Mixed-integer linear programming approaches for deterministic and stochastic lot-sizing problems

Céline Gicquel

université
PARIS-SACLAY

HABILITATION À DIRIGER DES RECHERCHES

Mixed-integer linear programming approaches for deterministic and stochastic lot-sizing problems

soutenue le 1er juillet 2021 par

CÉLINE GICQUEL

Jury

Président:	Pr Alain Denise	Université Paris Saclay, France
Rapporteurs:	Pr Stéphane Dauzères-Péres	Ecole des Mines de Saint Etienne, France
	Pr Raf Jans	HEC Montreal, Canada
	Pr André Rossi	Université Paris Dauphine, France
Examinateurs	Pr Dominique Quadri	Université Paris Saclay, France
	Pr Hande Yaman Patternote	Université KU Leuven, Belgique

Abstract

This thesis, presented in view of obtaining an accreditation to supervise research, describes the research work I carried out as an assistant professor at the Université Paris Saclay during the last ten years. This work deals with the development of mixed-integer linear programming approaches for difficult deterministic and stochastic combinatorial optimization problems, mainly coming from applications in manufacturing, supply chain management and telecommunication.

Part I is devoted to the work carried out on lot-sizing problems. We first study an extension of the discrete lot-sizing and scheduling problem in which the setup costs are sequence-dependent and propose a new family of multi-item multi-period valid inequalities to strengthen the mathematical formulation of this problem. We then consider an important aspect of production planning, namely the fact that it is based on input data which are relative to the near future and, as a consequence, are not always perfectly known at the time when the production plan has to be built. We thus study several stochastic programming approaches for lot-sizing. The first proposed approach assumes a rather simplified setting for the decision process. It namely considers a single-stage decision process in which the whole production plan is built before any additional information on the stochastic demand realization becomes available and cannot be updated afterward as the demand unfolds over time. We formulate this stochastic problem as a joint chance-constrained program and propose a new approximate solution approach for this problem called the partial sample approximation approach. Then, in order to further improve the modeling of the actual decision process, we investigate a multi-stage stochastic programming approach which explicitly takes into account the fact that production decisions are usually not made once and for all but rather adjusted over time according to the actual realizations of the uncertain parameters. Our main contribution consists in the development of a new algorithm capable of providing good-quality solutions for large-size instances of the stochastic single-item uncapacitated lot-sizing problem. This algorithm combines a nested decomposition algorithm called the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm with a cutting-plane generation approach based on known valid inequalities. Finally, in order to extend the previous work which focuses on single-item single-echelon single-resource production systems, we consider a multi-item multi-echelon multi-resource production system linked to the remanufacturing of used products and study a multi-stage stochastic programming model for this problem. We thus present a customized branch-and-cut algorithm based on a new family of valid inequalities for this problem.

In parallel to the above-mentioned work on lot-sizing, I studied among others two applied facility location problems through industrial collaborations. The corresponding work is reported in Part II of the manuscript. We first discuss a facility location problem linked to the design of the outbound logistics network of Renault. The main contribution of this work is related to the development of a tractable heuristic solution approach for this complex large-size location-routing problem. We also investigate the placement of virtual network functions in a telecommunication network to secure it against a distributed denial-of-service attack. The main novelty here consists in the development of a new robust optimization model and adversarial algorithm for this problem.

Contents

Chapter 1

Introduction

1.1 Context

Mixed-integer linear programming (MILP) deals with mathematical optimization problems in which part of the decision variables are restricted to be integers and the objective function and the constraints are linear. The birth of this field can be traced back to the seminal work of Gomory (1958). Since then, it has achieved a great success in the academic and business worlds (Jünger et al., 2008). As a consequence, mixed-integer linear programming is now a well established tool to model and solve practical optimization problems arising from applications in e.g. telecommunication, healthcare, energy, logistics or manufacturing. This success may be explained both by the wide range of optimization problems than can be tackled by mixed-integer linear programming and by the availability of powerful software products such as XpressMP, CPLEX or Gurobi which enable practitioners to solve large-size instances with a reasonable computational effort.

The core of these MILP solvers consists in a branch-and-cut algorithm, i.e. a branch-and-bound algorithm coupled with a cutting-plane generation approach.

Basically, a branch-and-bound algorithm looks for the best solution of an MILP by carrying out an implicit enumeration of all candidate solutions. This exploration relies on a search tree, each node of which represents a subset of feasible solutions. The set of all feasible solutions is thus explored by recursively splitting the feasible space into smaller spaces, i.e. by branching in the tree. Branching alone would amount to simply enumerating and evaluating all candidate solutions. To avoid this, before enumerating the candidate solutions of a branch, the algorithm estimates the quality of these solutions by computing an upper (or lower) bound on the objective value of the best solution in this branch. It then checks this bound against upper and lower estimated bounds on the optimal solution, and the branch is discarded if it cannot produce a better solution than the best one found so far by the algorithm.

The computational efficiency of a branch-and-bound algorithm heavily depends on the quality of the bound computed at each node of the search tree. In a nutshell, the better this bound, the sooner a branch can be discarded and the less nodes have to be explored before the algorithm converges. In the context of mixed-integer linear programming, this bound is most often obtained by solving the linear programming relaxation of the problem. The main advantage of this is that linear programs can be very efficiently solved by the simplex algorithm. However, in many cases, the linear programming bound is of poor quality, which negatively impacts the performance of the branch-and-bound algorithm. In order to solve this issue, branch-and-bound algorithms can be coupled with a cutting-plane generation approach, giving rise to a branch-and-cut algorithm. Basically, a cutting-plane generation approach aims at improving the quality of the linear programming bound by adding a set of linear inequalities to the problem formulation. These inequalities are chosen so as to cut away non-integer solutions that would otherwise be solutions of the continuous relaxation and to improve the value provided by the linear relaxation by restricting its feasible space. The problem of finding a cut separating a non-integer solution from the feasible space of the linear relaxation of the MILP is

called the separation problem. The branch-and-cut algorithms embedded in MILP solvers use generic cuts, i.e. cuts that may apply to any MILP or at least to a wide range of MILPs, such as Gomory fractional cuts, clique cuts or mixed-integer rounding cuts.

MILP solvers have undergone tremendous progress over the last thirty years. For instance, Bixby (2012) reported that the machine-independent computational performance of CPLEX solver improved by a factor of 29000 between the 1.2 version of the software released in 1991 and the 11.0 version released in 2007. Yet, despite this progress, there still are many MILPs whose direct resolution by a mathematical programming solver leads to prohibitive computation times. These difficulties come from two main reasons.

First, in some cases, the improvement in the quality of the linear programming bounds obtained by using the generic cuts embedded in MILP solvers is not sufficient to enable the algorithm to converge after exploring a computationally tractable number of nodes. In such cases, there is a need to use problem-specific cuts, i.e. cuts exploiting the particular structure of the optimization problem under study, and to develop an algorithm capable of efficiently solving the corresponding separation problem. Moreover, adding two many cuts to strengthen the linear relaxation of a problem may also negatively impact the computation time of a branch-and-cut algorithm by increasing the time spent at each node of the search tree to solve this linear relaxation with the simplex algorithm. The cutting-plane generation procedure should therefore be devised so as to reach the best possible trade-off between the desired formulation strengthening and the increase in the number of constraints involved in the problem formulation.

The second often encountered source of difficulty comes from the huge size of the MILP to be solved by the solver. Some MILPs can namely involve millions (or even tens of millions) of variables and constraints, leading to both memory issues and prohibitively long computation times. Such situations are found in particular when many variables and constraints are needed to obtain a modeling of the optimization problem sufficiently fine to be able to provide practically relevant solutions and/or when large instances of this problem are considered. Moreover, problems related to the size of the MILP are particularly acute when considering MILPs linked to scenario-based stochastic programming. Basically, stochastic programming is a framework enabling to model and solve optimization problems involving uncertainty. Stochastic programming relies on the fact that even if the exact value of the random problem parameters cannot be perfectly known at the time the decision has to be made, some knowledge about their possible value is available either in the form of a probability distribution or in the form of a discrete and finite set of scenarios. In this second case, the obtained stochastic program often takes the form of an MILP, the size of which is broadly proportional to the number of scenarios used to represent the possible outcomes of the random variables. The number of scenarios needed to obtain an accurate (or at least an acceptable approximate) representation of the random parameters is usually quite high and the size of the MILPs to be solved increases accordingly.

Research is thus needed to address these difficulties and further extend the scope of mixed-integer linear programming. This thesis, presented with the aim of obtaining an accreditation to supervise research, outlines my contributions to this field and focuses on two specific classes of combinatorial optimization problems: lot-sizing and facility location.

1.2 Research background

My initial background is in industrial production and logistics management. After completing my engineering degree, I worked two years as a logistics process engineer in a Procter & Gamble plant producing liquid laundry and household products. I then decided to go back to study in order to get a PhD. As I wanted to build on my previous professional experience, I applied for a doctoral research grant in industrial engineering at the Ecole Centrale Paris. This

is when I started working in the field of operations research and combinatorial optimization. More precisely, my PhD work dealt with a combinatorial optimization problem encountered in industrial production planning, namely lot-sizing.

Basically, a lot-sizing problem consists in determining when and how much to produce on a resource (such as an assembly line, a chemical reactor...) so as to satisfy the customers' demand for a set of finished products while minimizing the total cost of the production plan. This cost comprises linear production costs, fixed setup costs and inventory holding costs. Many lot-sizing problems can be quite naturally formulated as mixed-integer linear programs. However, this natural formulation usually involves a set of big-M type constraints, which translates in a poor quality of the lower bounds provided by the linear relaxation. Since the seminal work of Barany et al. (1984), there has been a lot of research to develop strong MILP formulations of lot-sizing problems (Pochet and Wolsey, 2006). During my PhD, I worked on exact solution approaches based on mixed-integer linear programming for a variant of lot-sizing problems called the discrete lot-sizing and scheduling problem (DLSP). In particular, I developed cut-and-branch algorithms based on strong reformulations and valid inequalities for an extension of the DLSP involving sequence-dependent setups.

After my PhD, I was recruited as an assistant professor at the Laboratoire de Recherche en Informatique (LRI) of the Université Paris Sud, which recently became the Laboratoire Interdisciplinaire des Sciences du Numériques and the Université Paris Saclay. At the LRI, I joined the research team on combinatorial and stochastic optimization. At that time, the team worked among others on the development of semi-definite relaxations for deterministic combinatorial optimization problems and on joint chance-constraint programming approaches for stochastic combinatorial optimization problems. This work was mostly focused on generic problems such as the quadratic assignment problem or the knapsack problem. In order to get opportunities to work and collaborate with my colleagues, I thus sought ways of applying and extending their results obtained on generic problems to lot-sizing problems. This first translated in a research project (funded by the Agence Nationale pour la Recherche through its program for young researchers) on the development of strong semi-definite relaxations for the DSLP with sequence-dependent setups.

I also started a collaboration with Jiangqian Cheng who was a post-doc in my team and we worked on joint chance-constraint programming approaches for lot-sizing under demand uncertainty. Namely, lot-sizing is about planning the activity of a resource or a set of resources for the near future. As a consequence, in practice, when solving a lot-sizing problem, we often have to rely on information (in particular on the future demand to be met) which is obtained through imperfect forecasting procedures. Hence, in many applications, it is necessary to explicitly consider the fact that the input data needed to compute the production plan are not all perfectly known at the time the production plan has to be built. This leads to handling lot-sizing as an optimization problem involving uncertainty. Since this collaboration with Jiangqian Cheng, stochastic lot-sizing has been one of my main research subjects and I am still working on it through the PhD work of Franco Quezada.

In parallel to this work on lot-sizing problems, I had the opportunity to broaden and strengthen my skills in operations research through the participation to industrial and academic collaborative projects. Regarding industrial collaborations, I worked in particular on two applied facility location problems arising in supply chain management (PhD work of Mouna Kchaou-Boujelben at Renault) and telecommunication (collaboration with a research engineer in cybersecurity at Orange). Moreover, I am currently involved in two other industrial collaborative projects. The first one is linked to the co-supervision of Bingqian Liu, a PhD student at the EDF R&D center in China working on the optimal design of local energy systems. The second one is a joint work with the consulting company DecisionBrain and colleagues from CentraleSupelec dealing with explainability for a workforce routing problem (PhD thesis of Mathieu Lerouge). As for academic collaborative projects, I initiated a collaboration with Oualid Jouini,

a former colleague at the Ecole Centrale Paris, through the PhD work of Mathilde Excoffier on call center shift scheduling under uncertainty. I also took part in a project led by my former PhD student, Mouna Kchaou-Boujelben, now an assistant professor at the United Arab Emirate University. This project dealt with facility location problems for the optimal deployment of electric vehicle charging stations. Note that a common point in all these works is the fact that they all heavily rely on mixed-integer linear programming to model and solve the studied optimization problems.

1.3 Contributions

This thesis reports the main research results corresponding to the work I carried out since I was recruited as an assistant professor in 2010. There were obtained through collaborations with colleagues and students, in particular 3 PhD students, one post-doctoral fellow and 10 master students.

1.3.1 Lot sizing

In direct line with the work I carried out for my PhD thesis, a large part of the work presented in this document is related to lot-sizing. More precisely, the presented contributions pertain to the development of solution approaches based on mixed-integer linear programming for deterministic and stochastic lot-sizing problems. They can be summarized as follows.

1. We propose a new set of valid inequalities for a deterministic multi-item lot-sizing problem called the discrete lot-sizing and scheduling problem with sequence-dependent setups. The main novelty of these valid inequalities is that they seek to better represent conflicts on multi-period time intervals between several items simultaneously requiring production on the available resource. This is in contrast with most previously known valid inequalities which focus on single-item sub-problems and do not take into account the competition between items to access the resource. These valid inequalities form the basis of a cut-and-branch algorithm, whose performance compares well with the one of the generic branch-and-cut algorithm embedded in CPLEX solver.

2. We then study a single-item capacitated lot-sizing problem in which the demand to be satisfied is subject to uncertainty. We consider a single-stage stochastic programming approach for this problem in which we seek to build the production plan before any additional information on the demand realization becomes available and do not consider the possibility of updating this plan as the demand unfolds over time. The problem is formulated as a joint chance-constrained program where the probability that an inventory shortage occurs during the planning horizon is limited to a maximum acceptable risk level. Our contribution here consists in the development of a new approximate solution approach for this problem called the partial sample approximation approach. The main advantage of the proposed method is that, unlike the previously published sample approximation approach, it only requires the introduction of additional continuous variables in the formulation and thus leads to the formulation of a deterministic mixed-integer linear program (MILP) having the same number of binary variables as the initial stochastic problem.

3. Production planning is intrinsically a multi-stage decision process in which production decisions are not made once and for all but rather adjusted over time according to the actual realizations of the uncertain parameters. In order to better exploit this flexibility

when solving stochastic lot-sizing problems, we then investigate a multi-stage stochastic programming approach for a single-item uncapacitated lot-sizing problem. We consider that the underlying stochastic input process has a finite probability space and represent the information on the evolution of the uncertain parameters by a discrete scenario tree. This leads to the formulation of a very large-size MILP. In order to solve it, we propose a new algorithm which combines a recently published nested decomposition algorithm called the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm with a cutting-plane generation approach based on known valid inequalities. The reported computational results show that this new algorithm clearly outperforms both the generic branch-and-cut algorithm embedded in CPLEX solver and the SDDiP algorithm at solving instances of the problem involving large-size scenario trees.

4. The single-item uncapacitated lot-sizing problem mentioned above relies on rather strong assumptions on the production system. Clearly, there is a need to study stochastic lot-sizing models with a better practical relevancy to meet the industrial needs in terms of production planning. As a first step towards closing this gap, we investigate an application of multi-stage stochastic programming in a production planning field where uncertainty is particularly present: the remanufacturing of used products. We thus study here a remanufacturing system which involves three key processes: disassembly of used products brought back by customers, refurbishing of the recovered parts and reassembly into like-new finished products. Uncertainties related to the quantity and quality of returned products, the customers' demand, and the costs are taken into account and represented through a scenario tree. Our contribution here consists in the development of a new set of valid inequalities for this problem. These valid inequalities are used within a branch-and-cut algorithm, the performance of which compares well with the one of CPLEX solver.

Note that all the contributions described here heavily rely on MILP solvers such as CPLEX. However, in link with the discussion presented at the end of Subsection 1.1, we try to circumvent the numerical difficulties encountered when trying to directly solve the obtained MILPs with a commercial solver. Thus, Contributions 1 and 4 aim at improving the quality of the lower bounds used at each node of the search tree through the use of problem-specific cuts so as to improve the numerical efficiency of the solver. Moreover, Contributions 2 and 3 seek to facilitate the resolution of the problem by the solver basically by reducing the size of the MILP to be solved. In Contribution 2, this is achieved by reducing the number of binary variables introduced in the formulation. As for Contribution 3, it exploits a decomposition approach to split the problem into a set of smaller sub-problems and uses the solver only to solve the obtained sub-problems which are MILPs of much smaller and tractable size.

1.3.2 Facility location

In parallel to the research on lot-sizing problems described in Subsection 1.3.1, I had the opportunity to study other deterministic and stochastic combinatorial optimization problems, mainly through the participation to academic and industrial collaborative projects. Part of this work dealt with facility location problems.

Broadly speaking, facility location problems aim at finding the optimal placement of facilities in a network so as to satisfy the demand of a set of demand points. Depending of the context, facilities can be e.g. plants or warehouses in a supply chain network, 5G antennas in a telecommunication network, hospitals or fire stations in a city road network or electricity generators in an electric micro-grid. Many facility location problems can rather naturally be formulated as mixed-integer linear programs, the size of which depends among others on the

size of the underlying network. Hence, numerical difficulties frequently arise when trying to directly solve facility location problems involving large-size networks.

The contributions presented in this thesis thus pertain to the development of models and solution approaches based on mixed-integer linear programming to solve two real-life facility location problems arising from applications in supply chain management (Renault) and telecommunication (Orange). They can be summarized as follows.

1. The first work deals with a deterministic facility location problem aiming at optimally designing the French distribution network of the car maker Renault. The objective is to determine where to locate distribution centers to optimize the distribution of cars from the assembly plants (or import ports) to the car dealers. The problem thus consists in selecting a subset of previously identified candidate location sites where distribution centers should be opened, to assign car dealers to the open distribution centers and to route the flows of cars in this transportation network such that the total distribution costs are minimized. The numerical complexity of this problem mainly comes from the need to incorporate two operational constraints into the modeling of this strategic problem. First, the transportation from distribution centers to car dealers is carried out through distribution routes starting at a DC, successively visiting several car dealers and coming back to the DC. Second, a large number of minimum volume constraints must be added to the problem formulation to guarantee that the volume transported on each open link is large enough to ensure transportation of cars by full truckload without deteriorating the customer delivery time. To handle this problem, we propose a two-phase heuristic in which car dealers are first grouped into clusters and the resulting location-allocation problem is then solved by a LP-based rounding heuristic.

2. The second work deals with a robust set covering problem arising in the context of cybersecurity in telecommunication networks. The problem aims at optimally placing virtual network functions in a 5G network in order to counter an on-going distributed denial-or-service (DDoS) attack and prevent the hackers from causing damages to their target. We take the perspective of an internet service provider such as the French company Orange which aims at providing a DDoS mitigation service to its customers in a 5G network. The main difficulty of this problem comes from the fact, due to a feature called network slicing, the exact routing in the network of the attack flow between its ingress points and its target is not known and that we would like to be able to stop all this flow whatever its routing in the network. We thus present a new robust optimization model for this problem and propose to solve it by using an adversarial approach.

Similar to the work related on lot-sizing, the solution approaches proposed for these two facility location problems use MILP solvers. In both cases, the strategy basically consists in reducing the size of the MILP to be solved to make it tractable by the solver. In the car distribution network case, this is achieved by decomposing the initial location-routing problem into a set of three smaller sub-problems (one for clustering the car dealers, one for locating the DCs, one for routing the product flows in the network) which are solved sequentially. In the telecommunication network case, the MILP size reduction is also achieved through a decomposition of the initial problem into a master problem and an adversarial sub-problem which are iteratively solved.

1.4 Manuscript organization

This thesis is organized into two parts.

Part I presents the work carried out on lot-sizing problems and comprises 5 chapters. Chapter 2 provides some background on lot-sizing problems to facilitate the reading of the

manuscript. Chapter 3 is devoted to the deterministic discrete lot-sizing and scheduling problem with sequence-dependent setups and introduces among others a new family of multi-item multi-period valid inequalities for this problem. Chapters 4 to 6 consider stochastic lot-sizing problems. More precisely, Chapter 4 focuses on the single-item capacitated lot-sizing problem with stochastic demand, investigates a joint chance-constrained model for this problem and describes the new partial sample approximation approach we propose to handle it. Chapter 5 focuses on a multi-stage stochastic programming approach for the stochastic single-item uncapacitated lot-sizing problem. It presents the extension of the Stochastic Dual Dynamic integer Programming algorithm we developed for this problem. Finally, Chapter 6 deals with a multi-stage stochastic lot-sizing problem arising in the context of remanufacturing of used products. In particular, new valid inequalities for this problem are investigated.

Part II presents the work carried out on facility location problems and comprises 2 chapters. Chapter 7 discusses the design of the French distribution network of Renault whereas Chapter 8 focuses on the placement of virtual network functions in a telecommunication network to secure it against a distributed denial-of-service attack.

Finally, Chapter 9 provides a general conclusion and discusses several possible directions for further research.

Part I

Lot-sizing

Chapter 2

Background

2.1 Introduction

This chapter aims at providing some background on lot-sizing problems to facilitate the reading of the manuscript. Note that our objective is not to provide a general overview nor an exhaustive literature review on lot-sizing but rather to introduce the notions and definitions which are necessary to understand the work presented in Chapters 3 to 6.

Lot-sizing is defined by Kuik et al. (1994) as "the clustering of items for transportation or manufacturing at the same time". Lot-sizing arises in production whenever set-up operations are required in order to prepare the production resource for the processing of a new type of product. Set-up actions can involve many different operations such as cleaning, preheating, tool change, machine calibration or test runs. Set-up costs account e.g. for the additional workforce needed to prepare the resource, for the production loss during the resource downtime, for the raw materials consumed during set-up operations, etc. In lot-sizing, we model these set-up costs as fixed costs, the amount of which does not depend on the quantity of items produced after the set-up operation. To minimize these set-up costs and obtain a more efficient use of production resources, production should be run using large lot sizes. However, this production policy generates inventory as the production cannot be synchronized with the actual demand pattern. Namely, products must be held in inventory between the time they are produced and the time they are actually used to satisfy customer demand. This generates inventory holding costs mainly because of tied up capital. The objective of lot-sizing is thus to reach the best possible trade-off between set-up and inventory holding costs while taking into account both the customer demand satisfaction and the technical limitations of the production system.

When all the input data needed to compute the optimal production plan are perfectly known at the time the plan should be built, lot-sizing is a deterministic combinatorial optimization problem. Section 2.2 introduces the deterministic lot-sizing models which are studied as such or used as starting points in the works discussed later in this document. However, lot-sizing is about planning the activity of a resource (or a set of resources) for the near future and in practice, we often have to rely on information about the value of some input parameters which is obtained through imperfect forecasting procedures. Hence, in many applications, assuming that all the input data needed to compute the production plan are deterministically known is not realistic. Section 2.3 thus discusses several stochastic lot-sizing models.

2.2 Deterministic lot-sizing

We introduce here the deterministic lot-sizing models investigated later in the document. We first present single-item lot-sizing models before discussing some multi-item lot-sizing models.

2.2.1 Single-item lot-sizing

Single-item uncapacitated lot-sizing

The simplest available lot-sizing model is the single-item uncapacitated lot-sizing problem (called ULS) introduced by Wagner and Whitin (1958). It considers a single type of item and aims at planning the production of this item over a finite discrete-time planning horizon involving a set $\mathcal{T} = \{1...T\}$ of periods. Producing a positive amount in period $t \in \mathcal{T}$ incurs a fixed set-up cost f_t together with a production cost g_t per unit produced and an inventory holding cost h_t per unit held in stock between two consecutive periods. The objective is to build a production plan such that the customers' demand d_t is met in each time period t and the total costs, i.e. the sum of setup, production, and inventory holding costs over the whole planning horizon, are minimized.

This problem can be formulated as a mixed-integer linear program by introducing the following decision variables:

- x_t: quantity produced in period t,
- y_t: set-up state of the resource. $y_t = 1$ if the resource is set-up to produce the item in t, $y_t = 0$ otherwise,
- s_t: inventory level at the end of period t.

With this notation, the ULS can be formulated as follows:

$$
\begin{cases}
Z^* = \min \sum_{t=1}^{T} \left(f_t y_t + g_t x_t + h_t s_t\right) & & (2.1)\\
x_t \leq M_t y_t & \forall t \in \mathcal{T} & (2.2)\\
s_t = s_{t-1} + x_t - d_t & \forall t \in \mathcal{T} & (2.3)\\
x_t \geq 0 & \forall t \in \mathcal{T}. & (2.4)\\
s_t \geq 0 & \forall t \in \mathcal{T}. & (2.5)\\
y_t \in \{0,1\} & \forall t \in \mathcal{T}. & (2.6)
\end{cases}
$$

The objective function (2.1) minimizes the sum of the set-up, production and inventory holding costs over the whole planning horizon. Constraints (2.2) ensure that, if production takes place in period t, the corresponding setup costs are incurred. Note that the value of constant M_t can be set to the value of the cumulative remaining demand to be satisfied till the end of the horizon, i.e. to $\sum_{\tau=t}^{T} d_\tau$. Constraints (2.3) are the inventory balance constraints. Together with Constraints (2.5), they ensure the timely satisfaction of the demand.

The ULS is known to be solvable in strongly polynomial time. A simple dynamic programming algorithm was proposed by Wagner and Whitin (1958). It is based on the zero-inventory-ordering property, i.e. production is undertaken in a period only if the entering inventory level drops to zero, and runs in $\mathcal{O}(T^2)$ time. This time complexity was later improved to $\mathcal{O}(T \log T)$ by Aggarwal and Park (1993) and Wagelmans et al. (1992). Moreover, Barany et al. (1984) proposed a family of valid inequalities, known as the (ℓ, S) inequalities. These inequalities, when added to Constraints (2.2)-(2.5), provide a full description of the convex hull of the feasible space of ULS.

Single-item capacitated lot-sizing

The ULS is an uncapacitated model which relies on the assumption that there is no limit on the quantity that can be produced in a period. Yet, in most practical situations, the production capacity cannot be assumed infinite. Hence, a first way to improve the practical relevancy

of problem ULS is to consider a limited production capacity c_t in each time period and a capacity consumption v_t per unit of item produced. The resulting problem, called LS_C, can be formulated as followed:

$$\begin{cases} Z^* = \min \sum_{t=1}^{T} (f_t y_t + g_t x_t + h_t s_t) & & (2.7) \\ v_t x_t \leq c_t y_t & \forall t \in \mathcal{T} & (2.8) \\ s_t = s_{t-1} + x_t - d_t & \forall t \in \mathcal{T} & (2.9) \\ x_t \geq 0 & \forall t \in \mathcal{T} & (2.10) \\ s_t \geq 0 & \forall t \in \mathcal{T} & (2.11) \\ y_t \in \{0,1\} & \forall t \in \mathcal{T} & (2.12) \end{cases}$$

Note how the big-M constant M_t has been replaced by a finite production capacity c_t in Constraints (2.9).

The complexity status of Problem LS_C is investigated among others by Florian and Klein (1971), Bitran and Yanasse (1982) and van Hoesel and Wagelmans (1996) and depends mainly on the structure of the capacity parameter. Thus, LS_C is polynomially solvable in $O(T^3)$ when the value of the production capacity c_t is constant over time: see van Hoesel and Wagelmans (1996). But Bitran and Yanasse (1982) show that most other variants of the problem in which c_t varies over time are NP-hard.

Many other extensions of the ULS have been proposed since the seminal work of Wagner and Whitin (1958) in order to improve its applicability in real-life situations by taking into account complicating features relative among others to the costs, the production resource or the demand service policy. We refer the reader to Brahimi et al. (2006) and Brahimi et al. (2017) for comprehensive surveys on the single-item dynamic lot-sizing problem.

2.2.2 Multi-item single-echelon lot-sizing

In many cases, the available resource is not dedicated to the production of a single type of item, but rather shared between multiple items. In the problem modeling, this situation can be handled either through large bucket or through small bucket models: see e.g. Drexl and Kimms (1997). Large bucket models rely on a coarse discretization of the planning horizon into a small number of long time periods (typically a week or a month) in which items of multiple types may be produced. In contrast, small bucket models use a fine discretization of the planning horizon into a large number of short time periods (typically an hour or a shift) and assume that at most one type of item may be produced per period.

Capacitated Lot-Sizing Problem

One of the most widely known large bucket lot-sizing models is the Capacitated Lot-Sizing Problem or CLSP. In the CLSP, we wish to determine the optimal production plan for a set $\mathcal{I} = \{1, ..., I\}$ of items over an horizon involving T periods. The demand for item $i \in \mathcal{I}$ to be satisfied at the end of period $t \in \mathcal{T}$ is denoted by d_{it}. In each time period, there is a limited production capacity c_t to be allocated between the products. Let v_{it} be the amount of production capacity needed to produce one unit of item i at time period t. As for the costs, we consider f_{it}, the fixed set-up cost to be paid if production for item i occurs during t, g_{it} the unit production cost for i in t and h_{it} the inventory holding cost per unit of item i held in inventory at the end of t.

We introduce the following decision variables:

- x_{it}: quantity of item i produced in period t,

- y_{it}: resource set-up state variable. $y_{it} = 1$ if the resource is set-up to produce i in t, $y_{it} = 0$ otherwise,
- s_{it}: inventory level of item i at the end of period t.

With this notation, the CLSP can be formulated as follows:

$$\begin{cases} Z^* = \min \sum_{i=1}^{I} \sum_{t=1}^{T} \left(f_{it} y_{it} + g_{it} x_{it} + h_{it} s_{it} \right) & & (2.13) \\ \sum_{i=1}^{I} v_{it} x_{it} \leq c_t & \forall t \in \mathcal{T} & (2.14) \\ v_{it} x_{it} \leq c_t y_{it} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.15) \\ s_{it} = s_{i,t-1} + x_{it} - d_{it} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.16) \\ x_{it} \geq 0 & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.17) \\ s_{it} \geq 0 & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.18) \\ y_{it} \in \{0, 1\} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.19) \end{cases}$$

The objective function (2.13) seeks to minimize the total set-up, production and inventory holding costs. Constraints (2.14) ensure that the total amount of capacity consumed by all the items produced on the resource during period t stays below the available production capacity c_t. Constraints (2.15)-(2.19) are similar to the constraints involved in the single-item LS-C problem.

The CLSP is known to be an NP-hard problem: see e.g. Florian et al. (1980) and Bitran and Yanasse (1982). As a consequence, a large variety of Operations Research techniques has been investigated to solve it: see e.g. the literature reviews provided by Karimi et al. (2003), Jans and Degraeve (2007) and Buschkühl et al. (2010).

Discrete Lot-sizing and Scheduling Problem

Among the small bucket lot-sizing models is the Discrete Lot-sizing and Scheduling Problem or DLSP. In the DLSP, a single type of product may be produced in each time period and a discrete or all-or-nothing production policy is used, i.e. production is either carried out at full capacity or does not occur. Moreover, as we use short time periods, a production lot can extend over several periods. In this case, set-up costs should not occur in each production period but only in the periods in which the production of a new lot begins. To model this situation, we introduce, in addition to the x_{it}, y_{it} and s_{it} variables defined above for the CLSP, binary start-up variables $z_{it}, i \in \mathcal{I}, t \in \mathcal{T}$. These new variables are defined by $z_{it} = 1$ if there is a start-up, i.e. if production of a new lot begins, in period t for item i, 0 otherwise.

With this notation, the DLSP can be formulated as follows:

$$
\begin{cases}
Z^* = \min \sum_{i=1}^{I} \sum_{t=1}^{T} \left(f_{it} z_{it} + g_{it} x_{it} + h_{it} s_{it} \right) & & (2.20) \\
\sum_{i=1}^{I} y_{it} \leq 1 & \forall t \in \mathcal{T} & (2.21) \\
v_{it} x_{it} = c_t y_{it} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.22) \\
z_{it} \geq y_{it} - y_{i,t-1} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.23) \\
s_{it} = s_{i,t-1} + x_{it} - d_{it} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.24) \\
x_{it}, s_{it} \geq 0 & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.25) \\
y_{it} \in \{0,1\}, z_{it} \in \{0,1\} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.26)
\end{cases}
$$

The objective function (2.20) seeks to minimize the total start-up, production and inventory holding costs. Constraints (2.21) ensure that at most one item is produced in each production period. Constraints (2.22) enforce the all-or-nothing production policy: if some production of item i occurs during t, the amount produced uses all the available production capacity. Constraints (2.23) link the new start-up variables with the set-up variables and make sure that start-up costs are incurred each time a production lot for a new item begins.

Salomon et al. (1991) showed that determining whether there is a feasible solution of the DLSP can be done in polynomial time but that solving it to optimality is NP-hard.

Note that there are many other variants of large and small bucket multi-item lot-sizing models. The reader is referred to e.g. the surveys provided by Drexl and Kimms (1997) and Jans and Degraeve (2008) for a more general description of multi-item lot-sizing models.

2.2.3 Multi-item multi-echelon lot-sizing

Multi-echelon lot-sizing problems arise in production system whenever the finished product is not obtained directly by transforming the raw materials through a single processing step but rather through a sequence of processing steps, each one using either raw materials or partially transformed intermediate products as input. The relationship between the finished product and these intermediate products or components is described by the bill of materials and is referred to as the product structure in the lot-sizing literature. An introduction on multi-echelon lot-sizing models involving various forms of product structure can be found e.g. in Pochet and Wolsey (2006). We focus here on the simplest possible structure, namely a production in series structure.

In a serial product structure, a linear sequence of I steps are needed to transform the raw materials into a finished product, each one being carried out on a dedicated resource. The intermediate product obtained after step $i = 1...I$ is referred to as item i and is obtained by processing one unit of item $i-1$ on the resource in charge of step i of the transformation. Note that, with this definition, product I corresponds to the finished product.

We denote by f_{it} (resp. g_{it}) the fixed set-up cost (resp. the unit production cost) on resource i and by h_{it} the unit inventory holding cost for item i at time period t. d_t corresponds to the demand for the finished product to be satisfied at the end of period t.

We introduce the following decision variables:

- x_{it}: quantity of item i produced in period t,
- y_{it}: set-up state of resource i in t. $y_{it} = 1$ if the resource i is set-up for production in t, $y_t = 0$ otherwise,

- s_{it}: inventory level of item i at the end of period t.

With this notation, the production in series extension of the ULS can be formulated as follows:

$$\begin{cases} Z^* = \min \sum_{i=1}^{I} \sum_{t=1}^{T} \left(f_{it} y_{i,t} + g_{it} x_{i,t} + h_{it} s_{i,t} \right) & & (2.27) \\ x_{i,t} \leq M_t y_{i,t} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.28) \\ s_{i,t} = s_{i,t-1} + x_{i,t} - x_{i+1,t} & \forall t \in \mathcal{T}, i \in \mathcal{I} \setminus \{I\} & (2.29) \\ s_{I,t} = s_{I,t-1} + x_{I,t} - d_t & \forall t \in \mathcal{T} & (2.30) \\ x_{i,t} \geq 0 & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.31) \\ s_{i,t} \geq 0 & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.32) \\ y_{i,t} \in \{0,1\} & \forall t \in \mathcal{T}, i \in \mathcal{I} & (2.33) \end{cases}$$

The objective function (2.27) seeks to minimize the total set-up, production and inventory holding costs. Constraints (2.28) ensure that, if production takes place on resource i in period t, the corresponding setup costs are incurred. Note that, similar to what is done for the ULS, the value of constant M_t can be set to $\sum_{\tau=t}^{T} d_\tau$. Constraints (2.29)-(2.30) are the inventory balance constraints. Constraints (2.30) are similar to the constraints found in the ULS formulation as they involve a demand term d_t which is an input parameter. However, Constraints (2.29) involve a dependent demand term $x_{i+1,t}$: the value of the demand to be satisfied at echelon i namely depends on the production plan which will be built for production echelon $i+1$.

Zangwill (1969) proposed a dynamic algorithm for Problem (2.27)- (2.33) which runs in $O(IT^4)$. A recent review on complexity results for more general variants of multi-echelon lot-sizing problems can be found in Brahimi et al. (2017).

2.3 Stochastic lot-sizing

As mentioned in Section 2.1, lot-sizing is about planning the activity of a resource or a set of resources for the near future. As a consequence, in practice, when solving a lot-sizing problem, we often have to rely on information (in particular on the future demand to be met) which is obtained through imperfect forecasting procedures. Forecasting errors lead both to stockouts occurring with unsatisfied demands and to inventory levels higher than planned. Hence, in many applications, it is necessary to explicitly consider that the input data needed to compute the production plan are not all deterministically known and to handle lot-sizing as an optimization problem involving uncertainty.

A wide variety of approaches have been proposed to handle production planning and lot-sizing under uncertainty: see Aloulou et al. (2014) for a general overview and Tempelmeier (2013) and Brahimi et al. (2017) for literature reviews focusing on single-item dynamic lot-sizing problems. In what follows, we assume that an accurate probabilistic description of the random variables is available under the form of probability distributions and thus only discuss stochastic programming approaches for lot-sizing under uncertainty. Moreover, for the sake of simplicity, we will use as a starting point the single-item uncapacitated lot-sizing (ULS) problem described in Subsection 2.2.1 and focus on the case in which only the demand is subject to uncertainty.

Let $\tilde{d}$ denote the random vector representing the stochastic demand over the planning horizon. Replacing d_t by $\tilde{d}_t$, for all t, in the formulation of the deterministic ULS leads to the following stochastic formulation:

$$
\begin{cases}
Z^* = \min \sum_{t=1}^{T} \left(f_t y_t + g_t x_t + h_t \tilde{s}_t \right) & & (2.34) \\
x_t \leq M_t y_t & \forall t \in \mathcal{T} & (2.35) \\
\tilde{s}_t = \tilde{s}_{t-1} + x_t - \tilde{d}_t & \forall t \in \mathcal{T} & (2.36) \\
x_t \geq 0 & \forall t \in \mathcal{T} & (2.37) \\
\tilde{s}_t \geq 0 & \forall t \in \mathcal{T} & (2.38) \\
y_t \in \{0,1\} & \forall t \in \mathcal{T} & (2.39)
\end{cases}
$$

A direct consequence of the introduction of $\tilde{d}$ in the problem modeling is that the inventory level at the end of each period t now also is a random variable denoted by $\tilde{s}_t$ in Problem (2.34)-(2.39). Under a stochastic demand, it may be very difficult, and even impossible if the probability distribution of $\tilde{d}_t$ has an infinite support for some period t, to find a production plan ensuring that, whatever the realization of the stochastic demand, the inventory on hand will be sufficient to satisfy all the demand. In other words, it may not be possible to find a production complying with Constraints (2.38) for any realization of $\tilde{d}$ and, even if the production capacity is unlimited, stockouts may happen. An additional difficulty comes from the fact that the objective function now involves stochastic terms corresponding to stochastic inventory holding costs.

Several stochastic programming approaches have been investigated to handle these difficulties. They differ among others with respect to the number of stages involved in the decision process. In the context of stochastic programming, a stage in the decision process can be basically defined as a future point in time at which new information on the stochastic parameters becomes available and decisions based on this newly available information have to be made. Note that stages do not necessarily coincide with planning periods, in particular a stage may comprise several periods. Namely, in lot-sizing, the time discretization used by the decision-makers to plan the production activities is usually finer than the one used to update the demand forecasts and possibly readjust the production plan. A planning period may thus typically correspond to an 8-hours shift or a day whereas a stage may correspond to a week or a month.

In what follows, we discuss single-stage, two-stage and multi-stage stochastic programming models for lot-sizing under uncertain demand.

2.3.1 Single-stage stochastic lot-sizing

In single-stage stochastic lot-sizing models, the value of all production decision variables, i.e. setup and production variables, is decided upon at the beginning of the planning horizon prior to the realization of the uncertain demand. In this type of model, stockouts are managed through the introduction of chance constraints imposing that the probability of a stockout stays below an acceptable risk level defined by the production manager. Disjoint chance constraints impose an upper bound on the probability of a stockout within each planning period: see e.g. Bookbinder and Tan (1988) and Chen (2007). As mentioned by Tempelmeier (2007), they correspond to defining a minimum value to the period α-service level or ready rate often used in supply chain management. Joint chance constraints impose an upper bound on the probability of a stockout within the whole planning horizon: see e.g. Beraldi and Ruszczyński (2002), Küçükyavuz (2012) and Zhang et al. (2014). They can be understood as a way of defining a minimum value to the horizon α-service level.

Modeling Problem (2.34)- (2.39) as a disjoint chance-constraint program leads to the following formulation:

$$
\begin{cases}
Z^* = \min \sum_{t=1}^{T} \left(f_t y_t + g_t x_t + h_t \mathbb{E}[\tilde{s}_t] \right) & & (2.40) \\
x_t \leq M_t y_t & \forall t \in \mathcal{T} & (2.41) \\
\tilde{s}_t = \tilde{s}_{t-1} + x_t - \tilde{d}_t & \forall t \in \mathcal{T} & (2.42) \\
x_t \geq 0 & \forall t \in \mathcal{T} & (2.43) \\
\Pr(\tilde{s}_t \geq 0) \geq 1 - \epsilon & \forall t \in \mathcal{T} & (2.44) \\
y_t \in \{0, 1\} & \forall t \in \mathcal{T} & (2.45)
\end{cases}
$$

Here, ϵ is an input parameter giving the maximum acceptable stockout risk level and $1 - \epsilon$ can be understood as the target ready rate level defined by the production manager. In Problem (2.40)- (2.45), the potentially infeasible constraints (2.38) are thus replaced by a series of disjoint chance-constraints, each one imposing that the probability that the demand in each period is satisfied without any delay stays above the targeted ready rate. Moreover, note how in the objective function, the terms corresponding to the stochastic inventory holding costs have been replaced by terms minimizing the expected inventory holding costs.

Finally, when Problem (2.34)- (2.39) is modeled as a joint chance-constraint program, Constraints (2.44) are replaced by a single joint chance constraint $\Pr(\tilde{s}_t \geq 0, \forall t \in \mathcal{T}) \geq 1 - \epsilon$ enforcing a minimum value to the probability that all demand satisfaction constraints are simultaneously respected.

2.3.2 Two-stage stochastic lot-sizing

Similar to one-stage stochastic lot-sizing models, most two-stage stochastic lot-sizing models assume that the value of all production decision variables, i.e. setup and production variables, is decided upon at the beginning of the planning horizon prior to the realization of the uncertain demand. They however differ in the way stockouts are handled in the model. Namely, whereas one-stage stochastic lot-sizing models consider stockouts as undesirable events whose probability of occurrence should be limited, two-stage stochastic lot-sizing models introduce recourse actions, i.e. corrective actions that may be taken after the random demand has realized in order to make the initial production plan feasible. One widely used recourse action consists in backlogging the demand, i.e. in delaying the demand satisfaction for some customers for one or several periods. Hence, two-stage stochastic lot-sizing models usually consider setup and production variables as first-stage decision variables and inventory and backlogging variables as second-stage decision variables. Note that these models implicitly assume that the information on the actual demand realization is revealed in one go for the whole planning horizon and that this knowledge can be exploited to optimize the recourse actions.

Modeling Problem (2.34)- (2.39) as a two-stage stochastic program with inventory holding and backlogging as recourse actions leads to the following formulation:

$$
\begin{cases}
Z^* = \min \sum_{t=1}^{T} \left(f_t y_t + g_t x_t + h_t \mathbb{E}[\tilde{s}_t] + e_t \mathbb{E}[\tilde{b}_t] \right) & & (2.46) \\
x_t \leq M_t y_t & \forall t \in \mathcal{T} & (2.47) \\
\tilde{s}_t - \tilde{b}_t = \tilde{s}_{t-1} - \tilde{b}_{t-1} + x_t - \tilde{d}_t & \forall t \in \mathcal{T} & (2.48) \\
x_t \geq 0 & \forall t \in \mathcal{T} & (2.49) \\
\tilde{s}_t, \tilde{b}_t \geq 0 & \forall t \in \mathcal{T} & (2.50) \\
y_t \in \{0, 1\} & \forall t \in \mathcal{T} & (2.51)
\end{cases}
$$

Here, $\tilde{b}_t$ represents the random amount of demand backlogged at the end of period t and e_t the unit backlogging penalty cost. Problem (2.34)- (2.39) was studied among others by Vargas (2009) and Piperagkas et al. (2012). As getting an accurate assessment of the unit backlogging penalty cost e_t is not always possible, Tempelmeier and Herpers (2011) propose to replace the penalty term in the objective function by a fill rate constraint imposing an upper bound on the amount of expected backlog $\mathbb{E}[\tilde{b}_t]$ in each period.

2.3.3 Multi-stage stochastic lot-sizing

In practice, lot-sizing is not a one or two-stage decision process in which all production decisions are made once and for all at the beginning of the planning horizon. Instead, it is intrinsically a multi-stage decision process, i.e. a process involving a sequence of production decisions that are made in reaction to outcomes of the random demand as they become known. In terms of mathematical modeling, this means that there is another way of dealing with the feasibility issue on Constraints (2.38) encountered in Problem (2.34)-(2.39). It consists in postponing the setup and production decisions relative to period t up to the point in time at which the actual value of the random demand $\tilde{d}_t$ will be accurately known.

For the sake of simplicity, let us assume here that each decision stage comprises a single production planning period. In this case, the multi-stage decision dynamics discussed above can be described as follows: realization of $\tilde{d}_1 \rightarrow$ decision on $(x_1, y_1) \rightarrow$ realization of $\tilde{d}_2 \rightarrow$ decision on $(x_2, y_2) \rightarrow ... \rightarrow$ realization of $\tilde{d}_T \rightarrow$ decision on (x_T, y_T).

At the beginning of each stage, i.e. of each period t, we thus first observe the realization of the random demand for this stage. We then make production decisions (x_t, y_t) for the current stage t taking into account both the observation of $\tilde{d}_t$ and the entering inventory level s_{t-1} resulting from the decisions made at stage $t-1$. While making these decisions, the objective is to minimize not only the production costs relative to stage t but also all the expected future costs.

We denote by $\tilde{d}_{[t,t']}$ the sequence of random demand variables corresponding to stages t to t' and by $d_{[t,t']}$ a realization of this vector on the time interval $[t, t']$. $C_t(x_t, y_t, s_t, d_t) = f_t y_t + g_t x_t + h_t s_t$ represents the total setup, production and inventory holding costs at stage t as a function of the production decisions (x_t, y_t, s_t) and of the realization d_t of the demand whereas $F_t(s_{t-1}, d_t) = \{(x_t, y_t, s_t) \in \mathbb{R}^+ \times \{0,1\} \times \mathbb{R}^+ | x_t \leq M_t y_t, s_t = s_{t-1} + x_t - d_t\}$ represents the feasible space at stage t as a function of the entering inventory level s_{t-1} and of the realization d_t of the demand.

Using this notation, the multi-stage decision dynamics described above leads to the following nested formulation.

$$
\begin{aligned}
Z^* = \min_{F_1} \Big\{ & C_1(x_1, y_1, s_1) \\
& + \mathbb{E}_{\tilde{d}_{[2,T]}|d_{[1,1]}} \Big[\min_{F_2(s_1,d_2)} \big\{ C_2(x_2, y_2, s_2, d_2) \\
& \quad + ... \\
& \quad + \mathbb{E}_{\tilde{d}_{[\sigma,T]}|d_{[1,\sigma-1]}} \Big[\min_{F_\sigma(s_{\sigma-1},d_\sigma)} \big\{ C_\sigma(x_\sigma, y_\sigma, s_\sigma, d_\sigma) \\
& \quad + \mathbb{E}_{\tilde{d}_{[\sigma+1,T]}|d_{[1,\sigma]}} \big[... \\
& \quad + \mathbb{E}_{\tilde{d}_{[T,T]}|d_{[1,T-1]}} \Big[\min_{F_T(s_{T-1},d_T)} C_T(x_T, y_T, s_T, d_T) \Big] \big] \big\} \Big] \Big\}
\end{aligned}
\tag{2.52}
$$

Here $\mathbb{E}_{\tilde{d}_{[\sigma,T]}|d_{[1,\sigma-1]}}[\,\cdot\,]$ denotes the expectation of $(\cdot)$ computed using the conditional distribution of the random vector $\tilde{d}_{[\sigma,T]}$ given the realization $d_{[1,\sigma-1]}$ of the demand over stages 1 to

$\sigma - 1$. Note how, in Formulation (2.52), the decisions relative to stage σ, $(x_\sigma, y_\sigma, s_\sigma)$, are made based on the realization of demand up to stage σ, i.e. are made given an observation $d_{[1,\sigma]}$. These decisions should lie within the feasible space described by $F_\sigma(s_{\sigma-1}, d_\sigma)$ and seek to minimize both the 'immediate costs', i.e. the costs $C_\sigma(x_\sigma, y_\sigma, s_\sigma, d_\sigma)$ relative to stage σ, but also the future expected costs computed using $\mathbb{E}_{\tilde{d}_{[\sigma+1,T]}|d_{[1,\sigma]}}\left[\,\cdot\,\right]$.

One way to develop computationally tractable solution approaches for this problem consists in approximating the stochastic process $(\tilde{d}_1, ..., \tilde{d}_T)$ by a process having finitely many realizations in the form of a scenario tree. With a slight abuse of notation, we will refer to this scenario tree by mentioning only its set of nodes $\mathcal{V}$. Each node $n \in \mathcal{V}$ corresponds to a single time period (and thus a single decision stage in the present case) denoted by t^n. Let $\mathcal{V}^t$ be the set of nodes belonging to time period t. Each node n represents the state of the system that can be distinguished by the information unfolded up to time period t^n, i.e. each node corresponds to a partial realization $d_{[1,t^n]}$ of the demand vector $\tilde{d}$ up to period t^n. Each node n has a unique predecessor node denoted by a^n and belonging to time period $t^n - 1$. As production decisions for stage 1 are made after the realization of the demand for this stage, there is a single node at stage 1, corresponding to the root node of the scenario tree. It is indexed by 1 and by convention, a^1 is set to 0.

At any non-leaf node of the tree, one or several branches indicate future possible outcomes of the random variables from the current node. Let $\mathcal{C}(n)$ be the set of immediate children of node n, $\mathcal{V}(n)$ the sub-tree of $\mathcal{V}$ rooted in n and $\mathcal{L}(n)$ the set of leaf nodes belonging to $\mathcal{V}(n)$. The conditional probability of moving from node n to one of its children $m \in \mathcal{C}(n)$ is given by π^{nm} and the probability associated with the state represented by node n is denoted by ρ^n. ρ^n can be computed by induction using $\rho^1 = 1$ and $\rho^n = \pi^{a^n,n}\rho^{a^n}$ for $n \in \mathcal{V} \setminus \{1\}$. The set of nodes on the path from node n to node m is denoted by $\mathcal{P}(n, m)$. A scenario is defined as a path $\mathcal{P}(1, l)$ from the root node to a leaf node $l \in \mathcal{L}(1)$ in the scenario tree and represents a possible outcome of the stochastic input parameters over the whole planning horizon.

Figure 2.1 displays a small illustrative scenario tree corresponding to a situation in which we plan production over an horizon spanning $T = 3$ periods using a decision process involving $\Sigma = 3$ decision stages. The tree involves $|\mathcal{V}| = 10$ nodes and $|\mathcal{V}^T| = 7$ scenarios. The predecessor of node 2 is $a^2 = 1$ and its set of children is given by $\mathcal{C}(2) = \{5, 6\}$.

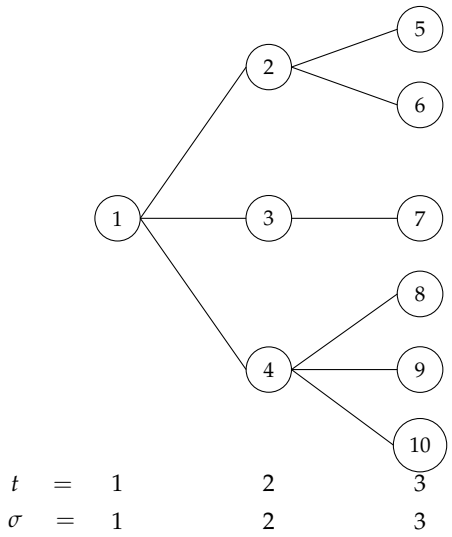

FIGURE 2.1: Scenario tree structure

Recall that, in the present setting, the decisions in a stage are made after observing the data realization up to this stage. The value of (x_t, y_t, s_t) at stage t is thus decided upon depending

of a given realization of the $d_{[1,t]}$ up to stage t, which corresponds to a given node $n \in \mathcal{V}^t$. We can therefore associate production decisions to nodes of the tree.

Let us introduce the following decision variables:

- x^n: quantity produced at node $n \in \mathcal{V}$,
- $y^n = 1$ if a setup for production is carried out at node $n \in \mathcal{V}$, $y^n = 0$ otherwise,
- s^n: inventory level at node $n \in \mathcal{V}$,

By replacing in (2.52) each expectation term $\mathbb{E}_{\tilde{d}_{[t+1,T]}|d_{[1,t]}}[\cdot]$ by the expression $\sum_{m\in\mathcal{C}(n)} \pi^{n,m}(\cdot)$, with n the node in the scenario tree corresponding to the partial realization $d_{[1,t]}$, and by unnesting the different terms, we obtain the following mixed-integer linear programming formulation.

$$\begin{cases} \min \sum_{n\in\mathcal{V}} \rho^n (f^n y^n + h^n s^n + g^n x^n) & & (2.53) \\ x^n \leq M^n y^n & \forall n \in \mathcal{V} & (2.54) \\ s^n + d^n = x^n + s^{a^n} & \forall n \in \mathcal{V} & (2.55) \\ x^n, s^n \geq 0, y^n \in \{0,1\} & \forall n \in \mathcal{V} & (2.56) \end{cases}$$

The objective function (2.53) aims at minimizing the expected total setup, inventory holding and production costs over all nodes of the scenario tree. Constraints (2.54) link the production quantity variables to the setup variables. Note that the value of constant M^n can be set by using an upper bound on the quantity to be processed at node n, usually defined as the maximum future demand as seen from node n, i.e. $M^n = \max_{\ell\in\mathcal{L}(n)} d^{n\ell}$, where $d^{n\ell} = \sum_{m\in\mathcal{P}(n,\ell)} d^m$. Constraints (2.55) are the inventory balance constraints. Constraints (2.56) provide the decision variables domain.

Several works focus on the polyhedral study of Problem (2.53)-(2.56) in order to strengthen its linear relaxation and improve the computational efficiency of the branch-and-cut algorithms embedded in MILP solvers. Valid inequalities are discussed by Guan et al. (2006), di Summa and Wolsey (2008) and Guan et al. (2009) and extended formulations are proposed by Ahmed et al. (2003) and Zhao and Guan (2014). In particular, Guan et al. (2009) propose a general method for generating cutting planes for multi-stage stochastic integer programs based on combining valid inequalities previously known for the deterministic variant of the corresponding problem and apply it on Problem (2.53)-(2.56). Their numerical results show that a branch-and-cut algorithm based on these new inequalities is more effective at solving instances on medium-size scenarios than a stand-alone mathematical programming solver.

2.4 Conclusion

We provided in this chapter an introduction to the lot-sizing models investigated in this document. The remainder of Part I is organized as follows.

- Chapter 3 focuses on a deterministic multi-item single-echelon lot-sizing problem: the Discrete Lot-sizing and Scheduling Problem or DLSP introduced in Subsection 2.2.2.
- Chapter 4 uses as a starting point the capacitated single-item lot-sizing problem (LS_C) described in Subsection 2.2.1 and studies an extension of this problem in which the demand is subject to uncertainty. A single-stage joint chance-constrained programming approach similar to the one discussed in Subsection 2.3.1 is investigated for this problem.

- Chapter 5 focuses on an extension of the uncapacitated single-item lot-sizing problem (ULS) described in Subsection 2.2.1 in which all input parameters (demand and cost) are subject to uncertainty. It investigates a multi-stage stochastic programming approach using scenario trees such as the one described in Subsection 2.3.3.
- Finally, Chapter 6 considers a multi-item multi-echelon remanufacturing system similar to the one presented in Subsection 2.2.3. It deals with planning the production for this system under demand, returns and cost uncertainty and is also based on a multi-stage stochastic programming approach.

Chapter 3

Discrete lot-sizing and scheduling with sequence-dependent setups

3.1 Introduction

This chapter presents the work carried out between 2011 and 2014 within the project LotRelax funded by the ANR through its program for young researchers. This project dealt with one of the lot-sizing problems I studied during my PhD, namely the multi-item DLSP with sequence-dependent start-up costs or DLSPSD, and aimed at proposing exact solution approaches based on strong relaxations for this problem.

The DLSPSD is an extension of the DLSP presented in Subsection 2.2.2. Similar to the DLSP, the DLSPSD considers a set of items which must be produced on a single capacitated production resource over a finite time horizon subdivided into discrete periods. It also assumes that at most one item can be produced on the resource in each period and uses a discrete or all-or-nothing production policy. However, the DLSPSD differs from the DLSP with respect to the start-up cost modeling. Namely, the DLSP uses sequence-independent start-up costs, i.e. start-up costs f_i whose value depends only on the next item (i) to be produced. In contrast, the DLSPSD considers sequence-dependent start-up costs, i.e. start-up costs sd_{ij} whose value depends on the production sequence, i.e depends on both the item i processed before the start-up and the item j processed after the start-up.

The objective of project LotRelax was to develop an exact solution approach for this NP-hard combinatorial optimization problem. A large amount of existing solution techniques in this area consists in formulating the problem as a mixed-integer linear program (MILP) and in relying on a branch-and-bound type procedure to solve the obtained MILP. However, the efficiency of such a procedure strongly depends on the quality of the lower bounds used to evaluate the nodes of the search tree. Much research has been devoted to the polyhedral study of lot-sizing problems in order to obtain tight linear relaxations and improve the corresponding lower bounds: see e.g. (Pochet and Wolsey, 2006) for a general overview of the related literature and (Belvaux and Wolsey, 2001; Eppen and Martin, 1987; van Eijl and van Hoesel, 1997) for contributions focusing specifically on the single-resource DLSP. During my PhD, I proposed to exploit the fact that items can often be described using a combination of physical attributes to reduce the size of the MILP formulation and ease the resolution of the DLSPSD by a mixed-integer linear programming solver. However, the corresponding numerical results provided by Gicquel et al. (2009) showed that, even when strengthening the reduced MILP formulation with the valid inequalities proposed by van Eijl and van Hoesel (1997), it was not possible to obtain optimal solutions for large-size instances of the problem within a reasonable computation time.

This difficulty can be partly explained by the fact that the valid inequalities proposed by van Eijl and van Hoesel (1997), as nearly all valid inequalities available for lot-sizing problems, focus on strengthening the formulation of the single-item sub-problems embedded in the formulation of the DLSPSD and thus fail at capturing the conflicts between multiple items sharing

the same resource capacity. During project LotRelax, we thus studied two ways which could enable us to better take into account the multi-item aspect of the DLSPSD.

A first part of the work carried out during the project was devoted to the study of semi-definite programing based relaxations for the DLSPSD. To achieve this, we reformulated the DLSPSD as a quadratically constrained quadratic binary program. We then used techniques developed for generic quadratic binary programs (Roupin, 2004) to efficiently reformulate the problem as a semi-definite program and proposed a cutting-plane generation algorithm to strengthen the initial convex relaxation of this semi-definite reformulation. Our numerical results showed that the semi-definite relaxation consistently provides lower bounds of significantly improved quality as compared with the ones provided by the continuous relaxation of the tightest MILP formulations known for the DLSPSD. In particular, the gap between the semi-definite relaxation and the optimal integer solution value can be closed for a significant proportion of the small-size instances, thus avoiding to resort to a tree search procedure. This improvement in the quality of the lower bounds is mainly explained by the fact that reformulating the DLSPSD as a semi-definite program amounts to lifting the problem into a much higher dimensional, as would be done by an RLT-1 extended reformulation obtained through the reformulation-linearization technique of Adams and Sherali (1990). This allows to capture among others relationships between variables relative to different items and different periods. However, solving a semi-definite program is much more computationally demanding than solving a linear program. As a consequence, the computation time needed to compute these very tight lower bounds was too high to consider using them in a branch-and-bound search procedure. Thus, solving practical lot-sizing problems using semi-definite relaxations instead of linear programming relaxations does not seem a possible option for the time being. This work was published in a journal paper: see Gicquel et al. (2014).

A second part of the work carried out during the project thus focused on improving the linear programming relaxation of the DLSPSD through a new family of multi-item multi-period inequalities. These inequalities seek to better represent conflicts on multi-period time intervals between several items simultaneously requiring production on the available resource. To the best of our knowledge, this was one of the first attempts at proposing multi-item valid inequalities for discrete lot-sizing problems. The corresponding separation problem could be formulated as a quadratic binary program and we proposed to solve it either exactly by relying on a quadratic programming solver or approximately through a variable depth search heuristic algorithm. The results of our computational results showed that the proposed inequalities are efficient at strengthening the linear relaxation of the problem and at decreasing the overall computation time needed to obtain guaranteed optimal solutions of the DLSPSD. This work was published in a journal paper: see Gicquel and Minoux (2015).

In what follows, we thus present in more detail this work on multi-item valid inequalities for the DSLPSD. The remainder of the chapter is organized as follows. In Section 3.2, we recall the initial MILP formulation of the multi-item DSLPSD as well as the previously published inequalities for the underlying single-item sub-problems. We then present in Section 3.3 the proposed multi-item inequalities and briefly discuss in Section 3.4 the resolution of the corresponding separation problem. A summary of our computational results is provided in Section 3.5.

3.2 Mathematical formulation

In this section, we first recall the initial MILP formulation of the DLSPSD. We use the network flow representation of change-overs between items, which was discussed among others by Belvaux and Wolsey (2001), as this leads to a tighter linear relaxation of the problem. We then

present the inequalities proposed by van Eijl and van Hoesel (1997) to strengthen the underlying single-item sub-problems.

3.2.1 Initial mixed-integer linear programming formulation

We wish to plan production for a set $\mathcal{I} = \{0, ..., I\}$ of items to be processed on a single production machine over a planning horizon involving a set $\mathcal{T} = \{1, ...T\}$ of periods. Item $i = 0$ represents the idle state of the machine and period $t = 0$ is used to describe the initial state of the production system.

Production capacity is assumed to be constant throughout the planning horizon. We can thus w.l.o.g. normalize the production capacity to one unit per period and apply a pretreatment on the original demand matrix resulting in a demand matrix containing only binary numbers: see e.g. (Belvaux and Wolsey, 2001). We denote d_{it} the demand for item i in period t: $d_{it} = 1$ in case there is a demand for item i in period t corresponding to producing i at full capacity in a period, $d_{it} = 0$ otherwise. Furthermore, we denote h_i the inventory holding cost per unit per period for item i and sd_{ij} the sequence-dependent start-up cost to be incurred whenever the production state of the resource is changed from item i to item j.

Using this notation, the DLSPSD can be seen as the problem of assigning at most one item to each period of the planning horizon while ensuring demand satisfaction and minimizing both inventory holding and start-up costs. We thus introduce the following binary decision variables:

- y_{it} where $y_{it} = 1$ if item i is assigned to period t, 0 otherwise.
- w_{ijt} where $w_{ijt} = 1$ if there is a change-over from item i to item j at the beginning of t, 0 otherwise.

This leads to the following MILP formulation denoted by DLSPSD0.

$$
\begin{cases}
Z^* = min \sum_{i=1}^{I} \sum_{t=1}^{T} h_i \sum_{\tau=1}^{t} (y_{i\tau} - d_{i\tau}) + \sum_{i,j=0}^{I} sd_{i,j} \sum_{t=1}^{T-1} w_{i,j,t} & & (3.1) \\
\sum_{\tau=1}^{t} y_{i\tau} \geq \sum_{\tau=1}^{t} d_{i\tau} & \forall i \in \mathcal{I}, \forall t \in \mathcal{T} & (3.2) \\
\sum_{i=0}^{I} y_{it} = 1 & \forall t \in \mathcal{T} & (3.3) \\
y_{i,t} = \sum_{j=0}^{I} w_{j,i,t} & \forall i \in \mathcal{I}, \forall t \in \mathcal{T} & (3.4) \\
y_{i,t} = \sum_{j=0}^{I} w_{i,j,t+1} & \forall i \in \mathcal{I}, \forall t \in \mathcal{T} & (3.5) \\
y_{it} \in \{0,1\} & \forall i \in \mathcal{I}, \forall t \in \mathcal{T} & (3.6) \\
w_{i,j,t} \in \{0,1\} & \forall i \in \mathcal{I}, \forall j \in \mathcal{I}, \forall t \in \mathcal{T} & (3.7)
\end{cases}
$$

The objective function (3.1) corresponds to the minimization of the inventory holding and start-up costs over the planning horizon. $\sum_{\tau=1}^{t}(y_{i\tau} - d_{i\tau})$ is the inventory level s_{it} of item i at the end of period t. Constraints (3.2) impose that the cumulative demand over interval $[1, t]$ is satisfied by the cumulative production over the same time interval. Constraints (3.3) ensure that, in each period, the resource is either producing a single item or idle. Constraints (3.4)-(3.5) link setup variables y_{it} with start-up variables w_{ijt} through equalities which can be seen as flow conservation constraints in a network: see e.g. Belvaux and Wolsey (2001). They ensure that in

case item i is produced in period t, there is a change-over from another item j (possible $j = i$) to item i at the beginning of period t and a change-over from item i to another item j (possible $i = j$) at the end of period t.

3.2.2 Single-item valid inequalities

We now recall the expression of the inequalities proposed by van Eijl and van Hoesel (1997) for the single-item DLSP. We denote $dc_{i,t,\tau}$ the cumulative demand for item i in the interval $\{t, ..., \tau\}$ and $\delta_{i,v}$ the v^{th} positive demand period for item i. $\delta_{i,dc_{i,1,t}+v}$ is thus the period in which the v^{th} positive unit demand for item i after period t occurs.

$$\sum_{\tau=1}^{t}(y_{i,\tau} - d_{i,\tau}) \geq u - \sum_{v=1}^{u}\left[y_{i,t+v} + \sum_{\tau=t+v+1}^{\delta_{i,dc_{i,1,t}+v}}\sum_{j\neq i} w_{j,i,\tau}\right] \qquad \forall i, \forall t, \forall u \in [1, dc_{i,t+1,T}] \tag{3.8}$$

The idea underlying constraints (3.8) is to compute a lower bound on the inventory level of a item i at the end of a period t, $\sum_{\tau=1}^{t}(y_{i,\tau} - d_{i,\tau})$, by considering each future unit demand $u = 1, ..., dc_{i,t+1,T}$ and the future states of the production resource for this item in the periods between $t+1$ and $\delta_{i,dc_{i,1,t}+u}$ at which this future unit demand occurs. The reader is referred to van Eijl and van Hoesel (1997) for a full proof of validity for these inequalities. In the computation experiments to be presented in Section 3.5, we use a standard cutting-plane generation algorithm to strengthen the formulation DLSPSD0 by adding violated inequalities of family (3.8). Since in practice the number of such inequalities is limited, the separation in this case is efficiently performed by enumeration. The resulting improved formulation is denoted by DLSPSD1.

Constraints (3.8) can be understood as a way to strengthen the demand satisfaction constraints (3.2) by expressing in a more detailed way the need for each individual item to access the resource in order to satisfy its own demand on a given sub-interval of the planning horizon. However, in the resulting DLSPSD1 formulation, the conflicts between different items simultaneously requiring production on the resource will only be handled by the single-period capacity constraints (3.3). In what follows, we propose to improve this representation of the conflicts between items by considering multi-period multi-item inequalities.

3.3 Multi-item valid inequalities

We now present the multi-period multi-item valid inequalities proposed to strengthen the linear relaxation of the DLSPSD.

3.3.1 General expression

As stated above, the proposed inequalities aim at improving the representation of the interactions between the items competing for the scarce capacity of the production resource. In formulation DSLPSD1, these interactions are only managed on a period by period basis via the capacity constraints (3.3). On the contrary, the proposed inequalities consider this competition on a multi-period time interval denoted by $[1, \theta]$ in what follows. More precisely, the competition is expressed as an opposition between two disjoint subsets of items denoted by $\mathcal{I}^P$ and $\mathcal{I}^D$. The item set $\mathcal{I}^P$ corresponds to items for which production in period $t \in [1, \theta]$ is considered and that may consequently take up the resource capacity in this period. If this is to happen, the items of the other set, $\mathcal{I}^D$, will not have access to the resource in period t and will loose the corresponding production capacity. The purpose of inequalities (3.9) below is to manage

this opposition by stating that, if one of the items in $\mathcal{I}^P$ is assigned for production in t, then we should make sure that the cumulative demand over interval $[1,\theta]$ for the items in the other set $\mathcal{I}^D$ will fit in the remaining production capacity. The right hand side of inequalities (3.9) computes a tight upper bound $\sum_{\tau=1}^{\theta} \bar{c}_\tau$ of this production capacity: $\bar{c}_\tau$ represents an upper bound of c_τ, the capacity available in period $\tau \in [1,\theta]$ for the items in $\mathcal{I}^D$ in case period t is devoted to the production of one of the items in $\mathcal{I}^P$.

Proposition 1.
Let $\mathcal{I}^P \subset \mathcal{I}$ and $\mathcal{I}^D \subset \mathcal{I}$ be two disjoint subsets of items.
Let $t \in [1,T]$ and consider $[1,\theta] \subset [1,T]$ a time interval including period t.
For each period $\tau \in [1,\theta]$, we denote $\mathcal{I}^D_\tau = \{j \in \mathcal{I}^D | \delta_{j,dc_{j,1,\theta}} \geq \tau\}$.
The following inequality is valid for the multi-item DLSPSD.

$$\Big[\sum_{j\in\mathcal{I}^D} dc_{j,1,\theta}\Big]\Big[\sum_{i\in\mathcal{I}^P} y_{it}\Big] \leq \sum_{\tau=1}^{\theta} \bar{c}_\tau \tag{3.9}$$

where $\bar{c}_\tau$ is defined by:

$$\bar{c}_\tau = min\Big(\sum_{j\in\mathcal{I}^D_\tau} y_{j,\tau}, \sum_{i\in\mathcal{I}^P} y_{i,t}\Big) \text{ if } \tau \in [1;t-2]\cup[t+2,\theta] \tag{3.9a}$$

$$\bar{c}_{t-1} = \sum_{j\in\mathcal{I}^D_{t-1}, i\in\mathcal{I}^P} w_{j,i,t} \tag{3.9b}$$

$$\bar{c}_t = 0 \tag{3.9c}$$

$$\bar{c}_{t+1} = \sum_{i\in\mathcal{I}^P, j\in\mathcal{I}^D_{t+1}} w_{i,j,t+1} \tag{3.9d}$$

Proof. Let (y,w) be a feasible solution of the DLSPSD. We arbitrarily choose a period t, an interval $[1,\theta]$ including t and two disjoint subsets of items $\mathcal{I}^P$ and $\mathcal{I}^D$ and show that the proposed inequality (3.9) is valid for the considered feasible solution.

We distinguish two main cases:
- Case 1: $\sum_{i\in\mathcal{I}^P} y_{it} = 0$

In this case, the left hand side of the inequality is equal to 0 whereas the right hand side is nonnegative. Inequality (3.9) is thus trivially valid.
- Case 2: $\sum_{i\in\mathcal{I}^P} y_{it} = 1$

In this case, the left hand side of inequality (3.9) is equal to the total cumulative demand over interval $[1,\theta]$ for the items belonging to $\mathcal{I}^D$, i.e. to $\sum_{j\in\mathcal{I}^D} dc_{j,1,\theta}$.

$\sum_{i\in\mathcal{I}^P} y_{it} = 1$ means that period t is devoted to the production of one of the items in $\mathcal{I}^P$ and thus cannot be used to satisfy the cumulative demand for items in $\mathcal{I}^D$ as these two item subsets are disjoint. Hence (y,w) can be a feasible solution of the DLSPSD if and only if the total cumulative production for items in $\mathcal{I}^D$ over the remaining periods $1...t-1, t+1...\theta$ is sufficient to satisfy the cumulative demand $\sum_{j\in\mathcal{I}^D} dc_{j,1,\theta}$.

We now seek to compute a tight upper bound $\bar{c}_\tau$ for the production capacity c_τ available in each period $\tau \in [1,t-1]\cup[t+1,\theta]$ for the items in $\mathcal{I}^D$:
- By capacity constraints (3.3), we have $c_\tau \leq 1$, i.e. $c_\tau \leq \sum_{i\in\mathcal{I}^P} y_{it}$.
- Moreover, the cumulative demand $dc_{j,1,\theta}$ for a item $j \in \mathcal{I}^D$ can only be satisfied by a production for this item in period τ if τ is within the interval $[1,\delta_{j,dc_{j,1,\theta}}]$, i.e. if there is at least of unit of demand belonging to $dc_{j,1,\theta}$ occurring after period τ. Thus, τ can be used to satisfy part of demand $\sum_{j\in\mathcal{I}^D} dc_{j,1,\theta}$ only if the resource is setup in τ for one of the items such that $\delta_{j,dc_{j,1,\theta}} \geq \tau$. This gives $c_\tau \leq \sum_{j\in\mathcal{I}^D_\tau} y_{j,\tau}$.

We thus obtain $c_\tau \leq min(\sum_{j\in\mathcal{I}^D_\tau} y_{j,\tau}, \sum_{i\in\mathcal{I}^P} y_{it}), \forall \tau \in [1,t-1]\cup[t+1,\theta]$. It leads to the following inequality stating that, in a feasible solution (y,w) of the DSLSPSD, in case period

t is devoted to the production of one item belonging to $\mathcal{I}^P$, the cumulative capacity available for items in $\mathcal{I}^D$ over periods $[1, t-1] \cup [t+1, \theta]$ should be large enough to produce the corresponding cumulative demand:

$$\Big[\sum_{j\in\mathcal{I}^D} dc_{j,1,\theta}\Big]\Big[\sum_{i\in\mathcal{I}^P} y_{it}\Big] \leq \sum_{\substack{\tau=1\ldots t-1\\ t+1\ldots\theta}} \Big[min(\sum_{j\in\mathcal{I}^D_\tau} y_{j,\tau}, \sum_{i\in\mathcal{I}^P} y_{it})\Big] \tag{3.10}$$

Now, we can exploit our knowledge of the production state of the resource in period t to further strengthen this inequality. Namely, we know that an item i belonging to $\mathcal{I}^P$ is produced in period t. A change-over to (resp. from) this item i thus has to take place at the beginning (resp. at the end) of period t. This means that:
- If period $t-1$ is to be used to satisfy the demand of one of the items belonging to $\mathcal{I}^D_{t-1}$, there must be a change-over from this item $j \in \mathcal{I}^D_{t-1}$ to the item $i \in \mathcal{I}^P$ at the beginning of period t. The production capacity available in period $\tau = t-1$ for the items in $\mathcal{I}^D_{t-1}$ is thus limited by $c_{t-1} \leq \sum_{i\in\mathcal{I}^P, j\in\mathcal{I}^D_{t-1}} w_{j,i,t}$.
- Similarly, if period $t+1$ is to be used to satisfy the demand of one of the items belonging to $\mathcal{I}^D_{t+1}$, there must be a change-over from the item $i \in \mathcal{I}^P$ to this item at the end of period t. The production capacity available in period $\tau = t+1$ for the items in $\mathcal{I}^D_{t+1}$ is thus limited by $c_{t+1} \leq \sum_{i\in\mathcal{I}^P, j\in\mathcal{I}^D_{t+1}} w_{i,j,t+1}$.

We can thus strengthen the upper bound of c_{t-1} (resp c_{t+1}) by replacing the term $min(\sum_{j\in\mathcal{I}^D_\tau} y_{j,\tau}, \sum_{i\in\mathcal{I}^P} y_{it})$ by $\sum_{i\in\mathcal{I}^P, j\in\mathcal{I}^D_{t-1}} w_{j,i,t}$ (resp. $\sum_{i\in\mathcal{I}^P, j\in\mathcal{I}^D_{t+1}} w_{i,j,t+1}$) and obtain the inequality (3.9) discussed in Proposition 1.

This completes the proof.

□

We point out here that, for any integer feasible solution of the DLSPSD, in case $\sum_{p\in\mathcal{I}^P} y_{pt} = 1$, we have:

$$\sum_{j\in\mathcal{I}^D_\tau} y_{j,\tau} \leq \sum_{i\in\mathcal{I}^P} y_{it}, \qquad \forall\tau \in [1, t-1] \cup [t+1, \theta] \tag{3.11}$$

$$\sum_{\substack{i\in\mathcal{I}^P\\ j\in\mathcal{I}^D_{t-1}}} w_{j,i,t} = \sum_{j\in\mathcal{I}^D_{t-1}} y_{j,t-1} \qquad \text{if } t \neq 1 \tag{3.12}$$

$$\sum_{\substack{i\in\mathcal{I}^P\\ j\in\mathcal{I}^D_{t+1}}} w_{i,j,t+1} = \sum_{j\in\mathcal{I}^D_{t+1}} y_{j,t+1} \qquad \text{if } t \neq \theta \tag{3.13}$$

We will thus have $c_\tau = \sum_{j\in\mathcal{I}^D_\tau} y_{j,\tau}, \forall\tau \in [1, t-1] \cup [t+1, \theta]$ in any integer feasible solution of the problem. However, in a fractional solution obtained by solving the linear relaxation of formulation DLSPSD1, we may encounter situations where $0 < \sum_{i\in\mathcal{I}^P} y_{it} < 1$ so that we may have $\sum_{i\in\mathcal{I}^P} y_{it} \leq \sum_{j\in\mathcal{I}^D} y_{j,\tau}$, $\sum_{\substack{i\in\mathcal{I}^P\\ j\in\mathcal{I}^D}} w_{j,i,t} \leq \sum_{j\in\mathcal{I}^D} y_{j,t-1}$ and $\sum_{\substack{i\in\mathcal{I}^P\\ j\in\mathcal{I}^D}} w_{i,j,t+1} \leq \sum_{j\in\mathcal{I}^D} y_{j,t+1}$. In these cases, it is interesting to have the flexibility to select for each period τ the smallest upper bound for the available production capacity c_τ as this will lead to tighter inequalities.

3.3.2 Illustrative example

We introduce a small instance of the DLSPSD in order to illustrate how the proposed multi-item inequalities may help at strengthening the MILP formulation of the problem. This instance involves $I = 4$ items and $T = 10$ periods. Table 3.1 gives the numerical data on the inventory holding costs, on the start-up costs and on the demand for this instance. We note that the start-up cost matrix displays a frequently encountered feature: the presence of two item families

(items $\{1,2\}$ and items $\{3,4\}$). The start-up costs between items belonging to different families are significantly higher than the ones between items belonging to the same family.

We provide in Table 3.2 the fractional solution obtained by solving the linear relaxation of formulation DLSPSD1, i.e. the initial formulation of the problem strengthened only by the single-item inequalities (3.8). In this solution, periods 1 and 2 are assigned to the production of a single item, which complies with the problem constraints. However, in periods 3 to 10, the resource capacity is shared between several items, which means that this solution is not feasible for the integer optimization problem. The corresponding cost, $Z = 563.25$, is thus a lower bound for the optimal integer solution value.

The fractional solution provided in Table 3.2 violates several inequalities belonging to the family described in Proposition 1.

One of them corresponds to period $t = 6$, interval $[1,\theta] = [1,7]$ and item sets $\mathcal{I}^P = \{2\}$ and $\mathcal{I}^D = \{3,4\}$. Namely, we have:

- $LHS = (dc_{3,1,7} + dc_{4,1,7})y_{2,6} = 2 * 0.5 = 1$
- $RHS = \sum_{\tau=1}^{7} \bar{c}_\tau = 0.75$ as:

$$
\begin{aligned}
-\bar{c}_1 &= min(y_{3,1} + y_{4,1}, y_{2,6}) = y_{3,1} + y_{4,1} = 0 && \text{see (3.9}a\text{)}\\
-\bar{c}_2 &= min(y_{3,2} + y_{4,2}, y_{2,6}) = y_{3,2} + y_{4,2} = 0 && \text{see (3.9}a\text{)}\\
-\bar{c}_3 &= min(y_{3,3} + y_{4,3}, y_{2,6}) = y_{3,3} + y_{4,3} = 0.25 && \text{see (3.9}a\text{)}\\
-\bar{c}_4 &= min(y_{3,4} + y_{4,4}, y_{2,6}) = y_{2,6} = 0.5 && \text{see (3.9}a\text{)}\\
-\bar{c}_5 &= w_{3,2,6} + w_{4,2,6} = 0 && \text{see (3.9}b\text{)}\\
-\bar{c}_6 &= 0 && \text{see (3.9}c\text{)}\\
-\bar{c}_7 &= w_{2,3,7} = 0 \text{ (Note that } \mathcal{I}_7^D = \{3\}) && \text{see (3.9}d\text{)}
\end{aligned}
$$

We can thus improve the problem formulation by adding the following cut:

$$(dc_{3,1,7} + dc_{4,1,7})y_{2,6} \leq y_{3,1} + y_{4,1} + y_{3,2} + y_{4,2} + y_{3,3} + y_{4,3} + y_{2,6} + w_{3,2,6} + w_{4,2,6} + w_{2,3,7}$$

The idea underlying this inequality is the following. We choose the subset $\mathcal{I}^P = \{2\}$. If item 2 is not assigned for production in period 6 (i.e. $y_{2,6} = 0$), the inequality is trivially respected. But if item 2 is assigned for production in period 6 (i.e. $y_{2,6} = 1$), then we have to make sure that we are able to satisfy the total cumulative demand over the interval $[1,7]$ for the items in subset $\mathcal{I}^D = \{3,4\}$ (i.e to satisfy $dc_{3,1,7} + dc_{4,1,7}$) on the remaining periods $1,2,..,5,7$. In this case, the right hand side of inequalities (3.9) computes a tight upper bound of the production capacity available over these periods for items 3 and 4.

Three additional multi-item inequalities are violated by the fractional solution, namely those corresponding to:

- $t = 4$, $[1,\theta] = [1,5]$, $\mathcal{I}^P = \{1\}$ and $\mathcal{I}^D = \{4\}$
- $t = 7$, $[1,\theta] = [1,10]$, $\mathcal{I}^P = \{3\}$ and $\mathcal{I}^D = \{2\}$
- $t = 9$, $[1,\theta] = [1,10]$, $\mathcal{I}^P = \{2\}$ and $\mathcal{I}^D = \{1,3,4\}$

We add these four multi-item inequalities to the formulation DLSPSD1 and solve the linear relaxation of the resulting strengthened formulation. We obtain the integer feasible solution described in Table 3.3, the cost of which is $Z = 574$ and corresponds to the optimal integer solution value.

TABLE 3.1: Small illustrative example: numerical input data

		sd_{ij}					d_{it}									
		j					t									
i	h_i	0	1	2	3	4	1	2	3	4	5	6	7	8	9	10
0	0	0	191	156	130	161	0	0	0	0	0	0	0	0	0	0
1	7	152	0	14	122	173	1	0	1	0	1	0	1	0	0	0
2	10	125	13	0	119	157	0	0	0	0	0	0	0	0	1	1
3	6	156	157	109	0	6	0	0	0	0	0	0	1	0	0	1
4	7	116	132	134	19	0	0	0	0	0	1	0	0	0	0	1

TABLE 3.2: Small illustrative example: solution obtained by computing the continuous relaxation of formulation DLSPSD1

Value of the setup variables y_{it}

i \ t	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	1	1	0.75	0.25	0.5	0	0.5	0	0	0
2	0	0	0	0	0	0.5	0	0.5	0.75	0.25
3	0	0	0	0.5	0	0	0.5	0.25	0.25	0.5
4	0	0	0.25	0.25	0.5	0.5	0	0.25	0	0.25

TABLE 3.3: Small illustrative example: solution obtained by computing the continuous relaxation of formulation DLSPSD1 strengthened by 4 multi-item inequalities

Value of the setup variables y_{it}

i \ t	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	1	1	0	0
4	0	0	0	0	1	1	0	0	0	0

3.4 Exact and heuristic algorithms for solving the separation problem

The number of inequalities (3.9) grows very fast with the problem size so that it is not possible to include all of them *a priori* in the MILP formulation. We thus investigate the resolution of the corresponding separation problem. Given a fractional solution $(\overline{y}, \overline{w})$ of Problem DLSPSD1, solving the separation problem for a pair of periods (t, θ) consists in finding an inequality (3.9), amongst the ones corresponding to periods (t, θ), which is violated by the solution $(\overline{y}, \overline{w})$ if such an inequality exists or proving that no such valid inequality exists.

To achieve this, we need to find the partition of the set of items $\mathcal{I}$ into 3 subsets (a subset $\mathcal{I}^P$, a subset $\mathcal{I}^D$ and a subset containing the rest of the items) providing the largest difference between the left and the right hand sides of inequalities (3.9). This tripartition problem can be formulated as a binary program involving a quadratic objective function and a series of linear constraints. Although this quadratic binary program can be solved to optimality using a quadratic binary programming solver such as the one embedded in CPLEX, the corresponding computation time is prohibitively long for a practical use. We thus develop a heuristic separation algorithm which relies on a variable depth search heuristic of Kernighan-Lin type: see (Kernighan and Lin, 1970). The reader is referred to (Gicquel and Minoux, 2015) for a detailed description of this algorithm.

Regarding the cutting-plane generation strategy, our preliminary computational experiments showed that when a violated inequality is found for a pair (t, θ), inequalities for pairs $(t, \theta + 1)$, $(t, \theta + 2)$... are most often also violated and the amount of the violation is identical to the one found for (t, θ). It is thus not computationally efficient to solve all separation subproblems (t, θ) for period t. In our cutting-plane generation algorithm, for a given period t, we thus stop the search for violated inequalities as soon as one has been found for a period $\theta \geq t$, as it appeared to be a better strategy. Thus, at most T inequalities are added at each iteration of the algorithm. In the computational experiments to be presented in Section 3.5, this cutting-plane generation algorithm is used at the root node of the branch-and-bound search tree and thus forms the basis of a cut-and-branch algorithm.

3.5 Computational experiments

We now discuss the results of the computational experiments carried out to evaluate the effectiveness of the proposed multi-item inequalities at strengthening the formulation of the multi-item DLSPSD and to assess their impact on the total computation time.

3.5.1 Instances

We randomly generated instances of the problem using a procedure similar to the one used by Salomon et al. (1997) for the DLSP with sequence-dependent start-up costs and times. The reader is referred to (Gicquel and Minoux, 2015) for a detailed description of this generation procedure. In what follows, we only recall the values used for the characteristics which seemed to have the strongest impact on the performance of the solution algorithms:

- *Problem dimension*. The problem dimension is represented by the number of items I and the number of periods T: we solved medium-size instances involving 4 to 12 items and 25 to 75 periods.
- *Start-up costs*. We used two different types of structure for the sequence-dependent start-up cost matrix sd. Instances of set A have a general cost structure: the cost of a change-over from item i to item j, sd_{ij}, was randomly generated from a discrete uniform

$DU(100, 200)$ distribution. Instances of set B correspond to the frequently encountered case where items can be grouped into item families: there is a high change-over cost between items of different families and a smaller change-over cost between items belonging to the same family. In this case, for items i and j belonging to different item families, sd_{ij} was randomly generated from a discrete uniform $DU(100, 200)$ distribution; for items i and j belonging to the same item family, sd_{ij} was randomly generated from a discrete uniform $DU(0, 100)$ distribution.

For each considered problem dimension and each possible structure of the start-up cost matrix, we generated 10 instances, leading to a total of 300 instances.

3.5.2 Results

All tests were run on an Intel Core i5 (2.7 GHz) with 4 Go of RAM, running under Windows 7. We used a standard MILP software (CPLEX 12.5) with the solver default settings to solve the problems with one of the following formulations:

- DLPSD1: initial MILP formulation DLSPSD0, i.e. formulation (3.1)-(3.7), strengthened by single-item inequalities (3.8). We used a standard cutting-plane generation strategy based on a complete enumeration of all possible inequalities to add them into the formulation.
- DLSPSD2e: formulation DLSPSD1 strengthened by multi-item inequalities (3.9). We used the cutting-plane generation algorithm with the *exact* separation algorithm discussed in Section 3.4.
- DLSPSD2h: formulation DLSPSD1 strengthened by multi-item inequalities (3.9). We used the cutting-plane generation algorithm with the *heuristic* separation algorithm discussed in Section 3.4.

For the sake of brevity, we only provide here a summary of the results obtained while using formulation DLSPSD1 and DLSPSD2h. Results obtained on small instances involving 10 to 20 periods with formulation DSLSSD2e are reported in Gicquel and Minoux (2015). In a nutshell, they show that inequalities (3.9) are efficient at strengthening the formulation of the DLSPSD but that the exact separation algorithm leads to prohibitively long computation times. Moreover, these results also show that using the heuristic separation algorithm to generate these inequalities provides a formulation strengthening of a similar quality that the one obtained with the exact separation algorithm.

Table 3.4 displays a summary of the computational results obtained while using formulations DLSPSD1 and DLSPSD2h. The instances are grouped according to the structure of the start-up cost matrix and to their number of periods T. Each set of 50 instances thus comprises 5 subsets of 10 instances, each subset corresponding to a value of the number of items I belonging to $\{4, 6, 8, 10, 12\}$. For each set of 50 instances, we provide:

- the structure of the start-up cost matrix: A corresponds to the general case, B to the case where items can be grouped into families.
- T: the number of planning periods involved in the production planning problem.
- $\#IV\_S$: the average number of single-item violated inequalities (3.8) added in the formulation.
- $\#IV\_M$: the average number of multi-item violated inequalities added in formulation DLSPSD2h by the heuristic separation algorithm.

TABLE 3.4: Computational results

			DLSPSD1			DLSPSD2h			
Start-up costs	T	$\#IV\_S$	Gap_1	N_1	T_1	$\#IV\_M$	Gap_{2h}	N_{2h}	T_{2h}
A	25	314	2.60%	9	1.4s	32	0.26%	1	0.7s
	50	1223	2.48%	96	17.3s	44	1.20%	35	14.2s
	75	2787	2.68%	725	146.1s	22	2.50%	658	137.2s
B	25	314	11.48%	171	3.4s	51	4.70%	44	2.1s
	50	1246	13.60%	8932	295.0s	76	9.94%	4724	191.7s
	75	2707	13.00%	19912	1561.3s	43	11.84%	17461	1483s

- Gap_1 (resp. Gap_{2h}): the average percentage gap between the linear relaxation of formulation DLSPSD1 (resp. DLSPSD2h) and the value of an optimal integer solution.
- N_1 (resp. N_{2h}): the average number of nodes explored by the branch-and-bound procedure before a guaranteed optimal integer solution is found or the computation time limit of 2700s is reached.
- T_1 (resp. T_{2h}): the average total computation time (cutting-plane generation and branch-and-bound search) needed to find a guaranteed optimal integer solution (we used the value of 2700s in case a guaranteed optimal integer solution could not be found within the computation time limit).

Results from Tables 3.4 show that the proposed approach is efficient at strengthening the problem formulation and at reducing the total computation time when the number of periods involved in the planning problem stays below 50. Namely, for the instances involving less than 50 periods, the average integrality gap is reduced from 7.6% to 4.1% while using the multi-item inequalities generated with the heuristic separation algorithm. This leads to a significant reduction of the average number of nodes explored by the branch-and-bound algorithm before a proven optimal solution is found (from 2307 to 1201) and consequently to a decrease in the overall average computation time (from 79.3s to 52.2s). We note in particular that the computation time is significantly decreased from 295.0s to 191.7s for the instances involving product families and 50 periods.

However, when the number of periods increases and reaches 75, the efficiency of the proposed approach at strengthening the MILP formulation and at reducing the computation time seems to decrease. This can be seen by the fact that, for the corresponding instances, the average number of violated inequalities generated by the cutting-plane generation algorithm decreases as compared to the number of inequalities generated for the instances involving a smaller number of periods. Consequently, the average integrality gap is only reduced from 7.8% to 7.1% and the average computation time is only decreased from 853.6s to 810.1s. This might indicate that the strength of the proposed multi-item inequalities decreases when the number of planning periods increases.

3.6 Conclusion and perspectives

This chapter was devoted to the work carried out on the multi-item discrete lot-sizing and scheduling problem with sequence-dependent start-up costs. In particular, a new family of multi-item valid inequalities for the problem has been exhibited and both an exact and a heuristic separation algorithms have been devised and computationally tested. Our results show that

the proposed inequalities, when used in a cut-and-branch algorithm, are efficient at strengthening the MILP formulation and at reducing the overall computation time needed to obtain guaranteed optimal solutions, at least for instances featuring a number of periods up to 50.

Among the possible research directions suggested by the present work, it might be worth exploring the development of a branch-and-cut algorithm in which the proposed valid inequalities would be added to the formulation not only at the root node of the search tree but at any nodes explored over the course of the algorithm. Furthermore, the extension of the proposed inequalities to other variants of discrete lot-sizing problems, in particular those involving multiple parallel resources or positive start-up times, could also be interesting.

Chapter 4

Joint chance-constrained lot-sizing

4.1 Introduction

The work presented in Chapter 3 dealt with a deterministic lot-sizing problem: we assumed that there is no uncertainty on the value of all the input parameters (demand and costs) needed to build the production plan. However, in practice, lot-sizing is about planning the activity of a resource for the near future and thus often relies on data which are obtained through forecasting procedures. Thus, in many applications, assuming deterministically known input data is not realistic. Examples of real-world lot-sizing problems with uncertain input parameters can be found among others in Camargo et al. (2014) for the spinning industry, Hu and Hu (2016) for a manufacturing company producing braking equipment, Ghamari and Sahebi (2017) for a chemical-petrochemical case study, Kilic et al. (2018) for a remanufacturing system, Macedo et al. (2016) for a hybrid manufacturing/remanufacturing system and Moreno et al. (2018) for humanitarian logistics.

After completion of the LotRelax project, I thus decided to focus my work on the study of stochastic variants of lot-sizing problems. This decision was partly influenced by the fact that, at that time, several of my colleagues at the LRI worked in the field of stochastic programming. In particular, I had the opportunity to work with Jiangqian Cheng who was a post-doc in my research team studying linear programs with joint chance constraints. As a starting point for our collaboration, we decided to investigate a joint chance-constrained lot-sizing problem. In view of the theoretical and numerical difficulties posed by chance-constrained programming, we chose to study a variant of lot-sizing problems simpler than the DLSPSD I used to study since my PhD and focused on the single-item single-resource capacitated lot-sizing problem with stochastic demand.

This chapter presents our joint work on this problem. As commonly done in stochastic programming, we assume that, even if the demand cannot be deterministically known, a description of the demand uncertainty is available in terms of a probability distribution. We propose to handle this problem through the use of a single-stage stochastic programming approach: we seek to build the production plan before any additional information on the demand realization becomes available and do not consider the possibility of updating it as the demand unfolds over time. We consider the case where the production plan should be feasible for nearly all possible outcomes of the demand. This leads to the formulation of a joint chance-constrained program where the probability that an inventory shortage occurs during the planning horizon is limited to a maximum acceptable risk level.

This problem was previously investigated by Beraldi and Ruszczyński (2002) and Küçükyavuz (2012). Beraldi and Ruszczyński (2002) assume that the demand in each period follows a discrete probability distribution and propose a solution method based on a partial enumeration of the p-efficient points of the joint distribution of the random vector representing the cumulative demand over the planning horizon. We recall that a point $v \in \mathbb{R}^n$ is a p-efficient point of the probability distribution F if $F(v) \geq p$ and there is no point $u \in \mathbb{R}^n, u \neq v$ such that

$u \leq v$ and $F(u) \geq p$. Küçükyavuz (2012) assumes that the demand in each period follows a finite discrete probability distribution and reformulates the joint chance-constrained problem as a mixed-integer linear program. She presents a new class of valid inequalities for this problem and shows that these inequalities are numerically efficient in strengthening the mixed-integer linear programming formulation and in reducing the computation time.

Both of the previously published approaches on this problem thus used as a starting point a discrete probability distribution whereas in practice, the error terms in forecasting models is often represented by a continuous probability distribution such as a normal distribution. Luedtke and Ahmed (2008) proposed a sample approximation approach for generic chance-constraint programs. This one relies on a Monte Carlo sampling of the continuous random variables to generate a set of discrete scenarios and leads to the formulation of a mixed-integer linear program in which a binary variable for each generated scenario must be introduced.

Our main contribution consists in the introduction of a new extension of this approach which we called the "partial sample approximation approach". Similarly to the sample approximation approach, the proposed method relies on a Monte Carlo sampling of the random variables $\tilde{d}_t, t \in \mathcal{T}$, representing the demand in periods 1 to T. However, this sampling is carried out on only part of the random variables, more precisely on all random variables except $\tilde{d}_1$. Provided there is no dependence between $\tilde{d}_1$ and the demand in the later periods, this partial sampling results in the formulation of a chance-constrained program featuring a series of joint chance constraints. Each of these constraints involves a single random variable and defines a feasible set for which a conservative convex approximation can be quite easily built. The main advantage of the proposed method is that, unlike the sample approximation approach, it only requires the introduction of additional continuous variables and thus leads to the formulation of a deterministic mixed-integer linear program (MILP) having the same number of binary variables as the initial stochastic problem. As will be shown by our computational experiments, the proposed method is more efficient at finding feasible solutions of the original stochastic problem than the sample approximation method and these solutions are less costly than the ones provided by the Bonferroni conservative approximation. Moreover, the computation time is shorter than the one needed for the sample approximation method.

The remainder of the chapter is organized as follows. We describe in Section 4.2 the joint chance-constrained programming formulation used to model the problem under study. We discuss two previously published solution approaches for this problem: the Bonferroni conservative approximation in Section 4.3 and the sample approximation approach in Section 4.4. Section 4.5 is devoted to the presentation of the proposed partial sample approximation approach. Computational results are provided in Section 4.6.

4.2 Mathematical formulations

We consider the single-item single-resource capacitated lot-sizing problem introduced in Subsection 2.2.1. We first briefly recall the MILP formulation of its deterministic variant. We then consider a stochastic variant of this problem in which the customer demand to be satisfied is subject to uncertainty and introduce the joint chance-constrained programming formulation studied in the present chapter.

4.2.1 Deterministic formulation

We wish to plan production for a single product to be processed on a single capacitated resource over a planning horizon involving a set $\mathcal{T} = \{1, ..., T\}$ of periods.

All problem parameters are assumed to be deterministically known at the time when the production plan is built. f_t denotes the fixed setup cost to be paid if production occurs on the

resource in period t, h_t the inventory holding cost per unit held in stock at the end of period t and c_t the production capacity available in period t. d_t is the demand to be satisfied at the end of each period t and $dc_{1t} = \sum_{\tau=1}^{t} d_\tau$ is the cumulative demand over interval $[1;t]$. The initial inventory level, s_0, is set to 0 without loss of generality.

We introduce the following decision variables:

- x_t: the quantity produced in period t.
- $y_t \in \{0,1\}$: the resource setup state in period t. $y_t = 1$ if a setup occurs in period t, 0 otherwise.

With this notation, the deterministic single-item single-resource capacitated lot-sizing problem or LS_C can be formulated as follows:

$$
\begin{cases}
Z^*_{DET} = \min \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T} h_t \big(\sum_{\tau=1}^{t} x_\tau - dc_{1t} \big) & & (4.1) \\
x_t \leq c_t y_t & \forall t \in \mathcal{T} & (4.2) \\
\sum_{\tau=1}^{t} x_\tau \geq dc_{1t} & \forall t \in \mathcal{T} & (4.3) \\
x_t \geq 0 & \forall t \in \mathcal{T}. & (4.4) \\
y_t \in \{0,1\} & \forall t \in \mathcal{T}. & (4.5)
\end{cases}
$$

The objective function (4.1) corresponds to the minimization of the setup and inventory holding costs over the planning horizon. Note that $\sum_{\tau=1}^{t} x_\tau - dc_{1t}$ computes the inventory level s_t at the end of period t as the difference between the cumulative production up to t and the cumulative demand up to t. Constraints (4.2) ensure that, if production takes place in period t, the corresponding setup costs are incurred and the capacity limit c_t is respected. Constraints (4.3) are the demand satisfaction constraints: they guarantee that the cumulative production over each interval $[1;t]$ is large enough to satisfy the cumulative demand over the same interval, and consequently that the inventory level at the end of period t, $\sum_{\tau=1}^{t} x_\tau - dc_{1t}$, is non-negative.

4.2.2 Stochastic formulation

We now consider the case where the customer demand to be satisfied in period t is not perfectly known at the time when the production plan is built. This might be due among others to the fact that the only available information on the future demand is based on forecasts (rather than on firm customer orders) and that forecast inaccuracies are unavoidable. We thus model the demand in period t as a random variable $\tilde{d}_t$, the probability distribution of which is assumed to be known. The cumulative demand over periods 1...t, denoted by $\widetilde{dc}_{1t} = \sum_{\tau=1}^{t} \tilde{d}_t$, is also a random variable. The deterministic parameter dc_{1t} is thus replaced by a random variable, $\widetilde{dc}_{1t}$, in the stochastic formulation of the problem.

This has several implications for the problem formulation. First, replacing dc_{1t} by its stochastic counterpart $\widetilde{dc}_{1t}$ implies that the inventory level at the end of period t is a random variable defined as $\tilde{s}_t = max(\sum_{\tau=1}^{t} x_\tau - \widetilde{dc}_{1t}, 0)$: see e.g. Beraldi and Ruszczyński (2002). Hence, the value of the inventory holding cost appearing in the objective function, $\sum_{t=1}^{T} h_t \tilde{s}_t$, is also a random variable. We therefore consider minimizing its expected value $\mathbb{E}\big[\sum_{t=1}^{T} h_t \tilde{s}_t\big]$ in the stochastic formulation. In the present work, we use the same approximation as the one used by Bookbinder and Tan (1988) and Beraldi and Ruszczyński (2002) to compute $\mathbb{E}\big[\sum_{t=1}^{T} h_t \tilde{s}_t\big]$, namely:

$$\mathbb{E}\Big[\sum_{t=1}^{T} h_t \tilde{s}_t\Big] = \mathbb{E}\Big[\sum_{t=1}^{T} h_t max(\sum_{\tau=1}^{t} x_\tau - \widetilde{dc}_{1t}, 0)\Big] \tag{4.6}$$

$$\approx \mathbb{E}\Big[\sum_{t=1}^{T} h_t(\sum_{\tau=1}^{t} x_\tau - \widetilde{dc}_{1t})\Big] \tag{4.7}$$

$$\approx \sum_{t=1}^{T} h_t\Big(\sum_{\tau=1}^{t} x_\tau - \mathbb{E}[\widetilde{dc}_{1t}]\Big) \tag{4.8}$$

Note that, with this approximation, our model does not compute the inventory holding cost exactly: it tends to underestimate it in the periods with a negative ending inventory.

Second, the fact that $\widetilde{dc}_{1t}$ is stochastic implies that it might be very costly (and even impossible depending on the support of the probability distributions) to build a production plan ensuring that the cumulative production is large enough to satisfy every possible realization of the demand. We thus have to consider the eventuality that the demand satisfaction constraints (4.3) will be violated for some demand realizations. As explained in Chapter 2, a possible way of handling this situation in the optimization problem consists in limiting the probability of these violations through the use of chance constraints. We thus define a maximum acceptable risk level ϵ and impose $1-\epsilon$ as a lower bound on the probability that there is no stockout over all periods of the planning horizon.

This leads to the following joint chance-constrained program denoted by SLS in what follows.

$$\begin{cases} Z^*_{SLS} = \min \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T} h_t\Big(\sum_{\tau=1}^{t} x_\tau - \mathbb{E}[\widetilde{dc}_{1t}]\Big) & & (4.9)\\ x_t \leq c_t y_t & \forall t \in \mathcal{T} & (4.10)\\ \Pr\Big(\sum_{\tau=1}^{t} x_\tau \geq \widetilde{dc}_{1t}, \forall t \in \mathcal{T}\Big) \geq 1-\epsilon & & (4.11)\\ x_t \geq 0 & \forall t \in \mathcal{T}. & (4.12)\\ y_t \in \{0,1\} & \forall t \in \mathcal{T}. & (4.13) \end{cases}$$

As explained e.g. by Luedtke and Ahmed (2008), mathematical programs involving a joint chance constraint such as (4.11) are still largely intractable except for a few exceptions. This can be explained by two main reasons. First, checking the feasibility of a given solution requires computing the value of the joint probability through multidimensional integration, which can be very time-consuming. Second, the feasible space defined by a joint chance-constraint is in general not convex. Yet, as mentioned e.g. by Nemirovski and Shapiro (2005), we need both efficient computation of the probability and convexity of the solution space to efficiently process chance-constraints. This is why a variety of tractable approximations have been proposed for chance-constrained problems. They rely either on conservative convex approximations (Bonferroni, 1936; Rockafellar and Uryasev, 2000; Nemirovski and Shapiro, 2006) or on a discretization of the probability distribution through sampling (Calafiore and Campi, 2005; Luedtke and Ahmed, 2008).

In Sections 4.3 to 4.5, we first describe the Bonferroni approximation which provides guaranteed feasible solutions of SLS and requires a limited computational effort but may lead to overly conservative and expensive solutions. We then discuss the sample approximation approach developed by Luedtke and Ahmed (2008) before presenting the extension proposed in this work, which we refer to as the "partial sample approximation" approach.

4.3 Bonferroni conservative approximation

Let E_t be the random event that there is no stockout at the end of period t, i.e. that $\sum_{\tau=1}^{t} x_\tau \geq \widetilde{dc}_{1t}$ and $\Pr(E_t)$ be the probability that this event occurs. The Bonferroni inequality states that:

$$\Pr(\cap_{t=1}^{T} E_t) \geq 1 - \sum_{t=1}^{T} [1 - \Pr(E_t)] \tag{4.14}$$

By replacing in (4.11) the joint probability by its lower bound and adjusting the reliability level appropriately, we can divide the joint chance constraint into T individual chance constraints: $\Pr\left(\sum_{\tau=1}^{t} x_\tau \geq \widetilde{dc}_{1t}\right) \geq 1 - \frac{\epsilon}{T}$. This leads to the following deterministic mixed-integer linear program denoted by BON:

$$\begin{cases} Z^*_{BON} = \min \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T} h_t\Big(\sum_{\tau=1}^{t} x_\tau - \mathbb{E}[\widetilde{dc}_{1t}]\Big) & (4.15) \\ x_t \leq c_t y_t \qquad \forall t \in \mathcal{T} & (4.16) \\ \sum_{\tau=1}^{t} x_\tau \geq F_{1t}^{-1}\Big(1 - \frac{\epsilon}{T}\Big) \qquad \forall t \in \mathcal{T} & (4.17) \\ x_t \geq 0 \qquad \forall t \in \mathcal{T}. & (4.18) \\ y_t \in \{0,1\} \qquad \forall t \in \mathcal{T}. & (4.19) \end{cases}$$

F_{1t} is the cumulative probability distribution of the random variable $\widetilde{dc}_{1t}$. The value of $F_{1t}^{-1}\left(1 - \frac{\epsilon}{T}\right)$ can be computed in a pre-optimization step either exactly or approximately depending of the probability distribution of $\tilde{d}_t$. In our numerical experiments, we consider the case in which $\tilde{d}_1, ..., \tilde{d}_T$ are independent random variables, each one following a normal distribution $\mathcal{N}(d_t, \chi_t)$ and use the fact that in this case, $\widetilde{dc}_{1t}$ follows a normal distribution $\mathcal{N}(\sum_{\tau=1}^{t} d_\tau, \sqrt{\sum_{\tau=1}^{t} \chi_\tau^2})$.

Problem (4.15)-(4.19) is a deterministic problem similar to (4.1)-(4.5) which can be solved with a limited computational effort. It provides guaranteed feasible solutions of SLS. However, as will be shown by the computational results provided in Section 4.6, most often, these solutions are overly conservative and expensive.

4.4 Sample approximation approach

The sample approximation approach proposed by Luedtke and Ahmed (2008) aims at providing approximate solutions for joint chance-constrained programs. The main idea consists in replacing the original continuous probability distribution of the random vector (the cumulative demand vector $\widetilde{dc}$ in our case) by an empirical discrete finite probability distribution obtained by Monte Carlo sampling. When randomness appears only on the right-hand side of the constraints (which is the case in problem SLS), this leads to the formulation of a large-size mixed-integer linear program, which can be handled by mixed-integer linear programming techniques.

In this section, we present how applying the sample approximation approach to problem SLS leads to the formulation of such a large-size MILP.

Let d^1,.., d^k,..., d^K be a Monte Carlo sample of the random vector $\tilde{d}$ and $dc_{1t}^k = \sum_{\tau=1}^{t} d_\tau^k$ be the cumulated demand over $[1, t]$ in scenario k. The K sampled scenarios are independent

and identically distributed observations of vector $\tilde{d}$ so that the probability of each scenario k is considered to be equal to $1/K$.

The sample approximation approach relies on the idea that, given a production plan x, the value of the joint probability involved in constraint (4.11) can be approximately computed as follows:

$$\Pr\Big(\sum_{\tau=1}^{t} x_\tau \geq \tilde{dc}_{1t}, \forall t \in \mathcal{T}\Big) \approx \frac{1}{K}\sum_{k=1}^{K} \mathbb{I}\Big(\sum_{\tau=1}^{t} x_\tau - dc_{1t}^k \geq 0 \quad \forall t\Big) \tag{4.20}$$

where $\mathbb{I}(.)$ denotes the indicator function taking the value 1 when . is true and 0 otherwise.

The idea underlying approximation (4.20) is the following. We check, for each scenario k, whether all demand satisfaction constraints are respected by the production plan x. We then count the total number K_{sat} of scenarios in which all demand satisfaction constraints are respected and use the ratio K_{sat}/K as an estimation of the value of the joint probability. The main advantage of this approximation is that it enables to reformulate problem SLS as a mixed-integer linear program.

This is done by introducing a new set of binary variables: $\alpha^k \in \{0,1\}$. α^k is defined by $\alpha^k = 1$ if at least one demand satisfaction constraint is violated in scenario k, $\alpha^k = 0$ otherwise.

This leads to the following mixed-integer linear program denoted by SA in the sequel of this chapter.

$$\begin{cases}
Z_{SA}^* = \min \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T} h_t\Big(\sum_{\tau=1}^{t} x_\tau - \mathbb{E}[\tilde{dc}_{1t}]\Big) & & (4.21)\\
x_t \leq c_t y_t & \forall t \in \mathcal{T} & (4.22)\\
\sum_{\tau=1}^{t} x_\tau \geq dc_{1t}^k(1-\alpha^k) & \forall t \in \mathcal{T}, \forall k = 1...K & (4.23)\\
\sum_{k=1}^{K} \alpha^k \leq \lfloor K\epsilon \rfloor & & (4.24)\\
x_t \geq 0 & \forall t \in \mathcal{T} & (4.25)\\
y_t \in \{0,1\} & \forall t \in \mathcal{T} & (4.26)\\
\alpha^k \in \{0,1\} & \forall k = 1...K & (4.27)
\end{cases}$$

Constraints (4.23) make sure that, if $\alpha^k = 0$, the T demand satisfaction constraints corresponding to scenario k are satisfied by the production plan x. Constraint (4.24) is the joint probability constraint: it limits the number of violated scenarios to $\lfloor K\epsilon \rfloor$, thus ensuring that the ratio K_{sat}/K is above $1-\epsilon$.

Problem SA is based on an approximate representation of constraint (4.11) of problem SLS. As a consequence, there is no definite guarantee that solving problem SA will provide a feasible solution of problem SLS. However, Luedtke and Ahmed (2008) showed that, when the uncertainty is located only on the right hand side of the constraints as it is the case here, the probability that problem SA provides a feasible solution to problem SLS increases exponentially fast with the sample size K and tends to 1 when K tends to infinity.

Hence the larger the sample size K, the higher the probability that problem SA provides a feasible solution to problem SLS. However, using a large sample size K means introducing a large number of binary variables α^k and of big-M type constraints (4.23) in the formulation. Thus, even if problem SA is a mixed-integer linear program, its resolution by a mathematical programming solver poses some computational difficulty in practice.

Several mixed-integer linear programming techniques have been recently proposed to remedy to this difficulty. Valid inequalities exploiting the fact that, for a given period t, constraints

(4.23) define a mixing set subject to the additional cardinality constraint (4.24) are proposed by Luedtke et al. (2010) and further improved by Küçükyavuz (2012). Luedtke et al. (2010) also investigate the use of a strong extended formulation. As our preliminary numerical results indicated that solving problem SA using this extended formulation was more efficient than solving it using the valid inequalities proposed by Luedtke et al. (2010) and Küçükyavuz (2012), we used this extended formulation in our computational experiments. This reformulation of Problem SA is denoted by SAExt in what follows.

4.5 Partial sample approximation approach

As will be shown by the computational results presented in Section 4.6, the sample approximation approach, even when using the extended formulation SAExt strengthened by lot-sizing valid inequalities, fails at providing feasible solutions of problem SLS for small samples and leads to a significant computation time when the sample size K increases. This is why we propose in what follows to use a new extension of this approach. This extension is based on the assumption that the demand in the first period $\tilde{d}_1$ is statistically independent of the demand in the other periods. Note that $\tilde{d}_1$ plays a special role in the joint chance constraint (4.11) as this is the only random variable appearing in all the demand satisfaction constraints. Similarly to the sample approximation approach, the proposed extension relies on a Monte Carlo sampling method. However, the sampling is not carried out on all the random variables involved in the stochastic problem (i.e. on $\tilde{d}_1$, $\tilde{d}_2$, ..., $\tilde{d}_T$) but only on part of them (more precisely on $\tilde{d}_2$, ..., $\tilde{d}_T$ in our case). We thus refer to it as the partial sample approximation approach.

We first explain in Subsection 4.5.1 how the proposed method leads to the formulation of a new chance-constrained program featuring a series of K joint chance constraints, each one involving a single random variable $\tilde{d}_1$. We then focus on the frequently encountered special case where $\tilde{d}_1$ follows a normal distribution (see Subsection 4.5.2). We show how, in this case, the use of a conservative convex approximation of the feasible set defined by each of the K joint chance constraints leads to the formulation of a deterministic MILP involving the same number of binary variables as the original problem SLS.

4.5.1 General case

Let $\widetilde{dc}_{2t}$ be the random variable representing the cumulative demand over periods 2 to t, for each period $t = 2...T$. Let d^1,.., d^k,..., d^K be a Monte Carlo sample of the random vector $\tilde{d}$ and $dc^k_{2t} = \sum_{\tau=2}^{t} d^k_\tau$ be the cumulated demand over $[2, t]$ in scenario k.

The partial sample approximation approach relies on the idea that, given a production plan x, the value of the joint probability involved in constraint (4.11) can be approximately computed as follows:

$$\Pr\Big(\sum_{\tau=1}^{t} x_\tau \geq \widetilde{dc}_{1t}, \forall t\Big) = \Pr\Big(\sum_{\tau=1}^{t} x_\tau - \widetilde{dc}_{2t} \geq \tilde{d}_1, \forall t\Big) \tag{4.28}$$

$$= \mathbb{E}\Big[\mathbb{I}\Big(\sum_{\tau=1}^{t} x_\tau - \widetilde{dc}_{2t} \geq \tilde{d}_1, \forall t\Big)\Big] \tag{4.29}$$

$$= \mathbb{E}_{\widetilde{dc}_{2t}, t=2...T}\mathbb{E}_{\tilde{d}_1}\Big[\mathbb{I}\Big(\sum_{\tau=1}^{t} x_\tau - \widetilde{dc}_{2t} \geq \tilde{d}_1, \forall t\Big)\Big] \tag{4.30}$$

$$= \mathbb{E}_{\widetilde{dc}_{2t}, t=2...T}\Big[\Pr\big(\sum_{\tau=1}^{t} x_\tau - \widetilde{dc}_{2t} \geq \tilde{d}_1, \forall t\big)\Big] \tag{4.31}$$

$$\approx \frac{1}{K}\sum_{k=1}^{K} \Pr\Big(\sum_{\tau=1}^{t} x_\tau - dc_{2t}^k \geq \tilde{d}_1, \forall t\Big) \tag{4.32}$$

Equalities (4.29) and (4.31) rely on the fact that the probability of an event is equal to the expected value of an indicator function that is one if the event has occurred and zero otherwise. Equality (4.30) makes use of the assumption that $\tilde{d}_1$ is statistically independent of $\widetilde{dc}_{2t}, t = 2...T$, to decompose the computation of the expected value in two parts. The expected value appearing in (4.31) is then approximately computed via a sample average approximation in (4.32). The idea underlying the proposed partial sample approximation approach can thus be understood as follows. For each scenario k, the probability $\psi^k = \Pr\Big(\sum_{\tau=1}^{t} x_\tau - dc_{2t}^k \geq \tilde{d}_1, \forall t \in \mathcal{T}\Big)$ that no stockout occurs during the whole planning horizon in case the realized demand over periods 2 to T corresponds to the k^{th} sampled demand vector is computed. The expected value of ψ^k over all scenarios, i.e. $\sum_{k=1}^{K} \psi^k/K$, is then used as an estimation of the value of the joint probability. We refer the reader to Cheng et al. (2019) for a more thorough mathematical discussion on this approximation of the joint probability.

This approximation enables us to reformulate problem SLS as a new joint chance-constrained program. This is done by introducing the decision variables $\psi^k, k = 1...K$. $\psi^k \in [0,1]$ represents the probability that no stock-out occurs during the whole planning horizon in case the realized demand over periods 2 to T corresponds to the k^{th} sampled demand vector d^k. Note that, contrary to what is done in the sample approximation where a binary variable α^k has to be introduced for each scenario k, the variables ψ^k introduced in the partial sample approximation approach are continuous variables.

This leads to the following chance-constrained program denoted by PSA in the sequel of the paper.

$$\begin{cases} Z_{PSA}^* = \min \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T} h_t\Big(\sum_{\tau=1}^{t} x_\tau - \mathbb{E}[\widetilde{dc}_{1t}]\Big) & & (4.33)\\ x_t \leq c_t y_t & \forall t \in \mathcal{T} & (4.34)\\ \psi^k = \Pr\Big(\sum_{\tau=1}^{t} x_\tau - dc_{2t}^k \geq \tilde{d}_1, \forall t \in \mathcal{T}\Big) & \forall k = 1...K & (4.35)\\ \frac{1}{K}\sum_{i=1}^{K} \psi^k \geq 1 - \epsilon & & (4.36)\\ x_t \geq 0 & \forall t \in \mathcal{T} & (4.37)\\ y_t \in \{0,1\} & \forall t \in \mathcal{T} & (4.38)\\ \psi^k \in [0,1] & \forall k = 1...K & (4.39) \end{cases}$$

Constraints (4.35) compute the probability that all demand satisfaction constraints are satisfied by the production plan x in scenario k. Constraint (4.36) is the joint probability constraint ensuring that the expected value of ψ^k over all scenarios is above the minimum acceptable value.

We note that, similarly to the sample approximation approach, problem PSA is based on an approximate representation of constraint (4.11) of problem SLS. There is thus no definite guarantee that it will provide feasible solutions of problem SLS. However, Cheng et al. (2019) show that the partial sample approximation approach has the same convergence properties as the sample approximation approach. For instance, under certain conditions, the optimal value of problem PSA converges to the optimal value of problem SLS with probability one when K tends to infinity.

Moreover, we would like to point out that formulation PSA involves a series of K joint chance-constraints (4.35). However, these constraints involve a single random variable $\tilde{d}_1$ and will thus be somewhat easier to handle than the initial joint chance-constraint (4.11). We namely have:

$$\psi^k = \Pr\left(min_{t\in\mathcal{T}}\{\sum_{\tau=1}^{t} x_\tau - dc_{2t}^k\} \geq \tilde{d}_1\right) \tag{4.40}$$

$$= F_1(min_{t\in\mathcal{T}}\{\sum_{\tau=1}^{t} x_\tau - dc_{2t}^k\}) \tag{4.41}$$

where F_1 denotes the cumulative probability distribution of $\tilde{d}_1$.

We refer the reader to Cheng et al. (2019) for a general discussion on how to handle the probabilistic constraints (4.35) for a variety of probability distributions F_1 and focus in what follows on the special case of a normally distributed demand.

4.5.2 Special case of a normally distributed demand

We now consider the case where $\tilde{d}_1$ follows a normal distribution with mean d_1 and standard deviation χ_1. The normal distribution is widely used to represent the demand and the forecasting errors in standard inventory management and forecasting models although it has the drawback of placing some probability on unrealistic negative demand values.

In this subsection, we discuss a conservative convex approximation of problem PSA in the case where $\tilde{d}_1$ follows a normal distribution. This approximation makes use of the fact that F_1 is convex over $]-\infty; d_1]$ and concave over the interval $[d_1, +\infty[$.

This conservative piecewise linear approximation of F_1, denoted by $\underline{F}_1$, uses a set of $B+1$ breakpoints $\phi_\beta, \beta = 0...B$, such that $\phi_0 = d_1$ and $\phi_\beta > \phi_{\beta-1}, \forall \beta = 1...B$. We define:

$$F_1(\varphi) \approx \underline{F}_1(\varphi) = \begin{cases} sl_0\varphi + cst_0 & \text{if } \varphi \leq \phi_0 \quad (4.42) \\ sl_\beta\varphi + cst_\beta & \text{if } \varphi \in [\phi_{\beta-1}; \phi_\beta], \forall \beta = 1...B \quad (4.43) \\ sl_B\phi_B + cst_B & \text{if } \varphi \geq \phi_B \quad (4.44) \end{cases}$$

where the slope and intercept of each segment are computed as follows:

$$\begin{cases} sl_0 = F_1'(\phi_0) \text{ and } cst_0 = F_1(\phi_0) - sl_0\phi_0 & (4.45) \\ sl_\beta = \dfrac{F_1(\phi_\beta) - F_1(\phi_{\beta-1})}{\phi_\beta - \phi_{\beta-1}} \text{ and } cst_\beta = F_1(\phi_\beta) - sl_\beta\phi_\beta \quad \forall \beta = 1...B & (4.46) \end{cases}$$

Note that we have $F_1(\varphi) \geq \underline{F}_1(\varphi), \forall \varphi$. Namely:

- $[\varphi \mapsto sl_0\varphi + cst_0]$ is the tangent line to the curve $[\varphi \mapsto F_1(\varphi)]$ at point d_1. As F_1 is convex over $]-\infty; d_1]$, we have: $F_{D_1}(\varphi) \geq sl_0\varphi + cst_0, \forall \varphi \leq \phi_0$.
- $[\varphi \mapsto sl_\beta\varphi + cst_\beta], \beta = 1...B$ are chords between two points $\phi_{\beta-1}$ and ϕ_β belonging $[d_1, +\infty[$. As F_1 is concave over this interval, we have: $F_1(\varphi) \geq sl_\beta\varphi + cst_\beta, \forall \varphi \geq \phi_0, \forall \beta = 1...B$.
- F_1 is strictly increasing over $\mathbb{R}$. This implies $F_1(\varphi) \geq F_1(\phi_B) = sl_B\phi_B + cst_B, \forall \varphi \geq \phi_B$.

This allows us to reformulate problem PSA as the following mixed-integer linear program denoted by PSACons_N.

$$
\begin{cases}
Z^*_{PSACons\_N} = min \sum_{t=1}^{T} f_t y_t & \\
\qquad + \sum_{t=1}^{T} h_t \Big(\sum_{\tau=1}^{t} x_\tau - \mathbb{E}[DC_t] \Big) & (4.47) \\
x_t \leq c_t y_t \qquad \forall t \in \mathcal{T} & (4.48) \\
\psi^k \leq sl_\beta \big(\sum_{\tau=1}^{t} x_\tau - dc^k_{2t} \big) + cst_b \qquad \forall k, \forall t \in \mathcal{T}, \forall \beta = 0...B & (4.49) \\
\psi^k \leq sl_B\phi_B + cst_B \qquad \forall k & (4.50) \\
\frac{1}{K} \sum_{k=1}^{K} \psi^k \geq 1 - \epsilon & (4.51) \\
x_t \geq 0 \qquad \forall t \in \mathcal{T} & (4.52) \\
y_t \in \{0,1\} \qquad \forall t \in \mathcal{T} & (4.53) \\
\psi^k \in [0,1] \qquad \forall k = 1...K & (4.54)
\end{cases}
$$

Note how constraints (4.49)-(4.50) compute the value of the probabilities in the approximation, ψ^k, via a set of linear inequalities.

Even if PSACons_N is a mixed-integer linear program, the high number of constraints (4.49) poses some numerical difficulties in practice. We thus investigate the determination of a non trivial value for the minimum deterministic cumulative demand to be satisfied by any feasible solution of problem SLS. This information will be used to ease the resolution of problem PSACons_N by a MILP solver through: (i) the reduction of the MILP size thanks to the early detection of redundant constraints (4.49), (ii) the formulation strengthening through valid inequalities available for deterministic single-item capacitated lot-sizing. The reader is referred to Gicquel and Cheng (2018) for more detail about it.

4.6 Computational experiments

We discuss in this section the results of some computational experiments carried out on small to medium size instances of the problem. The main objective of these experiments is to assess the effectiveness of the proposed partial sample approximation approach by comparing it with the Bonferroni conservative approximation and the sample approximation approach.

4.6.1 Instances

We generated test problems by considering a planning horizon of $T = 20$ periods. In each problem, the inventory holding and setup costs are assumed to be time-independent and are

set to $h = 1$ and $f = 50$. Similarly, the production capacity is assumed to be time-independent and is set to $c = 100$.

We consider the case in which $\tilde{d}_t, t \in \mathcal{T}$, are statistically independent of one another and follow a normal distribution of mean $d = 30$ and standard deviation $\chi = 10$.

The size of the sample obtained by the Monte Carlo method, K, is varied between 100 and 5000 scenarios: $K \in \{100, 500, 1000, 2000, 5000\}$. Note that, as the demand is generated from a normal distribution, it may happen in a few cases that the sampled value d_t^k takes a negative value. When this happens, we set the value of d_t^k to 0 to avoid introducing unrealistic negative demands in the sampled scenarios.

Five possible values are considered for the maximum acceptable risk level: $\epsilon \in \{0.15, 0.10, 0.05, 0.02, 0.01\}$.

We defined a reference set of 10 instances in which $K = 1000$ and $\epsilon = 0.05$. We then created 8 additional sets of instances, each time changing either the value of K or the value of ϵ in order to assess the impact of these parameters on the problem resolution. For each considered set, 10 samples of scenarios, i.e. 10 instances, were randomly generated, leading to a total of 90 instances.

All tests were run on an Intel Core i5 (2.6 GHz) with 4 Go of RAM, running under Windows 7. We used the mathematical programming solver CPLEX 12.6 with the default settings to solve the problem using the MILP formulations discussed in Sections 4.3 to 4.5. The piecewise linear approximation of F_1 used in formulation PSACons_N was built with $B = 4$, $\phi_0 = d_1$, $\phi_1 = d_1 + 0.5\chi_1$, $\phi_2 = d_1 + \chi_1$, $\phi_3 = d_1 + 1.5\chi_1$ and $\phi_4 = d_1 + 3\chi_1$.

4.6.2 Results

Tables 4.1 to 4.2 display the corresponding results. Each line corresponds to the average value over the corresponding 10 instances. We provide for each set of 10 instances:

- *Bin*: the average number of binary variables in the formulation.
- *Cons*: the average number of constraints in the formulation.
- *Cost*: the average value of the optimal solution of the corresponding MILP.
- *Prob*: the average value of the probability $\Pr\left(\sum_{\tau=1}^{t} x_\tau^* \geq \tilde{dc}_{1t}, \forall t \in \mathcal{T}\right)$ where x^* is the optimal solution of the MILP formulation. *Prob* corresponds to a post-optimization estimation of the joint probability. It is is obtained by using a sample of $K_c = 100000$ scenarios different from the ones used in the optimization phase. For each instance, we count the number K_s of scenarios for which all demand satisfaction constraints are satisfied by the corresponding production plan x^*. *Prob* is then computed as the proportion K_s/K_c of scenarios for which there is no violation. We consider that x^* is a feasible solution of problem SLS in case *Prob* is greater or equal to $1 - \epsilon$ and an infeasible solution otherwise.
- *Feas*: the number of instances (out of the 10 corresponding ones) for which the production plan x^* is a feasible solution of problem SLS (i.e. is such that $Prob \geq 1 - \epsilon$).
- *Time*: the average computation time needed to solve to optimality the MILP formulation.

Results from Table 4.2 first show that formulation BON based on the Bonferroni approximation is capable of providing feasible solutions of problem SLS within very short computation times (less than 1s in most cases). However, these solutions are rather conservative. The value of *Prob* is namely significantly higher than the required value $1 - \epsilon$ in all cases. As a result, these solutions are significantly more expensive than the solutions provided by the other formulations: for instance, the solutions of formulation BON are on average 14.1% more expensive than the ones of formulation PSACons_N.

Results from Tables 4.1 and 4.2 also show that formulation SAExt fails at providing feasible solutions of problem SLS for the sample sizes considered in this work. This can be seen by the facts that only 1 of the 90 instances considered in Tables 4.1 to 4.2 has a solution satisfying the joint probability constraint (4.11) when solved with SAExt and that the average value of *Prob* is below the targeted value $1 - \epsilon$ in all cases. In contrast, using formulation PSACons_N, feasible solutions of problem SLS are obtained for 71 out of the 90 instances and the average value of *Prob* is above the targeted value $1 - \epsilon$ in most cases. This difference might be explained by the fact that the partial approximation uses a conservative piecewise linear representation of the cumulative probability distribution F_1 whereas the sample approximation relies on a discrete approximation of F_1 obtained by sampling. Moreover, the average computation time is decreased from 171s when using formulation SAExt to 75s when using formulation PSACons_N. This difference is mainly explained by the fact that formulation PSACons_N involves a number of binary variables significantly smaller than formulation SAExt. Finally, we note that the solutions provided by PSACons_N might be too conservative. Namely, in some cases, the value of *Prob* is significantly larger than $1 - \epsilon$. This might indicate the existence of less expensive feasible solutions of problem SLS which none of the studied approaches is capable of finding.

Computational experiments carried out on a larger set of instances (including among others the case where $\tilde{d}_t, t \in \mathcal{T}$, follow a uniform distribution) are reported in Gicquel and Cheng (2018). In a nutshell, they show that:

- The computation time of the sample approximation and the partial sample approximation approaches significantly increases with the sample size K. However, a minimum sample size is required to ensure a good approximation of the joint probability and to obtain feasible solutions of problem SLS. For the partial sample approximation approach, the value $K = 1000$ seems to be an acceptable trade-off. But for the sample approximation approach, a sample size of at least $K = 5000$ scenarios seems to be necessary.
- When the value of the maximum acceptable risk level ϵ decreases, the size of the mixed-integer linear programs obtained with the sample approximation and the partial sample approximation approaches decreases, leading to an overall decrease in the computation time. However, it appears that the smaller ϵ, the more difficult it is to find feasible solutions of problem SLS.
- The horizon length T also has a direct impact of the size of the mixed-integer linear programs so that increasing T leads to longer computation times.
- The values of the ratio $\frac{f}{h}$, of the production capacity c and of the standard deviation χ_1 of $\tilde{d}_1$ appear to have a limited impact on both the number of feasible solutions obtained and the computation time.

4.7 Conclusion and perspectives

We studied the single-item single-resource capacitated lot-sizing problem with stochastic demand and proposed to handle this problem using a single-stage stochastic programming approach. More precisely, we formulated this stochastic problem as a joint chance-constrained program where the probability that an inventory shortage occurs during the planning horizon is limited to a maximum acceptable risk level.

TABLE 4.1: Impact of the sample size K
Instances with $\epsilon = 0.05$

K	Formulation	Bin	Cons	Cost	Prob	Feas	Time
100	SAExt	220	241	1933.0	0.856	0	1.3s
	PSACons_N	20	1108	2162.1	0.932	1	1.8s
500	SAExt	1020	1041	2100.9	0.919	0	6.3s
	PSACons_N	20	5102	2252.9	0.955	8	8.9s
1000	SAExt	2020	2041	2140.8	0.934	0	20.3s
	PSACons_N	20	10362	2265.8	0.958	10	25.1s
2000	SAExt	4020	4041	2150.6	0.937	0	114.8s
	PSACons_N	20	20706	2254.2	0.957	9	28.1s
5000	SAExt	10020	10041	2181.1	0.945	1	695.1s
	PSACons_N	20	51443	2265.2	0.960	10	309.3s

TABLE 4.2: Impact of the maximum acceptable risk level ϵ
Instances with $K = 1000$

ϵ	Formulation	Bin	Cons	Cost	Prob	Feas	Time
0.15	BON	20	40	2346.1	0.966	10	0.5s
	SAExt	4020	6041	1817.9	0.832	0	555.2s
	PSACons_N	20	47153	1931.3	0.880	10	233.0s
0.10	BON	20	40	2437.2	0.976	10	0.5s
	SAExt	3020	4041	1951.4	0.882	0	137.9s
	PSACons_N	20	17959	2068.8	0.920	10	53.6s
0.05	BON	20	40	2584.1	0.987	10	0.5s
	SAExt	2020	2041	2140.8	0.934	0	20.3s
	PSACons_N	20	10362	2265.8	0.958	10	25.1s
0.02	BON	20	40	2771.2	0.994	10	0.5s
	SAExt	1420	841	2346.5	0.964	0	5.1s
	PSACons_N	20	5420	2498.0	0.982	8	11.3s
0.01	BON	20	40	2897.6	0.997	10	0.5s
	SAExt	1220	441	2492.8	0.979	0	3.3s
	PSACons_N	20	3377	2658.2	0.990	5	6.3s

As the resulting probabilistic mixed-integer program is computationally difficult to handle, we investigated the development of an approximate solution method which can be seen as an extension of the previously published sample approximation approach. This extension is based on the assumption that the demand in the first period is statistically independent of the demand in the other periods. Similarly to the the sample approximation approach presented by Luedtke and Ahmed (2008), the proposed extension relies on a Monte Carlo sampling method. However, this sampling is carried out on only part of the random variables, more precisely on all random variables except $\tilde{d}_1$. Provided there is no correlation between $\tilde{d}_1$ and the demand in the later periods, this partial sampling results in the formulation of a chance-constrained program featuring a series of joint chance constraints. Each of these constraints involves a single random variable and defines a feasible set for which a conservative convex approximation can be quite easily built. The main advantage of the proposed partial sample approximation approach lies in the fact that it leads to the formulation of a deterministic mixed-integer linear problem having the same number of binary variables as the original problem. Our computational results show that the proposed solution method is more efficient at finding feasible solutions of the original stochastic problem than the sample approximation method and that these solutions are less costly than the ones provided by the Bonferroni conservative approximation. Moreover, the computation time is significantly shorter than the one needed for the sample approximation method. This work was published in a journal paper: see Gicquel and Cheng (2018).

Among the possible directions for future research suggested by the present work, it would be interesting to consider the multi-item capacitated lot-sizing problem with stochastic demand as this would help close the gap between the industrial need and the academic state of the art. This might be achieved e.g. by developing a Benders decomposition approach in which the master problem would decide on the production plan for all the items and each single-item sub-problem would focus on the feasibility of this production plan with respect to the corresponding joint chance-constraint.

Chapter 5

Multi-stage stochastic lot-sizing

5.1 Introduction

The joint chance-constrained lot-sizing problem investigated in Chapter 4 enables us to take into account in the problem modeling the uncertainty on the future demand and to control the probability that an inventory shortage occurs during the planning horizon. However, it is a single-stage stochastic model, requiring that all production decisions (production quantities and setups) are made at the beginning of the planning horizon. It thus does not take into account the possibility to adjust these decisions afterward when the value of the uncertain parameters relative to the beginning of the horizon is known. Yet, in practice, production planning is often carried out within a rolling horizon framework: we thus compute a production plan for an horizon spanning T periods but only implement the decisions relative to the first $T' < T$ periods. After the end of period T', we update the inventory level and the demand forecasts and recompute a new production plan for periods $T' + 1...T' + T$ in which the decisions relative to periods $T' + 1...T$ can be modified. Hence, production planning is intrinsically a multi-stage decision process in which production decisions are not made once and for all but rather adjusted over time according to the actual realizations of the uncertain parameters. Single-stage stochastic models such as the joint chance-constraint lot-sizing problem discussed in Chapter 4 do not take this flexibility into account and as a consequence, they might lead to overly conservative and expensive production plans.

Since 2016, I have thus been investigating stochastic programming models in which the multi-stage aspect of the production planning process is explicitly taken into account. The present chapter presents this work which was carried out as part of the work of Franco Quezada, a PhD student which I have been co-supervising with Safia Kedad-Sidhoum since 2018. A preliminary version was published as a conference paper (Quezada et al., 2019) and a full version is currently under revision for publication in a journal (Quezada et al., 2020a).

In view of the numerical difficulties lying ahead, we chose to focus on the simplest available lot-sizing problem, the uncapacitated single-item lot-sizing (ULS) problem, and investigated a multi-stage stochastic extension of this problem denoted by SULS in what follows. In this problem modeling, the value of the uncertain parameters (demand and costs) is assumed to unfold little by little following a discrete-time stochastic process and the production decisions can be made progressively as more and more information on the demand and cost realizations are collected. In order to address this problem, we rely on a multi-stage stochastic integer programming approach. We consider that the underlying stochastic input process has a finite probability space and represents the information on the evolution of the uncertain parameters by a discrete scenario tree. Moreover, we rely on the commonly used assumption that the scenario tree is stage-wise independent, i.e. that the set of children nodes of any two nodes belonging to the same stage are defined by identical data and conditional probabilities. Our objective is to develop an efficient solution approach for this problem.

Using a scenario tree to represent the evolution of the uncertain parameters leads to the MILP formulation introduced in Section 2.3.3. This mixed-integer linear program can be solved

using mathematical programming solvers, at least for small-size scenario trees. Several works focus on the polyhedral study of this MILP in order to strengthen its linear relaxation and improve the computational efficiency of the branch-and-cut algorithms embedded in MILP solvers. Valid inequalities are discussed by Guan et al. (2006), di Summa and Wolsey (2008) and Guan et al. (2009) and extended formulations are proposed by Ahmed et al. (2003) and Zhao and Guan (2014). In particular, Guan et al. (2009) propose a general method for generating cutting planes for multi-stage stochastic integer programs based on combining valid inequalities previously known for the deterministic variant of the corresponding problem and apply it on the SULS. Their numerical results show that a branch-and-cut algorithm based on these new inequalities is more effective at solving instances on medium-size scenarios than a stand-alone mathematical programming solver.

In general, solution approaches based on strengthening MILP formulations do not scale up well with the size of the scenario tree. They namely entail solving very large-scale (mixed-integer) linear programs, with millions of variables and constraints, which leads to memory issues and/or prohibitive computation times in practice. Decomposition methods, such as the nested Benders' decomposition algorithm, are thus an attractive alternative to tackle instances with large-size scenario trees. In particular, the Stochastic Dual Dynamic Programming (SDDP) approach is a sample-based nested Benders' decomposition appraoch proposed by Pereira and Pinto (1991) which has been widely used to solve large-size multi-stage stochastic linear programs. This approach relies on a dynamic programming formulation of the stochastic problem and leads to a decomposition of the overall problem into a series of small deterministic subproblems. Each of these problems focuses on making decisions for a small subset of nodes belonging to the same scenario and the same decision stage, taking into account not only the current cost of these decisions but also their future cost which is represented by an expected cost-to-go function. In a linear setting, the expected cost-to-go functions are convex and piecewise linear and can thus be under-approximated through a set of supporting hyperplanes. The SDDP algorithm builds such an approximation by iteratively adding Benders' cuts to each subproblem and converges to an optimal solution in a finite number of iterations.

Recently, Zou et al. (2019) proposed a new extension of the SDDP algorithm, called the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm, capable of solving multi-stage stochastic integer programs in which the state variables, i.e. the variables linking the nodes to one another, are restricted to be binary. One of their main contributions was to introduce a new class of cutting planes, called Lagrangian cuts, which satisfies the validity, tightness and finiteness conditions ensuring the convergence of the algorithm to optimality. For problems such as lot-sizing problems in which the state variables are not binary but continuous, the authors propose to introduce auxiliary binary variables in order to make a binary approximation of the state variables. However, for large size scenario trees, this approximation might be computationally inefficient and leads to large optimality gaps as shown e.g. by the numerical results presented by Quezada et al. (2019).

In order to improve its numerical efficiency, we propose here a new extension of the SDDiP algorithm which aims at combining this decomposition approach with a polyhedral approach based on strong linear relaxations. We focus on one of the key components of the SDDiP algorithm, the expected cost-to-go functions used at each stage, and seek to be more computationally efficient in their management. This is mainly achieved by reducing the number of these expected cost-to-go functions and by exploiting the current knowledge on the polyhedral structure of the SULS to more quickly build good piecewise linear approximations of these functions.

The contributions of this work are threefold. First, we propose a new extension of the SDDiP algorithm in which a partial decomposition of the scenario tree is used to generate subproblems. More precisely, whereas the SDDiP algorithm fully decomposes the original problem

into small deterministic sub-problems, we partially decompose the problem into a set of somewhat larger stochastic sub-problems, each one involving a subset of nodes forming a sub-tree of the initial scenario tree. This results in a reduction of the number of the expected cost-to-go functions for which an approximation has to be iteratively built and in an improvement of the feasible production plan obtained at a given iteration of the algorithm thanks to a less myopic decision making. To the best of our knowledge, this is the first time such an extension is studied in the context of the SDDiP algorithm. Second, we propose to take advantage of the tree structure of the sub-problems and exploit results on the polyhedral structure of the SULS to generate more cuts at each iteration. This enables us to more quickly build accurate approximations of the expected cost-to-go functions. Third, we carry out extensive computational experiments to assess the performance of the proposed algorithm at solving the SULS. We thus compare its performance with the one of a stand-alone mathematical solver and the one of the SDDiP algorithm proposed by Zou et al. (2019). The results show that this new algorithm outperforms ILOG-CPLEX and the SDDiP algorithm at solving large-size instances of the SULS.

The remaining part of this chapter is organized as follows. Section 5.2 introduces a deterministic equivalent mixed-integer linear programming formulation and a stochastic dynamic programming formulation of the SULS. Section 5.3 presents the extension of the SDDiP algorithm in which a partial decomposition of the scenario tree is used. Section 5.4 then describes two further enhancements of the algorithm. Finally, the results of our computational experiments are reported in Section 5.5. Conclusions and directions for further work are discussed in Section 5.6.

5.2 Mathematical formulations

We aim at planning the production of a single type of item on a single resource over a planning horizon $\mathcal{T} = \{1, ..., T\}$ of T periods under uncertain demand and costs. We consider a decision process involving Σ decision stages and denote by $\mathcal{S} = \{1, ..., \Sigma\}$ the set of stages. A stage may correspond to one or several consecutive planning periods. This is of particular interest in the context of lot-sizing problems as the time discretization used by the decision-makers to plan production activities is indeed usually finer than the one used to update the demand and cost forecasts and readjust the production plan. A planning period may thus typically correspond to an 8-hours shift or a day whereas a stage may correspond to a week or a month. Let $\mathcal{T}^\sigma$ be the set of time periods belonging to stage $\sigma \in \mathcal{S}$. Note that the sets $\{\mathcal{T}^\sigma, \sigma \in \mathcal{S}\}$ form a partition of $\mathcal{T}$.

We assume a stochastic input process with finite probability space. The resulting information structure can be represented by a scenario tree. With a slight abuse of notation, we will refer to this scenario tree (and all other scenario sub-trees involved in the present work) by mentioning only its set of nodes $\mathcal{V}$. Each node $n \in \mathcal{V}$ corresponds to a single time period t^n and a single-stage σ^n. Let $\mathcal{V}^t$ be the set of nodes belonging to time period t. Each node n represents the state of the system that can be distinguished by the information unfolded up to time period t^n. Each node n has a unique predecessor node denoted a^n belonging to time period $t^n - 1$. By convention, the root node of the scenario tree is indexed by 1 and a^1 is set to 0. At any non-leaf node of the tree, one or several branches indicate future possible outcomes of the random variables from the current node. Let $\mathcal{C}(n)$ be the set of immediate children of node n, $\mathcal{V}(n)$ the sub-tree of $\mathcal{V}$ rooted in n and $\mathcal{L}(n)$ the set of leaf nodes belonging to $\mathcal{V}(n)$. The probability associated with the state represented by node n is denoted by ρ^n. A scenario is defined as a path from the root node to a leaf node in the scenario tree and represents a possible outcome of the stochastic input parameters over the whole planning horizon. The set of nodes on the path from node n to node m is denoted by $\mathcal{P}(n, m)$. The reader can refer to Figure 5.1 for an illustration of this notation on a small scenario tree.

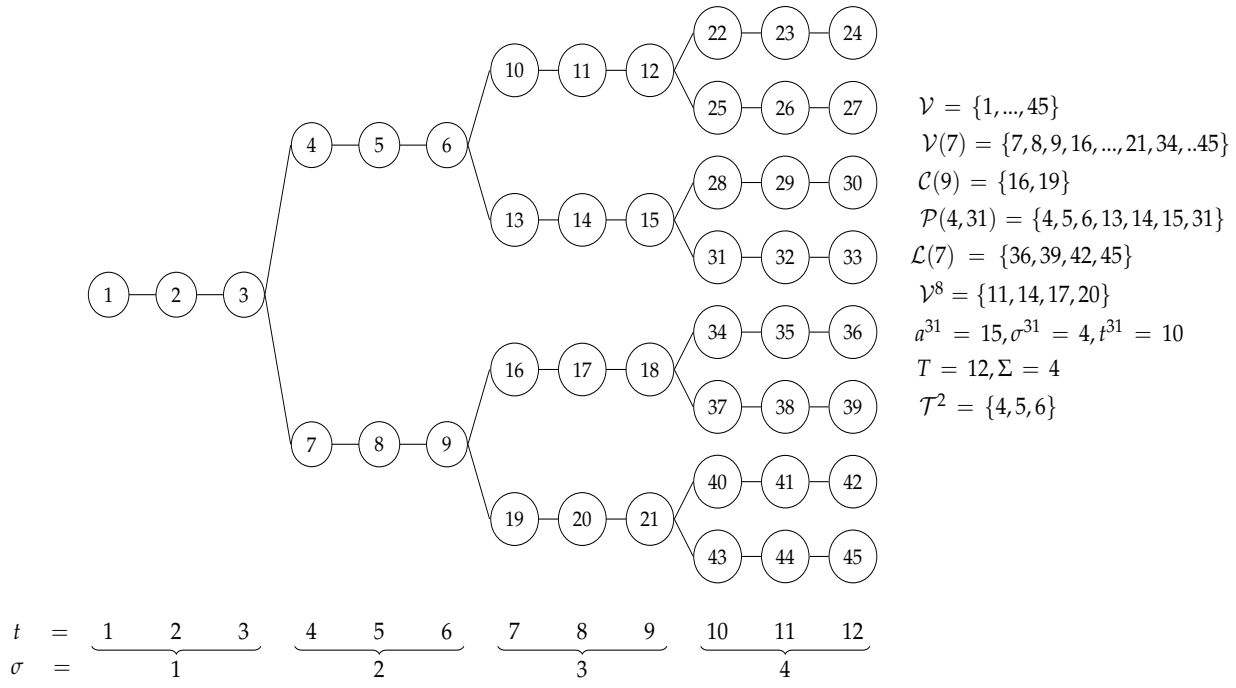

FIGURE 5.1: Scenario tree structure

The stochastic input parameters are defined as follows:

- d^n: demand at node $n \in \mathcal{V}$,
- f^n: setup cost at node $n \in \mathcal{V}$,
- h^n: unit inventory holding cost at node $n \in \mathcal{V}$,
- g^n: unit production cost at node $n \in \mathcal{V}$.

Moreover, we assume that at each stage, the realization of the random parameters happens before we have to make a decision for this stage. This means that the values of d^n, f^n, h^n and g^n, for all $n \in \mathcal{T}^\sigma$, are assumed to be known at the beginning of the first period belonging to stage σ.

5.2.1 Extensive MILP formulation

Based on the uncertainty representation described above, the SULS can be reformulated as an equivalent deterministic model in the form of an MILP. We introduce the following decision variables:

- x^n: quantity produced at node $n \in \mathcal{V}$,
- $y^n = 1$ if a setup for production is carried out at node $n \in \mathcal{V}$, $y^n = 0$ otherwise,
- s^n: inventory level at node $n \in \mathcal{V}$,

This leads to the following MILP formulation:

$$\begin{cases} \min \sum_{n \in \mathcal{V}} \rho^n (f^n y^n + h^n s^n + g^n x^n) & & (5.1) \\ x^n \leq M^n y^n & \forall n \in \mathcal{V} & (5.2) \\ s^n + d^n = x^n + s^{a^n} & \forall n \in \mathcal{V} & (5.3) \\ x^n, s^n \geq 0, y^n \in \{0,1\} & \forall n \in \mathcal{V} & (5.4) \end{cases}$$

The objective function (5.1) aims at minimizing the expected total setup, inventory holding and production costs over all nodes of the scenario tree. Constraints (5.2) link the production quantity variables to the setup variables. Note that the value of constant M^n can be set by using an upper bound on the quantity to be processed at node n, usually defined as the maximum future demand as seen from node n, i.e. $M^n = \max_{\ell \in \mathcal{L}(n)} d^{n\ell}$, where $d^{n\ell} = \sum_{m \in \mathcal{P}(n,\ell)} d^m$. Constraints (5.3) are the inventory balance constraints. Constraints (5.4) provide the decision variables domain.

Problem (5.1)-(5.4) can be solved using MILP solvers. Guan et al. (2009) propose to extend the (l, S) inequalities developed by Barany et al. (1984) for the ULS to strengthen its linear relaxation. In this chapter, these inequalities will be referred to as path inequalities when applied on a single scenario of $\mathcal{V}$ and as tree inequalities when applied on a subtree of $\mathcal{V}$: see Guan et al. (2009) for more detail about it. Nonetheless, the size of the formulation grows exponentially fast with the number of nodes $|\mathcal{V}|$ in the scenario tree. This leads to prohibitive computation times in practice even while using a strengthened formulation. We thus investigate in what follows a dynamic programming formulation which serves as a basis to develop a decomposition algorithm to solve the problem.

5.2.2 Dynamic programming formulation

An alternative to the extensive formulation of the SULS discussed above is a dynamic programming formulation involving nested expected cost-to-go functions. This approach decomposes the original problem into a series of smaller sub-problems linked together by dynamic programming equations. When applying the SDDiP algorithm proposed by Zou et al. (2019) on the SULS, a full decomposition of the problem is carried out, resulting in a large number of small sub-problems. Each of these sub-problems is a small deterministic lot-sizing problem aiming at planning production on a subset of nodes corresponding to a single scenario and a single decision stage. In what follows, we propose to consider a partial decomposition of the problem resulting in a smaller number of larger sub-problems, each one being a stochastic lot-sizing problem aiming at planning production on a sub-tree of the original scenario tree.

We introduce some additional notation in order to explain how this partial decomposition is carried out. We first partition the set of decision stages $\mathcal{S} = \{1, \ldots, \Sigma\}$ into a series of macro-stages $\mathcal{G} = \{1, \ldots, \Gamma\}$, where each macro-stage $\gamma \in \mathcal{G}$ contains a number of consecutive stages denoted $\mathcal{S}(\gamma)$. We let $t(\gamma)$ (resp. $t'(\gamma)$) represent the first (resp. the last) time period belonging to macro-stage γ.

Using the set of macro-stages $\mathcal{G}$ defined above, we can decompose the scenario tree $\mathcal{V}$ into a series of smaller sub-trees as follows. For a given macro-stage γ, each node η belonging to the first time period in γ, i.e. each node $\eta \in \mathcal{V}^{t(\gamma)}$, is the root node of a sub-tree defined by the set of nodes $\mathcal{W}^\eta = \cup_{t=t(\gamma),\ldots,t'(\gamma)} \mathcal{V}^t \cap \mathcal{V}(\eta)$. We recall that $\mathcal{V}(\eta)$ is the sub-tree of $\mathcal{V}$ rooted in η, $\mathcal{W}^\eta$ is thus the restriction of $\mathcal{V}(\eta)$ to the nodes belonging to macro-stage γ. Let $\mathfrak{L}(\eta) = \mathcal{W}^\eta \cap \mathcal{V}^{t'(\gamma)}$ be the set of leaf nodes of sub-tree $\mathcal{W}^\eta$. Finally, we denote as $\mho = \cup_{\gamma \in \mathcal{G}} \mathcal{V}^{t(\gamma)}$ the set of sub-tree root nodes induced by $\mathcal{G}$.

To illustrate the notation related to the macro-stages, we use the scenario tree depicted in Figure 5.1. The set of stages $\mathcal{S}$ is partitioned into $\Gamma = 2$ macro-stages with $\mathcal{S}(1) = \{1, 2\}$ and

$\mathcal{S}(2) = \{3,4\}$. The first time period of macro-stage $\gamma = 1$ is $t(1) = 1$, its last time period is $t'(1) = 6$. Similarly, we have $t(2) = 7$ and $t'(2) = 12$. In this case, the set of sub-tree root nodes is $\mho = \{1,10,13,16,19\}$. With this partition, node $\eta = 1$ is the root node of the subtree $\mathcal{W}^1 = \{1,2,3,4,5,6,7,8,9\}$ involving the set of leaf nodes $\mathfrak{L}(1) = \{6,9\}$. Node $\eta = 10$ is the root node of sub-tree $\mathcal{W}^{10} = \{10,11,12,22,23,24,25,26,27\}$ involving the set of leaf nodes $\mathfrak{L}(10) = \{24,27\}$. Sub-trees $\mathcal{W}^{13}$, $\mathcal{W}^{16}$ and $\mathcal{W}^{19}$ are defined in the same way as sub-tree $\mathcal{W}^{10}$.

For each node $\eta \in \mho$, sub-problem P^η is formulated as:

$$\begin{cases} Q^\eta(s^{a^\eta}) = \min \sum_{n\in\mathcal{W}^\eta} \rho^n(f^n y^n + h^n s^n + g^n x^n) + \sum_{\ell\in\mathfrak{L}(\eta)} \sum_{m\in\mathcal{C}(\ell)} Q^m(s^\ell) & (5.5) \\ x^n \leq M^n y^n & \forall n \in \mathcal{W}^\eta \quad (5.6) \\ s^n + d^n = s^{a^n} + x^n & \forall n \in \mathcal{W}^\eta \quad (5.7) \\ x^n, s^n \geq 0, y^n \in \{0,1\} & \forall n \in \mathcal{W}^\eta \quad (5.8) \end{cases}$$

Sub-problem P^η thus focuses on defining the production plan on sub-tree $\mathcal{W}^\eta$ based on the entering stock level s^{a^η} imposed by the parent node of η in the scenario tree. The objective function comprises two terms: a term related to the expected setup, production and inventory holding costs over sub-tree $\mathcal{W}^\eta$ and a term which represents the expected future costs incurred by the production decisions made in sub-tree $\mathcal{W}^\eta$.

In (5.5), $Q^\eta(s^{a^\eta})$ denotes the optimal value of sub-problem P^η as a function of the entering stock level s^{a^η} and $Q^m(s^\ell)$ the optimal value of sub-problem P^m as a function of the entering stock level s^ℓ. The expected cost-to-go function at node $\ell \in \mathfrak{L}(\eta)$ is defined as the expected value of $Q^m(\cdot)$ over all the children of ℓ in the initial scenario tree $\mathcal{V}$, i.e. over all $m \in \mathcal{C}(\ell)$, which gives $\mathcal{Q}^\ell(\cdot) = \sum_{m\in\mathcal{C}(\ell)} Q^m(\cdot)$. The expected future costs of the decisions made in $\mathcal{W}^\eta$ are thus computed as the sum, over all nodes $\ell \in \mathfrak{L}(\eta)$, of $\mathcal{Q}^\ell(s^\ell)$.

We note that in case of $\mathcal{G} \equiv \mathcal{S}$, i.e. in case each macro-stage corresponds to a single initial decision stage, each sub-tree $\mathcal{W}^\eta$ reduces to a set of nodes belonging to a single deterministic scenario involving T^{σ^η} periods and we obtain a decomposition similar to the one used by Zou et al. (2019).

5.3 Sub-tree-based SDDiP algorithm

We now present the proposed extension of the SDDiP algorithm applied to the SULS. This extension relies on the dynamic programming formulation (5.5)-(5.8) and corresponds to a partial decomposition of the original problem into a set of smaller problems, each one expressed on a sub-tree of the scenario tree. As described in the SDDiP proposed by Zou et al. (2019), the main idea is to solve a sequence of sub-problems in which the expected cost-to-go functions $\mathcal{Q}^\ell(\cdot), \ell \in \mathfrak{L}(\eta)$, of each sub-problem $P^\eta, \eta \in \mho$, are iteratively approximated by a piece-wise linear function. However, whereas the original SDDiP considers a large number of small deterministic sub-problems, we use a smaller number of medium-size stochastic sub-problems.

Note that a key assumption for developing a sampling-based nested Benders's decomposition algorithm such as the SDDiP algorithm is that the scenario tree displays the stage-wise independence property. When there are several time periods per decision stage, this property can be defined as follows. For any two nodes m and m' belonging to stage $\sigma - 1$ and such that $t^m = t^{m'} = \max\{t, t \in \mathcal{T}^{\sigma-1}\}$, the set of nodes $\cup_{t\in\mathcal{T}^\sigma} \mathcal{V}^t \cap \mathcal{V}(m)$ and $\cup_{t\in\mathcal{T}^\sigma} \mathcal{V}^t \cap \mathcal{V}(m')$ are defined by identical data and conditional probabilities.

Straightforwardly, when the stage-wise independence property holds, for any two nodes m and m' belonging to the last period $t'(\gamma - 1)$ of macro-stage $\gamma - 1$, the two sets $\cup_{\eta\in\mathcal{C}(m)} \mathcal{W}^\eta$, and $\cup_{\eta\in\mathcal{C}(m')} \mathcal{W}^\eta$ contain $R^\gamma = |\mathcal{C}(m)| = |\mathcal{C}(m')|$ sub-trees defined by identical data and conditional probabilities. The stochastic process can thus be represented at macro-stage γ by a

set $\mathcal{R}^\gamma = \{1, \dots, R^\gamma\}$ of independent realizations. Each realization $\mathcal{X}^{\gamma,r}$ corresponds to a subtree describing one of the possible evolutions of the uncertain parameters over periods $t(\gamma), \dots, t'(\gamma)$. Let $\xi^{\gamma,r}$ denote the root node of $\mathcal{X}^{\gamma,r}$ and $\mathfrak{L}(\gamma, r)$ denote the set of its leaf nodes.

The expected cost-to-go functions thus depend only on the macro-stage rather than on the node, i.e. we have $\mathcal{Q}^\ell(\cdot) \equiv \mathcal{Q}^\gamma(\cdot)$, for all $\ell \in \mathcal{V}^{t'(\gamma)}$. Hence, only one expected cost-to-go function has to be approximated per macro-stage and the cuts generated at different nodes $\ell \in \mathcal{V}^{t'(\gamma)}$ are added to a single set of cuts defining the piece-wise linear approximation of function $\mathcal{Q}^\gamma(\cdot)$. As a consequence, we can define a single sub-problem P^γ per macro-stage and each sub-problem $P^\eta, \eta \in \mathcal{U}$, will be described as $P^{\gamma^\eta}(s^{a^\eta}, \mathcal{X}^{\gamma^\eta,r})$ where $\mathcal{X}^{\gamma^\eta,r}$ is the realization corresponding to $\mathcal{W}^\eta$.

5.3.1 Sub-problem reformulation

We first describe how each sub-problem $P^\gamma(s^m, \mathcal{X}^{\gamma,r})$, for $m \in \mathcal{V}^{t'(\gamma-1)}$ and $r \in \mathcal{R}^\gamma$, can be reformulated to introduce binary state variables.

Namely, in the SULS, the state variables are the continuous inventory variables s^n. As the SDDiP developed by Zou et al. (2019) requires the state variables to be binary, we first carry out a binary approximation of the state variables before applying the algorithm to our problem. This binary approximation is obtained by replacing the continuous variable s^n by a set of binary variables $u^{n,\beta}$ such that $s^n = \sum_{\beta \in \mathcal{B}} 2^\beta u^{n,\beta}$, where $\mathcal{B} = \{1, \dots, B\}$. We have $u^{n,\beta} = 1$ if coefficient 2^β is used to compute the value of s^n and $u^{n,\beta} = 0$ otherwise. We note however that this binary approximation is not needed for all inventory variables, but only for those coupling the sub-problems $P^\gamma(\cdot,\cdot)$, to one another. Thus, in sub-problem $P^\gamma(s^m, \mathcal{X}^{\gamma,r})$, we use a binary approximation for the entering stock s^m at root node $\xi^{\gamma,r}$ and for the leaving stock s^ℓ at each leaf node $\ell \in \mathfrak{L}(\gamma, r)$.

Then, as indicated by Zou et al. (2019), we introduce local copies of the binary state variables relative to root node $\xi^{\gamma,r}$. More precisely, $\hat{u}^{\xi^{\gamma,r},\beta}$ is an auxiliary continuous decision variable representing the value of the state variable $u^{m,\beta}$ at the parent node m. It is thus a local copy in problem $P^\gamma(s^m, \mathcal{X}^{\gamma,r})$ of the state variable $u^{m,\beta}$, the value of which is considered as a given input parameter for this problem.

This leads to the following reformulation of sub-problem $P^\gamma(u^m, \mathcal{X}^{\gamma,r})$:

$$
\begin{cases}
Q^{\gamma,r}(u^m) = \min \displaystyle\sum_{n \in \mathcal{X}^{\gamma,r}} \rho^n (f^n y^n + h^n s^n + g^n x^n) & \\
\qquad\qquad + \displaystyle\sum_{\ell \in \mathfrak{L}(\gamma,r)} \mathcal{Q}^\gamma(u^\ell) & (5.9) \\
x^n \le M^n y^n & \forall n \in \mathcal{X}^{\gamma,r} \quad (5.10) \\
s^{\xi^{\gamma,r}} + d^{\xi^{\gamma,r}} = \displaystyle\sum_{\beta \in \mathcal{B}} 2^\beta \hat{u}^{\xi^{\gamma,r},\beta} + x^{\xi^{\gamma,r}} & (5.11) \\
\hat{u}^{\xi^{\gamma,r},\beta} = u^{m,\beta} & \forall \beta \in \mathcal{B} \quad (5.12) \\
s^n + d^n = s^{a^n} + x^n & \forall n \in \mathcal{X}^{\gamma,r} \setminus \{\xi^{\gamma,r}\} \quad (5.13) \\
s^\ell = \sum_{\beta \in \mathcal{B}} 2^\beta u^{\ell,\beta} & \forall \ell \in \mathfrak{L}(\gamma, r) \quad (5.14) \\
\hat{u}^{\xi^{\gamma,r},\beta} \in [0,1] & \forall \beta \in \mathcal{B} \quad (5.15) \\
u^{\ell,\beta} \in \{0,1\} & \forall \ell \in \mathfrak{L}(\gamma, r), \forall \beta \in \mathcal{B} \quad (5.16) \\
x^n, s^n \ge 0, y^n \in \{0,1\} & \forall n \in \mathcal{X}^{\gamma,r} \quad (5.17)
\end{cases}
$$

where u^n denotes the vector of binary variables $u^n = (u^{n0}, \dots, u^{n\beta}, \dots, u^{nB})$.

In this reformulation, Constraint (5.11) corresponds to the inventory balance at node $\xi^{\gamma,r}$ in which the entering stock level s^m is computed using the auxiliary variables $\hat{u}^{\xi^{\gamma,r}\beta}$. Equalities

(5.12) are copy constraints ensuring that the value of each auxiliary variable $\hat{u}^{\xi^{\gamma,r},\beta}$ is equal to the value of the corresponding state variable $u^{m,\beta}$ imposed by the parent node m. Constraints (5.13) ensure the inventory balance at each node of sub-tree $\mathcal{X}^{\gamma,r}$ except the root node $\xi^{\gamma,r}$. Constraints (5.14) define, for each leaf node $\ell \in \mathfrak{L}(\gamma, r)$, the value of the binary variables $u^{\ell,\beta}$, which will be used to compute the future expected costs as $\mathcal{Q}^{\gamma}(u^{\ell}) = \sum_{r' \in \mathcal{R}^{\gamma+1}} Q^{\gamma+1,r'}(u^{\ell})$. Note that, although variables $\hat{u}^{\xi^{\gamma,r},\beta}$ and constraints (5.12) are redundant for sub-problem $P^{\gamma}(u^{m}, \mathcal{X}^{\gamma,r})$, they will play a key role in the generation of the Lagrangian and strengthened Benders' cuts used in the SDDiP algorithm to approximate the expected cost-to-functions.

The main components of the proposed sub-tree-based SDDiP algorithm applied to the SULS are described in the following.

5.3.2 Sampling step

In the sampling step, a subset of K scenarios, i.e. a set of paths going from the root node to a leaf node, are randomly selected. Let $\Omega_i = \{\omega_i^1, \ldots, \omega_i^k, \ldots, \omega_i^K\}$ be the set of sampled scenarios, ω_i^k be the set of nodes belonging to scenario k at iteration i and $r_i^{k,\gamma}$ be the index of the realization in $\mathcal{R}^{\gamma}$ containing the values of the uncertain parameters in scenario ω_i^k at macro-stage γ.

5.3.3 Forward step

At iteration i, the forward step proceeds stage-wise from $\gamma = 1$ to Γ. For each sampled scenario ω_i^k and each macro-stage γ, we solve problem $P_i^{\gamma}(u_i^m, \mathcal{X}^{\gamma,r_i^{k,\gamma}})$ where $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma-1)}$ is the node in the sampled scenario ω_i^k belonging to the last period of γ. To solve this problem, the expected future costs are computed using an approximate representation of the expected cost-to-go functions $\mathcal{Q}^{\gamma}(\cdot)$.

Let $\psi_i^{\gamma}(\cdot)$ be the approximation of the expected cost-to-go function $\mathcal{Q}^{\gamma}(\cdot)$ available at iteration i for macro-stage γ. It is defined by the set of supporting hyperplanes generated until iteration i. We thus have:

$$\psi_i^{\gamma}(u^{\ell}) = \min\{\theta^{\gamma\ell} : \theta^{\gamma\ell} \geq \sum_{r \in \mathcal{R}^{\gamma+1}} \nu_j^{\gamma+1,r} + \mu_j^{\gamma+1,r} u^{\ell} \quad \forall j \in \{1, \ldots, i-1\}\} \tag{5.18}$$

where $\nu_j^{\gamma+1,r}$ and $\mu_j^{\gamma+1,r}$ are the coefficients of the cut generated at iteration $j < i$ by considering realization $r \in \mathcal{R}^{\gamma+1}$. This leads to the following sub-problem $\hat{P}_i^{\gamma}(u_i^m, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r_i^{k,\gamma}})$:

$$\begin{cases} \hat{Q}_i^{\gamma,r_i^{k,\gamma}}(u_i^m) = \min \sum_{n \in \mathcal{X}^{\gamma,r_i^{k,\gamma}}} \rho^n (f^n y^n + h^n s^n \\ \qquad\qquad + g^n x^n) + \sum_{\ell \in \mathfrak{L}(\gamma, r_i^{k,\gamma})} \theta^{\gamma\ell} & (5.19) \\ \theta^{\gamma\ell} \geq \sum_{r \in \mathcal{R}^{\gamma+1}} \nu_j^{\gamma+1,r} + \mu_j^{\gamma+1,r} u^{\ell} & \forall j \in \{1, \ldots, i-1\}, \forall \ell \in \mathfrak{L}(\gamma, r_i^{k,\gamma}) \quad (5.20) \\ \text{Constraints (5.10)} - \text{(5.17)} \text{ for } r = r_i^{k,\gamma} \end{cases}$$

The forward step at iteration i ends when sub-problem $\hat{P}_i^{\gamma}(u_i^m, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r_i^{k,\gamma}})$ has been solved for all sampled scenarios and all macro-stages. Its output is a feasible production plan for all nodes belonging to a sampled scenario. In particular, it provides a value u_i^m for all state variables u^m such that $m \in \omega_i^k \cap \mathcal{V}^{t'(\gamma)}, \gamma \in \mathcal{G}, k = 1, \ldots, K$. These values will be used in the backward step to generate additional cuts and improve the approximation of the expected cost-to-go functions $\mathcal{Q}^{\gamma}(\cdot), \gamma \in \mathcal{G}$.

5.3.4 Backward step

The aim of the backward step is to update the current approximation $\psi_i^\gamma(\cdot)$ of the expected cost-to-go function $\mathcal{Q}^\gamma(\cdot)$ for each macro-stage γ by generating new supporting hyperplanes and obtain a better approximation which is denoted by $\psi_{i+1}^\gamma(\cdot)$.

This step starts from macro-stage Γ and goes back to macro-stage 1. Note that the sub-problems relative to macro-stage Γ do not have any expected future costs, therefore $\psi_i^\Gamma \equiv 0$, for all i. At each macro-stage $\gamma = \Gamma - 1, \dots, 1$, the updating of the approximation of $\mathcal{Q}^\gamma(\cdot)$ is carried out as follows. For each scenario $k = 1, \dots, K$, each node $m \in \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$ and each realization $r \in \mathcal{R}^{\gamma+1}$, we solve a suitable relaxation of $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ and collect the cut coefficients $\{v_i^{\gamma+1,r}, \mu_i^{\gamma+1,r}\}$. These coefficients are then used to generate a new linear inequality of type (5.18) to be added to the current approximation of $\mathcal{Q}^\gamma(\cdot)$. The backward step continues iteratively until the approximation of the expected cost-to-go function at macro-stage $\gamma = 1$ is updated. Since ψ_{i+1}^1 is an under-approximation of the expected cost-to-go function $\mathcal{Q}^1(\cdot)$, the optimal value $\hat{Q}_{i+1}^{1,1}(0)$ of problem $\hat{P}_i^1(0, \psi_{i+1}^1, \mathcal{X}^{1,1})$ provides a lower bound of the optimal value of the stochastic problem.

5.3.5 Cut families

We now briefly recall the three types of cutting planes used in Zou et al. (2019) to improve the approximation of the expected cost-to-go functions during the backward step. Let us consider a macro-stage γ, a scenario index k and the node $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$. Let u_i^m be the value of the state variables u^m in the solution of problem $\hat{P}_i^\gamma(u_i^m, \psi_i^\gamma, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$ solved in the forward step of iteration i. The three following cuts can be added to compute the approximation ψ_i^γ of $\mathcal{Q}^\gamma(\cdot)$.

Integer optimality cut: The algorithm solves problem $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$, for each $r \in \mathcal{R}^{\gamma+1}$, with an updated approximation $\psi_{i+1}^{\gamma+1}$ of $\mathcal{Q}^{\gamma+1}(\cdot)$. Let $v_{i+1}^{\gamma+1,r}$ be its optimal objective value and $\bar{v}_{i+1}^{\gamma+1} = \sum_{r \in \mathcal{R}^{\gamma+1}} v_{i+1}^{\gamma+1,r}$. The integer optimality cut takes the following form:

$$\theta^{\gamma,m} \geq \bar{v}_{i+1}^{\gamma+1} \Big(\sum_{\beta=0}^{B} (u_i^{m,\beta} - 1) u^{m,\beta} + \sum_{\beta=0}^{B} (u^{m,\beta} - 1) u_i^{m,\beta} \Big) + \bar{v}_{i+1}^{\gamma+1}$$

Lagrangian cut: We consider, for each $r \in \mathcal{R}^{\gamma+1}$, the Lagrangian relaxation of problem $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ in which the copy constraints (5.12) are dualized. Each corresponding Lagrangian dual problem is solved to optimality. The generated Lagrangian cut takes the form of inequality (5.18), where $v_i^{\gamma+1,r}$ corresponds to the optimal value of the Lagrangian dual problem and coefficient $\mu_i^{\gamma+1,r,\beta}$ of variable $u^{m,\beta}$ to the optimal value of the Lagrangian multiplier relative to copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$.

Strengthened Benders' cut: We solve, for each $r \in \mathcal{R}^{\gamma+1}$, the linear relaxation of problem $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$. The value of each coefficient $\mu_i^{\gamma+1,r,\beta}$ is set to the dual value of the copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$ in this linear relaxation. The value of $v_i^{\gamma+1,r}$ is obtained by solving the Lagrangian relaxation of problem $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ in which each copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$ is dualized and its Lagrangian multiplier set to $\mu_i^{\gamma+1,r,\beta}$.

5.3.6 Stopping criteria

Two stopping criteria are commonly used for the SDDiP in the literature. The first one is based on a maximum number of consecutive iterations without any improvement of the lower bound, the second one on a maximum total number of iterations.

5.3.7 Summary

As a synthesis, the main steps of the proposed sub-tree-based SDDiP algorithm applied to the stochastic ULS are summarized in Algorithm 1.

Algorithm 1: SDDiP algorithm

Initialize $LB \leftarrow -\infty, UB \leftarrow +\infty, i \leftarrow 1$
while *no stopping criterion is satisfied* **do**
  Sampling step
  Randomly select K scenarios $\Omega_i = \{\omega_i^1, ..., \omega_i^K\}$
  Forward step
  for $k = 1, ..., K$ **do**
    for $\gamma = 1, ..., \Gamma$ **do**
      Solve $\hat{P}_i^\gamma(u_i^m, \psi_i^\gamma, \mathcal{X}^{\gamma, r_i^{k,\gamma}})$ for $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma-1)}$
      Record u_i^ℓ for $\ell = \omega_i^k \cap \mathfrak{L}(\gamma, r_i^{k,\gamma})$
    end
    $C^k \leftarrow \sum_{n \in \omega_i^k}(f^n y_i^n + h^n s_i^n + g^n x_i^n)$
  end
  $C \leftarrow \sum_{k=1}^K C^k$ and $\chi^2 \leftarrow \frac{1}{K-1}\sum_{k=1}^K (C^k - C)^2$
  $UB \leftarrow C + z_{\alpha/2}\frac{\chi}{\sqrt{K}}$
  Backward step
  for $\gamma = \Gamma - 1, ..., 1$ **do**
    for $k = 1, ..., K$ **do**
      Let $m = \omega_i^k \cap \mathcal{V}^{t'(\gamma)}$
      for $r \in \mathcal{R}^{\gamma+1}$ **do**
        Solve the linear relaxation of $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ and collect the coefficients of the strengthened Benders' cut
        Solve the Lagrangian relaxation of $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ and collect the constant value of the strengthened Benders' cut
        Solve $\hat{P}_i^{\gamma+1}(u_i^m, \psi_{i+1}^{\gamma+1}, \mathcal{X}^{\gamma+1,r})$ and collect the coefficients of the Integer optimality cut
        Solve the Lagrangian dual problem and collect the coefficients of the Lagrangian cut
      end
    end
    Add the three generated cuts to ψ_i^γ to get ψ_{i+1}^γ
  end
  $LB \leftarrow \hat{Q}_{i+1}^{1,1}(0)$
  $i \leftarrow i + 1$
end

Note that depending on the partition of $\mathcal{S}$, the number of macro-stages Γ can take any value between 1 and Σ. For $\Gamma = 1$, the forward step corresponds to solving the original problem (5.1)-(5.4) defined on the whole scenario tree $\mathcal{V}$ and no backward step is needed: Algorithm 1 thus directly solves the stochastic problem as a MILP, without any decomposition. For $\Gamma = \Sigma$, Algorithm 1 corresponds to the SDDiP algorithm of Zou et al. (2019). In general, we will have $\Gamma < \Sigma$, which means that the number of expected cost-to-go functions $\mathcal{Q}^\gamma(\cdot)$ to be approximated will be smaller in Algorithm 1 than the one to be handled in the SDDiP algorithm. This may have a positive impact on the global convergence of the algorithm. Namely, with $\Gamma < \Sigma$, each sub-problem $\hat{P}_i^\gamma(\cdot, \psi_i^\gamma, \mathcal{X}^{\gamma,r})$ covers a larger portion of the planning horizon and uses an approximation of its expected future costs which will be globally better as it will rely on a

smaller number of approximate expected cost-to-go functions. As a consequence, the feasible solution obtained by solving $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r})$ at a given iteration of the algorithm will tend to be less myopic and thus to provide lower and upper bounds LB and UB of better quality. However, each sub-problem $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r})$ is now an MILP expressed on a small sub-tree. In particular, the large number of binary variables $u^{\ell,\beta}$ needed to carry out the binary approximation of the leaving inventory at each leaf node $\ell \in \mathfrak{L}(\gamma, r)$ makes its resolution computationally more expensive than the one of a sub-problem expressed on a deterministic scenario involving a single leaf node. In what follows, we thus discuss two algorithmic enhancements aiming at further improving the numerical efficiency of Algorithm 1.

5.4 Algorithmic Enhancements

In this section, we aim at enhancing the numerical efficiency of Algorithm 1, mostly through a more efficient building of the approximation of the expected cost-to-go functions. In what follows, we detail the two proposed algorithmic enhancements.

5.4.1 Approximate sub-tree-based SDDiP

Hjelmeland et al. (2018) and Quezada et al. (2019) both proposed to use an approximate variant of the SDDiP algorithm in which the state variables may be continuous. In this case, the finite convergence of the algorithm is not theoretically guaranteed but, as this approximation leads to a significant reduction of the computational effort required at each iteration of the algorithm, it may positively impact the solution quality in practice. We thus explain in what follows how this approximate SDDiP algorithm can be adapted to the case where a partial sub-tree-based decomposition of the scenario tree is used.

The algorithm is based on a reformulation of problem $P^{\gamma}(s^m, \mathcal{X}^{\gamma,r})$ in which a single auxiliary variable $\tilde{s}^{\xi^{\gamma,r}}$ is introduced. Variable $\breve{s}^{\xi^{\gamma,r}}$ can be seen as a local copy of the inventory variable at the parent node s^m in $P^{\gamma}(s^m, \mathcal{X}^{\gamma,r})$. This results in the following reformulation of $P^{\gamma}(s^m, \mathcal{X}^{\gamma,r})$:

$$
\begin{cases}
Q^{\gamma,r}(s^m) = \min \displaystyle\sum_{n \in \mathcal{X}^{\gamma,r}} \rho^n (f^n y^n + h^n s^n + g^n x^n) & \\
\qquad\qquad + \displaystyle\sum_{\ell \in \mathfrak{L}(\gamma,r)} \mathcal{Q}^{\gamma}(s^{\ell}) & (5.21) \\
s^{\xi^{\gamma,r}} + d^{\xi^{\gamma,r}} = \breve{s}^{\xi^{\gamma,r}} + x^{\tilde{s}^{\xi^{\gamma,r}}} & (5.22) \\
\breve{s}^{\xi^{\gamma,r}} = s^m & (5.23) \\
s^n + d^n = s^{a^n} + x^n \qquad \forall n \in \mathcal{X}^{\gamma,r} \setminus \{\xi^{\gamma,r}\} & (5.24) \\
\text{Constraints (5.10), (5.17)} & (5.25)
\end{cases}
$$

In this reformulation, the expected cost-to-go function $\mathcal{Q}^{\gamma}(s^{\ell}) = \sum_{r' \in \mathcal{R}^{\gamma+1}} Q^{\gamma+1,r'}(s^{\ell})$ is a function of the continuous state variable s^{ℓ}. We thus build an under-approximation of $\mathcal{Q}^{\gamma}(\cdot)$ through a set of linear cuts involving continuous variables s^{ℓ} instead of binary variables $u^{\ell,\beta}$. Let $\breve{\psi}_i^{\gamma}(\cdot)$ be the approximation of the expected cost-to-go function $\mathcal{Q}^{\gamma}(\cdot)$ available at iteration i for macro-stage γ in the approximate SDDiP algorithm. We have:

$$
\breve{\psi}_i^{\gamma}(s^{\ell}) = \min\{\theta^{\gamma\ell} : \theta^{\gamma\ell} \geq \sum_{r' \in \mathcal{R}^{\gamma+1}} (\breve{v}_j^{\gamma+1,r'} + \breve{\mu}_j^{\gamma+1,r'} s^{\ell}) \quad \forall j \in \{1, ..., i-1\}\} \tag{5.26}
$$

where $\breve{v}_j^{\gamma+1,r'}$ and $\breve{\mu}_j^{\gamma+1,r'}$ are the coefficients of the cut generated at iteration $j < i$ by considering realization $r' \in \mathcal{R}^{\gamma+1}$.

This leads to the following approximate sub-problem $\check{P}_i^{\gamma}(s_i^m, \check{\psi}_i^{\gamma}, \mathcal{X}^{\gamma,r})$:

$$\begin{cases} \check{Q}_i^{\gamma,r}(s^m) = \min \sum_{n \in \mathcal{X}^{\gamma,r}} \rho^n (f^n y^n + h^n s^n \\ \qquad\qquad + g^n x^n) + \sum_{\ell \in \mathfrak{L}(\gamma,r)} \theta^{\gamma\ell} & (5.27) \\ \theta^{\gamma\ell} \geq \sum_{r' \in \mathcal{R}^{\gamma+1}} (\check{\nu}_j^{\gamma+1,r'} + \check{\mu}_j^{\gamma+1,r'} s^{\ell}) \qquad \forall j \in \{1,...,i-1\}, \forall \ell \in \mathfrak{L}(\gamma,r) & (5.28) \\ \text{Constraints (5.22)} - \text{(5.25)} & (5.29) \end{cases}$$

In our numerical experiments, only strengthened Benders' cuts are generated in the backward step of the approximate SDDiP algorithm. Moreover, this algorithm is used in an initial phase which is carried out before actually running Algorithm 1. The objective of this initial phase is to build a first under-approximation of each expected cost-to-go functions $\mathcal{Q}^{\gamma}(\cdot)$ and obtain a good initial lower bound for the problem with a reduced computational effort. This initial lower bound will then be further improved during a second phase in which Algorithm 1 is run and all integer optimality, Lagrangian and strengthened Benders' cuts are generated.

5.4.2 Generation of additional strengthened Benders' cuts

The second enhancement of the algorithm seeks to exploit the alternative MILP formulations currently known for SULS to generate additional strengthened Benders' cuts and improve the approximation of the expected cost-to-go functions at a relatively limited computational effort.

Recall that, to generate a strengthened Benders' cut to be added at a given iteration i to the approximation of $\mathcal{Q}^{\gamma-1}(\cdot)$, the algorithm first solves, for each $r \in \mathcal{R}^{\gamma}$, the linear relaxation of problem $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$, where m is a node belonging to $\mathcal{V}^{t'(\gamma-1)}$. This linear relaxation can be computed using different formulations of $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$. A first option is to use the initial MILP formulation defined in Subsection 5.3.3. Other options consist in using the initial MILP formulation only strengthened by the path inequalities proposed by Guan et al. (2009) or the initial MILP formulation strengthened by the path and the tree inequalities proposed by Guan et al. (2009). Then, the algorithm collects the dual value of the copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$ in the linear relaxation and set coefficient $\mu_i^{\gamma,r,\beta}$ to it. The Lagrangian relaxation of problem $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$ in which each copy constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$ is dualized with a Lagrangian multiplier set to $\mu_i^{\gamma,r,\beta}$ is then solved. Its optimal value provides coefficient $\nu_i^{\gamma,r}$.

A key observation here is that the dual value of constraint $\hat{u}^{\xi^{\gamma,r},\beta} = u_i^{m,\beta}$ in the linear relaxation will vary according to the MILP formulation used for $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$. Hence, for a given value of the entering stock described by the binary vector u_i^m, by considering the three alternative MILP formulations available for $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$, it is possible to generate three different strengthened Benders' cuts, i.e. three cuts corresponding to different values of coefficients $\{\nu_i^{\gamma,r}, \mu_i^{\gamma,r}\}$,

We point out here that, in general, there does not seem to be a dominance relationship between these three cuts. In other words, a cut generated using a stronger formulation of $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$ does not necessarily lead to a better approximation of $\mathcal{Q}^{\gamma-1}(\cdot)$. We thus propose an extension of Algorithm 1 in which strengthened Benders' cuts based on the three MILP formulations available for $\hat{P}_i^{\gamma}(u_i^m, \psi_{i+1}^{\gamma}, \mathcal{X}^{\gamma,r})$ are sequentially generated.

Let $\phi_i^{\gamma,r}$ denote the set of path and tree inequalities related to sub-problem $P^{\gamma}(\cdot, \mathcal{X}^{\gamma,r})$ generated until the beginning of iteration i. Let $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r}, Form)$ denote problem $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r})$ expressed using formulation $Form$ where $Form = IF(\emptyset)$ denote the initial formulation without any strengthening and $Form = IF(\phi_i^{\gamma,r})$ the initial formulation strengthened by a set of inequalities $\phi_i^{\gamma,r}$.

We propose the following strategy to sequentially add strengthened Benders' cuts to the approximations of the expected cost-to-go functions. This strategy is based on three increasing levels of formulation strengthening :

- Level $\lambda = 0$: Algorithm 1 is run as described in Section 5.3, i.e. we solve the linear relaxation of sub-problems $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r}, IF(\emptyset))$ to obtain the cut coefficients. The algorithm moves to the next level after a predefined number of consecutive iterations.

- Level $\lambda = 1$: Algorithm 1 is run using the linear relaxation of sub-problems $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r}, IF(\phi_i^{\gamma,r}))$ to obtain the cut coefficients. In this level, at each iteration, only path inequalities are added to $\phi_i^{\gamma,r}$ using a single run of a cutting plane generation procedure based on the separation algorithm presented by Barany et al. (1984). The algorithm moves to the next level after no violated path inequalities have been found during a predefined number of consecutive iterations.

- Level $\lambda = 2$: Algorithm 1 is run using the linear relaxation of sub-problems $\hat{P}_i^{\gamma}(\cdot, \psi_i^{\gamma}, \mathcal{X}^{\gamma,r}, IF(\phi_i^{\gamma,r}))$ to obtain the cut coefficients. In this level, at each iteration, tree inequalities are added to $\phi_i^{\gamma,r}$ using a single run of the cutting plane generation procedure presented in Guan et al. (2009).

Note that, tor the sake of clarity, in Subsection 5.4.2, we focused on explaining how this second algorithmic enhancement is carried out in Algorithm 1. It can, however, be straightforwardly adapted for the approximate version of Algorithm 1 described in Subsection 5.4.1.

In what follows, we will refer to the version of Algorithm 1 in which the two proposed enhancements discussed in this section have been implemented as extSDDiP.

5.5 Computational Experiments

In this section, we focus on assessing the performance of the extSDDiP algorithm proposed in Sections 5.3 and 5.4. This is done by comparing it with the performance of a stand-alone mathematical programming solver ILOG-CPLEX using the extensive MILP formulation (5.1)-(5.4) and the one of the SDDiP algorithm using the dynamic programming reformulation (5.5)-(5.8) with $\mathcal{G} \equiv \mathcal{S}$.

In what follows, we first describe the scheme used to randomly generate instances of the SULS and the experimental setup. We then discuss the results of our computational experiments.

5.5.1 Instance Generation

We randomly generated instances following the same procedure and numerical values as the ones used by Guan et al. (2009). We considered various scenario tree structures, several ratios of the production cost to inventory holding cost and several ratios of the setup cost to the inventory holding cost.

For the sake of brevity, we focus here on a set of 20 large-size instances in which there are $\Sigma = 12$ decision stages, a constant stage length $L = |\mathcal{T}^{\sigma}| = 1$, for all $\sigma \in \mathcal{S}$, and a constant number $R = R^{\sigma} = 3$, for all $\sigma \in \mathcal{S}$, of equiprobable realizations per stage. This leads to a scenario tree involving $|\mathcal{V}| = 265720$ nodes and $\mathcal{L}(1) = 177147$ scenarios. Additional experiments carried out on other sets of instances displaying other scenario tree structures are reported in Quezada et al. (2020b).

5.5.2 Experimental setup

Each instance is first solved with the mathematical programming solver CPLEX 12.8 using the initial MILP formulation (5.1)-(5.4). We use the standard branch-and-cut algorithm of solver CPLEX 12.8 as our preliminary experiments showed that, for the large instances, this was more efficient than using a customized branch-and-cut algorithm based on the path and tree inequalities proposed by Guan et al. (2009). This solution method is denoted by CPX in what follows.

Each instance is then solved by the SDDiP algorithm proposed by Zou et al. (2019) and by the extSDDiP algorithm. For both algorithms, the number of scenarios sampled at each iteration is set to $K = 1$. Moreover, the binary approximation of the continuous state variables s^n is carried out as follows. For each instance, we compute an upper bound of the inventory level at node n as $s_{max} = max_{\ell \in \mathcal{L}(1)} d^{0,\ell}$. The number B of binary variables $u^{n,\beta}$ is set to $B = \lceil log_2(s_{max}) \rceil$. Note that for the instances considered in our numerical experiments, introducing this binary approximation of the inventory variables will not lead to a sub-optimal solution. Namely, the randomly generated demand vectors only comprise integer components and the optimal leaving inventory at each node is known to take an integer value in this case: see Guan and Miller (2008).

Regarding the partition of the set of decision stages $\mathcal{S}$ into macro-stages $\mathcal{G}$, we consider only decompositions in which the number of stages per macro-stage, denoted by G, is constant, i.e. $G = |\mathcal{S}(\gamma)|$, for all $\gamma \in \mathcal{G}$. For each instance, we consider values of $G = \Sigma/\Gamma$ in the set $\{2, 4, 6\}$.

Furthermore, in order to better assess the impact of the two enhancements presented in Section 5.4, several variants of the extSDDiP algorithm are implemented.

First, to evaluate the usefulness of the approximate subtree-based SDDiP algorithm discussed in Subsection 5.4.1, we consider the following implementations of the extSDDiP algorithm:

- extSDDiP-II corresponds to the case where only Algorithm 1 is run.
- extSDDiP-I/II corresponds to a 2-phase algorithm in which phase I first runs the approximate subtree-based SDDiP algorithm to build an initial approximation of the expected cost-to-go functions and phase II runs Algorithm 1 to further improve these approximations.

Second, we also seek to assess the impact on the algorithmic performance of exploiting alternative MILP formulations of the SULS, as presented in Subsection 5.4.2. Each considered setting is described by the maximum level of formulation strengthening λ_{max} used to strengthen the sub-problem formulation. Thus, extSDDiP-I/II-λ_{max} denotes for instance a 2-phase implementation of the extSDDiP in which the sub-problem formulation strengthening levels $0, \ldots, \lambda_{max}$ are sequentially used following the strategy described at the end of Subsection 5.4.2.

Regarding the stopping criteria, the maximum number of consecutive iterations without any improvement of the lower bound LB is set to 30 and the maximum total number of iterations to 1000. The algorithm stops as soon as one of these two conditions is reached. Note that at this point, the upper bound UB is computed considering only $K = 1$ scenario and thus might not be statistically representative. Thus, after the algorithm has stopped, we compute a statistical upper bound based on a larger number of scenarios. We randomly sample $K' = 1000$ scenarios and compute a feasible solution for each of them using the final approximation of the expected cost-to-go functions to evaluate the objective function at each (macro)-stage. We then construct a 95% confidence interval and report the right endpoint of this interval as the statistical upper bound of the optimal value.

Each algorithm was implemented in C++ using the Concert Technology environment. All (mixed-integer) linear programs were solved using CPLEX 12.8 with the default settings and the Lagrangian dual problems were solved by a sub-gradient algorithm. All tests were run

on the computing infrastructure of the Laboratoire d'Informatique de Paris VI (LIP6), which consists of a cluster of Intel Xeon Processors X5690. We set the cluster to use two 3.46GHz cores and 12GB RAM to solve each instance. We impose a time limit of 1800 seconds to method CPX to solve each instance. For the SDDiP and extSDDiP algorithms, we impose a time limit of 900 seconds to compute a lower bound and 900 seconds to compute the true or statistical upper bound.

5.5.3 Results

Table 5.1 displays the numerical results. Column G indicates the number of stages per macro-stage in the partial decomposition of the scenario tree and Column "Method" indicates the algorithm used to solve each instance. Each line in the table thus provides the average results of the indicated resolution method over the 20 considered instances. Column "Gap" displays the gap between the lower bound (LB) and the upper bound (UB) found by each method, i.e. $Gap = |UB - LB|/UB$. The average total computation time in seconds is reported in Column "Time (s)", the average number of iterations in Column "# ite" and the total number of path and tree valid inequalities generated are provided in Column "# VI".

TABLE 5.1: Performance of each method at solving instances with $\Sigma = 12, L = 1, R = 3$ of the SULS problem

G	Method	Gap	Time (s)	# ite	# VI
1	SDDiP	29.12	1,716.06	100	0
2	extSDDiP-II-0	16.32	1,693.26	60	0
	extSDDiP-II-1	8.11	1,618.20	73	138
	extSDDiP-II-2	8.32	1,651.21	72	138
	extSDDiP-I/II-0	16.61	1,742.85	72	0
	extSDDiP-I/II-1	8.65	1,650.81	110	142
	extSDDiP-I/II-2	6.12	1,526.16	175	144
4	extSDDiP-II-0	11.67	1,853.11	13	0
	extSDDiP-II-1	7.76	1,726.47	27	741
	extSDDiP-II-2	7.79	1,712.67	27	740
	extSDDiP-I/II-0	10.46	1,853.51	28	0
	extSDDiP-I/II-1	5.25	1,853.44	47	1,543
	extSDDiP-I/II-2	6.20	1,821.16	72	3,017
6	extSDDiP-II-0	6.18	1,601.00	12	0
	extSDDiP-II-1	6.28	1,760.09	10	7,264
	extSDDiP-II-2	5.51	1,728.80	9	7,134
	extSDDiP-I/II-0	5.28	1,625.98	17	0
	extSDDiP-I/II-1	4.11	1,677.38	21	4,260
	extSDDiP-I/II-2	3.90	1,557.63	29	6,601
12	CPX	53.78	1,803.29	-	0

Results from Table 5.1 first show that method CPX performs rather poorly on the 20 instances considered here as it provides an average gap of 53.78%. We also observe that method SDDiP significantly outperforms method CPX in terms of solution quality as it provides an average gap of 29.12% for these instances.

Furthermore, these results show that algorithm extSDDiP significantly outperforms method SDDiP on these instances. We namely first note that, whatever the partial decomposition and the formulation strengthening setting used, i.e. whatever the value of $G \in \{2, 4, 6\}$ and $\lambda_{max} \in \{0, 1, 2\}$, the gap provided by algorithm extSDDiP-I/II is significantly smaller than the

one provided by algorithm SDDiP. In particular, if we consider the results obtained with algorithm extSDDiP-I/II-2 with a partial decomposition using $G = 6$ stages per macro-stage, we obtain an average gap over the 20 instances of 3.90%. This clearly shows that jointly using the partial decomposition of the scenario tree into sub-trees discussed in Section 5.3 and the two algorithmic enhancements presented in Section 5.4 leads to a significant improvement of the performance of the SDDiP algorithm proposed by Zou et al. (2019).

We now discuss the individual impact of each of these three elements on the performance of algorithm extSDDiP.

The separate impact of the partial decomposition of the scenario tree into sub-trees on the algorithmic performance can be evaluated by looking at the results obtained by algorithm extSDDiP-II-0. We thus observe that the average gap can be reduced from 29.12% with algorithm SDDiP to 11.16% with algorithm extSDDiP-II-0 using a partial decomposition involving $G = 2$ stages per macro-stage and to 6.18% with algorithm extSDDiP-II-0 using a partial decomposition involving $G = 6$ stages per macro-stage. This clearly shows the interest of decreasing the number of expected cost-to-go functions to be approximated: even if the sub-problems to be solved at each stage are larger and as a consequence the number of iterations carried out by the extSDDiP-II-0 is smaller than the one carried out by method SDDiP, the fact that each production plan is built using a less myopic vision of the future costs translates into a better quality of the obtained feasible solution.

We now focus on the impact of the first considered enhancement: the introduction of an initial phase based on an approximate sub-tree-based SDDiP algorithm. This impact can be measured by comparing the results obtained with algorithm extSDDiP-II-0 with the ones obtained with algorithm extSDDiP-I/II-0. We note that, over the three considered values of G, the gap is decreased from 11.39% when using extSDDiP-II-0 to 10.78% when using extSDDiP-I/II-0. The gap reduction obtained through the single use of the approximate version of the algorithm in an initial phase thus seems to be rather limited.

As for the second enhancement, its impact can be evaluated by comparing the results obtained with algorithms extSDDiP-II-1 (or extSDDiP-II-2) with the ones obtained with algorithm extSDDiP-II-0. We observe that, over the three considered values of G, the gap is decreased from 11.39% when using extSDDiP-II-0 to 7.38% when using extSDDiP-II-1 and 7.20% when using extSDDiP-II-2. Thus, using an MILP formulation of the sub-problem strengthened by (ℓ, S) inequalities expressed on paths of the scenario tree to generate more strengthened Benders' cuts has a positive impact on the solution quality. However, using more complex valid inequalities expressed on subtrees does not seem to have a visible impact on this quality.

Finally, we would like to point out that the combined used of these two algorithmic enhancements, i.e. the use of alternative MILP formulations of the SULS to generate additional strengthened Benders' cuts within the approximate extSDDiP algorithm, seems to significantly improve the algorithmic performance. The average gap, over the three considered values of G, is namely decreased from 11.39% when using extSDDiP-II-0 to 5.41% when using extSDDiP-I/II-2.

The results of additional experiments carried out on 120 additional instances are reported in Quezada et al. (2020b). Their conclusion can be briefly summarized as follows:

- The direct resolution of the extensive formulation (method CPX) outperforms algorithms SDDiP and extSDDiP on small instances involving a scenario tree of around 1000 nodes. But for medium to large size instances involving more than 8000 nodes, method CPX is outperformed by these two algorithms.

- On all considered instances, algorithm extSDDiP-I/II-1 significantly outperforms algorithm SDDiP.

- Using a stand-alone partial decomposition (without the algorithmic enhancements discussed in 5.4) provides an improvement in the solution quality provided the obtained sub-problems to be solved at each stage remain of tractable size. In particular, it may not always be worth using a high value of G as it leads to large-size sub-problems and a drastic reduction of the number of iterations carried out by algorithm extSDDiP, which sometimes negatively impacts the solution quality.
- The impact of the two proposed algorithmic enhancements is best seen when they are used in combination, i.e. when the available alternative MILP formulations of the SULS are exploited to generate more strengthened Benders' cuts during phase I of the extSDDiP algorithm.

5.6 Conclusion and perspectives

We investigated a multi-stage stochastic integer programming approach for the SULS problem and focused on the resolution of instances involving large-size scenario trees. We presented a new extension of the SDDiP algorithm proposed by Zou et al. (2019). This new extension is based on three main features: the partial decomposition of the stochastic problem into smaller stochastic sub-problems (rather than into deterministic sub-problems), the introduction of an initial phase in which the state variables are kept continuous and the exploitation of alternative MILP formulations of the stochastic sub-problems to generate additional strengthened Benders' cuts. Computational experiments carried out on randomly generated instances show that the proposed extended algorithm significantly outperforms the original SDDiP algorithm.

An interesting direction for further research could be to extend this work to single-item single-echelon stochastic lot-sizing problems involving complicating features such as the possibility of backlogging the demand, a limited production capacity or upper bounds on the inventory level. These extensions of the SULS problem would comprise a limited number of continuous state variables at each node. Moreover, valid inequalities that could be used to obtain alternative MILP formulations are known for most of them (see e.g. Pochet and Wolsey (2006)). It should thus be possible to adapt the two-phase algorithm extSDDiP-I/II for these problems. It might also be worth investigating whether the proposed extended algorithm could be used to solve multi-item and/or multi-echelon stochastic lot-sizing problems. For such problems, the number of continuous state variables for which a binary approximation would have to be built will be much larger so that the use of a single-phase algorithm extSDDiP-I might be more appropriate. Finally, in many practical settings, assuming a stage-wise independent stochastic process may not be suitable and there may be temporal correlations, in particular in the demand parameter. It would thus be interesting to study how the algorithmic enhancements proposed in the present work for the SDDiP may be exploited to improve the computational efficiency of extensions of the SDDiP dealing with stage-wise dependent stochastic processes such as the one investigated by Philpott and de Matos (2012).

Chapter 6

Multi-stage stochastic lot-sizing with returns

6.1 Introduction

Chapter 5 focused on a multi-stage stochastic programming extension of the simplest available lot-sizing model, i.e. the single-item uncapacitated lot-sizing model or ULS. Clearly, there is a need to study stochastic lot-sizing models with a better practical relevancy to meet the industrial needs in terms of production planning. The present chapter can be seen as a first step towards closing this gap. It namely focuses on an application of multi-stage stochastic programming in a production planning field where uncertainty is particularly present: the remanufacturing of used products. This work was carried out as part of the work of Franco Quezada, a PhD student which I have been co-supervising with Safia Kedad-Sidhoum since 2018, and was recently published as a journal paper: see Quezada et al. (2020b).

More precisely, we study here a remanufacturing system which involves three key processes: disassembly of used products brought back by customers, refurbishing of the recovered parts and reassembly into like-new finished products. This system can be seen as an extension of the multi-echelon system presented in Subsection 2.2.3 to the case where the product structure is not serial but rather comprises a disassembly and an assembly component. We aim at optimizing the production planning for this three-echelon system over a multi-period horizon. In this context, production planning includes making decisions on how much and when used products should be disassembled, refurbished or reassembled in order to build new or like-new products. The main objective is to meet customers' demand for the remanufactured products in the most cost-effective way.

As compared to classical manufacturing systems which produce end-products from virgin raw materials and new components, remanufacturing systems involve several complicating characteristics, among which is a high level of uncertainty in the input data needed to make planning decisions. This is mainly due to a lack of control on the return flows of used products, both in terms of quantity and quality, and to the difficulty of forecasting the demand for remanufactured products. The fact that production planning and control activities are more complex for remanufacturing firms due to uncertainties is extensively discussed e.g. in Guide et al. (1999) and Guide (2000).

We thus investigate a production planning model in which uncertainties related to the quantity and quality of returned products, the customers' demand, and the costs are simultaneously taken into account and seek to develop an approach explicitly considering the multi-stage aspect of the decision making process in production planning.

The contributions of the present work are twofold. Firstly, we propose a multi-stage stochastic integer programming approach for a stochastic lot-sizing problem arising in a multi-echelon multi-item remanufacturing system. This is in contrast with most previously published works which either consider two-stage stochastic programming approaches for complex remanufacturing systems or multi-stage stochastic programming approaches for single-echelon and/or

single-item systems. Second, we propose a branch-and-cut framework to solve the resulting large-size mixed integer linear program. The algorithm relies on a new set of valid inequalities obtained by mixing previously known path inequalities Loparic et al. (2001). The number of these valid inequalities increases exponentially fast with the size of the scenario tree. We provide an efficient cutting-plane generation strategy to identify the useful subset of this class. Our computational experiments show that the proposed method is capable of significantly decreasing the computation time needed to obtain guaranteed optimal solutions.

The remaining part of this chapter is organized as follows. Section 6.2 formally describes the problem and proposes a mixed integer linear programming model. In Section 6.3, a reformulation of the problem based on the echelon-stock concept is presented. This reformulation allows us to identify a series of single-echelon subproblems embedded in the general multi-echelon problem. Section 6.4 introduces a new class of valid inequalities to strengthen the linear relaxation of each single-echelon subproblem. Cutting-plane generation algorithms are developed in Section 6.5. Section 6.6 reports the results of computational experiments and discusses the performance of our branch-and-cut algorithm. Finally, Section 6.7 gives the conclusions with possible directions for further research.

6.2 Problem description and mathematical formulation

6.2.1 System description

We consider a remanufacturing system comprising three main production echelons (see Figure 6.1): disassembly, refurbishing and reassembly, and seek to plan the production activities in this system over a multi-period horizon. We assume that there is a single type of used product which, in each period, is returned in limited quantity by customers. These used products are first disassembled into parts. Due to the usage state of the used products, some of these parts are not recoverable and have to be discarded during disassembly. In order to reflect the variations in the quality of the used products, the yield of the disassembly process, i.e. the proportion of parts which will be recoverable, is assumed to be part-dependent and time-dependent. The remaining recoverable parts are then refurbished on dedicated refurbishing processes. The serviceable parts obtained after refurbishing are reassembled into remanufactured products which have the same bill-of-material as the used products. These remanufactured products are used to satisfy the dynamic demand of customers.

All the production processes are assumed to be uncapacitated. However, the system might not be able to satisfy the customer demand on time due to part shortages if there are not enough used products returned by customers or if their quality is low. In this situation, the corresponding demand is lost incurring a high penalty cost to account for the loss of customer goodwill. Moreover, note that some used products are allowed to be discarded before being disassembled: this option might be useful in case more used products are returned that what is needed to satisfy the demand for remanufactured products. Similarly, some of the recoverable parts obtained from the disassembly process may be discarded. In case there is a strong unbalance between the part-dependent disassembly yields, this option might be used in a production plan to avoid an unnecessary accumulation in inventory of the easy-to-recover parts.

We aim at finding an optimal production plan, i.e. a production plan complying with all the practical limitations of the system while minimizing the total production cost. This cost comprises the production fixed setup costs to be incurred each time a production takes place on a process, the inventory holding costs for all the items involved in the system, the lost-sales costs penalizing the unsatisfied demand and the disposal costs for the discarded used products and parts.

Ahn et al. (2011) studied a deterministic and particular case of the problem, in which the quantity of returned products is unlimited and the lost sales and the discarding quantities are

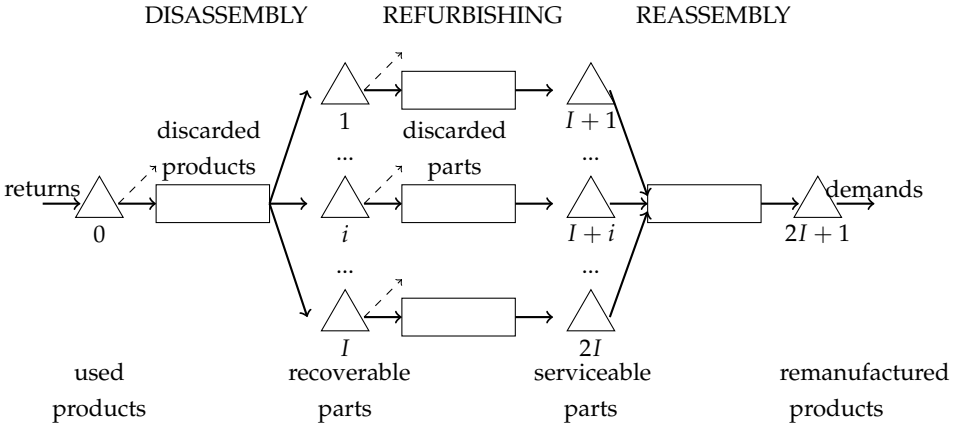

FIGURE 6.1: Illustration of studied remanufacturing system

assumed to be zero. The authors proved that, under these assumptions, the problem is NP-hard. Therefore, our problem is NP-hard as well.

6.2.2 Uncertainty

As mentioned in the introduction of this chapter, one of the main challenges to be faced when planning remanufacturing activities is the high level of uncertainty in the problem parameters. In what follows, we propose a production planning model in which all problem parameters, except the bill-of-material coefficients, are subject to uncertainty.

We consider a multi-stage decision process corresponding to the case where the value of the uncertain parameters unfolds little by little following a discrete-time stochastic process and the production decisions are adapted progressively as more and more information is collected. This leads to the representation of the uncertainty via a scenario tree. With a slight abuse of notation, we will refer to this scenario tree by mentioning only its set of nodes $\mathcal{V}$. Each node $n \in \mathcal{V}$ corresponds to a single planning period t^n and belongs to a single decision stage $\sigma^n \in \mathcal{S}$. It represents the state of the system that can be distinguished by the information unfolded up to that period t^n.

Each node n has a unique predecessor node denoted a^n belonging to time period $t^n - 1$. By convention, the root node of the scenario tree is indexed by 1 and a^1 is set to 0. At any non-leaf node of the tree, one or several branches indicate future possible outcomes of the random variables from the current node. Let $\mathcal{C}(n)$ be the set of immediate children of node n, $\mathcal{V}(n)$ the sub-tree of $\mathcal{V}$ rooted in n and $\mathcal{L}(n)$ the set of leaf nodes belonging to $\mathcal{V}(n)$. The probability associated with the state represented by node n is denoted by ρ^n. A scenario is defined as a path from the root node to a leaf node in the scenario tree and represents a possible outcome of the stochastic input parameters over the whole planning horizon. The set of nodes on the path from node n to node m is denoted by $\mathcal{P}(n, m)$.

We use the following notations for the problem formulation:

- I: number of part types involved in a returned / remanufactured product,
- $\mathcal{I} = \{0, ..., 2I + 1\}$: set of all items involved in the system where:
  - $i = 0$ corresponds to the returned product,
  - $\mathcal{I}_r = \{1, ..., I\}$ is the set of recoverable parts provided by the disassembly process,
  - $\mathcal{I}_s = \{I + 1, ..., 2I\}$ is the set of serviceable parts provided by the refurbishing processes,

  - $i = 2I + 1$ corresponds to the remanufactured product.

- $\mathcal{J} = \{0, ..., I + 1\}$: set of production processes, , where $j = 0$ corresponds to the disassembly process, $j = 1, ..., I$ correspond to the refurbishing processes and $j = I + 1$ corresponds to the reassembly process.,

The number of parts $i \in \mathcal{I}_r \cup \mathcal{I}_s$ embedded in a returned or remanufactured product, denoted by α_i, is assumed to be deterministic. All other input parameters are considered as stochastic and defined as follows:

- d^n: customers' demand at node $n \in \mathcal{V}$,
- r^n: quantity of used products (returns) collected at node $n \in \mathcal{V}$,
- p_i^n: proportion of recoverable parts $i \in \mathcal{I}_r$ obtained by disassembling one unit of returned product at node $n \in \mathcal{V}$,
- λ^n: unit lost-sales penalty cost at node $n \in \mathcal{V}$,
- f_j^n: setup cost for process $j \in \mathcal{J}$ at node $n \in \mathcal{V}$,
- h_i^n: unit inventory cost for part $i \in \mathcal{I}$ at node $n \in \mathcal{V}$,
- δ_i^n: unit cost for discarding item $i \in \mathcal{I}_r \cup \{0\}$ at node $n \in \mathcal{V}$,
- g^n: cost for discarding the unrecoverable parts obtained while disassembling one unit of returned product at node $n \in \mathcal{V}$.

Moreover, we assume that at each stage, the realization of the random parameters happens before we have to make a decision for this stage, i.e. we assume that the values of d^n, r^n, p_i^n, λ^n, f_j^n, h_i^n, δ_i^n and g^n are known before we have to decide on the production plan at node $n \in \mathcal{V}$. We also assume that $\lambda^n \gg g^n$ for all $n \in \mathcal{V}$.

6.2.3 MILP formulation

We propose a multi-stage stochastic integer programming model based on the uncertainty representation described above. The decision variables involved in the model are:

- x_j^n: quantity processed by process $j \in \mathcal{J}$ at node $n \in \mathcal{V}$,
- $y_j^n \in \{0, 1\}$: setup variable for process $j \in \mathcal{J}$ at node $n \in \mathcal{V}$,
- s_i^n: inventory level of part $i \in \mathcal{I}$ at node $n \in \mathcal{V}$,
- q_i^n: quantity of part $i \in \mathcal{I}_r \cup \{0\}$ discarded at node $n \in \mathcal{V}$,
- l^n: lost sales of remanufactured products at node $n \in \mathcal{V}$.

The mixed integer linear programming model is given below.

$$
\begin{cases}
Z^* = min \sum_{n\in\mathcal{V}} \rho^n \Big(\sum_{j\in\mathcal{J}} f_j^n y_j^n + \sum_{i\in\mathcal{I}} h_i^n s_i^n + \lambda^n l^n \\
\qquad\qquad + \sum_{i\in\mathcal{I}_r\cup\{0\}} \delta_i^n q_i^n + g^n x_0^n \Big) & & (6.1)\\
x_j^n \leq M_j^n y_j^n & \forall j \in \mathcal{J}, \forall n \in \mathcal{V} & (6.2)\\
s_0^n = s_0^{a^n} + r^n - x_0^n - q_0^n & \forall n \in \mathcal{V} & (6.3)\\
s_i^n = s_i^{a^n} + p_i^n \alpha_i x_0^n - x_i^n - q_i^n & \forall i \in \mathcal{I}_r, \forall n \in \mathcal{V} & (6.4)\\
s_i^n = s_i^{a^n} + x_{i-I}^n - \alpha_i x_{I+1}^n & \forall i \in \mathcal{I}_s, \forall n \in \mathcal{V} & (6.5)\\
s_{2I+1}^n = s_{2I+1}^{a^n} + x_{I+1}^n - d^n + l^n & \forall n \in \mathcal{V} & (6.6)\\
s_i^0 = 0 & \forall i \in \mathcal{I} & (6.7)\\
s_i^n \geq 0 & \forall i \in \mathcal{I}, \forall n \in \mathcal{V} & (6.8)\\
q_i^n \geq 0 & \forall i \in \mathcal{I}^r \cup \{0\}, \forall n \in \mathcal{V} & (6.9)\\
l^n \geq 0 & \forall n \in \mathcal{V} & (6.10)\\
x_j^n \geq 0, y_j^n \in \{0,1\} & \forall j \in \mathcal{J}, \forall n \in \mathcal{V} & (6.11)
\end{cases}
$$

The objective function (6.1) aims at minimizing the expected total cost, over all nodes of the scenario tree. This cost is the sum of the expected setup, inventory holding, lost sales and disposal costs. Constraints (6.2) link the production quantity variables to the setup variables. Note that the value of constant M_j^n can be set by using an upper bound on the quantity that can be processed on process j at node n. This quantity is limited by two elements: the availability of the used products already returned by customers and the future demand for remanufactured products. The reader is refered to Quezada et al. (2020b) for more detail about how to determine the value of this constant. Constraints (6.3)-(6.6) are the inventory balance constraints. Constraints (6.3) (resp. (6.4) and (6.5)) involve a term corresponding to a dependent demand x_0^n (resp. x_i^n and $\alpha_i x_{I+1}^n$) whereas Constraints (6.6) only involve an independent demand term d^n. Without loss of generality, we assume that the initial inventories are all set to 0: see Constraints (6.7). Finally, Constraints (6.8)-(6.11) provide the domain of the decision variables.

Even if the problem (6.1)-(6.11) is a mixed-integer linear program displaying a structure similar to the one of its deterministic counterpart, its resolution by a mathematical programming solver poses some computational difficulties in practice. This first comes from the problem size which, for a given planning horizon length, is much larger in the stochastic case than in the deterministic case. Namely, in the stochastic case, the mixed-integer linear programming formulation involves $\mathcal{O}(|\mathcal{V}||\mathcal{J}|)$ binary variables, $\mathcal{O}(|\mathcal{V}||\mathcal{I}|)$ continuous variables and $\mathcal{O}(|\mathcal{V}||\mathcal{I}|)$ constraints. The size of the scenario tree $|\mathcal{V}|$ is in $\mathcal{O}(R^{S+1})$ where R is the number of children per node and S the number of stages. The MILP formulation size thus grows exponentially fast with the number of decision stages S. Moreover, the presence of the big-M type constraints (6.2) leads to a poor quality of the lower bounds provided by the linear relaxation of the problem.

In what follows, we propose a branch-and-cut algorithm in order to solve to optimality medium-size instances of the problem. We first describe a reformulation of the problem that provides a way to decompose the multi-echelon problem into a series of single-echelon subproblems. We then investigate two sets of valid inequalities (path inequalities and tree inequalities) that can be used to strengthen the formulation of each of these single-echelon subproblems. These valid inequalities are added to the problem formulation using a cutting-plane strategy during the course of the branch-and-bound search.

6.3 Mathematical reformulation

The concept of echelon stock has been widely used to develop solution approaches for multi-echelon lot-sizing problems (the reader is referred to Pochet and Wolsey (2006) for further details). The main advantages of the reformulation is that it helps decomposing the multi-echelon problem into a series of single-echelon lot-sizing problems for which formulation strengthening techniques such as valid inequalities or extended reformulations are available. As each subproblem is a relaxed version of the overall multi-echelon problem, valid inequalities strengthening the linear relaxation of each subproblem will strengthen the linear relaxation of the overall multi-echelon problem.

6.3.1 Echelon stock reformulation

The echelon demand ed_i^n for an intermediate product can be understood as the translation of the external demand for the finished product into an independent demand for the intermediate product. For each product $i \in \mathcal{I}^r \cup \mathcal{I}^s$, we straightforwardly define the echelon demand as $ed_i^n = \alpha_i d^n$. We note however that, in our case, it is not possible to properly define such an echelon demand for the used product $i = 0$. Namely, this demand could be defined as $ed_0^n = \frac{d^n}{min_{i \in \mathcal{I}_r} p_i^n}$ by considering that the amount of used product to disassemble to satisfy the external demand d^n is determined by the disasssembly yield of the item $i \in \mathcal{I}_r$ which is the most difficult to recover at node n. However, as the disassembly yields are time-varying and stochastic, the actual amount of used product needed to satisfy the external demand d^n depends on the period in which it is disassembled and might be larger or smaller than $\frac{d^n}{min_{i \in \mathcal{I}_r} p_i^n}$. Hence, using the echelon demand ed_0^n might lead to inconsistent disassembly production decisions. We thus focus in what follows on defining echelon stock variables for products $i \in \mathcal{I}^r \cup \mathcal{I}^s$.

The echelon stock of a product in a multi-echelon production system corresponds to the total quantity of the product held in inventory, either as such or as a component within its successors in the bill-of-material. For each product $i \in \{1, ..., 2I+1\}$, we define the echelon inventory variables as follows:

- $es_i^n = s_i^n + es_{I+i}^n = s_i^n + s_{I+i}^n + \alpha_i s_{2I+1}^n$, for $i \in \mathcal{I}_r$, for $n \in \mathcal{V}$
- $es_i^n = s_i^n + \alpha_i es_{2I+1}^n = s_i^n + \alpha_i s_{2I+1}^n$, for $i \in \mathcal{I}_s$, for $n \in \mathcal{V}$
- $es_{2I+1}^n = s_{2I+1}^n$, for $n \in \mathcal{V}$

Moreover, we define the unit echelon inventory holding cost eh_i^n as follows:

- $eh_i^n = h_i^n$, for $i \in \mathcal{I}_s$, for $n \in \mathcal{V}$
- $eh_i^n = h_i^n - h_{i-I}^n$, for $i \in \mathcal{I}_r$, for $n \in \mathcal{V}$
- $eh_{2I+1}^n = h_{2I+1}^n - \sum_{i \in \mathcal{I}_r} \alpha_i h_i^n$, for $n \in \mathcal{V}$

This leads to the following mixed-integer linear programming formulation:

$$
\begin{cases}
Z^* = min \sum_{n\in\mathcal{V}} \rho^n \Big(\sum_{j\in\mathcal{J}} f_j^n Y_j^n + h_0^n s_0^n + \sum_{i\in\mathcal{I}\setminus\{0\}} eh_i^n es_i^n + \lambda^n l^n \\
\qquad\qquad + \sum_{i\in\mathcal{I}_r\cup\{0\}} \delta_i^n q_i^n + g^n x_0^n \Big) & & (6.12)\\
x_j^n \leq M_j^n Y_j^n & \forall j\in\mathcal{J}, \forall n\in\mathcal{V} & (6.13)\\
s_0^n = s_0^{a^n} + r^n - x_0^n - q_0^n & \forall n\in\mathcal{V} & (6.14)\\
es_i^n = es_i^{a^n} + p_i^n \alpha_i x_0^n - \alpha_i d^n + \alpha_i l^n - q_i^n & \forall i\in\mathcal{I}_r, \forall n\in\mathcal{V} & (6.15)\\
es_i^n = es_i^{a^n} + x_{i-I}^n - \alpha_i d^n + \alpha_i l^n & \forall i\in\mathcal{I}_s, \forall n\in\mathcal{V} & (6.16)\\
es_{2I+1}^n = es_{2I+1}^{a^n} + x_{I+1}^n - d^n + l^n & \forall n\in\mathcal{V} & (6.17)\\
s_0^0 = 0 & & (6.18)\\
es_i^0 = 0 & \forall i\in\mathcal{I}\setminus\{0\} & (6.19)\\
s_0^n \geq 0 & \forall n\in\mathcal{V} & (6.20)\\
es_i^n - es_{I+i}^n \geq 0 & \forall i\in\mathcal{I}_r, \forall n\in\mathcal{V} & (6.21)\\
es_i^n - \alpha_i es_{2I+1}^n \geq 0 & \forall i\in\mathcal{I}_s, \forall n\in\mathcal{V} & (6.22)\\
es_{2I+1}^n \geq 0 & \forall n\in\mathcal{V} & (6.23)\\
s_0^n \geq 0, l^n \geq 0 & \forall n\in\mathcal{V} & (6.24)\\
x_j^n \geq 0, y_j^n \in \{0,1\} & \forall p\in\mathcal{J}, \forall n\in\mathcal{V} & (6.25)
\end{cases}
$$

As in the previous formulation, the objective function (6.12) aims at minimizing the expected cost, over all nodes of the scenario tree. Constraints (6.13) are defined as Constraints (6.2) of the natural formulation. Constraints (6.14)-(6.17) are inventory balance constraints. Constraints (6.14) use the classical inventory variables, whereas Constraints (6.15)-(6.17) make use of the echelon inventory variables. Contrary to Constraints (6.4)-(6.5) of the natural formulation, Constraints (6.15)-(6.17) do not involve a dependent demand term but only an external demand term. Constraints (6.21)-(6.23) ensure consistency between the echelon inventory at the different levels of the bill-of-material and guarantee that the physical inventory of each product remains non-negative. Finally, Constraints (6.24)-(6.25) define the domain of the decision variables.

6.3.2 Single echelon subproblems

The introduction of echelon inventory variables leads to the elimination of the dependent demand term in the inventory balance equations of the (6.1)-(6.11) formulation. This induces that the constraint matrix of (6.12)-(6.25) displays a specific structure: it can be decomposed in a series of single-echelon single-resource lot-sizing subproblems coupled by the linking constraints (6.21)-(6.23).

For the sake of brevity, we will focus in what follows on the sub-problems relative to the refurbishing and reassembly processes. The reader is referred to Quezada et al. (2020b) for a detailed description of the sub-problems relative to the disassembly process.

For each process $j = 1...I+1$, we thus have the following single-echelon subproblem:

$$\begin{cases} Z_p^* = min \sum_{n\in\mathcal{V}} \rho^n \left(f_j^n y_j^n + eh_{j+I}^n es_{j+I}^n + \lambda^n l^n \right) & (6.26) \\ x_j^n \leq M_j^n y_j^n & \forall n \in \mathcal{V} \quad (6.27) \\ es_{j+I}^n = es_{j+I}^{a^n} + x_j^n - \alpha_j d^n + \alpha_j l^n & \forall n \in \mathcal{V} \quad (6.28) \\ es_{j+I}^0 = 0 & (6.29) \\ es_{j+I}^n \geq 0 & \forall n \in \mathcal{V} \quad (6.30) \\ l^n \geq 0, x_j^n \geq 0, y_j^n \in \{0,1\} & \forall n \in \mathcal{V} \quad (6.31) \end{cases}$$

Each subproblem (6.26)-(6.31) is an uncapacitated single-echelon single-item lot-sizing problem with lost sales. The deterministic variant of this problem was studied by Loparic et al. (2001) who proposed a family of valid inequalities called $(k,\mathcal{U})$ inequalities, to strengthen the linear relaxation. We discuss in Section 6.4 how these inequalities known for the deterministic variant of the problem can be used to solve the stochastic problem expressed on a scenario tree.

6.4 Valid inequalities

In this section, we provide $(k,\mathcal{U})$ inequalities for each single-echelon subproblem described in Section 6.3. We first exploit these $(k,\mathcal{U})$ inequalities considering their application to each individual scenario, i.e. to each individual path from a non-terminal node n to a leaf node $\ell \in \mathcal{L}(n)$. Next, we extend them to a more general class of inequalities. This is done by exploiting the scheme proposed by Guan et al. (2009) for generic multi-stage stochastic integer programs. The idea is to mix valid inequalities corresponding to different individual scenarios to obtain valid inequalities for the whole scenario tree (or for a subtree). Throughout this section we will refer to a $(k,\mathcal{U})$ inequality applied to an individual scenario as a *path inequality* and to a (k,U) inequality applied to a subtree as a *tree inequality*.

6.4.1 Path inequalities

We first introduce some additional notation. Let n be a non-leaf node in $\mathcal{V}$ and $\ell \in \mathcal{L}(n)$ a leaf node reachable from n. $succ_n^\ell \in \mathcal{C}(n)$ denotes the immediate successor of node n belonging to the path $\mathcal{P}(n,\ell)$, i.e. $succ_n^\ell = \mathcal{C}(n) \cap \mathcal{P}(n,\ell)$. Let $\mathcal{U}_{n,\ell} \subseteq \mathcal{P}(succ_n^\ell, \ell)$ be a subset of nodes belonging to the path from $succ_n^\ell$ to ℓ, not necessarily consecutive.

For each process $j \in \{1...I+1\}$, we have the following proposition:

Proposition 2. *Let $n \in \mathcal{V}$ and $\ell \in \mathcal{L}(n)$. Let $\mathcal{U}_{n,\ell} \subseteq \mathcal{P}(succ_n^\ell, \ell)$. The following inequality*

$$es_{j+I}^n \geq \alpha_j \sum_{m\in\mathcal{U}_{n,\ell}} \left[d^m \left(1 - \sum_{v\in\mathcal{P}(succ_n^\ell,\ell)} y_j^v \right) - l^m \right] \quad (6.32)$$

is valid for the problem (6.12)-(6.25).

The proof is direct following the proof of Loparic et al. (2001).

The intuition underlying these path inequalities can be understood as follows. We consider the inventory level of the product $j+I$ at node n and look for the future demands for this product in the path $\mathcal{P}(succ_n^\ell, \ell)$ linking node n to leaf node ℓ. For a node $m \in \mathcal{U}_{n,\ell}$, if $\sum_{v\in\mathcal{P}(succ_n^\ell,\ell)} y_j^v \geq 1$, the demand of node m, $\alpha_j d^m$, can be satisfied by a production in one of the nodes $v \in \mathcal{P}(succ_n^\ell, \ell)$ and does not have to be in stock when leaving node n. But if $\sum_{v\in\mathcal{P}(succ_n^\ell,\ell)} y_j^v = 0$, the demand $\alpha_j d^m$ cannot be produced in any node $v \in \mathcal{P}(succ_n^\ell, \ell)$ meaning

that the portion of this demand which will be satisfied by a production, $\alpha_j(d^m - l^m)$, should already be in stock at node n.

6.4.2 Tree inequalities

We now investigate a new family of valid inequalities obtained by considering a subtree of the scenario tree as proposed by Guan et al. (2006) and Guan et al. (2009). The authors proposed a general scheme to obtain valid inequalities for multi-stage stochastic integer programs by mixing several path inequalities. In what follows, we apply this scheme to derive a new set of tree inequalities based on a mixing of the path inequalities discussed above. We first introduce some additional notation to properly define this new set of valid inequalities. Let $\mathcal{U} = \cup_{\ell \in \mathcal{L}(n)} \mathcal{U}_{n,\ell}$ be a set of nodes defining a tree inequality. This enables us to introduce the following proposition for each process $j \in \{1, ..., I+1\}$.

Proposition 3. *Let $n \in \mathcal{V}$ and $\mathcal{U} \subset \mathcal{V}(n)$. Let $\mathcal{s} = \{\mathcal{s}_1, ..., \mathcal{s}_{|\mathcal{L}(n)|}\}$ be a sequence of leaf nodes belonging to $\mathcal{L}(n)$ in increasing order of cumulative demand $\sum_{m \in \mathcal{U}_{n,\ell}} d^m$: $\sum_{m \in \mathcal{U}_{n,\mathcal{s}_1}} d^m \leq ... \leq \sum_{m \in \mathcal{U}_{n,\mathcal{s}_\omega}} d^m \leq ... \leq \sum_{m \in \mathcal{U}_{n,\mathcal{s}_{|\mathcal{L}(n)|}}} d^m$. We set $\sum_{m \in \mathcal{U}_{n,\mathcal{s}_0}} d^m = 0$. The following inequality*

$$es^n_{j+I} + \alpha_j \sum_{m \in \mathcal{U}} l^m + \alpha_j \sum_{\nu \in \mathcal{V}(n) \setminus \{n\}} \phi^\nu y^\nu_j \geq \alpha_j \sum_{m \in \mathcal{U}_{n,\mathcal{s}_{|\mathcal{L}(n)|}}} d^m \tag{6.33}$$

is valid for problem (6.12)-(6.25), with

$$\phi^\nu = \min \Big\{ \max_{\ell \in \mathcal{L}(\nu)} \{ \sum_{m \in \mathcal{U}_{a^\nu,\ell}} d^m \}, \sum_{\omega = 1...|\mathcal{L}(n)| \text{ s.t } \mathcal{s}_\omega \in \mathcal{L}(\nu)} \Big(\sum_{m \in \mathcal{U}_{n,\mathcal{s}_\omega}} d^m - \sum_{m \in \mathcal{U}_{n,\mathcal{s}_{\omega-1}}} d^m \Big) \Big\}$$

The proof is based on Theorem 2 in Guan et al. (2009): see Quezada et al. (2020b) for more detail about it.

6.5 Cutting-plane generation

The number of valid inequalities (6.32) and (6.33) is too large to allow adding all of them a priori to the formulation. Hence, a cutting-plane generation strategy is needed to add only a subset of these valid inequalities into the MILP formulation. Consequently, the corresponding separation problems must be solved in order to identify the inequalities to be incorporated in the formulation. In what follows, we discuss an exact separation algorithm for the path inequalities and a heuristic one for the tree inequalities. These separation algorithms form the basis of a cutting-plane generation procedure aiming at strengthening the linear relaxation of the problem (6.12)-(6.25).

6.5.1 Path inequalities

Given a solution $(\hat{x}, \hat{y}, \hat{es}, \hat{I})$ of the linear relaxation of the single-echelon subproblem corresponding to process j, solving the separation problem for path inequalities consists in finding the most violated inequality (6.32) if it exists or proving that no such inequality exists.

For a given process j, node $n \in \mathcal{V}$ and leaf node $\ell \in \mathcal{L}(n)$, finding the most violated inequality corresponds to identifying the set $\mathcal{U}_{n,\ell}$ maximizing the right-hand side of the inequality. We note that the value of the term corresponding to a node $m \in \mathcal{U}_{n,\ell}$ in the right-hand side of (6.32) does not depend on the other nodes belonging to $\mathcal{U}_{n,\ell}$. Hence, each node of $\mathcal{P}(succ^\ell_n, \ell)$ can be considered individually: if it has a positive contribution in maximizing the right-hand side value of the inequality, we add it to set $\mathcal{U}_{n,\ell}$, if not, it is discarded. The set $\mathcal{U}_{n,\ell}$ is thus built by

adding all the nodes m in $\mathcal{P}(succ_n^\ell, \ell)$ such that $d^m\left(1 - \sum_{\nu \in \mathcal{P}(succ_n^\ell, \ell)} \hat{y}_j^\nu\right) - \hat{l}^m > 0$. We underline that the above strategy can be implemented in polynomial time: see Loparic et al. (2001). The proposed separation algorithm namely runs in $\mathcal{O}(P^2)$, where P corresponds to the number of nodes in the set $\mathcal{P}(succ_n^\ell, \ell)$.

However, even if inequalities (6.32) can be separated in polynomial time, our preliminary computational experiments showed that a simple cutting-plane generation strategy in which all the violated inequalities found at each iteration are added to the formulation led to the introduction of a too large number of additional constraints and a loss of computational efficiency of the branch-and-bound search tree. Moreover, many of these inequalities involved the same subsets of setup variables y_j^n and had thus similar effects in terms of strengthening the relaxation of the problem.

In order to limit the increase in the formulation size, we propose a cutting plane generation strategy to add violated path inequalities to the formulation. This strategy relies on two main ideas.

The first idea consists in adding, for a given process j and node $n \in \mathcal{V}$, at most one valid inequality at each iteration of the cutting-plane generation, namely the inequality corresponding to the leaf node $\ell \in \mathcal{L}(n)$ providing the largest violation of the path inequality, i.e. to the leaf node $\ell_{min} = \text{argmin}\left\{\ell \in \mathcal{L}(n), \hat{es}_{j+I}^n - \alpha_j \sum_{m \in \mathcal{U}_{n,\ell}} \left[d^m\left(1 - \sum_{\nu \in \mathcal{P}(succ_n^\ell, \ell)} \hat{y}_j^\nu\right) - \hat{l}^m\right]\right\}$.

The second idea aims at avoiding the addition of inequalities involving similar subsets of setup variables y_j^n during an iteration of the cutting-plane generation algorithm. This is achieved by using the following strategy. During a given iteration of the algorithm, each time a violated inequality is added to the formulation, we record ν_{min}, the last node of the path $\mathcal{P}(succ_n^{\ell_{min}}, \ell_{min})$ added to the set $\mathcal{U}_{n,\ell_{min}}$. The inequality added to the formulation involves a subset of setup variables y_j^ν corresponding to nodes ν belonging to the path $\mathcal{P}(succ_n^{\ell_{min}}, \nu_{min})$. As the valid inequalities generated when considering the leaf node ℓ_{min} at nodes $m \in \mathcal{P}(succ_n^{\ell_{min}}, \nu_{min})$ are likely to involve the same setup variables y_j^ν and have a redundant effect on the formulation strengthening, we do not consider generating them during the current iteration. Thus, if a cut involving leaf node ℓ_{min} is generated at node n at a given iteration, for all $m \in \mathcal{P}(n, \nu_{min})$, ℓ_{min} is removed temporarily, i.e. for the course of the current iteration, from the leaf node set $\mathcal{L}(m)$ considered for the search of violated valid inequalities at node m. It is then reintegrated into all leaf node sets at the beginning of the next iteration.

Note that this cutting-plane generation strategy implies that all valid inequalities are still potentially considered for inclusion in the formulation and that the separation problem is solved exactly.

6.5.2 Tree inequalities

Given a non-leaf node n, solving the separation problem for inequalities (6.33) requires to identify a subset of nodes $\mathcal{U} \subseteq \mathcal{V}(n)$ minimizing the difference between the left-hand side and the right-hand side of (6.33). This is challenging as contrary to the case of path inequalities, it is not possible to consider each node of $\mathcal{V}(n)$ individually. Namely, selecting a node m of $\mathcal{V}(n)$ in the set $\mathcal{U}$ not only changes the left-hand side of the inequality by a quantity $l^m + \phi^m y_j^m$, but also potentially impacts the value of the coefficient ϕ^ν for all other nodes $\nu \in \mathcal{V}(n) \setminus \{n\}$. In addition, selecting a node m of $\mathcal{V}(n)$ potentially changes the order of the sequence s and hence the value of the right-hand side of the inequality. These interactions significantly complicate the resolution of the separation problem.

Therefore, we consider a heuristic separation approach based on a neighborhood search to solve the separation problem for the tree inequalities. For each process j and each node $n \in \mathcal{V}$, the heuristic algorithm first builds an initial set $\mathcal{U}$ containing all nodes in $\mathcal{V}(n)$ that would be

selected in the sets $\mathcal{U}_{n,\ell}, \ell \in \mathcal{L}(n)$, when looking for a violated path inequality. If this initial set is empty, we stop. Otherwise, we try to find a tree inequality as violated as possible by removing, one by one, some nodes from $\mathcal{U}$. More precisely, we start by computing the amount of violation obtained with the initial set $\mathcal{U}$: this requires to determine the ordering λ of the leaf nodes in $\mathcal{L}(n)$ corresponding to set $\mathcal{U}$ and to compute coefficients ϕ^ν for every $\nu \in \mathcal{V}(n)$. We then explore the neighborhood of set $\mathcal{U}$ which consists of all subsets of $\mathcal{U}$ obtained by removing a single node. For each considered neighbor, we compute the amount of violation of the corresponding tree inequality: this operation is particularly time-consuming due to the fact that the ordering λ of the leaf nodes and the coefficients ϕ need to be recomputed for each neighbor. A first improvement strategy is used to explore the neighborhood of the current set, i.e. we update the current set $\mathcal{U}$ as soon as a better neighbor set $\mathcal{U}'$ is found. Finally, the algorithm stops when no neighbor set $\mathcal{U}'$ has a violation value lower than the one of the current set $\mathcal{U}$.

6.6 Computational experiments

We develop two branch-and-cut algorithms for solving problem (6.1)-(6.11). These algorithms rely on the cutting-plane generation algorithms proposed in Section 6.5 to add valid inequalities to the echelon stock reformulation (6.12)-(6.25) discussed in Section 6.3. We provide in this section the results of computational experiments carried out on randomly generated instances of the problem. The main objective of these experiments is to assess the effectiveness of the branch-and-cut algorithms by comparing them with the one of a stand-alone mathematical programming solver.

In what follows, we introduce the setting used to randomly generate instances based on the data presented in Ahn et al. (2011) and Jayaraman (2006) before discussing the detailed results of our computational experiments.

6.6.1 Instances

A detailed description of the instance generation is available in Quezada et al. (2020b). We thus provide here the main information:

- The demands for finished products d^n, the bill of material coefficient α_i, the set-up and inventory holding costs were randomly generated based on the instance generation scheme provided in Ahn et al. (2011).
- The number of parts in a finished product is set to $I = 10$.
- The proportion of recoverable parts p_i^n, $i \in \mathcal{I}_r$, obtained by disassembling one unit of used product at node $n \in \mathcal{V}$, is randomly generated following a uniform distribution. Three intervals, corresponding to three quality levels, are defined based on the values presented in the case study reported by Jayaraman (2006).
- The volume of returns r^n is also randomly generated following a uniform distribution for which three different intervals are considered.
- The lost-sales unit penalty cost λ^n is set to 10000.
- The cost for discarding one unit of recoverable part, $i \in \mathcal{I}_r \cup \{0\}$, is set to $\delta_i^n = h_i^n * \frac{T}{\beta}$, with β following a discrete uniform distribution over $[2, T]$.
- The cost for discarding the unrecoverable parts generated during the disassembly process is computed as $g^n = \sum_{i=1}^{I} \delta_i^n (1 - p_i^n)\alpha_i$.

Regarding the scenario tree, we consider 18 alternative structures which differ with respect to the number of stages, the number of periods per stage and the number of immediate successors R of each last-period-of-stage node. This leads to scenario trees involving between 126 and 1365 nodes. For each scenario tree structure, used product quality level and used product quantity level, we randomly generated 10 instances, resulting in a total of 1620 instances.

6.6.2 Results

Each instance was solved using the echelon stock formulation (6.12)-(6.25) discussed in Section 6.3 by three alternative branch-and-cut methods:

1. The standard branch-and-cut algorithm embedded in the mathematical programming solver CPLEX with the solver default settings.

2. BC1: a customized branch-and-cut algorithm using only path inequalities at the root node of the branch-and-bound search tree.

3. BC2: a customized branch-and-cut algorithm in which path and tree inequalities are added at the root node of the branch-and-bound search tree and UserConstraints callbacks are used to add tree valid inequalities to the formulation during the course of the branch-and-bound search tree.

All related linear programs and mixed-integer linear program were solved by CPLEX 12.8 with the solver default settings. Note that for a better understanding of the behavior of the branch-and-cut methods, the automatic generation of cuts by default CPLEX was turned off when running BC1 and BC2. The algorithms were implemented in C++ using the Concert Technology environment. All tests were run on the computing infrastructure of the Laboratoire d'Informatique de Paris VI (LIP6), which consists in a cluster of Intel Xeon Processors X5690. We set the cluster to use two 3.46GHz cores and 12GB RAM to solve each instance. We imposed a time limit of 900 seconds.

The corresponding results are displayed in Table 6.1 in which instances are grouped according to the number of nodes in the scenario tree. For each set of 90 instances and each solution method, we report five performance measures:

1. Gap_{LP} is the average percentage integrality gap. It is computed as the relative difference between the lower bound provided by the linear relaxation of the formulation and the value of the optimal integer solution. In case the instance could not be solved to optimality, the value of the best integer feasible solution found is used.

2. Gap_{MIP} is the average percentage residual gap reported by CPLEX. It is computed as the relative difference between the best lower bound and the best integer feasible solution found by the solver within the time limit.

3. *Time* is the average CPU time (in seconds) needed to find a guaranteed optimal integer solution (we used the value of 900s in case a guaranteed optimal integer solution could not be found within the computation time limit).

4. *#Opt* is the number of instances solved to optimality within the time limit.

5. *Cuts* reports the average number of cuts added to the formulation.

We first note from the results displayed in Table 6.1 that the proposed branch-and-cut methods BC1 and BC2 do not perform better than default CPLEX for the sets of small instances involving less than 682 nodes: they namely lead to larger values of the residual gap Gap_{MIP}. Nonetheless, for the large instances involving more than 728 nodes, the proposed methods

significantly outperform default CPLEX by providing much smaller residual gaps. Thus, the average residual gap over the corresponding 540 larger instances is decreased from 7.43% when using default CPLEX to to 1.16% when using method BC1 and 0.98% when using method BC2.

6.7 Conclusion and perspectives

We studied an uncapacitated multi-item multi-echelon lot-sizing problem within a remanufacturing system involving three production echelons: disassembly, refurbishing and reassembly. We considered a stochastic environment in which the input data of the optimization problem are subject to uncertainty and proposed a multi-stage stochastic integer programming approach relying on scenario trees to represent the uncertain information structure. This resulted in the formulation of a large-size mixed-integer linear program involving a series of big-M type constraints. We developed a branch-and-cut algorithm in order to solve the obtained MILP to optimality. This algorithm relies on a new set of tree inequalities obtained by combining valid inequalities previously known for each individual scenario of the scenario tree. The tree inequalities are used within a cutting-plane generation procedure based on a heuristic resolution of the corresponding separation problem. Computational experiments carried out on randomly generated instances show that the proposed branch-and-cut algorithm performs well on medium-size instances as compared to the use of a stand-alone mathematical solver.

However, neither CPLEX solver nor the proposed branch-and-cut algorithms seem to be able to solve instances involving very large-size scenario trees. Hence, an interesting direction for further research could be to study how a nested decomposition approach such as the one investigated in Chapter 5 might be used to solve this problem. A first attempt relying on the SDDiP, i.e. using a full decomposition of the problem into deterministic sub-problems, has been published as a conference paper: see Quezada et al. (2019). The use of the extSDDiP exploiting a partial decomposition of the problem into small stochastic sub-problems is currently under investigation.

TABLE 6.1: Comparison between default CPLEX configuration and the customized branch-and-cut algorithms.

Instances	CPLEX default				BC1 (Path)					BC2 (Path and Tree)				
Nodes	Gap_{LP}	Gap_{MIP}	Time	#Opt	Gap_{LP}	Gap_{MIP}	Time	#Opt	Cuts	Gap_{LP}	Gap_{MIP}	Time	#Opt	Cuts
126	7.71	0.14	871.23	7	1.40	0.23	900.63	0	2042	1.24	0.15	866.06	8	2289
189	9.56	0.27	895.13	1	1.22	0.32	900.66	0	3778	1.09	0.24	892.70	2	4030
242	6.81	0.26	900.37	0	1.48	0.49	900.54	0	3057	1.29	0.36	900.60	0	3539
254	8.04	0.36	900.18	0	1.37	0.52	900.66	0	4184	1.21	0.40	900.68	0	4653
255	6.15	0.24	897.71	1	1.66	0.67	900.68	0	2761	1.40	0.44	900.63	0	3900
255	7.45	0.27	900.13	0	1.33	0.46	900.66	0	3801	1.19	0.36	900.56	0	4119
363	8.81	0.46	906.08	0	1.45	0.74	900.67	0	5984	1.31	0.62	900.64	0	6182
381	8.86	0.41	906.39	0	1.09	0.60	900.75	0	7699	0.97	0.50	900.75	0	7985
468	7.62	0.51	900.20	0	1.51	0.85	900.78	0	6528	1.40	0.71	900.70	0	6682
510	8.34	0.79	900.24	0	1.35	0.88	900.71	0	8527	1.21	0.75	900.62	0	9082
511	6.52	0.44	900.78	0	1.69	0.96	900.77	0	5545	1.47	0.78	900.64	0	6741
682	6.74	0.66	900.25	0	1.61	0.94	900.85	0	7509	1.50	0.87	900.75	0	7904
728	10.01	3.20	903.11	0	1.63	0.98	900.87	0	9253	1.47	0.86	900.98	0	9826
765	14.01	5.47	900.65	0	1.53	1.19	901.11	0	15549	1.08	0.73	901.13	0	16121
777	9.93	3.51	905.34	0	1.47	0.88	900.86	0	10369	1.40	0.83	900.89	0	10578
1022	18.13	10.33	900.42	0	2.96	2.61	900.99	0	17129	1.55	1.22	901.53	0	18279
1093	13.45	8.45	900.48	0	2.46	1.72	900.98	0	7950	1.94	1.24	900.82	0	9831
1365	17.90	13.59	900.51	0	2.26	1.52	900.80	0	7690	1.74	0.98	900.89	0	9479

Part II

Facility Location

Chapter 7

Design of an outbound logistics network

In parallel to the research on lot-sizing problems described in the first part of this manuscript, I had the opportunity to investigate other combinatorial optimization problems, mainly through the participation in collaborative research projects and the co-supervision of PhD students. The second part of this document is devoted to the presentation of two pieces of this work, which deal with applied facility location problems. In both cases, one of the main challenges we had to face consisted in reaching a good trade-off between the need to built a detailed mathematical model of the real optimization problem in order to obtain practically relevant solutions and the necessity of keeping the obtained mathematical program small enough to be solvable with a reasonable numerical effort.

The present chapter presents a location-allocation problem to design the distribution network of the car maker Renault. We seek to determine where to locate distribution centers to optimize the distribution of cars in France from the assembly plants (or import ports) to the car dealers while taking into account a series of operational constraints. This work was carried out within the PhD thesis of Mouna Kchaou-Boujelben, which I co-supervized between 2010 and 2013 with Michel Minoux. This thesis was funded by an industrial research agreement (CIFRE) between Renault and the French National Agency for Research and Technology (ANRT).

7.1 Introduction

Modern cars are complex technological products involving a large number of mechanical and electronic sub–components. Accordingly, the automotive industry uses a large variety of production units (forge, foundries, mechanics, assembly, etc) but the car manufacturer outsources many of these activities to its suppliers. The resulting supply chain network is thus particularly complex due to the introduction of many levels of suppliers (1st tier, 2nd tier, 3rd tier and even more) in addition to assembly plants, logistical compounds and customers.

The work presented here focuses on the outbound logistics of Renault which consists of the flows of finished cars from the assembly plants to the car dealers. The whole outbound distribution process is mainly split into two sub–processes: primary transport from plants to distribution centers (DCs) and secondary transport from distribution centers to car dealers. One of the main advantages of using distribution centers is the consolidation of flows in order to make the best possible use of transport capacities. The main volume routes are from plants to distribution centers as these flows correspond to the aggregation of many customer demands transiting through intermediate DCs. This is why high–capacity modes of transport such as vessels and trains could be used, especially when manufacturing sites are scattered over several countries. These modes are namely less expensive and polluting than transportation by truck, making them an attractive option for long distance, high volume flows. Once arrived at distribution centers, cars are not stored but only held for a short transit time (typically a few

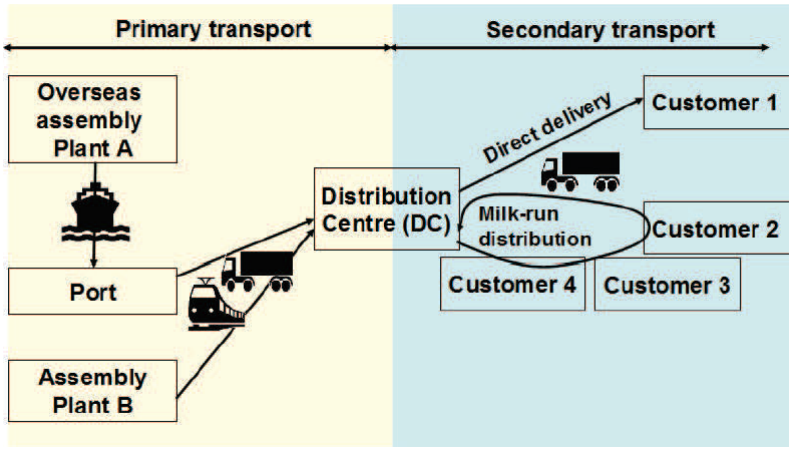

FIGURE 7.1: Overview of the outbound logistics of Renault

days) before being sent to car dealers. In this second step (secondary transport), only trucks are used to deliver cars to car dealers as the corresponding transport routes are short and usually in urban areas. See Figure 7.1 for an overview of the outbound distribution process.

In the present work, we limit our scope to the portion of the distribution network located in France. Thus, in case the transport from an overseas assembly plant to a distribution center involves maritime shipping and transshipments, we consider the import port in France as the sourcing point instead of the assembly plant. Moreover, a preliminary analysis showed that none of the potential transport links in France could meet the necessary minimum volume requirements to allow using train as transport mode. We thus focus on transportation by truck.

A distinctive feature of car distribution comes from the fact that cars are expensive, fragile and bulky products that have to be transported by dedicated trucks with limited capacities. Typically, a truck can carry up to 8 Renault Clio or 10 Renault Twingo. Dealing with voluminous products results in the fact that load efficiency is a key parameter in order to minimize transportation costs in car distribution. Thus, making the best possible use of transport capacities and in particular ensuring full truckload transport is one of the priorities of automotive outbound logistics.

Ensuring transportation between two points of the distribution network via full truckload would not be a major issue if it was possible to let the cars wait on a parking facility as long as needed to full the trucks. However, as there is a maximum delivery time to comply with, this policy is not feasible in practice. Thus, in order to ensure transportation by full truckloads while complying with the maximum delivery time, it is important to consolidate enough volume on each opened transport link. For primary transport from plants to distribution centers, this involves reducing the number of transport links starting at a given plant. However, for secondary transport from distribution centers to car dealers, this is not always possible. The demand of some car dealers could be indeed below the threshold corresponding to reaching a full truckload within the maximum waiting time allowed at a distribution center. This is why it is necessary to group deliveries: a given truck starting from a distribution center may have to visit two or three customers before coming back to the distribution centre.

7.2 Problem description

We study a three-level multi-product distribution network. This network consists in a set $\mathcal{S} = \{1, ..., S\}$ of sourcing points (assembly plants or import ports), a set $\mathcal{D} = \{1, ..., D\}$ of potential

candidate sites to locate a distribution center (DC) and a set $\mathcal{C} = \{1, ..., C\}$ of car dealers. We assume that the number and location of the sourcing points as well as the number and location of car dealers are fixed.

Let $\mathcal{P} = \{1, ..., P\}$ denote the set of products (or car types) that should be distributed using this logistics network. The annual demand of each car dealer for each car type is assumed to be already assigned to a sourcing plant. This is indeed a strategic decision made by high management and not at the distribution level. Demands are thus expressed as quantities $q_{p,s,c}$ of a given car type p to be distributed from a given source s to a given destination c in the network.

Our main concern is to locate DCs by selecting a subset of previously identified candidate location sites, to assign car dealers to DCs and to route the flows of cars in this transportation network such that the total distribution costs are minimized. This optimization problem can thus be seen as a location-allocation problem. It however significantly differs from the basic variant of this problem as multiple operational constraints have to be taken into account in the problem modeling to ensure that the proposed transportation network will meet the real-life requirements.

These operational constraints can be described as follows.

1. *Minimum volume constraints on primary transport links*. The total volume of cars to be transported on each opened transportation link between a sourcing point and a DC should be above a minimum value ensuring that it will be possible to carry out the transportation through full truckload without letting cars wait at the sourcing point more than the maximum allowed time (one week in practice). Let V_s^{min} be the minimum annual volume of cars to be transported on any primary link opened between sourcing point s and DC d and PTC_{sd} be the unit cost per car transported by full truckload between sourcing point s and DC d.

2. *Minimum volume constraints on each secondary transport route*. In many cases, transporting cars from DCs to car dealers through direct deliveries is not possible as the weekly volume of cars requested by a car dealer is too low to allow delivery by full truckload once a week. This is why the secondary transport is done through the use of distribution routes starting at a DC, successively visiting several car dealers and coming back to the DC. Each route should be built such that the total volume of cars to be distributed to the car dealers it visits is above a minimum value ensuring that it will be possible to carry out the transportation through full truckload at least once a week. Let W^{min} be the minimum annual volume of cars to be transported on any opened secondary route.

3. *Maximum covering distance constraints*. Deliveries from DCs to car dealers are carried out by drivers that have to come back to the DC at the end of each working day and are not allowed to drive longer than the legal daily driving time. This translates into the fact that the distance traveled on a secondary transport route should not be above a given limit denoted by L^{max}.

4. *Single sourcing restrictions*: In order to facilitate day-do-day operations, it is requested that each car dealer always receives all the car types coming from the same sourcing point through the same DC.

5. *Minimum and maximum volume constraints at each DC*: In practice, the management of the DCs is outsourced to logistics suppliers. In this context, locating a DC on a candidate location site does not mean that we actually build a DC but only that we contract with a logistics supplier to use its already existing DC. This is why there is no fixed opening costs in this problem. Instead, there is a unit transit cost TC_d to be paid for each car transiting

through a DC: this unit cost applies as long as the total annual volume of cars transiting through the DC stays within a predefined interval denoted by $[T_d^{min}, T_d^{max}]$.

7.3 Solution approach

As such, the problem described in Subsection 7.2 is an integrated location-routing problem, i.e. a problem in which we simultaneously determine the location of facilities and the route from these facilities to serve the demand points. However, the size of the instances coming from our case study exceeds by far the current numerical tractability of exact solution approaches for location-routing problems. We thus propose a two-phase heuristic solution approach.

7.3.1 Clustering of demand points

The first phase consists in building clusters of car dealers in order to get a good approximation of the secondary routing costs while keeping a manageable size for the facility location problem. These clusters should group together a small number of car dealers which are geographically close to one another and whose total demand is large enough to ensure delivery by full truckload once or twice a week. While building the list of potential clusters, we thus take into account three constraints: a maximum number of car dealers per cluster, a maximum distance between car dealers belonging to the same cluster and a total annual demand (for all products in $\mathcal{P}$) of the car dealers in the cluster above a minimum value. We propose to solve this clustering problem either exactly as a set partitioning problem or approximately using a dedicated heuristic. Let $\mathcal{K} = \{1, .., K\}$ be the set of selected clusters and $\mathcal{C}_k \subset \mathcal{C}$ be the set of car dealers belonging to cluster k.

Once the set of clusters is determined, the best delivery route from each candidate location site d to each cluster k is built by solving a small traveling salesman problem. If the length $L_{d,q}$ of this route is larger than the maximum covering distance L^{max}, the corresponding DC-cluster assignment is forbidden. Otherwise, we multiply this length by the transportation cost per kilometer and use the result as an estimation of the unit cost $STC_{d,k}$ to transport a car by full truckload from the DC d to cluster k.

Finally, recall that we have single sourcing restrictions for the car dealers. These restrictions impose that, for a given car dealer c, the set of car types it will receive from the same sourcing point s transits through the same DC d. This enables us to aggregate the demand of all car dealers $c \in \mathcal{C}_k$ for all products they should receive from the same sourcing point into a single demand denoted by $\gamma_{s,k} = \sum_{c \in \mathcal{C}_k} \sum_{p \in \mathcal{P}} q_{p,s,c}$.

7.3.2 Location-allocation problem

Mixed-integer linear programming formulation

The second phase aims at solving the initial facility location in which the demand points are not anymore the individual car dealers (served through routes) but the clusters of car dealers (served through direct deliveries) built in the first phase.

In order to formulate the problem, we introduce the following binary variables:

- $y_d = 1$ if DC d is selected in the solution, 0 otherwise;
- $x_{s,d,k} = 1$ if cluster k receives the products coming from sourcing point s through DC d, 0 otherwise;
- $a_{s,d} = 1$ if transportation link between sourcing point s and DC d is used, 0 otherwise;
- $b_{d,k} = 1$ if transportation route between DC d and cluster k is used, 0 otherwise.

Using these variables, the location-allocation problem can be formulated as follows.

$$
\begin{cases}
\min \sum_{s\in\mathcal{S}} \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} [PTC_{sd} + TC_d + STC_{d,k}]\gamma_{s,k} x_{s,d,k} & & (7.1)\\
\sum_{d\in\mathcal{D}} x_{s,d,k} = 1 & \forall s \in \mathcal{S}, \forall k \in \mathcal{K} & (7.2)\\
\sum_{k\in\mathcal{K}} \gamma_{s,k} x_{s,d,k} \geq V_s^{min} a_{sd} & \forall s \in \mathcal{S}, \forall d \in \mathcal{D} & (7.3)\\
\sum_{k\in\mathcal{K}} x_{s,d,k} \leq K a_{sd} & \forall s \in \mathcal{S}, \forall d \in \mathcal{D} & (7.4)\\
\sum_{s\in\mathcal{S}} \sum_{k\in\mathcal{K}} \gamma_{s,k} x_{s,d,k} \geq T_d^{min} y_d & \forall d \in \mathcal{D} & (7.5)\\
\sum_{s\in\mathcal{S}} \sum_{k\in\mathcal{K}} \gamma_{s,k} x_{s,d,k} \leq T_d^{max} y_d & \forall d \in \mathcal{D} & (7.6)\\
\sum_{s\in\mathcal{S}} \gamma_{s,k} x_{s,d,k} \geq W^{min} b_{d,k} & \forall d \in \mathcal{D}, \forall k \in \mathcal{K} & (7.7)\\
\sum_{s\in\mathcal{S}} x_{s,d,k} \leq S b_{d,k} & \forall d \in \mathcal{D}, \forall k \in \mathcal{K} & (7.8)\\
b_{d,k} = 0 & \forall (d,k) \text{ s.t. } L_{d,k} > L^{max} & (7.9)\\
y_d \in \{0,1\}, x_{s,d,k} \in \{0,1\}, a_{s,d} \in \{0,1\}, b_{d,k} \in \{0,1\} & \forall s \in \mathcal{S}, \forall d \in \mathcal{D}, \forall k \in \mathcal{K} & (7.10)
\end{cases}
$$

The objective function (7.1) seeks to minimize the total distribution costs which consist in the primary transportation costs, the transit costs and the secondary transportation costs. Constraints (7.2) ensure that the demand of each cluster k for the product types available at each sourcing point s is satisfied and that all the corresponding cars transit through a single DC. Constraints (7.3)-(7.4) impose the minimum volume requirements on each opened primary transporation link: if link (s,d) is opened, i.e. if a_{sd} is set to 1, the total amount of cars transported on this link must be above V_s^{min}. Similarly, Constraints (7.5)-(7.6) ensure that the total volume transiting through each opened DC lies within the prescribed interval and Constraints (7.7)-(7.8) impose the minimum volume requirements for each opened secondary route. Finally, Constraints (7.9) guarantee that the selected secondary transportation routes do not exceed the maximum covering distance of each DC.

Heuristic resolution of the mixed-integer linear program

Problem (7.1)-(7.10) is a binary program which can be solved directly by a mathematical programming solver for small instances. However, the computational time needed to solve to optimality the instances corresponding to our industrial case study was very long (more than 4 days of computation on a personal computer) in some cases. This is mainly explained by the large number of binary variables involved in the model: this one was namely up to 62800 for the largest instances considered in our numerical experiments.

We thus develop a MILP-based heuristic. The idea behind the proposed algorithm is to exploit as much as possible the information provided by the optimal solution of the linear relaxation of Problem (7.1)-(7.10). An argument supporting this approach is the tightness of the lower bounds provided by this linear relaxation: the average integrality gap on the studied instances was namely around 1.3%. The proposed heuristic comprises two main steps:

1. The algorithm first solves a relaxed version of Problem (7.1)-(7.10) in which only the location variables y are kept binary and all the other variables are relaxed as continuous variables in $[0,1]$. The solution of this relaxed problem is used to decide which candidate location sites should be selected.

2. The location of the DCs being given, the second step aims at determining the best possible DC-cluster assignment while complying with the minimum volume constraints both at the DCs and on each opened transportation link. A first possibility consists in solving again Problem (7.1)-(7.10) in which the value of each location variables y is now fixed to the value obtained during the previous step, variables a, b and x corresponding to closed DCs are set to 0 and the integrality constraints on the remaining a, b and x are reintroduced. The resulting binary program is easier to solve than the initial problem as the number of binary variables is significantly reduced. In order to further reduce the computation time, we also develop a 'fixing heuristic' which, based on the solution obtained during the first step of the algorithm, seeks to fix the value of as many assignment variables x as possible before reintroducing the integrality constraints on a, b and x and solving the obtained reduced-size MILP. We propose several ways of carrying out this variable fixing without creating infeasibility with respect to the minimum volume constraints. The reader is referred to Kchaou Boujelben et al. (2014) for a detailed description of this algorithm.

7.4 Computational experiments

We carried out numerical experiments based on the data coming from our case study. This one involves $S = 16$ sourcing points, $D = 51$ candidate sites, $C = 448$ car dealers and $P = 35$ car types.

We first run the clustering phase using a maximum of 3 car dealers per cluster, a maximum distance of 80km between car dealers belonging to the same cluster and a minimum annual demand allowing 2 full truckload deliveries per week. This resulted in the creation of around 60000 potential clusters, amongst which $K = 302$ were selected by the clustering algorithm. We then solved Problem (7.1)-(7.10) either exactly with a mathematical programming solver or heuristically with one of the algorithms described above. The reader is referred to Kchaou Boujelben et al. (2014) for an in-depth discussion about the algorithmic performance of the proposed heuristic solution approach. In a nutshell, our results showed that the implemented heuristic methods are able to provide good quality solutions within short computation times on instances for which a state-of-the-art MIP solver does not produce any feasible solution.

Regarding the case study, we noted that the network structure suggested by our solution displayed a rather high number of opened DCs. It namely chose to open a DC in 28 out of the 51 available candidate sites. This means that secondary transport considerably influences the network configuration. Namely, the maximum covering distance constraint imposes to open many DCs in order to be close to car dealers. Furthermore, the cost per car per kilometer for secondary transport is higher than the cost for primary transport, due to the difference of truck speed (secondary transport usually concerns last-mile deliveries in urban areas where trucks are slower). As customers are scattered all over the country, we try to get close to them by opening many DCs. Nonetheless, it is not possible to open a DC at each potential location as this may violate minimum throughput constraints for DCs. In our solution, the throughput of many opened DCs is indeed close to the minimum required quantity. This shows the impact of minimum throughput constraints on the network configuration. Finally, we observed that the number of DCs assigned to each cluster is relatively small (between 1 and 5 DCs per cluster) and that 34% of the clusters are served by a single DC from which they receive all their demand.

7.5 Conclusion and perspectives

We considered a multi-product distribution network design problem arising from a case-study in the automotive industry. Based on realistic assumptions, we introduced minimum volume,

maximum covering distance and single sourcing constraints, making the problem difficult to solve for large-size instances. We thus developed several heuristic procedures using various relaxations of the original MIP formulation of the problem. In our numerical experiments, we analyzed the structure of the obtained network. This work was published as a journal paper: see Kchaou Boujelben et al. (2014). A multi-period extension seeking to take into account the annual seasonality of car demand was also published as a journal paper: see Kchaou-Boujelben et al. (2016).

However, the model investigated here assume that all input parameters, in particular the future demand for cars and the transportation costs, are deterministic. This is a rather strong assumption in practice. The development of a stochastic programming approach taking into account the uncertainty on these parameters may thus be an interesting direction for further research.

Chapter 8

Optimal placement of virtual network functions for cybersecurity

The present chapter investigates a set covering problem arising in the context of cybersecurity in telecommunication networks. The problem aims at optimally placing virtual network functions in a 5G network in order to counter an on-going distributed denial-or-service attack and prevent the hackers from causing damages to their target.

The work presented in this section was carried out between 2017 and 2020 through a collaboration with Sonia Vanier, assistant professor at the University Paris I, and Kahina Lazri, research engineer in cybersecurity at Orange R&D. This project also involved a master student, Alexandros Papadimitriou, who did a 6-months internship at Orange R&D center in 2019.

8.1 Introduction

Distributed Denial of Service (DDoS) attacks are among the top threats to network operators and internet service providers (ISPs). A distributed denial of service is a type of cyberattack in which multiple compromised computer systems attack a target, such as a server or a website, and cause a denial of service for its legitimate users. The flood of incoming messages, connection requests or malformed packets exhausts the resources of the target and forces it to slow down or even shut down, thereby preventing it to provide service to its legitimate users.

DDoS attacks can be very damaging for the organization they target. For instance, a survey carried out in 2017 by the cybersecurity company Kapersky Lab estimated the average cost of a DDoS attack for large (1000+) businesses to be around $2.3 millions: see Berard (2018). This cost mainly comprises the cost incurred in fighting the attack and restoring service, the investment in an offline or back-up system while online services are unavailable, the loss of revenue or business opportunities and the loss of trust from customers and partners.

Many DDoS mitigation solutions have been proposed to protect organizations' networks, servers and services. The traditional approach consists in deploying specialized hardware security appliances that are fixed in terms of strength, functionality and capacity. This means in particular that the location and capacity (in terms of the volume of malicious traffic it can process) of the defense appliances are determined in advance, before the DDoS attacks actually take place. As explained e.g. by Fayaz et al. (2015), companies are thus forced to over provision by deploying appliances capable of handling a high but predefined volume of attack at several points in the network.

Network Function Virtualization (NFV) is a recent network architecture concept in which network functions (e.g. network address translation, firewalling, domain name service, etc.) are implemented as software and deployed as virtual machines running on general purpose

commodity hardware (Jakaria et al., 2016). NFV offers new possibilities to counter DDoS attacks. In particular, its flexibility and reactivity allows to postpone the DDoS defense deployment after the attack is detected. The defense mechanisms can therefore be placed where they are needed and their number can be adapted to the scale of the attack (Fayaz et al., 2015).

NFV is thus a promising technology to mitigate DDoS attacks. However, in order to fully leverage its potential, some difficulties should be overcome. First, virtual network functions (VNFs) are instantiated on virtual machines. These virtual machines consume the limited computing resources (CPU, memory,...) of the servers on which they run. When designing an NFV-based infrastructure to counter an on-going DDoS attack in a network, these limitations in the available computing resources should be taken into account. The number of VNFs which can be instantiated at each node of the network depends on the resources of the servers located at this node. Second, each VNF has a limited filtering capacity and can thus remove only part of the attack flow. The filtering capacity of a VNF corresponds to the maximum amount of malicious flow an instance of this VNF can stop. If the malicious flow going through a VNF is larger than its filtering capacity, the excess malicious flow is forwarded in the network and may thus reach its target. This translates into the fact that, in order to stop all the malicious traffic of an attack, several VNFs may have to be placed at different nodes on the paths used to route the flow between its source and its target. A carefully optimized VNF placement strategy taking into account both the limited computing resources in the network and the limited filtering capacity of a VNF is thus needed.

In the present work, we focus on the deployment of an architecture based on the NFV technology to secure a network against DDoS attacks. We assume that the on-going attack has been detected and that its ingress points, its volume and its target have been identified. Based on this information, we seek to determine the optimal number and location of VNFs in order to remove all the illegitimate traffic while trying to minimize the total cost of the activated VNFs.

We take here the perspective of an internet service provider (ISP) aiming at providing a DDoS mitigation service to its customers in a 5G network. Among the key features of 5G networks is network slicing: see e.g. Vyakaranam and Krishna (2018). Network slicing is an architecture in which the physical network infrastructure managed by an ISP is partitioned into multiple virtual independent networks termed slices. Each slice is an isolated end-to-end network which is lent by the ISP to a single customer. A slice is adapted to meet the specific requirements of its customer in terms of quality of service (bandwidth, reliability, latency, etc.). Network slicing thus provides an opportunity to the ISP to flexibly configure its physical network so as to simultaneously fulfill quality-of-service requirements that may strongly vary from one customer to the next. However, on each slice of the network, the routing of the flow will not be managed anymore by the ISP but by its customer which will rely on its own proprietary routing algorithms. This significantly enhances the difficulty for the ISP of providing a DDoS mitigation service as it will not control the exact routing of the malicious flow that needs to be stopped.

In what follows, we present a robust optimization (RO) model to optimally design an NFV-based DDoS mitigation infrastructure in the context of 5G network slicing. This model explicitly takes into account the fact that the ISP is not aware of the exact routing of the attack flow. This is done by considering the malicious flow routing as an input parameter of the optimization problem which is subject to uncertainty. To the best of our knowledge, this is the first time such a robust optimization model is investigated to design a DDoS mitigation infrastructure in 5G networks

8.2 Problem description

8.2.1 Problem definition

The network topology is modeled by a digraph $G = (\mathcal{N}, \mathcal{L})$ in which $\mathcal{N}$, the set of nodes, represents specific equipment in the network and $\mathcal{L}$, the set of arcs, corresponds to the links that can be used to route the traffic. The routing of the traffic in the network is limited by the bandwidth b_l of each link l. In practice, part of this bandwidth is used to route the legitimate traffic in the network. In the present work, for the sake of simplicity, we assume that the bandwidth consumed by the legitimate traffic is negligible as compared to the one consumed by the illegitimate traffic. We thus consider that the illegitimate traffic may use all the bandwidth of a link if needed.

The illegitimate traffic corresponding to the on-going DDoS attack is represented as a set $\mathcal{A}$ of attacks: attack $a \in \mathcal{A}$ corresponds to an illegitimate traffic of F^a Mbps between a source $s^a \in \mathcal{N}$ and the target $t \in \mathcal{N}$ of the DDoS attack. As explained in the introduction, in the present work, we consider the case in which an ISP lends slices of its physical network infrastructure to its customers and each of these customers uses its own flow routing algorithms to route the flow on the slice assigned to it. This translates into the fact that the exact routing in the network of the malicious flow to be stopped is not known by the ISP at the time when it has to decide about the NFV-based DDoS mitigation infrastructure. Let $\mathcal{P}^a$ be the set of all potential paths between s^a and t for attack a. $\mathcal{N}_p^a$ (resp. $\mathcal{L}_p^a$) denotes the set of nodes (resp. the set of links) belonging to path $p \in \mathcal{P}^a$ and $\mathcal{P}^a(n)$ denotes the subset of paths of $\mathcal{P}^a$ going through node n. The amount of malicious flow of attack $a \in \mathcal{A}$ on path $p \in \mathcal{P}^a$, denoted by $\tilde{f}_p^a$, is thus subject to uncertainty. However, even if the exact value of parameter $\tilde{f}_p^a$ is unknown, there are some restrictions on its potential value. Namely, we know that the total amount of malicious flow routed on the paths belonging to $\mathcal{P}^a$ may not be greater than F^a, the amount of illegitimate traffic of attack a. Moreover, the malicious flow routing must comply with the limited bandwidth of each link. These two pieces of information should be exploited as best as possible to avoid using more network resources than necessary for the DDoS attack mitigation.

In the considered DDoS mitigation framework, VNFs are used to filter and stop the illegitimate traffic before it reaches its target. Basically, filtering consists in selectively stopping unwanted traffic by exploiting the information contained in the header of each data packet. A VNF instantiated on a node $n \in \mathcal{N}$ of the network will thus analyze and filter the malicious flow going through node n. However, each instantiated VNF has a limited filtering capacity which corresponds to the maximum amount of malicious flow it can block: if the malicious flow the VNF has to handle is larger than its filtering capacity, the excess flow is forwarded in the network and may thus reach its target. The set of available VNF types is described by $\mathcal{V} = \{1, ..., V\}$. A VNF of type v is characterized by its filtering capacity ϕ^v, its cost K^v and its computing resources consumption. The set of computing resources (CPU, memory, etc.) is denoted by $\mathcal{R} = \{1, ..., R\}$. Let k^{rv} be the amount of computing resource r required by the instantiation of one VNF of type v and Cap_n^r the amount of computing resource r available at node n.

The optimization problem consists in identifying the location and number of VNFs to be placed in the network so as to stop all the malicious flow before it reaches its target, and this whatever its routing through the network, while minimizing the cost of the instantiated VNFs and complying with the limitations on the computing resources.

8.2.2 Mathematical formulation

We propose to handle this optimization problem using a robust optimization (RO) approach. A robust optimization problem is an optimization problem in which some parameters are subject

to uncertainty. In a RO problem, the uncertainty on the input parameters is not described in terms of probability distributions but rather by means of an uncertainty set containing all the possible values that these parameters may take. Solving a RO problem consists in finding a solution which is feasible for any realization of the uncertain parameters in the uncertainty set and which provides the best possible value of the objective function. The reader is referred to Gorissen et al. (2015) for a practical introduction on robust optimization.

In the present case, the routing of the malicious flow in the network is not known by the ISP. The amount of malicious flow of attack $a \in \mathcal{A}$ on path $p \in \mathcal{P}^a$, $\tilde{f}_p^a$, can thus be seen as an uncertain input parameter for the problem of optimally placing VNFs to counter the DDoS attack. However, even if the exact value of parameter $\tilde{f}_p^a$ is unknown, its value should comply with two restrictions. First, for each attack a, the total flow routed in the network may not be larger than the total attack traffic, i.e. we have $\sum_{p\in\mathcal{P}^a} \tilde{f}_p^a \leq F^a$ for each attack $a \in \mathcal{A}$. Second, the flow routed on each link l of the network may not exceed the bandwidth b_l of this link. We thus have $\sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a \text{ s.t. } l\in\mathcal{L}_p^a} \tilde{f}_p^a \leq b_l$ for each link l.

This means that the uncertain malicious flow routing, $\tilde{f}$, belongs to the uncertainty set $\mathcal{U}$ defined by:

$$\mathcal{U} = \{\tilde{f} \geq 0 | \sum_{p\in\mathcal{P}^a} \tilde{f}_p^a \leq F^a, \quad \forall a \in \mathcal{A}$$
$$\sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a \text{ s.t. } l\in\mathcal{L}_p^a} \tilde{f}_p^a \leq b_l, \quad \forall l \in \mathcal{L}\}$$

Note that the first restriction on $\tilde{f}$ is expressed as a inequality rather than as an equality. Namely, in some cases, it may not be possible to route all the malicious flow of the attack in the network due to the limited bandwidth of the network links. In these cases, expressing the restriction as an equality would lead to an empty uncertainty set. For the RO problem, this would mean that there is no malicious flow routed in the network, i.e. no malicious flow to be stopped by the VNF-based infrastructure, whereas in practice part (but not all) of the attack flow will be routed in the network.

We introduce the integer decision variables x_n^v which represent the number of VNFs of type v placed at node n. Using these variables, the robust optimization problem can be formulated as follows:

$$\begin{cases} Z^* = \min \sum_{v\in\mathcal{V}} \sum_{n\in\mathcal{N}} K^v x_n^v & (8.1) \\ \sum_{v\in\mathcal{V}} k^{rv} x_n^v \leq Cap_n^r \quad \forall n \in \mathcal{N}, \forall r \in \mathcal{R} & (8.2) \\ \sum_{n\in\mathcal{N}(\tilde{f})} \sum_{v\in\mathcal{V}} \phi^v x_n^v \geq \sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a} \tilde{f}_p^a \quad \forall \tilde{f} \in \mathcal{U} & (8.3) \\ x_t^v = 0 \quad \forall v \in \mathcal{V} & (8.4) \\ x_n^v \text{ integer} \quad \forall n \in \mathcal{N}, \forall v \in \mathcal{V} & (8.5) \end{cases}$$

The objective (8.1) is to minimize the total costs of the deployed VNFs. Constraints (8.2) ensure that the VNFs installed at each node n do not consume more than the available computing capacity for each computing resource. Constraints (8.3) translate the fact that we seek to avoid any damage to the target by stopping all the malicious flow before it reaches it. In Constraints (8.3), $\mathcal{N}(\tilde{f}) = \{n \in \mathcal{N} \setminus \{t\} | \sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a(n)} \tilde{f}_p^a > 0\}$ represents the subset of nodes n through which part of the malicious flow transits when considering the flow routing $\tilde{f}$. Constraints (8.3) impose that, for each possible routing $\tilde{f}$, the total filtering capacity installed on

the nodes traversed by a strictly positive amount of malicious flow in the routing $\tilde{f}$, i.e on the nodes belonging to $\mathcal{N}(\tilde{f})$, is larger than the total malicious flow actually routed through the network in $\tilde{f}$. Constraints (8.4) forbid any filtering at the targeted node. Note that Constraints (8.3) are robust constraints that should hold for any flow routing belonging to the uncertainty set $\mathcal{U}$.

Constraints (8.3) can be seen as aggregated attack filtering constraints ensuring that the total filtering capacity installed on the set of nodes traversed by $\tilde{f}$ is larger than the total malicious flow routed through the network. As such, they do not guarantee that the filtering capacity installed on each potential path $p \in \mathcal{P}^a$ of each attack a is enough to stop all the flow related to attack a routed on this path, i.e. that the filtering capacity installed on each path $p \in \mathcal{P}^a$ is larger than $\tilde{f}_p^a$. However, we prove that, for any feasible solution $\overline{x}$ of Problem (8.1)-(8.5) and any flow $\overline{f}$ belonging to $\mathcal{U}$, we can find at least one allocation of the filtering capacity installed at each node n to the flows going through n such that all the malicious traffic can be filtered. See Gicquel et al. (2020) for a more detailed discussion on this point.

Finally, Problem (8.1)-(8.5) is NP-hard, even if the uncertainty set $\mathcal{U}$ contains a finite and discrete set of potential routings. The proof is done by reduction from the minimum set cover problem: see Gicquel et al. (2020) for more detail.

8.3 Solution approach

As explained e.g. by Gorissen et al. (2015), Problem (8.1)-(8.5) may seem intractable as such as the number of constraints (8.3) is infinite. Two main ways have been proposed in the literature to handle this difficulty.

The first one consists in applying reformulation techniques which result in the formulation of a deterministic problem with a finite number of constraints: see e.g. Bertsimas and Sim (2004). In our case, the use of these reformulation techniques is not possible. Namely, the worst case reformation of Constraints (8.3) would lead to the following expression:

$$\min_{\tilde{f} \in \mathcal{U}} \sum_{n \in \mathcal{N}} \mathbb{I}\Big(\sum_{a \in \mathcal{A}} \sum_{p \in \mathcal{P}^a(n)} \tilde{f}_p^a > 0 \Big) \sum_{v \in \mathcal{V}} \phi^v x_n^v - \sum_{a \in \mathcal{A}} \sum_{p \in \mathcal{P}^a} \tilde{f}_p^a > 0 \tag{8.6}$$

where $\mathbb{I}\Big(\sum_{a \in \mathcal{A}} \sum_{p \in \mathcal{P}^a(n)} \tilde{f}_p^a > 0 \Big)$ is an indicator function that is equal to one if $\sum_{a \in \mathcal{A}} \sum_{p \in \mathcal{P}^a(n)} \tilde{f}_p^a > 0$ and zero otherwise. The resulting inner minimization problem cannot be formulated as a linear program (but rather as a mixed-integer linear program) due to the presence of this indicator function. It is thus not possible to use the duality theory to reformulate it and obtain a computationally tractable robust counterpart as is commonly done in this type of reformulation approach.

The second possible way of solving a RO problem such as Problem (8.1)-(8.5) consists in applying an adversarial approach. Such approaches are based on the decomposition of the initial problem into a master problem and a sub-problem. The master problem, called the decision maker problem in this context, can be seen as a restricted version of the original RO problem in which only a finite number of extreme points $\mathcal{U}_R \subset \mathcal{U}$ of the uncertainty set (instead of the whole uncertainty set $\mathcal{U}$) are used to express the robust constraints. This problem is a deterministic optimization problem with a finite number of constraints and is thus computationally tractable. The sub-problem is called the adversarial problem. Given the solution provided by the decision maker problem, the adversarial problem seeks to find an extreme point of $\mathcal{U}$ for which this solution is infeasible. If no such extreme point can be found, the current solution of the decision maker problem is optimal for the initial RO problem. If such an extreme point is found, we add it to the restricted set $\mathcal{U}_R$ and reiterate the process. The finite convergence of this algorithm is ensured by the fact that the uncertainty set $\mathcal{U}$ has a finite number of extreme

points. Adversarial approaches have been successfully used to solve RO problems arising in a variety of applications: see among others Bienstock and Özbay (2008), Attila et al. (2017), van Hulst et al. (2017) and Agra et al. (2018).

The proposed adversarial approach thus iteratively solves the decision maker problem and the adversarial sub-problem. At each iteration, the decision maker problem is solved using the current restricted uncertainty set $\mathcal{U}_R$ and provides a placement of the VNFs $\overline{x}$ which is optimal for this restricted uncertainty set. $\overline{x}$ being given, the adversarial problem is solved to find the worst-case routing of the malicious flow for the VNF placement described by $\overline{x}$, i.e. to find an extreme point of $\mathcal{U}$ wich maximises the infeasibility of $\overline{x}$ if it exists. In case such an extreme point is found, we update the restricted uncertainty set $\mathcal{U}_R$ by adding the newly found routing $\overline{f}$ and go on to the next iteration. Otherwise, $\overline{x}$ is feasible for all extreme points of $\mathcal{U}$, the current VNF placement $\overline{x}$ is optimal and the algorithm stops.

8.3.1 Decision maker sub-problem

The decision maker problem, denoted by $DMP(\mathcal{U}_R)$, can be formulated as follows:

$$
\begin{cases}
Z^*_{DMP} = \min \sum_{v\in\mathcal{V}} \sum_{n\in\mathcal{N}} K^v x^v_n & & (8.7)\\
\sum_{v\in\mathcal{V}} k^{rv} x^v_n \leq Cap^r_n & \forall n \in \mathcal{N}, \forall r \in \mathcal{R} & (8.8)\\
\sum_{n\in\mathcal{N}(f)} \sum_{v\in\mathcal{V}} \phi^v x^v_n \geq \sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a} f^a_p & \forall f \in \mathcal{U}_R & (8.9)\\
x^v_t = 0 & \forall v \in \mathcal{V} & (8.10)\\
x^v_n \text{ integer} & \forall n \in \mathcal{N}, \forall v \in \mathcal{V} & (8.11)
\end{cases}
$$

Problem $DMP(\mathcal{U}_R)$ thus displays the same structure as the initial RO problem but the number of Constraints (8.9) is now finite. Moreover, as will be shown by the numerical experiments provided in Section 8.4, in practice, the cardinality of $\mathcal{U}_R$, and as a consequence the number of Constraints (8.9) involved in the formulation, remain rather limited when implementing the adversarial approach. Problem $DMP(\mathcal{U}_R)$ can thus be directly solved by a mixed-integer linear programming solver with a reasonable computational effort.

8.3.2 Adversarial sub-problem

Let us now focus on the adversarial sub-problem. In order to formulate it, we introduce the following decision variables:
- f^a_p: amount of malicious flow of attack a routed on path $p \in \mathcal{P}^a$,
- $z_n \in \{0,1\}$: $z_n = 1$ if there is a positive amount of malicious flow transiting through node n, 0 otherwise.

Given the current VNF placement $\overline{x}$, the maximum amount of malicious flow which can reach its target can be found by solving the following mixed-integer linear program, denoted by $AP(\overline{x})$.

$$
\begin{cases}
Z^*_{AP}(\bar{x}) = \max \sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a} f^a_p - \sum_{n\in\mathcal{N}\setminus\{t\}} (\sum_{v\in\mathcal{V}} \phi^v \bar{x}^v_n) z_n & & (8.12)\\
\sum_{p\in\mathcal{P}^a} f^a_p \leq F^a & \forall a \in \mathcal{A} & (8.13)\\
\sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a \text{ s.t. } l\in\mathcal{L}^a_p} f^a_p \leq b_l & \forall l \in \mathcal{L} & (8.14)\\
\sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a(n)} f^a_p \leq (\sum_{a\in\mathcal{A}} F^a) z_n & \forall n \in \mathcal{N}\setminus\{t\} & (8.15)\\
f^a_p \geq 0 & \forall p \in \mathcal{P}^a & (8.16)\\
z_n \in \{0,1\} & \forall n \in \mathcal{N} & (8.17)
\end{cases}
$$

The linear variables f thus describe the worst-case routing of the malicious flow for the VNF placement $\bar{x}$. Constraints (8.13) ensure that, for each attack, the total amount of flow of attack a routed through the network is smaller that the total amount of flow of the attack F^a. Note that due to the limited bandwidth of the network links, it might not be possible to route all the flow of attack a through the network: Constraints (8.13) are thus formulated as inequalities rather than as equalities. Constraints (8.14) guarantee that the flow routed on each link does not exceed its bandwidth. In other words, Constraints (8.13), (8.14) and (8.16) make sure that the solution of problem $AP(\bar{x})$ provides a flow f belonging to the uncertainty set $\mathcal{U}$.

The objective function (8.12) seeks to maximize the amount of malicious flow which will reach its target, i.e. which will not be filtered by a VNF between its source and its target. Note that the filtering capacity $\sum_{v\in\mathcal{V}} \phi^v \bar{x}^v_n$ placed at node n can stop part of the malicious flow only if there is a positive flow routed through node n, i.e. only if $\sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a(n)} f^a_p > 0$. $\sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a} f^a_p - \sum_{n\in\mathcal{N}\setminus\{t\}} (\sum_{v\in\mathcal{V}} \phi^v \bar{x}^v_n) z_n$ thus computes the total amount of unfiltered flow as the difference between the total flow routed through the network, $\sum_{a\in\mathcal{A}} \sum_{p\in\mathcal{P}^a} f^a_p$, and the total amount of 'active' filtering capacity, $\sum_{n\in\mathcal{N}\setminus\{t\}} (\sum_{v\in\mathcal{V}} \phi^v \bar{x}^v_n) z_n$. This 'active' filtering capacity is given by the sum of the filtering capacities installed at the nodes n through which a positive amount of malicious flow transits. Constraints (8.15) ensure that, for each node n, variable z_n is equal to 1 as soon as there is some positive amount of malicious flow which is routed through node n.

Note that, similar to what is done in Constraint (8.3) of the initial RO problem, in the objective function (8.12) of the adversarial subproblem, the total amount of unfiltered flow is computed in a aggregate manner, i.e. by looking at the total routed flow and at the total active filtering capacity on all nodes of the network. In a feasible solution of problem $AP(\bar{x})$, this might lead to an underestimation of the malicious flow which will reach its target. However, we show that any optimal solution of $AP(\bar{x})$ provides the worst-case routing for the given VNF placement $\bar{x}$ (Gicquel et al., 2020).

The adversarial sub-problem $AP(\bar{x})$ is a mixed-integer linear program which could theoretically be solved directly by a mathematical programming solver. However, the number of paths that could possibly be used to route the malicious flow of a given attack a between its source s^a and the target t, and as a consequence the number of flow variables f^a_p, grows exponentially fast with the network size.

We thus developed a branch-and-price algorithm to solve it. This algorithm starts solving a restricted version of the adversarial sub-problem in which only a subset of flow variables f^a_p is taken into account. At each node of the branch-and-bound search tree, we solve the linear relaxation of this restricted sub-problem and seek for new flow variables to be added to the formulation by solving the pricing problem, which in the present case amounts to solving a shortest path problem in an oriented graph. When no new variable can be generated, i.e. when the linear relaxation of the restricted master problem has been solved to optimality at

the current branch-and-bound node, we either get an integer feasible solution of the initial problem $AP(\overline{x})$ or we branch on a fractional variable z_n to create new nodes in the search tree and continue with the branch-and-bound algorithm. The algorithm stops when there are no more open nodes in the search tree.

8.4 Computational experiments

8.4.1 Instances

We randomly generated a set of medium-size instances of the problem following the indications provided by public data released by different cloud and telecom providers.

Network. We used 4 internet network topologies. The first three ones correspond to three internet networks described in the Internet Topology Zoo library, IntelliFiber ($N = 73$, $L = 96$), Colt Telecom ($N = 153$, $L = 179$) and Cogentco ($N = 197$, $L = 245$): see Knight et al. (2011) and Knight et al. (2013) for more detail. We also used a topology corresponding to the former network of the French company Free ($V = 120$, $L = 167$): see Ferre (2010). The bandwidth b_l of each link was randomly generated using a discrete distribution with a support equal to $\{4.8, 12, 20, 40, 100\}$ Mbps.

Computing resources. $R = 2$ types of computing resources were taken into account at each node: the number of CPUs and the memory. We considered three types of nodes: low computing capacity with $Cap = (8, 32)$, medium computing capacity with $Cap = (40, 160)$ and high computing capacity with $Cap = (400, 1600)$. In each considered network topology, we assign each node to a type according to its degree. Thus, nodes with a degree less than 2 were assigned a low computing capacity, nodes with a degree between 3 and 5 were assigned a medium computing capacity and nodes with a degree larger than 6 were assigned a high computing capacity.

VNFs. $V = 1$ type of VNFs was considered requiring $\gamma^{1,1} = 4$ CPUs and $\gamma^{1,2} = 16$ units of memory, providing a filtering capacity of $\phi^n = 16$ Mbps, with a unit cost of $K^1 = 130$.

Attacks. The number of sources was set to $A \in \{5, 10, 15, 20, 30, 40\}$. In each instance, the sources and target of the attack were randomly selected. The intensity F^a of each attack (in Mbps) was randomly generated following the normal distribution $\mathcal{N}(50, 25)$.

For each considered network topology and value of A, we randomly generated 5 instances, leading to a total of 140 instances.

8.4.2 Results

Each generated instance was solved using the adversarial algorithm described in Section 8.3. The decision maker problem is solved as a mixed-integer linear program using the solver CPLEX 12.8.9 with the default settings whereas the adversarial sub-problem is solved using the branch-and-price algorithm embedded in the SCIP 7.0.0 solver. All tests were run on an Intel Core i5 (1.9GHz) with 16 Gb of RAM, running under Windows 10.

For each network topology and each considered value of A, we report in Table 8.1 the average value over the 5 corresponding instances of:

- *Cost*: the cost of the optimal VNF placement,
- $\#IT$: the average number of iterations of the algorithm,
- $\#P$: the total number of flow variables added to the adversarial subproblem by column generation over the course of the algorithm,
- *Time*: the total computation time in seconds of the algorithm.

Topology	A	*Cost*	*#IT*	*#P*	*Time*
IntelliFiber	5	936	9	25	3
	10	1066	13	36	10
	15	1248	11	45	4
	20	988	9	35	11
	30	1846	31	130	134
	40	1950	17	130	31
Free	5	1092	12	27	10
	10	1326	10	41	4
	15	1378	5	60	3
	20	650	5	28	1
	30	1092	8	71	3
	40	1352	7	63	4
Colt	5	1118	17	42	12
	10	806	7	35	2
	15	1170	20	63	38
	20	1092	10	63	14
	30	754	7	71	4
	40	1118	9	66	9
Cogentco	5	546	13	58	9
	10	754	14	70	24
	15	1222	18	95	20
	20	910	21	81	47
	30	728	13	84	34
	40	1066	20	101	107

TABLE 8.1: Numerical results

Results from Table 8.1 show that the proposed approach is able to provide optimal solutions to the RO problem with a reasonable computational effort. Namely, the average computation time, over the 140 considered instances, is 22s. This performance is mainly explained by the fact that both the number of iterations $\#IT$ of the algorithm (and as a consequence the number of Constraints (8.9) of $DMP(\mathcal{U}_R)$) and the number of flow variables in $AP(\bar{x})$ generated by column generation stay limited.

8.5 Conclusion and perspectives

This chapter described a new robust optimization approach for the defense against Distributed Denial of Service (DDoS) attacks in the context of 5G network slicing. More precisely, we considered the problem of optimally deploying virtual network functions in order to stop an ongoing DDoS attack. We assumed that the target, sources and volume of the attack is identified but that the exact routing of the illegitimate traffic on the network is not known. To take into account these uncertainties, we proposed a robust optimization (RO) model and developed an adversarial approach to solve it. This iterative approach is based on the decomposition of the initial problem into a master problem and a sub-problem. The master problem is a restricted version of the original RO problem in which only a finite number of possible malicious flow routings are used to express the robust constraints. Considering the current VNF placement provided by the solution of the master problem, the adversarial sub-problem seeks to find a malicious flow routing that maximizes the amount of attack reaching its target. We tested

the efficiency of our algorithm on medium-sized randomly generated instances. The results of computation experiments show that our approach is able of providing optimal solutions in short computation times.

This work suggests several possible directions for future research. A first research direction could consist in studying a disaggregated formulation of the robust filtering constraints. This could ensure that the instantiated VNFs will be able to stop all the malicious flows regardless of the allocation of filtering capacities. It would also be interesting to study how the legitimate traffic, which will consume network resources and whose routing is also unknown, could be taken into account in the model.

Chapter 9

Conclusion and perspectives

9.1 Conclusion

This thesis presented the main research results I obtained since I was recruited as an assistant professor in 2010. Over the last ten years, I worked on difficult deterministic and stochastic combinatorial optimization problems, mainly coming from applications in manufacturing, supply chain management and telecommunication. A common point in the presented contributions is the fact that they are all based on mixed-integer linear programming.

Part I was devoted to the work carried out on lot-sizing problems. Basically, all this work aimed at decreasing the gap between lot-sizing models and real-life production planning problems by relaxing some of the rather strong assumptions used to formulate these models. Thus, we studied in Chapter 3 an extension of the basic lot-sizing models in which the setup costs are sequence-dependent and proposed a new family of multi-item multi-period valid inequalities to strengthen the MILP formulation of this problem. Chapter 4 to 6 then considered another aspect of production planning, namely the fact that it is based on input data which are relative to the near future and, as a consequence, are not always perfectly known at the time when the production plan has to be built. Neglecting this fact when solving the obtained lot-sizing model may lead to a production plan which will perform poorer than expected in terms of demand satisfaction and/or production costs when implemented in practice. We thus studied several stochastic programming approaches for lot-sizing. The first proposed approach, presented in Chapter 4, assumed a rather simplified setting for the decision process. It namely considered a single-stage decision process in which the whole production plan is built before any additional information on the stochastic demand realization becomes available and cannot be updated afterward as the demand unfolds over time. We formulated this stochastic problem as a joint chance-constrained program and proposed a new approximate solution approach for this problem called the partial sample approximation approach. Then, in order to further improve the modeling of the actual decision process, we investigated a multi-stage stochastic programming approach which explicitly takes into account the fact that production decisions are usually not made once and for all but rather adjusted over time according to the actual realizations of the uncertain parameters. Our main contribution, reported in Chapter 5, consisted in the development of a new algorithm capable of providing good-quality solutions for large-size instances of the stochastic single-item uncapacitated lot-sizing problem. This algorithm combines a nested decomposition algorithm called the Stochastic Dual Dynamic integer Programming (SDDiP) algorithm with a cutting-plane generation approach based on known valid inequalities. Finally, in order to extend the work of Chapters 4 and 5 which focused on single-item single-echelon single-resource production systems, we considered a multi-item multi-echelon multi-resource production system linked to the remanufacturing of used products and studied a multi-stage stochastic programming model for this problem. Chapter 6 thus presented a customized branch-and-cut algorithm based on a new family of valid inequalities for this problem.

In parallel to the above-mentioned work on lot-sizing, I had the opportunity to study two applied facility location problems through industrial collaborations. The participation to these two projects allowed me to gain experience and improve my skills in the field of modeling real-life optimization problems with mathematical programming. The corresponding work was reported in Part II of the manuscript. Chapter 7 discussed a facility location problem linked to the design of the outbound logistics network of Renault. The main contribution of this work was related to the development of a tractable solution approach for this complex large-size location-routing problem. As for Chapter 8, it focused on the placement of virtual network functions in a telecommunication network to secure it against a distributed denial-of-service attack. The main novelty here consisted in the development of a new robust optimization model and adversarial algorithm for this problem.

9.2 Perspectives

The work presented in this manuscript raises several issues and research perspectives. We describe in what follows some research directions that we intend to pursue in the coming years.

9.2.1 Risk aversion in multi-stage stochastic lot-sizing

The multi-stage stochastic lot-sizing models investigated in Chapters 5 and 6 focus on minimizing the expected cost of the production plan and thus take a risk-neutral perspective. However, such models may lead to production plans displaying a good performance on average, but providing very poor results, i.e. production costs much higher than the expected value, in some unfavorable scenarios. Risk-neutral production planning models thus implicitly assume that the production planner is willing to accept the risk that the realized total production cost may be much larger than the expected value as long as these unfavorable scenarios are counterbalanced with more favorable ones, overall leading to a low expected value of the production costs. However, some production planners may be risk averse, i.e. may be more concerned about potential large monetary losses than about the average performance of the production plan.

In such cases, it is possible to include in the objective function terms that can measure exposition to risk so as to mitigate the effects of undesirable realizations of the stochastic parameters. There exists a large variety of singe-stage risk measures such as the conditional value-at-risk (CVaR), the upper partial mean or the semi-deviation that can be used to measure this exposition to risk. A first practical difficulty here comes from the fact that, as mentioned e.g. by Alem et al. (2020), there does not seem to be an unrestrictedly recommendable risk measure for production lot-sizing problems or for any other class of problems. The justification for adopting one risk measure over another one is usually given either regarding the preferences of the decision maker, the tractability of the resulting optimization model or the theoretical properties of the risk measure. Alem and Morabito (2013) and Alem et al. (2020) tried to provide some insights about this question by numerically comparing several risk-averse approaches on real-life lot-sizing instances. However, they focus on two-stage stochastic lot-sizing models, i.e. models involving only two decision stages such as the one presented in Subsection 2.3.2. As explained e.g. by de Mello and Pagnoncelli (2016), for two-stage stochastic programming models, the extension of a risk-neutral to a risk-averse model does not pose major modeling difficulties. Namely, the objective function comprises two terms: the deterministic first-stage costs and the random second-stage/recourse costs. It is thus quite natural to replace the expectation of this second-stage costs by a single-stage risk measure such as the CVaR.

Yet, the situation is different when dealing with multi-stage stochastic lot-sizing models. Namely, in a sequential decision making process, there does not seem to be one natural and obvious way of measuring risk (de Mello and Pagnoncelli, 2016). Risk may be measured e.g. stage by stage, scenario by scenario or in a nested way. Furthermore, one important issue encountered when dealing with risk-averse multi-stage stochastic programs is that of time consistency. Basically, a time consistent model will be such that if you solve the multi-stage stochastic program today and find a solution for each node of the scenario tree, you should find the same solution if you resolve the problem in the future knowing what has been observed and decided in between (de Mello and Pagnoncelli, 2016). Although this property may seem to be natural and desirable, it does not hold for all multi-stage risk measures. In particular, de Mello and Pagnoncelli (2016) showed that risk measures considering the risk stage by stage or scenario by scenario are most often time inconsistent.

Research on risk averse multi-stage stochastic lot-sizing is thus needed to compare the various existing multi-stage risk measures that may be used, to understand their potential advantages and disadvantages and to highlight the benefit of using a risk-averse model as compared to a more intuitive risk-neutral one. Subsequently, it may also be necessary to develop efficient solution approaches for these models, possible by relying on nested decomposition approaches such as the one presented in Chapter 5.

In order to contribute to this line of research, I am currently studying, together with Franco Quezada and Safia Kedad-Sidhoum, a risk-averse model for the multi-stage stochastic single-item uncapacitated lot-sizing problem. More precisely, we are investigating the use of a time-consistent multi-stage risk measure proposed by de Mello and Pagnoncelli (2016) called the expected conditional value-at-risk. Our on-going work consists in conducting simulation experiments to gain a quantitative assessment of the practical usefulness of using this measure in the objective function instead of a risk-neutral expectation term or a time-inconsistent risk measures. We are also seeking to generalize the partial decomposition approach presented in Chapter 5 to solve a risk-averse multi-stage stochastic ULS model in which risk is measured in the objective function using the expected conditional value-at-risk.

9.2.2 Multi-stage stochastic lot-sizing with intermittent renewable energy

All production systems consume energy to run. In the production planning models investigated in the present thesis, energy is taken as granted, i.e. considered as a utility which is available as needed and whose cost is negligible as compared to the other production and inventory costs. However, the rarefaction of natural resources, the hardening of environmental public legislation and the growing customer awareness for green products increasingly make a difference for industrial companies. The availability and affordability of energy is thus becoming a critical parameter affecting the whole life cycle of an industrial product, including its production phase (unpu and Ball, 2013). This translates into a growing attention of both researchers and practitioners on energy-efficient manufacturing.

Energy efficiency in manufacturing can be achieved among others through the on-site generation of renewable energy, such as solar and wind power, to supply power to a production facility. Indeed, it enables the industrial manufacturer to simultaneously minimize its energy cost, reduce its reliance on external utility companies and decrease its CO2 emissions. For instance, the British luxury car maker Bentley installed in 2013 more than 20000 solar panels on the rooftop of its factory located in Crewe, UK, and added in 2019 a solar car port comprising 10000 additional solar panels. The 7.7 MW energy system is expected to provide up to two third of the factory energy requirement at peak generation times (Szymkowski, 2018; Bentley, 2019). Similarly, the Anheuser-Busch brewery in Fairfield (California) gets 30% of its electricity from a wind turbine and 6500 solar panels located on the production site (Anheuser-Busch, 2021).

However, using renewable energy to power manufacturing processes poses a major challenge. Namely, the generation of renewable energy depends on the availability of its sources (sun, wind...) so that the instantaneous power it can supply is highly variable. Hence, renewable energy is not expected to fully replace grid electricity but rather to be used in combination with it. In order to fully exploit the renewable energy availability, a manufacturing company will thus have to synchronize the time-varying on-site produced energy and the grid energy supply with the energy demand in an efficient way. This means that on-site renewable energy generation should be integrated into production planning and scheduling processes (Keller et al., 2016; Pechmann and Zarte, 2017).

The incorporation of energy considerations in production planning and scheduling problems has attracted considerable attention over the last few years. As can be seen from the recent literature reviews provided by Biel et al. (2018) and Gahm et al. (2016), the vast majority of previously published studies on energy-aware production planning assume that there is a single source of energy available to supply the production processes and focus mostly on planning production so as to minimize the quantity of energy consumed or to reduce the total energy cost by taking advantage of time-varying energy prices. Among the works dealing with energy-aware dynamic lot-sizing problems, two main streams can be distinguished. A first stream assumes that the energy is available in an unlimited quantity and focus on explicitly integrating the energy costs in the objective function: see e.g. Özdamar and Birbil (1999), Uzel (2004), Tang et al. (2012) and Wichmann et al. (2019b). A second one seeks to take into account restrictions on the energy availability coming e.g. from demand-side management programs initiated by utility providers: see Masmoudi et al. (2017), Rapine et al. (2018b) and Rapine et al. (2018a).

Despite its increasing importance, the integration of on-site generated renewable energy and its use in combination with other energy sources to supply production has received only a limited attention. Mid-term production planning with intermittent renewable is considered by Golari et al. (2017), Wichmann et al. (2019a) and Rodoplu et al. (2019). Golari et al. (2017) study an aggregate production planning problem for a multi-plant manufacturing system powered with on-site and grid renewable energy. They propose a multi-stage stochastic programming approach based on a scenario tree to model the unpredictability of solar and wind energy generation. This leads to the formulation of a large-size linear program which is solved by a Benders decomposition algorithm. Wichmann et al. (2019a) investigate an extension of the deterministic General Lot Sizing and Scheduling Problem (GLSP) in which the production system is coupled with a decentralized renewable energy generation and storage system. They rely on three time scales to build the production and energy plan: macro-periods focusing on the industrial product demand, micro-periods of variable length focusing on the machine set-up and production states and micro-periods of fixed length for the energy management. Their optimization problem is formulated as a mixed-integer linear program and solved by a mathematical programming solver. Rodoplu et al. (2019) consider lot-sizing for a single-item multi-echelon serial production system. This system is powered by electricity which is both bought from the grid and generated on-site from renewable sources. The authors propose to take into account the uncertainty on the renewable energy availability through a set of disjoint chance constraints.

There is thus a significant gap between the state of the art and the industrial needs. In the forthcoming years, I would like to contribute in partially closing this gap by studying multi-stage stochastic programming approaches for production planning and lot-sizing with intermittent renewable energy. An idea would be to consider a small-bucket lot-sizing problem (such as the multi-item DLSP discussed in Chapter 3) and to incorporate in the production planning model the energy supply management. This energy might be either bought from the grid or generated on site from a renewable energy source. The first issue will be to handle in the problem modeling different time discretizations as the time slicing needed to manage demand

satisfaction, production planning and energy supply will be significantly different. Moreover, the production and energy supply planning will be based among others on forecasts on the future renewable energy generation. It is well-known that accurately predicting the energy generated by a wind turbine or a set of solar panels is a very challenging task. We thus will seek to take into account these uncertainties in the problem modeling through the development of a multi-stage stochastic programming approach. An important issue will consist in the generation of relevant scenario trees taking into account among others the temporal stochastic dependencies between forecasting errors: see e.g. Pinson et al. (2009) on the generation of scenarios for wind power production. Finally, once the mathematical model will be built, research work will be needed to develop an efficient solution algorithm. Namely, due to the existence of the temporal dependencies between the random variables representing the forecasting errors, the obtained scenario trees will not have the stage-wise independence property so that it will not be possible to use a sampling-based nested decomposition approach such as the one investigated in 5.

9.2.3 Explainable lot-sizing

As operations researchers working on lot-sizing problems, we are convinced that mathematical optimization can lead to more efficient decision making in production planning. We define mathematical models to represent the optimization problem and solve them either by relying on solvers based on well-defined algorithms or by developing our own solution algorithms. This helps us being confident in the quality of the production plans we are generating. However, from the point of view of the end user, things are less obvious. Namely, the production planners using a planning software and the factory workers in charge of implementing the production plan usually do not have the mathematical and computer background necessary to understand the content of the planning software and may consider it as an unexplainable black-box. They may thus inquire about the optimality and fairness of the proposed plan. For instance, a production planner may question the consistency of the computed production plan and may want to understand why demand for a given item cannot be met on time or why the same item is produced in two consecutive periods... Furthermore, a factory worker who will be directly affected by production planning decisions may wish to understand his situation and ask e.g. why he has to work overtime next week-end or why his team has been assigned complex and/or dangerous setup operations twice this week. All these questions, if not answered appropriately, may lead to a loss of trust in the planning software and may thus negatively impact its adoption.

One way to address this problem could be to improve the explainability of production planning and lot-sizing decision-aid tools. Explainability is a recently emerged concept mostly used in the context of artificial intelligence and machine learning. Explainability is basically described by the French Commission Nationale Informatique & Libertes as the fact that it is possible for the user to understand the global working logic of the software together with the reasons why a specific result was obtained (CNIL, 2017). Barredo Arrieta et al. (2020) define explainability as the details and reasons a software gives to make its functioning clear or easy to understand for a given audience. Note that explainability differs from transparency which refers to the fact that the source code of the software, together with its design documentation and parameters, is available to the public. As noted e.g. by the CNIL (2017), even if transparency may be desirable, it may be unsatisfying as releasing the source code of a software leaves the vast majority of its users unable to understand its working logic. Explainability thus requires the delivery of information that goes beyond describing the content of the software(Castellucia and Le Métayer, 2019). One of the challenges here consists in understanding what constitutes a good explanation. Barredo Arrieta et al. (2020) insist among others on the importance of clearly defining the targeted audience. Namely, whether an explanation has left

the functioning of a decision-aid tool clear or easy to understand fully depends on the persona of the explainee. Moreover, a good explanation should be simple and combine few causal relationships in order to be cognitively tractable for the explainee. It should also be efficiently attainable, i.e. quick to get in terms of computation time, as explanations are often provided through real-time interactions with the end user: see Cyras et al. (2019). Finally, in addition to this discussion about the required features of the content of a good explanation, there is a need to define how this content will be communicated to the explainee through e.g. natural language or graphical interfaces.

To the best of our knowledge, there is a single study seeking to provide explanations on the solution of a combinatorial optimization problem. Cyras et al. (2019) thus consider the problem of assigning medical tasks to hospital nurses so as to minimize the makespan. They develop an intermediate explanation layer, based on abstract argumentation, between the user (the head nurse in their case) and the optimization tool to enable her to interact with the optimization tool and obtain explanations about the (un)feasibility and (non)optimality of alternatives schedules she proposes. However, the presented explanation layer heavily relies on the specific structure of the considered combinatorial optimization problem and can therefore not be straightforwardly extended to provide explanations for any combinatorial optimization problem. I am currently involved in a collaborative research project dealing with the development of explanations schemes for a workforce routing problem. The project started in December 2020 and involves a PhD student (Mathieu Lerouge), researchers from CentraleSupelec (Vincent Mousseau and Ouassila Ouerdane) and engineers from the consulting company DecisionBrain (Daniel Godard and Désirée Rigonat). The optimization problem consists in assigning maintenance jobs to be carried out at geographically scattered facilities to technicians and in building, on a daily basis, the schedule and routes for these technicians so as to minimize the total transportation costs and lateness penalties. For the time being, we focus on providing local and contrastive explanations, i.e. explanations pertaining to a small part of the overall schedule and describing why the software proposes a given task-technician assignment and/or a given route rather than an alternative one suggested by the user (the workforce planner in our case). A typical example could be to answer the question: why does technician 1 carries out the sequence of tasks 1-3-2 rather than the sequence 1-2-3 ? A planner may ask such a question e.g. when the sequence 1-2-3 enables to reduce the total driven distance as compared to sequence 1-3-2 because tasks 1 and 2 are located at the same facility. In this case, possible explanations may be linked to the infeasibility of the proposed alternative (sequence 1-2-3 is unfeasible because it is not possible to carry out task 3 before its facility closes) or to the sub-optimality of the proposed alternative (sequence 1-2-3 is sub-optimal because, even if transportation costs are reduced, it results in a high lateness penalty on task 3 which strongly deteriorates the objective function value). My plan is to increase my knowledge and skills on the subject of explainability, which is totally new for me, thanks to the participation to this project and once I will have a sufficient mastery, I will seek to develop explanation schemes for production planning and lot-sizing problems.

Appendix A

Curriculum vitae

Céline Gicquel
Assistant professor in Operations Resarch and Computer Science

Université Paris Saclay
Laboratoire Interdisciplinaire des Sciences du Numérique (LISN)
Campus d'Orsay - bâtiment 660
F-91405 Orsay Cedex
e-mail: celine.gicquel@universite-paris-saclay.fr

Professional background

2010-present	Université Paris Saclay Assistant professor in Computer Science - Researcher at the Laboratoire Interdisciplinaire des Sciences du Numérique (LISN) - Teacher at the Institut Universitaire de Technologie d'Orsay (IUT Orsay)
2005-2010	Ecole Centrale Paris 2008-2010: Post-doctoral research fellow 2005-2008: Doctoral research fellow
2003-2004	Procter & Gamble Logistics process engineer

Academic background

2008	Ecole Centrale Paris PhD in Industrial Engineering Dissertation: 'MIP models and valid inequalities for the discrete lot-sizing and scheduling problem with sequence-dependent setups' Advisors: Pr Michel Minoux and Pr Yves Dallery
2005	Ecole Centrale Paris Research master degree in Industrial Engineering
2002	Ecole Nationale Supérieure des Techniques Avancées - Paris Engineering master degree in Production and Logistics managament

Student supervision

2020-present	Mathieu Lerouge, PhD student Explanation schemes for recommendations coming from optimization systems: application to workforce scheduling Funding: project AIDA in collaboration with IBM and DecisionBrain Co-supervision with Vincent Mousseau and Ouassila Ouerdane
2019-present	Bingqian Liu, PhD student Optimal design of local multi-energy systems Funding : EDF R&D - ANRT (CIFRE grant) Co-supervision with Dominique Quadri
2018-present	Franco Quezada, PhD student Multi-stage stochastic lot-sizing Funding: research grant from Sorbonne Universités Co-supervision with Safia Kedad-Sidhoum
2017-2018	Ahmed Kadri, post-doctoral fellow Location of electric vehicle charging stations under uncertain demand Funding: United Arab Emirates University Co-supervision with Mouna Kchaou-Boujelben
2012-2015	Mathilde Excoffier, PhD student Call center shift scheduling under demand uncertainty Funding: project SPACE funded by Digitéo Co-supervision with Abdel Lisser, Oualid Jouini and Steven Martin
2010-2013	Mouna Kchaou-Boujelben MILP models and heuristics for optimally designing Renault outbound logistics network Funding: Renault-ANRT (CIFRE grant) Co-supervision with Michel Minoux
2007-2021	Supervision or co-supervision of 12 master students

Research grants

2020-2023	Project funded by EDF R&D Optimal design of local energy systems
2017-2021	Project funded by the Université Paris Saclay (Labex Mathématiques Hadamard) Energy-efficient production planning and lot-sizing
2018-2019	Project funded by the Ile de France region (DIM RFSI program) Stochastic optimization of the daily scheduling of smart grids
2017-2019	Project funded by the United Arab Emirates University Location of electric vehicle charging stations

2015-2019	Project funded by the French Programme Gaspard Monge pour l'Optimisation (PGMO) Planning remanufacturing activities under uncertainty
2015-2016	Project funded by the Université de Lorraine Lot-sizing under energy availability constraints
2012-2015	Project funded by the Digiteo program Call-center shift scheduling under uncertain call arrival rates
2011-2014	Project funded by the French Agence Nationale pour la Recherche (program for young researchers) Strong linear and semi-definite relaxations for lot-sizing problems

Professional service

2018-present	Vice-president in charge of the external relationships and communication of the French Society for Operations Research and Decision Aid (ROADEF)
2021	Co-responsible for the transverse action on 'Robust decision and optimization" of the French CNRS research group on Operations Research (GDR-RO)
2021	President of the jury of the 2020 ROADEF master thesis competition
2019-2020	President of the jury of the 2019 & 2020 best student paper awards at the annual conference of the French Society for Operations Research and Decision Aid (ROADEF)
2018	Member of the jury of the 2018 Doctoral Dissertation Award of the Gaspard Monge program for Optimization (PGMO)
2017	Member of the jury of the 2017 best student paper award at the annual conference of the French Society for Operations Research and Decision Aid (ROADEF)
2012-2021	Member of 10 recruitment committees for assistant professor positions
2014-2020	Member of the jury of 7 PhD thesis
2019	Member of the organizing committee of the 2019 International Workshop on Lot-sizing (IWLS 2019)
2017-present	Member of the program committee of the annual conference of the French Society for Operations Research and Decision Aid (ROADEF)

Teaching

2010-present	University Paris Saclay - IUT Orsay - Computer science department 2011-2017: Responsible for the course 'Introduction to databases' (BSc) 2015-present: Local correspondent for the APOGEE education management software

2015-présent	University Paris Saclay - Faculty of Science - Maths department Co-responsible for the course 'Introduction to operations research' (MSc)
2008-2018	Ecole Centrale Paris Co-responsible for the course 'Network optimization' (MSc)

Main publications

Refereed international journal papers

1. M. Kchaou-Boujelben, C. Gicquel. Locating electric vehicle charging stations under uncertain battery energy status and power consumption. *Computers & Industrial Engineering*, 2020, vol. 149, 106752

2. A. Kadri, R. Perrouault, M. Kchaou Boujelben, C. Gicquel. A multi-stage stochastic integer programming approach for locating electric vehicle charging stations. *Computers & Operations Research*, 2020, vol. 117, 104888.

3. F. Quezada, C. Gicquel, S. Kedad-Sidhoum, D.Q. Vu. A multi-stage stochastic integer programming approach for a multi-echelon lot-sizing problem with returns and lost sales. *Computers & Operations Research*, 2020, vol. 116.

4. M. Kchaou-Boujelben, C. Gicquel. Efficient solution approaches for locating electric vehicle fast charging stations under driving range uncertainty. *Computers & Operations Research*, 2019, vol. 109, pp 288-299.

5. J. Cheng, C. Gicquel, A. Lisser. Partial sample approximation method for chance-constrained problems. *Optimization Letters*, 2019, vol. 13(4), pp 657–672.

6. C. Rapine, B. Penz, C. Gicquel, A. Akbalik. Capacity acquisition for the single-item lot-sizing problem under energy constraints. *Omega*, 2018, vol. 81, pp 112-122.

7. C. Gicquel, J. Cheng. A joint chance-constrained programming approach for the single-item capacitated lot-sizing problem with stochastic demand. *Annals of Operations Research*, 2018, vol. 264(1), pp 123-155.

8. M. Kchaou-Boujelben, C. Gicquel, M. Minoux. A MILP model and heuristic approach for facility location under multiple operational constraints. *Computers & Industrial Engineering*, 2016, vol. 98, pp 445-461.

9. M. Excoffier, C. Gicquel, O. Jouini. A joint chance-constraint programming approach for call centre workforce scheduling under uncertain call arrival forecasts. *Computers & Industrial Engineering*, 2016, vol. 96, pp 16-30.

10. C. Gicquel, M. Minoux. Multi-product valid inequalities for the discrete lot-sizing and scheduling problem. *Computers & Operations Research*, 2015, vol. 54, pp 12-20.

11. M. Kchaou-Boujelben, C. Gicquel, M. Minoux. A distribution network design problem in the automotive industry: MIP formulation and heuristics. *Computers & Operations Research*, 2014, vol. 52, pp 16-28.

12. C. Gicquel, M. Minoux, A. Lisser. An evaluation of semidefinite programming based approaches for discrete lot-sizing problems. *European Journal of Operational Research*, 2014, vol. 137(2), pp 498–507.

13. C. Gicquel, L. Wolsey, M. Minoux. On discrete lot-sizing and scheduling on identical parallel machines. *Optimization Letters*, 2012, vol. 6 (3), 545-557.

14. C. Gicquel, L. Hege, M. Minoux, W. van Canneyt. A discrete time exact solution approach for a complex hybrid flow-shop scheduling problem with limited-wait constraints. *Computers & Operations Research*, 2012, vol. 39 (3), 629-636.

15. C. Gicquel, M. Minoux, Y. Dallery. Exact solution approaches for the discrete lot-sizing and scheduling problem with identical parallel resources. *International Journal of Production Research*, 2011, vol. 49 (9), pp 2587-2603.

16. C. Gicquel, N. Miègeville, M. Minoux, Y. Dallery. Optimizing glass coating lines: MIP model and valid inequalities. *European Journal of Operational Research*, 2010, vol. 202 (3), pp 747-755.

17. C. Gicquel, N. Miègeville, M. Minoux, Y. Dallery. Discrete lot-sizing and scheduling using product decomposition into attributes. *Computers & Operations Research*, 2009, vol. 36, pp 2690-2698.

18. C. Gicquel, M. Minoux, Y. Dallery. On the discrete lot-sizing and scheduling problem with sequence-dependent changeover times. *Operations Research Letters*, 2009, vol. 37(1), pp 32-36.

Refereed international conference papers

1. B. Liu, C. Bissuel, F. Courtot, C. Gicquel, D. Quadri. A hierarchical decomposition approach for the optimal design of a district cooling system. *10^{th} International Conference on Operations Research and Enterprise Systems ICORES 2021*, Februrary 2021, online.

2. F. Quezada, C. Gicquel, S. Kedad-Sidhoum. Stochastic dual dynamic integer programming for the uncapacitated lot-sizing problem with uncertain demand and costs. *29^{th} International Conference on Automated Planning and Scheduling ICAPS2019*, June 2019, Berkeley, USA.

3. W. Makhlouf, M. Kchaou-Boujelben, C. Gicquel. A bi-level programming approach to locate capacitated electric vehicle charging stations. *IEEE International Conference on Control, Decision and Information Technologies CODIT 2019*, April 2019, Paris.

4. F. Quezada, C. Gicquel, S. Kedad-Sidhoum. Stochastic dual dynamic integer programming for a multi-echelon lot-sizing problem with remanufacturing and lost sales. *IEEE International Conference on Control, Decision and Information Technologies CODIT 2019*, April 2019, Paris.

5. S. Haddad-Vanier, C. Gicquel, L. Boukhatem, K. Lazri, P. Chaignon. Virtual network functions placement for defense against distributed denial of service attacks. *8^{th} International Conference on Operations Research and Enterprise Systems ICORES 2019*, Februrary 2019, Prague, Czech Republic.

6. M. Kchaou-Boujelben, C. Gicquel. Location of electric vehicle charging stations under uncertainty on the driving range. *9^{th} International Conference on Computational Logistics ICCL2018*, October 2018, Salerno, Italy.

7. C. Gicquel, S. Kedad-Sidhoum, D. Quadri. Remanufacturing planning under uncertainty: a two-stage stochastic programming approach. *International Conference on Information Systems, Logistics and Supply chain ILS2016*, June 2016, Bordeaux, France.

8. M. Excoffier, C. Gicquel, O. Jouini, A. Lisser. Scheduling problem in call centers with uncertain arrival rates forecasts: a distributionally robust approach. *4th International Conference on Operations Research and Enterprise Systems ICORES2015*, January 2015, Lisbon, Portugal.

9. M. Excoffier, C. Gicquel, O. Jouini, A. Lisser. Call center shift scheduling under uncertainty: a chance-constraint programming approach. *10th International Conference on Modelling, Optimization and Simulation MOSIM2014*, November 2014, Nancy, France.

10. C. Gicquel, J. Cheng. Solving a stochastic lot-sizing problem with a modified sample approximation approach. *44th International Conference on Computers and Industrial Engineering*, October 2014, Istanbul, Turkey.

11. C. Gicquel, M. Minoux. New multi-product valid inequalities for a discrete lot-sizing problem. *3rd International Conference on Operations Research and Enterprise Systems ICORES2014*, March 2014, Angers, France.

12. M. Excoffier, C. Gicquel, O. Jouini, A. Lisser. A stochastic programming approach for staffing and scheduling call centers with uncertain demand forecasts following continuous distributions. *3rd International Conference on Operations Research and Enterprise Systems ICORES2014*, March 2014, Angers, France.

13. C. Gicquel, M. Minoux. A tight MILP formulation based on multi-product valid inequalities for a lot-sizing problem. *International Conference on Industrial Engineering and Systems Management IESM2013*, October 2013, Rabat, Morocco.

14. M. Kchaou-Boujelben, C. Gicquel, M. Minoux. A linear relaxation based heuristic for a supply chain network design problem with minimum volume constraints. *2012 IEEE International Conference on Industrial Engineering and Engineering Management IEEM2012*, December 2012, Hong Kong, China.

15. M. Kchaou-Boujelben, C. Gicquel, M. Minoux. Considering transport flows consolidation in a network design problem. *9th International Conference of Modelling, Optimization and Simulation MOSIM2012*, June 2012, Bordeaux, France.

16. C. Gicquel, A. Lisser, M. Minoux. Tight lower bounds by semidefinite relaxations for the discrete lot-sizing and scheduling problem with sequence-dependent changeover costs. *9th International Conference of Modelling, Optimization and Simulation MOSIM2012*, June 2012, Bordeaux, France.

17. J. Cheng, C. Gicquel, A. Lisser. A second-order cone programming approximation to joint chance-constrained linear programs. *International Symposium of Combinatorial Optimisation ISCO2012*, April 2012, Athens, Greece. Published in Lecture Notes in Computer Science, 2012, vol. 7422, 71-80.

18. C. Gicquel, M. Minoux, Y. Dallery. A tight MIP formulation for the discrete lot sizing and scheduling problem with parallel resources. *9th International Conference on Computers and Industrial Engineering CIE39*, July 2009, Troyes, France.

19. C. Gicquel, M. Minoux, Y. Dallery. A tight MIP formulation for the discrete lot sizing and scheduling problem with sequence-dependent setup costs and times. *International Conference on Information Systems, Logistics and Supply Chain ILS2008*, May 2008, Madison, USA.

20. C. Gicquel, N. Miègeville, M. Minoux, Y. Dallery. Discrete Lot Sizing and Scheduling using product decomposition into attributes. *7th Conférence Francophone de Modélisation et de Simulation, MOSIM2008*, April 2008, Paris, France.

21. C. Gicquel, N. Miègeville, M. Minoux, Y. Dallery. Optimizing glass coating lines: MIP model and valid inequalities. *4th IFAC conference on Management and Control of Production and Logistics*, September 2007, Sibiu, Rumania.

Bibliography

Adams, W. P. and Sherali, H. D. (1990). Linearization strategies for a class of zero-one mixed integer programming problems. *Operations Research*, 38(2):217–226.

Aggarwal, A. and Park, J. K. (1993). Improved algorithms for economic lot size problems. *Operations Research*, 41:549–571.

Agra, A., Christiansen, M., Hvattum, L. M., and Rodrigues, F. (2018). Robust optimization for a maritime inventory routing problem. *Transportation Science*, 52(3):509–525.

Ahmed, S., King, A. J., and Parija, G. (2003). A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24.

Ahn, H.-D., Lee, D.-H., and Kim, H.-J. (2011). Solution algorithms for dynamic lot-sizing in remanufacturing systems. *International Journal of Production Research*, 49(22):6729–6748.

Alem, D. and Morabito, R. (2013). Risk-averse two-stage stochastic programs in furniture plants. *OR Spectrum*, 35:773–806.

Alem, D., Oliveira, F., and Peinado, M. C. R. (2020). A practical assessment of risk-averse approaches in production lot-sizing problems. *International Journal of Production Research*, 58(9):2581–2603.

Aloulou, M. A., Dolgui, A., and Kovalyov, M. Y. (2014). A bibliography of non-deterministic lot-sizing models. *International Journal of Production Research*, 52(8):2293–2310.

Anheuser-Busch (2021). Fairfield brewery. https://www.anheuser-busch.com/about/breweries-and-tours/fairfield-ca.html. Accessed 2021-03-26.

Attila, Ö. N., Agra, A., Akartunalı, K., and Arulselvan, A. (2017). A decomposition algorithm for robust lot sizing problem with remanufacturing option. In Gervasi, O., Murgante, B., Misra, S., Borruso, G., Torre, C. M., Rocha, A. M. A., Taniar, D., Apduhan, B. O., Stankova, E., and Cuzzocrea, A., editors, *Computational Science and Its Applications – ICCSA 2017*, pages 684–695. Springer International Publishing.

Barany, I., Van Roy, T. J., and Wolsey, L. A. (1984). Strong formulations for multi-item capacitated lot sizing. *Management Science*, 30(10):1255–1261.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115.

Belvaux, G. and Wolsey, L. A. (2001). Modelling practical lot-sizing problems as mixed-integer programs. *Management Science*, 47(7):993–1007.

Bentley (2019). Largest UK solar car port installed at Bentley factory in Crewe. https://www.bentleymotors.com/en/world-of-bentley/the-bentley-story/news/2019-news/bentley-installs-uks-largest-solar-car-port.html. Accessed 2021-03-26.

Beraldi, P. and Ruszczyński, A. (2002). A branch and bound method for stochastic integer problems under probabilistic constraints. *Optimization Methods and Software*, 17(3):359–382.

Berard, D. (2018). DDoS breach costs rise to over $2M for enterprises finds Kaspersky Lab report. https://usa.kaspersky.com/about/press-releases/2018_ddos-breach-costs-rise-to-over-2m-for-enterprises-finds-kaspersky-lab-report. Accessed 2020-12-19.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

Biel, K., Zhao, F., Sutherland, J. W., and Glock, C. H. (2018). Flow shop scheduling with grid-integrated onsite wind power using stochastic MILP. *International Journal of Production Research*, 56(5):2076–2098.

Bienstock, D. and Özbay, N. (2008). Computing robust basestock levels. *Discrete Optimization*, 5(2):389 – 414.

Bitran, G. R. and Yanasse, H. H. (1982). Computational complexity of the capacitated lot size problem. *Management Science*, 28(10):1174–1186.

Bixby, R. E. (2012). A brief history of linear and mixed-integer programming computation.

Bonferroni, C. (1936). Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62.

Bookbinder, J. H. and Tan, J.-Y. (1988). Strategies for the probabilistic lot-sizing problem with service-level constraints. *Management Science*, 34(9):1096–1108.

Brahimi, N., Absi, N., Dauzère-Pérès, S., and Nordli, A. (2017). Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, 263(3):838 – 863.

Brahimi, N., Dauzere-Peres, S., Najid, N. M., and Nordli, A. (2006). Single item lot sizing problems. *European Journal of Operational Research*, 168(1):1 – 16.

Buschkühl, L., Sahling, F., Helber, S., and Tempelmeier, H. (2010). Dynamic capacitated lot-sizing problems: a classification and review of solution approaches. *OR Spectrum*, 32:231–261.

Calafiore, G. and Campi, M. (2005). Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102:25–46.

Camargo, V. C., Toledo, F. M., and Almada-Lobo, B. (2014). HOPS–hamming-oriented partition search for production planning in the spinning industry. *European Journal of Operational Research*, 234(1):266–277.

Castellucia, C. and Le Métayer, D. (2019). Understanding algorithmic decision-making: Opportunities and challenges. Technical report, European Parliamentary Research Service.

Chen, H. (2007). A lagrangian relaxation approach for production planning with demand uncertainty. *European Journal of Industrial Engineering*, 1(4):370–390.

Cheng, J., Gicquel, C., and Lisser, A. (2019). Partial sample approximation method for chance-constrained problems. *Optimization Letters*, 13(4):657–672.

CNIL (2017). Comment permettre à l'homme de garder la main ? les enjeux éthiques des algorithmes et de l'intelligence artificielle. Synthèse du débat public animé par la CNIL dans le cadre de la mission de réflexion éthique confiée par la loi pour une république numérique.

Cyras, K., Letsios, D., Misener, R., and Toni, F. (2019). Argumentation for explainable scheduling. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*.

de Mello, T. H. and Pagnoncelli, B. K. (2016). Risk aversion in multistage stochastic programming: A modeling and algorithmic perspective. *European Journal of Operational Research*, 249(1):188–199.

di Summa, M. and Wolsey, L. A. (2008). Lot-sizing on a tree. *Operations Research Letters*, 36(1):7–13.

Drexl, A. and Kimms, A. (1997). Lot sizing and scheduling — survey and extensions. *European Journal of Operational Research*, 99(2):221 – 235.

Eppen, G. D. and Martin, R. K. (1987). Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, 35(6):832–848.

Fayaz, S. K., Tobioka, Y., Sekar, V., and Bailey, M. (2015). Bohatei: Flexible and elastic DDoS defense. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 817–832.

Ferre, L. (2010). Free SAS domestic network. https://fr.wikipedia.org/wiki/Free_(entreprise). Accessed 2020-12-19.

Florian, M. and Klein, M. (1971). Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18:12–20.

Florian, M., Lenstra, J. K., and Kan, A. H. G. R. (1980). Deterministic production planning: Algorithms and complexity. *Management Science*, 26(7):669–679.

Gahm, C., Denz, F., Dirr, M., and Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: A review and research framework. *European Journal of Operational Research*, 248(3):744–757.

Ghamari, A. and Sahebi, H. (2017). The stochastic lot-sizing problem with lost sales: A chemical-petrochemical case study. *Journal of Manufacturing Systems*, 44:53–64.

Gicquel, C. and Cheng, J. (2018). A joint chance-constrained programming approach for the single-item capacitated lot-sizing problem with stochastic demand. *Annals of Operations Research*, 264:12 3–155.

Gicquel, C., Lisser, A., and Minoux, M. (2014). An evaluation of semidefinite programming based approaches for discrete lot-sizing problems. *European Journal of Operational Research*, 237(2):498 – 507.

Gicquel, C., Miègeville, N., Minoux, M., and Dallery, Y. (2009). Discrete lot sizing and scheduling using product decomposition into attributes. *Computers & Operations Research*, 36(9):2690 – 2698.

Gicquel, C. and Minoux, M. (2015). Multi-product valid inequalities for the discrete lot-sizing and scheduling problem. *Computers & Operations Research*, 54:12 – 20.

Gicquel, C., Vanier, S., and Papadimitriou, A. (2020). Optimal deployment of virtual network functions for securing telecommunication networks against distributed denial of service attacks: a robust optimization approach.

Golari, M., Fan, N., and Jin, T. (2017). Multistage stochastic optimization for production-inventory planning with intermittent renewable energy. *Production and Operations Management*, 26(3):409–425.

Gomory, R. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278.

Gorissen, B. L., Yanikoglu, I., and den Hertog, D. (2015). A practical guide to robust optimization. *Omega*, 53:24 – 137.

Guan, Y., Ahmed, S., and Nemhauser, G. L. (2009). Cutting planes for multistage stochastic integer programs. *Operations Research*, 57(2):287–298.

Guan, Y., Ahmed, S., Nemhauser, G. L., and Miller, A. J. (2006). A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming*, 105(1):55–84.

Guan, Y. and Miller, A. J. (2008). Polynomial-time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research*, 56(5):1172–1183.

Guide, V. D. R. (2000). Production planning and control for remanufacturing: industry practice and research needs. *Journal of Operations Management*, 18(4):467–483.

Guide, V. D. R., Jayaraman, V., and Srivastava, R. (1999). Production planning and control for remanufacturing: a state-of-the-art survey. *Robotics and Computer-Integrated Manufacturing*, 15(3):221–230.

Hjelmeland, M. N., Zou, J., Helseth, A., and Ahmed, S. (2018). Nonconvex medium-term hydropower scheduling by stochastic dual dynamic integer programming. *IEEE Transactions on Sustainable Energy*, 10(1):481–490.

Hu, Z. and Hu, G. (2016). A two-stage stochastic programming model for lot-sizing and scheduling under uncertainty. *International Journal of Production Economics*, 180:198–207.

Jakaria, A. H. M., Yang, W., Rashidi, B., Fung, C., and Rahman, M. A. (2016). Vfence: A defense against distributed denial of service attacks using network function virtualization. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 431–436.

Jans, R. and Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, 177(3):1855 – 1875.

Jans, R. and Degraeve, Z. (2008). Modeling industrial lot sizing problems: a review. *International Journal of Production Research*, 46(6):1619–1643.

Jayaraman, V. (2006). Production planning for closed-loop supply chains with product recovery and reuse: an analytical approach. *International Journal of Production Research*, 44(5):981–998.

Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., and Pulleyblank, W. R. (2008). *50 years of integer programming: 1958-2008*. Springer.

Karimi, B., Fatemi Ghomi, S., and Wilson, J. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31(5):365 – 378.

Kchaou Boujelben, M., Gicquel, C., and Minoux, M. (2014). A distribution network design problem in the automotive industry: MIP formulation and heuristics. *Computers & Operations Research*, 52:16–28.

Kchaou-Boujelben, M., Gicquel, C., and Minoux, M. (2016). A MILP model and heuristic approach for facility location under multiple operational constraints. *Computers & Industrial Engineering*, 98:446–461.

Keller, F., Braunreuther, S., and Reinhart, G. (2016). Integration of on-site energy generation into production planning systems. *Procedia CIRP*, 48:254–258.

Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307.

Kilic, O. A., Tunc, H., and Tarim, S. A. (2018). Heuristic policies for the stochastic economic lot sizing problem with remanufacturing under service level constraints. *European Journal of Operational Research*, 267(3):1102–1109.

Küçükyavuz, S. (2012). On mixing sets arising in chance-constrained programming. *Mathematical Programming*, 132:31–56.

Knight, S., Nguyen, H. X., Falkner, N., Bowden, R., and Roughan, M. (2011). The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775.

Knight, S., Nguyen, H. X., Falkner, N., Bowden, R., and Roughan, M. (2013). The internet topology zoo. http://www.topology-zoo.org/index.html. Accessed 2020-12-19.

Kuik, R., Salomon, M., and van Wassenhove, L. N. (1994). Batching decisions: structure and models. *European Journal of Operational Research*, 75(2):243 – 263. Lotsizing models for production planning.

Loparic, M., Pochet, Y., and Wolsey, L. A. (2001). The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming*, 89(3):487–504.

Luedtke, J. and Ahmed, S. (2008). A sample approximation approach for optimization with probabilistic constraints. *SIAM Journal on Optimization*, 19(2):674–699.

Luedtke, J., Ahmed, S., and Nemhauser, G. (2010). An integer programming approach for linear programs with probabilistic constraints. *Mathematical Programming*, 122:247–272.

Macedo, P. B., Alem, D., Santos, M., Junior, M. L., and Moreno, A. (2016). Hybrid manufacturing and remanufacturing lot-sizing problem with stochastic demand, return, and setup costs. *The International Journal of Advanced Manufacturing Technology*, 82(5-8):1241–1257.

Masmoudi, O., Yalaoui, A., Ouazene, Y., and Chehade, H. (2017). Lot-sizing in a multi-stage flow line production system with energy consideration. *International Journal of Production Research*, 55(6):1640–1663.

Moreno, A., Alem, D., Ferreira, D., and Clark, A. (2018). An effective two-stage stochastic multi-trip location-transportation model with social concerns in relief supply chains. *European Journal of Operational Research*, 269(3):1050–1071.

Nemirovski, A. and Shapiro, A. (2005). Scenario approximations of chance constraints. In *Probabilistic and Randomized Methods for Design Under Uncertainty*, pages 3–48. Springer.

Nemirovski, A. and Shapiro, A. (2006). Convex approximations of chance constrained programs. *SIAM Journal of Optimization*, 17:969–996.

Özdamar, L. and Birbil, S. I. (1999). A hierarchical planning system for energy intensive production environments. *International Journal of Production Economics*, 58(2):115–129.

Pechmann, A. and Zarte, M. (2017). Procedure for generating a basis for PPC systems to schedule the production considering energy demand and available renewable energy. *Procedia CIRP*, 64:393–398.

Pereira, M. V. and Pinto, L. M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375.

Philpott, A. and de Matos, V. (2012). Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research*, 218(2):470–483.

Pinson, P., Madsen, H., Nielsen, H. A., Papaefthymiou, G., and Klöckl, B. (2009). From probabilistic forecasts to statistical scenarios of short-term wind power production. *Wind Energy*, 12(1):51–62.

Piperagkas, G., Konstantaras, I., Skouri, K., and Parsopoulos, K. (2012). Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics. *Computers & Operations Research*, 39(7):1555 – 1565.

Pochet, Y. and Wolsey, L. A. (2006). *Production planning by mixed-integer programming*. Springer.

Quezada, F., Gicquel, C., and Kedad-Sidhoum, S. (2019). Stochastic dual dynamic integer programming for a multi-echelon lot-sizing problem with remanufacturing and lost sales. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT 2019)*, pages 1254–1259.

Quezada, F., Gicquel, C., and Kedad-Sidhoum, S. (2019). A stochastic dual dynamic integer programming for the uncapacitated lot-sizing problem with uncertain demand and costs. *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2020)*, 29(1):353–361.

Quezada, F., Gicquel, C., and Kedad-Sidhoum, S. (2020a). Combining polyhedral approaches and stochastic dual dynamic integer programming for solving the uncapacitated lot-sizing problem under uncertainty.

Quezada, F., Gicquel, C., Kedad-Sidhoum, S., and Vu, D. Q. (2020b). A multi-stage stochastic integer programming approach for a multi-echelon lot-sizing problem with returns and lost sales. *Computers & Operations Research*, 116:104865.

Rapine, C., Goisque, G., and Akbalik, A. (2018a). Energy-aware lot sizing problem: Complexity analysis and exact algorithms. *International Journal of Production Economics*, 203:254–263.

Rapine, C., Penz, B., Gicquel, C., and Akbalik, A. (2018b). Capacity acquisition for the single-item lot sizing problem under energy constraints. *Omega*, 81:112–122.

Rockafellar, R. T. and Uryasev, S. P. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41.

Rodoplu, M., Arbaoui, T., and Yalaoui, A. (2019). Single item lot sizing problem under renewable energy uncertainty. In *9^{th} IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019*.

Roupin, F. (2004). From linear to semidefinite programming: An algorithm to obtain semidefinite relaxations for bivalent quadratic problems. *Journal of Combinatorial Optimization*, 8:469–493.

Salomon, M., Kroon, L. G., Kuik, R., and Van Wassenhove, L. N. (1991). Some extensions of the discrete lotsizing and scheduling problem. *Management Science*, 37(7):801–812.

Salomon, M., Solomon, M. M., Van Wassenhove, L. N., Dumas, Y., and Dauzère-Pérès, S. (1997). Solving the discrete lotsizing and scheduling problem with sequence dependent set-up costs and set-up times using the travelling salesman problem with time windows. *European Journal of Operational Research*, 100(3):494 – 513.

Szymkowski, S. (2018). Bentley will add 10000 more solar panels to its plant. https://www.motorauthority.com/news/1116198_bentley-will-add-10000-more-solar-panels-to-its-plant. Accessed 2021-03-26.

Tang, L., Che, P., and Liu, J. (2012). A stochastic production planning problem with nonlinear cost. *Computers & Operations Research*, 39(9):1977–1987.

Tempelmeier, H. (2007). On the stochastic uncapacitated dynamic single-item lotsizing problem with service level constraints. *European Journal of Operational Research*, 181(1):184 – 194.

Tempelmeier, H. (2013). *Stochastic Lot Sizing Problems*, pages 313–344. Springer New York, New York, NY.

Tempelmeier, H. and Herpers, S. (2011). Dynamic uncapacitated lot sizing with random demand under a fillrate constraint. *European Journal of Operational Research*, 212(3):497 – 507.

unpu, K. and Ball, P. (2013). Energy efficient manufacturing from machine tools to manufacturing systems. *Procedia CIRP*, 7:634 – 639.

Uzel, E. (2004). A mathematical modeling approach to energy cost saving in a manufacturing plant. Master's thesis, Izmir Institute of Technology.

van Eijl, C. and van Hoesel, C. (1997). On the discrete lot-sizing and scheduling problem with Wagner-Whitin costs. *Operations Research Letters*, 20(1):7 – 13.

van Hoesel, C. P. M. and Wagelmans, A. P. M. (1996). An O(T^3) algorithm for the economic lot-sizing problem with constant capacities. *Management Science*, 42(1):142–150.

van Hulst, D., den Hertog, D., and Nuijten, W. (2017). Robust shift generation in workforce planning. *Computational Management Science*, 14:115–134.

Vargas, V. (2009). An optimal solution for the stochastic version of the Wagner–Whitin dynamic lot-size model. *European Journal of Operational Research*, 198(2):447 – 451.

Vyakaranam, N. and Krishna, D. (2018). 5G: Network as a service - how 5G enables the telecom operators to lease out their network. https://netmanias.com/en/post/blog/13311/5g/5g-network-as-a-service-how-5g-enables-the-telecom-operators-to-lease-out-their-network. Accessed 2020-12-19.

Wagelmans, A., Van Hoesel, S., and Kolen, A. (1992). Economic lot sizing: an $\mathcal{O}(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40(1-supplement-1):S145–S156.

Wagner, H. M. and Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96.

Wichmann, M. G., Johannes, C., and Spengler, T. S. (2019a). Energy-oriented lot-sizing and scheduling considering energy storages. *International Journal of Production Economics*, 216:204–214.

Wichmann, M. G., Johannes, C., and Spengler, T. S. (2019b). An extension of the general lot-sizing and scheduling problem (GLSP) with time-dependent energy prices. *Journal of Business Economics*, 89(5):481–514.

Zangwill, W. I. (1969). A backlogging model and a multi-echelon model of a dynamic economic lot size production system-a network approach. *Management Science*, 15(9):506–527.

Zhang, M., Küçükyavuz, S., and Goel, S. (2014). A branch-and-cut method for dynamic decision making under joint chance constraints. *Management Science*, 60(5):1317–1333.

Zhao, C. and Guan, Y. (2014). Extended formulations for stochastic lot-sizing problems. *Operations Research Letters*, 42(4):278–283.

Zou, J., Ahmed, S., and Sun, X. A. (2019). Stochastic dual dynamic integer programming. *Mathematical Programming*, 175:461–502.