LOT-SIZING AND SCHEDULING IN FLOW SHOPS

by

Murat Güngör

B.S., Industrial Engineering, Boğaziçi University, 2005

M.S., Mathematics, Boğaziçi University, 2009

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Industrial Engineering

Boğaziçi University

2018

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor Ali Tamer Ünal for his guidance throughout my PhD study and for generously sharing his insights derived from years of practical experience. I am indebted to Z. Caner Taşkın for being my unofficial co-advisor and for his valuable contribution to our joint work. I am grateful to Hande Küçükaydın, Necati Aras, and Erinç Albey for taking part in my thesis committee. I am much obliged to Bulut Aslan for helping me with the ICRON software. I thank all my professors and colleagues in the department for everything I have learned from them and for the friendly atmosphere they provided. Last but not least, I thank my beloved family for their endless support.

ABSTRACT

LOT-SIZING AND SCHEDULING IN FLOW SHOPS

Lot-sizing is concerned with the product quantities, whereas scheduling considers the specific machines and product sequences. *Lot-sizing and scheduling* refers to procedures that aim to solve these interdependent problems simultaneously. In this thesis, we investigate three lot-sizing and scheduling problems encountered in flow shops. The first one is a single-level parallel machine problem with a secondary resource and cumulative demand, where the objective is to minimize the number of setups and teardowns. The second examines a two-machine flow shop with a serial bill-of-materials. The goal is to first minimize back orders and then the number of changeovers. The third problem arises in the two-machine flow shop as well. The question is whether there exists a feasible production plan to satisfy all requirements. For the first two problems, we suggest reformulations, prove several optimality conditions, and carry out computational experiments. For the last one, we develop a polynomial algorithm after giving some theoretical results.

ÖZET

AKIŞ TİPİ ATÖLYELERDE LOTLAMA VE ÇİZELGELEME

Lotlamanın konusu üretim miktarlarının belirlenmesi, çizelgelemenin konusu ise işlerin belirli makineler üzerinde sıralanmasıdır. *Lotlama ve çizelgeleme*, birbirine bağlı bu iki problemi aynı anda çözmeye yönelik yöntemleri ifade eden genel bir tabirdir. Biz bu tezde akış tipi atölyelerde karşılaşılan üç lotlama ve çizelgeleme problemini inceliyoruz. Bunların ilki; tek seviyeli, paralel makineli, ikincil kaynaklı, birikimli talepli bir problem. Amaç toplam takım-söküm sayısını en aza indirmek. İkincisi; iki makineli, seri ürün ağaçlı bir akış tipi atölyede geçiyor. Birincil hedef gecikmiş sipariş sayısını en küçültmek, ikincil hedef üründen ürüne geçişlerin toplam sayısını en küçültmek. Üçüncü problemde, yine iki makineli akış tipi atölyede, "Hiç gecikmiş siparişin olmadığı bir üretim planı var mıdır?" sorusuna cevap arıyoruz. İlk iki problem için yeniden biçimlendirmeler öneriyoruz, çeşitli en iyilik koşulları gösteriyoruz ve bilgisayar deneyleri gerçekleştiriyoruz. Sonuncu problem için ise birtakım teorik sonuçlar verdikten sonra polinom zamanlı bir algoritma geliştiriyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

b_{it}	back order of end item i at the end of period t
c_1	per unit and per period back order cost
c_2	changeover cost
d_{it}	demand for end item i in period t
$d(i,t)$	demand for end item i in period t
i	index for secondary resource types (Ch. 2) *or* item index (Ch. 3) *or* end item index (Ch. 4)
I	set of all end items
j	machine index (Ch. 2) *or* intermediate item index (Ch. 4)
$j(i)$	index of the unique predecessor of i
J	set of all intermediate items
m	machine index
M	number of machines
$r(i)$	(possibly fractional) number of units of intermediate item $j(i)$ required to produce one unit of end item i
s_{it}^{+}	number of setups required for secondary resources of type i in transition from period $t-1$ to t
s_{it}^{-}	number of teardowns required for secondary resources of type i in transition from period $t-1$ to t
s_{mit}	stock of item i in machine m at the end of period t
t	time index
T	number of time periods
$\overline{x_i}$	demand as average number of secondary resources of type i to be used in a period
x_{it}	number of secondary resources of type i used in period t
x_{mit}	production of item i in machine m in period t

y_{1jt}	1 if item j is produced on machine 1 in period t, and 0 otherwise
y_{2it}	1 if item i is produced on machine 2 in period t, and 0 otherwise
y_{ijt}	1 if machine j is set up for a secondary resource of type i in period t, and 0 otherwise
y_{mit}	binary variable for the setup state of item i in machine m in period t
z^{+}_{ijt}	1 if a setup for a secondary resource of type i is performed on machine j in transition from period $t-1$ to t, and 0 otherwise
z^{-}_{ijt}	1 if a teardown for a secondary resource of type i is performed on machine j in transition from period $t-1$ to t, and 0 otherwise
z_{mt}	changeover variable in machine m in period t

LIST OF ACRONYMS/ABBREVIATIONS

BOM	Bill-of-Materials
CLSP	Capacitated Lot-Sizing Problem
CPU	Central Processing Unit
CSLP	Continuous Setup Lot-Sizing Problem
DLSP	Discrete Lot-Sizing and Scheduling Problem
EDD	Earliest Due Date
GLSP	General Lot-Sizing and Scheduling Problem
LP	Linear Programming
MIP	Mixed-Integer Linear Programming
PLSP	Proportional Lot-Sizing and Scheduling Problem

1. INTRODUCTION

Managers of an enterprise are to make many decisions: What due date should be assigned to a new customer order? Do the prices of end-products need to be modified? Shall an order be placed for some raw material? Any action to be taken about installed capacity? What to produce today, on which machine and how much? Planning, in general, aims to find good answers to such questions.

Effect of a decision is not always observed on the same day. For instance, it may take months for an import to be delivered after a purchase order is released; similarly for the realization of capacity revisions. Such lead times together with phenomena like seasonality of demand essentially determine the planning horizon.

Complexity of the problem—not only in the computational sense—usually renders a monolithic approach impossible. Therefore one speaks of long-, medium- and short-term planning, to be performed in an integrative way. While medium-term plans are aggregate, comprising several months to a year in most cases, short-term or operational plans are detailed, seeking for a few weeks' time a profitable answer to when, where and how much to produce. As selling prices normally remain fixed for a short period of time, "profitable" here is to be understood as "least costly."

This doctoral thesis is chiefly concerned with short-term planning in a prevalent production environment called flow shops, in which products undergo in the same order a sequence of operations, with a progressive increase in variety. Flow shops are quite common in many industries, examples of which will be given shortly. By *hybrid flow shop* (also called *flexible flow shop*) one understands a flow shop with each stage having possibly several machines in parallel.

Main goal of short-term planning is to find a balance between setup and inventory holding costs while respecting the due dates as much as possible. One extreme would be to produce for long durations the same or similar items to reduce setup costs; this

naturally leads to larger inventories. By inventory holding cost, we not only understand the related opportunity cost; product diversity and dynamic market conditions are also issues in this regard. The other extreme would be to adopt an aggressive inventory-minimizing strategy, ignoring setups entirely. In this case, however, in addition to direct costs, the time spent for changeovers imply an effective decrease in installed production capacity. Hence, a compromising solution is to be sought.

Setups are usually dictated by the need for cleaning and preparation of the production line. These are sometimes so time-consuming and costly that in order to avoid them even prestigious firms accept the risk of facing severe consequences. A few years ago, this was the case for a renowned Turkish company specialized in delicatessen. The ministry had announced that some of company's red-meat-only products contained traces of poultry. Later on, company officials declared that on the same line both kinds of products were being produced, and that it was almost impossible to fully clean the system during changeovers, whence some samples unfortunately failed to pass those sensible food tests.

1.1. Real-Life Examples of Flow Shops

The problem of short-term planning in (hybrid or pure) flow shops is a practically relevant one, and herein lies our motivation. In subsequent paragraphs, we shall discuss three real-life examples of flow shops, namely tissue paper, pharmaceutical, and aluminum alloy wheel production.

Tissue paper production [1] consists of two major phases: paper production and converting. In the first phase, chemical compounds such as cellulose are mixed in huge containers. The mixture then runs through a line where it is dried and flattened. Resulting thin paper is coiled up to obtain massive bobbins whose width is approximately two-and-a-half meters and whose weight differs from half a ton up to four tons depending on the paper type. There may be tens of paper types, each being used in production of many end items. In the second phase, paper bobbins are loaded on converting machines where the paper is cut into required sizes and packaged. Each

machine is composed of one single line in which no interruption between cutting and packaging exists. Converting machines can be divided into four major groups: toilet paper and paper towel (where cylindrical items are produced), tissue paper, napkin (serviette), and facial tissue machines.

Paper production in the first phase is performed on large dedicated machines that can process many paper types. Product changes on those machines necessitate significant setups because remnants of chemicals in the mixing containers should be removed so as to prevent mixture with the chemicals of the next product, whereby the nature of the next product is not harmed. Those setups require long durations, workforce, and high energy consumption.

Pharmaceutical manufacturing is a nice example of flow shops, too. First, medicaments are produced in bulk, by mixing certain chemicals in large tanks. Second, if the medicament is in solid form, either it is given the required shape as a pill and then coated by a thin film, or it is directly encapsulated. Third, pills and capsules are packaged. If the medicament is in liquid form, it is bottled right after mixing.

Clearly, cleaning of tanks during changeovers is much more critical in this setting. Moreover, work-in-process inventories cannot wait a long time for processing, otherwise the medicament in question can be spoiled or lose its effect. A third issue special to pharmaceutical manufacturing is that there might be restrictions put by legislative bodies on lot sizes; that is, the choice of lot size is not totally arbitrary in general.

Finally, let us describe aluminum alloy wheel production briefly: A furnace heats high grade aluminum ingots, and the molten metal is injected by casting machines into steel molds. After casting, wheels undergo a complex heat treatment process. Then rough edges are trimmed off. The shape now finalized, a few tests are performed. Next, wheels proceed to painting line and packaged appropriately.

For machining, every item needs, apart from the wheel that determines its model, a set of specific equipments. This brings about the issue of secondary scarce resources.

Moreover, setups become sequence-dependent even though for each particular equipment they are constant.

Many branches of food production (one of which has been mentioned before), glass production, etc. are similarly examples of flow shops, all with their own peculiarities.

1.2. General Problem Definition

We consider a hybrid (flexible) flow shop. Products undergo in the same order a number of stages, each composed of several machines in parallel. The question is when and how much to produce in the short term to fulfill a given collection of requirements (customer orders and forecasts).

Let C denote the set of all raw materials, work-in-process, and end-products. Bills-of-materials (BOMs) altogether can be thought of as a directed acyclic (possibly disconnected) graph G with node set C. The arcs are weighted according to relevant gozinto factors. If every node in this graph has at most one predecessor, G is said to be divergent; if every node has at most one successor, it is said to be convergent. Otherwise one speaks of a general structure. The special case in which G is both divergent and convergent is called serial. A positive integer called BOM level can be assigned to each element of C as follows: the nodes in G with no successor are said to be of level one; deleting these together with the arcs thereon, the nodes in the remaining graph with no successor are said to be of level two, and so on. This is precisely the same procedure used to topologically order any directed acyclic graph. Thus, end-products are of level one while raw materials assume the largest levels.

Let us denote by I the elements of C with at least one predecessor in G. So I is the set of all items produced—intermediate or final (end). Every such item is obtained as a result of a certain operation. Let M be the collection of all machines. A machine can perform several operations. We assume that the sets of operations performed by two machines either coincide or are disjoint; in the former case, we say that the machines in question are equivalent. This equivalence relation partitions M. Let $\overline{M}$ denote the

resulting set of equivalence classes. Machines in a class can be identical, of different speeds, or unrelated. Eligibility constraints can be expressed by a mapping $i \mapsto \mu(i)$ from I to $\overline{M}$. As the environment is a flow shop, machines can perform operations that produce items of only a particular BOM level; symbolically, $l(i) = l(i')$ for all items i, i' with $\mu(i) = \mu(i')$, where l stands for BOM level. Machines that produce items of the same level form a production stage.

The demand is dynamic unlike the economic order quantity model or the economic lot scheduling problem. A finite number of requirements must be satisfied without back-ordering. Let q_r and d_r denote the quantity and due date of requirement r, respectively, and i_r the item associated with requirement r. Distinct requirements may belong to one and the same item.

For simplicity of notation, we suppose that all machines in a stage are equivalent and identical. We denote by a_i the unit production time for each item i. Procurement lead times for raw materials are assumed to be zero. For items produced in the same stage, either sequence-independent or sequence-dependent setup times (s_i or s_{ik}) are valid. In the former case, setup time depends only on the item whose production is to start, whereas in the latter it changes also with respect to the immediately preceding item.

Minimum batch sizes m_i are the minimum quantities to be produced once a machine is set up for a specific item i. Such a restriction might stem from technological constraints, shop floor requirements, or a managerial decision. Another parameter is the transfer batch size b_i. If hundred units of an item is processed at a stage, then in case $b_i = 1$ each one of these hundred units become available for the next stage or customer as soon as its processing is finished. This is referred to as job availability or open production, and is typically the situation when two consecutive stages are linked by means of a conveyor. The case $b_i > 1$ is referred to as batch availability or closed production. This is quite common for initial stages of production in process industries, and for shops where work-in-process is transferred on a pallet by the help of a forklift. We suppose $m_i = 0$ and $b_i = 1$ unless stated otherwise.

The simplest task is to find out whether or not all requirements could be met on time—this is the feasibility problem. Apart from its direct relation to optimization problems with objectives like maximum lateness, feasibility problem is important in its own right. Table 1.1 gives a list of basic problem parameters together with short explanations. All parameters are integers except gozinto factors. Let us emphasize that the planning horizon is not discretized a priori; in other words, we assume a continuous time axis.

Table 1.1. General problem parameters.

Symbol(s)	Explanation
C	set of all raw materials, work-in-process, and end-products
G	bill-of-materials: a directed acyclic graph with node set C
I	intermediate or final items: subset of C consisting of elements with at least one predecessor in G (indices: i, k)
M	set of all machines (index: j)
R	set of all requirements (index: r)
i_r	item associated with requirement r
q_r	quantity of requirement r
d_r	due date of requirement r
a_i	unit production time for item i
s_i or s_{ik}	sequence-independent or sequence-dependent setup times
m_i	minimum batch size for item i
b_i	transfer batch size for item i

1.3. Literature Review

Lot-sizing is concerned with the product quantities, whereas scheduling considers the specific machines and product sequences. These two problems are interdependent, especially when setup times are sequence-dependent. The relationship holds true even if setup times are sequence-independent. *Lot-sizing and scheduling* refers to procedures that aim to solve these two problems simultaneously.

As lot-sizing calculates how many units of a product to produce in a period, it is usually implicit in this terminology that the planning horizon is divided into a finite number of time buckets—consider the classical problem of Wagner and Whitin [2], namely the single-item uncapacitated lot-sizing problem, and its many extensions analyzed by Pochet and Wolsey [3]. In scheduling, to the contrary, the time scale is often continuous, as can be seen from Pinedo's textbook [4]. Most lot-sizing and scheduling research is based on discrete-time or hybrid formulations. Copil *et al.* [5] present a comprehensive survey of this literature in a structured way.

Inevitably, bucket-based formulations are only approximations of reality. Therefore, researchers assuming a discrete time scale often attempt to make up for this deficiency at least partially. The most common issues include setup carryover (linked lot sizes), setup crossover (period-overlapping setups), and long setups covering a number of consecutive periods. For single-level production, Suerie [6] has significant contributions to the literature; the area of research is still active [7, 8]. For multiple stages, there is the additional issue of lead time. Alternative approaches are proposed to deal with synchronization between stages [9, 10].

Researchers starting directly with a continuous time scale, on the other hand, usually ignore the problem's lot-sizing aspect. Even if group technology assumption is dropped and nonpermutation schedules are allowed as in Shen *et al.* [11], jobs are still regarded as irreducible entities, and the problem rather belongs to the realm of pure scheduling. The situation is the same for the multi-level problems discussed by Jordan [12]. In lot streaming, lots (jobs) can be split into sublots to be processed simultaneously over different machines [13], but the review of Cheng *et al.* [14] reveals that very few papers consider flow shops with multiple product types and variable sublots, and in their problem definition there is no external demand to be fulfilled.

The problem described in §1.2 is computationally difficult to solve in many aspects. It is well-known that sequence-dependent setups lead to the traveling salesman problem. For sequence-independent setup times, the single-machine feasibility problem is shown to be (binary) NP-complete by Bruno and Downey [15]. Subsequently, Cheng

et al. [16] improve this result by showing that the minimization of maximum lateness, which is polynomially equivalent to the feasibility problem, is strongly NP-hard. Synchronization of multiple stages also contribute to the problem's complexity. Lenstra *et al.* [17] show that the two-machine flow shop problem with due date constraints is strongly NP-hard. Later on, some complexity results for special cases are provided by Koulamas [18] and Lin [19].

Potts and van Wassenhove [20] propose a general model for integrating scheduling with lot-sizing, but they do not treat multiple stages in depth. Floudas and Lin [21] review continuous- and discrete-time approaches for scheduling of chemical processes. Quadt and Kuhn [22] present a conceptual framework for lot-sizing and scheduling of flexible flow lines: they start with bottleneck planning for part families, then roll out the resulting schedule to other production stages, and finally make product-to-slot assignments. More details can be found in Quadt's book [23]. Stefansson *et al.* [24] solve a large real-world scheduling problem from a pharmaceutical company. The production process consists of four stages. They decompose the problem in two parts, and compare continuous- and discrete-time representations for solving the individual parts. Camargo *et al.* [25] suggest three mathematical models with different time scales for a two-stage lot-sizing and scheduling problem present in process industries. The upstream production continuously feeds the downstream machines. Seeanner *et al.* [26] combine the principles of variable neighborhood decomposition search and the fix-and-optimize heuristic to solve a multi-level lot-sizing and scheduling problem in which the bottleneck stage may shift dynamically.

1.4. Some Observations

We list below some observations about the problem defined in §1.2.

- Feasibility problem is computationally equivalent to the minimization of maximum lateness ($L_{\max}$) or tardiness ($T_{\max}$). Indeed, $L^*_{\max}$ is equal to the smallest integer t^* for which the problem with due dates $d_r + t^*$ is feasible, and $T^*_{\max} = \max\{L^*_{\max}, 0\}$. Feasibility problem is equivalent to makespan minimiza-

tion ($C_{\max}$) as well. Consider the decision version of the makespan minimization problem: Does there exist a schedule such that $C_{\max} \leq k$? This question has an affirmative answer if and only if the problem with due dates redefined as $d'_r := \min\{d_r, k\}$ is feasible.

- We may assume, without loss of generality, that all requirements are of size 1. In fact, each requirement (i, q, d) can be split into q requirements $(i, 1, d), (i, 1, d - a_i), \ldots, (i, 1, d-(q-1)a_i)$; the resulting problem is equivalent to the original one. This can be expressed by saying that every lot-sizing and scheduling problem can be written equivalently as a pure scheduling problem (with possibly a very large number of jobs).
- In the single-machine setting with sequence-independent setup times, requirements need not be split in a feasible schedule [27,28]. In other words, there exists a feasible schedule such that every requirement is satisfied from one and only one lot (by lot we mean any maximal interval of production devoted to a particular item). This statement is no longer true for flow shops, even in case of two machines, zero setup times, and serial BOM: Suppose there are two end items i, k produced from intermediate items i', k' with gozinto factors 1. Unit production times for i', k', i, k are 1, 2, 2, 1. If there are two requirements for i and k with quantities 2 and 1, and due dates 6 and 4, respectively, then the only feasible schedule is the one depicted in Figure 1.1. Note that the requirement for i is satisfied from two different lots.

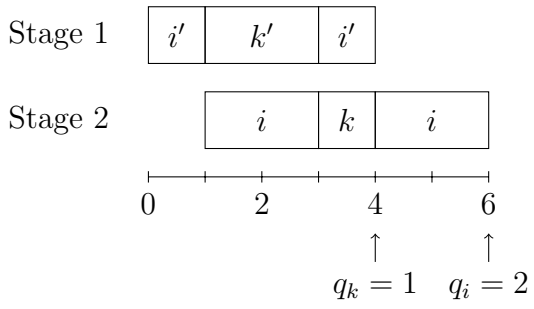

Figure 1.1. Gantt chart of the feasible schedule associated with the example showing that requirements may need to be split in flow shops.

1.5. Outline of the Thesis

The thesis is mainly composed of three fairly independent chapters to follow. In Chapter 2, we investigate a parallel machine multi-item lot-sizing and scheduling problem with a secondary resource, in which demands are given for the entire planning horizon rather than for every single period. This is an abstraction of a single-level problem encountered in a real-life flow shop. All-or-nothing assumption of the discrete lot-sizing and scheduling problem is valid so that a machine is either idle or works at full capacity in a period. The objective is to minimize the number of setups and teardowns. In Chapter 3, we consider the problem defined in §1.2 for a two-machine flow shop where setup times are ignored. More precisely, there are two stages of production, each consisting of one machine. The items produced in the first and second stages are referred to as intermediate and end items, respectively. There are a number of requirements for end items. Different requirements may be associated with the same end item. Although setup times are ignored, a constant cost is incurred when production on a machine is switched from one item to another. We call this a changeover. Back orders are allowed to make sure that any instance will be feasible. The objective is to first minimize back orders and then the number of changeovers. In Chapter 4, we examine the complexity of essentially a simplified version of the problem studied in Chapter 3, where changeovers are not a matter of concern anymore and the all-or-nothing assumption is valid. The question is whether there exists a feasible production plan to satisfy all requirements.

Chapter 2 of this thesis is actually a joint paper with Ali Tamer Ünal and Z. Caner Taşkın that was sent to the International Journal of Production Research. After two revisions, it was accepted on November 7, 2017, and published online on December 5 in the same year. Its DOI (Digital Object Identifier) is 10.1080/00207543.2017.1406675.

2. A PARALLEL MACHINE LOT-SIZING AND SCHEDULING PROBLEM WITH A SECONDARY RESOURCE AND CUMULATIVE DEMAND

We investigate a parallel machine multi-item lot-sizing and scheduling problem with a secondary resource, in which demands are given for the entire planning horizon rather than for every single period. All-or-nothing assumption of the discrete lot-sizing and scheduling problem is valid so that a machine is either idle or works at full capacity in a period. The objective is to minimise the number of setups and teardowns. We prove that the problem is NP-hard, and present two equivalent formulations. We show some properties of the optimal objective value, give optimality conditions, and suggest a heuristic algorithm. We discuss and formulate two possible extensions related to real-life applications. Finally, we carry out computational experiments to compare the two formulations, to determine the effect of our proposed modeling improvements on solution performance, and to test the quality of our heuristic.

2.1. Introduction

Lot-sizing and scheduling is a practical problem arising in many production environments. Although there are similarities between problems originating in different settings, each production process has its own characteristics. Motivated by a real-life application, we address in this chapter a novel parallel machine multi-item lot-sizing and scheduling problem with a secondary resource. The novelty lies in the assumption that demands are given for the entire planning horizon rather than for every single period.

The problem can be described as follows: A number of items are to be produced on identical parallel machines. In order to produce an item, a piece of equipment (secondary resource) specific to that item must be installed in the machine. The planning horizon is divided into buckets for which all-or-nothing assumption is valid;

that is to say, a machine is either idle or works at full capacity in a period. Total demand quantity of each item is known in advance. A constant cost is incurred for every setup and teardown (mount and dismount) of equipments on machines, and the objective is to minimize the total number of setups and teardowns.

Our motivation to study this problem comes from aluminum alloy wheel production, which actually takes place in a flow shop consisting mainly of casting, heat treatment, painting, and packaging stages. Planning of the foundry leads essentially to the problem described above. In particular, items are the wheels to be cast, and equipments are the molds used together with casting machines. Although planning time buckets are shifts, due to a common practice in automotive industry, daily shipment quantities to the car manufacturers are managed based on maintaining a weekly delivered quantity budget rather than fulfilling specific day-based or shift-based due dates and delivery quantities. In other words, demand information is cumulative. At the beginning of each week, all molds are dismounted, and one shift is devoted to cleaning of machines for quality assurance. Thus, different weeks are somewhat independent from each other and constitute natural planning horizons by themselves.

As the demand to be fulfilled is cumulative, there is no need for inventory balance equations. The main goals are to decrease the number of changeovers, and to increase thereby the utilization of resources. Therefore, inventory-related costs are not taken into account. In other words, excess production is not penalized unlike most lot-sizing models in the literature. Number of molds is not restricted. Setup and teardown times are ignored because they are more or less constant and do not constitute a large portion of the bucket capacity. In spite of all these simplifying assumptions, the problem turns out to be NP-hard, as we shall prove.

Lot-sizing and scheduling is a broad subject, which has been studied extensively in the literature. Besides books addressing different aspects of the topic [3,6,12,23,29,30], there are several comprehensive surveys [5, 31–36].

Most models assume that the planning horizon is divided into buckets. We shall refer to them as discrete-time formulations. These are usually derived from a few basic models, namely discrete lot-sizing and scheduling problem (DLSP), continuous setup lot-sizing problem (CSLP), and proportional lot-sizing and scheduling problem (PLSP). Purely continuous formulations are rarely proposed. However, hybrid formulations, namely variants of general lot-sizing and scheduling problem (GLSP) and capacitated lot-sizing with sequence-dependent setups (CLSD), are quite popular. See Drexl and Kimms [31] and Copil *et al.* [5] for a thorough discussion of the basic models just mentioned.

Lasdon and Terjung [37] investigate for a major tire manufacturer a problem closely related to ours. They consider multiple products with dynamic demand to be produced on identical parallel machines. In order to produce an item, a die must be installed in the machine, and the number of dies available in each period is limited. Their model is based on DLSP. The main decision variables are the numbers of machines to be used for production of an item in a period. For the Lasdon-Terjung model, Eppen and Martin [38] give an extended reformulation, whereas Vanderbeck and Wolsey [39] propose valid inequalities involving only the natural variables. Gicquel *et al.* [40] discuss exact solution approaches for DLSP with identical parallel machines. Gicquel *et al.* [41] introduce a disaggregate formulation in which the machines are indexed explicitly. In their terminology, the Lasdon-Terjung model is an aggregate formulation. They prove that the two formulations are equivalent, and show how a family of inequalities developed by van Eijl and van Hoesel [42] can be adapted to give valid inequalities for the aggregate formulation. Jans and Degraeve [43] present an industrial extension of DLSP for a tire manufacturer. Quadt and Kuhn [44] make use of aggregate variables to solve a variant of the capacitated lot-sizing problem with linked lot sizes. Kaczmarczyk [45] applies the same idea to PLSP with identical parallel machines. Fiorotto and de Araujo [46] develop a Lagrangian heuristic for unrelated parallel machines based on the strategy of Eppen and Martin [38].

The problem we consider is essentially different from those studied in the articles cited above because of its cumulative demand structure, and to the best of our knowl-

edge, it has not been addressed in the literature before. On the one hand, it can be classified within DLSP since all-or-nothing assumption is valid. On the other hand, it can be seen as a GLSP model with only one macroperiod made up of a number of fixed-length microperiods.

Outline of the chapter is as follows: in §2.2 we prove that the problem is NP-hard, present two mixed-integer programming formulations, and show their equivalence. Then we give some properties of the optimal objective value in §2.3.2, and prove several optimality conditions in §2.3.3. Next, in §2.3.4 we suggest a heuristic algorithm. Afterwards, we discuss and formulate two possible extensions to the problem in §2.4. Finally, we present a detailed computational study in §2.5.

2.2. Problem Definition

Let I, M, T denote the number of secondary resource types, machines, and time periods, respectively, and i, j, t the indices thereof. All machines are identical. Let $\overline{x_i}$ be the demand as average number of secondary resources of type i to be used in a period. In other words, if d_i is the total demand for items of type i, and c_i the number of items of type i a machine can produce in one period, then $\overline{x_i} = \frac{1}{T} \times \frac{d_i}{c_i}$. One and only one type of secondary resource can be installed in a machine during a period. The problem is to find an assignment of secondary resources to machines throughout the planning horizon such that all demands are satisfied and the total number of setups and teardowns is minimized. By convention, setups in the first period and teardowns in the last period are not counted. We shall see in §2.2.3 that one can assume $T\,\overline{x_i} \in \mathbb{Z}_{\geq 0}$ for all i without loss of generality, and under this assumption the problem is feasible if and only if $\sum_i \overline{x_i} \leq M$.

2.2.1. Complexity

We prove that the lot-sizing and scheduling problem defined above is NP-hard by reduction from the well-known Partition problem.

Theorem 2.1. *The problem is NP-hard.*

Proof. Let $a_1, \ldots, a_m$ and $b = \frac{1}{2}\sum_{k=1}^{m} a_k$ be positive integers. PARTITION is the following problem: Can we divide the collection $a_1, \ldots, a_m$ into two subsets such that the numbers in each subset sum up to b? Now consider an instance of our problem where $M = 2$, $I = m$, $T = b$, and $\overline{x_i} = \frac{a_i}{T}$. We will show that PARTITION has a positive answer if and only if the optimal objective value z^* is less than or equal to $2I-4$ for this instance. Assume there exist S_1, S_2 such that $S_1 \cup S_2 = \{1, \ldots, m\}$, $S_1 \cap S_2 = \varnothing$ and $b = \sum_{k\in S_1} a_k = \sum_{k\in S_2} a_k$. Assign the secondary resources in S_j to machine j $(j = 1, 2)$. Then total number of setups and teardowns is $(2|S_1|-2)+(2|S_2|-2) = 2I-4$, whence $z^* \leq 2I - 4$. Conversely, suppose $z^* \leq 2I - 4$, and consider an optimal schedule. Let w_j be the number of different secondary resource types that appear in machine j. At least $2w_j - 2$ setups and teardowns must take place in j. Moreover, $w_1 + w_2 \geq I$. So $2I - 4 \geq z^* \geq (2w_1 - 2) + (2w_2 - 2) \geq 2I - 4$. Therefore $z^* = 2I - 4$ and $w_1 + w_2 = I$, implying that PARTITION has an affirmative answer. □

2.2.2. Disaggregate and Aggregate Formulations

It is natural to keep an explicit account of secondary resource–machine assignments when modeling the problem at hand. This is the essential feature of the disaggregate formulation F1 to be given below. Let y_{ijt} be a binary variable defined as 1 if in period t machine j is set up for a secondary resource of type i, and as 0 otherwise. Define another binary variable z^+_{ijt} as 1 if a setup for a secondary resource of type i is performed on machine j in transition from period $t-1$ to t, and as 0 otherwise; similarly define z^-_{ijt} for teardowns $(t > 1)$.

$$\text{F1:} \quad \min \quad \sum_{i,j,t} (z^+_{ijt} + z^-_{ijt}) \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{i} y_{ijt} \leq 1 \qquad \text{for all } j,t \tag{2.1b}$$

$$\frac{1}{T} \sum_{j,t} y_{ijt} \geq \overline{x_i} \qquad \text{for all } i \tag{2.1c}$$

$$y_{ijt} - y_{ij,t-1} = z^+_{ijt} - z^-_{ijt} \qquad \text{for all } i,j,t \tag{2.1d}$$

$$z^+_{ijt}, z^-_{ijt} \geq 0 \qquad \text{for all } i,j,t \tag{2.1e}$$

$$y_{ijt} \in \{0,1\} \qquad \text{for all } i,j,t \tag{2.1f}$$

The first constraint (2.1b), together with (2.1f), states that a machine is either idle or works at full capacity in a period (being set up for one and only one secondary resource). This is the so-called all-or-nothing assumption. The next set of inequalities (2.1c) guarantees that enough items are produced of each type. Constraint (2.1d) holds an account of setups and teardowns. Expressions involving z^+_{ijt}, z^-_{ijt} are subject to $t > 1$. Note that z^+_{ijt} and z^-_{ijt} can be relaxed as continuous variables in view of the objective since the y_{ijt} are binary.

The problem admits a simpler formulation – to be called F2 – since the machines in question are identical. The idea, which goes back to Lasdon and Terjung [37], is to keep track only of the total number of secondary resources used for each type in a period. This determines the secondary resource–machine assignments implicitly. Let x_{it} be the total number of secondary resources of type i used in period t. Let s^+_{it} be the number of setups required for type i in transition from period $t-1$ to t; similarly define s^-_{it} for teardowns ($t > 1$). A complete list of indices, parameters, and decision variables for F1 and F2 can be found in Table 2.1.

Table 2.1. Indices, parameters, and decision variables for formulations F1 and F2.

Symbol(s)	Explanation
i, j, t	indices for secondary resource types, machines, and time periods
M, T	number of machines and time periods
$\overline{x_i}$	demand as average number of secondary resources of type i to be used in a period
x_{it}	number of secondary resources of type i used in period t
y_{ijt}	1 if machine j is set up for a secondary resource of type i in period t, and 0 otherwise
s_{it}^+ /s_{it}^-	number of setups/teardowns required for secondary resources of type i in transition from period $t-1$ to t
z_{ijt}^+/z_{ijt}^-	1 if a setup/teardown for a secondary resource of type i is performed on machine j in transition from period $t-1$ to t, and 0 otherwise

The aggregate formulation F2 is given below. Expressions involving s_{it}^+ and s_{it}^- are subject to $t > 1$. The set of nonnegative integers is designated by $\mathbb{Z}_{\geq 0}$.

$$\text{F2:} \quad \min \quad \sum_{i,t} (s_{it}^+ + s_{it}^-) \tag{2.2a}$$

$$\text{s.t.} \quad \sum_{i} x_{it} \leq M \qquad \text{for all } t \tag{2.2b}$$

$$\frac{1}{T} \sum_{t} x_{it} \geq \overline{x_i} \qquad \text{for all } i \tag{2.2c}$$

$$x_{it} - x_{i,t-1} = s_{it}^+ - s_{it}^- \qquad \text{for all } i, t \tag{2.2d}$$

$$s_{it}^+, s_{it}^- \geq 0 \qquad \text{for all } i, t \tag{2.2e}$$

$$x_{it} \in \mathbb{Z}_{\geq 0} \qquad \text{for all } i, t \tag{2.2f}$$

The first constraint (2.2b) ensures that the total number of secondary resources used in a period cannot exceed the number of machines. The second one (2.2c) is the demand fulfillment restriction. It is appropriate to call (2.2d) as 'setup balance equations'. Note that the continuous variables s_{it}^+, s_{it}^- assume integer values in an optimal solution

by virtue of the objective since the x_{it} are integer. Note also that M is a natural upper bound for all decision variables.

2.2.3. Equivalence of the Two Formulations

Lemma 2.2. *The inequalities* $\sum_{j,t} y_{ijt} \geq \lceil T\,\overline{x_i} \rceil$ *and* $\sum_t x_{it} \geq \lceil T\,\overline{x_i} \rceil$ *are valid for* F1 *and* F2*, respectively, for all* i*.*

Proof. For any feasible solution of F1, the y_{ijt} will be integers, so the sums $\sum_{j,t} y_{ijt}$ will be integers greater than or equal to $T\,\overline{x_i}$. Consequently, the inequality $\sum_{j,t} y_{ijt} \geq \lceil T\,\overline{x_i} \rceil$ is satisfied for all i. The argument is similar for F2. □

From this point on, we assume without loss of generality that $T\,\overline{x_i} \in \mathbb{Z}_{\geq 0}$ for all i.

Lemma 2.3. *The following statements are equivalent:*
(i) F1 *is feasible.*
(ii) F2 *is feasible.*
(iii) $\sum_i \overline{x_i} \leq M$*.*

Proof. If $y = (y_{ijt})$ is a feasible solution of F1, then $x = (x_{it})$ defined as $x_{it} = \sum_j y_{ijt}$ is a feasible solution of F2 (note that y_{ijt} and x_{it} uniquely determine z_{ijt} and s_{it}). Indeed, according to this definition, constraints (2.1b) and (2.1c) imply (2.2b) and (2.2c), respectively. If F2 is feasible, then $\overline{x_i} \leq \frac{1}{T}\sum_t x_{it}$ and $\sum_i x_{it} \leq M$, implying $\sum_i \overline{x_i} \leq \frac{1}{T}\sum_i \sum_t x_{it} \leq \frac{1}{T}\sum_t M = M$. Finally, if $\sum_i \overline{x_i} \leq M$, then $\sum_i T\,\overline{x_i} \leq TM$. Hence, the TM slots in the planning horizon are sufficient to cover all demand (the assumption $T\,\overline{x_i} \in \mathbb{Z}$ is crucial here), implying that F1 is feasible. □

Let S_1 and S_2 be the feasible solution spaces of F1 and F2, respectively. For each $y \in S_1$, let $\alpha(y) = x$ be defined by $x_{it} = \sum_j y_{ijt}$. Then $\alpha(y) \in S_2$ as we have seen in the proof of Lemma 2.3. So α is a many-to-one mapping from S_1 to S_2. It is not reasonable to perform a setup for a secondary resource which has just been torn down from another machine in the immediately preceding period. We designate by S_1' the schedules reasonable in this sense. Thus, S_1' consists of those $y \in S_1$ for which there

exists no pair of machines $j_1 \neq j_2$ such that $z^+_{ij_1t} = 1$ and $z^-_{ij_2t} = 1$. Let us denote by $z_1(y)$ and $z_2(x)$ the objective function values of F1 and F2 evaluated at $y \in S_1$ and $x \in S_2$.

Lemma 2.4. *(i)* $z_1(y) \geq z_2(\alpha(y))$ *for all* $y \in S_1$.
(ii) $z_1(y) = z_2(\alpha(y))$ *for all* $y \in S'_1$.
(iii) For all $x \in S_2$, *there exists* $y \in S'_1$ *such that* $\alpha(y) = x$.

Proof. Let $y \in S_1$ and $x := \alpha(y)$. Summing up (2.1d) over j, we get $\sum_j y_{ijt} - \sum_j y_{ij,t-1} = \sum_j z^+_{ijt} - \sum_j z^-_{ijt}$ so that $s^+_{it} - s^-_{it} = \sum_j z^+_{ijt} - \sum_j z^-_{ijt}$ by (2.2d). In view of (2.1a) and (2.2a), the inequality $s^+_{it} + s^-_{it} \leq \sum_j z^+_{ijt} + \sum_j z^-_{ijt}$ always holds, two sides being equal if and only if $y \in S'_1$. This proves (i) and (ii). Now let $x \in S_2$. We define $y \in S_1$ as follows: For the first period, make an arbitrary assignment of secondary resources x_{i1} to machines. Then, for each subsequent period, first perform teardowns for all i by choosing arbitrarily s^-_{it}-many machines in which i has been installed in the previous period, and then perform setups for all i by choosing s^+_{it}-many empty machines. Thus $y \in S'_1$ and $\alpha(y) = x$. □

assign the x_{i1} arbitrarily to machines
for $t = 2$ **to** T **do**
  for all i **do**
    choose arbitrarily s^-_{it}-many machines in which secondary resource i has been installed in period $t-1$, and perform a teardown
  for all i **do**
    choose arbitrarily s^+_{it}-many empty machines, perform a setup of secondary resource i, and update the empty machine list

Figure 2.1. Disaggregation algorithm described in the proof of Lemma 2.4.

Theorem 2.5. *Formulations* F1 *and* F2 *are equivalent in the sense that feasibility of one implies that of the other, in which case the optimal objective values are the same.*

Proof. We have already shown the assertion about feasibility in Lemma 2.3. Now let z_1^* and z_2^* be the optimal objective values of F1 and F2. The first part of Lemma 2.4 implies $z_1^* \geq z_2^*$, whereas the other two parts imply $z_1^* \leq z_2^*$. Therefore, $z_1^* = z_2^*$ provided that the problem is feasible. □

There is an equivalence between not only the mixed-integer programs but also the linear programming relaxations of the two formulations. We prove this in the next lemma.

Lemma 2.6. *Linear programming bounds of* F1 *and* F2 *are the same.*

Proof. For any feasible instance of F2, if the integrality constraints are relaxed, setting $x_{it} := \overline{x_i}$ for all i, t yields a solution with zero optimal value. Similarly, setting $\lfloor \overline{x_i} \rfloor$-many y_{ijt} to 1 and one y_{ijt} to $\overline{x_i} - \lfloor \overline{x_i} \rfloor$ for all i, t, avoiding possible conflicts on machines, gives a solution with zero optimal value for the relaxation of F1. □

In case some machines become temporarily unavailable within the planning horizon, F2 is no longer a valid representation of the problem even if M is replaced by M_t in the formulation. The simplest example demonstrating this fact consists of one secondary resource with $\overline{x_i} = 1$, two periods, and two machines, where only machine j is available in period j. Then F2 yields 0 as optimal value whereas it is actually 2. Let us note that an aggregate formulation is not possible when the machines are nonidentical.

2.3. Algorithmic and Modeling Improvements

2.3.1. Symmetry-Breaking Constraints

The disaggregate formulation F1 has about M times as many variables and constraints as the aggregate formulation F2, with many alternative symmetric solutions burdening the model [47]. For the lot-sizing problem on identical parallel machines, Jans [48] proposes eight families of symmetry-breaking constraints. The last three of these involve setup times or costs, and are not applicable in our case, whereas the first

five can be readily adapted to F1. In our notation, the second family, denoted (SBC2) by Jans [48], reads

$$\sum_{i=1}^{I} 2^{I-i} y_{ijt} \geq \sum_{i=1}^{I} 2^{I-i} y_{i,j+1,t} \quad \text{for all } 1 \leq j \leq M-1 \text{ and } t. \tag{2.3}$$

For example, when $I = 3$, this can be written explicitly as $4y_{11t} + 2y_{21t} + y_{31t} \leq 4y_{12t} + 2y_{22t} + y_{32t} \leq 4y_{13t} + 2y_{23t} + y_{33t} \leq \cdots$ for all t. Thus, if item 1 is produced in period t, it must be on the first machine(s), because $y_{1jt} = 1$ implies $y_{1j't} = 1$ for all $j' < j$; similarly for items 2 and 3. In general, inequalities (2.3) impose a unique lexicographic ordering on machines in each period. This is also true of (SBC1) that contains (2.3) as a subset. The families (SBC3–5) impose a partial ordering only. The computational study in Jans [48] shows that (SBC2) turns out to be one of the most efficient families, so we use (2.3) as a means of improving F1.

2.3.2. Properties of Optimal Objective Value

We denote the optimal objective value by z^*.

Lemma 2.7. *z^* is zero if and only if $\sum_i \lceil \overline{x_i} \rceil \leq M$.*

Proof. If $\sum_i \lceil \overline{x_i} \rceil \leq M$, setting $x_{it} := \lceil \overline{x_i} \rceil$ yields a feasible solution with zero objective value. Conversely, if $z^* = 0$, the assignment in any period determines the schedule for the entire planning horizon for each machine; more precisely, $y_{ijt} = 1$ implies $y_{ijt'} = 1$ for all t'. It follows that $M \geq \sum_i \lceil \overline{x_i} \rceil$ since in any feasible solution there must exist a period in which at least $\lceil \overline{x_i} \rceil$-many distinct machines are dedicated for secondary resource i. □

Lemma 2.8. *There exists an optimal schedule such that no machine is kept idle in any period.*

Proof. Any idle periods on a machine can be filled up by prolonging the production in neighboring periods, which does not worsen the objective value because excess production is not penalized. □

Let us denote by S the collection of feasible solutions $y \in S_1$ such that $\sum_i y_{ijt} = 1$ for all j, t. We shall refer to such solutions as full schedules. According to Lemma 2.8, there always exists a full optimal schedule.

Corollary 2.9. *z^* must be an even integer.*

Proof. The objective value associated with any full schedule must be an even integer since every teardown is necessarily paired with a setup there. The result follows from Lemma 2.8. □

Lemma 2.10. *$z^* \leq 2I - 2$.*

Proof. Write $T\,\overline{x_i} = T\,\lfloor \overline{x_i} \rfloor + r_i$, where r_i is an integer such that $0 \leq r_i \leq T-1$. Allocate, for each i, $T\,\lfloor \overline{x_i} \rfloor$-many secondary resources starting with the first machine. These allocations do not contribute to the objective. Next, fill up the remaining machines with the r_i-many secondary resources one by one, in ascending order with respect to the machine and time indices. At this stage, each secondary resource type increases the objective value by at most two except the first and the last types for which the contribution is at most one. The result follows. □

2.3.3. Optimality Conditions

Due to Theorem 2.1 the problem is inherently difficult, and one cannot expect to solve it to optimality for large instances in reasonable time. Still, its mixed-integer formulation can be improved by means of linear constraints that do not change the optimal objective value when added to the formulation. In this subsection, we prove several optimality conditions by means of which such constraints can be obtained.

Given a solution, let $b(j, t, l)$ represent the block of assignments on machine j in the l consecutive periods $t, t+1, \ldots, t+l-1$. For $y \in S$, consider the operation of exchanging two blocks $b(j_1, t_1, l_1)$ and $b(j_2, t_2, l_2)$ to obtain a new solution y'. We assume that the blocks are of the same length ($l_1 = l_2$) unless they are on the same machine ($j_1 = j_2$) and adjacent ($t_1 + l_1 = t_2$ or $t_2 + l_2 = t_1$). So y' is well-defined. It will be convenient to define z_{ijt} to be z^+_{ijt} or $-z^-_{ijt}$ according as $z^+_{ijt} \geq 0$ or $z^-_{ijt} > 0$. Let

$c_{jt}(y)$ be 1 if $z_{ijt} \neq 0$ for some i, and 0 otherwise. In other words, $c_{jt}(y)$ is 1 if there is a changeover on machine j from period $t-1$ to t. Now we can express in a simple way the difference $d := z_1(y') - z_1(y)$ in the objective function value (2.1a): the equality $z_1(y) = 2\sum_{j,t} c_{jt}(y)$ holds for all $y \in S$; consequently, $d = 2\sum_{j,t}(c_{jt}(y') - c_{jt}(y)) = 2\sum_{j,t} d_{jt}$ where $d_{jt} := c_{jt}(y') - c_{jt}(y)$. For convenience, we define $c_{j1}(y)$ and $c_{j,T+1}(y)$ as 0, too.

Lemma 2.11. *(i) If $j_1 \neq j_2$ or the blocks are nonadjacent, then $\frac{1}{2}d = d_{j_1 t_1} + d_{j_1, t_1+l} + d_{j_2 t_2} + d_{j_2, t_2+l}$ where $l := l_1 = l_2$.*
(ii) If $j_1 = j_2 =: j$ and $t_1 + l_1 = t_2$, then $\frac{1}{2}d = d_{jt_1} + d_{j,t_2+l_2} + c_{j,t_1+l_2}(y') - c_{j,t_1+l_1}(y)$. An analogous equation holds for the case $t_2 + l_2 = t_1$.

Proof. When two blocks are exchanged, all d_{jt} are zero except those related to the borders. Writing them out explicitly, the equalities follow. □

Theorem 2.12. *There exists an optimal schedule such that*

$$x_{it} \geq \lfloor \overline{x_i} \rfloor \quad \text{for all } i, t. \tag{2.4}$$

Proof. Fix a secondary resource type i arbitrarily. It is enough to find an optimal $x \in S_2$ satisfying $x_{it} \geq \lfloor \overline{x_i} \rfloor$ for all t. Indeed, rewriting the formulation in terms of $x_{it} - \lfloor \overline{x_i} \rfloor$ instead of x_{it}, we obtain a problem of the same form with strictly smaller M, and the result follows by repeating the same argument.

Let $x \in S_2$ be an optimal solution, and suppose $x_{it_0} < \lfloor \overline{x_i} \rfloor$ for some t_0. Take a schedule $y \in S_1'$ such that $\alpha(y) = x$ (Lemma 2.4). Since $z^* \neq 0$, y must be full (Lemma 2.8). Let $x_i^{\max} := \max_t\{x_{it}\}$, $x_i^{\min} := \min_t\{x_{it}\}$, and let n_i be the number of periods t for which $x_{it} = x_i^{\max}$ or $x_{it} = x_i^{\min}$. Consider the ordered pair $w_i(x) := (x_i^{\max} - x_i^{\min}, n_i)$. It is sufficient to show the following: as long as the inequality $x_i^{\max} - x_i^{\min} \geq 2$ is satisfied, one can obtain by exchanging suitable blocks an alternative optimal $y' \in S_1$ such that $w_i(x')$ is lexicographically smaller than $w_i(x)$, where $x' := \alpha(y')$.

Consider a list $t_1, t_1 + 1, \ldots, t_1 + l_1 - 1 =: t_1'$ of consecutive periods t for which $x_{it} = x_i^{\max}$, maximal in the sense that $x_{i,t_1-1} < x_i^{\max}$ (unless $t_1 = 1$) and $x_{i,t_1'+1} < x_i^{\max}$ (unless $t_1' = T$). Similarly, consider a list $t_2, t_2 + 1, \ldots, t_2 + l_2 - 1 =: t_2'$ of periods

t for which $x_{it} = x_i^{\min}$, maximal in the sense that $x_{i,t_2-1} > x_i^{\min}$ (unless $t_2 = 1$) and $x_{i,t_2'+1} > x_i^{\min}$ (unless $t_2' = T$). We may assume without loss of generality (see Figure 2.1) that there exists a machine j_1 such that $y_{ij_1t} = 1$ for all $t_1 \le t \le t_1'$, $y_{ij_1,t_1-1} = 0$, and $y_{ij_1,t_1'+1} = 0$. Also, there exists a machine j_2 such that $y_{ij_2t} = 0$ for all $t_2 \le t \le t_2'$, and $y_{ij_2,t_2-1} = 1$. In case $t_2 = 1$, we may assume $y_{ij_2,t_2'+1} = 1$ as well.

First, suppose that $j_1 \ne j_2$ (Figure 2.2) or that the lists are nonadjacent. Let $l := \min\{l_1, l_2\}$. If $l_1 \le l_2$, exchange the blocks $b(j_1, t_1, l)$ and $b(j_2, t_2, l)$ to obtain y'. The change $d := z_1(y') - z_1(y)$ in the objective function value is given by $\frac{1}{2}d = d_{j_1t_1} + d_{j_1,t_1+l} + d_{j_2t_2} + d_{j_2,t_2+l}$ (Lemma 2.11). Since $d_{j_1t_1} \le 0, d_{j_1,t_1+l} \le 0$, and $d_{j_2t_2} = -1$, we have $d \le 0$. So y' is still optimal, and $w_i(x') < w_i(x)$. In case $t_2 = 1$, we shall take $b(j_2, t_2' - l + 1, l)$ as the second block. If $l_1 > l_2$, exchange $b(j_1, t_1, l)$ and $b(j_2, t_2, l)$ as before; however, in this case, $\frac{1}{2}d$ may turn out to be 1 if $y_{ij_2,t_2'+1} = 0$. Nevertheless, if $y_{ij_2,t_2'+1} = 0$ there must exist a distinct machine j_3 such that $y_{ij_3t_2'} = 0$ and $y_{ij_3,t_2'+1} = 1$. Doing a second exchange of blocks $b(j_2, t_2'+1, T-t_2')$ and $b(j_3, t_2'+1, T-t_2')$ guarantees that $d \le 0$.

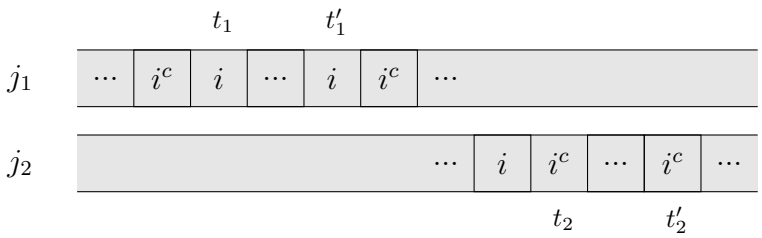

Figure 2.2. Machines, secondary resources, and time periods in case $j_1 \ne j_2$ in the proof of Theorem 2.12, where i^c represents any secondary resources other than i.

Next, suppose that $j_1 = j_2 =: j$ and the lists are adjacent. We exchange the two blocks $b(j, t_1, l_1)$ and $b(j, t_2, l_2)$. If $t_2' + 1 = t_1$, then $\frac{1}{2}d = d_{jt_2} + d_{j,t_1+l_1} + c_{j,t_2+l_1}(y') - c_{j,t_2+l_2}(y)$ (Lemma 2.11). Since $d_{jt_2} = -1, d_{j,t_1+l_1} \le 0$, and $c_{j,t_2+l_1}(y') = c_{j,t_2+l_2}(y) = 1$, we have $d < 0$. This shows that $t_2' + 1 = t_1$ cannot be true, so $t_1' + 1 = t_2$. Then $\frac{1}{2}d$ may be 1, but after a second exchange as in the preceding paragraph, we obtain a solution for which $d \le 0$. This completes the proof. □

Corollary 2.13. *Every instance* $(\overline{x_i}, M, T)$ *of the problem is equivalent to* $(\overline{x_i} - \lfloor \overline{x_i} \rfloor, M - \sum_i \lfloor \overline{x_i} \rfloor, T)$.

Proof. By Theorem 2.12, we can add the inequalities $x_{it} \geq \lfloor \overline{x_i} \rfloor$ to the formulation and still get the same optimal value. Define new variables $x'_{it} := x_{it} - \lfloor \overline{x_i} \rfloor$ for all i, t. Rewriting constraints (2.2b) and (2.2c) yields the result. □

Theorem 2.14. *(i) There exists an optimal schedule such that $s^+_{it} \leq 1$ for all i, t.*
(ii) There exists an optimal schedule such that $s^-_{it} \leq 1$ for all i, t.
(iii) There exists an optimal schedule such that $s^+_{i2} \leq 1, s^-_{i2} \leq 1, s^+_{iT} \leq 1$, and $s^-_{iT} \leq 1$ for all i.

Proof. (i) Fix a secondary resource type i arbitrarily. Let $x \in S_2$ be an optimal solution, and suppose $s^+_{it} > 1$ for some t. Take a schedule $y \in S'_1$ such that $\alpha(y) = x$ (Lemma 2.4). As $z^* \neq 0$, y must be full (Lemma 2.8). Since $s^+_{it} > 1$ and $y \in S'_1$, there are at least two machines j_1, j_2 such that $z^+_{ij_1t} = z^+_{ij_2t} = 1$. Let i' be the secondary resource on machine j_2 in period $t-1$, let $t_1 := \max\{\tau \geq t : y_{ij_1\tau} = 1\}$, $t_2 := \min\{\tau \leq t-1 : y_{i'j_2\tau} = 1\}$, and $l := \min\{t_1 - t + 1, t - t_2\}$. Exchange the two blocks $b(j_1, t, l)$ and $b(j_2, t_2, l)$ to obtain a new schedule y' (Figure 2.3). Then $\frac{1}{2}d = d_{j_1t} + d_{j_1,t_1+1} + d_{j_2t_2} + d_{j_2t}$ (Lemma 2.11). Here $d_{jt} \leq 0, d_{j_2t} = -1$, and at most one of $d_{j_1,t_1+1}, d_{j_2t_2}$ is 1 so that $d \leq 0$. Hence, y' is still optimal.

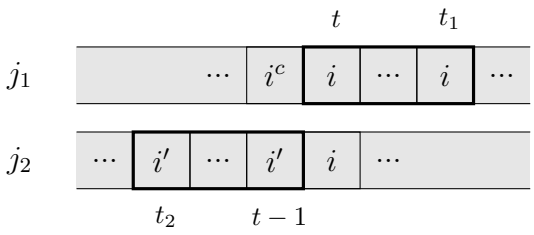

Figure 2.3. Two blocks $b(j_1, t, l)$ and $b(j_2, t_2, l)$ in y to be exchanged in the proof of Theorem 2.14, where i^c represents any secondary resources other than i.

Consider the ordered T-tuple $w_i(x) := (x_{i1}, \ldots, x_{iT})$. Clearly, $w_i(x') > w_i(x)$ where $x' := \alpha(y')$. Hence, repeating the same argument we can obtain in a finite number of steps an optimal solution for which $s^+_{it} \leq 1$ for all t. Define s_{it} to be s^+_{it} or $-s^-_{it}$ according as $s^+_{it} \geq 0$ or $s^-_{it} > 0$. Notice that the exchange above increases $s_{i'\tau}$ only for $\tau = t$, and $s_{i't}$ is increased by 2. However, as $s_{i't}$ is initially negative, it cannot exceed 1 after the exchange. Therefore, all secondary resources can be handled one by one until an optimal solution satisfying the assertion is found.

(ii) Analogous to (i).

(iii) The argument is essentially the same as in (i), and makes use of the fact that some terms drop in the expression in Lemma 2.11 when the endpoints happen to be the first or the last period. □

2.3.4. Heuristic Algorithm

In this subsection, we introduce a heuristic with polynomial time complexity $O((I^3+M)T)$, producing high-quality feasible solutions (see §2.5 for details of computational results). The algorithm starts with reducing the problem: for each secondary resource type i, as many as $\lfloor \overline{x_i} \rfloor$ resources are assigned to machines right away. Thus it remains to consider the fractional demands $\overline{x_i} - \lfloor \overline{x_i} \rfloor$ only, with a smaller number of machines. Notice that this reduced problem is equivalent to the original one by Corollary 2.13. Next, the problem is compressed; that is to say, a machine is allotted for the type with largest (fractional) demand as long as there is enough capacity. Although this step is quite reasonable, we will shortly present an example where it leads to suboptimal solutions. What is left after compression is a fairly tight scheduling problem. At this point, there will surely be at least one setup or teardown for each of the remaining secondary resource types. Therefore, it makes sense to find pairs (if any) whose demands complement each other, and further reduce the problem by assigning every such pair to one of the available machines. Next, we repeat this once again, taking this time also the slack capacity (if any) into account, and then perform a similar search for possible triplets. Finally, in descending order with respect to demand, we do a straightforward allocation. Figure 2.4 gives a pseudocode that outlines the heuristic described. Notice that finding triplets and reduction take $O(I^3T)$ and $O(MT)$ steps, respectively, and these two dominate the remaining parts of the algorithm. Hence, the overall complexity is $O((I^3+M)T)$.

Let us demonstrate the algorithm for a specific instance. Say $T = 10$, $I = 10$, the $\overline{x_i}$ are 1.9, 0.2, 5.8, 1.2, 3.0, 7.7, 3.2, 1.4, 0.4, 2.0, and $M = 27$. After reduction, we are

{reduction}

$j_{\text{current}} \leftarrow 1$

for all i **do**

  for $j = j_{\text{current}}$ **to** $j_{\text{current}} + \lfloor \overline{x_i} \rfloor$ **do**

    $y_{ijt} \leftarrow 1$ for all t

  $\overline{x_i} \leftarrow \overline{x_i} - \lfloor \overline{x_i} \rfloor$

  $j_{\text{current}} \leftarrow j_{\text{current}} + \lfloor \overline{x_i} \rfloor$

{compression}

$slack \leftarrow M - j_{\text{current}} - \sum \overline{x_i}$

while $j_{\text{current}} < M$ **do**

  if $slack \geq 1 - \max\{\overline{x_i}\}$ **then**

    $i \leftarrow \arg\max\{\overline{x_i}\}$

    $y_{ijt} \leftarrow 1$ for all t

    $\overline{x_i} \leftarrow 0$

    $j_{\text{current}} \leftarrow j_{\text{current}} + 1$

    $slack \leftarrow M - j_{\text{current}} - \sum \overline{x_i}$

{finding pairs}

for $i = 1$ **to** $I - 1$ **do**

  for $i' = i$ **to** I **do**

    if $\overline{x_i} + \overline{x_{i'}} = 1$ **then**

      $y_{ijt} \leftarrow 1$ for all $t \leq T\,\overline{x_i}$

      $y_{i'jt} \leftarrow 1$ for all $t > T\,\overline{x_i}$

      $\overline{x_i} \leftarrow 0$

      $\overline{x_{i'}} \leftarrow 0$

      $j_{\text{current}} \leftarrow j_{\text{current}} + 1$

{finding pairs utilizing slack}

repeat the previous step (lines 18-25) with the conditional in line 20 relaxed to $\overline{x_i} + \overline{x_{i'}} \geq 1 - slack$, updating $slack$ after line 25

Figure 2.4. Heuristic algorithm.

{finding triplets}

perform a search for triplets analogous to the one for pairs (lines 18-25)

{straightforward allocation}

$t_{\text{current}} = 1$

for all i **do**

  if $\overline{x_i} > 0$ **then**

    $y_{ijt} \leftarrow 1$ for all $T\,\overline{x_i}$-many periods

    update t_{current} and (if need be) j_{current}

if *slack* > 0 **then**

  prolong the production of the last secondary resource on the last machine

Figure 2.4. Heuristic algorithm (cont.).

left with eight secondary resource types with demands 0.9, 0.2, 0.8, 0.2, 0.7, 0.2, 0.4, 0.4, and there are only four machines. Note that the slack capacity is $4 - (0.9 + 0.2 + \ldots + 0.4) = 0.2$. Compression allots one machine to 0.9 in full. Thus, the numbers of types and machines are decreased by one, and the new slack is 0.1. Demands 0.2 and 0.8 complement each other. So do 0.2 and 0.7 if slack capacity is utilized. Finally, we detect a triplet 0.2, 0.4, 0.4. Then nothing is left, and the objective function value associated with the resulting schedule is $z = 8$.

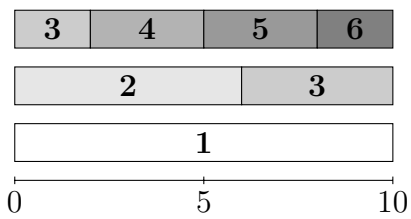

Figure 2.5. Gantt chart associated with the heuristic solution of the instance for which the compression step in the heuristic algorithm leads to suboptimality.

When the heuristic is applied to the instance where $T = 10$, $I = 6$, the $\overline{x_i}$ are 0.7, 0.6, 0.6, 0.3, 0.3, 0.2 (so that the $T\,\overline{x_i}$ are 7, 6, 6, 3, 3, 2), and $M = 3$, the compression step allots one machine to the first secondary resource type for all ten

periods, and it follows that $z = 8$ (see the Gantt chart in Figure 2.5 – the numbers on the blocks represent secondary resource types in the order given above). However, the optimal objective value is 6, which can be obtained by assigning the three appropriate pairs to three machines. This shows that the compression step may lead to suboptimal solutions.

2.4. Model Extensions

In this section we extend our models to take into account two sets of constraints that might be important in real life, namely vertical constraints and minimum lot size constraints.

2.4.1. Vertical Constraints

By vertical constraints, we mean, as in Lasdon and Terjung [37], the following inequalities limiting by a parameter $\overline{s}$ the number of setups and teardowns performed in between two consecutive periods:

$$\sum_i (s_{it}^+ + s_{it}^-) \leq \overline{s} \quad \text{for all } t > 1. \tag{2.5}$$

Clearly, the problem is still NP-hard in general: for $\overline{s}$ large enough, it is equivalent to F2. Although Lemma 2.8 no longer holds, z^* must be an even integer in case of feasibility. Lemma 2.3 does not hold either: consider for $\overline{s} \leq 3$ the instance where $T = 2$, $I = 4$, the $\overline{x_i}$ are all 0.5, and $M = 2$; we have $\sum_i \overline{x_i} \leq M$, but there exists no feasible solution. Also, the argument used to prove Lemma 2.10 is invalid for F2 with (2.5); nevertheless, it is not easy to find a counterexample because the upper bound given there is rather loose. The proofs of Theorems 2.12 and 2.14 are invalid too, but computational tests suggest that the assertions may still be true.

For the problem with vertical constraints, we modify the heuristic described in §2.3.4 slightly as follows: every step of the algorithm in Figure 2.4 is to be ap-

plied as is, except for finding pairs, finding pairs utilising slack, and finding triplets. In these three steps, we first check whether inclusion of the pair/triplet into the schedule would violate the vertical constraints. If inclusion is not possible without increasing the number of changeovers, then we simply continue searching for other pairs/triplets. In the modified heuristic, the complexity of finding pairs and triplets remain the same since there are two and six possible sequences for a pair and triplet, respectively. However, the complexity of finding pairs utilising slack is increased from $\mathrm{O}(I^2T)$ to $\mathrm{O}(I^2T^2)$ so that the overall complexity becomes $\mathrm{O}((I^3 + I^2T + M)T)$.

2.4.2. Minimum Lot Size Constraints

Minimum lot size constraints translate in our problem as a lower bound on the number of periods a secondary resource should remain within a machine after being installed. Such a restriction might stem from shop floor requirements (e.g., materials to be processed may only be stored and retrieved in certain large batches), or a managerial decision based on the observation that scrap rates gradually decrease as one produces more and more after a new setup.

Minimum lot size restriction can be incorporated into F2 as follows. The idea is to couple each teardown with a setup, and make sure that every teardown is associated with a setup which is performed at least m periods earlier, where m stands for the lower bound mentioned. This is guaranteed by the inequalities

$$\sum_{t=1+m}^{u+m} s_{it}^{-} \leq \sum_{t=1}^{u} s_{it}^{+} \quad \text{for all } i \text{ and } 1 \leq u \leq T-(m-1) \tag{2.6}$$

together with $s_{i2}^{-} = \ldots = s_{im}^{-} = 0 = s_{i,T-(m-2)}^{+} = \ldots = s_{iT}^{+}$, where $s_{i1}^{+} := x_{i1}$ and $s_{i,T+1}^{-} := x_{iT}$ for all i. To see why, fix a secondary resource type i. Let us number consecutively each setup (including the ones before the first period) and teardown (including the ones after the last period) throughout the planning horizon. Thus, the number associated with a setup made in period t will be strictly smaller than one made in period t' if $t < t'$ (the numbering within a period is immaterial); similarly for

teardowns. Given a feasible solution of F2, instead of making the secondary resource assignments in the algorithm in Figure 2.1 arbitrarily, if one adopts the policy of performing the nth teardown (not on an arbitrary machine but) on the machine with the nth setup, then the best schedule with respect to minimum lot size is obtained. That is to say, for the given solution, this policy leads for i to a schedule with the largest possible minimum lot size – call it m_i'. We can express m_i' in terms of our setup and teardown variables: for $2 \leq k \leq T+1$, let

$$t_{ik} := \min\{l \geq 1 : s^-_{i2} + \ldots + s^-_{ik} \leq s^+_{i1} + \ldots + s^+_{il}\}$$

and $m_{ik} := k - t_{ik}$. Then, for those k with $s^-_{ik} > 0$, the integer t_{ik} stands for the period containing the setup of the last resource of type i torn down at period k. Consequently, m_{ik} denotes how many periods ago this setup was performed. Hence

$$m_i' = \min\{m_{ik} : 2 \leq k \leq T+1, s^-_{ik} > 0\}.$$

Let $m' := \min_i\{m_i'\}$. Now we can state the minimum lot size restriction as a simple inequality: $m' \geq m$. Equivalently, $s^-_{ik} > 0$ implies $m_{ik} \geq m$ for all i and $2 \leq k \leq T+1$. Writing this out, we get (2.6) and the equalities thereafter. Figure 2.6 outlines the assignment procedure just described.

assign the x_{i1} arbitrarily to machines
number for each i every setup $s^+_{i1}, \ldots, s^+_{iT}$ and teardown $s^-_{i2}, \ldots, s^-_{i,T+1}$ in order
for $t = 2$ **to** T **do**
  for all i **do**
    for $n = 1 + \sum_{t'=2}^{t-1} s_{it'}$ **to** $\sum_{t'=2}^{t} s^-_{it'}$ **do**
      perform teardown n on the machine with setup n
  assign, by respecting the minimum lot size constraint, $\sum_i s^+_{it}$-many secondary resources to those machines which have just been emptied

Figure 2.6. Disaggregation algorithm under minimum lot size.

As is the case with the vertical constraints, F2 with (2.6) is NP-hard. Lemmas 2.7 and 2.8, hence Corollary 2.9, hold true. We were unable to find a counterexample that disproves Theorem 2.12 or 2.14. We adapt the heuristic in §2.3.4 to the problem with minimum lot size restriction as follows: after line 7 of the algorithm in Figure 2.4, we enlarge the remaining fractional demands so as to satisfy the minimum lot size constraint. The subsequent steps are applied as is. Complexity of the new algorithm is the same as the original heuristic's.

2.5. Computational Study

We implemented the mixed-integer programs and the heuristic with C# programming language using CPLEX 12.7 as solver, on a PC with Intel(R) Core(TM)2 Quad CPU (2.40GHz) processor and 4 GB RAM, running a 64-bit Windows 7 operating system. We generated instances as follows: Given I, T, a lower bound l, an upper bound u, and a seed, the $\overline{x_i}$ are randomly generated as fractions such that $l \leq \overline{x_i} < u$ and $T\,\overline{x_i} \in \mathbb{Z}$. In other words, each $\overline{x_i}$ is of the form $a_i + \frac{b_i}{T}$, where a_i is a random integer between l and $u-1$ (inclusive), and b_i is a random integer between 0 and $T-1$. Then we generated M as a random integer between $\lceil \sum \overline{x_i} \rceil$ and $\sum \lceil \overline{x_i} \rceil$. Thus, M is greater than or equal to the smallest integer for which the problem has a feasible solution, and less than or equal to the smallest integer for which the objective value is zero (see Lemmas 2.3 and 2.7).

We shall take $l = 0$ and $T \geq 18$ throughout. Then the $\overline{x_i}$ are discrete uniform random variables assuming Tu-many values. We have $E(\overline{x_i}) = \frac{u-1}{2} + \frac{1}{2}\frac{T-1}{T} \approx \frac{u}{2}$ and $E(\lceil \overline{x_i} \rceil) = \frac{u+1}{2}$. Consequently, $E(\lceil \sum \overline{x_i} \rceil) \approx E(\sum \overline{x_i}) \approx \frac{Iu}{2}$ and $E(\sum \lceil \overline{x_i} \rceil) = \frac{I(u+1)}{2}$. Hence, expectation of M as a random variable is approximately $\frac{I(u+0.5)}{2}$. Let us designate this quantity by $\overline{M}$. Therefore, for fixed u, the average number $\overline{M}$ of machines in our instances is directly proportional to the number I of secondary resource types. We shall assume $u = 5$ so that $\overline{M} = 2.75I$.

By absolute optimality gap we mean the best (smallest) objective function value z_f minus the best (largest) lower bound z_l, and by relative optimality gap the ratio of

this difference to z_f. Absolute MIP gap tolerance is a positive number `EpAGap` such that the solver terminates processing as soon as $z_f - z_l \leq$ `EpAGap`. Relative MIP gap tolerance `EpGap` is defined analogously. We know that z^* must be an even integer (Corollary 2.9), consequently `EpAGap` can be set to $2 - \varepsilon$. We take ε to be 10^{-6} (this is the default value for `EpAGap`). It is possible that z_f be sometimes odd in the course of solution process, so the optimal value in the output is to be understood as the largest even integer less than or equal to z_f. Making use of the evenness of z^* together with the inequality $z^* \leq 2I - 2$ (Lemma 2.10), one can also show that `EpGap` can be set to $1/I$; after setting `EpAGap` as $2 - \varepsilon$, however, this is redundant.

In our first experiment, we compared F1, F1 with symmetry-breaking constraints (2.3), and F2 on six sets of 25 instances each, where $I \in \{4, 8, 16\}$ and $T \in \{18, 36\}$. A time limit of 600 seconds is chosen. Statistics summarizing the experiment results are given in Table 2.2. The column *S (Solved)* shows the number of instances solved to optimality, *G (Gap)* the average percent (relative) optimality gap, and *T (Time)* the average CPU time in seconds.

Table 2.2. Comparison of formulations F1, F1 with symmetry-breaking constraints (2.3), and F2.

		F1			F1 with (2.3)			F2		
T	I	S	G (%)	T (s)	S	G (%)	T (s)	S	G (%)	T (s)
	4	25	0	2.5	25	0	2.1	25	0	0
18	8	21	6.91	162	17	25.13	353.5	25	0	0
	16	9	57.09	416.7	6	76	468.1	25	0	9.1
	4	25	0	17.2	25	0	16.9	25	0	0
36	8	17	18.96	255.3	7	71.2	459.8	25	0	0.1
	16	4	80.02	525.5	0	100	600	25	0	20.1

Table 2.2 shows that F2 performs much better than F1. This can be explained by the fact that F1 has about M times as many variables and constraints as F2, possessing many alternative symmetric solutions. Inclusion of symmetry-breaking con-

straints (2.3) make the situation even worse. Symmetry-breaking constraints reduce the extent of feasible region that must be explored, but also make size of the linear programs to be solved at each node of the branch-and-bound tree larger, which could be the reason behind the disappointing solution performance of F1 with (2.3). In view of Table 2.2, it does not make sense to further consider F1 instead of F2 unless this is dictated by a change of underlying assumptions. Therefore we focus on F2. Although CPU time is virtually zero for small instances, this situation is bound to change for large ones as the problem is shown to be NP-hard (Theorem 2.1).

In our second experiment, we investigated the effect of setting the absolute MIP gap tolerance `EpAGap` to $2-\varepsilon$ in F2. For this purpose, solution performances of F2 (with `EpAGap` assuming its default value of 10^{-6}) and F2 with `EpAGap` $= 2-\varepsilon$ are compared in six sets of 25 instances each, where $I \in \{100, 200, 400\}$ and $T \in \{18, 36\}$. See Table 2.3 for the statistics thus obtained, which shows that this parameter's redefinition leads to a slight improvement in solution performance. As the optimal objective z^* is known to be an even integer for the model extensions as well, we shall take absolute tolerance as $2 - \varepsilon$ for all the experiments from this point on.

Table 2.3. Effect of changing the absolute MIP gap tolerance.

		F2			F2 with `EpAGap` $= 2-\varepsilon$		
T	I	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
	100	25	0	1.7	25	0	1.6
18	200	24	0.06	32.4	24	0.06	32.9
	400	23	0.13	72.6	23	0.13	72
	100	22	0.35	79.9	24	0.2	51.4
36	200	23	4.04	85.3	23	4.04	74
	400	25	0	99.3	25	0	76.3

Solution performance of our heuristic algorithm is given in Table 2.4. Gap here is defined like relative optimality gap above, with z_f replaced by the objective value z_h associated with the heuristic solution. We use as lower bound the number z_l provided

for F2 by the solver in 600 seconds. The column Solved shows the number of instances for which the heuristic solution is optimal (i.e., $z_h - z_l < 2$).

Table 2.4. Performance of the heuristic algorithm.

	Small instances				Large instances			
T	I	Solved	Gap (%)	Time (s)	I	Solved	Gap (%)	Time (s)
	4	21	16	0	100	7	10.2	0
18	8	24	4	0	200	10	8.36	0.3
	16	25	0	0	400	4	5.26	1.9
	4	22	12	0	100	8	4.35	0.1
36	8	23	1.8	0	200	11	5.02	0.4
	16	22	1.31	0	400	6	5.81	2.2

According to Table 2.4, the heuristic algorithm has found an optimal solution in more than 30% of the large instances in less than one second on average. The results are much better for the small instances. Hence, it is reasonable to feed the solver with an initial heuristic solution. We performed tests to see the effect of this on solution performance. We also ran the same experiments by incorporating the cut (2.4) into the formulation. The results are tabulated in Table 2.5.

Table 2.5. Effect of providing an initial heuristic solution and adding the cut (2.4).

		F2 with heuristic			F2 with (2.4)			F2 with both		
T	I	S	G (%)	T (s)	S	G (%)	T (s)	S	G (%)	T (s)
	100	25	0	1.5	25	0	3.9	25	0	3.9
18	200	24	0.06	31.6	24	0.06	33.6	24	0.06	33.5
	400	23	0.12	81.3	23	0.08	71.7	23	0.08	75.4
	100	23	0.19	62.2	24	0.12	34.4	24	0.08	32.6
36	200	23	0.37	72	23	0.13	78	23	0.15	73.3
	400	25	0	73.9	25	0	90.1	25	0	76.6

When we compare the results in Table 2.5 with those for F2 with `EpAGap` $= 2-\varepsilon$ in Table 2.3, we see that the initial heuristic solution leads to a significant improvement in gap for the parameter set $(I, T) = (200, 36)$. However, for $(I, T) = (100, 36)$, the number of instances solved to optimality is one less. The results for F2 with (2.4) are even better except for the relatively larger solution time for $(I, T) = (400, 36)$. The most promising implementation is F2 with heuristic and constraints (2.4). This combination yields generally the best statistics.

Next, we provide computational results for two realizations of the model extensions discussed in §2.4. First, we implemented F2 with vertical constraints (2.5), limiting by $\overline{s}$ the total number of setups and teardowns in between every two consecutive periods. We took $\overline{s} = 2$, and carried out tests for the same six sets of instances used in Table 2.2. Second, we ran F2 with minimum lot size constraints (2.6), setting a lower bound m on the number of periods a secondary resource should remain installed in a machine. We took $m = 3$. Statistics are summarised in Table 2.6. In comparison to Table 2.2, we see how the additional constraints (2.5) and (2.6) worsen solution performance, especially when I is large relative to T.

Table 2.6. Performance of the mixed-integer programs for the extended problems.

		F2 with (2.5) where $\overline{s} = 2$			F2 with (2.6) where $m = 3$		
T	I	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
	4	25	0	0	25	0	0
18	8	25	0	0.3	25	0	0.3
	16	24	0.96	26.8	24	0.95	26.5
	4	25	0	0.1	25	0	0.1
36	8	25	0	0.3	25	0	0.5
	16	22	3.05	134.5	21	3.77	158.7

Finally, in Table 2.7 we give statistics that show the performance of the heuristics described in §2.4.1 and §2.4.2 for the extended problems. Here the column *Feasible* shows the number of instances for which the algorithms were able to find a feasible

solution. Note that most of the time our heuristics could come up with an optimal schedule. The average gaps are not very large as well, although they were considerably smaller for the original problem as shown by Table 2.4.

Table 2.7. Performance of the heuristics for the extended problems.

		Vertical constraints ($\overline{s} = 2$)				Min. lot size constraints ($m = 3$)			
T	I	Feasible	S	G (%)	T (s)	Feasible	S	G (%)	T (s)
	4	25	21	16	0	24	20	16.67	0
18	8	25	20	11.33	0	24	19	11.46	0
	16	21	14	9.63	0	24	16	8.44	0
	4	25	22	12	0	25	22	12	0
36	8	24	21	6.23	0	25	22	5.28	0
	16	20	8	17.91	0	24	6	21.94	0

2.6. Conclusion and Further Research

We studied in this chapter a parallel machine multi-item lot-sizing and scheduling problem with a secondary resource, in which demands are given for the entire planning horizon rather than for every single period. We proved that the problem is NP-hard, and presented two equivalent formulations as mixed-integer linear programs. The second formulation, using aggregate integer variables instead of individual binary variables for each machine, turned out to be more efficient. We showed some properties the optimal objective value z^* must satisfy, proposed optimality conditions, and gave a heuristic algorithm. In particular, z^* must be even so that the absolute MIP gap tolerance can be set to $2-\varepsilon$. This redefinition slightly improves the solution performance of the aggregate formulation, which is also true of feeding the solver with an initial heuristic solution in general. Incorporation of the cut (2.4) leads to better results, making up the best implementation together with the heuristic.

We discussed two possible extensions to the problem, namely vertical and minimum lot size constraints. We showed how these can be formulated, and mentioned

some properties of the new problems arising thereby. Furthermore, we suggested two modifications of the heuristic. Computational tests indicate that it is significantly more time-consuming to solve these extended problems.

We conjecture for the original problem that there exist optimal schedules satisfying the following two sets of inequalities:

$$\begin{aligned} &\lfloor \overline{x_i} \rfloor \leq x_{it} \leq \lceil \overline{x_i} \rceil && \text{for all } i, t; \\ &\sum_t s_{it}^+ \leq 1, \quad \sum_t s_{it}^- \leq 1 && \text{for all } i. \end{aligned}$$

It can be interesting for further research to provide a proof or disproof of this claim, and investigate other practical situations such as some machines being temporarily unavailable within the planning horizon.

3. LOT-SIZING AND SCHEDULING IN A TWO-MACHINE FLOW SHOP TO MINIMIZE BACK ORDERS AND CHANGEOVERS

In this chapter, we consider essentially the problem defined in §1.2 for a two-machine flow shop where setup times are ignored. More precisely, there are two stages of production, each consisting of one machine. The items produced in the first and second stages are referred to as intermediate and end items, respectively. The bill-of-materials structure is serial so that there is a one-to-one correspondence between the sets of intermediate and end items. We assume without loss of generality that all gozinto factors are 1. There are a number of requirements for end items. Different requirements may be associated with the same end item. Possibly, requirements are of size greater than 1 so that splitting may be necessary. Although setup times are ignored, a constant cost is incurred when production on a machine is switched from one item to another. We call this a changeover. Back orders are allowed to make sure that any instance will be feasible. The objective is to first minimize back orders and then the number of changeovers. In other words, among solutions with the least amount of back orders possible, we want to find one with the least number of changeovers.

Production of an end item cannot start unless relevant intermediate items have positive stock. The satisfaction of this hard constraint is especially difficult to control in formulations based on a purely continuous time scale. Therefore, we assume that the planning horizon is divided into buckets. Bucket forming can be viewed as the result of scattering some inventory checkpoints throughout the planning horizon. These checkpoints, which are usually equidistant, define material flow by means of balance equations.

In large-bucket models, a number of items can be produced in a period. Capacitated lot-sizing problem (CLSP) is an umbrella term for many such formulations. In small-bucket models, on the other hand, only one or two items can be produced in

a period. Discrete lot-sizing and scheduling problem (DLSP), continuous setup lot-sizing problem (CSLP), and proportional lot-sizing and scheduling problem (PLSP) are examples of small-bucket models. The trademark of DLSP is its all-or-nothing assumption: a machine is either idle or works at full capacity in a period. CSLP is very similar to DLSP except that production quantities can be of any continuous size. PLSP allows for the production of two distinct items within a period; still, only one setup can be made. In all these formulations, buckets are usually fixed and identical. General lot-sizing and scheduling problem (GLSP) can also be characterized as a small-bucket model in view of the assumptions regarding microperiods. Detailed information about these problems can be found in Drexl and Kimms [31] and Copil *et al.* [5].

Kimms [29] studies a multi-level PLSP in depth. He discusses instance generation, lower-bounding techniques, and several heuristic solution procedures. Back-ordering is not allowed, so instances may turn out to be infeasible. Inventory costs are also taken into account. The book by Pochet and Wolsey [3] is a basic reference for polyhedral analysis of lot-sizing models. The authors mainly treat single-item lot-sizing, but there is also a chapter on multi-level problems. Our formulation will be based on CSLP. In order to guarantee a feasible material flow, we assume a lead time of one period for internal demand.

3.1. Mixed-Integer Linear Formulation

Let i, m, t denote item, machine, and time indices, respectively. Let a_{mi} designate the production rate in units per period for item i in machine m. We write c_1 for per unit and per period back order cost, and c_2 for changeover cost. As usual, d_{it} stands for the demand to be fulfilled by the end of period t. We represent by b_{it} the amount of end item i back-ordered at the end of period t. Variables z_{mt} assume the value 1 if there is a changeover in machine m in period t, and 0 otherwise ($t > 1$). The symbols $s_{mit}, x_{mit}, y_{mit}$ denote the stock, production, and setup state variables. Indices, parameters, and decision variables can be found in Table 3.1.

Table 3.1. Indices, parameters, and decision variables.

Symbol(s)	Explanation
i, m, t	item, machine, time indices
a_{mi}	production rate for item i in machine m (units/period)
c_1	per unit and per period back order cost
c_2	changeover cost
d_{it}	demand for end item i in period t
b_{it}	back order of end item i at the end of period t
s_{mit}	stock amount of item i in machine m at the end of period t
x_{mit}	production amount of item i in machine m in period t
y_{mit}	binary variable for the setup state of item i in machine m in period t
z_{mt}	binary changeover variable in machine m in period t ($t > 1$)

The mixed-integer linear programming (MIP) formulation of the problem is given below. All expressions involving z_{mt} are subject to $t > 1$:

$$\min \quad c_1 \sum_{i,t} b_{it} + c_2 \sum_{m,t} z_{mt} \tag{3.1a}$$

$$\text{s.t.} \quad s_{2i,t-1} + x_{2it} = d_{it} + s_{2it} + b_{i,t-1} - b_{it} \qquad \text{for all } i, t \tag{3.1b}$$

$$s_{1i,t-1} + x_{1i,t-1} = x_{2it} + s_{1it} \qquad \text{for all } i, t \tag{3.1c}$$

$$\sum_{i} y_{mit} \leq 1 \qquad \text{for all } m, t \tag{3.1d}$$

$$x_{mit} \leq a_{mi} y_{mit} \qquad \text{for all } m, i, t \tag{3.1e}$$

$$z_{mt} \geq y_{mit} - y_{mi,t-1} \qquad \text{for all } m, i, t \tag{3.1f}$$

$$x_{mit}, s_{mit}, b_{it} \geq 0;\ y_{mit} \in \{0, 1\};\ z_{mt} \in [0, 1] \qquad \text{for all } m, i, t. \tag{3.1g}$$

Constraints (3.1b) and (3.1c) are inventory balance equations for independent and dependent demand. We assume $s_{2i0} = b_{i0} = 0$ and $s_{1i0} = x_{1i0} = 0$ unless stated otherwise. Constraint (3.1d) says that on a machine at most one item can be produced in a period. Constraint (3.1e) indicates that the machine capacity is finite. Note that

y_{mit} can be 1 even if x_{mit} is zero; consequently, setup state is preserved during idle periods as in CSLP [31]. Constraint (3.1f) keeps account of changeovers. By virtue of the objective, the continuous variables z_{mt} will always assume binary values.

The coefficient c_1 in the objective function (3.1a) must be large enough with respect to c_2 to ensure that any optimal solution will yield the least amount of back orders possible. In general, if f_1 and f_2 are primary and secondary objectives in a mixed-integer linear program with feasible set S, and if

- $f_1(x) \in \mathbb{Z}$ for any extreme point x of the convex hull of S, and
- there exists u, v such that $u \leq f_2(x) \leq v$ for all $x \in S$,

then one can solve the program in one phase for $cf_1 + f_2$, where $c > v - u$, instead of solving it in two phases (first for f_1 and then for f_2 with an additional constraint).

3.2. Complexity

Changeover minimization is NP-hard even on a single machine [15]. However, the complexity of back order minimization ($c_2 = 0$), to the best of our knowledge, has not been addressed in the literature. If requirements cannot be split, the question as to whether there exists a solution without back orders is precisely the pure scheduling problem $F2 \mid d_i \mid \cdot$ that asks for a feasible schedule in a two-stage flow shop with deadlines. By a reduction from Subset Sum, Lenstra *et al.* [17] show that $F2 \mid d_i \mid \cdot$ is NP-complete. Koulamas [18] gives an alternative proof using Partition. Later, Lin [19] strengthens this result by doing a reduction from 3-Partition. All of these proofs strictly depend on the assumption that jobs are irreducible, so they do not apply in our case.

Practically, back order minimization is solved much faster than changeover minimization. This can be explained by the fact that linear programming (LP) bounds are usually very strong for the former problem. In fact, most of the time the LP relaxation yields the optimal value for back order minimization (see Table 3.3). Nevertheless,

this is not always the case as the following example shows: Suppose there are two end items i, j and three requirements $d_{i2} = d_{j3} = d_{i4} = 2$. Production rates for the first and second stages are 4 and 2 units per period for i, and 1 and 2 units per period for j. Figure 3.1 shows an optimal solution where the requirement for j is back-ordered for one period. The associated objective value is 2. For the LP relaxation, however, the optimal value is 2/3. An optimal solution is given by $y_{1i2} = 1, y_{1i3} = 1/3, y_{1i4} = 2/3, y_{1j3} = 2/3, y_{1j4} = 1/3, y_{2i1} = 2/3, y_{2i3} = 1/3, y_{2j1} = 1/3, y_{2j2} = 1, y_{2j3} = 2/3$ (Figure 3.2).

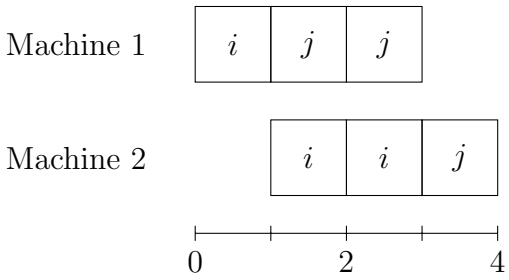

Figure 3.1. An optimal solution for the MIP formulation ($z^*_{\text{MIP}} = 2$).

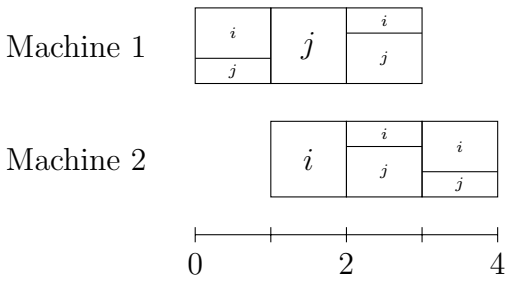

Figure 3.2. An optimal solution for the LP relaxation ($z^*_{\text{LP}} = 2/3$).

In Chapter 4, we show that under the all-or-nothing assumption the question as to whether there exists a solution without back orders can be answered in polynomial time. Consequently, the minimization of maximum lateness is polynomially solvable. Note that this is not the same problem as (3.1) with $c_2 = 0$ (the minimization of total back orders).

3.3. Optimality Conditions

Proposition 3.1. *There exists an optimal schedule such that*
(i) $x_{1it} = a_{1i}y_{1it}$ *for all* i, t*;*
(ii) $\sum_i y_{mit} = 1$ *for all* m, t*.*

Proof. (i) If $\sum_i y_{mi1} < 1$, any y_{mi1} can be defined to be 1. Now assume $\sum_i y_{mi,t-1} = 1$, but $\sum_i y_{mit} < 1$. Then $y_{mi_0,t-1} = 1$ for some i_0. We can set $y_{mi_0t} = 1$ without increasing the objective function value.
(ii) If $x_{1it} < a_{1i}y_{1it}$, then redefine x_{1it} and the stock variables $s_{1it'}$ for $t' \geq t$ so that $x_{1it} = a_{1i}y_{1it}$ and (3.1c) holds. □

Let z_{mit} be 1 if there is a changeover for item i in machine m in period t, and 0 otherwise ($t > 1$). Changeovers can also be formulated via z_{mit} instead of z_{mt}: Constraint (3.1f) is replaced by

$$z_{mit} \geq y_{mit} - y_{mi,t-1} \quad \text{for all } m, i, t; \tag{3.2}$$

and the sum $\sum_{m,t} z_{mt}$ in the objective function (3.1a) is replaced by $\sum_{m,i,t} z_{mit}$.

Proposition 3.2. *For the formulation with* z_{mit}*, there exists an optimal schedule such that* $z_{mit} \leq y_{mit}$ *and* $z_{mit} \leq 1 - y_{mi,t-1}$ *for all* m, i, t*.*

Proof. If $z_{mit} > y_{mit}$ or $z_{mit} > 1 - y_{mi,t-1}$, then $y_{mit} = 0$ or $y_{mi,t-1} = 1$. In any case, we can set $z_{mit} = 0$ so that the stated inequalities are fulfilled. □

The next result is adapted from Proposition 12.4 in the textbook by Pochet and Wolsey [3].

Proposition 3.3. *If the constraints in Proposition 3.2 are added to the formulation with* z_{mit}*, then the inequalities* $z_{mit} + \sum_{j \neq i}(y_{mjt} - z_{mjt}) \leq 1 - y_{mi,t-1}$ *are valid for all* m, i, t*.*

Proof. Since $y_{mjt} - z_{mjt} \geq 0$ and $\sum_j y_{mjt} \leq 1$, we have $0 \leq \sum_{j \neq i}(y_{mjt} - z_{mjt}) \leq 1$. In fact, the sum $\sum_{j \neq i}(y_{mjt} - z_{mjt})$ equals 1 if and only if $y_{mjt} - z_{mjt} = 1$, or equivalently $y_{mjt} = y_{mj,t-1} = 1$, for some $j \neq i$. The proposed inequality simply says that the three possibilities associated with the binary expressions z_{mit}, $\sum_{j \neq i}(y_{mjt} - z_{mjt})$, $y_{mi,t-1}$ are mutually exclusive. □

The following proposition gives a set of inequalities that are satisfied in an optimal solution provided that there is a solution without back orders.

Proposition 3.4. *Let* $I_t := \{i \in I : d_{it'} > 0 \text{ for some } t' \leq t\}$ *and* $n_t := |I_t|$. *If there exists a solution without back orders, then in any optimal solution the inequalities* $\sum_{t' \leq t} \sum_{i \in I_t} z_{2it'} \geq n_t - 1$ *must hold for all* $t > 1$.

Proof. If there is a schedule without back orders, then there must be at least $n_t - 1$ changeovers on the second machine up to period t since n_t different types of end items are demanded until this point in the planning horizon. Clearly, only those variables $z_{2it'}$ with $t' \leq t$ and $i \in I_t$ need be considered. □

It does not make sense to add the inequalities given in Proposition 3.4 for every $t > 1$. Indeed, the inequality for t_1 implies the one for $t_2 > t_1$ as long as $n_{t_1} = n_{t_2}$, so it suffices to consider the jumping points of n_t.

One of the most important favorable properties of an MIP formulation is the strength of its LP relaxation [49]. Next, we look at whether the constraints given in the previous propositions improve the LP bound or not. We shall write a^+ for $\max\{a, 0\}$.

Proposition 3.5. *The constraints given in Proposition 3.1 do not improve the LP bound.*

Proof. If $\sum_i y_{mi1} < 1$, any y_{mi1} can be increased to make the sum $\sum_i y_{mi1}$ equal to 1. Now assume $\sum_i y_{mi,t-1} = 1$, but $\sum_i y_{mit} < 1$. Those y_{mit} for which the difference $y_{mi,t-1} - y_{mit}$ is positive can be increased to make the sum $\sum_i y_{mit}$ equal to 1. This

is possible since $\sum_i (y_{mi,t-1} - y_{mit})^+ \geq \sum_i (y_{mi,t-1} - y_{mit}) = \sum_i y_{mi,t-1} - \sum_i y_{mit} = 1 - \sum_i y_{mit}$. Argument for the second set of constraints is the same as in the proof of Proposition 3.1. □

The formulation with z_{mit} may improve the LP bound as demonstrated by the following example: Suppose there are three end items i, j, k with production rates being 1 unit per period on both machines, and there are three requirements $d_{i2} = d_{j4} = d_{k4} = 1$. The optimal objective values for the LP relaxations of the original formulation (with z_{mt}) and the formulation with z_{mit} are 1 and 2, respectively. A schedule that is optimal in both cases is given in Figure 3.3.

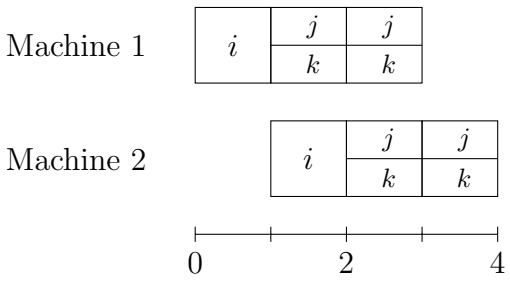

Figure 3.3. LP bound of the formulation with z_{mit} may be strictly better than that of the formulation with z_{mt}.

Proposition 3.6. *The constraints given in Propositions 3.2 and 3.3 do not improve the LP bound of the formulation with* z_{mit}.

Proof. Starting with any feasible solution, we can set $z_{mit} = (y_{mit} - y_{mi,t-1})^+$ so that the inequalities $z_{mit} \leq y_{mit}$ and $z_{mit} \leq 1 - y_{mi,t-1}$ are satisfied. If $y_{mit} \geq y_{mi,t-1}$, then $z_{mit} = y_{mit} - y_{mi,t-1}$, hence $\sum_{j \neq i} y_{mjt} \leq 1 - y_{mit} = 1 - (z_{mit} + y_{mi,t-1})$, implying $z_{mit} + \sum_{j \neq i} (y_{mjt} - z_{mjt}) \leq z_{mit} + \sum_{j \neq i} y_{mjt} \leq 1 - y_{mi,t-1}$. If $y_{mit} < y_{mi,t-1}$, then $z_{mit} = 0$. Since $z_{mjt} \geq y_{mjt} - y_{mj,t-1}$, we have $y_{mj,t-1} \geq y_{mjt} - z_{mjt}$, so $\sum_{j \neq i} (y_{mjt} - z_{mjt}) \leq \sum_{j \neq i} y_{mj,t-1} \leq 1 - y_{mi,t-1}$. Therefore, the inequality $z_{mit} + \sum_{j \neq i} (y_{mjt} - z_{mjt}) \leq 1 - y_{mi,t-1}$ holds in any case. □

Just like the formulation with z_{mit}, the inequalities in Proposition 3.4 may improve the LP bound in case there is a solution without back orders. Examples can be constructed easily.

3.4. All-or-Nothing Variant

All-or-nothing assumption says that in each period a machine is either idle or works at full capacity. In the original formulation, it is enough to convert (3.1e) into an equality in order to have this assumption fulfilled:

$$x_{mit} = a_{mi} y_{mit} \quad \text{for all } m, i, t. \tag{3.3}$$

Let $Y_{mit} := \sum_{t'=1}^{t} y_{mit'}$ and $D_{it} := \sum_{t'=1}^{t} d_{it'}$. A reformulation without x_{mit} and s_{mit} is possible for this new problem:

$$\begin{aligned}
\min \quad & c_1 \sum_{i,t} b_{it} + c_2 \sum_{m,i,t} z_{mit} && && \text{(3.4a)}\\
\text{s.t.} \quad & s_{2i0} + a_{2i} Y_{2it} \geq D_{it} - b_{it} && \text{for all } i, t && \text{(3.4b)}\\
& a_{2i} Y_{2it} \leq s_{1i0} + a_{1i} Y_{1i,t-1} && \text{for all } i, t && \text{(3.4c)}\\
& \sum_{i} y_{mit} \leq 1 && \text{for all } m, t && \text{(3.4d)}\\
& z_{mit} \geq y_{mit} - y_{mi,t-1} && \text{for all } m, i, t && \text{(3.4e)}\\
& b_{it} \geq 0;\ y_{mit} \in \{0,1\};\ z_{mit} \in [0,1] && \text{for all } m, i, t. && \text{(3.4f)}
\end{aligned}$$

In view of Proposition 3.1, the original problem (3.1), too, can be reformulated in a similar fashion without using x_{1it} and s_{mit} (we need to retain x_{2it}).

3.5. Computational Study

We implemented the mixed-integer programs with C# programming language using CPLEX 12.7 as solver, on a PC with Intel(R) Core(TM) i5-2450M CPU (2.50GHz) processor and 4 GB RAM, running a 64-bit Windows 7 operating system.

Our instance generation routine takes four parameters: the number I of end items, the number T of time periods, the average number $\overline{n}$ of requirements per item, and the utilization factor α. A pseudocode is given in Figure 3.4. Here, $\{x_1, \ldots, x_n\}$ denotes the outcome of a discrete probability distribution with support $x_1, \ldots, x_n$, and n_i is the number of requirements for end item i. The distributions $\{0.5, 1, 2\}$, $\{\overline{n}-1, \overline{n}, \overline{n}+1\}$, $\{l, l+2\}$ are defined such that their expected values are 1, $\overline{n}$, μ. Note that μ approximately equals $T/(I\overline{n})$, namely the average requirement size in number of periods. The factors $\overline{n}/(\overline{n}+1)$ and $(T-1)/T$ in the definition of μ account for the uniform scattering of requirements throughout the planning horizon and the one-period lead time, respectively. In principle, the a_{2i} could be assigned any constant; however, in accordance with the definition of a_{1i}, we set them to 2, which is the smallest number such that all a_{mi} are guaranteed to be integers. Hence, total back order quantity $\sum_{i,t} b_{it}$ will assume an integer value in every basic solution. Also, we let l be even so as to make sure that all dependent demands are integer multiples of a period.

$\mu \leftarrow \alpha \times (T-1)/T \times \overline{n}/(\overline{n}+1) \times T/(I\overline{n})$
$l \leftarrow$ the largest even integer less than or equal to μ
for all i **do**
  $a_{2i} \leftarrow 2$
  $a_{1i} \leftarrow \{0.5, 1, 2\} \times a_{2i}$
  $n_i \leftarrow \{\overline{n}-1, \overline{n}, \overline{n}+1\}$
  pick n_i periods arbitrarily
  for $r = 1$ **to** n_i **do**
    $q_r \leftarrow \{l, l+2\} \times a_{2i}$

Figure 3.4. Instance generation.

We performed five experiments. In all of them, we consider three (I, T) combinations, namely (2, 18), (5, 45), and (10, 90). We take $\overline{n} = 2$ and $\alpha = 0.8$. For each parameter set, 10 random instances are generated. The time limit is 180 seconds.

In our first experiment, we compare the one-phase and the two-phase approaches together with the earliest-due-date (EDD) heuristic. Recall that the primary objective is to minimize back orders, and the secondary objective is to minimize the number of changeovers. Hence, it is natural to solve the problem in two phases, namely by first minimizing $\sum_{i,t} b_{it}$ subject to (3.1b)–(3.1e) to obtain an optimal value b^*, and then minimizing $\sum_{m,t} z_{mt}$ subject to (3.1b)–(3.1f) plus an additional constraint $\sum_{i,t} b_{it} \leq b^*$. Alternatively, as $\sum_{i,t} b_{it} \in \mathbb{Z}$ in any basic solution and $0 \leq \sum_{m,t} z_{mt} \leq 2(T-1)$, the problem can also be solved in one phase provided that $c_1/c_2 > 2(T-1)$ (see §3.1). A pseudocode for the EDD heuristic is given in Figure 3.5.

In the two-phase method, if the first phase is solved within the time limit, we start the solver in the second phase with the optimal solution obtained in the first phase. We apply the one-phase method with $c_1 = 2T$ and $c_2 = 1$. Statistics are summarized in Table 3.2. Here the column *Solved* shows the number of instances solved to optimality, and *Gap* the average percent (relative) optimality gap. In the gap computation for the two-phase method, lower bound is defined as the lower bound of the first phase if the first phase could not be solved to optimality within the time limit, and as b^* plus the lower bound of the second phase otherwise. Heuristic gap is computed against the lower bound of the one-phase solution. Run time of the heuristic is virtually zero for all the instances.

```
calculate the dependent demands
produce intermediate items according to EDD
for all t do
    produce an end item according to EDD by respecting intermediate item stocks
    update intermediate item stocks throughout the planning horizon
```

Figure 3.5. EDD heuristic.

According to Table 3.2, there does not seem to be a significant difference in the solution performance of the one- and the two-phase methods, and the EDD heuristic

generally does not yield satisfactory results for this problem.

Table 3.2. Comparison of the one-phase and the two-phase approaches together with the EDD heuristic.

	One-Phase		Two-Phase		Heuristic	
(I, T)	Solved	Gap (%)	Solved	Gap (%)	Solved	Gap (%)
(2, 18)	10	0	10	0	8	5.08
(5, 45)	4	16.13	5	15.61	0	29.95
(10, 90)	1	45.89	1	45.02	0	62.22

In our second experiment, we investigate back order minimization only ($c_1 = 1, c_2 = 0$). In this case, the model can be reduced by omitting the changeover variables together with Constraint (3.1f) as in the first phase of the two-phase method. Statistics are summarized in Table 3.3. The column *Time* shows the average CPU time in seconds, *Solved at Root* the number of instances for which the LP relaxation yields the optimum, and *Gap at Root* the average percent optimality gap computed with respect to the LP bound. We observe that LP bounds are very strong for back order minimization, and most of the time the branch-and-bound search terminates at the root node.

Table 3.3. Computational results for back order minimization.

(I, T)	Solved	Gap (%)	Time (s)	Solved at Root	Gap at Root (%)
(2, 18)	10	0	0	10	0
(5, 45)	10	0	0.1	8	0.90
(10, 90)	10	0	0.5	8	0.26

In our third experiment, we compare the formulations with z_{mt} and z_{mit}. We perform the comparison by using the one-phase method with $c_1 = 2T$ and $c_2 = 1$ as before; similarly for the next two experiments. Statistics are given in Table 3.4. The formulation with z_{mit} outperforms the one with z_{mt}. So we use the formulation with

z_{mit} from this point on.

Table 3.4. Comparison of the formulations with z_{mt} and z_{mit}.

	Formulation with z_{mt}			Formulation with z_{mit}		
(I, T)	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
(2, 18)	10	0	0	10	0	0
(5, 45)	4	16.13	109.7	10	0.01	4.3
(10, 90)	1	45.89	163	7	5.58	101.8

In our fourth experiment, we compare for the all-or-nothing variant of the problem the formulation with z_{mit} and the reformulation (3.4). Statistics are shown in Table 3.5. There does not seem to be a substantial difference in the performance of the two formulations.

Table 3.5. Comparison of the all-or-nothing variant with (3.3) and the reformulation (3.4).

	Formulation with (3.3)			Reformulation (3.4)		
(I, T)	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
(2, 18)	10	0	0.1	10	0	0
(5, 45)	10	0	2.5	10	0	4.1
(10, 90)	5	2.68	118.4	5	2.48	111.8

In our fifth experiment, we test the effect of including a priori in (3.1) the cuts given in Proposition 3.4 for 10 random instances with no back order for each of the three (I, T) combinations. Table 3.6 shows that the cuts may be useful for large instances. On the other hand, the cuts given in Propositions 3.1, 3.2, and 3.3 do not improve solution performance; to the contrary, they give rise to larger gaps and CPU times on average.

Table 3.6. Effect of adding the cuts in Proposition 3.4 for instances without back orders.

	Formulation without cuts			Formulation with cuts		
(I, T)	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
(2, 18)	10	0	0	10	0	0
(5, 45)	10	0	18.3	9	0.62	24.9
(10, 90)	5	10.26	121.7	5	6.87	106.8

3.6. Conclusion

We investigated a lot-sizing and scheduling problem in a two-machine flow shop where the bill-of-materials structure is serial and the objective is to first minimize back orders and then the number of changeovers. We proposed several optimality conditions and reformulations. There does not seem to be a significant difference in the performance of the one- and the two-phase methods. Back order minimization is solved much faster compared to changeover minimization. Surprisingly, the reformulation with z_{mit}, although it has more variables, turns out to be much better than the original formulation with z_{mt}. Our computational study, in general, supports the view that adding a cut to a formulation makes sense only if it strengthens the LP bound.

4. A POLYNOMIAL ALGORITHM FOR LOT-SIZING AND SCHEDULING IN A TWO-MACHINE FLOW SHOP

We investigate a multi-item discrete lot-sizing and scheduling problem in a two-machine flow shop. There is external demand for end items, and internal demand for intermediate items. All-or-nothing assumption is valid so that a machine is either idle or works at full capacity in a period. The question is whether there exists a feasible production plan to satisfy all requirements. In case of serial bill-of-materials, this is closely related to the strongly NP-hard pure scheduling problem of minimizing maximum lateness in a two-stage flow shop. In contrast with the complexity of $F2 \mid\mid L_{\max}$, we propose a polynomial algorithm that provides an answer to the question posed in $O(T^2)$ time, where T is the length of the planning horizon.

4.1. Introduction

We consider a two-machine flow shop. On the first and second machines, intermediate and end items are produced, respectively. Planning horizon is divided into finitely many time periods. There is external demand for end items, and internal demand for intermediate items. Bill-of-materials structure is divergent so that each end item has a unique predecessor. At most one type of item can be produced in a period; moreover, machines are either idle or work at full capacity. This is the so-called all-or-nothing assumption. For satisfaction of dependent demand, a lead time of one period is supposed to guarantee feasible material flow in between the two stages. The question is whether there exists a feasible production plan to satisfy all requirements.

In most lot-sizing models, the time axis is discrete and the objective is basically to minimize the sum of setup and inventory holding costs. For single item and level, the uncapacitated problem can be solved in polynomial time [2]. This is also true of the constant capacity case [50–52]. If capacities vary throughout the planning horizon, the problem becomes NP-hard [53, 54]. For single item and multiple levels (production

in series), there are polynomial algorithms for the uncapacitated version [55,56] as well as for capacitated problems at the first level [57–59] or at both levels [60, 61]. On the other hand, the discrete lot-sizing and scheduling problem, consisting of multiple items and a single level, is strongly NP-hard [62, 63].

In the particular case of serial bill-of-materials, a closely related pure scheduling problem is that of finding a feasible solution in a two-stage flow shop with deadlines. Researchers usually prefer to study the polynomially equivalent optimization version $F2 \mid\mid L_{\max}$, where the objective is to minimize maximum lateness. This problem is strongly NP-hard [17–19]. The proofs strictly depend on the classical assumption of flow shop scheduling that jobs cannot start being processed on a machine unless they are completed on the previous one.

In contrast with the complexity of $F2 \mid\mid L_{\max}$, we show that the feasibility problem for discrete lot-sizing and scheduling in a two-machine flow shop is efficiently solvable by an algorithm of run-time order $O(T^2)$, where T is the length of the planning horizon. To the best of our knowledge, this is the first result in the literature concerning solvability of a multi-item, multi-level lot-sizing and scheduling problem with constant capacities at each level.

Outline of the chapter is as follows: in Section 4.2, we provide a binary linear programming formulation of the problem, and set a theoretical basis for the algorithm. In Section 4.3, we give the algorithm's pseudocode, discuss its complexity, demonstrate it on an example, and indicate possible extensions. We conclude the chapter in Section 4.4 with some future research opportunities.

4.2. Theoretical Results

Let I, J denote the sets of all end and intermediate items, respectively, and i, j the indices thereof. There exists a divergent bill-of-materials structure. Let $j(i)$ be the index of the unique predecessor of i, and $I(j)$ the set of all successors of j. Time periods are indexed by t as usual. By redefining units if necessary, we may assume

that production rates are 1 unit per period. Let $r(i)$ denote the gozinto factor for i, namely the number of units of $j(i)$ required to produce one unit of i. We note that $r(i)$ is possibly fractional; in fact, it can be any positive real number. Let $d(i,t)$ be the demand for end item i to be fulfilled in period t. Binary variables y_{1jt} take the value 1 if item j is produced on machine 1 in period t, and 0 otherwise; y_{2it} is defined similarly. A list of indices, sets, parameters, and decision variables for the problem can be found in Table 4.1.

Table 4.1. Indices, sets, parameters, and decision variables for the problem.

Symbol(s)	Explanation
i, j, t	indices for end items, intermediate items, time periods
$j(i)$	index of the unique predecessor of i
I, J	sets of all end and intermediate items
$I(j)$	set of all successors of j
$r(i)$	(possibly fractional) number of units of intermediate item $j(i)$ required to produce one unit of end item i
$d(i,t)$	demand for end item i in period t
y_{1jt}	binary variable defined as 1 if item j is produced on machine 1 in period t, and as 0 otherwise
y_{2it}	binary variable defined as 1 if item i is produced on machine 2 in period t, and as 0 otherwise

As a binary linear program, the problem can be formulated as follows:

$$\sum_{t'=1}^{t} y_{2it'} \geq \sum_{t'=1}^{t} d(i,t') \qquad \text{for all } i,t \tag{4.1a}$$

$$\sum_{t'=1}^{t-1} y_{1jt'} \geq \sum_{t'=1}^{t} \sum_{i \in I(j)} r(i) y_{2it'} \qquad \text{for all } j,t \tag{4.1b}$$

$$\sum_{j} y_{1jt} \leq 1, \quad \sum_{i} y_{2it} \leq 1 \qquad \text{for all } t \tag{4.1c}$$

$$y_{1jt}, y_{2it} \in \{0,1\} \qquad \text{for all } i,j,t. \tag{4.1d}$$

Constraints (4.1a) and (4.1b) guarantee the satisfaction of independent and dependent demand throughout the planning horizon. We assume a lead time of one period as indicated by the upper bound of summation on the left-hand side of (4.1b). This sum is understood to be zero for $t = 1$. Constraints (4.1c) and (4.1d) state that in a period a machine is either idle or works at full capacity to produce one and only one type of item.

The question is whether the solution space defined by (4.1) is nonempty or not. In the former case, we say that the instance is feasible. We call two instances equivalent if one's feasibility implies that of the other's. Let us designate the set of integers by $\mathbb{Z}$, and the ceiling function that rounds up its argument to the nearest integer by $\lceil\ \rceil$.

Proposition 4.1. *Let $i \in I$ and suppose $d(i,t) \notin \mathbb{Z}$ for some t. Let $t_1 := \min\{t : d(i,t) \notin \mathbb{Z}\}$.*

- *If $d(i,t) = 0$ for all $t > t_1$, we can redefine $d(i,t_1)$ as $\lceil d(i,t_1) \rceil$.*
- *If $d(i,t) > 0$ for some $t > t_1$, let $t_2 := \min\{t > t_1 : d(i,t) > 0\}$ and $s := \lceil d(i,t_1) \rceil - d(i,t_1)$. If $s < d(i,t_2)$, we can redefine $d(i,t_1)$ and $d(i,t_2)$ as $\lceil d(i,t_1) \rceil$ and $d(i,t_2) - s$, respectively; if $s \geq d(i,t_2)$, we can redefine $d(i,t_1)$ and $d(i,t_2)$ as $d(i,t_1) + d(i,t_2)$ and 0, respectively.*

Proof. In a feasible schedule, $\lceil d(i,t_1) \rceil$-many periods will be assigned to the production of item i anyway. Hence, if $d(i,t_2) > 0$ for some $t_2 > t_1$, a certain portion of this demand, namely the minimum of s and $d(i,t_2)$, can be transferred to $d(i,t_1)$. □

By repeated application of Proposition 4.1, we may obtain an equivalent instance where all demand quantities are integers. From this point on, we assume without loss of generality that $d(i,t) \in \mathbb{Z}$ for all i,t. The next result gives a simple necessary condition for feasibility that will be needed later.

Proposition 4.2. *An instance is feasible only if the inequalities*

$$\sum_i \sum_{t'=1}^{t} d(i,t') \le t-1,$$
$$\sum_j \sum_{i\in I(j)} \sum_{t'=1}^{t} r(i)d(i,t') \le t-1$$

hold for all t*.*

Proof. It follows by letting $t = 1$ in (4.1b) that $y_{2i1} = 0$ for all i. Consequently, $\sum_i \sum_{t'=1}^{t} d(i,t') \le \sum_i \sum_{t'=2}^{t} y_{2it'} = \sum_{t'=2}^{t} \sum_i y_{2it'} \le t-1$ by (4.1a) and (4.1c). The second inequality is proved similarly. □

The following proposition says that if demands for two end items having the same predecessor are due at the same period, then the one for the item with smaller gozinto factor can be carried one period to the left.

Proposition 4.3. *Let* $i_1, i_2 \in I(j)$ *with* $d(i_1,t) \ge 1$ *and* $d(i_2,t) \ge 1$*. If* $r(i_1) \le r(i_2)$*, we can decrease* $d(i_1,t)$ *by 1 and increase* $d(i_1,t-1)$ *by 1.*

Proof. Consider a feasible schedule. If the last period allocated to satisfy $d(i_1,t)$ takes place after that of $d(i_2,t)$, the production in those two periods can be exchanged without creating any infeasibility due to internal demand since $r(i_1) \le r(i_2)$. □

By applying Proposition 4.3 iteratively, the demand data can be modified to create an equivalent instance such that

$$\sum_{i\in I(j)} d(i,t) \le 1 \text{ for all } j,t. \tag{4.2}$$

In particular, every nonzero demand is of size 1. If the stronger inequality

$$\sum_i d(i,t) \le 1 \text{ for all } t \tag{4.3}$$

holds true, then feasibility can be checked easily, as stated in the next proposition.

Proposition 4.4. *If $\sum_i d(i,t) \leq 1$ for all t, feasibility can be checked in polynomial time.*

Proof. Under the given hypothesis, there is one and only one way to produce as late as possible on the second machine. Calculate the dependent demands accordingly. The instance is feasible if and only if the dependent demands can be met on time. We can check this by constructing an earliest-due-date-first schedule after a preprocessing step as suggested by Proposition 4.1. □

The idea is to obtain an equivalent instance satisfying the hypothesis of Proposition 4.4. A main result to this end is Proposition 4.5, which implies that it is enough to have integer (possibly zero) gozinto factors.

Proposition 4.5. *Let $i_1, i_2 \in I$ with $d(i_1,t) \geq 1$, $d(i_2,t) \geq 1$, and $0 \leq r(i_1) \leq r(i_2)$. If $r(i_1)$ and $r(i_2)$ are integers, then we can decrease $d(i_1,t)$ by 1 and increase $d(i_1,t-1)$ by 1.*

Proof. Similar to the proof of Proposition 4.3. □

It will be convenient to introduce an alternative representation for instances that satisfy (4.2). For each j, let ρ_j be a vector defined as

$$\rho_j(t) := \begin{cases} r(i), & \text{if } d(i,t) = 1 \text{ for some } i \in I(j); \\ -1, & \text{otherwise.} \end{cases} \tag{4.4}$$

Since (4.2) is satisfied, the ρ_j are well-defined. The -1 appearing in the definition has no special meaning; in fact, any negative number would also do. We make an important observation: each unit demand can be associated with a different end item. More precisely, if $d(i,t) \geq 1$, we can decrease $d(i,t)$ by 1 and enlarge I by adding a new item i' such that $j(i') = j(i)$, $r(i') = r(i)$, and $d(i',t) = 1$. Therefore, every instance that satisfies (4.2) is completely described by the vectors ρ_j.

Proposition 4.6. *Consider an instance that satisfies* (4.2). *Let* $j \in J$ *and suppose* $\rho_j(t) > 0$ *and* $\rho_j(t) \notin \mathbb{Z}$ *for some* t. *Let* $t_1 := \min\{t : \rho_j(t) > 0 \text{ and } \rho_j(t) \notin \mathbb{Z}\}$.

- *If* $\rho_j(t) \leq 0$ *for all* $t > t_1$, *we can redefine* $\rho_j(t_1)$ *as* $\lceil \rho_j(t_1) \rceil$.
- *If* $\rho_j(t) > 0$ *for some* $t > t_1$, *let* $t_2 := \min\{t > t_1 : \rho_j(t) > 0\}$ *and* $s := \lceil \rho_j(t_1) \rceil - \rho_j(t_1)$. *If* $s < \rho_j(t_2)$, *we can redefine* $\rho_j(t_1)$ *and* $\rho_j(t_2)$ *as* $\lceil \rho_j(t_1) \rceil$ *and* $\rho_j(t_2) - s$, *respectively; if* $s \geq \rho_j(t_2)$, *we can redefine* $\rho_j(t_1)$ *and* $\rho_j(t_2)$ *as* $\rho_j(t_1) + \rho_j(t_2)$ *and* 0, *respectively.*

Proof. Similar to the proof of Proposition 4.1. □

By using Proposition 4.6 finitely many times, we obtain an instance with integer gozinto factors. The result then follows from Propositions 4.5 and 4.4, as mentioned above. We formalize this argument in the next section.

4.3. Algorithm

Based on the theoretical results of the previous section, we can determine whether the system (4.1) is feasible or not: First of all, we obtain an equivalent instance where all demand quantities are integers (Proposition 4.1). Then we check the first inequality in Proposition 4.2. If it is satisfied, we modify the instance so that (4.2) holds (Proposition 4.3). Subsequently, we transform the ρ_j into integer vectors (Proposition 4.6), construct a vector ρ of gozinto factors for an equivalent instance that fulfills (4.3) (Proposition 4.5), and finally check feasibility on the first machine (Proposition 4.4). A pseudocode for the overall procedure is provided in Figure 4.1. Here T denotes the length of the planning horizon, and ρ is defined as $\rho(t) = r(i)$ if $d(i,t) = 1$ for some i with $r(i) \geq 1$. Let $\rho(t) = 0$ for the remaining periods. Note that those demand points with $r(i) = 0$ do not give rise to any dependent demand, and as a consequence, they have no effect on the instance's feasibility as long as the first inequality in Proposition 4.2 is satisfied. Therefore, we do not store in ρ any information related to them.

{obtaining an instance with integer demands}
for all i **do**
  while true do
    if $d(i,t) \in \mathbb{Z}$ for all t **then**
      break
    else
      $t_1 \leftarrow \min\{t : d(i,t) \notin \mathbb{Z}\}$
      if $d(i,t) = 0$ for all $t > t_1$ **then**
        $d(i,t_1) \leftarrow \lceil d(i,t_1) \rceil$
        break
      else
        $t_2 \leftarrow \min\{t > t_1 : d(i,t) > 0\}$
        $s \leftarrow \lceil d(i,t_1) \rceil - d(i,t_1)$
        if $s < d(i,t_2)$ **then**
          $d(i,t_1) \leftarrow \lceil d(i,t_1) \rceil$
          $d(i,t_2) \leftarrow d(i,t_2) - s$
        else
          $d(i,t_1) \leftarrow d(i,t_1) + d(i,t_2)$
          $d(i,t_2) \leftarrow 0$
{checking the first inequality in Proposition 4.2}
if $\sum_i \sum_{t'=1}^{t} d(i,t') > t - 1$ for some t **then**
  return infeasible

Figure 4.1. A polynomial algorithm.

{modifying the instance so that (4.2) holds}
for all j **do**
  let $i_1, i_2, \ldots, i_{|I(j)|} \in I(j)$ be ordered such that $r(i_k) \geq r(i_{k+1})$ for all k
  for $t = T$ **to** 1 **do**
    for $k = 1$ **to** $|I(j)|$ **do**
      if $d(i_k, t) > 0$ **then**
        $d(i_k, t-1) \leftarrow d(i_k, t-1) + d(i_k, t) - 1, \quad d(i_k, t) \leftarrow 1$
        for $l = k$ **to** $|I(j)|$ **do**
          $d(i_l, t-1) \leftarrow d(i_l, t-1) + d(i_l, t), \quad d(i_l, t) \leftarrow 0$
{transforming the ρ_j into integer vectors}
for all j **do**
  while true do
    if $\rho_j(t) \leq 0$ or $\rho_j(t) \in \mathbb{Z}$ for all t **then**
      break
    else
      $t_1 \leftarrow \min\{t : \rho_j(t) > 0 \text{ and } \rho_j(t) \notin \mathbb{Z}\}$
      if $\rho_j(t) \leq 0$ for all $t > t_1$ **then**
        $\rho_j(t_1) \leftarrow \lceil \rho_j(t_1) \rceil$
        break
      else
        $t_2 \leftarrow \min\{t > t_1 : \rho_j(t) > 0\}$
        $s \leftarrow \lceil \rho_j(t_1) \rceil - \rho_j(t_1)$
        if $s < \rho_j(t_2)$ **then**
          $\rho_j(t_1) \leftarrow \lceil \rho_j(t_1) \rceil$
          $\rho_j(t_2) \leftarrow \rho_j(t_2) - s$
        else
          $\rho_j(t_1) \leftarrow \rho_j(t_1) + \rho_j(t_2)$
          $\rho_j(t_2) \leftarrow 0$

Figure 4.1. A polynomial algorithm (cont.).

```
{defining the vector ρ}
t ← T
for t' = T to 1 do
   t ← min{t, t'}
   for j = 1 to |J| do
      if ρ_j(t') > 0 then
         ρ(t) ← ρ_j(t')
         t ← t − 1
{checking feasibility on the first machine}
if Σ_{t'=1}^{t} ρ(t') ≤ t − 1 for all t then
   return feasible
else
   return infeasible
```

Figure 4.1. A polynomial algorithm (cont.).

For simplicity of notation, let I, J designate in this paragraph the number (rather than sets) of end and intermediate items. The complexity of obtaining an instance with integer demand quantities is $O(IT)$ since for each i there are at most T demand points. The same is true of checking the first inequality in Proposition 4.2, which can be performed by going over the $I \times T$ demand matrix (essentially) once. In order to modify the instance to satisfy (4.2), we first sort the indices i in a nonincreasing manner with respect to gozinto factors in $O(I \log I)$ time. Then we carry unit demands to neighboring periods one by one; this takes $O(T^2)$ steps in the worst case. The run-time order of defining the ρ_j is $O(IT)$ whereas transforming them into integer vectors is $O(T)$ as all demand points are visited once. It takes $O(JT)$ steps to construct ρ, and $O(T^2)$ steps to check feasibility on the first machine. Therefore, as $J \leq I \leq T$, the overall complexity of the algorithm is $O(T^2)$. Note that this applies for the most efficient implementation possible, which we do not present in Figure 4.1 for the sake of clarity and brevity.

Let us demonstrate the algorithm on an example. Suppose $J = \{a, b\}$, $I = \{a', a'', b'\}$, $j(a') = j(a'') = a$, and $j(b') = b$. So there are two intermediate items a, b, and three end items a', a'', b'. Let $r(a') = 3/2$, $r(a'') = 1/2$, and $r(b') = 2/3$. This means that in order to produce one unit of a', $3/2$ units of a are required; similarly for a'' and b'. We assume a planning horizon of $T = 10$ periods. Suppose $d(a', 4) = d(a', 7) = d(a'', 10) = d(b', 10) = 1$, $d(a'', 4) = 1.5$, $d(b', 8) = 2.8$, and $d(b', 9) = 0.2$. Demand sets formed by the algorithm, together with the original one, are depicted in Figure 4.2. Here a demand due at period t is represented by an arrow pointing the borderline between t and $t + 1$. The associated end item is written underneath. Double-headed arrows show conflicting demands.

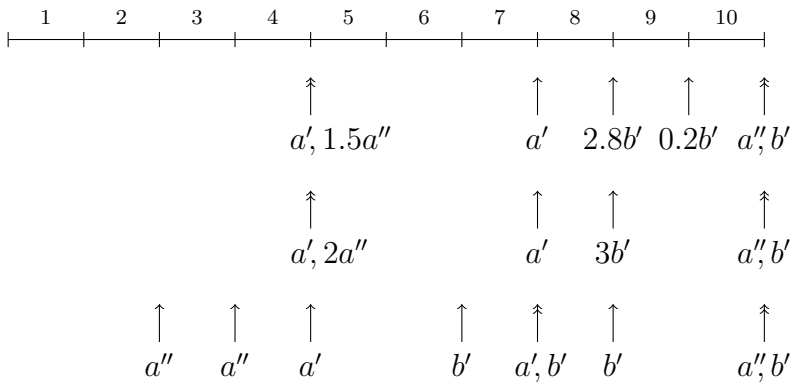

Figure 4.2. Original demand set (top), its integer equivalent (middle), and the one that satisfies (4.2) (bottom).

Table 4.2. Gozinto vectors formed by the algorithm.

t	1	2	3	4	5	6	7	8	9	10
ρ_a		$\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$			$\frac{3}{2}$			$\frac{1}{2}$
ρ_b						$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$		$\frac{2}{3}$
integer ρ_a		1	0	2			1			1
integer ρ_b						1	1	0		1
ρ		1		2	1	1	1		1	1

Consequent gozinto vectors ρ_a, ρ_b, their integer equivalents, and the vector ρ are given in Table 4.2. We omit the -1's in ρ_a, ρ_b, and the 0's in ρ for the sake of readability. The instance turns out to be feasible. Figure 4.3 shows the Gantt chart of a feasible schedule.

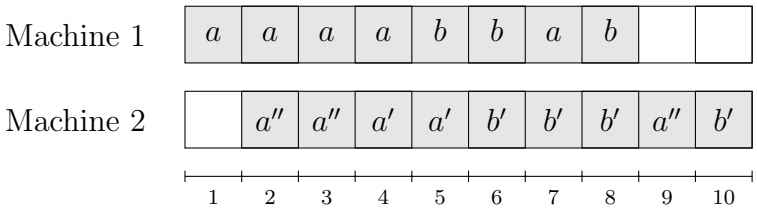

Figure 4.3. Gantt chart of a feasible schedule.

We close this section with a discussion of possible extensions. It is easy to modify the algorithm for a hybrid flow shop with a finite number of identical parallel machines in the first stage. However, if there is more than one machine in the second stage, Proposition 4.3 fails to hold, and the algorithm no longer works. Similarly, it cannot be applied for a small-bucket model without the all-or-nothing assumption because Proposition 4.1 is invalid then. Finally, we note that the algorithm can also be used to solve the polynomially equivalent optimization problem of minimizing maximum lateness.

4.4. Conclusion and Further Research

We examined the lot-sizing and scheduling counterpart of the strongly NP-hard pure scheduling problem $F2 \,||\, L_{\max}$, and showed that in this case feasibility can be determined by a polynomial algorithm of run-time order $O(T^2)$, where T is the length of the planning horizon. The algorithm applies to divergent bills-of-materials as well as to hybrid flow shops with a finite number of identical parallel machines in the first stage. An interesting future research topic would be to study the complexity of the same small-bucket problem without the all-or-nothing assumption.

5. CONCLUSION

In this thesis, we investigated three lot-sizing and scheduling problems encountered in flow shops. The first one is a parallel machine multi-item lot-sizing and scheduling problem with a secondary resource, in which demands are given for the entire planning horizon rather than for every single period. We proved that the problem is NP-hard, and presented two equivalent formulations as mixed-integer linear programs. The second formulation, using aggregate integer variables instead of individual binary variables for each machine, turned out to be more efficient. We showed some properties the optimal objective value z^* must satisfy, proposed optimality conditions, and gave a heuristic algorithm. In particular, z^* must be even so that the absolute MIP gap tolerance can be set to $2 - \varepsilon$. This redefinition slightly improves the solution performance of the aggregate formulation, which is also true of feeding the solver with an initial heuristic solution in general. Incorporation of the cut (2.4) leads to better results, making up the best implementation together with the heuristic. We discussed two possible extensions to the problem as well, namely vertical and minimum lot size constraints. We showed how these can be formulated, and mentioned some properties of the new problems arising thereby. Furthermore, we suggested two modifications of the heuristic. Computational tests indicate that it is significantly more time-consuming to solve these extended problems.

The second problem considers a two-machine flow shop where the bill-of-materials structure is serial and the objective is to first minimize back orders and then the number of changeovers. We proposed several optimality conditions and reformulations. There does not seem to be a significant difference in the performance of the one- and the two-phase methods. Back order minimization is solved much faster compared to changeover minimization. Surprisingly, the reformulation with z_{mit}, although it has more variables, turns out to be much better than the original formulation with z_{mt}. Our computational study, in general, supports the view that adding a cut to a formulation makes sense only if it strengthens the LP bound.

The third problem is the lot-sizing and scheduling counterpart of the strongly NP-hard pure scheduling problem $F2 \mid\mid L_{\max}$. We showed that feasibility can be checked by a polynomial algorithm of run-time order $O(T^2)$, where T is the length of the planning horizon. The algorithm applies to divergent bills-of-materials as well as to hybrid flow shops with a finite number of identical parallel machines in the first stage. An interesting future research topic would be to study the complexity of the same small-bucket problem without the all-or-nothing assumption.

REFERENCES

1. Teksan, Z. M., *Integrated Production Planning, Shift Planning and Detailed Scheduling in a Tissue Paper Manufacturer*, M.S. Thesis, Boğaziçi University, 2011.

2. Wagner, H. M. and T. M. Whitin, "Dynamic Version of the Economic Lot Size Model", *Management Science*, Vol. 5, No. 1, pp. 89–96, 1958.

3. Pochet, Y. and L. A. Wolsey, *Production Planning by Mixed Integer Programming*, Springer, 2006.

4. Pinedo, M. L., *Scheduling: Theory, Algorithms, and Systems*, Springer, 2012.

5. Copil, K., M. Wörbelauer, H. Meyr and H. Tempelmeier, "Simultaneous lotsizing and scheduling problems: a classification and review of models", *OR Spectrum*, Vol. 39, No. 1, pp. 1–64, 2017.

6. Suerie, C., *Time Continuity in Discrete Time Models*, Springer, 2005.

7. Fiorotto, D. J., R. Jans and S. A. de Araujo, "An analysis of formulations for the capacitated lot sizing problem with setup crossover", *Computers and Industrial Engineering*, Vol. 106, pp. 338–350, 2017.

8. Mahdieh, M., A. Clark and M. Bijari, "A novel flexible model for lot sizing and scheduling with non-triangular, period overlapping and carryover setups in different machine configurations", *Flexible Services and Manufacturing Journal*, pp. 1–40, 2017.

9. Seeanner, F. and H. Meyr, "Multi-stage simultaneous lot-sizing and scheduling for flow line production", *OR Spectrum*, Vol. 35, No. 1, pp. 33–73, 2013.

10. Stadtler, H. and F. Sahling, "A lot-sizing and scheduling model for multi-stage

flow lines with zero lead times", *European Journal of Operational Research*, Vol. 225, No. 3, pp. 404–419, 2013.

11. Shen, L., J. N. D. Gupta and U. Buscher, "Flow shop batching and scheduling with sequence-dependent setup times", *Journal of Scheduling*, Vol. 17, No. 4, pp. 353–370, 2014.

12. Jordan, C., *Batching and Scheduling*, Springer, 1996.

13. Sarin, S. C. and P. Jaiprakash, *Flow Shop Lot Streaming*, Springer, 2007.

14. Cheng, M., N. J. Mukherjee and S. C. Sarin, "A review of lot streaming", *International Journal of Production Research*, Vol. 51, No. 23-24, pp. 7023–7046, 2013.

15. Bruno, J. and P. Downey, "Complexity of Task Sequencing with Deadlines, Setup Times and Changeover Costs", *SIAM Journal on Computing*, Vol. 7, No. 4, pp. 393–405, 1978.

16. Cheng, T. C. E., C. T. Ng and J. J. Yuan, "The single machine batching problem with family setup times to minimize maximum lateness is strongly NP-hard", *Journal of Scheduling*, Vol. 6, No. 5, pp. 483–490, 2003.

17. Lenstra, J. K., A. H. G. Rinnooy Kan and P. Bruker, "Complexity of machine scheduling problems", *Annals of Discrete Mathematics*, Vol. 1, pp. 343–362, 1977.

18. Koulamas, C., "On the complexity of two-machine flowshop problems with due date related objectives", *European Journal of Operational Research*, Vol. 106, No. 98, pp. 95–100, 1998.

19. Lin, B. M. T., "Scheduling in the two-machine flowshop with due date constraints", *International Journal of Production Economics*, Vol. 70, No. 2, pp. 117–123, 2001.

20. Potts, C. N. and L. N. van Wassenhove, "Integrating Scheduling with Batching and

Lot-Sizing: a Review of Algorithms and Complexity", *Journal of the Operational Research Society*, Vol. 43, No. 5, pp. 395–406, 1992.

21. Floudas, C. A. and X. Lin, "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review", *Computers and Chemical Engineering*, Vol. 28, No. 11, pp. 2109–2129, 2004.

22. Quadt, D. and H. Kuhn, "Conceptual framework for lot-sizing and scheduling of flexible flow lines", *International Journal of Production Research*, Vol. 43, No. 11, pp. 2291–2308, 2005.

23. Quadt, D., *Lot-Sizing and Scheduling for Flexible Flow Lines*, Springer, 2004.

24. Stefansson, H., S. Sigmarsdottir, P. Jensson and N. Shah, "Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry", *European Journal of Operational Research*, Vol. 215, No. 2, pp. 383–392, 2011.

25. Camargo, V. C. B., F. M. B. Toledo and B. Almada-Lobo, "Three time-based scale formulations for the two-stage lot sizing and scheduling in process industries", *Journal of the Operational Research Society*, Vol. 63, No. 11, pp. 1613–1630, 2012.

26. Seeanner, F., B. Almada-Lobo and H. Meyr, "Combining the principles of variable neighborhood decomposition search and the fix&optimize heuristic to solve multi-level lot-sizing and scheduling problems", *Computers and Operations Research*, Vol. 40, No. 1, pp. 303–317, 2013.

27. Monma, C. L. and C. N. Potts, "On the Complexity of Scheduling with Batch Setup Times", *Operations Research*, Vol. 37, No. 5, pp. 798–804, 1989.

28. Unal, A. T. and A. S. Kiran, "Batch Sequencing", *IIE Transactions*, Vol. 24, No. 4, pp. 73–83, 1992.

29. Kimms, A., *Multi-Level Lot Sizing and Scheduling*, Physica-Verlag, 1997.

30. Seeanner, F., *Multi-Stage Simultaneous Lot-Sizing and Scheduling*, Springer Gabler, 2013.

31. Drexl, A. and A. Kimms, "Lot sizing and scheduling—Survey and extensions", *European Journal of Operational Research*, Vol. 99, No. 2, pp. 221–235, 1997.

32. Karimi, B., S. M. T. Fatemi Ghomi and J. M. Wilson, "The capacitated lot sizing problem: a review of models and algorithms", *Omega*, Vol. 31, No. 5, pp. 365–378, 2003.

33. Zhu, X. and W. E. Wilhelm, "Scheduling and lot sizing with sequence-dependent setup: A literature review", *IIE Transactions*, Vol. 38, No. 11, pp. 987–1007, 2006.

34. Quadt, D. and H. Kuhn, "A taxonomy of flexible flow line scheduling procedures", *European Journal of Operational Research*, Vol. 178, No. 3, pp. 686–698, 2007.

35. Jans, R. and Z. Degraeve, "Modeling industrial lot sizing problems: a review", *International Journal of Production Research*, Vol. 46, No. 6, pp. 1619–1643, 2008.

36. Buschkühl, L., F. Sahling, S. Helber and H. Tempelmeier, "Dynamic capacitated lot-sizing problems: a classification and review of solution approaches", *OR Spectrum*, Vol. 32, No. 2, pp. 231–261, 2010.

37. Lasdon, L. S. and R. C. Terjung, "An Efficient Algorithm for Multi-Item Scheduling", *Operations Research*, Vol. 19, No. 4, pp. 946–969, 1971.

38. Eppen, G. D. and R. K. Martin, "Solving multi-item capacitated lot-sizing problems using variable redefinition", *Operations Research*, Vol. 35, No. 6, pp. 832–848, 1987.

39. Vanderbeck, F. and L. A. Wolsey, "Valid Inequalities for the Lasdon-Terjung Production Model", *The Journal of the Operational Research Society*, Vol. 43, No. 5,

pp. 435–441, 1992.

40. Gicquel, C., M. Minoux and Y. Dallery, "Exact solution approaches for the discrete lot-sizing and scheduling problem with parallel resources", *International Journal of Production Research*, Vol. 49, No. 9, pp. 2587–2603, 2011.

41. Gicquel, C., L. A. Wolsey and M. Minoux, "On discrete lot-sizing and scheduling on identical parallel machines", *Optimization Letters*, Vol. 6, No. 3, pp. 545–557, 2012.

42. van Eijl, C. A. and C. P. M. van Hoesel, "On the discrete lot-sizing and scheduling problem with Wagner-Whitin costs", *Operations Research Letters*, Vol. 20, No. 1, pp. 7–13, 1997.

43. Jans, R. and Z. Degraeve, "An industrial extension of the discrete lot-sizing and scheduling problem", *IIE Transactions*, Vol. 36, No. 1, pp. 47–58, 2004.

44. Quadt, D. and H. Kuhn, "Capacitated Lot-Sizing and Scheduling with Parallel Machines, Back-Orders, and Setup Carry-Over", *Naval Research Logistics*, Vol. 56, No. 4, pp. 366–384, 2009.

45. Kaczmarczyk, W., "Proportional lot-sizing and scheduling problem with identical parallel machines", *International Journal of Production Research*, Vol. 49, No. 9, pp. 2605–2623, 2011.

46. Fiorotto, D. J. and S. A. de Araujo, "Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines", *Annals of Operations Research*, Vol. 217, No. 1, pp. 213–231, 2014.

47. Sherali, H. D. and J. C. Smith, "Improving Discrete Model Representations via Symmetry Considerations", *Management Science*, Vol. 47, No. 10, pp. 1396–1407, 2001.

48. Jans, R., "Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints", *INFORMS Journal on Computing*, Vol. 21, No. 1, pp. 123–136, 2009.

49. Vielma, J. P., "Mixed Integer Linear Programming Formulation Techniques", *SIAM Review*, Vol. 57, No. 1, pp. 3–57, 2015.

50. Florian, M. and M. Klein, "Deterministic Production Planning with Concave Costs and Capacity Constraints", *Management Science*, Vol. 18, No. 1, pp. 12–20, 1971.

51. van Hoesel, C. P. M. and A. P. M. Wagelmans, "An O(T^3) Algorithm for the Economic Lot-sizing Problem with Constant Capacities", *Management Science*, Vol. 42, No. 1, pp. 142–150, 1996.

52. Van Vyve, M., "Algorithms for Single-Item Lot-Sizing Problems with Constant Batch Size", *Mathematics of Operations Research*, Vol. 32, No. 3, pp. 594–613, 2007.

53. Florian, M., J. K. Lenstra and A. H. G. Rinnooy Kan, "Deterministic Production Planning: Algorithms and Complexity", *Management Science*, Vol. 26, No. 7, pp. 669–679, 1980.

54. Bitran, G. R. and H. H. Yanasse, "Computational Complexity of the Capacitated Lot Size Problem", *Management Science*, Vol. 28, No. 10, pp. 1174–1186, 1982.

55. Zangwill, W. I., "A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System—A Network Approach", *Management Science*, Vol. 15, No. 9, pp. 506–527, 1969.

56. Melo, R. A. and L. A. Wolsey, "Uncapacitated two-level lot-sizing", *Operations Research Letters*, Vol. 38, No. 4, pp. 241–245, 2010.

57. van Hoesel, S., H. E. Romeijn, D. R. Morales and A. P. M. Wagelmans, "Integrated

Lot Sizing in Serial Supply Chains with Production Capacities", *Management Science*, Vol. 51, No. 11, pp. 1706–1719, 2005.

58. Hwang, H. C., H. S. Ahn and P. Kaminsky, "Basis Paths and a Polynomial Algorithm for the Multistage Production-Capacitated Lot-Sizing Problem", *Operations Research*, Vol. 61, No. 2, pp. 469–482, 2013.

59. Hwang, H. C., H. S. Ahn and P. Kaminsky, "Algorithms for the two-stage production-capacitated lot-sizing problem", *Journal of Global Optimization*, Vol. 65, No. 4, pp. 777–799, 2016.

60. Ahmed, S., Q. He, S. Li and G. L. Nemhauser, "On the Computational Complexity of Minimum-Concave-Cost Flow in a Two-Dimensional Grid", *SIAM Journal of Optimization*, Vol. 26, No. 4, pp. 2059–2079, 2016.

61. Goisque, G. and C. Rapine, "An efficient algorithm for the 2-level capacitated lot-sizing problem with identical capacities at both levels", *European Journal of Operational Research*, Vol. 261, No. 3, pp. 918–928, 2017.

62. Brüggemann, W. and H. Jahnke, "Discrete lot-sizing and scheduling problem: complexity and modification for batch availability", *European Journal of Operational Research*, Vol. 124, No. 3, pp. 511–528, 2000.

63. Brüggemann, W. and H. Jahnke, "Erratum to 'Discrete lot-sizing and scheduling problem: complexity and modification for batch availability'", *European Journal of Operational Research*, Vol. 126, No. 3, p. 688, 2000.