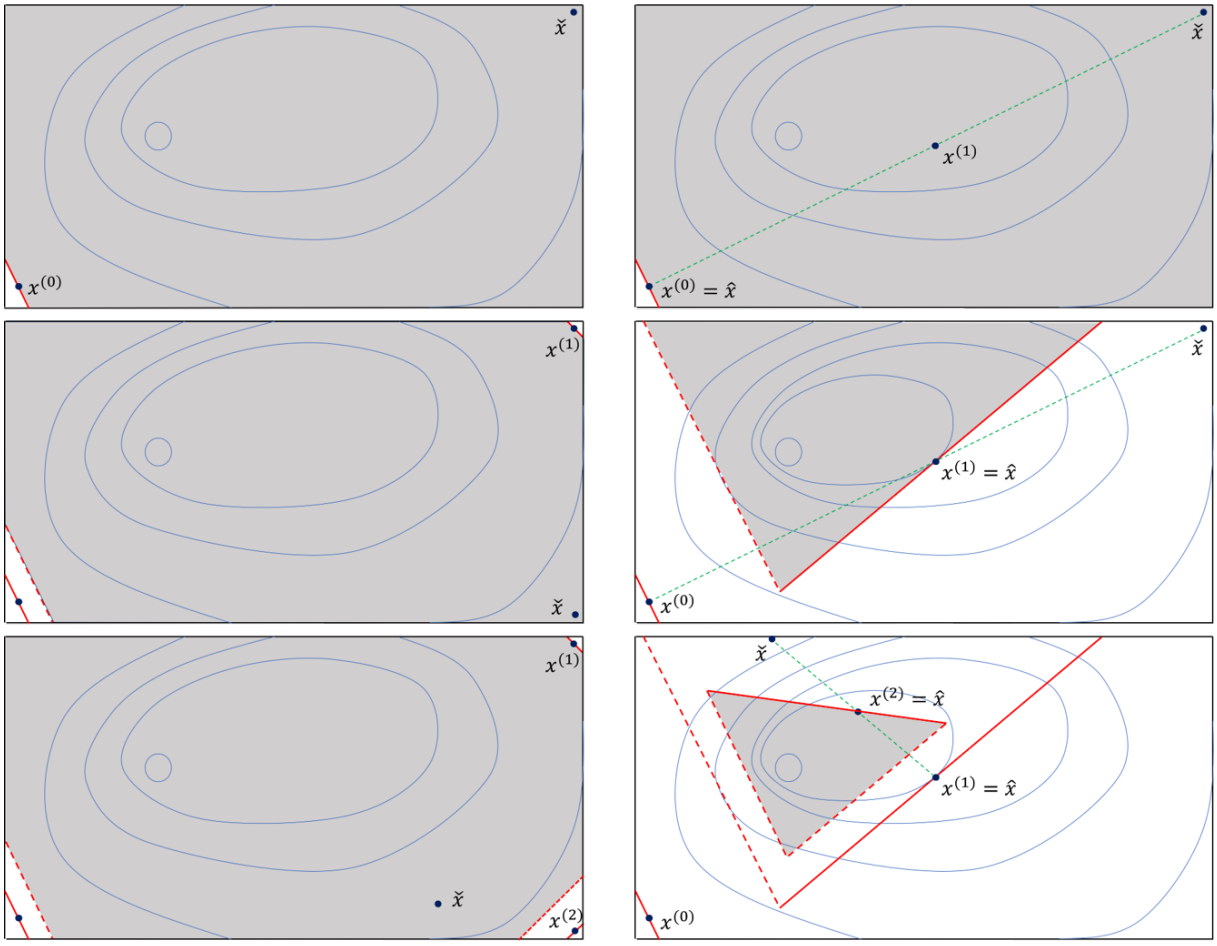


Figure 3.4: Two iteration of Benders decomposition, without stabilization (on the left) and with in-out stabilization (on the right). The level sets of the objective function are represented in blue. $\check{x}$ represents the solution to the relaxed master program at each iteration, and the straight lines in red represent the cuts, at the level of the separation point. The gray sets represent the level sets, in the relaxed master program, at the value of the best first-stage solution evaluated. At a given iteration of the algorithm, we only know that the optimal solution belongs to this set. We clearly see the impact of the primal stabilization, as the volume of this set decreases way faster in the stabilized method.

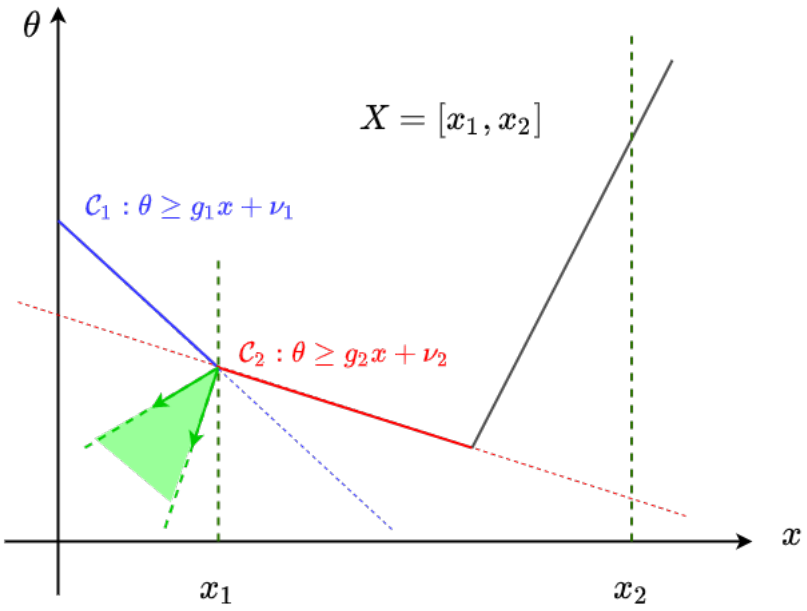


Figure 3.5: A polyhedral function where the first-stage feasible set $\mathcal{X} = [x_1, x_2]$, and x_1 is singular. However, as x_1 is not in the relative interior of $\mathcal{X}$, only the red cut $\mathcal{C}_2$ is pareto-optimal, as the blue one is dominated by the red on $\mathcal{X}$.

adding also cuts from first-stage solutions known to generate directly a pareto-optimal cut, and a practical procedure to generate such points. Finally, Sherali and Lunday (2013) interpret the Magnanti-Wong procedure as a multi-objective optimization problem and propose a unique subproblem leading to pareto-optimal cuts by adding a penalization term in the right-hand side of the primal subproblems. They show that, with a small enough perturbation value, they can bypass the two-phase resolution proposed by Magnanti and Wong (1981). In their computational experiments, they also propose to use perturbation values independent to the problem values, such as 10^{-6}, which could lead to sub-optimal global solutions, but already allows to accelerate significantly the global optimization procedure on the tested instances.

3.1.4 Deterministic second-stage cost function

Many works on Benders decomposition to solve stochastic two-stage programs focus on solving a specific type of stochastic programs which satisfy two hypotheses :

- **Fixed-recourse hypothesis**, which means that matrices W_s in subproblems (2.5) are the same for every scenario : $W_s = W, \ \forall s \in \mathcal{S}$
- **Deterministic second-stage cost function**, which means that vectors g_s in subproblems (2.5) are the same for every scenario : $g_s = g, \ \forall s \in \mathcal{S}$

Under those two conditions, the subproblems have an interesting property, they have the same dual polyhedron. This means that the polyhedron of the dual subproblem for a given scenario $s \in \mathcal{S}$, $\Pi_s = \{\pi \in \mathbb{R}^m | W_s^\top \pi \leq g_s\} = \{\pi \in \mathbb{R}^m | W^\top \pi \leq g\}$, does not depend on the scenario anymore. Those hypotheses allow the development of efficient specific methods.

Wets (1983) first propose the so-called *bunching* method. After the resolution of a given subproblem $s_0 \in \mathcal{S}$ at a given first-stage solution $x_0 \in \mathcal{X}$, they get, when it is feasible, a dual solution, say $\pi_0 \in \Pi$. As the feasible space of all the subproblems is the same, this solution is dual feasible for the subproblems associated with every scenario. The principle of bunching is to verify if this solution is primal feasible for the other scenarios. This solution π_0 is dual optimal for every scenario where primal feasibility is satisfied, and we can skip the resolution of the associated subproblems.

An other successful method based on those hypotheses is the *stochastic decomposition* proposed in (Higle and Sen, 1991). The idea is to compute cuts based on samples of the uncertainty, and samples of the dual vertices. As the sample of the uncertainty converges to its actual probability distribution by the law of large numbers, and the set of known dual vertices converges to the whole set of vertices, the cuts computed tend to be closer to actual Benders cuts and the algorithm converges to an optimal solution. Using notations of Section 2.2, we denote by $(u_i)_{1\leq i\leq \ell}$ the set of extreme points of the dual polyhedron Π. The idea of stochastic decomposition is to sample only one scenario s^k at each iteration k of the algorithm according to the probability distribution, and to store its dual optimal solution in a set, say $V^{(k)}$. At each iteration, a cut is computed as follow

$$\theta \geq cx + \frac{1}{k}\sum_{t=1}^{k} \pi_t^k (d_s^t - T_s^t x)$$

where $\pi_t^k \in Argmax(\pi(d_s^t - T_s^t x^{(k)}) | \pi \in V_k)$ with $x^{(k)}$ the separation point at iteration k. As the cuts are computed only on samples of scenarios, the cuts are not valid. They may cut some parts of the recourse function epigraph. To ensure the validity of the method, the coefficients of the cuts are multiplied at each iteration by $\frac{k-1}{k}$ at iteration k of the algorithm, so that they become inactive when the number of iteration increases. The idea of the convergence of the method is based on the population of set $V^{(k)}$ to contain all the desired dual optimal solutions, and on the law of large numbers, so that the expected cut computed converge to actual Benders cuts as the weight associated to each scenario will converge to its probability. Some enhancements have been proposed to stochastic decomposition, such as in (Higle and Sen, 1994) where the use of a quadratic stabilization allows to delete some cuts in the master program, or in (Gangammanavar et al., 2018) where the authors adapt the method to the case where the subproblem do not share the same dual polyhedron, by checking primal feasibility as in the bunching method before computing the cuts. It has also been used successfully in (Gangammanavar et al., 2016) to solve an economic energy dispatch problem in the presence of wind farms.

Crainic et al. (2021) propose to add in the master program existing or artificial subproblems, so that, in the early iterations, the master program would not be totally blind about subproblems information. They propose, among other propositions, to add an artificial subproblem in the master where the coefficients are computed as the expectation of all the random coefficients of the subproblems. The hypothesis of fixed recourse and deterministic cost function ensures that the epigraph formulation of the artificial scenario contains the epigraph formulation of the recourse function. Said differently, the value of the artificial scenario give, for every $x \in \mathcal{X}$, a lower bound on the value of the recourse function. They show the efficiency of the method on some stochastic network design problems. The method has also been successfully applied on some network design problems applied to logistic services (Belieres et al., 2022). Finally

Song and Luedtke (2015) propose the *adaptative partition-based method*. They show, under the hypotheses of fixed recourse and deterministic second-stage function, that there exists a partition of the subproblems such that the problem formulated with the expected subproblem over each elements of the partition leads to an optimal solution to the original problem. (van Ackooij et al., 2017) proposed to solve the resulting problem with Benders decomposition and a level bundle stabilization at each iteration to accelerate the solution process and show noticeable accelerations compared to classical level bundle algorithm. We show in Table 3.1 a comparison between methods of the literature designed to solve large-scale two-stage stochastic linear programs.

Paper	Randomness hypothesis*	Solve all SPs	Monocut or multicut	Exact method	Finite convergence	Cut aggregation	Stabilization
(Crainic et al., 2021)	$q_s = q, W_s = W \, \forall s \in \mathcal{S}$	Yes	Both	Yes	Yes	No	No
(Song and Luedtke, 2015)	$q_s = q, W_s = W \, \forall s \in \mathcal{S}$	Yes	Not applicable	Yes	Yes	No	No
(van Ackooij et al., 2017)	$q_s = q, W_s = W \, \forall s \in \mathcal{S}$	No	Both	Yes	Yes	No	Level
(Wets, 1983)	$q_s = q, W_s = W \, \forall s \in \mathcal{S}$	No	Both	Yes	Yes	No	No
(Dantzig and Infanger, 1991)	$q_s = q, W_s = W \, \forall s \in \mathcal{S}$	No	Monocut	No	Yes	No	No
(Higle and Sen, 1991)	$q_s = q, W_s = W \, \forall s \in \mathcal{S}$	No	Monocut	Yes	No	No	No
(Trukhanov et al., 2010)	No	Yes	Multicut	Yes	Yes	Yes	No
(Linderoth and Wright, 2003)	No	Yes	Multicut	Yes	Yes	No	Trust-region
(Wolf et al., 2014)	No	All or none	Monocut and Multicut	Yes	Yes	No	Level
(Oliveira et al., 2011)	$W_s = W \, \forall s \in \mathcal{S}$	No	Monocut	Inexact	Yes	No	Proximal bundle

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

Table 3.1: Comparison of stochastic methods to accelerate Benders decomposition. (SPs: subproblems)

3.2 Bilevel programming

As stated in Chapter 1, we also study in this work some extensions of stochastic expansion planning problems which can be modeled with bilevel programming. In particular, we focus in this section on methods to solve bilevel linear programs in which there might be integer variables only in the upper level. We first introduce some formulations and notations, and present classical applications of bilevel programming. Then, we present classical methods to solve linear bilevel programs, and finally some tailored methods focusing on the mixed-integer structure of the upper-level program.

3.2.1 General principle and applications

The notion of bilevel programming initially comes from economics, more specifically from Stackelberg games (von Stackelberg and Von, 1952). In Stackerberg games, a leader - for example a company - takes a decision, and an other company, called follower, reacts to this decision. They both want to maximize their utility. Bilevel programming is particularly adapted to situations in which different actors (a leader and followers, an attacker and a defender, ...) take decisions in a sequential way. Bilevel programming has been successfully applied in several fields. Early applications were motivated by security or military subjects such as in Bracken and McGill (1974) or traffic and transportation (LeBlanc and Boyce, 1986; Marcotte, 1986; Ben-Ayed et al., 1988). The question of bilevel optimization in network design is still an active research field (Fontaine and Minner, 2014; Basciftci and Van Hentenryck, 2020) with applications to air traffic scheduling for example (Liu et al., 2013). It has also been appllied to supply chain problem (Ryu

et al., 2004; Yue and You, 2017; Reisi et al., 2019) and portfolio optimization Leal et al. (2020); González-Díaz et al. (2021). Bilevel optimization have also been very sucessfully applied to pricing problems and toll optimization, in which the leader decides prices to set on some arcs, and followers minimizes their cost (Bialas and Karwan, 1984; Labbé et al., 1998). Some extensions have been proposed such as in (Brotcorne et al., 2008), in which the leader fixes the prices and the design of the graph. A survey on network pricing can be found in (Labbé and Violin, 2016).

Many applications to bilevel programming arise in the field of energy. Motto et al. (2005) analyze a security problem in power grids under disruptive threats. Generation and transmission expansion planning in markets environments have been studied in (Garces et al., 2009; Jin and Ryan, 2011; Baringo and Conejo, 2012). Grimm et al. (2019); Kleinert and Schmidt (2019) study the optimal price zones in electricity markets. A recent survey on bilevel programming in energy and electricity market can be found in (Wogrin et al., 2020). We refer the reader to (Kleinert et al., 2021) and the references therein for a recent survey on bilevel programming.

Bracken and McGill (1973) first formalized the notations of a bilevel program. A linear bilevel program is a program of the following form:

$$\begin{cases} min\ c^\top x + q^\top y \\ \quad s.t.\ Ax \geq b \\ \qquad Cx + Dy \geq f \\ \qquad y \in \arg\min \{g^\top y' : \quad Wy' + Tx \geq d, y' \in \mathbb{R}^{n_2}_+\} \\ \qquad x \in \mathbb{Z}^{n_{1,I}} \times \mathbb{R}^{n_{1,C}} \end{cases} \tag{3.4}$$

where $c \in \mathbb{R}^{n_1}$, $q \in \mathbb{R}^{n_2}$, $A \in \mathbb{R}^{m_1 \times n_1}$, $b \in \mathbb{R}^{m_1}$, $C \in \mathbb{R}^{p_1 \times n_1}$, $D \in \mathbb{R}^{p_1 \times n_2}$, $f \in \mathbb{R}^{p_1}$, $g \in \mathbb{R}^{n_2}$, $W \in \mathbb{R}^{m_2 \times n_2}$, $T \in \mathbb{R}^{m_2 \times n_1}$, $d \in \mathbb{R}^{m_2}$. Variables x are called upper-level variables. $n_1 = n_{1,I} + n_{1,C}$, with $n_{1,I} \in \mathbb{N}$ represents the number of integer upper-level variables and $n_{1,C} \in \mathbb{N}$ represents the number of continuous upper-level variables. Variables y are called lower-level variables as they are variables of the inner optimization problem. This problem is called lower-level program and defined as a parametric optimization program. Let $x \in \mathbb{Z}^{n_{1,I}} \times \mathbb{R}^{n_{1,C}}$, the lower-level program of the bilevel program (3.4) evaluated at x can be formulated as the following linear parametric program:

$$\phi(x) = \begin{cases} min\ g^\top y \\ \quad s.t.\ Wy' \geq d - Tx \\ \qquad y' \in \mathbb{R}^{n_2}_+ \end{cases} \tag{3.5}$$

where $\phi(x)$ denotes its optimal value. Constraints $Cx + Dy \geq f$ are called the coupling constraints. Those constraints couple, in the upper-level program, the upper-level variables with the lower-level variables, and have to be satisfied only by optimal solutions to the lower-level program. Finally, we call linking variables the upper-level variables which appear in the lower-level variables. In many situations, only a subset of the upper-level variables have non-zero coefficients in matrix T. The notation **MIP-LP bilevel program** means that upper-level program is a mixed-integer program and that the lower-level program is a linear program, following the nomenclature

presented in (Kleinert et al., 2021).

We also define the *value-function reformulation* of a bilevel program (Ye and Zhu, 1995; Mitsos et al., 2008) as:

$$\begin{cases} min\ c^\top x + q^\top y \\ s.t.\ Ax \geq b \\ \quad Cx + Dy \geq f \\ \quad g^\top y \leq \phi(x) \\ \quad Wy + Tx \geq d \\ \quad y \in \mathbb{R}_+^{n_2} \\ \quad x \in \mathbb{Z}^{n_{1,I}} \times \mathbb{R}^{n_{1,C}} \end{cases} \tag{3.6}$$

in which the optimality conditions of the lower-level variables is hidden in the cost constraint $g^\top y \leq \phi(x)$.

We finally define the so-called *high-point relaxation* of the bilevel program (3.4) (Bialas and Karwan, 1984) as the following single-level program:

$$\begin{cases} min\ c^\top x + q^\top y \\ s.t.\ Ax \geq b \\ \quad Cx + Dy \geq f \\ \quad Wy + Tx \geq d \\ \quad y \in \mathbb{R}_+^{n_2} \\ \quad x \in \mathbb{Z}^{n_{1,I}} \times \mathbb{R}^{n_{1,C}} \end{cases} \tag{3.7}$$

The high-point relaxation consists of the relaxation of the optimality condition on the lower-level program. This leads to a single-level program which is indeed a relaxation of the initial bilevel program. In the following, we will denote by $(HPR(P))$ the high-point relaxation of a bilevel program (P).

Bilevel programs are in general complex programs. One of the reason is that the feasible region of such programs is generally non-convex, and may be also defined as the union of several disconnected spaces because of some coupling constraints. We illustrate this with the example provided in the survey of Kleinert et al. (2021), in Section 3.1. The feasible region of the following

bilevel program

$$\begin{cases} min\ y \\ s.t.\ 0 \leq x \leq 1 \\ \quad y \leq 1.5 \\ \quad y \in \arg\min \{ -y' : \quad y' \leq 1 + x, \\ \qquad\qquad\qquad\quad y' \leq 3 - x, \\ \qquad\qquad\qquad\quad y' \in \mathbb{R}_+ \} \\ \quad x \in \mathbb{R} \end{cases} \tag{3.8}$$

is represented in Figure 3.6 and is indeed non-convex and defined as the union of two disconnected spaces.

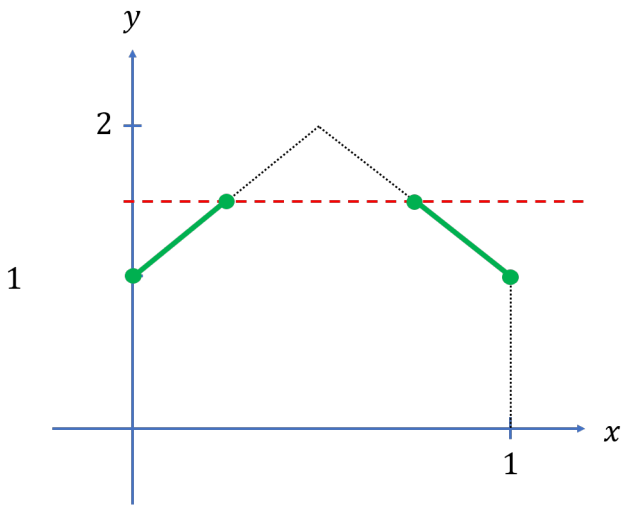


Figure 3.6: An example of the feasible region of a LP-LP bilevel program. The feasible region is the union of the two green lines, which defines a non-convex and disconnected feasible space. Figure taken from (Kleinert et al., 2021)

Because of this structure, designing generic methods to solve bilevel program is complex, and many methods from the literature require some additional assumptions on the problem, such as no coupling constraints, or that every linking variable being integer.

3.2.2 Solution techniques

We present in this subsection several general solution techniques to solve MIP-LP bilevel programs. One of the most successful reformulations used to solve bilevel programs with continuous convex lower-lever is the so-called **KKT-reformulation**. It is based on the Karush-Kuhn-Tucker conditions on a couple of primal-dual variables to define an optimal solution of a problem. By reformulating the optimality condition of the lower-level variables in problem (3.4) with the KKT

conditions, we obtain the following single-level non linear problem:

$$\begin{cases} min\ c^\top x + q^\top y \\ \quad s.t.\ Ax \geq b \\ \qquad Cx + Dy \geq f \\ \qquad Wy + Tx \geq d \\ \qquad W^\top \lambda \leq g \\ \qquad \lambda^\top (Wy + Tx - d) = 0 \\ \qquad y \in \mathbb{R}_+^{n_2}, \lambda \in \mathbb{R}_+^{m_2} \\ \qquad x \in \mathbb{Z}^{n_{1,I}} \times \mathbb{R}^{n_{1,C}} \end{cases} \tag{3.9}$$

Constraints $Wy + Tx \leq d$ are the primal feasibility constraints, constraints $W^\top \lambda \leq g$ define the dual feasibility, and $\lambda^\top (Wy + Tx - d) = 0$ are the complementary slackness constraints. Those are the only non linear constraints in formulation (3.9).

Fortuny-Amat and McCarl (1981) first proposed to linearize the complementary slackness constraints. Let $m_2 \in \mathbb{N}$ be the number of constraints in the lower-level problem. For every index $1 \leq i \leq m_2$, the i^{th} constraint of the lower-level problem is denoted as $W_{i.}y + T_{i.}x \geq d_i$. The complementary slackness constraints say that, for every constraint $W_{i.}y + T_{i.}x \geq d_i$, $1 \leq i \leq m_2$, either the constraint is satisfied to equality or the dual variable associated to this constraint is equal to zero. Then, each complementary slackness constraint is equivalent to the two following constraints:

$$\begin{cases} \lambda_i \leq M z_i \\ W_{i.}y + T_{i.}x \leq M(1 - z_i) \end{cases}$$

where z_i is a binary variable and M a large enough constant value. This allows to reformulate the bilevel program (3.4) as a mixed-integer single-level program, and to use for example Branch&Bound techniques to solve it. However, M have to define a valid upper bound on dual variables λ. In a general setting, this problem can be hard to solve. In fact, Kleinert et al. (2020) showed that finding a correct Big-M value is as hard as solving the bilevel program in general. Moreover, Pineda and Morales (2019) showed that the classical way to define Big-M values, consisting of setting a value, and increasing it if there exists a binding constraint $\lambda_i \leq M z_i$, may lead to invalid solutions. Then, from a theoretical point of view, this method should be used only if a valid upper bound on dual variables can be easily found from the formulation. Pineda et al. (2018) show in an extensive computational study that the method of Fortuny-Amat and McCarl (1981) is able to solve to optimality even some large-scale instances (with only continuous variables in both levels) when the value of Big-M is wisely chosen. However, choosing a too small value of M may lead to incorrect solutions, and a too large value may lead to numerical instabilities in the solvers and then to poor numerical results.

An other successful method to solve the KKT reformulation is the SOS-1 method proposed in (Siddiqui and Gabriel, 2013). By adding a new variable $u = Wy + Tx - d$, one can notice that the sets $\{\lambda_i, u_i\}$ define a SOS-1 for every $1 \leq i \leq m_2$. Then, this problem can be solved

with a branching procedure. Beginning by solving the problem without any complementary slackness constraint, one can define a new branching rule. Is a complementary slackness constraint is violated, then the algorithm creates two nodes, the first with the additional constraint $\lambda_i = 0$ and the second with the additional constraint $u_i = 0$. The branching procedure was initially proposed by Bard and Moore (1990) to solve LP-QP bilevel programs. The use of SOS-1 allow to embed this branching procedure in a more general branching scheme, and to solve MIP-LP bilevel programs directly with MIP solvers. The procedure is similar to the procedure proposed by Fortuny-Amat and McCarl (1981), and converges to an optimal solution to the bilevel program. Moreover, this allows to alleviate the Big-M problems mentioned earlier. In their computational study, Pineda et al. (2018) show that the SOS-1 method is able to solve to global optimality many instances (their small and middle sized instances and some large instances).

On the other hand, the non linear KKT reformulation of problem (3.4) also leads to solution techniques from non-linear programming to solve LP-LP bilevel programs. White and Anandalingam (1993) first proposed a penalty approach to solve linear programs with complementarity constraints, and Hu and Ralph (2004) further studied the convergence property of the method. It consists of a lagrangian relaxation of the complementarity constraints. This leads to a single-level program with a non-linear objective defined on a polytope. It can then be solved to local optimality with off-the-shelf non-linear solvers. Lv et al. (2007) apply the method to bilevel programs. Scholtes (2001) proposed a regularization method to solve linear programs with complementarity constraints. The method consists of modifying the right-hand side of the complementarity constraints by a positive value t, and to solve iteratively the resulting problem while making the value of t decrease to 0. Both the regularization and the penalty methods converge to good quality solutions in a very small amount of time (a few seconds compared to many hours for the mixed-integer techniques) in the computational study of Pineda et al. (2018).

Finally, from seminal papers to recent works, many papers have proposed tailored solution techniques to special cases of MIP-MIP bilevel programs. Those algorithms often require the problem to satisfy particular assumptions, such as the absence of coupling constraints, the integrity of all linking upper-level variables, or the compactness of the high-point relaxation feasible region.

Bard and Moore (1992) first proposed a Branch&Bound algorithm when there are only binary variables in both levels. They were able to solve problems with up to 45 variables in a few minutes, but reached their time limit of 900s with problems with 50 variables. Faísca et al. (2007) proposed to use the fact that the lower-level program is a parametric program and that its value function is piecewise linear. They assume that variables of both levels are continuous or binary and that the high-point relaxation is compact. Then, by enumeration of the linear parts of the value function of the lower-level and binary variables of the lower-level, they derive an exponential number of single-level programs which can be evaluated separately to find an optimal solution. They only test their algorithm on a toy example. Saharidis and Ierapetritou (2009) proposed a Benders-like method also under the hypothesis of compact high-point relaxation. They consider continuous and binary variables in the upper-level and continuous variables in the lower-level.

They also present their algorithm on a toy example. Zeng and An (2014) proposed a method based on strong duality reformulation and an enumeration of integer lower-level variables. They apply a column and constraint generation procedure and solve problems with up to 20 variables and 30 constraints. Lozano and Smith (2017) proposed a method based on the value-function reformulation to solve problems with only integer variables in the upper-level. They test their algorithm on the test set of Xu and Wang (2014) containing instances with up to 920 variables and 368 constraints.

More recently, several Branch&Cut methods have been developed. DeNegre and Ralphs (2009) proposed a Branch&Cut in the case of general integer problems in both levels. They assume that there are no coupling constraint, and propose to discard infeasible bilevel solutions by applying no-good cuts. Tahernejad et al. (2020) proposed an extension to the method when there are continuous variables in both levels. Xu and Wang (2014) proposed to apply a multi-way branching procedure when upper-level variables are all integer and bounded. They solve randomly generated instances with up to 460 variables and 184 constraints in both stage. Wang and Xu (2017) proposed the so-called Watermelon algorithm for problems with only integer variables. They also apply a multi-way branching to exclude polyhedrons containing only infeasible solutions. Finally Fischetti et al. (2018) proposed a Branch&Cut method using off the shelf MIP solvers. They use intersection cuts and tailored fathoming rules to solve problems with integer coupling variables. They accept coupling constraints and do not require the high-point relaxation to be compact. They also performed extended numerical experiments containing problems with up to 80000 variables and 5000 constraints. Their algorithms solved to optimality 104 out of 125 interdiction problems in one hour and 20 out of 57 of very challenging instances derived from the MIPLIB test set.

We summarize the different algorithms to solve MIP-MIP bilevel programs presented here in Table 3.2. We show some of the hypotheses of the algorithms such as the type of variables that they can handle, the possibility of having coupling constraint and the compactness requirement of the high-point relaxation. When the authors provide an experimental study based on an implementation of their algorithm, we also show the maximum number of constraints and variables in the problems that they were able to solve. It is clear that bilevel programs are very challenging programs, and it seems that the largest instances that we can solve have around a few thousands variables and constraints in the general case.

Paper	$x \in \mathbb{Z}$	$x \in \mathbb{R}$	$y \in \mathbb{Z}$	$y \in \mathbb{R}$	Coupling Constraint	HPR Compact	Max Variables	Max Constraints
Bard and Moore (1992)	✓(b)		✓(b)			✓	45	18
Faísca et al. (2007)	✓(b)	✓	✓(b)	✓	✓	✓	-	-
Saharidis and Ierapetritou (2009)	✓(b)	✓		✓	✓	✓	-	-
Zeng and An (2014)	✓	✓	✓	✓			20	30
Lozano and Smith (2017)	✓		✓	✓	✓		920	368
Xu and Wang (2014)	✓		✓	✓	✓	✓	920	368
DeNegre and Ralphs (2009)	✓		✓				20	40
Wang and Xu (2017)	✓		✓		✓		920	1288
Fischetti et al. (2018)	✓	✓*	✓	✓	✓		$80000^{(1)}$	$5000^{(1)}$
Tahernejad et al. (2020)	✓	✓*	✓	✓	✓		4961	4944

*But should not appear in the lower-level program

✓(b) stands for only binary variables

$^{(1)}$: The algorithms solves to optimality 20 out of the 57 largest instances, and reaches an average optimality gap of 26.12% of those 57 instances

Table 3.2: Comparison of algorithms to solve MIP-MIP bilevel programs

Chapter 4

The Benders by batch algorithm

Design and stabilization of an enhanced algorithm to solve multicut Benders reformulation of two-stage stochastic programs

4.1 Introduction

We propose in this chapter an algorithm to solve two-stage stochastic linear programs. We assume that the probability distribution is given by a finite set of scenarios and focus on problems with a large number of scenarios. We consider the following linear program with a scenario block structure:

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s g_s^\top y_s \\ s.t. : W_s y_s = d_s - T_s x,\ \forall s \in \mathcal{S} \\ \qquad y_s \in \mathbb{R}^{n_2}_+,\ \forall s \in \mathcal{S} \\ \qquad x \in \mathcal{X} \end{cases} \tag{4.1}$$

where $x \in \mathbb{R}^{n_1}$, $c \in \mathbb{R}^{n_1}$, $\mathcal{S}$ is a finite set of scenarios, $p_s \in \mathbb{R}^+$ is a positive weight associated with a scenario $s \in \mathcal{S}$ (e.g., a probability), $g_s \in \mathbb{R}^{n_2}$, $W_s \in \mathbb{R}^{m\times n_2}$, $T_s \in \mathbb{R}^{m\times n_1}$, $d_s \in \mathbb{R}^m$, and $\mathcal{X} \subset \mathbb{R}^{n_1}$ is a polyhedral set. Variables x are called *first-stage variables* and variables y_s are called *second-stage variables* or *recourse variables*. Problem (4.1) is called the *extensive formulation* of a two-stage stochastic problem.

When the number of scenarios is large, problem (4.1) becomes intractable for LP solvers. Its reformulation as

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s \phi(x,s) \\ s.t.\ x \in \mathcal{X} \end{cases} \tag{4.2}$$

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

$$\phi(x,s) = \begin{cases} \min\limits_y\ g_s^\top y \\ s.t.\ W_s y = d_s - T_s x \\ \qquad y \in \mathbb{R}^{n_2}_+ \end{cases} \tag{4.3}$$

makes the use of decomposition methods attractive. If we fix the first-stage variables to $\hat{x} \in \mathcal{X}$,

then the resulting problem becomes separable according to the scenarios. We denote by $(SP(\hat{x}, s))$ the subproblem associated with a scenario $s \in \mathcal{S}$ and by $\phi(\hat{x}, s)$ its value.

Let $\Pi_s = \{\pi \in \mathbb{R}^m : \ W_s^\top \pi \leq g_s\}$ be the polyhedron associated with the dual of $(SP(\hat{x}, s))$, which does not depend on first-stage variables x. We denote by $\text{Rays}(\Pi_s)$ the set of extreme rays of Π_s, and by $\text{Vert}(\Pi_s)$ the set of extreme points of Π_s. By Farkas' Lemma, we can write an expression of the domain of $\phi(\cdot, s)$ as $\text{dom}\Big(\phi(\cdot, s)\Big) = \{x \in \mathbb{R}^{n_1} : \ r_s^\top (d_s - T_s x) \leq 0, \ \forall r_s \in \text{Rays}(\Pi_s)\}$. Then we can replace in formulation (4.2) the polyhedral mapping $x \mapsto \phi(x, s)$ by its outer linearization on its domain. Using an epigraph variable θ_s for every $s \in \mathcal{S}$, we obtain the multicut Benders reformulation (Birge and Louveaux, 1988) of problem (4.1):

$$\begin{cases} \min\limits_{x,\theta} \ c^\top x + \sum\limits_{s \in \mathcal{S}} p_s \theta_s & \\ s.t. : \theta_s \geq \pi_s^\top (d_s - T_s x), \ \forall s \in \mathcal{S}, \ \forall \pi_s \in \text{Vert}(\Pi_s) & (i) \\ \qquad 0 \geq r_s^\top (d_s - T_s x), \quad \forall s \in \mathcal{S}, \ \forall r_s \in \text{Rays}(\Pi_s) & (ii) \\ \qquad x \in \mathcal{X}, \theta \in \mathbb{R}^{\text{card}(\mathcal{S})} & \end{cases} \tag{4.4}$$

Constraints (i) are called optimality cuts, and constraints (ii), feasibility cuts. Without loss of generality, we assume that the problem has *relatively complete recourse* (i.e., $\mathcal{X} \subset \text{dom}\,(\phi(\cdot, s))$ for every scenario $s \in \mathcal{S}$), meaning that every subproblem is feasible for every $x \in \mathcal{X}$. As a result, only optimality cuts are required in the Benders decomposition algorithm, and every $x \in \mathcal{X}$ defines an upper bound on the optimal value of the problem. Every two-stage linear stochastic program can be reformulated to a problem satisfying this hypothesis by introducing slack variables with large enough coefficients in the objective function (see e.g. (Bodur and Luedtke, 2022) or (Shapiro and Nemirovski, 2005)).

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 (4.1), referred to as the *Benders by batch* algorithm. The term *batch* refers to a given fixed partition of all subproblems into separate batches. We propose a new stopping criterion that allows us to identify that a solution cannot be proven optimal at the current iteration without necessarily having to solve all the subproblems. As a result, only few subproblems are generally solved at a first-stage candidate solution. To prevent introducing too many cuts in the relaxed master program, the algorithm can use partial cut aggregation, thus generating a single cut from all subproblems that belong to an identical batch. If the number of batches is equal to one, the Benders by batch algorithm is equivalent to the classical version of the Benders decomposition algorithm (multicut or monocut, depending on the use of cut aggregation).

Several existing methods based on similar ideas require fixed recourse ($W_s = W, \ \forall s \in \mathcal{S}$ in problem (4.1)) (Oliveira et al., 2011) and deterministic second-stage objective function ($g_s = g, \ \forall s \in \mathcal{S}$ in problem (4.1)) (Wets, 1983; Dantzig and Infanger, 1991; Higle and Sen, 1991).Moreover, some of them do not have finite convergence (Higle and Sen, 1991), or are not exact (Dantzig and Infanger, 1991). The method proposed in this work is exact, has finite convergence, and does not require any assumption on the value of the random parameters

g_s, W_s, d_s, T_s in problem (4.1).

We also show how to stabilize the proposed algorithm. As the classical primal stabilization methods of the literature (Ben-Ameur and Neto, 2007; Lemaréchal et al., 1995) are designed for algorithms which solve all the subproblems at each iteration, it is not possible to apply them directly. They require the actual value of the recourse function at each iteration, at least to evaluate their stopping criterion. We therefore propose a generic framework to stabilize the Benders by batch algorithm and prove the finite convergence and exact behavior of the stabilized algorithm. Our algorithm is also compatible with classical dual stabilization techniques (Magnanti and Wong, 1981; Papadakos, 2008; Sherali and Lunday, 2013). We show in Table 4.1 a comparison between the current work and classical algorithm to solve large-scale two-stage stochastic programs.

Paper	Randomness hypothesis*	Solve all SPs	Monocut or multicut	Exact method	Finite convergence	Cut aggregation	Stabilization
(Crainic et al., 2021)	$g_s = g, W_s = W \, \forall s \in \mathcal{S}$	Yes	Both	Yes	Yes	No	No
(Song and Luedtke, 2015)	$g_s = g, W_s = W \, \forall s \in \mathcal{S}$	Yes	Not applicable	Yes	Yes	No	No
(van Ackooij et al., 2017)	$g_s = g, W_s = W \, \forall s \in \mathcal{S}$	No	Both	Yes	Yes	No	Level
(Wets, 1983)	$g_s = g, W_s = W \, \forall s \in \mathcal{S}$	No	Both	Yes	Yes	No	No
(Dantzig and Infanger, 1991)	$g_s = g, W_s = W \, \forall s \in \mathcal{S}$	No	Monocut	No	Yes	No	No
(Higle and Sen, 1991)	$g_s = g, W_s = W \, \forall s \in \mathcal{S}$	No	Monocut	Yes	No	No	No
(Trukhanov et al., 2010)	No	Yes	Multicut	Yes	Yes	Yes	No
(Linderoth and Wright, 2003)	No	Yes	Multicut	Yes	Yes	No	Trust-region
(Wolf et al., 2014)	No	All or none	Monocut and Multicut	Yes	Yes	No	Level
(Oliveira et al., 2011)	$W_s = W \, \forall s \in \mathcal{S}$	No	Monocut	Inexact	Yes	No	Proximal bundle
This work	No	No	Multicut	Yes	Yes	Yes	In-out

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

Table 4.1: Comparison of stochastic methods to accelerate Benders decomposition. (SPs: subproblems)

The contributions of this chapter can be summarized as follows:

- We propose a new exact algorithm to solve the Benders reformulation of two-stage linear stochastic programs with finite probability distribution. This algorithm is based on a sequential stopping criterion relying on a partition of the subproblems. This stopping criterion allows the algorithm to solve only a few subproblems at most iterations by detecting that a first-stage candidate solution cannot be proven optimal early in the subproblems solution process.
- We develop a general framework to apply primal stabilization to the Benders by batch algorithm, as classical primal stabilization methods cannot be applied if all the subproblems are not solved at each iteration. We state sufficient conditions for the stabilized algorithm to be exact and have finite convergence and provide two effective primal stabilization schemes.
- We perform an extensive numerical study showing the efficiency of the developed algorithm on some classical stochastic instances from the literature compared to classical implementations of the monocut and multicut Benders decomposition algorithm, with and without in-out stabilization, the static multicut aggregation approach of Trukhanov et al. (2010), and a level bundle method.

The paper is organized as follows. In section 4.2, we present the Benders by batch algorithm. Section 4.3 presents a general framework to stabilize our algorithm and two stabilization schemes:

the first one based on the classical in-out separation scheme, and the second one based on exponential moving averages. Section 4.4 presents extensive computational experiments. Then, section 4.5 concludes and outlines perspectives.

4.2 The Benders by batch algorithm

We propose a new algorithm, hereafter referred to as the Benders by batch algorithm, to solve exactly the multicut Benders reformulation (4.4) of a two-stage stochastic linear program. The algorithm consists of solving the subproblems by batch and stopping solving subproblems at an iteration as soon as we identify that the current first-stage solution cannot be proven optimal. This is made possible by checking, after solving of a subset of subproblems, if the gap between their optimal values and their epigraph approximations in the relaxed master program already exceeds the optimality gap.

We first present some notations necessary to formally describe the algorithm. We consider an ordered set of scenarios $\mathcal{S} = \{s_1, s_2, ..., s_{\text{card}(\mathcal{S})}\}$ and a given batch size $1 \leq \eta \leq \text{card}(\mathcal{S})$. We define $\kappa = \lceil \text{card}(\mathcal{S})/\eta \rceil$ as the number of batches of subproblems. For every $i \in [\![1, \kappa]\!]$, the i^{th} batch of subproblems $\mathcal{S}_i$ is defined as $\mathcal{S}_i = \{s_{(i-1)\eta+1}, ..., s_{(i-1)\eta+\eta_i}\}$, where η_i is the size of batch i, $\eta_1 = \cdots = \eta_{\kappa-1} = \eta$ and $\eta_\kappa = (\text{card}(\mathcal{S}) \mod \eta)$. Family $(\mathcal{S}_i)_{i \in [\![1,\kappa]\!]}$ defines a partition of $\mathcal{S}$. We restrict ourselves to batches of the same size, but the method remains valid for any partition of $\mathcal{S}$. We denote by $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s \in \mathcal{S}})$ the optimal solution to $(RMP)^{(k)}$ at iteration k of the algorithm, where $\check{x}^{(k)}$ denotes the optimal value to the first-stage variables and $\check{\theta}_s^{(k)}$ the optimal value to the epigraph variable associated with scenario $s \in \mathcal{S}$. A lower bound on the optimal value of problem (4.1) is then computed as $LB^{(k)} = c^\top \check{x}^{(k)} + \sum_{s \in \mathcal{S}} p_s \check{\theta}_s^{(k)}$. For a first-stage solution $x \in \mathcal{X}$, we denote by $UB(x) = c^\top x + \sum_{s \in \mathcal{S}} p_s \phi(x, s)$ an upper bound on the optimal value of problem (4.1). Let $\varepsilon \geq 0$ be the optimality gap of the algorithm. We first define the notion of *provable optimality* in cutting-planes methods.

Definition 4. *Let $\varepsilon \geq 0$ be the optimality gap of the algorithm and k an iteration of the algorithm. We say that a first-stage solution $x \in \mathcal{X}$ cannot be proven optimal at an iteration k of the algorithm iff $UB(x) - LB^{(k)} > \varepsilon$.*

Saying that a first-stage solution x cannot be proven optimal at an iteration k of the algorithm means that, either x is not an optimal solution to problem (4.1), or the current lower bound given by $(RMP)^{(k)}$ is too low to prove the optimality of an optimal solution. The classical stopping criterion $UB - LB \leq \varepsilon$ of the Benders decomposition algorithm is based on such an optimality proof, but cannot be directly applied if not all the subproblems are solved. Specifically, an upper bound on the optimal value of the problem is only known after computing, for a first-stage solution $x \in \mathcal{X}$, the optimal value $\phi(x, s)$ of every subproblem $(SP(x, s))$.

We propose hereafter a new stopping criterion, which detects, when it occurs, that the current first-stage solution $\check{x}^{(k)}$ to $(RMP)^{(k)}$ cannot be proven optimal without necessarily having to solve all the subproblems. If after having solved some batches of subproblems, the sum of the differences between their value and their epigraph approximation in $(RMP)^{(k)}$ already exceeds the optimality gap ε, the algorithm does not solve the remaining batches of subproblems, as we already know that $\check{x}^{(k)}$ cannot be proven optimal (see Proposition 1). In this way, the Benders by

batch algorithm is likely to explore more first-stage solutions than classical Benders decomposition algorithms as it tends to solve only a few number of subproblems at most iterations. The proposed stopping criterion is based on the concept of ε_i-approximation that we define below.

Definition 5 (ε_i-approximation)**.** *Let $\varepsilon \geq 0$ 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 $\mathcal{S}_{\sigma(i)}$ is ε_i-approximated by $(RMP)^{(k)}$ if*

$$\sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)} \right) \leq \varepsilon_i \tag{4.5}$$

with $\varepsilon_i = \varepsilon - \sum_{t=1}^{i-1} \sum_{s \in \mathcal{S}_{\sigma(t)}} p_s \left(\phi\left(\check{x}^{(k)}, s\right) - \check{\theta}_s^{(k)} \right)$.

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

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

Proposition 1. *Let $\varepsilon \geq 0$ 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 $\mathcal{S}_{\sigma(i)}$ is ε_i-approximated by $(RMP)^{(k)}$ for every index $i \in [\![1, \kappa]\!]$.*

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

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

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

$$\begin{aligned}
& \sum_{t=1}^{\kappa} \sum_{s \in \mathcal{S}_{\sigma(t)}} p_s \left(\phi\left(\check{x}^{(k)}, s\right) - \check{\theta}_s^{(k)} \right) \leq \varepsilon \\
\iff & \sum_{t=i}^{\kappa} \sum_{s \in \mathcal{S}_{\sigma(t)}} p_s \left(\phi\left(\check{x}^{(k)}, s\right) - \check{\theta}_s^{(k)} \right) \leq \varepsilon_i, \ \forall i \in \{1, \ldots, \kappa\}
\end{aligned}$$

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

$$\sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \left(\phi\left(\check{x}^{(k)}, s\right) - \check{\theta}_s^{(k)} \right) \leq \varepsilon_i, \ \forall i \in \{1, \ldots, \kappa\}$$

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

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

$$\sum_{s \in \mathcal{S}_{\sigma(\kappa)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)}\right) \leq \varepsilon_\kappa \tag{4.6}$$

By definition of ε_κ we have:

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

Then, using equation (4.6), we have:

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

which implies that $\check{x}^{(k)}$ is an optimal solution to problem (4.2). □

Corollary 3. *Let $\varepsilon \geq 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration, and σ a permutation of $[\![1, \kappa]\!]$. If there exists an index $i \in [\![1, \kappa]\!]$ such that $\sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)}\right) > \varepsilon_i$, then $\check{x}^{(k)}$ cannot be proven optimal.*

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

We now present the Benders by batch algorithm (Algorithm 4.1). The while loop from lines 3 to 20 will be referred hereafter as the *master loop*. Each pass of this loop corresponds to an iteration of the algorithm. At iteration k, the relaxed master program $(RMP)^{(k)}$ is solved to obtain a new first-stage solution $\check{x}^{(k)}$. A permutation σ of $[\![1, \kappa]\!]$ is then chosen. This permutation defines the order in which the batches of subproblems $(\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_\kappa)$ will be solved at the current first-stage solution. The while loop from lines 8 to 19 will be referred as the *optimality loop*. In each pass in this loop:

- the subproblems of the current batch $\mathcal{S}_{\sigma(i)}$ are solved (lines 9 to 10). This part of the algorithm can be parallelized, as in the classical Benders decomposition algorithm, to accelerate the procedure.
- the cuts defined by the solutions of the subproblems are added to the relaxed master program (lines 11 to 15). We add a parameter `cutAggr` to the algorithm. If this parameter is set to `False`, the cuts of each subproblem are added independently to the relaxed master program, as it is the case in the classical multicut Benders decomposition algorithm. If

Algorithm 4.1: The Benders by batch algorithm

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

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

2 Define a partition $(\mathcal{S}_i)_{i \in [\![1,\kappa]\!]}$ of the subproblems according to batch size η

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

4 $\quad k \leftarrow k + 1$

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

6 $\quad i \leftarrow 1$, $\varepsilon_1 \leftarrow \varepsilon$, `stay_at_x` $\leftarrow$ `True`

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

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

9 $\quad\quad$ **for** $s \in \mathcal{S}_{\sigma(i)}$ **do**

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

11 $\quad\quad$ **if** `cutAggr` **then**

12 $\quad\quad\quad$ Add $\sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \theta_s \geq \sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \left(\pi_s^\top (d_s - T_s x)\right)$ to $(RMP)^{(k)}$

13 $\quad\quad$ **else**

14 $\quad\quad\quad$ **for** $s \in \mathcal{S}_{\sigma(i)}$ **do**

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

16 $\quad\quad\quad$ **if** $\sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \left(\phi(\check{x}^{(k)}, s) - \check{\theta}_s^{(k)}\right) \leq \varepsilon_i$ **then**

17 $\quad\quad\quad\quad \varepsilon_{i+1} \leftarrow \varepsilon_i - \sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \left(\phi\left(\check{x}^{(k)}, s\right) - \check{\theta}_s^{(k)}\right)$

18 $\quad\quad\quad\quad i \leftarrow i + 1$

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

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

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

this parameter is set to `True`, we add only one cut, computed as the weighted sum of all the cuts of the batch according to the probability distribution.

- the gap between the value of the subproblems and the value of their outer linearization is checked (line 16 to 19). If the batch is ε_i-approximated by $(RMP)^{(k)}$, then i is increased by one, and the boolean `stay_at_x` still equals `True`. The algorithm returns to line 8 and solves a new batch at the same first-stage solution, as i has been incremented. If it reaches $i = \kappa + 1$, then all batches are ε_i-approximated by $(RMP)^{(k)}$ according to permutation σ, and $\check{x}^{(k)}$ is an optimal solution to problem (4.2). If one of the batches is not ε_i-approximated by $(RMP)^{(k)}$, then $\check{x}^{(k)}$ cannot be proven optimal. Then there exists at least one of the cuts which excludes the solution $(\check{x}^{(k)}, \check{\theta}^{(k)})$ from the relaxed master program. The algorithm exits the optimality loop, and goes to line 3 to solve again the relaxed master program.

Remark 2 (Partial cut aggregation)**.** *One of the most important drawbacks of the multicut Benders decomposition algorithm is the large number of cuts added to the relaxed master program at each iteration. As this number of cuts increases, the time needed to solve the master program can increase dramatically. The Benders by batch algorithm might suffer from the same effect, even if this effect might be delayed by the method (it adds fewer cuts at each iteration). We propose to aggregate the cuts of a batch, and add only one cut computed as* $\sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \theta_s \geq$

$\sum_{s\in\mathcal{S}_{\sigma(i)}} p_s\left(\pi_s^\top(d_s - T_s x)\right)$. *As the subproblems are linearly independent, this cut is the Benders cut associated with the problem created by concatenation of the subproblems of a batch. As the partition of the subproblems into batches is done prior to the algorithm, the cuts of the same subproblems are always aggregated together. This can be seen as the static cut aggregation strategy used in (Trukhanov et al., 2010).*

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

Proposition 2. *Let $\varepsilon \geq 0$ be the optimality gap. The Benders by batch algorithm converges to an optimal solution to problem (4.1) in a finite number of iterations.*

Proof of proposition 2. We solve each subproblem at most once for every optimal solution to $(RMP)^{(k)}$ because $(\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_\kappa)$ defines a partition of $\mathcal{S}$. Then if there exists a cut violated by $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in\mathcal{S}})$, we find it in at most $\text{card}(\mathcal{S})$ iterations in the optimality loop. Then, as the total number of optimality cuts is finite and equal to $\sum_{s\in\mathcal{S}} \text{card}(\text{Vert}(\Pi_s))$, this algorithm converges in at most $\text{card}(\mathcal{S}) \times \sum_{s\in\mathcal{S}} \text{card}(\text{Vert}(\Pi_s))$ iterations. When the cuts are aggregated, if the cut of a subproblem separates the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in\mathcal{S}})$, 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 $\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_\kappa$ and $\sigma = id$ at the first iteration. When we choose a new permutation, at the beginning of a master loop, the *ordered strategy* consists of starting from the first batch of subproblems that has not been solved at the previous first-stage solution. We introduce the following cyclic permutation μ of the batches:

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

Let N be the number of batches solved at the previous first-stage solution. Then, the ordered strategy consists of defining the new permutation σ at line 7 of Algorithm 4.1 as $\sigma \leftarrow \mu^N \circ \sigma$.

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

4.3 Stabilization of the Benders by batch algorithm

The Benders by batch algorithm introduced in the previous section (Algorithm 4.1) may suffer, as every cutting-plane algorithm, from strong oscillations of the first-stage variables, and thus may compute, in the early iterations, cuts that exclude solutions that are far away from the optimal solution (see e.g. (Vanderbeck, 2005) section 7). However, the classical primal stabilization procedures presented in Chapter 3 do not apply directly if we do not solve all the subproblems at each iteration as they require the value of the recourse function for the current first-stage

solution. We propose in this section a general framework to stabilize our algorithm, and show a sufficient condition for the convergence of the stabilized algorithm.

4.3.1 The stabilized Benders by batch algorithm

Many effective primal stabilization methods for cutting-plane algorithms solve, at each iteration, a separation problem in a point $x^{(k)}$ (hereafter referred to as the separation point) that is different from the current optimal first-stage solution $\check{x}^{(k)}$ to the relaxed master program (Zverovich et al., 2012; Pessoa et al., 2013). We define hereafter formally a primal stabilization scheme, in which the separation point is computed as the image by a given mapping of a vector defining the state of the stabilization. Such a scheme must also incorporate a way to update this state vector.

Definition 6 (Primal stabilization scheme)**.** *A primal stabilization scheme is characterized by a triplet* $(\mathcal{D}, \psi_1, \psi_2)$ *where* $\mathcal{D}$ *is a stabilization state space and* (ψ_1, ψ_2) *is a pair of mappings* $\begin{cases} \psi_1 : X \times \mathcal{D} \to \mathcal{D} \\ \psi_2 : \mathcal{D} \to X \end{cases}$ *such that* ψ_2 *is surjective.*

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

We now present how to adapt the Benders by batch algorithm (Algorithm 4.1) when such a primal stabilization scheme is used. We generalize Definition 5 and Proposition 1 to take into account that the lower bound at a given iteration k is computed based on the current optimal solution $\check{x}^{(k)}$ to RMP, while the subproblems are solved at a separation point x that is usually different from $\check{x}^{(k)}$. As this difference between the first-stage solutions induces a difference in the first-stage cost, we subtract in the definition of the remaining gap ε_i the difference $c^\top(x - \check{x}^{(k)})$. Because $\check{\theta}_s^{(k)}$ is a lower bound on $\phi\left(\check{x}^{(k)}, s\right)$, but not on $\phi(x, s)$, we also need to account for cases where $\phi(x, s) - \check{\theta}_s^{(k)} < 0$.

Definition 7 ($\varepsilon_i(x)$-approximation at a first-stage solution x)**.** *Let* $\varepsilon \geq 0$ *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* $\mathcal{S}_{\sigma(i)}$ *is* $\varepsilon_i(x)$*-approximated by* $(RMP)^{(k)}$ *at* $x \in \mathcal{X}$ *if*

$$\Big[\sum_{s \in \mathcal{S}_{\sigma(i)}} p_s \left(\phi(x, s) - \check{\theta}_s^{(k)} \right) \Big]^+ \leq \varepsilon_i(x)$$

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

Remark 3. *Saying that a batch* $\mathcal{S}_{\sigma(i)}$ *is* $\varepsilon_i(\check{x}^{(k)})$*-approximated by* $(RMP)^{(k)}$ *is equivalent to saying that* $\mathcal{S}_{\sigma(i)}$ *is* ε_i*-approximated by* $(RMP)^{(k)}$ *in Algorithm 4.1.*

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

Proposition 3. *Let $\varepsilon \geq 0$ be the optimality gap of the algorithm, $k \in \mathbb{Z}^+$ an iteration of the algorithm, and σ a permutation of $[\![1, \kappa]\!]$. If there exists a first-stage solution $x \in \mathcal{X}$ such that batch $\mathcal{S}_{\sigma(i)}$ is $\varepsilon_i(x)$-approximated by $(RMP)^{(k)}$, for all $i \in [\![1, \kappa]\!]$, then x is an optimal solution to problem (4.2).*

Proof of proposition 3. Let $x \in \mathcal{X}$ be a first-stage solution such that batch $\mathcal{S}_{\sigma(i)}$ is $\varepsilon_i(x)$-approximated by $(RMP)^{(k)}$, for all $i \in [\![1, \kappa]\!]$. Then, $\mathcal{S}_{\sigma(\kappa)}$ is $\varepsilon_\kappa(x)$-approximated by $(RMP)^{(k)}$. This means:

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

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

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

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

and x is an optimal solution to problem (4.2). □

We now present the stabilized Benders by batch algorithm (Algorithm 4.2).

As, at each iteration, the cuts are now generated from a first-stage solution $x^{(k)}$ that may be different from the first-solution to $(RMP)^{(k)}$, there is no guarantee that the cuts added separate the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s \in \mathcal{S}})$. When there is no cut, among added cuts, that separates the solution to the relaxed master program, we say that first-stage solution $x^{(k)}$ induces a **mis-pricing** (Pessoa et al., 2013). We represent such a case in Figure 4.1. Then, there is no need to solve again the relaxed master program as its solution remains the same. A boolean variable `misprice` appears in Algorithm 3 to handle such a case.

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

Algorithm 4.2: The stabilized Benders by batch algorithm

Parameters: $\varepsilon \geq 0$, $\eta \in [\![1, \text{card}(\mathcal{S})]\!]$ the batch size, `cutAggr` $\in \{$`True`, `False`$\}$, a primal stabilization scheme $(\mathcal{D}, \psi_1, \psi_2)$ and an initial stabilization state vector $d^{(0)} \in \mathcal{D}$.

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

- **Case 1:** The current batch is $\varepsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$. It satisfies the condition of line 20 of Algorithm 4.2. Then, `stay_at_x` still equals `True` at the end of the loop, and i is incremented by one. If the algorithm reaches $i = \kappa + 1$, then the algorithm stops, and $x^{(k)}$ is an optimal solution to the problem with an optimality gap $\varepsilon \geq 0$. Otherwise, the algorithm solves the next batch of subproblems at the same first-stage solution.

- **Case 2:** The current batch $\mathcal{S}_{\sigma(i)}$ is not $\varepsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$ and there exists no cut derived from this batch of subproblems, or a previous batch, which separates the solution $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s \in \mathcal{S}})$ to the relaxed master program [see Figure 4.1]. The variable `misprice` still equals `True`. As the solution to the relaxed master program has not been

cut, it is useless to solve the relaxed master program again. We exit the optimality loop, but stay in the separation loop. We define a new separation point $x^{(k)}$, a new permutation of $[\![1, \kappa]\!]$, and begin a new optimality loop.

- **Case 3:** The current batch $\mathcal{S}_{\sigma(i)}$ is not $\varepsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$ and at least one of the cuts derived from this batch of subproblems separates the solution $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in\mathcal{S}})$ to the relaxed master program [see Figure 4.2]. This means that `misprice` is set to `False`. The variable `stay_at_x` is set to `False` and we exit the optimality loop. Since `misprice` equals `False`, we exit the separation loop. We then go to line 3, and solve again the relaxed master program.

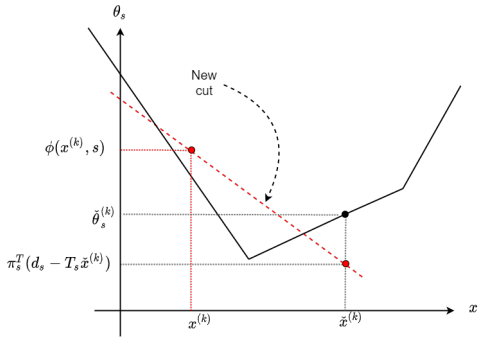


Figure 4.1: The cut derived from first-stage solution $x^{(k)}$ does not separate the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in\mathcal{S}})$. The solution to $(RMP)^{(k)}$ remains the same. The separation point $x^{(k)}$ induces a mis-pricing.

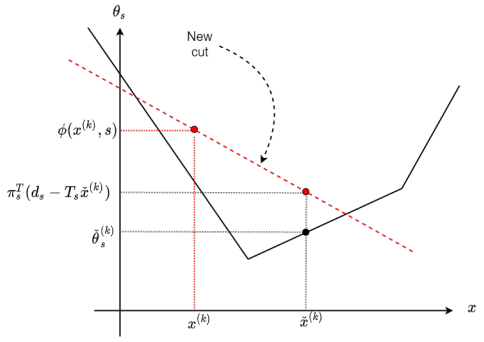


Figure 4.2: The cut derived from first-stage solution $x^{(k)}$ separates the solution to the relaxed master program $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in\mathcal{S}})$.

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

In this section we prove that, if the sequence of separation points produced by the primal stabilization scheme converges to the solution to the relaxed master program when this latter solution remains constant over the iterations (i.e., during a mis-pricing sequence), then the stabilized Benders by batch algorithm (Algorithm 4.2) converges to an optimal solution to problem (4.1) in a finite number of iterations.

Definition 8 (Convergence property and finite convergence property of a primal stabilization scheme)**.** *Let $(\mathcal{D}, \psi_1, \psi_2)$ be a primal stabilization scheme. For every $(x, d) \in \mathcal{X} \times \mathcal{D}$ we define $(d_x^\ell)_{\ell\in\mathbb{N}^*}$ as*

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

the sequence of stabilization state vectors obtained by successive applications of ψ_1 on a constant first-stage solution $x \in \mathcal{X}$.

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

if:

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

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

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

We first need to prove the following intermediate results to show that the stabilized Benders by batch algorithm effectively converges to an optimal solution to problem (4.1).

Proposition 4. *Let* $\varepsilon > 0$ *(resp.* $\varepsilon \geq 0$*) be the optimality gap of Algorithm 4.2,* $k \in \mathbb{Z}^+$ *an iteration, and* $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s \in \mathcal{S}})$ *an optimal solution to* $(RMP)^{(k)}$*. If* $(x^{(k+r)})_{r \in \mathbb{N}}$ *is a sequence of elements of* $\mathcal{X}$ *converging to* $\check{x}^{(k)}$ *(resp. converging to* $\check{x}^{(k)}$ *in a finite number of iterations) and* $(\sigma^{(k+r)})_{r \in \mathbb{N}}$ *a sequence of permutations of* $[\![1, \kappa]\!]$*, then there exists* $t \in \mathbb{N}$ *such that one of the following assertions is true:*

1. *First-stage solution* $x^{(k+t)}$ *is proven to be an optimal solution to problem (4.1) with an optimality gap of* $\varepsilon > 0$ *(resp.* $\varepsilon \geq 0$*).*
2. *There exists a cut generated in* $x^{(k+t)}$ *which separates* $(\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s \in \mathcal{S}})$*.*

Proof of proposition 4. The proof consists of two cases:

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

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

Sub-case 1.1: Assume that there exists $t_0 \in \mathbb{N}$ such that for all $l \geq t_0$ and for each index $i \in [\![1, \kappa]\!]$, batch $\mathcal{S}_{\sigma^{(k+l)}(i)}$ is $\varepsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$ with an initial gap of $\frac{\varepsilon}{4}$. This means that for every $l \geq t_0$ and for every index $i \in [\![1, \kappa]\!]$,

$$\Big[\sum_{s \in \mathcal{S}_{\sigma^{(k+l)}(i)}} p_s \left(\phi\left(\check{x}^{(k)}, s\right) - \check{\theta}_s^{(k)} \right) \Big]^+ \leq \frac{\varepsilon}{4} - \Big[\sum_{t=1}^{i-1} \sum_{s \in \mathcal{S}_{\sigma^{(k+l)}(t)}} p_s \left(\phi\left(\check{x}^{(k)}, s\right) - \check{\theta}_s^{(k)} \right) \Big]^+ \tag{4.7}$$

As the number of permutations of $[\![1, \kappa]\!]$ is finite, as for every $l \geq t_0$ and for each index $i \in [\![1, \kappa]\!]$, the application $x \mapsto \left[\sum_{s \in \mathcal{S}_{\sigma^{(k+l)}(i)}} p_s \left(\phi\left(x, s\right) - \check{\theta}_s^{(k)} \right) \right]^+$ is continuous, and as sequence

$\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ converges to $\check{x}^{(k)}$, there exists $t_1 \in \mathbb{N}, t_1 \geq t_0$ such that, for every $l \geq t_1$ and for every index $i \in [\![1,\kappa]\!]$:

$$\Big[\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(i)}} p_s\left(\phi\left(x^{(k+l)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+ \leq \Big[\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(i)}} p_s\left(\phi\left(\check{x}^{(k)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+ + \frac{\varepsilon}{4} \tag{4.8}$$

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

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

$$\Rightarrow -\Big[\sum_{t=1}^{i-1}\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(i)}} p_s\left(\phi\left(\check{x}^{(k)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+ \leq -\Big[\sum_{t=1}^{i-1}\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(t)}} p_s\left(\phi\left(x^{(k+l)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+ + \frac{\varepsilon}{4} \tag{4.9}$$

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

Then, by setting $t_4 = \max\{t_1, t_2, t_3\}$, and jointly using (4.7), (4.8) and (4.9), we have, for every $l \geq t_4$ and for every index $i \in [\![1,\kappa]\!]$:

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

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

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

And for every index $i \in [\![1,\kappa]\!]$, batch $\mathcal{S}_{\sigma^{(k+t_4)}(i)}$ is $\varepsilon_i(x^{(k+t_4)})$-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 (4.2).

Sub-case 1.2: Now assume that for all $t_0 \in \mathbb{N}$, there exists $l \geq t_0$ and an index $i \in [\![1,\kappa]\!]$ such that batch $\mathcal{S}_{\sigma^{(k+l)}(i)}$ is not $\varepsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$ with an initial optimality gap of $\frac{\varepsilon}{4}$. This means, that for all $t_0 \in \mathbb{N}$, there exists $l \geq t_0$ and an index $i \in [\![1,\kappa]\!]$ such that:

$$\Big[\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(i)}} p_s\left(\phi\left(\check{x}^{(k)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+ > \frac{\varepsilon}{4} - \Big[\sum_{t=1}^{i-1}\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(t)}} p_s\left(\phi\left(\check{x}^{(k)},s\right)-\check{\theta}_s^{(k)}\right)\Big]^+ \tag{4.10}$$

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

first index such that (4.10) occurs) such that:

$$\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(i)}} p_s\left(\phi(\check{x}^{(k)},s)-\check{\theta}_s^{(k)}\right)>\delta \tag{4.11}$$

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

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

By continuity of $\phi(.,s)$ for all $s\in\mathcal{S}$ and as the total number of cuts is finite, there exists $L>0$ such that for every $l\in\mathbb{N}$ and for every $i\in[\![1,\kappa]\!]$, $||g_i^{(k+l)}||_2\leq L$. 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 $l\geq t_1$ and for all $i\in[\![1,\kappa]\!]$,

$$|g_i^{(k+l)\top}(\check{x}-x^{(k+l)})|<\frac{\delta}{3} \tag{4.12}$$

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

$$\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(i)}} p_s\phi(\check{x}^{(k)},s)<\sum_{s\in\mathcal{S}_{\sigma^{(k+l)}(i)}} p_s\phi(x^{(k+l)},s)+\frac{\delta}{3} \tag{4.13}$$

Then, let $t_3=\max\{t_1,t_2\}$. Let $i\in[\![1,\kappa]\!]$ and $l_0\geq t_3$ be the first indices such that (4.11) occurs. We have, by combining (4.11), (4.12) and (4.13):

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

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

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

  As $\left(x^{(k+r)}\right)_{r\in\mathbb{N}}$ converges to $\check{x}^{(k)}$, the proof of case 1 holds also in this case for every $\varepsilon>0$. We need to prove that the proposition is true if $\varepsilon=0$. Let t_0 be the first iteration such that $x^{(k+t_0)}=\check{x}^{(k)}$. Either, for each index $i\in[\![1,\kappa]\!]$, batch $\mathcal{S}_{\sigma^{(k+t_0)}(i)}$ is $\varepsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$ with an optimality gap of 0, and by proposition 3, $x^{(k+t_0)}$ is an optimal solution to problem (4.2) with an optimality gap $\varepsilon=0$, or there exists a batch which is not $\varepsilon_i(\check{x}^{(k)})$-approximated by $(RMP)^{(k)}$, and the aggregated cut derived from this batch separates the solution to the relaxed master program.

□

Proposition 5. *If the primal stabilization scheme satisfies the* ***convergence property*** *(resp.* ***finite convergence property****) of Definition 8, then the stabilized Benders by batch algorithm converges to an optimal solution to problem (4.1) in a finite number of iterations, for every $\varepsilon > 0$ (resp. $\varepsilon \geq 0$).*

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

1. $\forall i \in [\![1, \kappa]\!]$, batch $\mathcal{S}_{\sigma(i)}$ is $\varepsilon_i(x^{(k)})$-approximated by $(RMP)^{(k)}$. Then $x^{(k)}$ is an optimal solution to problem (4.1) with an optimality gap of $\varepsilon > 0$ (resp. $\varepsilon \geq 0$).

2. There exists an index $i \in [\![1, \kappa]\!]$ such that solving the subproblems of batch $\mathcal{S}_{\sigma(i)}$ generates a cut which separates the solution to $(RMP)^{(k)}$. As the total number of cuts is finite, we can only be in this situation a finite number of times.

3. There exists no cut derived at $x^{(k)}$ which separates the solution to $(RMP)^{(k)}$. Then, $x^{(k)}$ induces a mis-pricing. The solution to $(RMP)^{(k+1)}$ remains the same. Let suppose that this happens during an infinite number of consecutive iterations. Then, as the primal stabilization scheme satisfies the convergence property (resp. the finite convergence property) , the sequence of separation points converges to $\check{x}^{(k)}$ (resp. in a finite number of iterations) . Prop. 4 states that in that case, we end up in a finite number of iterations in case 1 or case 2.

In conclusion, the stabilized Benders by batch algorithm ends in a finite number of iterations in case 1, and finds an optimal solution to problem (4.1). □

Remark 4. *The classical Benders decomposition algorithm is equivalent to the Benders by batch algorithm with a batch size $\eta = card(\mathcal{S})$. Therefore, Algorithm 4.2 describes a valid way to add primal stabilization to the classical Benders decomposition algorithm (providing that the primal separation scheme satisfies the convergence property).*

4.3.3 Two primal stabilization schemes satisfying the convergence property

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

We present below two primal stabilization schemes.

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

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

for $k \geq 1$, and $x^{(0)} \in \mathcal{X}$ is a feasible first-stage solution. This basically consists in doing $100\alpha\%$ of the way from the previous separation point to the solution to the master program. This can be seen as an *in-out stabilization*, updating the stability center to the last separation point at each iteration. By convexity of $\mathcal{X}$, $x^{(k)}$ belongs to $\mathcal{X}$ for every $k \in \mathbb{N}$.

The basic stabilization scheme can be expressed according to Definition 6 as:

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

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

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

Proof of proposition 6. Let $(x,(y,z)) \in \mathcal{X} \times \mathcal{D}$. We have:

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

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

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

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

Scheme 2 - Solution memory stabilization: This stabilization uses an exponentially weighted average of the previous master solutions to compute the separation point. We choose a stabilization parameter $\alpha \in (0,1]$ and a memory parameter $\beta \in [0,1)$. We also define the exponentially weighted averaged point $\bar{x}^{(k)}$ on master solutions. The separation point is computed as follows:

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

for $k \geq 1$, and $x^{(0)} = \bar{x}^{(0)} \in \mathcal{X}$ is a feasible first-stage solution. By convexity of $\mathcal{X}$, $x^{(k)}$ belongs to $\mathcal{X}$ for every $k \in \mathbb{N}$. This stabilization takes inspiration from the stochastic gradient algorithm with momentum (Polyak, 1964) that has proven its efficiency in solving large-scale stochastic programs in the field of deep learning (Sutskever et al., 2013).

The solution memory stabilization scheme can be expressed according to Definition 6 as:

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

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

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

Proof of proposition 7. Let $(x, (y, z)) \in \mathcal{X} \times \mathcal{D}$. We have:

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

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

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

By induction, we have

$$d_x^\ell = (x + \beta^\ell u, x + \alpha(\textstyle\sum_{i=1}^{\ell-1} \beta^i (1-\alpha)^{\ell-i-1})u + (1-\alpha)^{\ell-1} v), \ \forall \ell \geq 2$$

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

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

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

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

4.4 Experimental design and numerical results

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

4.4.1 Instances

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

As those instances have a tremendous number of scenarios (see Table 4.2), 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 $\mathcal{S}$, with random seeds $\mathcal{S} + k,\ k \in \{0, 1, 2\}$ so that two instances of different sample size should not share sub-samples. This leads to a benchmark of 84 different instances. In the following, we will refer to the instances of problem *prob* with *#scenarios* scenarios as *prob-N#scenarios*.

4.4.2 Experimental Design

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

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

problem	first-stage	second-stage	scenarios
LandS	2×4	7×12	10^6
gbd	4×17	5×10	$\sim 10^5$
20term	3×64	124×764	$\sim 10^{12}$
ssn	1×89	175×706	$\sim 10^{70}$
storm	185×121	528×1259	$\sim 10^{81}$
Fleet20_3	3×60	321×1921	$> 3^{200}$
product	75×1500	700×1450	3^{450}

Table 4.2: Instances sizes, given in the format *lines* $\times$ *columns*

associated with the subproblems to 0 as it is a valid lower bound on LandS, gbd, ssn, storm, Fleet20_3 and 20term instances and to -10^{10} on product instances as 0 is not a valid lower bound on those instances.

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

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

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

CPLEX Benders is run with the following parameter values: `benders strategy 2` (an annotation file contains the first-stage variables, and CPLEX automatically decomposes the subproblems), `threads 1` (to run CPLEX using one core, as the other methods), `timelimit 43200` (time limit of twelve hours). **Classic multicut** follows Algorithm 2.1. In **Classic monocut** and **In-out monocut**, we compute the cuts as $\sum_{s\in\mathcal{S}} p_s\theta_s \geq \sum_{s\in\mathcal{S}} p_s\Big(\pi_s^\top(d_s - T_s x)\Big)$.

The subproblems are solved with the dual simplex algorithm for all methods. In all our implementations, the first-stage variables appear as variables in all the subproblems, and are fixed to the desired values during the optimization process. The coefficients of the cuts are computed as the reduced cost of those variables in an optimal solution to the subproblems.

In **Level Bundle**, **In-out multicut**, **In-out monocut** and **In-out CutAggr** and **BbB**

with stabilization, the starting solution $x^{(0)}$ is obtained by solving the mean-value problem. We use a dynamic strategy to update the stabilization parameter α in **In-out monocut**, **In-out multicut** and **In-out CutAggr**. If $c^\top x^{(k)} + \sum_{s\in\mathcal{S}} p_s \phi(s, x^{(k)}) < c^\top \hat{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \phi(s, \hat{x}^{(k)})$, then the separation point has a lower cost than the current stability center. If we had separated farther, we could have found an even better point, so we increase α with the rule $\alpha \leftarrow \min\{1.0, 1.2\alpha\}$. If $c^\top x^{(k)} + \sum_{s\in\mathcal{S}} p_s \phi(s, x^{(k)}) \geq c^\top \hat{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \phi(s, \hat{x}^{(k)})$, we did not stabilize enough, and we therefore decrease the stabilization parameter α with the rule $\alpha \leftarrow \max\{0.1, 0.8\alpha\}$. We initialize α to 0.5. Such a procedure cannot be used in the stabilized Benders by batch algorithm as the actual value of the recourse function is required. **Level Bundle** is tested with a level parameter $\lambda = 0.5$ and a stability center tolerance $\kappa = 0.1$ as in (van Ackooij et al., 2017).

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

4.4.3 Numerical results

This section shows the numerical results obtained on the 84 instances of our benchmark. When an algorithm is stopped at its time limit of 12 hours (43 200s), the computing time is denoted $+\infty$, and the ratio to the best time will be denoted $> \frac{43200}{best\ time}$ in the tables, which means that this algorithm is at least this ratio slower than the best algorithm present in the table. All the tables presented in this section show, for each method, the average computing time to solve the three instances of each size, and the time ratio with respect to the best time obtained in this table. Results instance by instance are presented in Appendix D of the supplementary material. We always present the average time on the three instances of each size for each problem, rounded to the second (when computing times are larger than one second).

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

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

The performance profile of a method $m \in \mathcal{M}$ is the cumulative distribution function of its performance ratios computed over a set of problems $\mathcal{P}$. It is defined as $\rho_m(\tau) = \text{card}(\{p \in \mathcal{P} : r_{p,m} \leq \tau\})$

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

Performance of BbB without stabilization

We first present the results of **BbB** without stabilization. We analyze the impact of the batch size, both without (Table 4.3) and with partial cut aggregation (Table 4.4). Each column of Tables 4.3 and 4.4 contains the average time in second to solve the instances and the ratio to the best time. We analyze batch sizes from 1% to 20% of the total number of subproblems

(respectively denoted by **BbB 1%**, **BbB 5%**, **BbB 10%** and **BbB 20%**). The variants with cut aggregation are respectively designated by **BbB 1% CutAggr**, **BbB 5% CutAggr**, **BbB 10% CutAggr** and **BbB 20% CutAggr**.

In order to estimate only the effect of performing an optimality check after solving each batch of subproblems, we compare the Benders by batch algorithm without cut aggregation (**BbB**) to **Classic multicut** and the Benders by batch algorithm with cut aggregation (**BbB CutAggr**) to **Classic CutAggr 1%** and **Classic CutAggr 5%**, where 1% and 5% represent the same partitions of subproblems as the ones used in **BbB 1% CutAggr** and **BbB 5% CutAggr**. **Classic multicut** can be seen as the Benders by batch algorithm without cut aggregation with a batch size equal to the total number of subproblems, **Classic CutAggr 1%** and **Classic CutAggr 5%** can be seen as the equivalent algorithms as the Benders by batch algorithm with partial cut aggregation, in which all the subproblems are solved at each iteration. We also present the results of **Classic monocut** in each table, as a classical alternative to **Classic multicut** in Table 4.3 and as a fully aggregated method in Table 4.4.

Table 4.3: Results for the Benders by batch algorithm without partial cut aggregation, with batch sizes from 1% to 20% of the total of subproblems.

	Classic monocut		Classic multicut		BbB 1%		BbB 5%		BbB 10%		BbB 20%	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	2	3.0	0.75	1.1	2	2.7	0.83	1.3	0.72	1.1	**0.66**	**1.0**
LandS-N5000	11	1.7	9	1.5	13	2.2	8	1.3	7	1.1	**6**	**1.0**
LandS-N10000	22	1.1	29	1.5	38	2.0	25	1.3	21	1.1	**20**	**1.0**
LandS-N20000	**45**	**1.0**	105	2.3	130	2.9	89	2.0	80	1.8	72	1.6
gbd-N1000	2	3.3	0.94	1.4	2	3.6	**0.65**	**1.0**	0.84	1.3	0.96	1.5
gbd-N5000	12	1.9	10	1.7	16	2.5	**6**	**1.0**	7	1.1	8	1.3
gbd-N10000	23	1.2	33	1.7	47	2.5	**19**	**1.0**	22	1.2	25	1.3
gbd-N20000	**48**	**1.0**	121	2.5	96	2.0	61	1.3	71	1.5	87	1.8
ssn-N1000	2408	611.6	7	1.8	6	1.6	**4**	**1.0**	4	1.1	5	1.2
ssn-N5000	13460	590.1	57	2.5	32	1.4	**24**	**1.0**	28	1.2	32	1.4
ssn-N10000	25901	444.1	188	3.2	71	1.2	79	1.3	**59**	**1.0**	79	1.3
ssn-N20000	$+\infty$	>364.8	488	4.1	**145**	**1.2**	274	2.3	624	5.2	2821	24.9
storm-N1000	24	3.7	11	1.7	21	3.2	8	1.3	**6**	**1.0**	8	1.3
storm-N5000	114	2.1	106	1.9	175	3.2	60	1.1	**55**	**1.0**	65	1.2
storm-N10000	224	1.4	496	3.2	492	3.2	**156**	**1.0**	159	1.0	189	1.2
storm-N20000	**458**	**1.0**	2370	5.2	1390	3.0	580	1.3	672	1.5	588	1.3
20term-N1000	577	15.2	757	19.9	**38**	**1.0**	82	2.2	49	1.3	74	1.9
20term-N5000	3506	5.6	24429	38.6	**634**	**1.0**	2101	3.3	1335	2.1	2247	3.6
20term-N10000	6901	3.0	$+\infty$	>19.9	**2270**	**1.0**	10733	4.7	6199	2.7	10413	4.6
20term-N20000	**13687**	**1.3**	$+\infty$	>6.2	20625	1.7	$+\infty$	>4.2	$+\infty$	>4.2	$+\infty$	>4.2
Fleet20-N1000	533	9.1	225	3.9	145	2.5	95	1.7	102	1.7	**74**	**1.2**
Fleet20-N5000	2757	1.5	5330	2.9	2417	1.3	1950	1.0	**1873**	**1.0**	2097	1.1
Fleet20-N10000	**5710**	**1.0**	28933	5.1	9903	1.7	19913	3.4	8537	1.5	21383	3.7
Fleet20-N20000	**11300**	**1.0**	$+\infty$	>4.1	34900	3.1	$+\infty$	>3.8	$+\infty$	>3.9	$+\infty$	>3.9
productLarge-N1000	1947	19.0	186	1.8	270	2.6	123	1.2	105	1.0	**103**	**1.0**
productLarge-N5000	10467	7.6	3497	2.5	3730	2.7	1873	1.4	1483	1.1	**1377**	**1.0**
productLarge-N10000	20200	3.7	15200	2.8	13300	2.5	6893	1.3	5583	1.0	**5397**	**1.0**
productLarge-N20000	43000	1.9	$+\infty$	>2.0	$+\infty$	>1.9	29700	1.3	24733	1.1	**23067**	**1.0**

We first notice in Table 4.3 that **BbB 1%** solves all the instances, except Fleet20-N20000 where it only succeeds to solve one out of three problems, whereas **Classic Multicut** fails to solve optimally four groups of instances. As the algorithm avoids solving many subproblems and adding cuts in the relaxed master program, it overcomes the issue of the time spent in solving subproblems and delays the size growth of the relaxed master program. However, as we still add one cut for each solved subproblem in the Benders by batch algorithm, it still does not scale well when the number of subproblems becomes large. **Classic monocut** outperforms

BbB on large-scale instances such as 20term with 20000 subproblems or Fleet20_3 with 20000 subproblems.

Table 4.4: Results for the Benders by batch algorithm with partial cut aggregation, with batch sizes from 1% to 20% of the total number of subproblems.

	Classic monocut		Classic 1% CutAggr		Classic 5% CutAggr		BbB 1% CutAggr		BbB 5% CutAggr		BbB 10% CutAggr		BbB 20% CutAggr	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	2	2.5	1	1.3	1	1.7	2	2.1	0.88	1.1	**0.78**	**1.0**	0.89	1.1
LandS-N5000	11	2.6	7	1.8	8	2.0	9	2.3	5	1.1	**4**	**1.0**	4	1.1
LandS-N10000	22	2.7	16	2.0	19	2.3	16	2.0	**8**	**1.0**	8	1.0	9	1.2
LandS-N20000	45	2.6	34	1.9	39	2.3	44	2.6	**17**	**1.0**	18	1.0	20	1.2
gbd-N1000	2	3.6	1	2.0	2	2.7	2	2.7	**0.61**	**1.0**	0.78	1.3	0.93	1.5
gbd-N5000	12	3.6	9	2.6	10	3.0	9	2.7	**3**	**1.0**	4	1.1	4	1.3
gbd-N10000	23	3.7	19	3.1	21	3.3	15	2.3	**6**	**1.0**	8	1.3	9	1.5
gbd-N20000	48	3.6	41	3.0	46	3.4	41	3.1	**14**	**1.0**	15	1.1	19	1.4
ssn-N1000	2408	175.8	24	1.8	142	10.5	**14**	**1.0**	61	4.5	134	9.8	242	17.7
ssn-N5000	13460	150.6	399	4.5	1582	17.7	**89**	**1.0**	322	3.6	659	7.4	1322	14.8
ssn-N10000	25901	140.4	1246	6.7	4858	26.1	**185**	**1.0**	707	3.8	1423	7.7	2914	15.8
ssn-N20000	$+\infty$	>98.4	8603	20.0	26122	58.9	**441**	**1.0**	1615	3.7	3386	7.7	6757	15.4
storm-N1000	24	3.8	12	2.0	15	2.4	12	1.9	**6**	**1.0**	7	1.1	9	1.5
storm-N5000	114	3.4	72	2.1	94	2.8	52	1.5	**34**	**1.0**	36	1.1	55	1.6
storm-N10000	224	3.0	164	2.2	198	2.7	110	1.5	**74**	**1.0**	82	1.1	104	1.4
storm-N20000	458	2.9	369	2.3	423	2.6	226	1.4	**163**	**1.0**	169	1.1	238	1.5
20term-N1000	577	39.4	272	18.5	313	21.4	**15**	**1.0**	37	2.5	68	4.6	141	9.6
20term-N5000	3506	50.3	1604	23.2	1945	28.0	**70**	**1.0**	193	2.8	395	5.7	839	12.1
20term-N10000	6901	53.2	3364	26.0	4840	37.4	**130**	**1.0**	402	3.1	898	6.9	1978	15.3
20term-N20000	13687	49.1	7032	25.2	16287	57.3	**280**	**1.0**	914	3.3	2051	7.3	18312	65.2
Fleet20-N1000	533	18.9	125	4.4	222	7.9	**28**	**1.0**	42	1.5	74	2.6	131	4.7
Fleet20-N5000	2757	25.7	903	8.4	1530	14.3	**107**	**1.0**	211	2.0	358	3.3	649	6.1
Fleet20-N10000	5710	26.9	2000	9.4	3460	16.3	**212**	**1.0**	440	2.1	721	3.4	1310	6.2
Fleet20-N20000	11300	27.0	5053	12.1	7860	18.8	**419**	**1.0**	876	2.1	1520	3.6	2777	6.6
product-N1000	1947	20.0	190	2.0	431	4.4	**98**	**1.0**	141	1.5	253	2.6	505	5.2
product-N5000	10467	28.9	1523	4.2	3323	9.2	**362**	**1.0**	773	2.1	1567	4.3	2873	7.9
product-N10000	20200	25.0	3827	4.8	7757	9.7	**823**	**1.0**	1523	1.9	3053	3.8	5530	6.9
product-N20000	43000	25.7	9963	6.0	19367	11.6	**1693**	**1.0**	3367	2.0	6320	3.8	12500	7.5

Table 4.4 shows that when partial cut aggregation is used, all the presented methods clearly outperform **Classic monocut**. As we aggregate the cuts over each batch, the size of the relaxed master program remains reasonable, and as the cuts are only computed on samples of subproblems, the algorithms avoid many symmetries due to the sum of the cuts over the subproblems. The table shows also that the best batch sizes are 1% and 5% (respectively **BbB 1% CutAggr** and **BbB 5% CutAggr**), except for two small instances. The two methods can be up to 25 times faster than **Classic CutAggr 1%** and more than 58 times faster than **Classic CutAggr 5%**.

The better performance of the Benders by batch algorithm with partial cut aggregation can be explained by Figure 4.3. We see that at almost all the iterations, **BbB CutAggr** solves only one batch of subproblems to show that the current first-stage candidate cannot be proven optimal, and to separate the solution to the relaxed master program. It follows that **BbB CutAggr 1%** needs to solve less subproblems than **Classic CutAggr 1%** to converge. For a storm instance with 20000 subproblems, **BbB CutAggr 1%** needs to solve twice less subproblems than **Classic CutAggr 1%**, for a 20term instance with 20000 subproblems, **BbB CutAggr 1%** needs to solve 23 times less subproblems than **Classic CutAggr** to converge. Although **Classic CutAggr 1%** evaluates almost three times less first-stage solutions for the 20term instance (and more than 5 times less for the storm instance), it takes ultimately more time to converge than the Benders by batch algorithm: 1006 seconds for **Classic CutAggr 1%** compared to

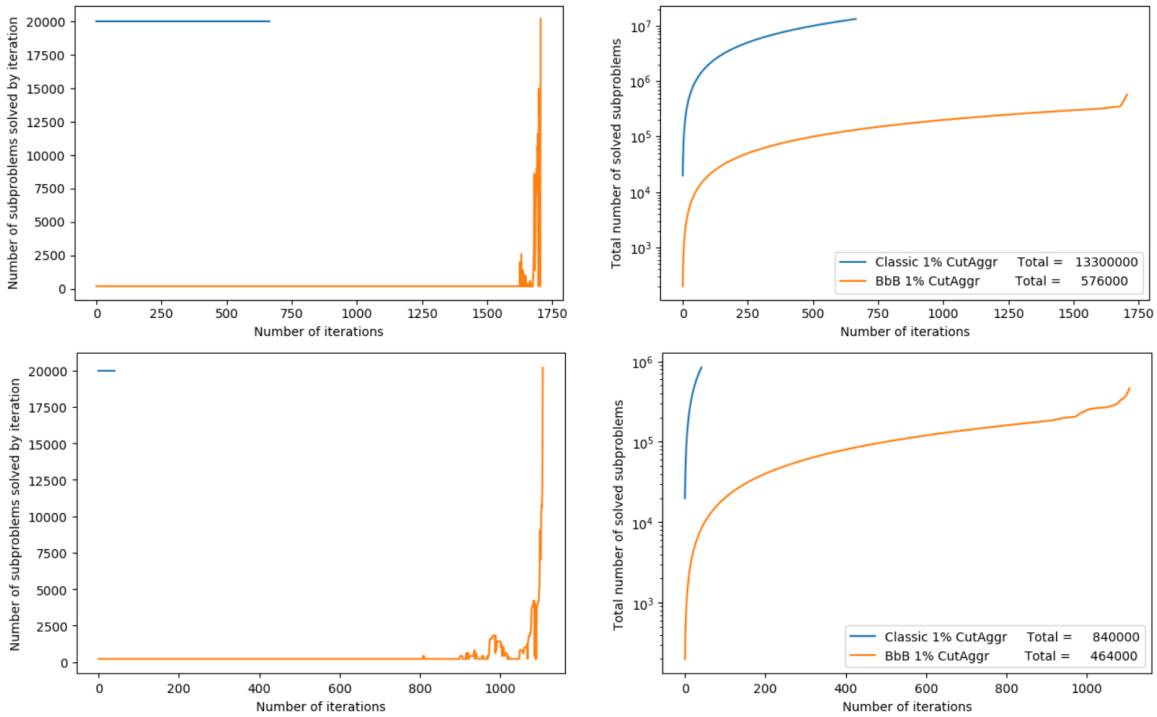


Figure 4.3: Number of subproblems solved at each iteration by **BbB CutAggr 1%** and **Classic CutAggr 1%** (left plots), with their cumulative evolution (right plots) for a 20term instance with 20000 subproblems (top plots) and a storm instance with 20000 subproblems (bottom plots). The "Total" in the legend shows the total number of subproblems evaluated during the algorithms.

261 seconds for **BbB 1% CutAggr** to solve the 20term instance and 370 seconds for **Classic CutAggr 1%** compared to 216 seconds for **BbB 1% CutAggr** to solve the storm instance.

Impact of the stabilization on BbB

We now present the results obtained when the two stabilization schemes presented in §4.3.3 are applied to the most competitive versions of Bbb (batch sizes of 1% and 5%, and with partial cut aggregation). Figures 4.4 and 4.5 show the performance profiles of **BbB CutAggr** with and without stabilization. We present the results with basic stabilization for $\alpha \in \{0.1, 0.5, 0.9\}$ and with solution memory stabilization for $\alpha \in \{0.1, 0.5, 0.9\}$ and $\beta \in \{0.1, 0.5, 0.9\}$. Each stabilized method is denoted by **BbB 1% CutAggr** or **BbB 5% CutAggr** followed by the values for the parameters.

Figure 4.4 shows that the proposed stabilization schemes accelerate **BbB 1% CutAggr**, and can be up to 70% faster than the unstabilized algorithm. Four stabilizations are more efficient on the tested instances and give similar results, namely the basic stabilization with $\alpha = 0.5$, and the solution memory stabilization with $(\alpha, \beta) \in \{(0.5, 0.1), (0.5, 0.5), (0.9, 0.5)\}$.

Figure 4.5 shows similar results for **BbB 5% CutAggr**. The same four methods are the most efficient and equivalent to each other. The algorithm with a solution memory stabilization parameterized by $(\alpha, \beta) = (0.1, 0.9)$ is less efficient than **BbB 5% CutAggr**. In this case, a small step size ($\alpha = 0.1$) and a high memory parameter ($\beta = 0.9$) slow down the convergence. For all the other cases, the use of a primal stabilization scheme accelerates the algorithm.

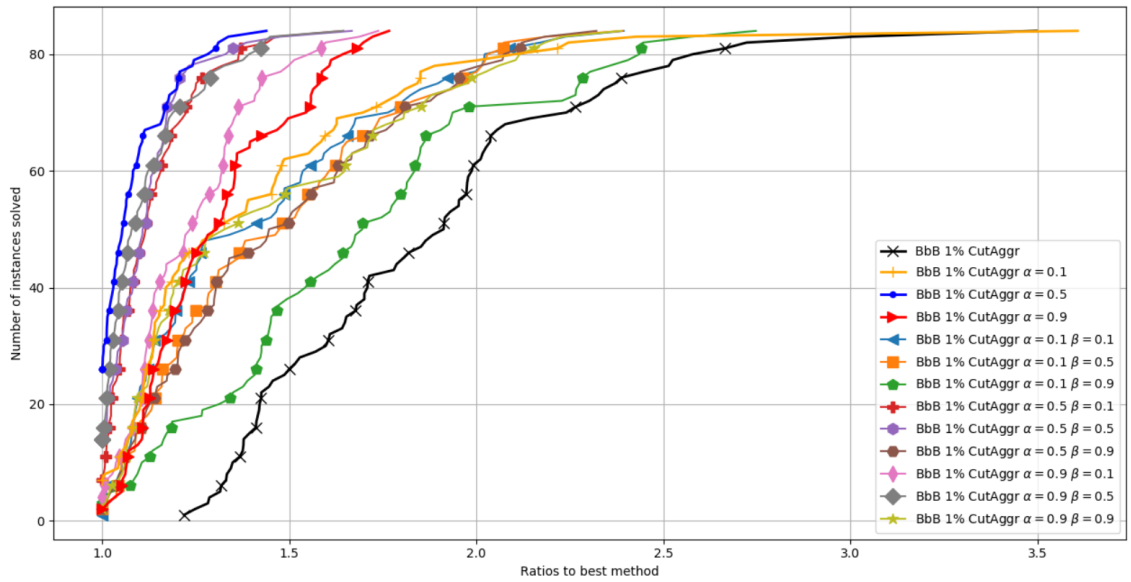


Figure 4.4: Performance profiles of the stabilized Benders by batch algorithm with batch size of 1% and cut aggregation.

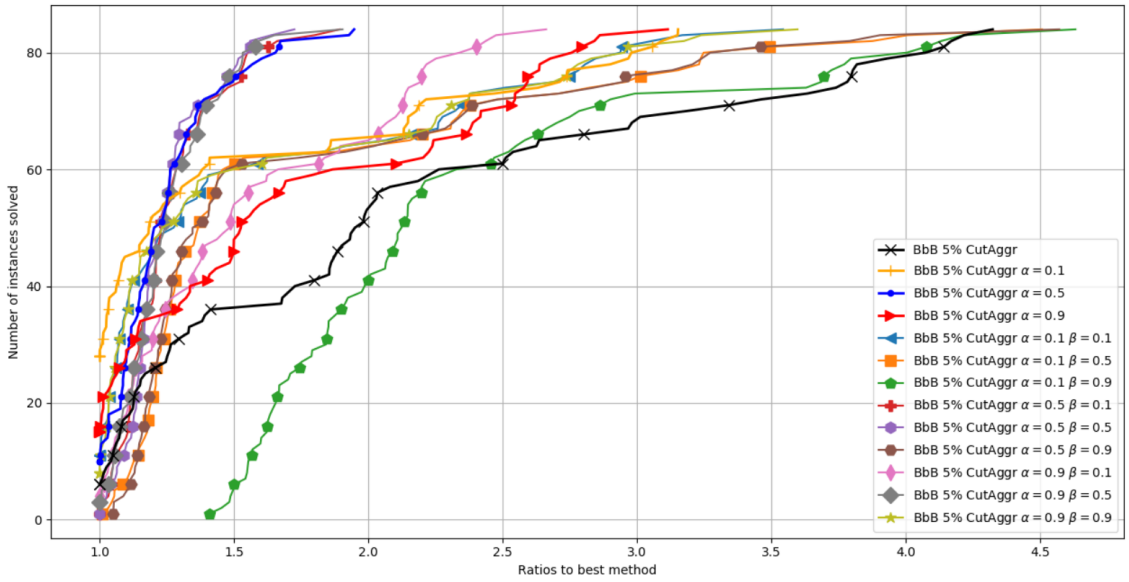


Figure 4.5: Performance profiles of the stabilized Benders by batch algorithm with batch size of 5% and cut aggregation.

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

Comparison with state-of-the-art methods

We now compare the stabilized Benders by batch algorithm to classical methods of the literature. We show in Table 4.5 the times and ratios of **CPLEX Barrier** and all the stabilized methods of our benchmark, **In-out monocut**, **In-out multicut**, **Level bundle**, **In-out CutAggr 1%** and

Table 4.5: Final results, the best stabilized Benders by batch algorithm compared to all stabilized benchmark methods.

	CPLEX Barrier		Level Bundle		In-out multicut		In-out monocut		In-out 1% CutAggr		In-out 5% CutAggr		BbB 1% CutAggr $\alpha = 0.5$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000	**0.07**	**1.0**	1	17.3	0.89	12.4	1	20.0	0.71	9.7	0.98	13.4	0.96	13.2
LandS-N5000	**1**	**1.0**	7	9.0	8	10.5	9	10.5	5	6.0	6	7.2	5	6.7
LandS-N10000	**1**	**1.0**	14	14.0	24	23.6	16	15.6	10	9.7	11	11.1	9	9.0
LandS-N20000	**5**	**1.0**	27	6.8	62	16.5	41	10.4	22	5.6	22	5.5	21	5.4
gbd-N1000	**0.04**	**1.0**	2	61.2	1	36.6	2	58.8	1	33.6	2	44.8	0.88	25.6
gbd-N5000	**0.17**	**1.0**	10	60.1	10	60.9	10	64.0	7	41.8	8	47.1	4	24.8
gbd-N10000	**0.35**	**1.0**	24	69.5	23	67.5	21	61.7	16	45.7	17	50.3	8	22.2
gbd-N20000	**0.91**	**1.0**	44	48.8	82	89.8	54	60.6	30	34.3	34	39.1	17	18.5
ssn-N1000	32	6.0	90	17.1	**6**	**1.0**	137	27.3	10	1.8	19	3.6	8	1.5
ssn-N5000	310	10.6	657	22.2	**31**	**1.0**	795	27.4	70	2.4	133	4.5	47	1.6
ssn-N10000	1223	20.3	1501	25.2	**63**	**1.0**	1464	23.3	171	2.9	312	5.2	91	1.5
ssn-N20000	2619	13.7	3109	16.3	243	1.3	2861	15.2	400	2.1	736	3.9	**191**	**1.0**
storm-N1000	41	5.8	15	2.1	9	1.3	14	2.1	8	1.1	9	1.4	**7**	**1.0**
storm-N5000	316	9.7	76	2.3	41	1.3	62	1.9	49	1.5	52	1.6	**33**	**1.0**
storm-N10000	764	11.8	145	2.3	125	1.9	201	3.1	99	1.5	110	1.7	**65**	**1.0**
storm-N20000	2390	17.4	288	2.1	573	4.2	252	1.8	211	1.5	232	1.7	**137**	**1.0**
20term-N1000	14	1.3	217	20.9	36	3.5	114	10.8	27	2.6	44	4.3	**10**	**1.0**
20term-N5000	82	1.7	1044	21.2	482	9.7	681	13.8	197	4.0	269	5.5	**50**	**1.0**
20term-N10000	199	2.0	2450	24.4	2805	27.9	1190	11.8	474	4.7	593	5.9	**100**	**1.0**
20term-N20000	455	2.3	4843	24.7	10992	56.0	1754	8.9	1010	5.1	1371	7.0	**197**	**1.0**
Fleet20-N1000	23	1.3	107	6.2	50	2.9	93	5.4	26	1.5	41	2.4	**17**	**1.0**
Fleet20-N5000	269	3.6	500	6.7	719	9.6	473	6.3	184	2.4	250	3.3	**75**	**1.0**
Fleet20-N10000	809	5.5	1004	6.9	3747	25.6	1029	7.0	435	3.0	590	4.0	**146**	**1.0**
Fleet20-N20000	2446	7.9	2730	8.8	17000	54.7	1780	5.8	1018	3.3	1313	4.2	**310**	**1.0**
product-N1000	179	2.3	625	8.2	81	1.1	513	6.7	113	1.5	183	2.4	**76**	**1.0**
product-N5000	2121	6.7	3200	10.3	1127	3.6	2690	8.7	787	2.5	1380	4.4	**312**	**1.0**
product-N10000	4397	8.0	7173	13.0	5357	9.8	5730	10.4	1970	3.6	3133	5.7	**552**	**1.0**
product-N20000	15463	13.6	14300	12.5	$+\infty$	>40.5	12333	10.8	4887	4.3	7983	7.0	**1140**	**1.0**

In-out CutAggr 5% with the best performing stabilized Benders by batch **BbB 1% CutAggr with** $\alpha = 0.5$. We first observe that, on the small instances LandS and gbd, **CPLEX Barrier** converges faster than all the other methods. As those instances have very few variables both in first and second stages, they remain small even with 20000 subproblems, and are solved very efficiently by **CPLEX Barrier**. However, we can notice that even for these small instances, **BbB 1% CutAggr** $\alpha = 0.5$ is the best method among all the cutting planes algorithms. Table 4.5 shows clearly that the stabilized Benders by batch algorithm outperforms all the other methods on the large instances, and can be up to more than 25 times faster than **Level Bundle** or 15 times faster than **In-out monocut**. We also show that, even if **In-out CutAggr** outperforms other classical stabilized methods from the literature, the stabilized Benders by batch algorithm can be up to 5 times faster. This shows that, firstly, using a static cut aggregation combined with primal stabilization allows to speed up classical methods used to benchmark algorithms from the literature, and secondly, that not solving systematically all the subproblems allows to further improve the computing times on the test instances. Indeed, we show in Figure 4.6 that **BbB 1% CutAggr** $\alpha = 0.5$ needs to solve way less subproblems than other methods to converge, and that the time spent in solving the subproblems represent almost all the computing times in all presented methods.

Figure 4.7 shows the evolution of the relative gap between the lower bound and the optimal value, of four different algorithms, on four different instances, according to the time. We see that adding only a few cuts at each iterations allows the lower bound to converge faster to the optimal value to the problem. Moreover, we observe that, on three of the four presented instances, **BbB**

1% CutAggr $\alpha = 0.5$ reaches a relative gap of 10^{-6} while all the other algorithms still have a large relative gap (e.g. 10^0 on ssn or 10^{-1} on Fleet).

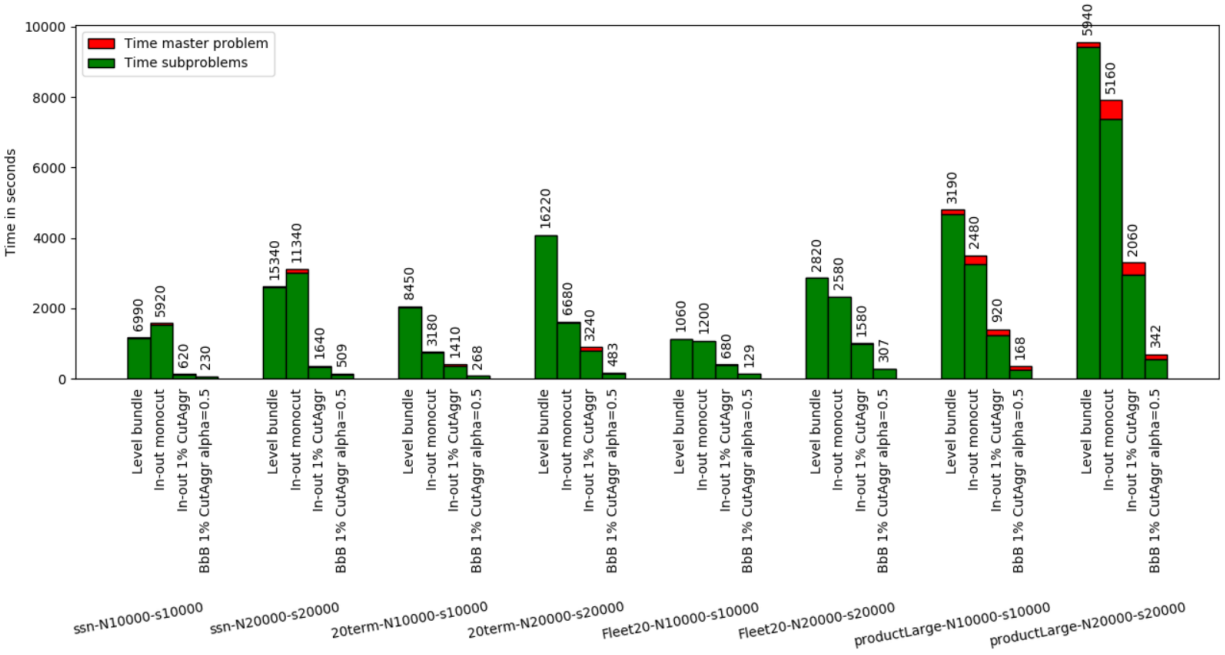


Figure 4.6: Time spent in solving the master program and the subproblems, for 8 different instances, solved by Level bundle, In-out monocut, In-out CutAggr 1% and BbB 1% CutAggr $\alpha = 0.5$. The total number of solved subproblems is written vertically on the top of each bar.

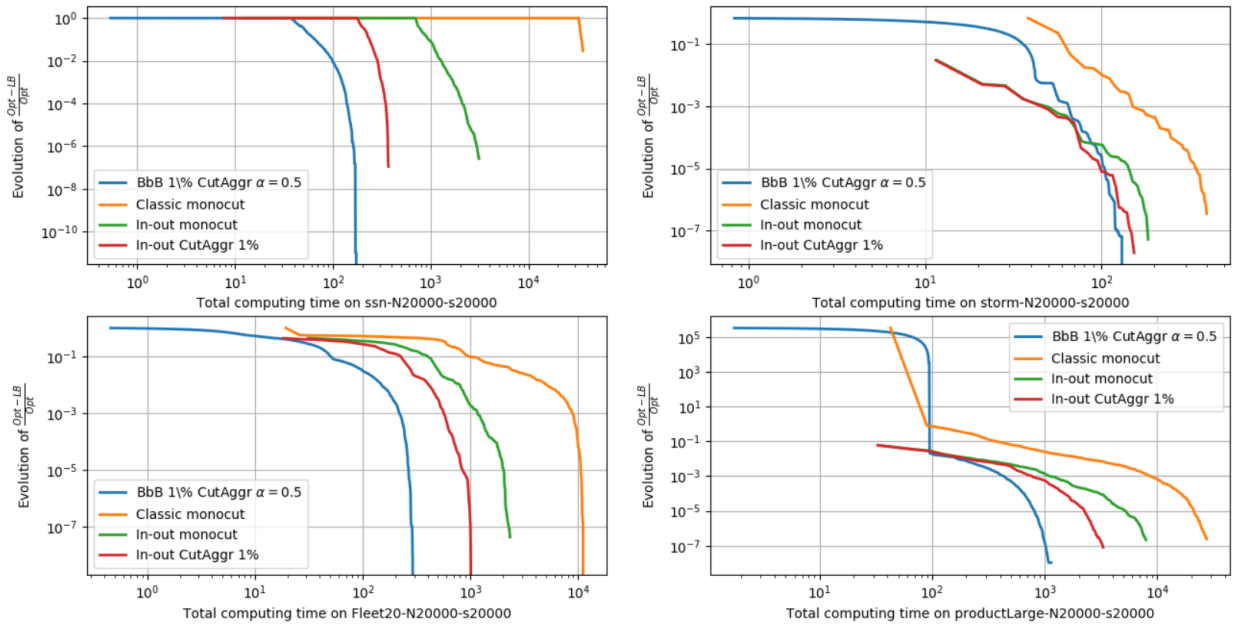


Figure 4.7: Evolution of the relative gap between the lower bound and the optimal value as a function of time, on a four different instances with 20000 subproblems

Sensitivity of BbB to parameters

We finally present the impact on the computing time of two parameters of the Benders by batch algorithm, the optimality gap and the initial order of the subproblems.

We first analyze the impact of the optimality gap on the convergence of the algorithm. The choice of a different optimality gap ε in the Benders by batch algorithm might have an impact on the number of batches that would be solved at each iteration. With a larger optimality gap, the algorithm tends to solve more batches at each iteration, and to add more cuts. As this might have an impact on the first-stage iterates, and then on the computing times, we show on Figure 4.8 the cumulative distribution of the computing times to solve our 84 instances with **BbB 1% CutAggr and** $\alpha = 0.5$ with four different optimality gaps $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. The figure shows that different optimality gaps have a negligible impact on the computing times on most instances. A smaller optimality gap induces larger computing times on the largest instances of our test set, but this would also be the case with other classical algorithms.

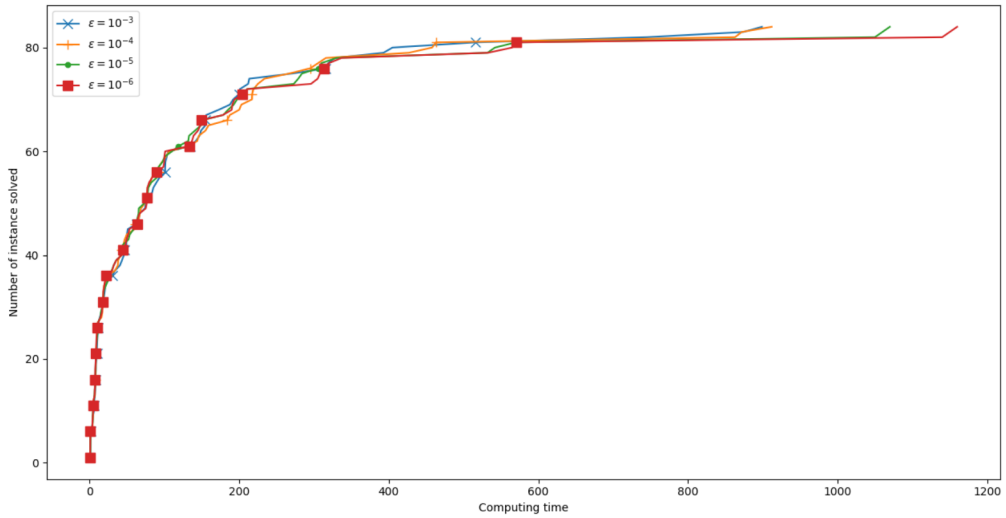


Figure 4.8: Cumulative distribution of the computing times on our 84 instances, for BbB with cut aggregation and base stabilization with $\alpha = 0.5$, and with optimality gaps in $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-5}\}$

We finally ran several experiments testing different initial orders to assess the sensitivity of our method to this choice. We ran **BbB** 1% **CutAggr** $\alpha = 0.5$, for 500 different initial orders, on one instance with 5000 subproblems and one with 10000 subproblems for each tested problem. We report in Table 4.6 the minimum and maximum times observed, the median, and the first and ninth decile on computing times. We observe that the initial order has usually a limited impact on the efficiency of our algorithm. We also remark that the stabilized Benders by batch algorithm present lower computing times than **In-out CutAggr 1%**, the best performing method used as comparison in the numerical results, even for the maximum time observed. Although the impact is in general limited, we observe that the initial order can have an impact on the computing time for some instances, such as LandS or gbd. However, the computing times observed are almost always smaller than the computing times of **In-out CutAggr 1%**, the best performing method presented in the paper.

Table 4.6: Computing times for **BbB 1% CutAggr** $\alpha = 0.5$ on 500 different initial orders of the subproblems

	Min Time		10%		50%		90%		Max Time		In-out CutAggr 1%	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N5000	**4.1**	**1.0**	4.5	1.1	5.3	1.3	6.2	1.5	7.3	1.8	5.0	1.2
LandS-N10000	**8.3**	**1.0**	9.2	1.1	10.2	1.2	11.9	1.4	15.6	1.9	10.0	1.2
gbd-N5000	**3.1**	**1.0**	3.5	1.1	4.1	1.3	5.0	1.6	7.1	2.3	7.0	2.3
gbd-N10000	**6.0**	**1.0**	7.2	1.2	8.3	1.4	10.3	1.7	14.0	2.3	16.0	2.7
ssn-N5000	**40.2**	**1.0**	44.3	1.1	46.8	1.2	49.8	1.2	54.1	1.3	70.0	1.7
ssn-N10000	**82.5**	**1.0**	87.3	1.1	92.5	1.1	102.0	1.2	122.4	1.5	171.0	2.1
storm-N5000	**28.0**	**1.0**	29.8	1.1	31.4	1.1	34.5	1.2	43.5	1.6	49.0	1.8
storm-N10000	**58.0**	**1.0**	60.5	1.0	64.2	1.1	69.7	1.2	83.2	1.4	99.0	1.7
20term-N5000	**43.5**	**1.0**	47.8	1.1	54.1	1.2	61.6	1.4	77.2	1.8	197.0	4.5
20term-N10000	**82.0**	**1.0**	91.5	1.1	103.2	1.3	115.0	1.4	136.2	1.7	474.0	5.8
Fleet20-N5000	**72.5**	**1.0**	74.7	1.0	76.6	1.1	78.7	1.1	83.3	1.1	184.0	2.5
Fleet20-N10000	**142.0**	**1.0**	148.0	1.0	152.0	1.1	157.0	1.1	166.0	1.2	435.0	3.1
productLarge-N5000	**268.0**	**1.0**	279.0	1.0	292.0	1.1	315.0	1.2	355.0	1.3	787.0	2.9
productLarge-N10000	**528.0**	**1.0**	553.0	1.0	573.0	1.1	603.0	1.1	679.0	1.3	1970.0	3.7

4.5 Conclusion

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

To avoid strong oscillations of the first-stage variables, we also introduced a stabilized version of the algorithm. This algorithm is based on a primal stabilization scheme responsible for generating the points at which the subproblems are solved. We presented a sufficient condition for a primal stabilization scheme that ensures the convergence of the Benders by batch algorithm and proposed two schemes satisfying it. The stabilized Benders by batch algorithm can be up to 25 times faster than the level bundle method, or 5 times faster than Benders decomposition with in-out stabilization and static partial cut aggregation of (Trukhanov et al., 2010).

Applying dual stabilization (Magnanti and Wong, 1981; Sherali and Lunday, 2013) to the Benders by batch algorithm is straightforward and could improve the results. The algorithm can be parallelized and may benefit from effective parallelized methods, such as the asynchronous method of Linderoth and Wright (2003). The use of more advanced cut aggregation strategies is also a path worth exploring. Finally, an interesting perspective is to adapt the Benders by batch algorithm to solve mixed-integer master programs within a Branch&Cut framework.

Chapter 5

A reliability constraint for a stochastic expansion planning problem

Bilevel formulation and Benders-based heuristic solution method

5.1 Introduction

We propose in this chapter to model a stochastic expansion planning problem taking into account a reliability constraint imposed to RTE by French legislation. In prospective studies conducted at RTE, the cost associated to an unsatisfied demand is set to 20000€/MWh. Since the investment costs are high, optimal solutions to the stochastic expansion planning problem have limited production and transmission capacities, and present a large number of unsatisfied demands. French legislation (Article D141-12-6 from the French Energy Code) imposes that there are less than three hours of power system failure in expectation when RTE operates the system. RTE applies a similar rule in their prospective studies. In investment solutions presented in their prospective studies, they require that there are less than three hours at which the demand is not fully satisfied in expectation over random scenarios. This constraint is currently not taken into account in the expansion planning models. Solutions have to be modified by hand after the optimization step in order to satisfy this criterion. We present a formulation of a stochastic expansion planning problem taking this constraint into account, and a solution method to find feasible solutions.

The stochastic expansion planning problem we consider here can be formulated as follows. Considering a network, the problem consists in investing in production capacities on the nodes or in transmission capacities on the arcs. Many parameters can be random such as the electricity demands or the weather forecasts, and randomness is modeled with a finite set of scenarios. For each scenario and each investment solution, we model an operational problem. In this problem, there is a finite and discrete time horizon, and an electricity demand at each node of the network and each time step. Given a set of linear operational constraints, the operational problem finds how the electricity is produced and routed over the network to satisfy the demands at every time

step. The global objective of the problem is to minimize the sum of the investment costs and the expectation of the operational costs over the scenarios. The reliability constraint we add to this problem limits, in expectation over the scenarios, the number of nodes and time steps at which the demand is not fully satisfied in a solution. We formulate this problem as a bilevel program and propose a heuristic based on a Benders decomposition algorithm to obtain feasible solutions.

We describe hereafter the problem without the reliability constraint. Let $G = (\mathcal{N}, \mathcal{A})$ be a graph, $\mathcal{S}$ a finite set of scenarios modeling the uncertainty and $\mathcal{T}$ a finite set of time steps. We denote by $x \in \mathbb{R}^{n_1}$ the vector of first-stage variables modeling investment decisions. These decisions are either to invest on transmission capacities on the arcs of the network, or on production capacities on the nodes of the network. Then, for every scenario $s \in \mathcal{S}$, a linear problem models a minimum cost operational planning to satisfy stochastic demands. The stochastic linear expansion planning problem can be modeled as follows:

$$\begin{cases} \min\ c^\top x + \sum_{s \in \mathcal{S}} p_s \phi(x, s) \\ s.t.\ x \in \mathcal{X} \end{cases} \tag{5.1}$$

where $\mathcal{X} \subset \mathbb{R}^{n_1}$ is a polyhedral set containing constraints on the investment decisions. First-stage variables are continuous. In every scenario $s \in \mathcal{S}$, and at each node and time step $(n, t) \in \mathcal{N} \times \mathcal{T}$, we introduce a positive variable $u_{s,n,t}$ that represents the amount of demand that is not satisfied at node n and time step t. For every $s \in \mathcal{S}$, we denote by u_s the vector of variables $u_{s,n,t}$ of size $\text{card}(\mathcal{N} \times \mathcal{T})$. These variables u_s are penalized by a constant cost $M \in \mathbb{R}_+$ in the objective function. We denote by e the vector of $1's$ of size $\text{card}(\mathcal{N} \times \mathcal{T})$. All other operational variables such as production variables or flow variables are included in variables $y_s \in \mathbb{R}^{n_2}$. For every $x \in \mathcal{X}$ and every $s \in \mathcal{S}$:

$$\phi(x, s) = \begin{cases} \min\ g_s^\top y_s + M \mathrm{e}^\top u_s \\ s.t.\ W_s y_s + Q u_s \geq d_s - T_s x \\ \qquad y_s \in \mathbb{R}^{n_2}_+,\ u_s \in \mathbb{R}^{\text{card}(\mathcal{N} \times \mathcal{T})}_+ \end{cases} \tag{5.2}$$

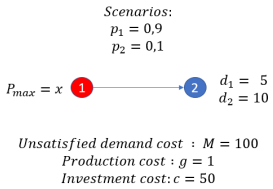
Vector $c \in \mathbb{R}^{n_1}$ represents the investment costs. The operational costs are modeled with vector $g_s \in \mathbb{R}^{n_2}$. the value $m \in \mathbb{N}$ is the number of operational constraints. Matrix $W_s \in \mathbb{R}^{m \times n_2}$ and matrix $T_s \in \mathbb{R}^{m \times n_1}$. The probability associated to scenario $s \in \mathcal{S}$ is $p_s \in (0, 1]$. Matrix $Q \in \{0, 1\}^{m \times \text{card}(\mathcal{N} \times \mathcal{T})}$.

5.1.1 On the necessity of a reliability constraint

Generally, in optimization problems, every single demand has to be satisfied, and variables u_s are associated to a very high cost in order to ensure feasibility of the second-stage for any first-stage decision $x \in \mathcal{X}$, as stated in (Bodur and Luedtke, 2022) for example. In the problem we study, the value of M is an economic cost. This cost is always high enough so that when it is possible, it is cheaper to produce and route electricity to a demand rather than paying this cost. But depending on the investment costs, it may be cheaper to choose an investment solution for which all the demands are not satisfied, and to pay the penalty cost associated to these demands. Then,

optimal solutions to problem (5.1) can present a large number of unsatisfied demands. We show this situation in Example 1.

Example 1. *We consider the example presented in the following figure, with a network with two nodes, one time step and two scenarios.*



It is possible to invest in production capacity at node 1*, and the demands are located at node* 2*. The investment cost is equal to* 50*, the production cost equal to* 1 *and the cost associated with the quantity of unsatisfied demand equal to* 100*. The arcs have unlimited capacity and the transmission cost is* 0*, which means that no cost has to be paid when sending a flow along an arc. This example is associated with the following formulation:*

$$\begin{cases} \min\ 50x + 0.9(y_1 + 100u_1) + 0.1(y_2 + 100u_2) \\ s.t. : y_s + u_s \geq d_s,\ \ s \in \{1,2\} \\ \qquad y_s \leq x,\ \ s \in \{1,2\} \\ \qquad y_1, y_2 \in \mathbb{R}_+ \\ \qquad u_1, u_2 \in \mathbb{R}_+ \\ \qquad x \in \mathbb{R}_+ \end{cases}$$

where y_s models the production in scenario $s \in \{1,2\}$, and u_s counts the unsatisfied electricity demand in scenario $s \in \{1,2\}$. We show in the following table the distribution of the cost for different investment solutions. The unsatisfied demands are denoted by (u_1, u_2).

Value of x	*Investment cost*	*Operational Cost*	*Total cost*	*Unsatisfied demand*
$x = 0$	0	550	550	$(5, 10)$
$x = 5$	250	55	305	$(0, 5)$
$x = 10$	500	5.5	505.5	$(0, 0)$

The optimal solution is associated with investment solution $x = 5$. We observe that in this solution, the demand is not fully satisfied in scenario 2. Investing more in order to satisfy all the demands in every scenario is more expensive, and is not economically viable as the probability associated with this demand is low (0.1 in this example).

The example shows that if a given level of reliability of the system must be ensured, it is necessary to explicitly add a constraint to the problem. The reliability constraint that we consider here limits, in expectation over the scenarios, the number of nodes and time steps at which the demand is not fully satisfied in an optimal solution.

The question of the reliability of the transmission system have been widely studied in the literature. There is in general no clear formulation of what is a reliable system in the literature

(Leite da Silva et al., 2010; Lei et al., 2018). Several methods consider reliability by penalizating the quantity of unsatisfied demands in the objective function (Contreras and Wu, 2000; Gil and da Silva, 2001). In (Choi et al., 2005), the reliability of the system is checked by a Monte-Carlo simulation procedure. Braga and Saraiva (2005) consider the reliability as an other objective, and develop a method to compute different pareto-optimal solutions. Alizadeh and Jadid (2011) consider a general reliability criterion, but does not include it in the formulation. A heuristic method is used to converge to feasible solutions. It imposes sequentially demands in different scenarios to be satisfied until the required reliability criterion is satisfied.

5.1.2 Contribution and chapter organization

In this chapter, we model and analyze the formulation of a stochastic expansion planning problem with the reliability constraint described above. We show that this problem can be naturally formulated as a bilevel program with the same objective function in both levels. Despite this particular structure, we show that this problem remains a bilevel program. We analyze it high-point relaxation, and prove that it does not satisfy the time-consistency principle, a classical property of two-stage stochastic programs. We finally show that, in a optimal solution to the high-point relaxation, the ratio between the cost of the optimal recourse variables and the optimal recourse value associated with the first-stage solution is not bounded.

Solving instances with reasonable size is out of reach of exact classical solution methods. We then propose a heuristic method to produce good quality solutions based on a Benders decomposition algorithm and a binary search on the investment costs. We show that, under some assumptions on the stochastic expansion planning problem, there exists a feasible solution for the bilevel stochastic expansion planning problem with the reliability constraint, and that the proposed heuristic converges to a feasible solution in a finite number of iterations. Finally, we show on randomly generated instances with up to 6 million variables and 8 million constraints that the proposed method allows to find feasible solutions in a reasonable time, whereas the SOS-1 method to solve the KKT reformulation of bilevel programs (Fortuny-Amat and McCarl, 1981; Siddiqui and Gabriel, 2013) is not able to find any feasible solution in 6 hours, even on small instances.

The chapter is organized as follows. In Section 5.2, we present the reliability constraint that we want to add to the stochastic expansion planning problem. We also show that a single-level linear formulation, obtained by adding the reliability constraint in the extensive formulation of the stochastic expansion planning problem, would not be a two-stage stochastic program anymore. In Section 5.3, we present a heuristic based on Benders decomposition and a binary search on investment costs to find feasible solutions in a reasonable time. In Section 5.4, we present the result of our heuristic and we compare them to the results obtained by solving the KKT reformulation of the bilevel program with the SOS-1 method. We finally conclude and present several perspectives to this work in Section 5.5.

5.2 Bilevel formulation of the reliable stochastic expansion planning problem

In this section we present the bilevel formulation of the stochastic expansion planning problem with the reliability constraint in Subsection 5.2.1 and analyze the high-point relaxation in Subsection 5.2.2.

5.2.1 Reliability constraint and bilevel formulation

We want to add in formulation (5.1) a constraint which limits, in expectation over the scenarios, the number of nodes and time steps at which the demand is not fully satisfied. We call this constraint the *reliability constraint*. In order to formalize this constraint, we introduce the following notations.

For every $x \in \mathcal{X}$, the set of feasible first-stage solutions in program (5.1), and for every scenario $s \in \mathcal{S}$, we define $\Omega_s(x) = \{(y,u) \in \mathbb{R}^{n_2} \times \mathbb{R}_+^{\text{card}(\mathcal{N}\times\mathcal{T})} : \ W_s y + Qu + T_s x \geq d_s\}$ the set of feasible recourse solutions and $\mathcal{R}_s(x)$, the set of optimal recourse solutions as $\mathcal{R}_s(x) = \arg\min\{g_s^\top y + Me^\top u : \ (y,u) \in \Omega_s(x)\}$. As, for every $x \in \mathcal{X}$, the value of the recourse function is defined as the optimal value to the recourse problem, the recourse solutions are restricted by the formulation to belong to $\mathcal{R}_s(x)$ for every $s \in \mathcal{S}$.

Let $\varepsilon > 0$ be a given threshold. We say that there is an unsatisfied demand at node $n \in \mathcal{N}$ and time step $t \in \mathcal{T}$ for a given scenario $s \in \mathcal{S}$ if $u_{s,n,t} > \varepsilon$. Such a threshold ε is required to count with binary variables the nodes and times steps at which demands are not fully satisfied. For every $x \in \mathcal{X}$, $s \in \mathcal{S}$, and for every $(y_s, u_s) \in \mathcal{R}_s(x)$, we denote by $U_\varepsilon(x, y_s, u_s) = card(\{(n,t) \in \mathcal{N} \times \mathcal{T} : u_{s,n,t} > \varepsilon\})$ the number of unsatisfied demands in scenario s. We define the *expected number of unsatisfied demands* in solution $(x, (y_s, u_s)_{s\in\mathcal{S}})$ as $\sum_{s\in\mathcal{S}} p_s U_\varepsilon(x, y_s, u_s)$. Let $\alpha \geq 0$ be the parameter defining the maximal value of the expected number of unsatisfied demands that we authorize in a solution. We say that a solution $(x, (y_s, u_s)_{s\in\mathcal{S}})$ satisfies the reliability constraint if:

$$\begin{cases} (y_s, u_s) \in \mathcal{R}_s(x), \ \forall s \in \mathcal{S} \\ \sum_{s\in\mathcal{S}} p_s U_\varepsilon(x, y_s, u_s) \leq \alpha \end{cases}$$

This leads to the following bilevel formulation:

$$\begin{cases} \min \ c^\top x + \sum_{s\in\mathcal{S}} p_s \phi(x,s) \\ s.t. \ \sum_{s\in\mathcal{S}} p_s U_\varepsilon(x, y_s, u_s) \leq \alpha \\ \qquad (y_s, u_s) \in \mathcal{R}_s(x), \ \forall s \in \mathcal{S} \\ \qquad x \in \mathcal{X} \end{cases} \tag{5.3}$$

We call problem (5.3) the *reliable stochastic expansion planning problem* hereafter.

We now present an equivalent formulation to problem (5.3) highlighting its two-stage stochastic structure. For every scenario $s \in \mathcal{S}$, we introduce binary variables $\delta_s \in \{0,1\}^{\text{card}(\mathcal{N}\times\mathcal{T})}$ such that for every node and time step $(n,t) \in \mathcal{N} \times \mathcal{T}$, variable $\delta_{s,n,t}$ equals 1 if the quantity of unsatisfied

demand exceeds a given threshold ε and 0 otherwise. We define mapping $g_s : \mathbb{R}^{n_1} \mapsto \mathbb{R}$ as the following parametric problem:

$$g_s(x) = \begin{cases} \min & \mathrm{e}^\top \delta_s \\ s.t. & (y_s, u_s) \in \mathcal{R}_s(x) \\ & C\delta_s \geq u_s - \varepsilon \mathrm{e} & (5.4) \\ & \delta_s \leq (1/\varepsilon) u_s & (5.5) \\ & \delta_s \in \{0,1\}^{\mathrm{card}(\mathcal{N}\times\mathcal{T})} & (5.6) \end{cases}$$

where e is the vector of $1's$ of size $\mathrm{card}(\mathcal{N} \times \mathcal{T})$ and C a large enough positive real number (an upper bound on variables u).

Proposition 8. *For every* $x \in \mathcal{X}$, $g_s(x) = \min\Big\{U_\varepsilon(x, y_s, u_s) : (y_s, u_s) \in \mathcal{R}_s(x)\Big\}$

Proof. For any $(n,t) \in \mathcal{N} \times \mathcal{T}$, if $u_{s,n,t} < \varepsilon$ then, constraint (5.4) is inactive, and by constraint (5.5), $\delta_{s,n,t} \leq (1/\varepsilon) u_{s,n,t} < 1$. As $\delta_{s,n,t}$ is a binary variable, $\delta_{s,n,t} = 0$.

If $u_{s,n,t} > \varepsilon$ then constraint (5.5) is inactive, and by constraint (5.4), $\delta_{s,n,t} \geq u_{s,n,t} - \varepsilon > 0$. As $\delta_{s,n,t}$ is a binary variable, $\delta_{s,n,t} = 1$.

If there exists $(n,t) \in \mathcal{N} \times \mathcal{T}$ such that $u_{s,n,t} = \varepsilon$, then constraints (5.4) - (5.5) lead to $0 \leq \delta_{s,n,t} \leq 1$, which means that $\delta_{s,n,t}$ can be either 0 or 1. As the objective function minimizes the number of variables δ_s equal to 1, $\delta_{s,n,t}$ will be set to 0. Finally, for every $x \in \mathcal{X}$, $g_s(x) = \min\Big\{\mathrm{card}(\{(n,t) \in \mathcal{N} \times \mathcal{T} : u_{s,n,t} > \varepsilon\}) : (y_s, u_s) \in \mathcal{R}_s(x)\Big\}$.

□

Proposition 9. *The bilevel formulation of the reliable stochastic expansion planning problem* (5.3) *is equivalent to the following problem:*

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s \phi(x,s) \\ s.t.\ \sum_{s\in\mathcal{S}} p_s g_s(x) \leq \alpha \\ \quad x \in \mathcal{X} \end{cases} \tag{5.7}$$

Proof. ($\Rightarrow$) Let $x \in \mathcal{X}$ be a feasible solution to problem (5.7). We have $\sum_{s\in\mathcal{S}} p_s g_s(x) \leq \alpha$. For every scenario $s \in \mathcal{S}$, we denote by $(y_s^*, u_s^*, \delta_s^*)$ an optimal solution to the problem defining mapping $g_s(x)$. Then, by Proposition 8, $\sum_{s\in\mathcal{S}} p_s U_\varepsilon(x^*, y_s^*, u_s^*) \leq \alpha$ and $(x, (y_s^*, u_s^*)_{s\in\mathcal{S}})$ is a feasible solution to problem (5.3).

($\Leftarrow$) If $(x, (y_s, u_s)_{s\in\mathcal{S}})$ is a feasible solution to problem (5.3), then $\sum_{s\in\mathcal{S}} p_s U_\varepsilon(x, y_s, u_s) \leq \alpha$. By Proposition 8 and as, $\forall s \in \mathcal{S}$, $p_s \geq 0$, $\sum_{s\in\mathcal{S}} p_s g_s(x) \leq \sum_{s\in\mathcal{S}} p_s U_\varepsilon(x, y_s, u_s)$, and $\sum_{s\in\mathcal{S}} p_s g_s(x) \leq \alpha$. Then, x is a feasible solution to problem (5.7).

Finally, the two problems are equivalent since they have the same objective function. □

Formulation (5.7) highlights that the reliability constraint is a constraint on first-stage variables, as remarked by Birge and Louveaux (2011), page 68, on a similar constraint.

Remark 5. *Formulation (5.7) defines an optimistic bilevel problem. Indeed the formulation requires that only one recourse solution of every scenario satisfies the reliability constraint to obtain bilevel feasibility. It is possible to model the pessimistic version of the bilevel reliable stochastic expansion planning problem by replacing mapping g_s for every $s \in \mathcal{S}$ with the following mapping r_s:*

$$r_s(x) = \begin{cases} \max\ e^\top \delta_s \\ s.t.\ (y_s, u_s) \in \mathcal{R}_s(x) \\ \qquad (5.4) - (5.6) \end{cases}$$

leading to the following bilevel program:

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s \phi(x,s) \\ s.t.\ \sum_{s\in\mathcal{S}} p_s r_s(x) \le \alpha \\ \qquad x \in \mathcal{X} \end{cases} \tag{5.8}$$

The maximum in the objective function ensures that every follower optimal solution satisfies the constraint, which is the definition of a pessimistic bilevel feasible solution. In this formulation, if there exists a variable $u_{s,n,t} = \varepsilon$ in a feasible solution, the associated variable $\delta_{s,n,t}$ will be set to 1. Then, in such a formulation, a node and time step (n,t) is considered to have an unsatisfied demand if $u_{s,n,t} = \varepsilon$.

Finally, as in formulation (5.3), variables (y_s, u_s) are required to belong to $\mathcal{R}_s(x)$ for every $x \in \mathcal{X}$, we know that the second-stage cost associated to these solutions is equal to $\phi(x,s)$ for every scenario $s \in \mathcal{S}$. Then, we can reformulate the bilevel reliable stochastic expansion planning problem as the following program:

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s (g_s^\top y_s + M e^\top u_s) \\ s.t.\ \sum_{s\in\mathcal{S}} p_s e^\top \delta_s \le \alpha \\ \qquad C\delta_s \ge u_s - \varepsilon e,\ \forall s \in \mathcal{S} \\ \qquad \delta_s \le (1/\varepsilon) u_s,\ \forall s \in \mathcal{S} \\ \qquad (y_s, u_s) \in \mathcal{R}_s(x),\ \forall s \in \mathcal{S} \\ \qquad x \in \mathcal{X} \end{cases} \tag{5.9}$$

This formulation defines an optimistic bilevel problem with one leader and card($\mathcal{S}$) followers, corresponding to the recourse problem for each scenario $s \in \mathcal{S}$. For every $x \in \mathcal{X}$, the set $\mathcal{R}_s(x)$ is called the *reaction set* associated to scenario s of upper-level variable x.

Remark 6. *The objective function in the formulation of $\mathcal{R}_s(x)$ is the same as in formulation of $\phi(\cdot, s)$, for every $s \in \mathcal{S}$. As $c^\top x$ is constant in the lower-level, we can add it to the lower-level objective function without modifying the optimal solutions. Then, both levels in formulation (5.9) have the same objective function.*

We have presented three equivalent formulations for the reliable stochastic expansion plan-

ning problem. Formulation 5.3 is a natural formulation of the problem. Formulation 5.7 shows the two-stage stochastic structure of the problem. Finally, formulation (5.9) shows the bilevel structure of the reliable stochastic expansion planning problem. Each lower-level problem is a linear program. Depending on the number of scenarios, there might be a large number of linear lower-level problems to solve. The complexity of the problem comes from the upper-level problem. First, the variables x that appear in the lower-level are continuous, which prevent the use of algorithms from the literature based on implicit enumeration of the coupling variables (Xu and Wang, 2014; Lozano and Smith, 2017; Fischetti et al., 2018). Then, there are coupling constraints, which means constraints in the upper-level problem involving lower-level variables. Finally, the upper-level problem involves a large number of binary variables: there are $\text{card}(\mathcal{S} \times \mathcal{N} \times \mathcal{T})$ variables δ_s. With such a large number of binary variables, even the high-point relaxation can be really challenging to solve for real-size instances. One can think that, because the objective functions in both levels are the same, the reliable stochastic expansion planning problem is equivalent to its high-point relaxation and can be reformulated as a single-level linear program. We show in the following that despite this particular structure, we cannot replace $(y_s, u_s) \in \mathcal{R}_s(x)$ by $(y_s, u_s) \in \Omega_s(x)$ in problem (5.9).

5.2.2 High-point relaxation and inconsistency of a single-level linear formulation

In this section, we analyze the high-point relaxation of problem (5.9). We first show that the high-point relaxation does not satisfy the *time-consistency principle*, which is a classical property of two-stage stochastic programs. The high-point relaxation of problem (5.9) can be formulated as follows:

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s (g_s^\top y_s + M \mathrm{e}^\top u_s) \\ s.t.\ T_s x + W_s y_s + Q u_s \geq d_s,\ \forall s \in \mathcal{S} \\ \quad \sum_{s\in\mathcal{S}} p_s \mathrm{e}^\top \delta_s \leq \alpha \\ \quad (5.4) - (5.6) \\ \quad y_s \in \mathbb{R}^{n_2}_+, u_s \in \mathbb{R}^{\text{card}(\mathcal{N}\times\mathcal{T})}_+,\ \forall s \in \mathcal{S} \\ \quad x \in \mathcal{X} \end{cases} \tag{5.10}$$

This problem could have been formulated equivalently by adding the reliability constraint directly in the extensive formulation of problem (5.1). We show the structure of the constraint matrix of problem (5.10) in Figure 5.1. We see that the reliability constraint couples the scenarios together in this formulation.

We now show that this formulation violates the **time-consistency principle**, as presented in a multistage setting by Shapiro et al. (2009), Chapter 6 page 321: *"At every state of a stochastic program, optimality of a decision should not depend on realizations which we already know cannot happen in the future"*. In the case of two-stage stochastic programming, after the first-stage decision is taken, a realization of the randomness is observed. At this time, we know that all the other scenarios will not happen. Then, the set of feasible solutions associated with a given

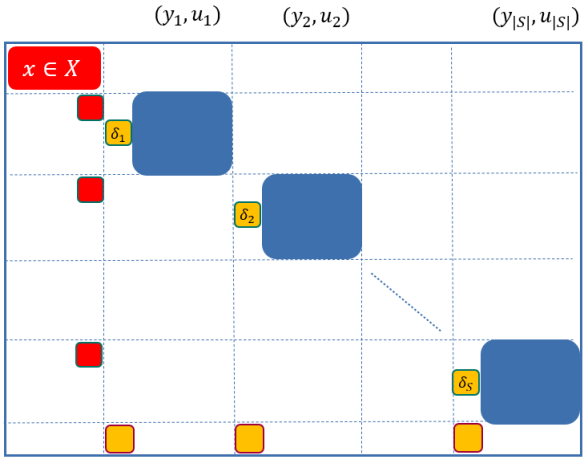


Figure 5.1: Structure of the matrix associated to problem (5.10). Each blue block corresponds to the coefficients of the matrix of the subproblems associated with each scenario. The yellow block in the bottom line correspond to the reliability constraint.

scenario must not depend on the other scenarios.

In order to formalize the time-consistency principle in the case of two-stage stochastic programming, we first introduce several notations. Let $\mathcal{S}$ be a finite set of scenarios. A mapping $p : \mathcal{S} \mapsto [0, 1]$ is a probability distribution over $\mathcal{S}$ if and only if $\sum_{s\in\mathcal{S}} p(s) = 1$. The support of a probability distribution p over $\mathcal{S}$, denoted by $\mathcal{S}(p)$ is the following set $\mathcal{S}(p) = \{s \in \mathcal{S} : \ p(s) \neq 0\}$.

Definition 9. *Let $\mathcal{S}$ be a finite set of scenarios. For any probability distribution p over $\mathcal{S}$, we define a mathematical program $(\Theta(p))$ parameterized by p. Let $n_1 \in \mathbb{N}$ the number of first-stage variables and for every $s \in \mathcal{S}$, $n_s \in \mathbb{N}$ the number of recourse variables in scenario $s \in \mathcal{S}$. We denote by $x \in \mathbb{R}^{n_1}$, the first-stage variables and by $v_s \in \mathbb{R}^{n_s}$ the recourse variables associated with scenario $s \in \mathcal{S}$. We formulate program $(\Theta(p))$ as follows:*

$$\begin{cases} \min \ f_p(x, (v_s)_{s\in\mathcal{S}}) \\ s.t. \ h_p(x, (v_s)_{s\in\mathcal{S}}) \geq 0 \\ \quad v_s \in \mathbb{R}^{n_s}, \ \forall s \in \mathcal{S} \\ \quad x \in \mathbb{R}^{n_1} \end{cases}$$

For any given first-stage solution $\bar{x} \in \mathbb{R}^{n_1}$ and any probability distribution p', we define the set of optimal recourse solutions parameterized p' as:

$$\mathcal{V}(\bar{x}, p') = \left\{(v_s)_{s\in\mathcal{S}(p')} : \ (v_s)_{s\in\mathcal{S}} \in \operatorname{argmin}\{f_{p'}(\bar{x}, (v_s)_{s\in\mathcal{S}}) : \ h_{p'}(\bar{x}, (v_s)_{s\in\mathcal{S}}) \geq 0, \ v_s \in \mathbb{R}^{n_s}, \ \forall s \in \mathcal{S}\}\right\}$$

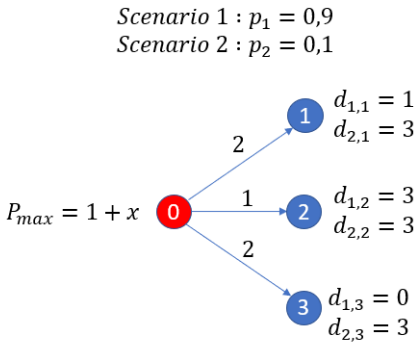
*For a given probability distribution $\bar{p}$ over $\mathcal{S}$ such that $\mathcal{S}(\bar{p}) = \mathcal{S}$, problem $(\Theta(\bar{p}))$ satisfies the **time-consistency principle** if, for any first-stage solution $x \in \mathbb{R}^{n_1}$ and for any probability distribution p' over $\mathcal{S}$, $\mathcal{V}(\bar{x}, p')$ is the set of the subfamilies of optimal recourse solutions to $(\Theta(\bar{p}))$, on the subset of scenarios $\mathcal{S}(p')$:*

$$\left\{(v_s)_{s\in\mathcal{S}(p')} : \ (v_s)_{s\in\mathcal{S}} \in \mathcal{V}(\bar{x}, \bar{p})\right\} \subseteq \mathcal{V}(\bar{x}, p')$$

Proposition 10. *Formulation (5.10) violates the time-consistency principle.*

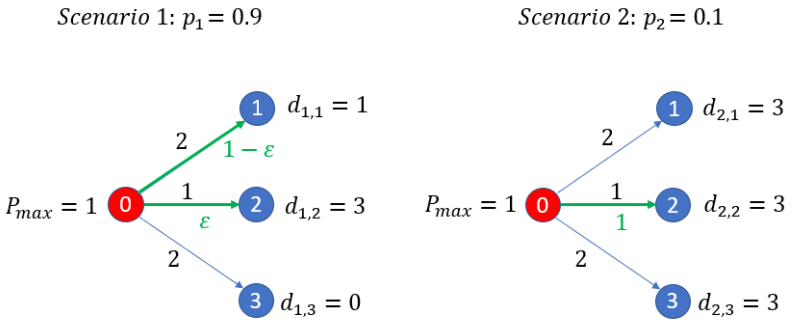
Proof. We present an instance of problem (5.10) which violates the time-consistency principle. The instance is formulated on a graph with four nodes, one time step and two scenarios. The instance can be represented as follows:

Parameters	
Unsupplied cost	10
Investment cost	20
Production cost	0
Arc capacity	$+\infty$
α	2.0
ϵ	0,1



The demand at node i in scenario s is denoted by $d_{s,i}$. The production is only located at note 0. There is unlimited transmission capacity, and the transmission cost is equal to 2 on the arcs from node 0 to node 1 and 3, and equal to 1 on the arc from node 0 to node 2. The investment cost on production capacity is set to 20. The cost associated with an unsatisfied demand is set to 10. We set $\varepsilon = 0.1$ and $\alpha = 2.0$. Variables $(v_s)_{s\in\mathcal{S}}$ of Definition 9 are variables $(y_s, u_s, \delta_s)_{s\in\mathcal{S}}$ in Formulation (5.10). We show that this problem, associated with the probability distribution $\bar{p}$ such that $\bar{p}(1) = 0.9$ and $\bar{p}(2) = 0.1$, does not satisfy the time-consistency principle. We analyze the set of recourse solutions at $\bar{x} = 0$ with distribution probability $\bar{p}$.

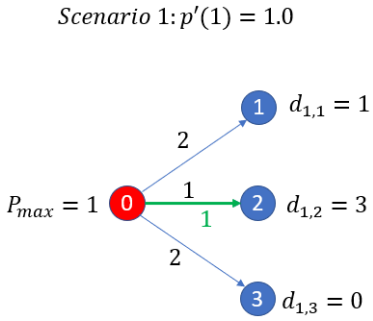
There exists a unique optimal solution to problem (5.10) such that $x = 0$. In scenario 1, $1-\varepsilon$ unit of flow is sent to node 1 and ε unit of flow is sent to node 2, 0 to node 3. The number of unsatisfied demands in scenario 1 is equal to 1 (only node 2). In scenario 2, 1 unit of flow is sent to node 2. The total number of unsatisfied demands is equal to 3 in scenario 2. The expected number of unsatisfied demands is equal to $1.2 \leq \alpha = 2$. We show this solution hereafter:



As the total production capacity is equal to 1 for $x = 0$, the total number of unsatisfied demands in scenario 2 is equal to 3 in any feasible solution. If, in scenario 1, we send less than $1-\varepsilon$ unit of flow to node 1, the number of unsatisfied demands in scenario 1 increases to 2 and the solution is no longer feasible as $\sum_{s\in\mathcal{S}} p_s \mathrm{e}^\top \delta_s = 2.1 > \alpha$. So this solution is the only optimal

solution to problem (5.10) such that $x = 0$.

We now consider the following probability distribution p': $p'(1) = 1$ and $p'(2) = 0$. We have $\mathcal{S}(p') = \{1\}$. Now, the expected number of unsatisfied demands is equal to the total number of unsatisfied demands in scenario 1: $\sum_{s\in\mathcal{S}} p'(s)\mathrm{e}^\top \delta_s = \mathrm{e}^\top \delta_1$. In the solution consisting of sending 1 unit of flow to node 2 in scenario 1, there are 2 unsatisfied demands. Then this solution is feasible as $\alpha = 2$. We show this solution on the following figure:



This solution is optimal as the transmission cost are larger than 1 on the other arcs. Then $\mathcal{V}(0, p')$ is a singleton containing only this solution.

Finally, in the optimal solution to problem (5.10) such that $x = 0$, obtained with the probability distribution p', the flow sent to node 1 in scenario 1 is lower than $1 - \varepsilon$. Then this solution is not part of any feasible solution obtained with the initial probability distribution $\bar{p}$. Then, $\left\{(v_s)_{s\in\mathcal{S}(p')} : (v_s)_{s\in\mathcal{S}} \in \mathcal{V}(0, \bar{p})\right\} \notin \mathcal{V}(0, p')$ and this problem violates the time-consistency principle. □

Corollary 4. *Problem (5.10) is not the extensive formulation of a two-stage stochastic program.*

Proof. By definition, a two-stage stochastic program can be written according to formulation (5.1). Then, for any given fixed first-stage solution $x \in \mathcal{X}$, the problem becomes decomposable according to the scenarios. Let $\bar{p}$ the initial probability distribution, and p' a probability distribution over the set of scenarios $\mathcal{S}$. For any first-stage solution $x \in \mathcal{X}$, we have $\left\{(v_s)_{s\in\mathcal{S}(p')} : (v_s)_{s\in\mathcal{S}} \in \mathcal{V}(x, \bar{p})\right\} = \mathcal{V}(x, p')$. This means that any two-stage stochastic program satisfies the time-consistency principle. By contrapositive, formulation (5.10) does not satisfied the time-consistency principle, and therefore is not a formulation of a two-stage stochastic program. □

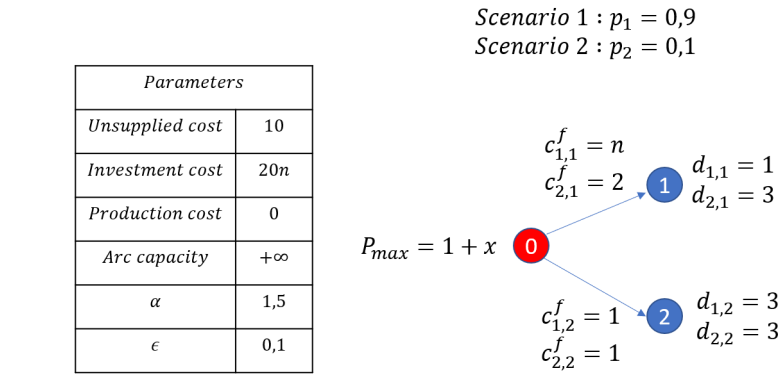
This result shows that constraint $\sum_{s\in\mathcal{S}} p_s \mathrm{e}^\top \delta_s \leq \alpha$, which couples the scenarios together, breaks the two-stage stochastic structure of the problem. The bilevel formulation presented in Subsection 5.2.1 is required to formulate the problem as a two-stage stochastic program.

Let $(x^*, (y_s^*, u_s^*, \delta_s^*)_{s\in\mathcal{S}})$ an optimal solution to problem (5.10). Violating the time-consistency principle implies that there exists a scenario $s \in \mathcal{S}$ such that $g_s^\top y_s^* + M\mathrm{e}^\top u_s^* > \phi(x^*, s)$. We show in the following proposition that the ratio between these two values in not bounded.

Proposition 11. *Let $A > 0$ a positive real value. There exists a stochastic expansion planning problem in which, in any optimal solution $(x^*, (y_s^*, u_s^*)_{s\in\mathcal{S}})$, $g_s^\top y_s^* + Me^\top u_s^* > A\phi(x^*, s)$.*

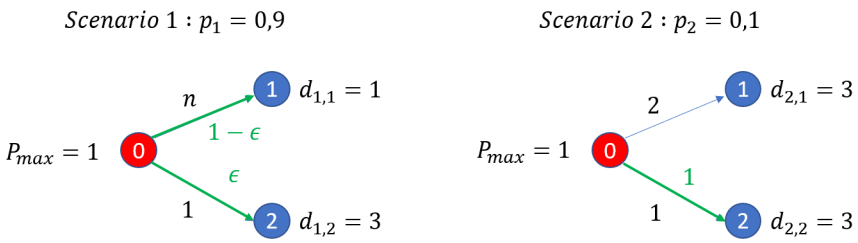
Proof. We show this with the following stochastic expansion planning problem parameterized by $n \in \mathbb{N}^*$, with three nodes, one time step and two scenarios:

Parameters	
Unsupplied cost	10
Investment cost	$20n$
Production cost	0
Arc capacity	$+\infty$
α	1,5
ϵ	0,1



We denote by $d_{s,i}$ the demand at node i in scenario s. The production is only located at note 0. $c^f_{s,i}$ represents the flow cost on arc from node 0 to node i in scenario $s$and there is unlimited transmission capacity. The investment cost on production capacity is set to $20n$. The cost associated with an unsatisfied demand is set to 10. We set $\varepsilon = 0.1$ and $\alpha = 1.5$.

With the high-point relaxation formulation (5.10), we remark that, by sending $1 - \varepsilon$ unit of flow to node 1 in scenario 1, one can satisfy the reliability constraint, as the expected number of unsatisfied demands will be equal to $0.9 + 0.1 * 2 = 1.1 < 1.5 = \alpha$. We show this solution in the following figure:



This solution is the optimal solution of the high-point relaxation (5.10). The cost associated with recourse variables of scenario 1 is then equal to $v^*_n = (1 - \varepsilon)n + \varepsilon + 3M$. The optimal value of the recourse problem (5.2) associated with scenario 1 evaluated at $x = 0$ is equal to $\phi(0, 1) = 1 + 3M$, as it is cheaper to send all the energy to node 2. Then we have:

$$\lim_{n \to \infty} \frac{v^*_n}{\phi(x^*, 1)} = +\infty$$

□

Proposition 11 means that, for any value $A > 0$, we can create a problem in which, in an optimal solution, there exists a scenario for which we pay A times more than an optimal recourse solution in order to satisfy the reliability constraint. This does not make sense as, after the realization of the random variables, there would be no reason to pay for example 1000 times the cost of an optimal recourse solution to satisfy a constraint involving other scenarios that we

already know will not realize. This shows the fundamental difference between problem (5.9) and its high-point relaxation. This also shows that, despite the objectives functions in both levels are the same, we cannot replace $(y_s, u_s) \in \mathcal{R}_s(x)$ by $(y_s, u_s) \in \Omega_s(x)$ in problem (5.9).

5.3 A Benders-based heuristic solution procedure

We present in this section a Benders-based heuristic to find feasible solutions to problem (5.9). As the stochastic expansion planning problems involved in studies from RTE may have up to several million variables, we do not expect classical methods from the literature to be able to solve them in practice (see e.g. Table 3.2). The objective of our approach is to define a sequence of single-level problems that converges to feasible solutions to problem (5.9), using the fact that, if we relax the reliability constraint, problem (5.9) becomes a single-level problem. We propose a procedure based on a binary search on total the investment cost, and show that under some assumptions, such a binary search converges to a bilevel feasible solution. We introduce hereafter the investment free problem, and an hypothesis based on this problem which ensures both the feasibility of the bilevel problem, and the convergence of the proposed heuristic to feasible solutions.

Definition 10. *The investment free stochastic expansion planning problem, hereafter referred to as the investment free problem, is the following stochastic program:*

$$\begin{cases} \min \sum_{s \in \mathcal{S}} p_s (g_s^\top y_s + M e^\top u_s) \\ s.t. : T_s x + W_s y_s + Q u_s \geq d_s, \ \forall s \in \mathcal{S} \\ \qquad y_s \in \mathbb{R}_+^{n_2}, \ \forall s \in \mathcal{S} \\ \qquad u_s \in \mathbb{R}_+^{\text{card}(\mathcal{N} \times \mathcal{T})}, \ \forall s \in \mathcal{S} \\ \qquad x \in \mathcal{X} \end{cases} \tag{5.11}$$

In the following, we make the following assumption:

Assumption 1. *In any optimal solution to problem (5.11), variables $u_s = 0$ for every scenario $s \in \mathcal{S}$.*

We refer to this assumption as the *investment free problem hypothesis*. This means that, if adding production or transmission capacities costs nothing, we always choose to add enough capacities in order to satisfy all the demands in every scenarios. This basically traduces the fact that the cost M associated to an unsatisfied demand is large enough to prefer satisfying a demand whenever it is possible. This also implies that there exists a feasible investment which allows to satisfy all the demands. This hypothesis is reasonable in industrial instances.

Proposition 12. *Let (x_f, y_f, u_f) denote an optimal solution to the investment free problem. (x_f, y_f, u_f) is a bilevel feasible solution to problem (5.9) for any $\alpha \geq 0$ and any $\varepsilon \geq 0$, and there exists no bilevel solution with a higher investment cost and a lower total cost.*

Proof. Due to the investment free problem hypothesis, we have $u_f = 0$, and (x_f, y_f, u_f) is therefore a bilevel feasible solution for any $\alpha \geq 0$ and any $\varepsilon \geq 0$.

Now for any Λ such that $\Lambda \geq c^\top x_f$, we denote by $(x_\Lambda, y_\Lambda, u_\Lambda)$ a feasible solution to (Inv(Λ)). As (x_f, y_f, u_f) is a solution to the investment free problem, we know that $\sum_{s\in\mathcal{S}} p_s(g_s^\top y_{\Lambda,s} + Me^\top u_{\Lambda,s}) \geq \sum_{s\in\mathcal{S}} p_s(g_s^\top y_{f,s} + Me^\top u_{f,s})$. Then, the investment cost constraint $\Lambda \geq c^\top x_f$ implies that $c^\top x_\Lambda \geq c^\top x_f$. Finally, any solution with a higher investment cost than the investment free problem has a higher cost than (x_f, y_f, u_f). □

This property allows to restrict the search of a solution to the bilevel program. An optimal solution to problem (5.9) has necessary an investment cost lower than the investment cost of an optimal solution to the investment free program (5.11).

In order to formally describe our Benders-based heuristic, we introduce the minimum investment problem, formulated as follows:

Definition 11. *Let $\Lambda \geq 0$ be a positive real number, the minimum investment problem is denoted as* (Inv(Λ)) *and formulated as:*

$$v(\Lambda) = \begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s(g_s^\top y_s + Me^\top u_s) \\ s.t. : T_s x + W_s y_s + Q u_s \geq d_s,\ \forall s \in \mathcal{S} \\ \quad c^\top x \geq \Lambda \\ \quad y_s \in \mathbb{R}_+^{n_2},\ \forall s \in \mathcal{S} \\ \quad u_s \in \mathbb{R}_+^{\text{card}(\mathcal{N}\times\mathcal{T})},\ \forall s \in \mathcal{S} \\ \quad x \in \mathcal{X} \end{cases} \tag{5.12}$$

Algorithm 5.1 outlines the general structure of our heuristic. The core idea is to perform a binary search on the minimum investment value Λ, and to solve the minimum investment problem (Inv(Λ)) for every value of Λ selected during the binary search. Given a lower bound $\Lambda_{\min}$ on the investment value (initially 0) and an upper bound on the investment value $\Lambda_{\max}$ (the investment cost of a first-stage optimal solution to the investment free problem), we solve the minimum investment problem for an investment value $\Lambda = \tau\Lambda_{\min} + (1-\tau)\Lambda_{\max}$, with $\tau \in (0,1)$ the binary search parameter. If there is no bilevel feasible solution with a cost lower than the best bilevel feasible solution computed so far, then $\Lambda_{\min}$ is set to Λ. Otherwise $\Lambda_{\max}$ is set to the lowest invest cost associated to a bilevel feasible solution. At the first iteration of the binary search, the minimum investment value is set to 0, which means that we solve the stochastic expansion planning problem (5.1). Indeed, if the solution to the stochastic expansion planning problem is bilevel feasible, there is no need to perform a binary search as this would be a bilevel optimal solution (an optimal solution to a relaxation which is feasible for the original problem).

Since the problem (Inv(Λ)) is as hard as the stochastic expansion planning problem (5.1), we solve (Inv(Λ)) with a Benders decomposition algorithm (Lines 7 to 16 in Algorithm 5.1). The subproblem associated with scenario $s \in \mathcal{S}$, evaluated at first-stage solution $x \in \mathcal{X}$, denoted by

$(SP(x,s))$, is the following:

$$\phi(x,s) = \begin{cases} \min\ g_s^\top y_s + M\mathrm{e}^\top u_s \\ s.t.\ W_s y_s + Q u_s \geq d_s - T_s x \\ \qquad y_s \in \mathbb{R}^{n_2}_+,\ u_s \in \mathbb{R}^{\mathrm{card}(\mathcal{N}\times\mathcal{T})}_+ \end{cases}$$

We denote by $\Pi_s = \{\pi_s \in \mathbb{R}^m :\ W_s^\top \pi_s \leq g_s,\ Q_s^\top \pi_s \leq M\mathrm{e}\}$ the set of feasible dual variables of $(SP(x,s))$, for any $x \in \mathcal{X}$, and by $\mathrm{Vert}(\Pi_s)$ the set of extreme points of Π_s. At iteration k of the Benders decomposition algorithm, we denote by $\mathrm{Vert}(\Pi_s)^{(k)}$ the subset of extreme points defining the cuts present in the relaxed master program. The relaxed master program, denoted by $(RMP)^{(k)}$, is the following program:

$$\begin{cases} \min\ c^\top x + \sum_{s\in\mathcal{S}} p_s \theta_s \\ s.t.: \theta_s \geq \pi_s^\top (d_s - T_s x),\ \forall s \in \mathcal{S},\ \forall \pi_s \in \mathrm{Vert}(\Pi_s)^{(k)} \\ \qquad c^\top x \geq \Lambda \\ \qquad x \in \mathcal{X}, \theta_s \in \mathbb{R},\ \forall s \in \mathcal{S} \end{cases}$$

The benefit of using the Benders decomposition is that we can perform the entire binary search with only one relaxed master program. The Benders cuts computed when solving $(\mathrm{Inv}(\Lambda))$ for a given $\Lambda \geq 0$ are still valid when solving $(\mathrm{Inv}(\Lambda'))$ for any $\Lambda' \geq 0$. Also, it allows to check if every first-stage solution evaluated during the Benders decomposition algorithm is bilevel feasible (lines 17 to 21 in Algorithm 5.1).

In Algorithm 5.1 line 19, we check if the optimal recourse solution returned by the solver satisfies the reliability constraint or not. We refer to this strategy as the **Random strategy**. However, if this solution does not satisfy the constraint, there might exist an other optimal recourse solution which does satisfy the constraint. We then introduce the **Min strategy** based on the auxiliary subproblem defined hereafter. Let $x \in \mathcal{X}$ be a feasible first-stage solution for the minimum investment problem. Let $\phi(x,s)$ be the optimal cost associated to scenario $s \in \mathcal{S}$ evaluated at x. The auxiliary subproblem, denoted $(\mathrm{Aux}(x,s))$ hereafter, is the following:

$$N(x,s) = \begin{cases} \min \sum_{n\in\mathcal{N}} \sum_{t\in\mathcal{T}} \delta_{n,t} \\ s.t.\ W_s y + Q u + T_s x \geq d_s \\ \qquad \sum_{s\in\mathcal{S}} p_s (g_s^\top y + M e^\top u) \leq \phi(x,s) \\ \qquad \delta_{n,t} \leq \frac{1}{\varepsilon} u_{n,t},\ \forall n \in \mathcal{N},\ \forall t \in \mathcal{T} \\ \qquad C\delta_{n,t} \geq u_{n,t} - \varepsilon,\ \forall n \in \mathcal{N},\ \forall t \in \mathcal{T} \\ \qquad y \in \mathbb{R}^{n_2} \\ \qquad u \in \mathbb{R}^{\mathrm{card}(\mathcal{N}\times\mathcal{T})} \\ \qquad \delta \in \{0,1\}^{\mathrm{card}(\mathcal{N})\times\mathrm{card}(\mathcal{T})} \end{cases} \tag{5.13}$$

Algorithm 5.1: Benders-based heuristic (All-Rand version)

Parameters: $\varepsilon_{\text{opt}} > 0$, $\varepsilon_{\text{inv}} > 0$, heuristic parameters $\alpha \geq 0$, $\varepsilon > 0$, binary search parameter $t \in (0,1)$

Initialization: $k \leftarrow 0$, `found_feasible` $\leftarrow$ `False`, $\Lambda \leftarrow 0$,$\Lambda_{\min} \leftarrow 0$

Solve invest free problem (5.11) and retrieve x_{best} an optimal first-stage solution, and $(\phi(x_{\text{best}}, s))_{s\in\mathcal{S}}$

$v_{\text{best}} \leftarrow c^\top x_{\text{best}} + \sum_{s\in\mathcal{S}} p_s \phi(x_{\text{best}}, s)$

$\Lambda_{\max} \leftarrow c^\top x_{\text{best}}$

while $\Lambda_{\max} - \Lambda_{\min} > \varepsilon_{\text{inv}}$ **do**

`/* Chosing a new bound on investment */`

  Set investment constraint to $c^\top x \geq \Lambda$ in $(RMP)^{(k)}$

  $UB^{(k)} \leftarrow +\infty$,$LB^{(k)} \leftarrow -\infty$

  `found_feasible` $\leftarrow$ `False`

  `/* Solving the Benders decomposition associated with investment constraint */`

  while $UB^{(k)} > LB^{(k)} + \varepsilon_{\text{opt}}$ **do**

    $k \leftarrow k+1$

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

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

    for $s \in \mathcal{S}$ **do**

      Solve $(SP(\check{x}^{(k)}, s))$

      Retrieve $\pi_s \in \text{Vert}(\Pi_s)$ and (y_s, u_s) optimal solution

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

    $UB^{(k)} \leftarrow min\Big(UB^{(k-1)}, c^\top \check{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \pi_s^\top (d_s - T_s \check{x}^{(k)})\Big)$

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

    `/* Checking bilevel feasibility */`

    if $\sum_{s\in\mathcal{S}} p_s U_\varepsilon(\check{x}^{(k)}, y_s, u_s) \leq \alpha$ `and` $c^\top \check{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \pi_s^\top (d_s - T_s \check{x}^{(k)}) < v_{\text{best}}$ **then**

      `found_feasible` $\leftarrow$ `True`

      $x_{\text{best}} \leftarrow \check{x}^{(k)}$

      $v_{\text{best}} \leftarrow c^\top \check{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \pi_s^\top (d_s - T_s \check{x}^{(k)})$

      $\Lambda_{\max} \leftarrow \min\{\Lambda_{\max}, c^\top \check{x}^{(k)}\}$

  if `found_feasible` $=$ `False` **then**

    $\Lambda_{\min} \leftarrow \Lambda$

  $\Lambda \leftarrow \tau\Lambda_{\min} + (1-\tau)\Lambda_{\max}$

Return x_{best}

We denote by $N(x,s)$ its optimal value. This auxiliary subproblem searches, among all optimal solutions of the subproblem, the one which minimizes the number of unsatisfied demands. Solving all the auxiliary subproblems for a given first-stage allows to determine if this latter is bilevel feasible or not. We formalize this statement in the following proposition.

Proposition 13. *Let $\alpha \geq 0$, $\varepsilon \geq 0$ be the parameters of the reliability constraint and $x \in \mathcal{X}$. There exists (y,u) feasible recourse solution such that (x,y,u) is bilevel feasible if and only if $\sum_{s\in\mathcal{S}} p_s N(x,s) \leq \alpha$.*

Proof. For every $s \in \mathcal{S}$, let (y_s, u_s, δ_s) an optimal solution to $(\text{Aux}(x,s))$.

($\Rightarrow$) Assume that $(x, (y_s)_{s\in\mathcal{S}}, (u_s)_{s\in\mathcal{S}})$ is bilevel feasible. Then, by definition of the auxiliary subproblem, $N(x,s) \leq U_\varepsilon(x, y_s, u_s),\ \forall s \in \mathcal{S}$. Because $(x, (y_s)_{s\in\mathcal{S}}, (u_s)_{s\in\mathcal{S}})$ is bilevel feasible, we have $\sum_{s\in} p_s U_\varepsilon(x, y_s, u_s) \leq \alpha$ and therefore $\sum_{s\in\mathcal{S}} p_s N(x,s) \leq \alpha$.

($\Leftarrow$) Assume $\sum_{s\in\mathcal{S}} p_s N(x,s) \leq \alpha$. The constraint $\sum_{s\in\mathcal{S}} p_s(g_s^\top y + Me^\top u) \leq \phi(x,s)$ ensures that (y_s, u_s) is an optimal solution to $(SP(x,s))$, and $(x, (y_s)_{s\in\mathcal{S}}, (u_s)_{s\in\mathcal{S}})$ is bilevel feasible.

□

This proposition shows that the bilevel feasibility of a solution can be checked using the separability of the subproblems, whereas the constraint couples them together.

We present in detail the **Min strategy** in Algorithm 5.2. Applying the **Min strategy** consists of calling Algorithm 5.2 instead of checking the condition of Line 17 in Algorithm 5.1. If the Min strategy procedure returns `True`, then we apply Lines 18-21 in Algorithm 5.1.

Algorithm 5.2: Min strategy

Parameters : $\alpha \geq 0$, $\varepsilon > 0$, $\check{x}^{(k)}$, $(y_s, u_s, \pi_s)_{s\in\mathcal{S}}$, v_{best}
1 **if** $\sum_{s\in\mathcal{S}} p_s U_\varepsilon(\check{x}^{(k)}, y_s, u_s) \leq \alpha$ `and` $c^\top \check{x}^{(k)} + \sum_{s\in\mathcal{S}} p_s \pi_s^\top (d_s - T_s \check{x}^{(k)}) < v_{\text{best}}$ **then**
2 Return `True`
3 **else**
4 **for** $s \in \mathcal{S}$ **do**
5 Solve $(\text{Aux}(\check{x}^{(k)}, s))$
6 Retrieve $N(\check{x}^{(k)}, s)$ optimal value of $(\text{Aux}(\check{x}^{(k)}, s))$
7 **if** $\sum_{s\in\mathcal{S}} p_s N(\check{x}^{(k)}, s) \leq \alpha$ **then**
8 Return `True`
9 Return `False`

The Min strategy involves solving many integer program, with $\text{card}(\mathcal{N}\times\mathcal{T})$ binary variables in each, which can be quite large. We accelerate this procedure by stopping solving the auxiliary subproblems after having solved a subset $\mathcal{S}_0 \subset \mathcal{S}$ as soon as $\sum_{s\in\mathcal{S}_0} p_s N(\check{x}^{(k)}, s) > \alpha$.

The algorithm proposed in this section uses the Benders decomposition algorithm as a way to sample possible first-stage solutions in order to find good feasible solutions. A similar procedure can be done by only checking the bilevel feasibility of an optimal solution to the minimum investment problem (5.12). We define the **frequency** of the bilevel feasibility checking. This parameter can be set to **All**, if the algorithm checks the bilevel feasibility for all the first-stage solutions evaluated during the optimization process, or to **Opt**, if the algorithm checks the bilevel feasibility of only optimal solutions to the minimum investment problem. We finally obtain four different values of parameters to run the heuristic:

1. **All-Rand** : Using a Benders decomposition algorithm, the algorithm evaluates the bilevel feasibility of every first-stage solution evaluated, using the random optimal solution of the subproblems given by the solver.

2. **All-Min** : Using a Benders decomposition algorithm, the algorithm evaluates the bilevel feasibility of every first-stage solution evaluated by solving the auxiliary subproblems.

3. **Opt-Rand** : The algorithm only evaluates the bilevel feasibility of the optimal solution to the minimum investment problem, using the random optimal solution of the subproblems given by the solver.

4. **Opt-Min** : The algorithm only evaluates the bilevel feasibility of the optimal solution to the minimum investment problem by solving the auxiliary subproblems.

5.4 Experimental design and numerical results

We present in this section the results we obtain for the four versions of the heuristic presented in Section 5.3 on randomly generated instances. We compare these results to the results obtained by using the classical SOS-1 method to solve the KKT-reformulation of the bilevel program. We first present our instance generation procedure and the parameters we used to generate our instances, then we present the numerical results.

5.4.1 Instance generation

We present here the method we use to generate instances of reliable stochastic expansion planning problems. The instance generation is divided in two parts. First, we generate a random graph representing a network. Then, on this network, we generate random data to model investments and the operational behavior.

Graph generation

We first present the graph generation algorithm we use. In preliminary experiments, we noticed that classical graph generation algorithms such as the ones proposed in (Erdos and Rényi, 1960; Hakimi, 1962; Viger and Latapy, 2005) did not produce satisfying results. Either the produced graphs did not model well the actual structure of transmission systems networks, or they were not similar to networks generated based on geographical positions. To address this issue, we propose a dedicated graph generation method described in Algorithm 5.3. This method is composed of three parts:

1. Sampling nodes positions
2. Adding edges until connectivity
3. Removing edges until a desired density is reached

The first part consists of sampling some integer positions on a rectangular grid. Let N be number of nodes of the graph we want to create. Let R be the ratio between the horizontal size and the vertical size of the rectangle, and P be the proportion on integer points on which there will be a node of the graph. We define the limits on the horizontal and vertical coordinates as $X_{max} = \sqrt{\frac{NG}{P}}$ and $Y_{max} = \sqrt{\frac{N}{PG}}$. Then, the algorithm starts by sampling N uniformly distributed integer positions on the grid defined by $[1, X_{max}] \times [1, Y_{max}]$. Figure 5.2 shows a possible result of such a sampling procedure. In the example, $X_{max} \approx 8.66$ and $Y_{max} \approx 5.77$. We finally observe an exact proportion of sampled points of $8/40 = 12.5\%$.

The second part consists of adding edges to the graph so that we obtain connectivity. The procedure is the following. We begin with a distance $D_{max} = 1$, and we add all the edges between nodes which are at an Euclidean distance of at most D_{max}. If the resulting graph is connected,

Algorithm 5.3: Undirected graph generation procedure

Parameters : Number of nodes N, Grid ratio G, proportion of nodes P, final density d, `max_try` $\in \mathbb{N}^*$

Initialization: $V \leftarrow \varnothing$ the set of nodes, $E \leftarrow \varnothing$ the set of edges, `try` $\leftarrow 0$

`/* Sampling nodes */`

Compute $X_{max} = \sqrt{\frac{NG}{P}}$ and $Y_{max} = \sqrt{\frac{N}{PG}}$

Sample N integer points in $[1, X_{max}] \times [1, Y_{max}]$ and add them to V

Let $D_{max} \leftarrow 0$

`/* Adding edges until connectivity */`

while G *not connected* **do**

  $D_{max} \leftarrow D_{max} + 1$

  $E \leftarrow \{(i,j), 1 \leq i \leq N, i+1 \leq j \leq N \mid dist(i,j) \leq D_{max}\}$

`/* Removing edges to get required density */`

while $\frac{2card(E)}{N(N-1)} > d$ **do**

  `try` $\leftarrow 1$

  Choose (i,j) in E

  while $(V, E\backslash\{(i,j)\})$ *is not connected* **do**

    if `try` $=$ `max_try` **then**

      Return $G = (V, E)$ `// Unable to reach density`

    `try` $\leftarrow$ `try` $+ 1$

    Choose (i,j) in E

  $E \leftarrow E\backslash\{(i,j)\}$s

Return $G = (V, E)$

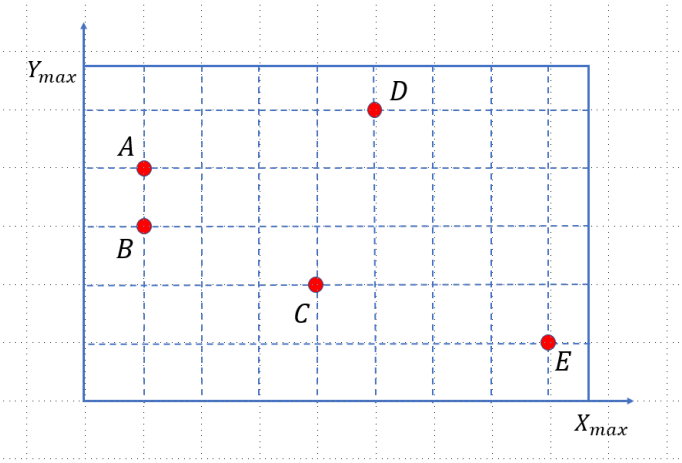


Figure 5.2: A possible result of sampling 5 nodes on a grid with ratio $G = 1.5$ and a proportion parameter $P = 0.1$.

we stop, otherwise, $D_{max} \leftarrow D_{max} + 1$ and we restart the process. In the example presented in Figure 5.2, this leads to the connected graph of Figure 5.3.

After this procedure, we get a graph with a given density. A parameter of the graph generation procedure is the density d the user want to get at the end. Then, the algorithm removes random edges letting the graph connected until it reaches the required density or it fails at having a connected graph after a given number of trials. For example, in the graph of Figure 5.3, edge CE cannot be removed as it would lead to a disconnected graph. We finally present a random graph with 30 nodes and a final tree that we would obtain by removing all the possible arcs according to a given order in Figure 5.5 and Figure 5.6.

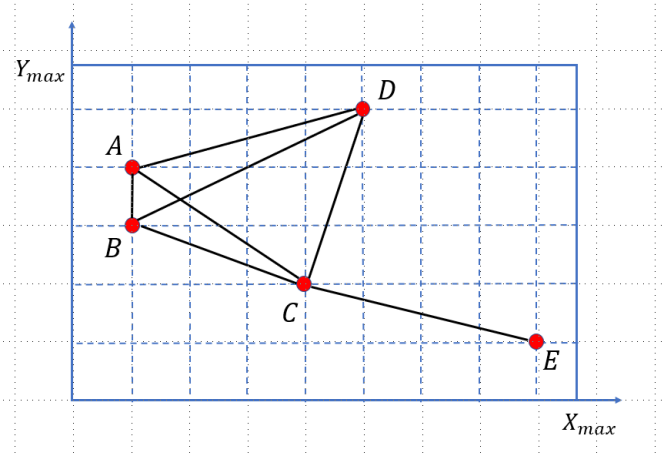


Figure 5.3: First connected graph, obtained after adding all the edges between points at an Euclidean distance of at most 5.

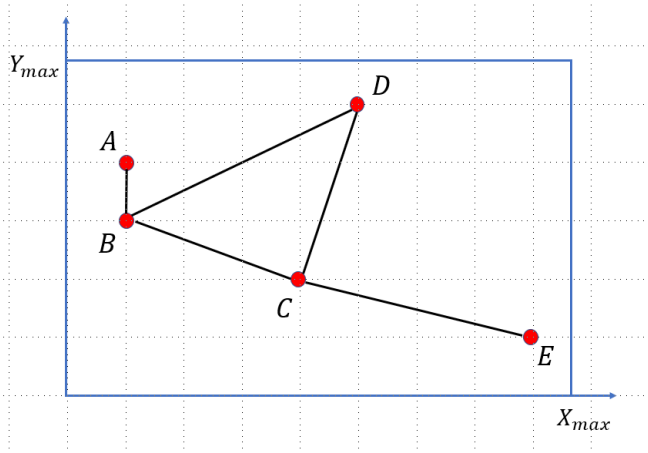


Figure 5.4: Final graph obtained after removing edges to reach a density of 0.5.

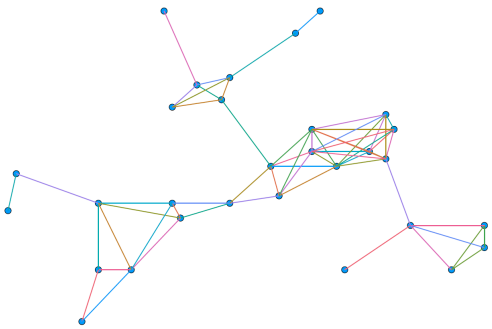


Figure 5.5: A random graph of 30 nodes, before deleting edges. Initial density = $124/870 \approx 0.14$

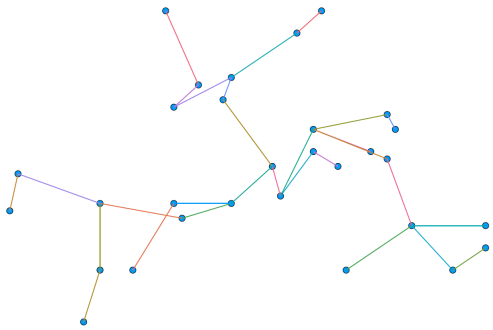


Figure 5.6: Final tree obtained after deleting the maximum number of edges. Final density = $2 * 29/870 \approx 0.067$

Model and data generation

We present hereafter the model we use to evaluate the performance of our proposed heuristic. Let $\mathcal{T}$ be a set of time steps, $G = (\mathcal{N}, \mathcal{E})$ an undirected graph and $\mathcal{S}$ a set of scenarios. We denote by $\mathcal{N}_P \subset \mathcal{N}$ the set of nodes where the is a production capacity. One can invest on

any transmission capacity, and any production capacity on production nodes. We assume that investment variables are continuous here. The flow in the recourse model is algebraic, and counted as its absolute value in the objective function. There exists a maximum gradient value on production variables. Demands, production costs, and transmission costs are stochastic. We list hereafter the parameters of the problem:

- $\varepsilon > 0$: Threshold above which an unsupplied demand is accounted
- $\alpha \geq 0$: Maximum number of expected unsatisfied demands
- $c_{inv}^{prod} \in \mathbb{R}^{\text{card}(\mathcal{N}_P)}$: Vector of production capacity investment costs for each node $n \in \mathcal{N}_P$
- $c_{inv}^{flow} \in \mathbb{R}^{\text{card}(\mathcal{A})}$: Vector of transmission capacity investment costs for each arc $a \in \mathcal{A}$
- $p_s \in [0, 1]$: probability associated with scenario $s \in \mathcal{S}$
- $g_s^{prod} \in \mathbb{R}^{\text{card}(\mathcal{N}_P \times \mathcal{T})}$:Vector of production costs, for each node $n \in \mathcal{N}_P$ and each time step $t \in \mathcal{T}$, in scenario $s \in \mathcal{S}$
- $g^{flow} \in \mathbb{R}$: Transmission cost, the same for each arc $a \in \mathcal{A}$ and each time step $t \in \mathcal{T}$
- $M \geq 0$: Cost associated with an unsatisfied demand
- $C > 0$: Large enough real number to set variables δ_s to one when required
- $F^{init} \in \mathbb{R}^{\text{card}(\mathcal{A})}$: Vector of initial transmission capacities, on each arc $a \in \mathcal{A}$
- $\Delta \in [0, 1]$: Maximum gradient on production variables, as a proportion of the total production capacity
- $d_s \in \mathbb{R}^{\text{card}(\mathcal{N} \times \mathcal{T})}$: Vector of demands in scenario $s \in \mathcal{S}$, for each node $n \in \mathcal{N}$ and time step $t \in \mathcal{T}$

We also present hereafter the optimization variables:

- $P^{max} \in \mathbb{R}^{\text{card}(\mathcal{N}_P)}$: Vector of production capacity investment variables
- $F^{max} \in \mathbb{R}^{\text{card}(\mathcal{A})}$: Vector of transmission capacity investment variables
- $\delta_s \in \{0, 1\}^{\text{card}(\mathcal{N} \times \mathcal{T})}$: Vector of binary variable counting if there is an unsatisfied demand of at least ε in scenario $s \in \mathcal{S}$, for each node $n \in \mathcal{N}$ and time step $t \in \mathcal{T}$
- $\pi_s \in \mathbb{R}^{\text{card}(\mathcal{N}_P \times \mathcal{T})}$: Vector of production variables, in scenario $s \in \mathcal{S}$, for each production node $n \in \mathcal{N}_P$ and time step $t \in \mathcal{T}$
- $f_s \in \mathbb{R}^{\text{card}(\mathcal{A} \times \mathcal{T})}$: Vector of algebraic flow variables in scenario $s \in \mathcal{S}$, for each edge $a \in \mathcal{A}$ and time step $t \in \mathcal{T}$
- $\bar{f}_s \in \mathbb{R}^{\text{card}(\mathcal{A} \times \mathcal{T})}$: Vector of absolute value of flows in scenario $s \in \mathcal{S}$, for each edge $a \in \mathcal{A}$ and time step $t \in \mathcal{T}$
- $u_s \in \mathbb{R}^{\text{card}(\mathcal{N} \times \mathcal{T})}$: Vector of unsatisfied demands in scenario $s \in \mathcal{S}$, for each node $n \in \mathcal{N}$ and time step $t \in \mathcal{T}$

We can now write formally the proposed bilevel stochastic expansion planning problem:

$$\begin{cases} \min\ (c_{inv}^{prod})^\top P^{max} + (c_{inv}^{flow})^\top F^{max} \\ \qquad + \sum_{s\in\mathcal{S}} p_s \Big((g_s^{prod})^\top \pi_s + g^{flow} \sum_{a\in A}\sum_{t\in T} \bar{f}_{s,a,t} + M\mathrm{e}^\top u_s \Big) \\ s.t. : (\pi_s, f_s, \bar{f}_s, u_s) \in \mathcal{R}_s(x),\ \forall s \in \mathcal{S} \\ \qquad u_s - \varepsilon \mathrm{e} \le C\delta_s,\ \forall s \in \mathcal{S} \\ \qquad \delta_s \le \varepsilon^{-1} u_s,\ \forall s \in \mathcal{S} \\ \qquad \sum_{s\in\mathcal{S}} p_s \mathrm{e}^\top \delta_s \le \alpha,\ \forall s \in \mathcal{S} \\ \qquad P^{max} \in \mathbb{R}_+^{\mathrm{card}(N_P)}, F^{max} \in \mathbb{R}_+^{\mathrm{card}(A)} \\ \qquad \delta_s \in \{0,1\}^{\mathrm{card}(N\times T)},\ \forall s \in \mathcal{S} \end{cases} \tag{5.14}$$

where $\mathcal{R}_s(x)$ is defined as the following set:

$$\mathcal{R}_s(x) = \begin{cases} \arg\min\ (g_s^{prod})^\top \pi_s + g_s^{flow} \sum_{a\in A}\sum_{t\in T} \bar{f}_{s,a,t} + M\mathrm{e}^\top u_s \\ \quad s.t. : \pi_{s,n,t} \le P_n^{max},\ \forall (n,t) \in \mathcal{N}\times\mathcal{T} \\ \qquad -(F_a^{init} + F_a^{max}) \le f_{s,a,t} \le F_a^{init} + F_a^{max},\ \forall (a,t) \in \mathcal{A}\times\mathcal{T} \\ \qquad \bar{f}_{s,a,t} \ge f_{s,a,t},\ \forall (a,t) \in \mathcal{A}\times\mathcal{T} \\ \qquad \bar{f}_{s,a,t} \ge -f_{s,a,t},\ \forall (a,t) \in \mathcal{A}\times\mathcal{T} \\ \qquad \pi_{s,n,t} \le \pi_{s,n,t-1} + \Delta P_n^{max},\ \forall (n,t) \in \mathcal{N}\times(\mathcal{T}\backslash\{0\}) \\ \qquad \pi_{s,n,t} \ge \pi_{s,n,t-1} - \Delta P_n^{max},\ \forall (n,t) \in \mathcal{N}\times(\mathcal{T}\backslash\{0\}) \\ \qquad \pi_{s,n,0} \le \pi_{s,n,T} + \Delta P_n^{max},\ \forall n \in \mathcal{N} \\ \qquad \pi_{s,n,0} \ge \pi_{s,n,T} - \Delta P_n^{max},\ \forall n \in \mathcal{N} \\ \qquad \pi_{s,n,t} + \sum_{a\in\Gamma^+(n)} f_{s,a,t} - \sum_{a\in\Gamma^-(n)} f_{s,a,t} + u_{s,n,t} \ge d_{s,n,t},\ \forall (n,t) \in \mathcal{N}_\mathcal{P}\times\mathcal{T} \\ \qquad \sum_{a\in\Gamma^+(n)} f_{s,a,t} - \sum_{a\in\Gamma^-(n)} f_{s,a,t} + u_{s,n,t} \ge d_{s,n,t},\ \forall (n,t) \in (\mathcal{N}\backslash\mathcal{N}_\mathcal{P})\times\mathcal{T} \\ \qquad \pi_s \in \mathbb{R}_+^{\mathrm{card}(\mathcal{N}_\mathcal{P}\times\mathcal{T})}, f_s, \bar{f}_s \in \mathbb{R}^{\mathrm{card}(\mathcal{A}\times\mathcal{T})}, u_s \in \mathbb{R}_+^{\mathrm{card}(\mathcal{N}\times\mathcal{T})} \end{cases} \tag{5.15}$$

We generate random data with the following parameters:

Nodes	Scenarios	Time steps	Flow cost
10	5	5	$\{0.1, 1, 5\}$
20	$\{50, 250\}$	$\{24, 168\}$	$\{0.1, 1, 5\}$
30	$\{50, 250\}$	$\{24, 168\}$	$\{0.1, 1, 5\}$

Table 5.1: Parameters of the random generated data

For each set of parameters, we generate three random instances with seeds in $\{|\mathcal{N}|+|\mathcal{T}|+|\mathcal{S}|+i : i = 0, 1, 2\}$. All the other parameters are listed hereafter. Notation $[a : b]$ means that the values are samples randomly (with a uniform law) between a and b:

- $\varepsilon = 10^{-3}$

- $\alpha = 3$
- $c^{prod}_{inv,n} \in [1000, 8000],\ \forall n \in \mathcal{N}_P$
- $c^{flow}_{inv,a} \in [1000, 8000],\ \forall a \in \mathcal{A}$
- $g^{prod}_{s,n,t} \in [20, 80],\ \forall (n,t) \in \mathcal{N}_P \times \mathcal{T},\ \forall s \in \mathcal{S}$
- $M = 1000$
- $C = 800$
- $F^{init}_a \in [0, 100],\ \forall a \in \mathcal{A}$
- $\Delta = 0.3$
- $d_{s,n,t} \in [50, 400],\ \forall (n,t) \in (\mathcal{N} \times \mathcal{T}),\ \forall s \in \mathcal{S}$

This leads to a dataset of 81 instances. The minimum and maximum number of variables and constraints of instances for each number of nodes, time steps and scenarios are given in Table 5.2.

Nodes	Time steps	Scenarios	Min - Max Vars	Min - Max Constr
10	5	5	1267 - 1319	1526 - 1676
20	24	50	128836 - 127238	153601 - 160801
30	24	50	190856 - 194459	237601 - 248401
20	24	250	624036 - 642039	768001 - 822001
30	24	250	936053 - 984061	1134001 - 1278001
20	168	50	873636 - 890438	1075201 - 1125601
30	168	50	1352458 - 1402864	1713601 - 1864801
20	168	250	4410037 - 4494039	5502001 - 5754001
30	168	250	6594054 - 6720057	8064001 - 8442001

Table 5.2: Minimum and maximum number of variables and constraints of instances, for each number of nodes, time steps and scenarios.

5.4.2 Numerical results

In order to evaluate the proposed heuristic, we run the four strategies, namely **All-Rand**, **All-Min**, **Opt-Rand**, **Opt-Min** on one core (sequential mode), on an Intel® Xeon® Gold SKL-6130 processor at 2,1 GHz with 96 GB of RAM with the TURBO boost (up to 3.7 GHz). The time limit is fixed to six hours for every algorithm. Julia JuMP is used a modeler, and the solver is CPLEX 12.10 (IBM, 2019). We also solve every bilevel problem directly with CPLEX 12.10, by using the SOS-1 KKT-reformulation with the Julia Package BilevelJuMP (Garcia et al., 2022), referred to as **SOS-1** hereafter.

We first show a plot of the evolution of the expected number of unsatisfied demands according to the minimum investment value Λ on an instance with 10 nodes, 5 time steps and 5 scenarios in Figure 5.7. We sample 500 different values of Λ according to a normal law. The mean is computed as the best feasible solution value obtained with the All-Rand heuristic. The standard deviation is computed so that the probability to sample a value 20% higher than the mean is 0.1. For every value of Λ, we show the expected number of unsatisfied demands given by the solver in an optimal solution to the minimum investment problem. We also evaluate, with the auxiliary subproblems, the minimum number of unsatisfied demands. Finally, we solve the

auxiliary subproblems in maximizing the number of unsatisfied demands.

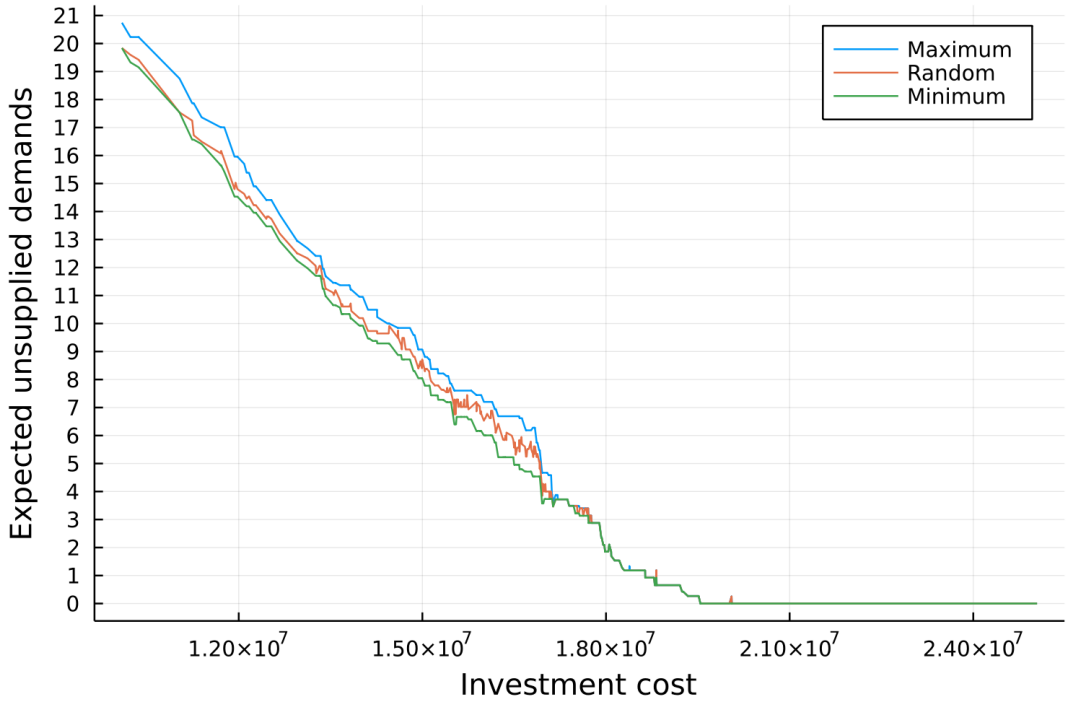


Figure 5.7: Evolution of the expected number of unsatisfied demands according to the minimum investment value Λ, on an instance with 10 nodes, 5 time steps and 5 scenarios.

We first observe that the three curves are not perfectly decreasing, although there is a clear correlation between a higher value of Λ with a lower number of unsatisfied demands. The curves are decreasing almost everywhere, which justifies the use of the proposed heuristic in order to find a good feasible solution. We also observe that for many investment values, there exists several equivalent optimal solutions, as the difference between the minimum and the maximum expected number of unsatisfied demands is positive. This shows that solving the auxiliary subproblems may help to find better solutions.

We present in Table 5.3 the results obtained on the 81 instances we generated. On each line, we write the expected time and expected gap to the best feasible solution found on the three instances with the same parameters. We denote by $\mathcal{P}(N, T, S, g^{flow})$ the set of three instances with N nodes, T time steps, S scenarios and a flow cost of g^{flow}. Let $\mathcal{M}$ be the set of the different methods to solve the instances. We denote by $v_{p,m}$ the best value obtained on problem p with method m, and by $\bar{v}_p = \min_{m \in \mathcal{M}}\{v_{p,m}\}$ the best value obtained with all the methods on problem p. The expected gap shown in Table 5.3 is computed as $\frac{1}{\text{card}(\mathcal{P}(N,T,S,g^{flow}))} \sum_{p \in \mathcal{P}(N,T,S,g^{flow})} \frac{v_{p,m} - \bar{v}_p}{\bar{v}_p}$.

We first remark that, even on the small instances with 10 nodes, 5 time steps and 5 scenarios, the SOS-1 method of CPLEX12.10 to solve the KKT reformulation of the bilevel program is not able to find any feasible solution in 6 hours. This result encourages the use of a heuristic solution method in an operational setting to find feasible solutions to the bilevel stochastic expansion planning problem. We also see, as we could expect, that the All-Min strategy is not able to finish

	Opt-Rand		All-Rand		Opt-Min		All-Min		SOS-1	
Instance	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap
N10-S5-T5-flow01	11.56	0.29%	11.67	0.29%	12.59	0.00%	14.45	0.00%	21600.00	–
N10-S5-T5-flow1	11.76	0.47%	11.86	0.12%	12.58	0.35%	14.36	0.00%	21600.00	–
N10-S5-T5-flow5	11.57	0.57%	11.62	0.25%	12.57	0.00%	14.82	0.00%	21600.00	–
N20-S50-T24-flow01	66.45	0.28%	69.65	0.18%	145.80	0.00%	403.40	0.00%	21600.00	–
N20-S50-T24-flow1	70.95	0.29%	74.24	0.22%	135.28	0.04%	411.81	0.00%	21600.00	–
N20-S50-T24-flow5	73.05	0.38%	76.41	0.18%	156.13	0.05%	451.74	0.00%	21600.00	–
N30-S50-T24-flow01	171.32	2.11%	178.72	0.70%	343.67	0.37%	1217.57	0.00%	21600.00	–
N30-S50-T24-flow1	154.42	0.78%	162.17	0.42%	341.48	0.10%	1168.56	0.00%	21600.00	–
N30-S50-T24-flow5	156.59	1.81%	164.05	0.79%	317.82	1.23%	1174.49	0.00%	21600.00	–
N20-S250-T24-flow01	422.90	0.27%	436.77	0.16%	847.50	0.00%	1729.02	0.00%	21600.00	–
N20-S250-T24-flow1	465.59	0.40%	486.82	0.15%	860.54	0.16%	1901.98	0.00%	21600.00	–
N20-S250-T24-flow5	455.92	0.38%	476.75	0.21%	861.77	0.10%	2036.31	0.00%	21600.00	–
N30-S250-T24-flow01	1203.59	0.25%	1232.15	0.19%	2025.75	0.02%	4456.90	0.00%	21600.00	–
N30-S250-T24-flow1	1147.62	0.29%	1179.92	0.25%	1979.65	0.17%	4367.37	0.00%	21600.00	–
N30-S250-T24-flow5	1226.13	0.29%	1255.79	0.21%	2049.97	0.09%	4822.33	0.00%	21600.00	–
N20-S50-T168-flow01	406.94	14.55%	472.96	0.00%	2439.40	13.90%	19943.22	0.00%	21600.00	–
N20-S50-T168-flow1	412.26	10.25%	470.42	0.08%	2078.35	10.25%	20080.08	0.00%	21600.00	–
N20-S50-T168-flow5	380.15	5.40%	437.25	0.11%	1611.97	5.27%	18766.62	0.00%	21600.00	–
N20-S250-T168-flow01	2355.89	6.20%	2966.55	0.00%	13030.07	5.85%	21600.00	1.74%	21600.00	–
N20-S250-T168-flow1	2468.53	3.96%	3050.63	0.00%	13780.36	3.96%	21600.00	0.47%	21600.00	–
N20-S250-T168-flow5	2643.71	2.02%	3097.23	0.00%	17224.53	1.97%	21600.00	34.22%	21600.00	–
N30-S50-T168-flow01	1092.90	9.80%	1204.49	0.00%	7928.70	9.13%	21600.00	2.07%	21600.00	–
N30-S50-T168-flow1	1019.33	6.28%	1130.36	0.00%	7869.48	5.50%	21600.00	0.00%	21600.00	–
N30-S50-T168-flow5	975.12	3.73%	1084.62	0.00%	5210.29	3.49%	21600.00	3.25%	21600.00	–
N30-S250-T168-flow01	4947.79	15.35%	6430.68	0.00%	21600.00	14.84%	21600.00	63.89%	21600.00	–
N30-S250-T168-flow1	5324.71	16.67%	6669.99	0.00%	16568.41	16.67%	21600.00	65.80%	21600.00	–
N30-S250-T168-flow5	6763.42	10.89%	8364.26	0.00%	19465.20	10.89%	21600.00	26.51%	21600.00	–

Table 5.3: Solving time and gap to the best feasible solution found for the four heuristic strategies and the SOS-1 method to solve the KKT reformulation.

the binary search in 6 hours on the largest instances. Indeed, even if we manage to verify the bilevel feasibility by using the separability of the subproblems, every auxiliary subproblem is a large-scale binary program. Solving the auxiliary subproblems for every first-stage solution evaluated during the algorithm is very time-consuming and the algorithm does not manage to finish the binary search in 6 hours. We can see that, even for the Opt-Min strategy, in which the auxiliary subproblems are only solved for an optimal solution to the minimum investment problem, the heuristic terminates almost at the time limit.

The second remark is that both the All-Rand and the All-Min strategies show the best results, mostly on the large instances with 168 time steps. They can find solution up to 16% better than the Opt strategy. This shows that the use of the Benders decomposition as an exploratory algorithm to find several first-solutions allows to improve the results. The result of the All strategy are always better than their Opt equivalent (All-Rand compared to Opt-Rand and All-Min compared to Opt-Min), except when the All-Min strategy does not finish in the time limit. We see however, for example for the N30-S250-T168-flow1 instances, the All-Min strategy finds the best solutions, whereas it reaches the time limit of 6 hours. Finally, it seems that, depending on the size of the problem, either the All-Min or the All-Rand are the best methods.

We now analyze the sensitivity of the heuristic to the optimality gap of the Benders decom-

position. As we can see on Figure 5.8, when the optimality gap of the Benders decomposition algorithm solved at each iteration is larger, the evolution of the expected number of unsatisfied demands tends to be fuzzier. Indeed, with a larger optimality gap, the Benders decomposition algorithm may return investment solutions in which the total quantity of unsatisfied demands is larger than in an optimal solution, and then may present a larger expected number of nodes with unsatisfied demands. Then, a lower optimality gap tends to produce better results, whereas solving the minimum investment problem at each iteration may be more time-consuming. We therefore present in Table 5.4 the results on our 81 instances for the All-Rand strategy in which the optimality gap to the Benders decomposition algorithm is set to a relative gap of 10^{-6}, as in the results of Table 5.3, and to 10^{-10}.

	All-Rand 10^{-6}		All-Rand 10^{-10}		All-Min	
Instance	Time	Gap	Time	Gap	Time	Gap
N10-S5-T5-flow01	11.67	0.29%	11.72	0.21%	14.45	0.00%
N10-S5-T5-flow1	11.86	0.12%	11.93	0.11%	14.36	0.00%
N10-S5-T5-flow5	11.62	0.32%	11.91	0.26%	14.82	0.07%
N20-S50-T24-flow01	69.65	0.42%	95.20	0.00%	403.40	0.24%
N20-S50-T24-flow1	74.24	0.44%	100.20	0.00%	411.81	0.22%
N20-S50-T24-flow5	76.41	0.28%	102.71	0.00%	451.74	0.10%
N30-S50-T24-flow01	178.72	1.96%	305.37	0.00%	1217.57	1.25%
N30-S50-T24-flow1	162.17	1.41%	288.43	0.01%	1168.56	0.99%
N30-S50-T24-flow5	164.05	0.79%	281.32	0.44%	1174.49	0.00%
N20-S250-T24-flow01	436.77	0.22%	706.59	0.14%	1729.02	0.06%
N20-S250-T24-flow1	486.82	0.19%	737.62	0.22%	1901.98	0.04%
N20-S250-T24-flow5	476.75	0.21%	788.50	0.18%	2036.31	0.00%
N30-S250-T24-flow01	1232.15	0.45%	2321.90	0.00%	4456.90	0.26%
N30-S250-T24-flow1	1179.92	0.35%	2544.04	0.00%	4367.37	0.10%
N30-S250-T24-flow5	1255.79	0.24%	2612.04	0.05%	4822.33	0.03%
N20-S50-T168-flow01	472.96	2.10%	696.77	0.01%	19943.22	2.10%
N20-S50-T168-flow1	470.42	1.13%	680.51	0.00%	20080.08	1.05%
N20-S50-T168-flow5	437.25	0.60%	668.13	0.71%	18766.62	0.49%
N20-S250-T168-flow01	2966.55	0.70%	7003.94	0.34%	21600.00	2.44%
N20-S250-T168-flow1	3050.63	1.86%	5100.85	0.02%	21600.00	2.34%
N20-S250-T168-flow5	3097.23	2.29%	5104.82	0.00%	21600.00	35.35%
N30-S50-T168-flow01	1204.49	1.35%	2178.98	0.00%	21600.00	3.46%
N30-S50-T168-flow1	1130.36	1.26%	2082.04	0.33%	21600.00	1.26%
N30-S50-T168-flow5	1084.62	3.69%	2032.77	0.00%	21600.00	7.06%
N30-S250-T168-flow01	6430.68	6.16%	11648.15	0.00%	21600.00	73.79%
N30-S250-T168-flow1	6669.99	3.30%	12146.08	2.37%	21600.00	70.47%
N30-S250-T168-flow5	8364.26	1.75%	14958.71	33.63%	21600.00	28.79%

Table 5.4: Solving time and gap to the best feasible solution found for the All-Rand heuristic with an optimality gap of 10^{-10} in Benders decomposition compared to All-Rand and All-Min with an optimality gap of 10^{-6}.

As expected, we can see in Table 5.4 that the total computing time for the All-Rand heuristic is significantly increased when the optimality gap of the Benders decomposition is smaller. However, the All-Rand heuristic with an optimality gap of 10^{-10} in the Benders decomposition gives indeed better results than with an optimality gap of 10^{-6}, as we could have expected from the results presented in Figure 5.8. It is competitive with the All-Min heuristic on middle-sized instances, and outperforms the other methods on large sized instances. It seems that, in order to get the best possible results from this heuristic, one should try to reach the best possible optimality gap in the Benders decomposition algorithm. However, a small optimality gap might

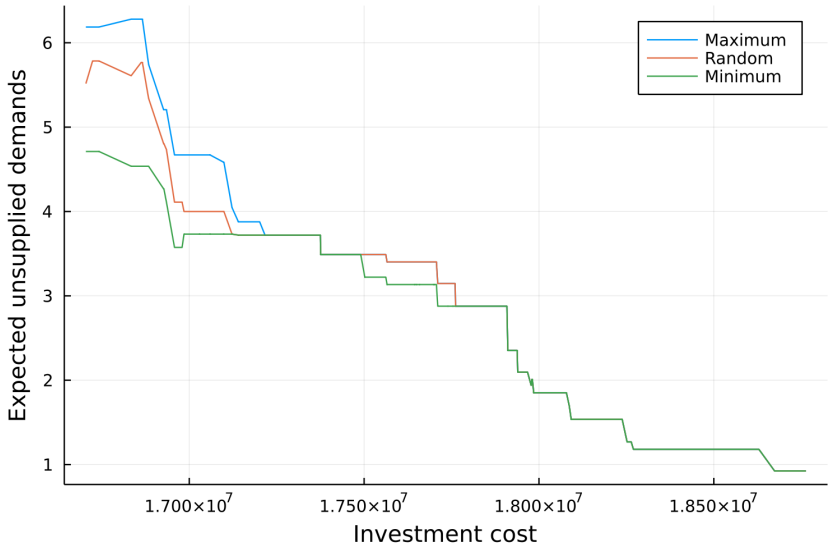
be hard to reach on large-scale instance, and one should find a compromise between the quality of the solution and the computing time.

We finally present in Table 5.5 a comparison between the best feasible solutions found with the proposed heuristic compared with an optimal solution to the stochastic expansion planning problem without the reliability constraint. We see in the column *Unsatisfied* that optimal solutions to the stochastic expansion planning without the reliability constraint have large expected numbers of unsatisfied demands. The best lower bound on the optimal value of the reliable expansion planning problem is evaluated as the maximum between the value to the problem without the reliability constraint, and the best bound obtained in the **SOS-1** method to solve the KKT reformulation. We do not expect these bounds to give a good approximation of the optimal value to the reliable expansion planning problem. However, they allow to compute a first evaluation of the quality of the solutions returned by the heuristic.

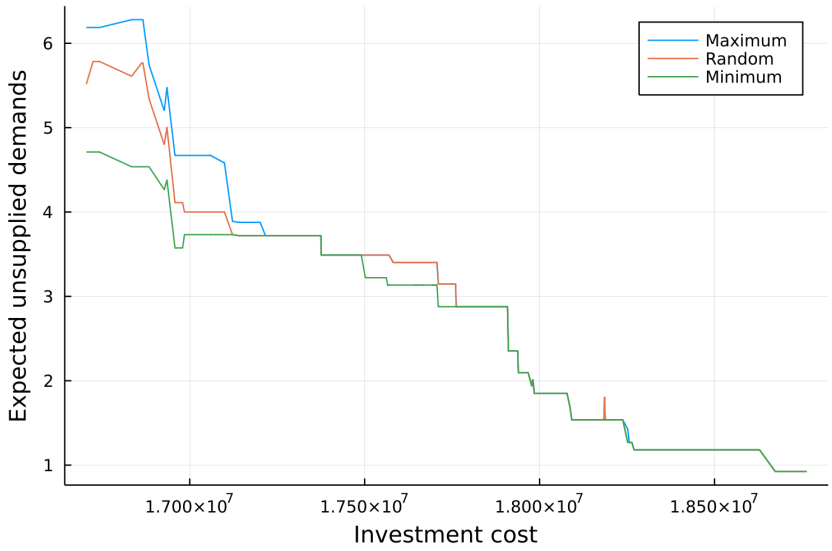
Instance	Unsatisfied	Not reliable value	Best LB	Best Gap
N10-S5-T5-eps01	32.85	$9.256.10^6$	$1.286.10^7$	21.54%
N10-S5-T5-eps1	28.61	$9.258.10^6$	$1.291.10^7$	21.10%
N10-S5-T5-eps5	30.11	$9.266.10^6$	$1.302.10^7$	20.21%
N20-S50-T24-eps01	54.15	$4.184.10^7$	$4.222.10^7$	11.80%
N20-S50-T24-eps1	56.54	$4.193.10^7$	$4.232.10^7$	11.73%
N20-S50-T24-eps5	56.27	$4.231.10^7$	$4.273.10^7$	11.85%
N30-S50-T24-eps01	83.24	$5.998.10^7$	$6.014.10^7$	18.81%
N30-S50-T24-eps1	83.92	$6.011.10^7$	$6.035.10^7$	18.20%
N30-S50-T24-eps5	87.03	$6.065.10^7$	$6.090.10^7$	16.81%
N20-S250-T24-eps01	55.93	$4.087.10^7$	$4.087.10^7$	13.53%
N20-S250-T24-eps1	54.47	$4.095.10^7$	$4.095.10^7$	13.41%
N20-S250-T24-eps5	56.97	$4.130.10^7$	$4.130.10^7$	13.35%
N30-S250-T24-eps01	65.98	$5.801.10^7$	$5.801.10^7$	14.85%
N30-S250-T24-eps1	66.45	$5.815.10^7$	$5.815.10^7$	14.73%
N30-S250-T24-eps5	65.79	$5.875.10^7$	$5.875.10^7$	14.61%
N20-S50-T168-eps01	28.22	$7.638.10^7$	$7.638.10^7$	8.84%
N20-S50-T168-eps1	30.74	$7.695.10^7$	$7.695.10^7$	11.22%
N20-S50-T168-eps5	30.81	$7.941.10^7$	$7.941.10^7$	7.02%
N30-S50-T168-eps01	46.97	$1.129.10^8$	$1.129.10^8$	15.34%
N30-S50-T168-eps1	42.99	$1.138.10^8$	$1.138.10^8$	13.89%
N30-S50-T168-eps5	45.93	$1.173.10^8$	$1.173.10^8$	9.64%
N20-S250-T168-eps01	26.74	$7.456.10^7$	$7.456.10^7$	8.90%
N20-S250-T168-eps1	25.26	$7.518.10^7$	$7.518.10^7$	9.08%
N20-S250-T168-eps5	26.29	$7.779.10^7$	$7.779.10^7$	6.73%
N30-S250-T168-eps01	53.26	$1.212.10^8$	$1.212.10^8$	16.20%
N30-S250-T168-eps1	45.64	$1.223.10^8$	$1.223.10^8$	19.37%
N30-S250-T168-eps5	49.27	$1.268.10^8$	$1.268.10^8$	13.88%

Table 5.5: Comparison between the solution to the stochastic expansion planning problem without the reliability constraint and the best bilevel solution found. *Unsatisfied* represents the expected number of unsatisfied demands in an optimal solution without the reliability constraint. *Not reliable value* is the optimal value to the stochastic expansion planning problem without the reliability constraint. *Best LB* is the best lower bound found, computed as the maximum between the Stochastic Value and the best bound found in the **SOS-1** method. *Best Gap* is the gap between the best feasible solution found with our heuristic and the Best LB.

↓ Optimality gap of 10^{-10} in the Benders decomposition algorithm



↓ Optimality gap of 10^{-8} in the Benders decomposition algorithm



↓ Optimality gap of 10^{-6} in the Benders decomposition algorithm

Figure 5.8: A zoom on the evolution of the solutions to the minimum investment problem according to the minimum investment value, for three different relative optimality gaps in Benders decomposition : 10^{-10} in the upper figure, 10^{-8}, and 10^{-6} in the last figure, on an instance with 10 nodes, 5 time steps and 5 scenarios.

5.5 Conclusion

We presented in this chapter a new model to take into account a reliability constraint is stochastic expansion planning problems to limit the expected number of nodes and time steps at which the demands are not fully satisfied in an optimal solution. We have shown that modeling such a constraint induces a bilevel formulation. We have also presented that the relaxation of the optimality conditions of the lower-level, leading to a single-level linear formulation which could be seen as the extensive formulation of the stochastic program in which we added the reliability constraint, is not anymore a two-stage stochastic program. Such a single-level linear formulation may lead to invalid solutions, as the ratio between the cost of the recourse variables in an optimal solution and the optimal recourse value may be unbounded. Finally, because of the complexity of the resulting program, we proposed a heuristic solution method based on iterative solutions of stochastic expansion planning problems with minimum investment constraints. This solution method allows to find feasible solutions even on large-scale instances with several millions variables and constraints in less than 6 hours.

Because of the numerical complexity of the studied problem, we did not produce good enough lower bounds on the optimal value in order to evaluate the quality of the solutions given by the heuristics. Finding such lower-bounds is a important perspective to further evaluate the efficiency of the method. The high-point relaxation may give such good lower bounds. Although being a very large-scale two-stage stochastic mixed-integer program, its structure seems adapted to solution methods based on Benders decomposition, by setting all variables δ_s in the master program.

We presented numerical results on instances with continuous investment variables. However, the method remains valid if some investment variables are integer. As the Benders cuts found during one iteration of the heuristic are valid for any minimum investment value, it is possible to solve at each iteration a Benders decomposition with a Branch&Cut method using all the previously found cuts as a warm-start. Finally, this solution method will be applied in real-world instances from RTE, in order to evaluate if it allows to produce better solutions than the ones produced by transmission system experts.

Chapter 6

Conclusion

We have studied in this thesis two different problems arising in stochastic expansion planning studies performed at RTE from a mathematical programming point of view.

First, we have presented in Chapter 4 an efficient algorithm to solve the multicut reformulation of two-stage stochastic linear programs formulated with a large number of scenarios. Such problems arise in stochastic expansion planning problems as the number of uncertain parameters may be large. The presented *Benders by batch* algorithm neither requires the fixed recourse hypothesis nor the second-stage cost function to be deterministic. It is based on a sequential stopping criterion that allows to not solve all the subproblems at most iterations. We also presented a general framework to stabilize this algorithm and two stabilization procedures. We have shown in an extensive numerical study that the proposed algorithm outperforms some classical algorithms from the literature, namely the level bundle method and a Benders decomposition with in-out stabilization and static cut aggregation. Any acceleration technique that does not require an upper bound on the optimal value can be directly applied to the Benders by batch algorithm, such as dual stabilization techniques (Magnanti and Wong, 1981; Sherali and Lunday, 2013) or asynchronous parallelization methods (Linderoth and Wright, 2003).

We presented in Chapter 4 a random way to generate batches of subproblems. Depending on the solved problem, finding better ways to generate the batches of subproblems could improve significantly the performance of the algorithm. Clustering methods that find groups of subproblems based on their objective functions and right-hand sides can be an interesting perspective. The choice of the permutation defining in what order the batches of subproblems are solved at each iteration can also have an impact on the performance of the algorithm. The idea is to solve the batches of subproblems that are the worse approximated in the relaxed master problem first, in order to exclude non-optimal first-stage solutions as fast as possible. This can be based on criteria such as the distance between the separation point and the first-stage solutions at which cuts were generated for a given batch of subproblems, or with more complex oracles such as neural networks if there are sufficient data to train them.

There are other interesting and valuable perspectives to improve this algorithm. First, as only a few number of subproblems are solved at most iteration, the algorithm does not provide

any upper bound on the optimal value of the problem. This means that, if the algorithm stops at its time limit before finding an optimal solution, it does not provides any evaluated feasible solution. This drawback can be circumvented by solving all the subproblems at some iterations of the algorithm. Secondly, the algorithm could be extended to problems with integer first-stage variables in a Branch&Cut framework. The core point of such an extension should be to solve all the subproblems when the solution of the relaxed master problem at a given node of the Branch&Bound tree is integer and then may improve the upper bound. Finally, the proposed procedure could be extended to any problems with convex and separable subproblems.

In Chapter 5, we have extended the models used in prospective studies performed at RTE to take into account a reliability constraint. This constraint limits the number of nodes and time steps at which the demands are not fully satisfied in an optimal solution, in expectation over the scenarios. We have shown that adding this constraint in the models implies a bilevel formulation. We have also shown that solving the high-point relaxation may lead to inconsistent solutions. Then, we have presented a heuristic solution method based on Benders decomposition and a binary search on the investment costs. We have performed a numerical study on large-scale instance with up to several million variables and constraints. The proposed heuristic finds feasible solutions in 6 hours whereas the classical SOS-1 method to solve the KKT reformulation is not able to find any feasible solution.

An important limitation of the numerical experiment presented in Section 5.4 is that we could not compute tight lower bounds on the optimal values of the reliable expansion planning problems. The quality of the solutions found with the proposed heuristic could not be evaluated. The high-point relaxation can give a tighter lower bound on the optimal value of the problem. As it is a large-scale mixed-integer program, this problem is hard to solve. Developing tailored solution methods to solve the high-point relaxation is an interesting perspective. By affecting the investment variables and the binary variables δ in a master program, the problem can be solved with a Benders decomposition algorithm. The number of binary variables is large. Generating only the binary variables $\delta_{s,n,t}$ at nodes and time steps where unsatisfied demands are observed could lead to solution methods in which the number of binary variables in the program is reasonable. Finally, the problem could be solved with a Dantzig-Wolfe reformulation and a column generation algorithm. In such an algorithm, each column represents a global affectation of the authorized unsatisfied demands.

In the heuristic proposed in Section 5.3, after solving only a subset of subproblems, if the expected unsatisfied demands in their recourse solutions already exceeds the maximal number of authorized unsatisfied demands, we know that the solution is not bilevel feasible. Then, solving first the subproblems with a large number of unsatisfied demands is an interesting perspective, an the heuristic could be done with an adapted Benders by batch algorithm.

The question of the existence of an exact algorithm to solve the reliable expansion planning problem is still open, and is a very interesting research perspective. We proposed in this work a methods based on a binary search on the investment costs. One could search for feasible solutions

to this problem with a partition of the first-stage feasible space, and a procedure to refine this partition. A tree-search method, choosing at each node a first-stage variable, and generating two child nodes by defining a bound on the variable is a promising method. At each node, on could evaluate one first-stage solution in the resulting feasible space.

Finally, the proposed methods have to be tested on real-size instances at RTE. The Benders by batch algorithm have already been tested on a few instances and showed noticeable improvements on the computing times compared to the classical Benders decomposition algorithm currently used at RTE. This now has to be tested on other prospective studies instances to see if it allows to solve instances with larger number of subproblems. In the case of RTE instances, it is possible to generate batches of subproblems by putting together subproblems associated with winter weeks together, and summer weeks together for example.

The heuristic proposed in Chapter 5 has not currently been tested on RTE instances. Comparing the proposed feasible solutions to the solutions found by experts is a first step to validate the method. Applying experts rules in the proposed heuristic could also significantly improved the quality of the solutions.

Bibliography

Alizadeh, B. and Jadid, S. (2011). Reliability constrained coordination of generation and transmission expansion planning in power systems using mixed integer programming. *IET Generation, Transmission & Distribution*, 5(9):948–960.

Alvarez Lopez, J., Ponnambalam, K., and Quintana, V. H. (2007). Generation and Transmission Expansion Under Risk Using Stochastic Programming. *IEEE Transactions on Power Systems*, 22(3):1369–1378.

Bard, J. F. and Moore, J. T. (1990). A Branch and Bound Algorithm for the Bilevel Programming Problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2):281–292.

Bard, J. F. and Moore, J. T. (1992). An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)*, 39(3):419–435.

Baringo, L. and Conejo, A. J. (2012). Transmission and Wind Power Investment. *IEEE Transactions on Power Systems*, 27(2):885–893.

Basciftci, B. and Van Hentenryck, P. (2020). Bilevel Optimization for On-Demand Multimodal Transit Systems. In Hebrard, E. and Musliu, N., editors, *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Lecture Notes in Computer Science, pages 52–68, Cham. Springer International Publishing.

Belieres, S., Hewitt, M., Jozefowiez, N., and Semet, F. (2022). Meta partial benders decomposition for the logistics service network design problem. *European Journal of Operational Research*, 300(2):473–489.

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

Ben-Ayed, O., Boyce, D. E., and Blair, C. E. (1988). A general bilevel linear programming formulation of the network design problem. *Transportation Research Part B: Methodological*, 22(4):311–318.

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

Bialas, W. F. and Karwan, M. H. (1984). Two-Level Linear Programming. *Management Science*, 30(8):1004–1020.

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.

Birge, J. R. and Louveaux, F. (2011). *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2nd ed edition.

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

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

Bracken, J. and McGill, J. T. (1973). Mathematical Programs with Optimization Problems in the Constraints. *Operations Research*, 21(1):37–44.

Bracken, J. and McGill, J. T. (1974). Defense Applications of Mathematical Programs with Optimization Problems in the Constraints. *Operations Research*, 22(5):1086–1096.

Braga, A. and Saraiva, J. (2005). A multiyear dynamic approach for transmission expansion planning and long-term marginal costs computation. *IEEE Transactions on Power Systems*, 20(3):1631–1639.

Brotcorne, L., Labbé, M., Marcotte, P., and Savard, G. (2008). Joint Design and Pricing on a Network. *Operations Research*, 56(5):1104–1115.

Choi, J., Tran, T., El-Keib, A., Thomas, R., Oh, H., and Billinton, R. (2005). A method for transmission system expansion planning considering probabilistic reliability criteria. *IEEE Transactions on Power Systems*, 20(3):1606–1615.

Conforti, M., Cornuéjols, G., and Zambelli, G. (2014). *Integer Programming*, volume 271 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham.

Contreras, J. and Wu, F. (2000). A kernel-oriented algorithm for transmission expansion planning. *IEEE Transactions on Power Systems*, 15(4):1434–1440.

Crainic, T. G., Hewitt, M., Maggioni, F., and Rei, W. (2021). Partial Benders Decomposition: General Methodology and Application to Stochastic Network Design. *Transportation Science*, 55(2):414–435.

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

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

DeNegre, S. T. and Ralphs, T. K. (2009). A Branch-and-cut Algorithm for Integer Bilevel Linear Programs. In Chinneck, J. W., Kristjansson, B., and Saltzman, M. J., editors, *Operations Research and Cyber-Infrastructure*, Operations Research/Computer Science Interfaces, pages 65–78, Boston, MA. Springer US.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Erdos, P. and Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60.

Faísca, N. P., Dua, V., Rustem, B., Saraiva, P. M., and Pistikopoulos, E. N. (2007). Parametric global optimisation for bilevel programming. *Journal of Global Optimization*, 38(4):609–623.

Fischetti, M., Ljubić, I., Monaci, M., and Sinnl, M. (2018). On the use of intersection cuts for bilevel optimization. *Mathematical Programming*, 172(1):77–103.

Fischetti, M., Ljubić, I., and Sinnl, M. (2016). Redesigning Benders Decomposition for Large-Scale Facility Location. *Management Science*, 63(7):2146–2162.

Fischetti, M. and Salvagnin, D. (2010). An In-Out Approach to Disjunctive Optimization. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140, pages 136–140. Springer Berlin Heidelberg, Berlin, Heidelberg.

Fontaine, P. and Minner, S. (2014). Benders Decomposition for Discrete–Continuous Linear Bilevel Problems with application to traffic network design. *Transportation Research Part B: Methodological*, 70:163–172.

Forcier, M. and Leclère, V. (2021). Generalized adaptive partition-based method for two-stage stochastic linear programs : Convergence and generalization. *arXiv:2109.04818 [math]*.

Fortuny-Amat, J. and McCarl, B. (1981). A Representation and Economic Interpretation of a Two-Level Programming Problem. *Journal of the Operational Research Society*, 32(9):783–792.

Gangammanavar, H., Liu, Y., and Sen, S. (2018). Stochastic Decomposition for Two-stage Stochastic Linear Programs with Random Cost Coefficients. Technical report.

Gangammanavar, H., Sen, S., and Zavala, V. M. (2016). Stochastic Optimization of Sub-Hourly Economic Dispatch With Wind Energy. *IEEE Transactions on Power Systems*, 31(2):949–959.

Garces, L. P., Conejo, A. J., Garcia-Bertrand, R., and Romero, R. (2009). A Bilevel Approach to Transmission Expansion Planning Within a Market Environment. *IEEE Transactions on Power Systems*, 24(3):1513–1522.

Garcia, J. D., Bodin, G., and Street, A. (2022). BilevelJuMP.jl: Modeling and Solving Bilevel Optimization in Julia. *arXiv:2205.02307*.

Gil, H. A. and da Silva, E. L. (2001). A reliable approach for solving the transmission network expansion planning problem using genetic algorithms. *Electric Power Systems Research*, 58(1):45–51.

González-Díaz, J., González-Rodríguez, B., Leal, M., and Puerto, J. (2021). Global optimization for bilevel portfolio design: Economic insights from the Dow Jones index. *Omega*, 102.

Grimm, V., Kleinert, T., Liers, F., Schmidt, M., and Zöttl, G. (2019). Optimal price zones of electricity markets: A mixed-integer multilevel model and global solution approaches. *Optimization Methods and Software*, 34(2):406–436.

Hakimi, S. L. (1962). On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph. I. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506.

Higle, J. L. and Sen, S. (1991). Stochastic Decomposition : An Algorithm for Two-Stage Linear Programms with Recours. *Mathematics of Operations Research*, 16(3):447–669.

Higle, J. L. and Sen, S. (1994). Finite master programs in regularized stochastic decomposition. *Mathematical Programming*, 67(1):143–168.

Hu, X. M. and Ralph, D. (2004). Convergence of a Penalty Method for Mathematical Programming with Complementarity Constraints. *Journal of Optimization Theory and Applications*, 123(2):365–390.

IBM (2019). IBM (2019) IBM ILOG CPLEX 12.10 User's Manual (IBM ILOG CPLEX Division, Incline Village, NV). page 596.

Jin, S. and Ryan, S. M. (2011). Capacity Expansion in the Integrated Supply Network for an Electricity Market. *IEEE Transactions on Power Systems*, 26(4):2275–2284.

Kelley, J. E., J. (1960). The Cutting-Plane Method for Solving Convex Programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712.

Kleinert, T., Labbé, M., Ljubić, I., and Schmidt, M. (2021). A Survey on Mixed-Integer Programming Techniques in Bilevel Optimization. *EURO Journal on Computational Optimization*, 9:100007.

Kleinert, T., Labbé, M., Plein, F., and Schmidt, M. (2020). Technical Note—There's No Free Lunch: On the Hardness of Choosing a Correct Big-M in Bilevel Optimization. *Operations Research*, 68(6):1716–1721.

Kleinert, T. and Schmidt, M. (2019). Global optimization of multilevel electricity market models including network design and graph partitioning. *Discrete Optimization*, 33:43–69.

Labbé, M., Marcotte, P., and Savard, G. (1998). A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing. *Management Science*, 44(12-part-1):1608–1622.

Labbé, M. and Violin, A. (2016). Bilevel programming and price setting problems. *Annals of Operations Research*, 240(1):141–169.

Leal, M., Ponce, D., and Puerto, J. (2020). Portfolio problems with two levels decision-makers: Optimal portfolio selection with pricing decisions on transaction costs. *European Journal of Operational Research*, 284(2):712–727.

LeBlanc, L. J. and Boyce, D. E. (1986). A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows. *Transportation Research Part B: Methodological*, 20(3):259–265.

Lei, Y., Zhang, P., Hou, K., Jia, H., Mu, Y., and Sui, B. (2018). An Incremental Reliability Assessment Approach for Transmission Expansion Planning. *IEEE Transactions on Power Systems*, 33(3):2597–2609.

Leite da Silva, A. M., Rezende, L. S., Manso, L. A. F., and Anders, G. J. (2010). Transmission expansion planning: A discussion on reliability and "N-1" security criteria. In *2010 IEEE 11th International Conference on Probabilistic Methods Applied to Power Systems*, pages 244–251.

Lemaréchal, C., Nemirovskii, A., and Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, 69(1):111–147.

Li, Z., Wu, W., Tai, X., and Zhang, B. (2021). A Reliability-Constrained Expansion Planning Model for Mesh Distribution Networks. *IEEE Transactions on Power Systems*, 36(2):948–960.

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.

Liu, W., Zheng, Z., and Cai, K.-Y. (2013). Bi-level programming based real-time path planning for unmanned aerial vehicles. *Knowledge-Based Systems*, 44:34–47.

Louveaux, F. and Smeers, Y. (1988). Optimal Investments for Electricity Generation: A Stochastic Model and a Test-Problem. In *Numerical Techniques for Stochastic Optimization*, Y. Ermoliev andR.J.-B. Wets (Eds.), pages 445–454, Berlin. Springer-Verlag,.

Lozano, L. and Smith, J. C. (2017). A Value-Function-Based Exact Approach for the Bilevel Mixed-Integer Programming Problem. *Operations Research*, 65(3):768–786.

Lumbreras, S. and Ramos, A. (2016). The new challenges to transmission expansion planning. Survey of recent practice and literature review. *Electric Power Systems Research*, 134:19–29.

Lv, Y., Hu, T., Wang, G., and Wan, Z. (2007). A penalty function method based on Kuhn–Tucker condition for solving linear bilevel programming. *Applied Mathematics and Computation*, 188(1):808–813.

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.

Marcotte, P. (1986). Network design problem with congestion effects: A case of bilevel programming. *Mathematical Programming*, 34(2):142–162.

Micheli, G. and Vespucci, M. T. (2021). A Survey on Modeling Approaches for Generation and Transmission Expansion Planning Analysis. In Al-Baali, M., Purnama, A., and Grandinetti, L., editors, *Numerical Analysis and Optimization*, Springer Proceedings in Mathematics & Statistics, pages 183–207, Cham. Springer International Publishing.

Mitsos, A., Lemonidis, P., and Barton, P. I. (2008). Global solution of bilevel programs with a nonconvex inner program. *Journal of Global Optimization*, 42(4):475–513.

Motto, A., Arroyo, J., and Galiana, F. (2005). A mixed-integer LP procedure for the analysis of electric grid security under disruptive threat. *IEEE Transactions on Power Systems*, 20(3):1357–1365.

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

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

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

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

Pineda, S., Bylling, H., and Morales, J. M. (2018). Efficiently solving linear bilevel programming problems using off-the-shelf optimization software. *Optimization and Engineering*, 19(1):187–211.

Pineda, S. and Morales, J. M. (2019). Solving Linear Bilevel Problems Using Big-Ms: Not All That Glitters Is Gold. *IEEE Transactions on Power Systems*, 34(3):2469–2471.

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

Ramirez-Pico, C. and Moreno, E. (2021). Generalized adaptive partition-based method for two-stage stochastic linear programs with fixed recourse. *Mathematical Programming*.

Reisi, M., Gabriel, S. A., and Fahimnia, B. (2019). Supply chain competition on shelf space and pricing for soft drinks: A bilevel optimization approach. *International Journal of Production Economics*, 211:237–250.

Roldán, C., Sánchez de la Nieta, A. A., García-Bertrand, R., and Mínguez, R. (2018). Robust dynamic transmission and renewable generation expansion planning: Walking towards sustainable systems. *International Journal of Electrical Power & Energy Systems*, 96:52–63.

RTE (2021). Energy Pathways to 2050, Key Results.

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

Ryu, J.-H., Dua, V., and Pistikopoulos, E. N. (2004). A bilevel programming framework for enterprise-wide process networks under uncertainty. *Computers & Chemical Engineering*, 28(6):1121–1129.

Saharidis, G. K. and Ierapetritou, M. G. (2009). Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44(1):29–51.

Scholtes, S. (2001). Convergence Properties of a Regularization Scheme for Mathematical Programs with Complementarity Constraints. *SIAM Journal on Optimization*, 11(4):918–936.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. John Wiley & Sons.

Sen, S., Doverspike, R. D., and Cosares, S. (1994). Network planning with random demand. *Telecommunication Systems*, 3(1):11–30.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2009). *Lectures on Stochastic Programming: Modeling and Theory*. Society for Industrial and Applied Mathematics.

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

Sherali, H. D. and Lunday, B. J. (2013). On generating maximal nondominated Benders cuts. *Annals of Operations Research*, 210(1):57–72.

Siddiqui, S. and Gabriel, S. A. (2013). An SOS1-Based Approach for Solving MPECs with a Natural Gas Market Application. *Networks and Spatial Economics*, 13(2):205–227.

Song, Y. and Luedtke, J. (2015). An Adaptive Partition-Based Approach for Solving Two-Stage Stochastic Programs with Fixed Recourse. *SIAM Journal on Optimization*, 25(3):1344–1367.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.

Tahernejad, S., Ralphs, T. K., and DeNegre, S. T. (2020). A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation. *Mathematical Programming Computation*, 12(4):529–568.

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 der Weijde, A. H. and Hobbs, B. F. (2012). The economics of planning electricity transmission to accommodate renewables: Using two-stage optimisation to evaluate flexibility and the cost of disregarding uncertainty. *Energy Economics*, 34(6):2089–2101.

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

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

Viger, F. and Latapy, M. (2005). Efficient and Simple Generation of Random Simple Connected Graphs with Prescribed Degree Sequence. In Wang, L., editor, *Computing and Combinatorics*, pages 440–449, Berlin, Heidelberg. Springer Berlin Heidelberg.

von Stackelberg, H. and Von, S. H. (1952). *The Theory of the Market Economy*. William Hodge, Oxford, oxford university press edition.

Wang, L. and Xu, P. (2017). The Watermelon Algorithm for The Bilevel Integer Linear Programming Problem. *SIAM Journal on Optimization*, 27(3):1403–1430.

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

White, D. J. and Anandalingam, G. (1993). A penalty function approach for solving bi-level linear programs. *Journal of Global Optimization*, 3(4):397–419.

Wogrin, S., Pineda, S., and Tejada-Arango, D. A. (2020). Applications of Bilevel Optimization in Energy and Electricity Markets. In Dempe, S. and Zemkoho, A., editors, *Bilevel Optimization: Advances and Next Challenges*, Springer Optimization and Its Applications, pages 139–168. Springer International Publishing, Cham.

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.

Wu, Z., Du, X., Gu, W., Zhang, X.-P., and Li, J. (2018). Automatic Selection Method for Candidate Lines in Transmission Expansion Planning. *IEEE Access*, 6:11605–11613.

Xu, P. and Wang, L. (2014). An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions. *Computers & Operations Research*, 41:309–318.

Ye, J. J. and Zhu, D. L. (1995). Optimality conditions for bilevel programming problems. *Optimization*, 33(1):9–27.

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.

Yue, D. and You, F. (2017). Stackelberg-game-based modeling and optimization for supply chain design and operations: A mixed integer bilevel programming framework. *Computers & Chemical Engineering*, 102:81–95.

Zeng, B. and An, Y. (2014). Solving Bilevel Mixed Integer Program by Reformulations and Decomposition – Optimization Online. Technical report.

Zhang, X. and Conejo, A. J. (2018). Candidate line selection for transmission expansion planning considering long- and short-term uncertainty. *International Journal of Electrical Power & Energy Systems*, 100:320–330.

Zverovich, V., Fábián, C. I., Ellison, E. F. D., and Mitra, G. (2012). A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition. *Mathematical Programming Computation*, 4(3):211–238.

Appendix A

Appendices of chapter 4

A.1 Detailed benchmark algorithms

Algorithm A.1 describes our implementation of **In-out monocut** (`cutAggr=True`) and **In-out multicut** (`cutAggr=False`).

Algorithm A.1: The Benders decomposition algorithm with in-out stabilization

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

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

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

- $k \leftarrow k + 1$
- Solve $(RMP)^{(k)}$ and retrieve $\check{x}^{(k)}, (\check{\theta}_s^{(k)})_{s\in S}$
- $LB^{(k)} \leftarrow c^\top \check{x}^{(k)} + \sum_{s\in S} p_s \check{\theta}^{(k)}$
- $x^{(k)} \leftarrow \alpha_k \check{x}^{(k)} + (1 - \alpha_k)\hat{x}^{(k)}$
- **for** $s \in S$ **do**
  - Solve $(SP(x^{(k)}, s))$ and retrieve π_s an extreme point of Π_s
- **if** `cutAggr` **then**
  - Add $\sum_{s\in S} p_s \theta_s \geq \sum_{s\in S} p_s \pi_s^\top (d_s - T_s x)$
- **else**
  - **for** $s \in S$ **do**
    - Add $\theta_s \geq \pi_s^\top (d_s - T_s x)$ to $(RMP)^{(k)}$
- **if** $UB^{(k-1)} > c^\top x^{(k)} + \sum_{s\in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$ **then**
  - $UB^{(k)} \leftarrow c^\top x^{(k)} + \sum_{s\in S} p_s \pi_s^\top (d_s - T_s x^{(k)})$
  - $\hat{x}^{(k+1)} \leftarrow x^{(k)}$
  - $\alpha_{k+1} \leftarrow \min\{1.0, 1.2\alpha_k\}$
- **else**
  - $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$, $UB^{(k)} \leftarrow UB^{(k-1)}$
  - $\alpha_{k+1} \leftarrow \max\{0.1, 0.8\alpha_k\}$
- $(RMP)^{(k+1)} \leftarrow (RMP)^{(k)}$

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

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

and f_{lev} is the following:

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

We denote by $(RQMP)^{(k)}(\hat{x}, f_{lev})$ its relaxation at iteration k of the algorithm and by $\kappa \in (0,\lambda)$ a acceptation tolerance to update the stability center. Algorithm A.2 describes our implementation of **Level bundle**.

Algorithm A.2: Level bundle method

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

Initialization: $k \leftarrow 0$, $UB^{(0)} \leftarrow c^\top x^{(0)} + \sum_{s\in S} p_s\pi_s^\top(d_s - T_s\hat{x}^{(0)})$, $\hat{x}^{(1)} \leftarrow x^{(0)}$
while $UB^{(k)} > LB^{(k)} + \epsilon$ **do**
  $k \leftarrow k+1$
  $f_{lev}^{(k)} = (1-\lambda)UB^{(k-1)} + \lambda LB^{(k-1)}$
  Solve $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$
  if $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$ *is infeasible* **then**
    $LB^{(k)} \leftarrow f_{lev}(k)$
    $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$
    $UB^{(k)} \leftarrow UB^{(k-1)}$
  else
    Retrieve $x^{(k)}$ solution to $(RQMP)^{(k)}(\hat{x}^{(k)}, f_{lev}^{(k)})$
    for $s \in S$ **do**
      Solve $(SP(x^{(k)}, s))$ and retrieve π_s an extreme point of Π_s
    Add $\sum_{s\in S} p_s\theta_s \geq \sum_{s\in S} p_s\pi_s^\top(d_s - T_s x)$
    if $c^\top x^{(k)} + \sum_{s\in S} p_s\pi_s^\top(d_s - T_s x^{(k)}) < (1-\kappa)UB^{(k-1)} + \kappa f_{lev}^{(k)}$ **then**
      $UB^{(k)} \leftarrow c^\top x^{(k)} + \sum_{s\in S} p_s\pi_s^\top(d_s - T_s x^{(k)})$
      $\hat{x}^{(k+1)} \leftarrow x^{(k)}$
    else
      $\hat{x}^{(k+1)} \leftarrow \hat{x}^{(k)}$
      $UB^{(k)} \leftarrow UB^{(k-1)}$
    $LB^{(k)} \leftarrow LB^{(k-1)}$
  $(RQMP)^{(k+1)} \leftarrow (RQMP)^{(k)}$
Return $\hat{x}^{(k+1)}$

A.2 Detailed numerical results

This section show the detailed numerical results.

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

	Classic multicut		Classic monocut		BbB 1%		BbB 5%		BbB 10%		BbB 20%	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	2	3.2	0.81	1.3	2	2.8	0.91	1.5	0.75	1.2	**0.62**	**1.0**
LandS-N1000-s1001	2	2.9	0.72	1.0	2	2.4	0.86	1.2	0.74	1.1	**0.70**	**1.0**
LandS-N1000-s1002	2	3.0	0.72	1.1	2	2.9	0.71	1.1	**0.65**	**1.0**	0.66	1.0
LandS-N5000-s5000	11	1.6	9	1.3	12	1.9	8	1.2	7	1.1	**7**	**1.0**
LandS-N5000-s5001	10	1.6	10	1.6	15	2.5	8	1.3	6	1.1	**6**	**1.0**
LandS-N5000-s5002	11	1.9	9	1.5	13	2.2	8	1.3	7	1.2	**6**	**1.0**
LandS-N10000-s10000	22	1.1	26	1.3	41	2.0	25	1.2	**20**	**1.0**	21	1.0
LandS-N10000-s10001	22	1.1	30	1.5	36	1.8	25	1.2	22	1.1	**20**	**1.0**
LandS-N10000-s10002	20	1.1	30	1.7	37	2.0	25	1.4	22	1.2	**18**	**1.0**
LandS-N20000-s20000	**49**	**1.0**	96	1.9	134	2.7	86	1.7	78	1.6	71	1.4
LandS-N20000-s20001	**43**	**1.0**	119	2.8	130	3.0	92	2.1	77	1.8	71	1.7
LandS-N20000-s20002	**44**	**1.0**	99	2.2	125	2.8	90	2.0	85	1.9	73	1.7
gbd-N1000-s1000	2	2.7	0.95	1.4	2	3.3	**0.68**	**1.0**	0.78	1.1	0.95	1.4
gbd-N1000-s1001	2	3.7	0.90	1.4	2	3.8	**0.65**	**1.0**	0.90	1.4	0.94	1.5
gbd-N1000-s1002	2	3.6	0.96	1.6	2	3.7	**0.62**	**1.0**	0.83	1.3	0.99	1.6
gbd-N5000-s5000	13	2.0	10	1.7	18	2.9	**6**	**1.0**	7	1.2	8	1.4
gbd-N5000-s5001	11	1.9	10	1.7	14	2.3	**6**	**1.0**	7	1.1	8	1.3
gbd-N5000-s5002	12	1.8	11	1.6	15	2.4	**6**	**1.0**	7	1.1	9	1.3
gbd-N10000-s10000	24	1.2	34	1.8	54	2.8	**19**	**1.0**	21	1.1	26	1.4
gbd-N10000-s10001	24	1.3	32	1.7	41	2.2	**19**	**1.0**	24	1.3	26	1.4
gbd-N10000-s10002	23	1.2	32	1.7	46	2.4	**19**	**1.0**	22	1.1	24	1.2
gbd-N20000-s20000	**48**	**1.0**	119	2.5	97	2.0	63	1.3	71	1.5	86	1.8
gbd-N20000-s20001	**51**	**1.0**	120	2.3	100	2.0	64	1.2	73	1.4	90	1.8
gbd-N20000-s20002	**47**	**1.0**	125	2.7	92	2.0	57	1.2	70	1.5	85	1.8
ssn-N1000-s1000	2279	552.2	7	1.7	6	1.3	**4**	**1.0**	5	1.1	5	1.2
ssn-N1000-s1001	2720	679.7	7	1.8	6	1.6	**4**	**1.0**	4	1.0	5	1.2
ssn-N1000-s1002	2226	602.8	7	1.8	6	1.8	**4**	**1.0**	4	1.1	5	1.3
ssn-N5000-s5000	13425	580.9	62	2.7	31	1.3	**23**	**1.0**	33	1.4	33	1.4
ssn-N5000-s5001	14260	631.1	45	2.0	33	1.5	**23**	**1.0**	27	1.2	31	1.4
ssn-N5000-s5002	12695	558.4	64	2.8	31	1.4	25	1.1	**23**	**1.0**	31	1.4
ssn-N10000-s10000	26559	420.0	185	2.9	**63**	**1.0**	123	2.0	64	1.0	79	1.3
ssn-N10000-s10001	26228	449.1	193	3.3	72	1.2	**58**	**1.0**	59	1.0	78	1.3
ssn-N10000-s10002	24916	463.1	187	3.5	80	1.5	56	1.0	**54**	**1.0**	79	1.5
ssn-N20000-s20000	$+\infty$	>382.6	512	4.5	152	1.3	**113**	**1.0**	120	1.1	8143	72.1
ssn-N20000-s20001	$+\infty$	>355.0	503	4.1	**122**	**1.0**	588	4.8	128	1.1	167	1.4
ssn-N20000-s20002	$+\infty$	>356.6	450	3.7	160	1.3	**121**	**1.0**	1624	13.4	154	1.3
storm-N1000-s1000	23	3.6	10	1.6	19	3.0	8	1.3	**6**	**1.0**	8	1.3
storm-N1000-s1001	24	3.7	11	1.6	23	3.5	8	1.3	**7**	**1.0**	8	1.3
storm-N1000-s1002	24	3.8	11	1.7	21	3.3	8	1.3	**6**	**1.0**	8	1.3
storm-N5000-s5000	110	2.0	100	1.8	159	2.9	58	1.1	**54**	**1.0**	65	1.2
storm-N5000-s5001	117	2.2	118	2.2	184	3.4	59	1.1	**54**	**1.0**	65	1.2
storm-N5000-s5002	116	2.1	99	1.8	181	3.3	63	1.1	**55**	**1.0**	65	1.2
storm-N10000-s10000	215	1.4	468	3.0	508	3.2	162	1.0	**159**	**1.0**	191	1.2
storm-N10000-s10001	225	1.5	479	3.1	494	3.2	**154**	**1.0**	161	1.1	188	1.2
storm-N10000-s10002	233	1.5	542	3.5	474	3.1	**153**	**1.0**	157	1.0	189	1.2
storm-N20000-s20000	**465**	**1.0**	2240	4.8	1470	3.2	581	1.2	704	1.5	574	1.2
storm-N20000-s20001	**434**	**1.0**	2460	5.7	1300	3.0	585	1.3	669	1.5	603	1.4
storm-N20000-s20002	**476**	**1.0**	2410	5.1	1400	2.9	574	1.2	642	1.3	587	1.2
20term-N1000-s1000	544	13.5	749	18.6	**40**	**1.0**	82	2.0	46	1.1	74	1.8
20term-N1000-s1001	584	16.1	646	17.8	**36**	**1.0**	82	2.3	47	1.3	72	2.0
20term-N1000-s1002	604	16.0	877	23.2	**38**	**1.0**	82	2.2	53	1.4	76	2.0
20term-N5000-s5000	3095	4.7	29455	44.6	**660**	**1.0**	2059	3.1	1497	2.3	1951	3.0
20term-N5000-s5001	3699	5.4	22490	33.0	**681**	**1.0**	2066	3.0	1333	2.0	2302	3.4
20term-N5000-s5002	3725	6.6	21342	38.0	**561**	**1.0**	2178	3.9	1176	2.1	2486	4.4
20term-N10000-s10000	6803	3.1	$+\infty$	>20.4	**2193**	**1.0**	9654	4.4	5526	2.5	11592	5.3
20term-N10000-s10001	6404	2.7	$+\infty$	>19.5	**2330**	**1.0**	11062	4.7	7874	3.4	9436	4.1
20term-N10000-s10002	7494	3.3	$+\infty$	>19.6	**2288**	**1.0**	11483	5.0	5196	2.3	10212	4.5
20term-N20000-s20000	**13429**	**1.0**	$+\infty$	>5.7	$+\infty$	>3.2	$+\infty$	>3.2	$+\infty$	>3.2	$+\infty$	>3.2
20term-N20000-s20001	12763	1.4	$+\infty$	>5.0	**9062**	**1.0**	$+\infty$	>4.8	$+\infty$	>4.8	$+\infty$	>4.8
20term-N20000-s20002	14868	1.5	$+\infty$	>8.1	**9613**	**1.0**	$+\infty$	>4.5	$+\infty$	>4.6	$+\infty$	>4.6
Fleet20-N1000-s1000	513	9.4	224	4.1	143	2.6	105	1.9	102	1.9	**55**	**1.0**
Fleet20-N1000-s1001	539	10.1	228	4.3	139	2.6	110	2.1	100	1.9	**53**	**1.0**
Fleet20-N1000-s1002	546	7.7	224	3.2	154	2.2	**70**	**1.0**	103	1.5	115	1.6
Fleet20-N5000-s5000	2780	1.5	5530	2.9	2380	1.3	2050	1.1	**1880**	**1.0**	2110	1.1
Fleet20-N5000-s5001	2760	1.5	5090	2.8	2260	1.2	**1850**	**1.0**	1870	1.0	2070	1.1
Fleet20-N5000-s5002	2730	1.5	5370	2.9	2610	1.4	1950	1.0	**1870**	**1.0**	2110	1.1
Fleet20-N10000-s10000	**5860**	**1.0**	29600	5.1	10400	1.8	$+\infty$	>7.4	8780	1.5	11000	1.9
Fleet20-N10000-s10001	**5480**	**1.0**	28200	5.1	8310	1.5	8350	1.5	8560	1.6	9950	1.8
Fleet20-N10000-s10002	**5790**	**1.0**	29000	5.0	11000	1.9	8190	1.4	8270	1.4	$+\infty$	>7.5
Fleet20-N20000-s20000	**11400**	**1.0**	$+\infty$	>4.0	$+\infty$	>3.8	$+\infty$	>3.8	$+\infty$	>3.8	$+\infty$	>3.9
Fleet20-N20000-s20001	**11500**	**1.0**	$+\infty$	>3.8	18200	1.6	$+\infty$	>3.8	$+\infty$	>3.8	$+\infty$	>3.8
Fleet20-N20000-s20002	**11000**	**1.0**	$+\infty$	>4.6	$+\infty$	>3.9	$+\infty$	>3.9	$+\infty$	>3.9	$+\infty$	>4.0
product-N1000-s1000	1920	17.9	184	1.7	259	2.4	123	1.1	109	1.0	**107**	**1.0**
product-N1000-s1001	2070	19.9	197	1.9	302	2.9	125	1.2	109	1.0	**104**	**1.0**
product-N1000-s1002	1850	19.1	178	1.8	249	2.6	120	1.2	**97**	**1.0**	**97**	**1.0**
product-N5000-s5000	10500	8.0	3220	2.5	3630	2.8	1830	1.4	1390	1.1	**1310**	**1.0**
product-N5000-s5001	10100	7.4	3440	2.5	3830	2.8	1700	1.2	1480	1.1	**1360**	**1.0**
product-N5000-s5002	10800	7.4	3830	2.6	3730	2.6	2090	1.4	1580	1.1	**1460**	**1.0**
product-N10000-s10000	20200	3.6	15300	2.7	14000	2.5	7330	1.3	5820	1.0	**5580**	**1.0**
product-N10000-s10001	19100	3.7	13300	2.5	11800	2.3	6580	1.3	5560	1.1	**5230**	**1.0**
product-N10000-s10002	21300	4.0	17000	3.2	14100	2.6	6770	1.3	**5370**	**1.0**	5380	1.0
product-N20000-s20000	$+\infty$	>1.7	$+\infty$	>2.0	$+\infty$	>1.7	32700	1.3	26000	1.0	**25200**	**1.0**
product-N20000-s20001	42600	2.1	$+\infty$	>2.2	$+\infty$	>2.2	26600	1.3	24100	1.2	**20000**	**1.0**
product-N20000-s20002	$+\infty$	>1.8	$+\infty$	>1.8	$+\infty$	>1.8	29800	1.2	24100	1.0	**24000**	**1.0**

Table A.2: Results for the Benders by batch algorithm with aggregation, with batch sizes from 1% to 20% of the total number of subproblems.

	Classic monocut		Classic 1% CutAggr		Classic 5% CutAggr		BbB 1% CutAggr		BbB 5% CutAggr		BbB 10% CutAggr		BbB 20% CutAggr	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	2	2.6	0.94	1.2	1	1.6	2	2.2	0.89	1.2	**0.77**	**1.0**	0.86	1.1
LandS-N1000-s1001	2	2.7	1.00	1.3	1	1.8	2	2.1	0.75	1.0	**0.75**	**1.0**	0.90	1.2
LandS-N1000-s1002	2	2.3	1	1.3	1	1.7	2	2.0	0.99	1.2	**0.84**	**1.0**	0.91	1.1
LandS-N5000-s5000	11	2.7	7	1.8	8	2.0	10	2.6	5	1.2	**4**	**1.0**	4	1.1
LandS-N5000-s5001	10	2.3	7	1.6	9	2.0	9	2.1	5	1.2	**4**	**1.0**	4	1.0
LandS-N5000-s5002	11	2.9	7	1.9	8	2.1	9	2.2	**4**	**1.0**	4	1.0	4	1.1
LandS-N10000-s10000	22	2.7	16	1.9	18	2.2	17	2.0	**8**	**1.0**	9	1.1	9	1.1
LandS-N10000-s10001	22	2.8	16	2.0	20	2.5	14	1.8	8	1.0	**8**	**1.0**	9	1.2
LandS-N10000-s10002	20	2.6	16	2.0	18	2.2	17	2.1	8	1.0	**8**	**1.0**	9	1.2
LandS-N20000-s20000	49	3.0	34	2.0	39	2.3	45	2.7	**17**	**1.0**	18	1.1	19	1.2
LandS-N20000-s20001	43	2.4	35	1.9	39	2.2	42	2.4	**18**	**1.0**	18	1.0	21	1.2
LandS-N20000-s20002	44	2.6	32	1.9	40	2.3	45	2.6	18	1.0	**17**	**1.0**	19	1.1
gbd-N1000-s1000	2	3.5	1	2.4	2	3.1	2	2.9	**0.53**	**1.0**	0.68	1.3	0.89	1.7
gbd-N1000-s1001	2	3.6	1	1.6	2	2.5	2	2.4	**0.67**	**1.0**	0.99	1.5	1	1.5
gbd-N1000-s1002	2	3.6	1	1.9	2	2.5	2	3.0	**0.61**	**1.0**	0.68	1.1	0.88	1.4
gbd-N5000-s5000	13	3.8	8	2.4	11	3.2	10	3.0	**3**	**1.0**	4	1.1	4	1.3
gbd-N5000-s5001	11	3.6	9	2.8	10	3.2	8	2.5	**3**	**1.0**	4	1.1	4	1.4
gbd-N5000-s5002	12	3.4	9	2.6	9	2.7	9	2.6	**3**	**1.0**	4	1.1	5	1.3
gbd-N10000-s10000	24	3.4	18	2.5	21	2.9	18	2.5	**7**	**1.0**	8	1.1	9	1.2
gbd-N10000-s10001	24	4.0	19	3.3	19	3.2	13	2.1	**6**	**1.0**	8	1.4	9	1.5
gbd-N10000-s10002	23	3.8	20	3.4	23	3.9	14	2.3	**6**	**1.0**	8	1.4	11	1.8
gbd-N20000-s20000	48	3.8	39	3.2	47	3.7	50	4.0	**12**	**1.0**	16	1.3	20	1.6
gbd-N20000-s20001	51	3.6	42	3.0	45	3.2	31	2.2	15	1.1	**14**	**1.0**	19	1.4
gbd-N20000-s20002	47	3.4	41	3.0	45	3.3	43	3.2	**14**	**1.0**	14	1.0	19	1.4
ssn-N1000-s1000	2279	168.5	25	1.9	146	10.8	**14**	**1.0**	63	4.6	129	9.5	235	17.4
ssn-N1000-s1001	2720	185.6	24	1.7	135	9.2	**15**	**1.0**	63	4.3	130	8.8	253	17.3
ssn-N1000-s1002	2226	173.3	23	1.8	146	11.4	**13**	**1.0**	59	4.6	144	11.2	238	18.5
ssn-N5000-s5000	13425	152.4	371	4.2	1685	19.1	**88**	**1.0**	337	3.8	630	7.2	1342	15.2
ssn-N5000-s5001	14260	158.7	411	4.6	1536	17.1	**90**	**1.0**	322	3.6	672	7.5	1343	15.0
ssn-N5000-s5002	12695	140.6	416	4.6	1524	16.9	**90**	**1.0**	308	3.4	674	7.5	1280	14.2
ssn-N10000-s10000	26559	151.5	1212	6.9	3343	19.1	**175**	**1.0**	672	3.8	1396	8.0	2771	15.8
ssn-N10000-s10001	26228	140.6	1378	7.4	6126	32.8	**187**	**1.0**	760	4.1	1477	7.9	3143	16.8
ssn-N10000-s10002	24916	129.1	1147	5.9	5105	26.4	**193**	**1.0**	690	3.6	1397	7.2	2827	14.6
ssn-N20000-s20000	$+\infty$	>94.6	7066	15.5	18068	39.6	**457**	**1.0**	1651	3.6	3463	7.6	6588	14.4
ssn-N20000-s20001	$+\infty$	>94.3	5558	12.1	40319	88.0	**458**	**1.0**	1651	3.6	3065	6.7	6749	14.7
ssn-N20000-s20002	$+\infty$	>106.2	13186	32.4	19979	49.1	**407**	**1.0**	1543	3.8	3630	8.9	6934	17.0
storm-N1000-s1000	23	3.7	12	2.0	15	2.4	12	1.9	**6**	**1.0**	7	1.1	10	1.6
storm-N1000-s1001	24	3.8	12	1.9	16	2.5	12	1.9	**6**	**1.0**	7	1.1	9	1.4
storm-N1000-s1002	24	3.7	13	2.0	15	2.3	13	2.0	**6**	**1.0**	7	1.1	9	1.4
storm-N5000-s5000	110	3.3	73	2.2	92	2.8	44	1.3	**33**	**1.0**	35	1.1	54	1.6
storm-N5000-s5001	117	3.6	72	2.2	97	3.0	54	1.6	**33**	**1.0**	36	1.1	56	1.7
storm-N5000-s5002	116	3.2	72	2.0	93	2.6	58	1.6	37	1.0	**36**	**1.0**	55	1.5
storm-N10000-s10000	215	3.0	157	2.2	202	2.8	121	1.7	**73**	**1.0**	82	1.1	105	1.4
storm-N10000-s10001	225	3.0	169	2.2	198	2.6	90	1.2	**76**	**1.0**	83	1.1	101	1.3
storm-N10000-s10002	233	3.2	166	2.3	194	2.7	118	1.6	**73**	**1.0**	80	1.1	107	1.5
storm-N20000-s20000	465	2.9	370	2.3	434	2.7	216	1.3	167	1.0	**161**	**1.0**	232	1.4
storm-N20000-s20001	434	2.7	380	2.4	413	2.6	245	1.5	**161**	**1.0**	179	1.1	246	1.5
storm-N20000-s20002	476	3.0	356	2.2	422	2.6	218	1.4	**160**	**1.0**	167	1.0	236	1.5
20term-N1000-s1000	544	36.7	272	18.4	310	20.9	**15**	**1.0**	36	2.5	71	4.8	140	9.5
20term-N1000-s1001	584	40.0	239	16.4	266	18.2	**15**	**1.0**	37	2.5	67	4.6	135	9.3
20term-N1000-s1002	604	41.4	305	20.9	364	25.0	**15**	**1.0**	37	2.5	65	4.5	148	10.2
20term-N5000-s5000	3095	46.0	1627	24.2	2026	30.1	**67**	**1.0**	199	3.0	401	6.0	830	12.4
20term-N5000-s5001	3699	47.2	1453	18.5	1911	24.4	**78**	**1.0**	197	2.5	381	4.9	794	10.1
20term-N5000-s5002	3725	57.8	1733	26.9	1898	29.5	**64**	**1.0**	182	2.8	404	6.3	893	13.9
20term-N10000-s10000	6803	52.5	3885	30.0	4741	36.6	**129**	**1.0**	411	3.2	892	6.9	1874	14.5
20term-N10000-s10001	6404	52.5	3193	26.2	4915	40.3	**122**	**1.0**	409	3.3	914	7.5	1970	16.1
20term-N10000-s10002	7494	54.5	3015	21.9	4864	35.4	**137**	**1.0**	388	2.8	886	6.4	2089	15.2
20term-N20000-s20000	13429	51.5	7375	28.3	10772	41.3	**261**	**1.0**	860	3.3	1913	7.3	7032	27.0
20term-N20000-s20001	12763	43.2	7433	25.1	26284	88.9	**296**	**1.0**	985	3.3	2139	7.2	4704	15.9
20term-N20000-s20002	14868	52.5	6287	22.2	11803	41.7	**283**	**1.0**	897	3.2	2101	7.4	$+\infty$	>152.6
Fleet20-N1000-s1000	513	18.6	123	4.5	221	8.0	**28**	**1.0**	42	1.5	71	2.6	127	4.6
Fleet20-N1000-s1001	539	20.0	126	4.7	219	8.1	**27**	**1.0**	40	1.5	73	2.7	131	4.9
Fleet20-N1000-s1002	546	18.2	126	4.2	225	7.5	**30**	**1.0**	43	1.4	77	2.6	135	4.5
Fleet20-N5000-s5000	2780	25.7	905	8.4	1570	14.5	**108**	**1.0**	218	2.0	354	3.3	675	6.2
Fleet20-N5000-s5001	2760	26.5	930	8.9	1500	14.4	**104**	**1.0**	209	2.0	363	3.5	645	6.2
Fleet20-N5000-s5002	2730	24.8	873	7.9	1520	13.8	**110**	**1.0**	205	1.9	356	3.2	628	5.7
Fleet20-N10000-s10000	5860	27.4	2030	9.5	3430	16.0	**214**	**1.0**	426	2.0	725	3.4	1290	6.0
Fleet20-N10000-s10001	5480	26.2	1960	9.4	3520	16.8	**209**	**1.0**	467	2.2	721	3.4	1290	6.2
Fleet20-N10000-s10002	5790	27.2	2010	9.4	3430	16.1	**213**	**1.0**	426	2.0	716	3.4	1350	6.3
Fleet20-N20000-s20000	11400	28.4	5200	12.9	8040	20.0	**402**	**1.0**	886	2.2	1510	3.8	2810	7.0
Fleet20-N20000-s20001	11500	26.8	4820	11.2	7690	17.9	**429**	**1.0**	856	2.0	1490	3.5	2750	6.4
Fleet20-N20000-s20002	11000	25.9	5140	12.1	7850	18.5	**425**	**1.0**	885	2.1	1560	3.7	2770	6.5
product-N1000-s1000	1920	18.5	191	1.8	415	4.0	**104**	**1.0**	140	1.3	246	2.4	471	4.5
product-N1000-s1001	2070	21.3	197	2.0	452	4.7	**97**	**1.0**	149	1.5	266	2.7	528	5.4
product-N1000-s1002	1850	20.2	182	2.0	425	4.6	**91**	**1.0**	135	1.5	247	2.7	515	5.6
product-N5000-s5000	10500	29.8	1530	4.3	3290	9.3	**352**	**1.0**	734	2.1	1550	4.4	3180	9.0
product-N5000-s5001	10100	29.3	1460	4.2	3250	9.4	**345**	**1.0**	787	2.3	1420	4.1	2580	7.5
product-N5000-s5002	10800	27.7	1580	4.1	3430	8.8	**390**	**1.0**	797	2.0	1730	4.4	2860	7.3
product-N10000-s10000	20200	28.7	3830	5.4	8170	11.6	**704**	**1.0**	1620	2.3	2980	4.2	5670	8.1
product-N10000-s10001	19100	25.3	3910	5.2	7480	9.9	**756**	**1.0**	1400	1.9	2980	3.9	5140	6.8
product-N10000-s10002	21300	21.1	3740	3.7	7620	7.5	**1010**	**1.0**	1550	1.5	3200	3.2	5780	5.7
product-N20000-s20000	$+\infty$	>24.1	9820	5.5	19300	10.8	**1790**	**1.0**	3330	1.9	6740	3.8	13500	7.5
product-N20000-s20001	42600	23.3	9670	5.3	19200	10.5	**1830**	**1.0**	3230	1.8	5950	3.3	11500	6.3
product-N20000-s20002	$+\infty$	>29.6	10400	7.1	19600	13.4	**1460**	**1.0**	3540	2.4	6270	4.3	12500	8.6

Table A.3: Detailed results for the Benders by batch algorithm, with a batch size of 1%, cut aggregation, and stabilization (basic or solution memory) compared to without stabilization

	BbB 1% CutAggr		BbB 1% CutAggr $\alpha = 0.1$		BbB 1% CutAggr $\alpha = 0.5$		BbB 1% CutAggr $\alpha = 0.9$		BbB 1% CutAggr $\alpha = 0.1$ $\beta = 0.1$		BbB 1% CutAggr $\alpha = 0.1$ $\beta = 0.5$		BbB 1% CutAggr $\alpha = 0.1$ $\beta = 0.9$		BbB 1% CutAggr $\alpha = 0.5$ $\beta = 0.1$		BbB 1% CutAggr $\alpha = 0.5$ $\beta = 0.5$		BbB 1% CutAggr $\alpha = 0.5$ $\beta = 0.9$		BbB 1% CutAggr $\alpha = 0.9$ $\beta = 0.1$		BbB 1% CutAggr $\alpha = 0.9$ $\beta = 0.5$		BbB 1% CutAggr $\alpha = 0.9$ $\beta = 0.9$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	2	2.0	1	1.6	1.00	1.2	1	1.3	1	1.8	2	1.8	**0.85**	**1.0**	0.98	1.2	0.95	1.1	1	1.8	1	1.4	0.95	1.1	1	1.7
LandS-N1000-s1001	2	1.7	2	1.8	1	1.1	1	1.4	2	1.7	2	1.6	**0.92**	**1.0**	0.97	1.1	**0.92**	**1.0**	2	1.6	1	1.2	1	1.1	2	1.7
LandS-N1000-s1002	2	2.1	0.92	1.1	0.88	1.4	0.83	1.0	**0.83**	**1.0**	0.90	1.1	0.85	1.0	0.88	1.1	0.92	1.1	1	1.5	0.92	1.1	0.90	1.1	0.90	1.1
LandS-N5000-s5000	10	2.3	7	1.7	**4**	**1.0**	5	1.2	7	1.7	4	1.0	5	1.1	5	1.2	7	1.7	6	1.3	5	1.1	7	1.6	7	1.6
LandS-N5000-s5001	9	2.3	5	1.1	5	1.3	6	1.6	4	1.1	**4**	**1.0**	5	1.2	5	1.2	4	1.0	6	1.5	5	1.3	4	1.1	4	1.1
LandS-N5000-s5002	9	1.9	7	1.6	5	1.1	5	1.1	7	1.7	7	1.6	8	1.7	6	1.2	**4**	**1.0**	7	1.7	4	1.0	5	1.2	7	1.7
LandS-N10000-s10000	17	2.0	**9**	**1.0**	9	1.1	12	1.4	9	1.0	15	1.8	15	1.8	10	1.2	10	1.2	15	1.8	10	1.2	10	1.2	9	1.0
LandS-N10000-s10001	14	1.5	15	1.6	**9**	**1.0**	13	1.3	15	1.6	16	1.6	16	1.7	11	1.1	16	1.7	15	1.6	10	1.1	11	1.1	16	1.6
LandS-N10000-s10002	17	2.0	9	1.0	9	1.0	10	1.1	9	1.0	9	1.1	15	1.7	11	1.3	9	1.1	15	1.7	9	1.1	11	1.3	**9**	**1.0**
LandS-N20000-s20000	45	2.4	30	1.6	21	1.1	21	1.1	31	1.6	32	1.7	**19**	**1.0**	21	1.1	21	1.1	31	1.6	25	1.3	21	1.1	31	1.6
LandS-N20000-s20001	42	2.4	31	1.8	21	1.2	21	1.2	31	1.8	33	1.9	**17**	**1.0**	20	1.2	23	1.3	32	1.9	24	1.4	20	1.2	31	1.8
LandS-N20000-s20002	45	2.7	29	1.7	20	1.2	29	1.7	30	1.8	30	1.8	21	1.2	**17**	**1.0**	31	1.8	18	1.1	22	1.3	31	1.8
gbd-N1000-s1000	2	1.9	1	1.8	**0.81**	**1.0**	0.99	1.2	2	1.9	2	2.4	0.90	1.1	0.96	1.2	2	1.9	1	1.7	0.85	1.0	2	1.9
gbd-N1000-s1001	2	2.0	2	2.2	1	1.3	1	1.8	2	2.0	2	2.3	0.88	1.1	0.83	1.1	2	1.9	1	1.3	**0.79**	**1.0**	2	2.2
gbd-N1000-s1002	2	2.3	1	1.9	0.82	1.0	1	1.6	1	1.9	2	2.0	2	2.4	0.81	1.0	0.80	1.0	2	2.0	1	1.3	**0.79**	**1.0**	1	1.9
gbd-N5000-s5000	10	2.7	7	2.0	4	**1.0**	6	1.7	7	1.9	8	2.1	9	2.4	4	1.2	5	1.4	8	2.1	6	1.7	4	1.2	7	1.9
gbd-N5000-s5001	8	2.3	8	2.2	4	1.1	**3**	**1.0**	7	2.2	8	2.2	8	2.4	5	1.4	4	1.1	8	2.2	4	1.0	5	1.4	7	2.2
gbd-N5000-s5002	9	2.6	9	2.4	5	1.4	4	1.2	8	2.4	8	2.4	6	1.6	5	1.5	8	2.3	**4**	**1.0**	6	1.6	8	2.4
gbd-N10000-s10000	18	2.4	14	1.8	**7**	**1.0**	10	1.3	15	2.0	16	2.2	10	1.4	9	1.2	15	2.0	8	1.1	10	1.4	15	2.0	14	1.9
gbd-N10000-s10001	13	1.7	14	1.9	**7**	**1.0**	9	1.2	13	1.8	14	2.0	17	2.3	11	1.4	8	1.1	14	2.0	10	1.4	14	1.4	14	1.9
gbd-N10000-s10002	14	2.0	14	2.1	8	1.2	7	1.1	14	2.0	14	2.0	19	2.7	8	1.2	7	1.1	15	2.1	**7**	**1.0**	8	1.2	14	2.0
gbd-N20000-s20000	50	3.5	52	3.6	19	1.3	19	1.3	32	2.2	30	2.1	17	1.2	15	1.1	16	1.1	30	2.1	**14**	**1.0**	15	1.1	32	2.2
gbd-N20000-s20001	31	1.9	26	1.6	**17**	**1.0**	22	1.3	28	1.7	29	1.7	30	1.8	18	1.1	19	1.1	30	1.8	17	1.0	18	1.1	29	1.7
gbd-N20000-s20002	43	3.0	27	1.9	15	1.0	**14**	**1.0**	30	2.1	29	2.0	32	2.3	20	1.4	18	1.2	30	2.1	16	1.1	20	1.4	30	2.1
ssn-N1000-s1000	14	1.8	10	1.4	**8**	**1.0**	11	1.4	9	1.2	10	1.3	11	1.4	9	1.2	9	1.2	11	1.4	10	1.3	8	1.1	9	1.2
ssn-N1000-s1001	15	1.9	9	1.2	8	1.1	13	1.6	12	1.6	9	1.2	11	1.4	9	1.2	8	1.1	10	1.3	12	1.6	**8**	**1.0**	10	1.3
ssn-N1000-s1002	13	1.6	8	1.0	**8**	**1.0**	11	1.3	9	1.1	11	1.4	8	1.0	8	1.0	10	1.2	11	1.3	8	1.0	9	1.1	9	1.1
ssn-N5000-s5000	88	2.0	51	1.1	47	1.1	70	1.6	54	1.2	52	1.2	56	1.3	47	1.0	**45**	**1.0**	54	1.2	64	1.4	46	1.0	54	1.2
ssn-N5000-s5001	90	2.0	48	1.0	**46**	**1.0**	65	1.4	49	1.1	52	1.1	60	1.3	47	1.0	46	1.0	53	1.2	62	1.4	46	1.0	49	1.1
ssn-N5000-s5002	90	2.0	51	1.1	49	1.1	68	1.5	50	1.1	52	1.1	58	1.3	52	1.1	**46**	**1.0**	52	1.1	61	1.3	48	1.1	52	1.1
ssn-N10000-s10000	175	2.1	117	1.4	**84**	**1.0**	126	1.5	101	1.2	108	1.3	120	1.4	92	1.1	95	1.1	113	1.3	115	1.4	92	1.1	106	1.3
ssn-N10000-s10001	187	2.0	112	1.2	98	1.1	129	1.4	112	1.2	111	1.2	128	1.4	93	1.0	105	1.1	106	1.1	119	1.3	**93**	**1.0**	106	1.2
ssn-N10000-s10002	193	2.2	101	1.2	90	1.0	134	1.6	108	1.3	107	1.2	123	1.4	93	1.1	**86**	**1.0**	112	1.3	115	1.3	88	1.0	101	1.2
ssn-N20000-s20000	457	2.5	221	1.2	**181**	**1.0**	279	1.5	242	1.3	235	1.3	270	1.5	203	1.1	183	1.0	232	1.3	244	1.3	198	1.1	213	1.2
ssn-N20000-s20001	458	2.5	207	1.1	190	1.0	284	1.6	232	1.3	228	1.3	265	1.5	**182**	**1.0**	186	1.0	230	1.3	259	1.4	186	1.0	221	1.2
ssn-N20000-s20002	407	2.1	215	1.1	201	1.1	305	1.6	215	1.1	228	1.2	255	1.3	**190**	**1.0**	200	1.1	235	1.2	251	1.3	193	1.0	226	1.2
storm-N1000-s1000	12	1.9	9	1.5	8	1.2	7	1.2	10	1.5	10	1.6	7	1.1	8	1.3	**6**	**1.0**	10	1.6	7	1.1	7	1.0	10	1.6
storm-N1000-s1001	12	2.0	7	1.1	7	1.2	8	1.4	7	1.1	10	1.6	7	1.2	**6**	**1.0**	7	1.2	9	1.6	7	1.2	8	1.3	7	1.1
storm-N1000-s1002	13	2.0	9	1.5	7	1.0	8	1.2	10	1.5	10	1.5	7	1.0	7	1.1	8	1.2	10	1.5	7	1.1	6	**1.0**	10	1.5
storm-N5000-s5000	44	1.4	33	1.1	32	1.1	37	1.2	32	1.0	33	1.1	36	1.2	**31**	**1.0**	37	1.2	33	1.1	35	1.1	31	1.0	31	1.0
storm-N5000-s5001	54	1.7	47	1.5	35	1.1	37	1.2	47	1.5	34	1.1	35	1.1	42	1.3	32	1.0	33	1.0	33	1.0	**32**	**1.0**	48	1.5
storm-N5000-s5002	58	1.9	33	1.1	**30**	**1.0**	32	1.1	34	1.1	32	1.1	33	1.1	32	1.0	33	1.1	32	1.1	37	1.2	32	1.1	34	1.1
storm-N10000-s10000	121	2.0	65	1.1	64	1.1	81	1.4	67	1.1	67	1.1	109	1.8	64	1.1	68	1.1	68	1.1	62	1.0	**59**	**1.0**	67	1.1
storm-N10000-s10001	90	1.4	68	1.1	**64**	**1.0**	68	1.1	68	1.1	66	1.0	108	1.7	67	1.1	66	1.0	67	1.0	71	1.1	65	1.0	68	1.1
storm-N10000-s10002	118	1.9	66	1.1	66	1.1	98	1.6	67	1.1	101	1.6	70	1.1	69	1.1	64	1.0	100	1.6	**62**	**1.0**	66	1.1	67	1.1
storm-N20000-s20000	216	1.7	141	1.1	139	1.1	162	1.3	139	1.1	138	1.1	144	1.1	130	1.0	**127**	**1.0**	136	1.1	152	1.2	131	1.0	139	1.1
storm-N20000-s20001	245	2.0	134	1.1	137	1.1	127	1.0	140	1.1	129	1.0	146	1.2	130	1.1	**123**	**1.0**	128	1.0	137	1.1	126	1.0	141	1.1
storm-N20000-s20002	218	1.7	145	1.1	135	1.0	137	1.1	**130**	**1.0**	135	1.0	143	1.1	141	1.1	135	1.0	133	1.0	192	1.5	152	1.2	131	1.0
20term-N1000-s1000	15	1.7	13	1.5	11	1.3	14	1.6	14	1.6	12	1.4	16	1.8	**9**	**1.0**	10	1.2	10	1.1	12	1.4	11	1.3	15	1.7
20term-N1000-s1001	15	1.5	10	1.1	10	1.0	13	1.4	15	1.6	17	1.7	18	1.9	11	1.2	11	1.2	16	1.7	12	1.3	10	1.0	**10**	**1.0**
20term-N1000-s1002	15	1.6	11	1.3	10	1.2	14	1.6	18	2.0	12	1.4	22	2.4	11	1.2	11	1.2	14	1.5	12	1.3	**9**	**1.0**	16	1.8
20term-N5000-s5000	67	1.3	60	1.2	52	1.0	64	1.3	60	1.2	67	1.3	84	1.6	51	1.0	57	1.1	**51**	**1.0**	58	1.1	66	1.3	61	1.2
20term-N5000-s5001	78	1.8	64	1.5	**43**	**1.0**	67	1.5	67	1.5	67	1.5	84	1.9	51	1.2	46	1.1	74	1.7	58	1.3	48	1.1	74	1.7
20term-N5000-s5002	64	1.4	54	1.2	54	1.2	58	1.3	70	1.5	69	1.5	117	2.6	56	1.2	**45**	**1.0**	65	1.4	55	1.2	53	1.2	68	1.5
20term-N10000-s10000	129	1.3	114	1.1	101	1.0	116	1.2	118	1.2	147	1.5	188	1.9	**101**	**1.0**	101	1.0	135	1.3	101	1.0	102	1.0	113	1.1
20term-N10000-s10001	122	1.3	148	1.6	101	1.1	114	1.3	135	1.5	139	1.5	152	1.7	115	1.3	**91**	**1.0**	151	1.7	115	1.3	110	1.2	179	2.0
20term-N10000-s10002	137	1.6	138	1.6	100	1.2	125	1.5	126	1.5	126	1.5	169	2.0	87	1.0	**85**	**1.0**	102	1.2	136	1.6	101	1.2	176	2.1
20term-N20000-s20000	261	1.4	279	1.5	**191**	**1.0**	258	1.4	193	1.0	330	1.7	367	1.9	226	1.2	244	1.3	361	1.9	251	1.3	222	1.2	276	1.4
20term-N20000-s20001	296	1.4	311	1.5	**210**	**1.0**	256	1.2	289	1.4	337	1.6	326	1.6	241	1.2	243	1.2	272	1.3	236	1.1	224	1.1	267	1.3
20term-N20000-s20002	283	1.8	**159**	**1.0**	191	1.2	270	1.7	237	1.5	178	1.1	361	2.3	233	1.5	212	1.3	288	1.8	254	1.6	230	1.4	337	2.1
Fleet20-N1000-s1000	28	1.7	20	1.2	**17**	**1.0**	19	1.1	18	1.1	21	1.3	24	1.4	17	1.0	18	1.1	22	1.3	19	1.1	17	1.0	19	1.1
Fleet20-N1000-s1001	27	1.6	**17**	**1.0**	17	1.0	20	1.2	18	1.1	20	1.2	24	1.4	17	1.0	18	1.0	22	1.3	19	1.1	18	1.1	18	1.1
Fleet20-N1000-s1002	30	1.7	20	1.2	18	1.0	21	1.2	21	1.2	21	1.2	27	1.5	19	1.1	18	1.0	22	1.2	20	1.2	**18**	**1.0**	18	1.1
Fleet20-N5000-s5000	108	1.4	86	1.2	**75**	**1.0**	83	1.1	89	1.2	96	1.3	125	1.7	78	1.0	84	1.1	95	1.3	83	1.1	76	1.0	89	1.2
Fleet20-N5000-s5001	104	1.4	95	1.3	77	1.0	84	1.1	90	1.2	94	1.2	135	1.8	**76**	**1.0**	81	1.1	105	1.4	80	1.1	78	1.0	103	1.4
Fleet20-N5000-s5002	110	1.5	96	1.3	**74**	**1.0**	84	1.1	93	1.3	101	1.4	137	1.9	74	1.0	79	1.1	107	1.4	82	1.1	77	1.0	101	1.4
Fleet20-N10000-s10000	214	1.5	172	1.2	**147**	**1.0**	163	1.1	184	1.3	197	1.3	270	1.8	155	1.1	159	1.1	191	1.3	163	1.1	152	1.0	174	1.2
Fleet20-N10000-s10001	209	1.4	196	1.3	**148**	**1.0**	167	1.1	183	1.2	193	1.3	240	1.6	155	1.0	164	1.1	193	1.3	180	1.2	156	1.1	175	1.2
Fleet20-N10000-s10002	213	1.5	192	1.4	144	1.0	161	1.1	206	1.5	220	1.5	275	1.9	154	1.1	163	1.1	213	1.5	163	1.1	**142**	**1.0**	181	1.3
Fleet20-N20000-s20000	402	1.3	358	1.2	**302**	**1.0**	347	1.1	426	1.4	482	1.6	557	1.8	307	1.0	327	1.1	434	1.4	340	1.1	315	1.0	401	1.3
Fleet20-N20000-s20001	429	1.4	391	1.3	310	1.0	333	1.1	364	1.2	416	1.4	534	1.8	322	1.1	332	1.1	460	1.5	340	1.1	**301**	**1.0**	422	1.4
Fleet20-N20000-s20002	425	1.4	424	1.4	317	1.0	355	1.2	385	1.3	462	1.5	564	1.8	311	1.0	333	1.1	460	1.5	337	1.1	**305**	**1.0**	389	1.3
product-N1000-s1000	104	1.4	87	1.2	76	1.0	89	1.2	96	1.3	91	1.2	124	1.6	85	1.1	**76**	**1.0**	92	1.2	88	1.2	80	1.1	95	1.3
product-N1000-s1001	97	1.2	88	1.1	**78**	**1.0**	79	1.0	83	1.1	81	1.0	145	1.9	89	1.1	82	1.1	79	1.0	85	1.1	88	1.1	83	1.1
product-N1000-s1002	91	1.2	84	1.1	75	1.0	83	1.1	82	1.1	75	1.0	110	1.5	76	1.0	77	1.0	**75**	**1.0**	79	1.1	76	1.0	81	1.1
product-N5000-s5000	352	1.3	**264**	**1.0**	305	1.2	312	1.2	266	1.0	306	1.2	382	1.4	290	1.1	296	1.1	294	1.1	331	1.3	291	1.1	288	1.1
product-N5000-s5001	345	1.3	**269**	**1.0**	335	1.2	303	1.1	290	1.1	293	1.1	396	1.5	303	1.1	286	1.1	293	1.1	315	1.2	281	1.0	288	1.1
product-N5000-s5002	390	1.4	285	1.0	295	1.0	301	1.1	298	1.1	308	1.1	400	1.4	287	1.0	**283**	**1.0**	305	1.1	306	1.1	286	1.0	298	1.1
product-N10000-s10000	704	1.3	**522**	**1.0**	565	1.1	630	1.2	549	1.1	636	1.2	824	1.6	579	1.1	590	1.1	638	1.2	594	1.1	581	1.1	544	1.0
product-N10000-s10001	756	1.4	563	1.1	**534**	**1.0**	567	1.1	540	1.0	620	1.2	735	1.4	554	1.0	543	1.0	626	1.2	579	1.1	553	1.0	542	1.0
product-N10000-s10002	1010	1.8	**555**	**1.0**	556	1.0	593	1.1	610	1.1	583	1.1	749	1.3	561	1.0	588	1.1	562	1.0	614	1.1	562	1.0	610	1.0
product-N20000-s20000	1790	1.6	1300	1.1	**1140**	**1.0**	1280	1.1	1300	1.1	1370	1.2	1830	1.6	1170	1.0	1330	1.2	1370	1.2	1240	1.1	1150	1.0	1300	1.1
product-N20000-s20001	1830	1.6	1260	1.1	1140	1.0	1360	1.2	1170	1.1	1160	1.0	1610	1.5	1120	1.0	1220	1.1	1230	1.1	1160	1.0	**1110**	**1.0**	1180	1.1
product-N20000-s20002	1460	1.3	1230	1.1	1140	1.0	1230	1.1	1260	1.1	1260	1.1	1690	1.5	1190	1.1	1300	1.2	1250	1.1	1240	1.1	**1110**	**1.0**	1250	1.1

Table A.4: Detailed results for the Benders by batch algorithm, with a batch size of 5%, cut aggregation, and stabilization (basic or solution memory) compared to without stabilization

	BbB 5% CutAggr		BbB 5% CutAggr $\alpha = 0.1$		BbB 5% CutAggr $\alpha = 0.5$		BbB 5% CutAggr $\alpha = 0.9$		BbB 5% CutAggr $\alpha = 0.1$ $\beta = 0.5$		BbB 5% CutAggr $\alpha = 0.1$ $\beta = 0.9$		BbB 5% CutAggr $\alpha = 0.5$ $\beta = 0.1$		BbB 5% CutAggr $\alpha = 0.5$ $\beta = 0.5$		BbB 5% CutAggr $\alpha = 0.5$ $\beta = 0.9$		BbB 5% CutAggr $\alpha = 0.9$ $\beta = 0.1$		BbB 5% CutAggr $\alpha = 0.9$ $\beta = 0.5$		BbB 5% CutAggr $\alpha = 0.9$ $\beta = 0.9$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	0.89	1.2	0.79	1.1	0.80	1.1	0.81	1.1	0.96	1.3	0.84	1.1	2	2.8	0.91	1.2	**0.74**	**1.0**	0.84	1.1	0.75	1.0	0.86	1.2	0.85	1.1
LandS-N1000-s1001	**0.75**	**1.0**	0.87	1.2	0.93	1.2	0.97	1.3	2	2.4	0.96	1.3	2	2.6	0.84	1.1	0.90	1.2	0.88	1.2	0.89	1.2	0.84	1.1	2	2.3
LandS-N1000-s1002	0.99	1.4	0.99	1.4	0.71	1.0	0.75	1.1	0.99	1.4	0.89	1.3	1	1.7	0.78	1.1	0.72	1.0	0.89	1.3	0.78	1.1	**0.70**	**1.0**	0.92	1.3
LandS-N5000-s5000	5	1.2	8	2.1	**4**	**1.0**	4	1.1	8	2.3	9	2.4	10	2.8	5	1.2	4	1.1	9	2.4	4	1.1	4	1.2	8	2.2
LandS-N5000-s5001	5	1.4	4	1.2	5	1.2	4	1.2	5	1.4	5	1.4	7	1.9	4	1.0	4	1.2	5	1.3	4	1.1	**4**	**1.0**	5	1.4
LandS-N5000-s5002	4	1.0	8	2.2	5	1.3	**4**	**1.0**	9	2.3	9	2.3	10	2.9	4	1.2	5	1.3	9	2.3	4	1.1	4	1.2	9	2.4
LandS-N10000-s10000	8	1.1	16	2.1	**7**	**1.0**	9	1.1	10	1.3	10	1.4	14	1.8	9	1.2	8	1.1	11	1.4	8	1.1	9	1.2	10	1.3
LandS-N10000-s10001	8	1.0	17	2.1	10	1.2	**8**	**1.0**	18	2.3	18	2.3	15	1.8	9	1.1	10	1.3	18	2.3	9	1.1	8	1.1	18	2.2
LandS-N10000-s10002	8	1.1	9	1.2	**7**	**1.0**	8	1.0	18	2.4	18	2.5	22	2.9	8	1.1	8	1.1	18	2.5	8	1.1	8	1.1	18	2.4
LandS-N20000-s20000	**17**	**1.0**	36	2.1	18	1.1	17	1.1	38	2.3	38	2.5	44	2.7	20	1.2	19	1.2	38	2.3	19	1.2	20	1.2	38	2.3
LandS-N20000-s20001	18	1.1	36	2.2	18	1.1	**17**	**1.0**	36	2.2	20	1.2	42	2.5	19	1.1	17	1.0	20	1.2	18	1.1	19	1.1	35	2.1
LandS-N20000-s20002	18	1.1	35	2.2	18	1.1	**16**	**1.0**	37	2.3	38	2.4	48	3.0	20	1.2	20	1.2	38	2.4	19	1.2	20	1.2	37	2.3
gbd-N1000-s1000	**0.53**	**1.0**	2	3.2	1	1.9	0.61	1.1	2	3.5	2	3.9	2	4.6	0.81	1.5	0.87	1.6	2	3.8	0.82	1.6	0.81	1.5	2	3.6
gbd-N1000-s1001	0.67	1.2	2	3.1	0.89	1.5	**0.58**	**1.0**	2	3.1	2	4.0	2	4.2	0.89	1.5	0.90	1.6	2	3.9	0.75	1.3	0.80	1.4	2	3.2
gbd-N1000-s1002	0.61	1.0	2	2.7	0.61	1.0	**0.61**	**1.0**	2	2.8	2	3.0	2	3.7	0.93	1.5	0.92	1.5	2	3.0	0.65	1.1	0.89	1.5	2	2.8
gbd-N5000-s5000	3	1.0	9	2.6	4	1.1	**3**	**1.0**	9	2.8	10	3.0	12	3.8	4	1.1	4	1.3	10	2.9	4	1.1	3	1.1	9	2.8
gbd-N5000-s5001	3	1.1	9	3.1	3	1.0	3	1.0	9	3.2	13	4.5	7	2.5	5	1.5	3	1.1	13	4.6	**3**	**1.0**	5	1.5	9	3.2
gbd-N5000-s5002	3	1.1	10	3.2	4	1.2	**3**	**1.0**	8	2.5	11	3.2	12	3.7	4	1.3	4	1.3	10	3.2	3	1.0	4	1.2	8	2.6
gbd-N10000-s10000	7	1.0	17	2.5	7	1.1	7	1.0	18	2.7	18	2.7	25	3.6	8	1.2	9	1.3	18	2.7	**7**	**1.0**	9	1.3	18	2.7
gbd-N10000-s10001	**6**	**1.0**	17	2.9	9	1.5	7	1.2	17	2.9	19	3.2	24	4.1	7	1.3	9	1.5	19	3.2	7	1.2	7	1.3	17	3.0
gbd-N10000-s10002	**6**	**1.0**	18	3.0	7	1.2	6	1.1	18	2.9	19	3.2	24	4.0	7	1.2	8	1.3	20	3.3	7	1.2	7	1.2	18	3.0
gbd-N20000-s20000	12	1.1	35	3.0	13	1.1	**12**	**1.0**	19	1.6	37	3.2	49	4.1	16	1.4	14	1.2	37	3.1	14	1.2	16	1.4	19	1.6
gbd-N20000-s20001	15	1.2	36	2.7	14	1.1	**13**	**1.0**	37	2.8	46	3.5	49	3.8	17	1.3	20	1.5	45	3.5	15	1.1	17	1.3	37	2.8
gbd-N20000-s20002	**14**	**1.0**	37	2.7	16	1.2	14	1.0	38	2.8	50	3.7	17	1.2	19	1.4	39	2.8	15	1.1	17	1.2	37	2.7		
ssn-N1000-s1000	63	4.3	**14**	**1.0**	17	1.2	38	2.6	15	1.1	16	1.1	22	1.5	18	1.3	17	1.2	16	1.1	82	2.2	19	1.3	15	1.1
ssn-N1000-s1001	63	4.2	**15**	**1.0**	19	1.3	41	2.8	16	1.0	16	1.1	22	1.5	18	1.2	16	1.1	17	1.1	35	2.3	19	1.2	15	1.0
ssn-N1000-s1002	59	3.9	15	1.0	19	1.3	43	2.9	17	1.1	17	1.1	22	1.5	18	1.2	17	1.2	17	1.1	31	2.1	18	1.2	**15**	**1.0**
ssn-N5000-s5000	337	4.1	**81**	**1.0**	111	1.4	197	2.4	89	1.1	94	1.2	127	1.6	111	1.4	96	1.2	99	1.2	172	2.1	112	1.4	93	1.1
ssn-N5000-s5001	322	3.8	**85**	**1.0**	106	1.3	205	2.4	85	1.0	100	1.2	126	1.5	108	1.3	100	1.2	101	1.2	182	2.1	112	1.3	85	1.0
ssn-N5000-s5002	308	3.8	**81**	**1.0**	110	1.4	209	2.6	95	1.2	99	1.2	141	1.8	113	1.4	100	1.2	99	1.2	172	2.1	116	1.4	90	1.1
ssn-N10000-s10000	672	3.6	196	1.1	233	1.3	471	2.5	**185**	**1.0**	204	1.1	277	1.5	232	1.3	213	1.2	212	1.1	389	2.1	222	1.2	194	1.1
ssn-N10000-s10001	760	4.2	**180**	**1.0**	236	1.3	512	2.8	209	1.2	231	1.3	301	1.7	235	1.3	217	1.2	211	1.2	439	2.4	244	1.4	187	1.0
ssn-N10000-s10002	690	3.7	185	1.0	231	1.3	469	2.5	186	1.0	193	1.0	289	1.6	222	1.2	195	1.1	218	1.2	406	2.2	228	1.2	**184**	**1.0**
ssn-N20000-s20000	1651	4.1	**405**	**1.0**	524	1.3	1027	2.5	432	1.1	491	1.2	672	1.7	531	1.3	524	1.3	492	1.2	866	2.1	529	1.3	432	1.1
ssn-N20000-s20001	1651	3.8	**436**	**1.0**	549	1.3	1043	2.4	474	1.1	485	1.1	728	1.7	561	1.3	516	1.2	475	1.1	893	2.0	551	1.3	446	1.0
ssn-N20000-s20002	1543	3.8	**406**	**1.0**	550	1.4	1052	2.6	434	1.1	506	1.2	650	1.6	554	1.4	499	1.2	489	1.2	914	2.3	558	1.4	450	1.1
storm-N1000-s1000	6	1.1	7	1.3	**6**	**1.0**	6	1.1	10	1.8	12	2.0	11	1.9	6	1.0	7	1.2	12	2.0	6	1.1	6	1.1	10	1.8
storm-N1000-s1001	6	1.2	8	1.4	**5**	**1.0**	6	1.1	8	1.4	8	1.4	10	1.8	6	1.1	6	1.2	8	1.4	6	1.0	6	1.1	8	1.4
storm-N1000-s1002	6	1.1	11	1.8	**6**	**1.0**	6	1.0	7	1.2	11	1.9	10	1.8	6	1.1	6	1.1	11	1.9	6	1.0	6	1.1	7	1.2
storm-N5000-s5000	33	1.1	54	1.9	**29**	**1.0**	32	1.1	40	1.4	40	1.4	56	1.9	30	1.1	33	1.1	41	1.4	30	1.0	30	1.1	40	1.4
storm-N5000-s5001	33	1.1	54	1.9	**29**	**1.0**	29	1.0	37	1.3	39	1.3	52	1.8	31	1.1	34	1.2	39	1.3	30	1.1	31	1.1	37	1.3
storm-N5000-s5002	37	1.3	38	1.3	30	1.0	56	1.9	41	1.4	47	1.6	30	1.0	37	1.3	41	1.4	**29**	**1.0**	30	1.0	30	1.0	36	1.9
storm-N10000-s10000	73	1.3	79	1.4	62	1.1	**58**	**1.0**	78	1.4	127	2.2	114	2.0	65	1.1	65	1.1	127	2.2	60	1.0	64	1.1	79	1.4
storm-N10000-s10001	76	1.3	75	1.2	60	1.0	**60**	**1.0**	79	1.3	130	2.2	163	2.7	63	1.0	64	1.1	127	2.1	**60**	**1.0**	63	1.0	79	1.3
storm-N10000-s10002	73	1.3	75	1.3	61	1.1	**56**	**1.0**	116	2.1	82	1.5	118	2.1	67	1.2	70	1.2	82	1.5	59	1.0	66	1.2	117	2.1
storm-N20000-s20000	167	1.4	147	1.2	132	1.1	**120**	**1.0**	148	1.2	173	1.4	256	2.1	138	1.1	140	1.2	172	1.4	139	1.2	138	1.1	147	1.2
storm-N20000-s20001	161	1.3	157	1.3	**121**	**1.0**	122	1.0	180	1.5	186	1.5	243	2.0	127	1.0	143	1.2	185	1.5	125	1.0	127	1.0	181	1.5
storm-N20000-s20002	160	1.3	169	1.4	130	1.1	**120**	**1.0**	153	1.3	170	1.4	240	2.0	141	1.2	148	1.2	171	1.4	127	1.1	141	1.2	153	1.3
20term-N1000-s1000	36	2.5	**15**	**1.0**	21	1.4	33	2.2	15	1.0	21	1.4	30	2.1	22	1.5	20	1.4	18	1.3	27	1.8	20	1.3	16	1.1
20term-N1000-s1001	37	2.6	**14**	**1.0**	19	1.3	32	2.2	16	1.1	18	1.2	31	2.2	18	1.3	19	1.3	17	1.2	29	2.0	21	1.5	15	1.1
20term-N1000-s1002	37	3.0	15	1.2	21	1.7	33	2.7	15	1.2	18	1.4	32	2.6	21	1.7	19	1.5	18	1.4	25	2.0	20	1.6	**12**	**1.0**
20term-N5000-s5000	199	2.8	81	1.1	106	1.5	156	2.2	**71**	**1.0**	89	1.3	157	2.2	94	1.3	91	1.3	94	1.3	134	1.9	104	1.5	87	1.2
20term-N5000-s5001	197	3.0	78	1.2	106	1.6	158	2.4	69	1.0	87	1.3	137	2.1	103	1.6	97	1.5	98	1.5	146	2.2	101	1.5	**66**	**1.0**
20term-N5000-s5002	182	2.5	83	1.1	101	1.4	161	2.2	88	1.2	92	1.3	144	2.0	102	1.4	83	1.1	92	1.3	132	1.8	99	1.4	**73**	**1.0**
20term-N10000-s10000	411	3.0	**136**	**1.0**	206	1.5	352	2.6	145	1.1	192	1.4	285	2.1	208	1.5	201	1.5	158	1.2	302	2.2	217	1.6	140	1.0
20term-N10000-s10001	409	2.5	166	1.0	218	1.4	381	2.4	**161**	**1.0**	163	1.0	342	2.1	213	1.3	204	1.3	210	1.3	305	1.9	197	1.2	173	1.1
20term-N10000-s10002	388	2.6	159	1.1	231	1.6	309	2.1	**147**	**1.0**	189	1.3	273	1.9	214	1.5	206	1.4	155	1.1	313	2.1	214	1.5	152	1.0
20term-N20000-s20000	860	3.3	**257**	**1.0**	501	1.9	802	3.1	306	1.2	339	1.3	635	2.5	488	1.9	444	1.7	449	1.7	685	2.7	490	1.9	309	1.2
20term-N20000-s20001	985	3.5	311	1.1	476	1.7	795	2.8	**284**	**1.0**	427	1.5	610	2.1	512	1.8	441	1.6	350	1.2	684	2.4	436	1.5	328	1.2
20term-N20000-s20002	897	3.2	**281**	**1.0**	466	1.7	743	2.6	315	1.1	389	1.4	710	2.5	458	1.6	439	1.6	390	1.4	697	2.5	465	1.7	298	1.1
Fleet20-N1000-s1000	42	1.9	**22**	**1.0**	25	1.1	32	1.5	24	1.1	27	1.3	36	1.7	24	1.1	26	1.2	24	1.1	30	1.4	24	1.1	24	1.1
Fleet20-N1000-s1001	40	1.9	**22**	**1.0**	25	1.2	32	1.5	22	1.0	25	1.2	33	1.5	24	1.1	25	1.2	26	1.2	30	1.4	23	1.1	22	1.0
Fleet20-N1000-s1002	43	1.9	**23**	**1.0**	25	1.1	33	1.4	25	1.1	28	1.2	36	1.5	26	1.1	29	1.2	32	1.3	25	1.1	24	1.0	24	1.1
Fleet20-N5000-s5000	218	2.0	114	1.0	126	1.1	165	1.5	**110**	**1.0**	130	1.2	179	1.6	126	1.1	127	1.2	128	1.2	152	1.4	129	1.2	122	1.1
Fleet20-N5000-s5001	209	1.7	**121**	**1.0**	125	1.0	160	1.3	123	1.0	143	1.2	188	1.6	122	1.0	126	1.0	127	1.0	151	1.2	122	1.0	125	1.0
Fleet20-N5000-s5002	205	1.7	125	1.0	126	1.0	163	1.3	**122**	**1.0**	126	1.0	172	1.4	126	1.0	124	1.0	140	1.1	152	1.2	**122**	**1.0**	126	1.0
Fleet20-N10000-s10000	426	1.9	232	1.0	264	1.2	343	1.5	**226**	**1.0**	253	1.1	367	1.6	254	1.1	273	1.2	261	1.2	315	1.4	255	1.1	248	1.1
Fleet20-N10000-s10001	467	2.0	249	1.1	250	1.1	345	1.5	264	1.1	279	1.2	376	1.6	256	1.1	259	1.1	269	1.2	310	1.3	258	1.1	**230**	**1.0**
Fleet20-N10000-s10002	426	1.9	228	1.0	258	1.1	324	1.4	**226**	**1.0**	290	1.3	394	1.7	254	1.1	259	1.1	294	1.3	307	1.4	252	1.1	265	1.2
Fleet20-N20000-s20000	886	1.7	556	1.1	**520**	**1.0**	695	1.3	586	1.1	612	1.2	885	1.7	543	1.0	525	1.0	639	1.2	635	1.2	524	1.0	577	1.1
Fleet20-N20000-s20001	856	1.8	488	1.0	523	1.1	671	1.4	509	1.1	544	1.1	783	1.6	530	1.1	541	1.1	539	1.1	637	1.3	518	1.1	**476**	**1.0**
Fleet20-N20000-s20002	885	1.7	545	1.0	546	1.0	681	1.3	554	1.0	662	1.3	819	1.6	**528**	**1.0**	562	1.1	670	1.3	671	1.3	551	1.0	576	1.1
product-N1000-s1000	140	1.9	80	1.1	83	1.1	115	1.5	**75**	**1.0**	79	1.1	122	1.6	85	1.1	86	1.1	79	1.1	108	1.4	84	1.1	75	1.0
product-N1000-s1001	149	2.0	**76**	**1.0**	92	1.2	119	1.6	79	1.0	92	1.2	168	2.2	92	1.2	90	1.2	92	1.2	114	1.5	92	1.2	79	1.0
product-N1000-s1002	135	2.0	**68**	**1.0**	82	1.2	127	1.9	108	1.6	84	1.2	146	2.1	93	1.4	80	1.2	85	1.2	106	1.6	93	1.4	109	1.6
product-N5000-s5000	734	1.9	**380**	**1.0**	454	1.2	607	1.6	382	1.0	458	1.2	835	2.2	432	1.1	459	1.2	448	1.2	554	1.5	439	1.2	382	1.0
product-N5000-s5001	787	2.0	**387**	**1.0**	438	1.1	580	1.5	421	1.1	468	1.2	880	2.3	449	1.2	448	1.2	488	1.3	574	1.5	449	1.2	392	1.0
product-N5000-s5002	797	2.1	**383**	**1.0**	474	1.2	627	1.6	411	1.1	454	1.2	737	1.9	463	1.2	483	1.3	437	1.1	585	1.5	478	1.2	405	1.1
product-N10000-s10000	1620	2.3	**716**	**1.0**	955	1.3	1280	1.8	740	1.0	926	1.3	1510	2.1	866	1.2	875	1.2	910	1.3	1160	1.6	863	1.2	740	1.0
product-N10000-s10001	1400	2.0	**696**	**1.0**	921	1.3	1160	1.7	771	1.1	889	1.3	1620	2.3	913	1.3	901	1.3	897	1.3	1130	1.6	894	1.3	772	1.1
product-N10000-s10002	1550	2.2	**697**	**1.0**	880	1.3	1180	1.7	743	1.1	948	1.4	1470	2.1	862	1.2	892	1.3	909	1.3	1160	1.7	868	1.2	802	1.2
product-N20000-s20000	3330	1.9	**1750**	**1.0**	2070	1.2	2670	1.5	1820	1.0	2120	1.2	3610	2.1	2120	1.2	2200	1.3	2130	1.2	2600	1.5	2150	1.2	1790	1.0
product-N20000-s20001	3230	2.0	**1620**	**1.0**	1880	1.2	2550	1.6	1640	1.0	1930	1.2	3040	1.9	1860	1.1	1840	1.1	1920	1.2	2430	1.5	1870	1.2	1740	1.1
product-N20000-s20002	3540	2.2	1640	1.0	1930	1.2	2730	1.7	**1620**	**1.0**	1990	1.2	3480	2.1	1960	1.2	2060	1.3	2110	1.3	2410	1.5	2060	1.3	**1620**	**1.0**

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

	CPLEX Barrier		Level Bundle		In-out monocut		In-out multicut		In-out 1% CutAggr		In-out 5% CutAggr		BbB 1% CutAggr $\alpha = 0.5$	
instance	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio	time	ratio
LandS-N1000-s1000	**0.07**	**1.0**	1	17.3	1	15.6	2	29.4	0.71	10.1	1	14.4	1.00	14.2
LandS-N1000-s1001	**0.08**	**1.0**	1	17.0	0.59	7.4	1	15.0	0.74	9.3	1	12.7	1	12.8
LandS-N1000-s1002	**0.07**	**1.0**	1	17.8	0.99	14.1	1	15.6	0.69	9.9	0.91	13.0	0.88	12.5
LandS-N5000-s5000	**1**	**1.0**	8	5.7	8	6.3	10	7.6	5	3.5	5	3.9	4	3.3
LandS-N5000-s5001	**0.41**	**1.0**	7	17.2	8	19.4	6	15.5	5	11.2	6	13.5	5	13.2
LandS-N5000-s5002	**1**	**1.0**	6	4.2	8	5.8	12	8.4	4	3.2	6	4.3	5	3.5
LandS-N10000-s10000	**0.96**	**1.0**	14	14.5	24	24.8	11	11.6	9	9.4	12	12.4	9	9.4
LandS-N10000-s10001	**1**	**1.0**	13	12.1	24	22.1	13	11.9	10	9.4	10	9.3	9	8.7
LandS-N10000-s10002	**0.97**	**1.0**	15	15.5	23	23.8	23	23.4	10	10.3	11	11.7	9	9.0
LandS-N20000-s20000	**7**	**1.0**	28	4.1	71	10.4	42	6.1	22	3.2	26	3.8	21	3.1
LandS-N20000-s20001	**2**	**1.0**	26	12.4	67	32.4	40	19.0	22	10.5	21	9.9	21	10.3
LandS-N20000-s20002	**7**	**1.0**	29	4.0	48	6.7	43	6.0	22	3.1	21	2.9	20	2.7
gbd-N1000-s1000	**0.03**	**1.0**	2	58.1	1	42.2	3	88.7	0.97	32.2	2	57.0	0.81	26.9
gbd-N1000-s1001	**0.03**	**1.0**	2	78.4	1	42.0	2	53.0	1	46.8	2	50.9	1	33.6
gbd-N1000-s1002	**0.05**	**1.0**	2	46.9	1	25.4	2	34.6	1	21.8	1	26.5	0.82	16.4
gbd-N5000-s5000	**0.15**	**1.0**	8	55.7	7	48.5	13	89.3	7	48.3	9	58.5	4	24.4
gbd-N5000-s5001	**0.18**	**1.0**	11	61.4	11	63.7	9	50.5	7	37.2	7	41.3	4	20.1
gbd-N5000-s5002	**0.17**	**1.0**	11	63.1	12	70.5	9	52.0	7	39.8	7	41.5	5	29.8
gbd-N10000-s10000	**0.32**	**1.0**	23	70.9	19	57.9	30	93.1	17	54.5	18	54.8	7	23.0
gbd-N10000-s10001	**0.35**	**1.0**	26	74.3	32	91.1	18	50.5	14	39.2	17	47.6	7	21.0
gbd-N10000-s10002	**0.37**	**1.0**	23	63.4	20	53.5	15	41.5	16	43.4	18	48.6	8	22.4
gbd-N20000-s20000	**1**	**1.0**	45	40.1	107	94.6	56	49.7	30	26.5	34	30.1	19	16.5
gbd-N20000-s20001	**0.86**	**1.0**	47	54.1	72	83.4	55	64.5	30	34.7	31	35.9	17	19.4
gbd-N20000-s20002	**0.75**	**1.0**	39	52.3	69	91.4	51	67.6	31	41.8	38	51.3	15	19.6
ssn-N1000-s1000	32	7.9	97	24.0	**4**	**1.0**	187	46.4	9	2.3	19	4.8	8	1.9
ssn-N1000-s1001	32	5.2	85	13.6	**6**	**1.0**	117	18.7	10	1.5	19	3.1	8	1.3
ssn-N1000-s1002	31	4.9	87	13.8	**6**	**1.0**	106	16.9	10	1.6	19	3.0	8	1.3
ssn-N5000-s5000	293	8.3	621	17.6	**35**	**1.0**	936	26.5	67	1.9	139	3.9	47	1.3
ssn-N5000-s5001	327	9.4	719	20.6	**35**	**1.0**	597	17.1	69	2.0	128	3.7	46	1.3
ssn-N5000-s5002	311	14.1	631	28.5	**22**	**1.0**	852	38.5	74	3.4	133	6.0	49	2.2
ssn-N10000-s10000	1271	15.1	1440	17.1	86	1.0	1937	23.0	167	2.0	319	3.8	**84**	**1.0**
ssn-N10000-s10001	1332	25.0	1613	30.2	**53**	**1.0**	1261	23.6	185	3.5	318	6.0	98	1.8
ssn-N10000-s10002	1064	20.8	1451	28.3	**51**	**1.0**	1195	23.3	161	3.1	298	5.8	90	1.8
ssn-N20000-s20000	2592	14.3	3232	17.9	245	1.4	3791	21.0	441	2.4	729	4.0	**181**	**1.0**
ssn-N20000-s20001	2070	10.9	2986	15.7	237	1.2	2460	12.9	365	1.9	743	3.9	**190**	**1.0**
ssn-N20000-s20002	3195	15.9	3108	15.4	246	1.2	2332	11.6	395	2.0	735	3.6	**201**	**1.0**
storm-N1000-s1000	41	5.4	14	1.9	10	1.3	11	1.4	8	1.0	10	1.3	**8**	**1.0**
storm-N1000-s1001	41	6.0	16	2.2	**7**	**1.0**	21	3.0	7	1.0	10	1.4	7	1.1
storm-N1000-s1002	41	6.2	15	2.3	11	1.7	12	1.8	7	1.1	9	1.4	**7**	**1.0**
storm-N5000-s5000	348	10.7	74	2.3	41	1.3	63	1.9	52	1.6	53	1.6	**32**	**1.0**
storm-N5000-s5001	294	8.4	78	2.2	38	1.1	61	1.7	51	1.5	53	1.5	**35**	**1.0**
storm-N5000-s5002	305	10.1	76	2.5	43	1.4	63	2.1	45	1.5	51	1.7	**30**	**1.0**
storm-N10000-s10000	808	12.7	140	2.2	108	1.7	212	3.3	94	1.5	100	1.6	**64**	**1.0**
storm-N10000-s10001	732	11.5	149	2.3	105	1.6	201	3.2	104	1.6	117	1.8	**64**	**1.0**
storm-N10000-s10002	751	11.3	147	2.2	161	2.4	189	2.8	99	1.5	114	1.7	**66**	**1.0**
storm-N20000-s20000	2510	18.1	316	2.3	515	3.7	259	1.9	218	1.6	237	1.7	**139**	**1.0**
storm-N20000-s20001	2362	17.2	266	1.9	633	4.6	251	1.8	202	1.5	230	1.7	**137**	**1.0**
storm-N20000-s20002	2297	17.0	283	2.1	570	4.2	246	1.8	214	1.6	228	1.7	**135**	**1.0**
20term-N1000-s1000	14	1.2	197	17.3	27	2.4	128	11.3	24	2.1	41	3.6	**11**	**1.0**
20term-N1000-s1001	14	1.4	214	22.1	43	4.5	74	7.6	26	2.7	46	4.8	**10**	**1.0**
20term-N1000-s1002	14	1.3	241	23.2	38	3.7	139	13.4	31	3.0	45	4.4	**10**	**1.0**
20term-N5000-s5000	83	1.6	994	19.1	581	11.2	661	12.7	188	3.6	271	5.2	**52**	**1.0**
20term-N5000-s5001	80	1.8	1059	24.4	423	9.7	650	14.9	206	4.7	277	6.4	**43**	**1.0**
20term-N5000-s5002	84	1.6	1078	20.1	443	8.3	732	13.7	198	3.7	257	4.8	**54**	**1.0**
20term-N10000-s10000	205	2.0	2305	22.8	2491	24.7	863	8.5	465	4.6	649	6.4	**101**	**1.0**
20term-N10000-s10001	199	2.0	2647	26.3	3382	33.6	1389	13.8	491	4.9	560	5.6	**101**	**1.0**
20term-N10000-s10002	194	1.9	2400	24.1	2543	25.5	1317	13.2	467	4.7	569	5.7	**100**	**1.0**
20term-N20000-s20000	457	2.4	4562	23.9	13423	70.4	1834	9.6	1007	5.3	1412	7.4	**191**	**1.0**
20term-N20000-s20001	457	2.2	4378	20.9	10267	49.0	1680	8.0	980	4.7	1407	6.7	**210**	**1.0**
20term-N20000-s20002	451	2.4	5588	29.3	9286	48.7	1748	9.2	1043	5.5	1295	6.8	**191**	**1.0**
Fleet20-N1000-s1000	24	1.5	104	6.2	61	3.7	71	4.3	27	1.6	42	2.5	**17**	**1.0**
Fleet20-N1000-s1001	23	1.3	103	6.0	34	2.0	103	6.0	26	1.5	39	2.3	**17**	**1.0**
Fleet20-N1000-s1002	22	1.2	114	6.3	55	3.1	106	5.9	25	1.4	43	2.4	**18**	**1.0**
Fleet20-N5000-s5000	266	3.6	485	6.5	933	12.5	552	7.4	181	2.4	239	3.2	**75**	**1.0**
Fleet20-N5000-s5001	273	3.6	509	6.6	541	7.1	331	4.3	172	2.2	264	3.4	**77**	**1.0**
Fleet20-N5000-s5002	267	3.6	506	6.8	682	9.2	535	7.2	198	2.7	248	3.4	**74**	**1.0**
Fleet20-N10000-s10000	784	5.3	988	6.7	3540	24.1	1150	7.8	435	3.0	598	4.1	**147**	**1.0**
Fleet20-N10000-s10001	816	5.5	1040	7.0	4750	32.1	1230	8.3	422	2.9	550	3.7	**148**	**1.0**
Fleet20-N10000-s10002	826	5.7	984	6.8	2950	20.5	708	4.9	448	3.1	623	4.3	**144**	**1.0**
Fleet20-N20000-s20000	2488	8.2	2630	8.7	14900	49.3	2470	8.2	1070	3.5	1270	4.2	**302**	**1.0**
Fleet20-N20000-s20001	2469	8.0	2910	9.4	14100	45.5	1490	4.8	945	3.0	1240	4.0	**310**	**1.0**
Fleet20-N20000-s20002	2381	7.5	2650	8.4	22000	69.4	1380	4.4	1040	3.3	1430	4.5	**317**	**1.0**
productLarge-N1000-s1000	185	2.5	479	6.4	**75**	**1.0**	480	6.4	108	1.4	180	2.4	76	1.0
productLarge-N1000-s1001	186	2.4	718	9.2	83	1.1	539	6.9	124	1.6	179	2.3	**78**	**1.0**
productLarge-N1000-s1002	165	2.2	677	9.0	84	1.1	519	6.9	108	1.4	189	2.5	**75**	**1.0**
productLarge-N5000-s5000	1374	4.5	3290	10.8	1070	3.5	2840	9.3	820	2.7	1460	4.8	**305**	**1.0**
productLarge-N5000-s5001	3073	9.2	3150	9.4	1100	3.3	2550	7.6	724	2.2	1330	4.0	**335**	**1.0**
productLarge-N5000-s5002	1916	6.5	3160	10.7	1210	4.1	2680	9.1	817	2.8	1350	4.6	**295**	**1.0**
productLarge-N10000-s10000	4991	8.8	6910	12.2	4940	8.7	5750	10.2	2030	3.6	3130	5.5	**565**	**1.0**
productLarge-N10000-s10001	3850	7.2	6670	12.5	6860	12.8	5920	11.1	2000	3.7	2810	5.3	**534**	**1.0**
productLarge-N10000-s10002	4351	7.8	7940	14.3	4270	7.7	5520	9.9	1880	3.4	3460	6.2	**556**	**1.0**
productLarge-N20000-s20000	14757	12.9	13200	11.6	$+\infty$	>43.5	12700	11.1	4700	4.1	8300	7.3	**1140**	**1.0**
productLarge-N20000-s20001	14346	12.6	13900	12.2	$+\infty$	>46.7	11700	10.3	4690	4.1	7580	6.6	**1140**	**1.0**
productLarge-N20000-s20002	17287	15.2	15800	13.9	35600	31.2	12600	11.1	5270	4.6	8070	7.1	**1140**	**1.0**