636 LECTURE NOTES IN ECONOMICS AND MATHEMATICAL SYSTEMS

J. Christian Lang

Production and Inventory Management with Substitutions

Springer

Lecture Notes in Economics and Mathematical Systems 636

J. Christian Lang

Production and Inventory Management with Substitutions

Springer

Dr. J. Christian Lang
Technische Universität Darmstadt
Department of Law, Business and Economics
Chair of Operations Research
Hochschulstr. 1
64289 Darmstadt
Germany
j.christian.lang@gmail.com

ISSN 0075-8442
ISBN 978-3-642-04246-1 e-ISBN 978-3-642-04247-8
DOI 10.1007/978-3-642-04247-8
Springer Heidelberg Dordrecht London New York

zgl. Dissertation, Technische Universität Darmstadt, 2009, D17, submitted under the title "Production and Inventory Management with Flexible Bills-of-Materials and Substitutions"

Library of Congress Control Number: 2009934481

Cover design: SPi Publisher Services

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

Quantitative approaches for solving production planning and inventory management problems in industry have gained growing importance in the past years. Due to the increasing use of Advanced Planning Systems, a widespread practical application of the sophisticated optimization models and algorithms developed by the Production Management and Operations Research community now seem within reach.

The possibility that products can be replaced by certain substitute products exists in various application areas of production planning and inventory management. Substitutions can be useful for a number of reasons, among others to circumvent production and supply bottlenecks and disruptions, increase the service level, reduce setup costs and times, and lower inventories and thereby decrease capital lockup. Considering the current trend in industry towards shorter product life cycles and greater product variety, the importance of substitutions appears likely to grow. Closely related to substitutions are flexible bills-of-materials and recipes in multi-level production systems.

However, so far, the aspect of substitutions has not attracted much attention in academic literature. Existing lot-sizing models matching complex requirements of industrial optimization problems (e.g., constrained capacities, sequence-dependent setups, multiple resources) such as the Capacitated Lot-Sizing Problem with Sequence-Dependent Setups (CLSD) and the General Lot-Sizing and Scheduling Problem for Multiple Production Stages (GLSPMS) do not feature in substitution options.

This was the point where J. Christian Lang initiated his PhD project. In his project, he devised a graphical modeling framework for substitution options. Using this framework, he developed the following extensions of existing dynamic lot-sizing models to include product substitutions:

1. Uncapacitated and capacitated single-level dynamic lot-sizing models with substitutions
2. A capacitated single-level dynamic lot-sizing model with substitutions and sequence-dependent setups
3. An extension of the Multi-Level Capacitated Lot-Sizing Problem (MLCLSP) by flexible bills-of-materials

4. An extension of the General Lot-Sizing and Scheduling Problem for Multiple Production Stages (GLSPMS) to map substitutions and flexible production sequences

In addition, he developed exact and heuristic solution approaches for model categories 1 and 2 to solve problems with acceptable computational efforts, and analyzed these approaches in extensive computational experiments.

He also came up with a number of very interesting and illustrative examples of application areas where substitutions are of importance. He delved into one of these areas, blood transfusion inventory management, and developed a simulation-based optimization model for this complex stochastic optimization problem, demonstrating the potential of automated optimization approaches also for *inventory management* with substitutions.

J. Christian Lang's PhD thesis is a very innovative piece of research. It contains a large number of new aspects and inventive ideas: It contributes a comprehensive classification scheme, extensive literature review, and a novel modeling approach for lot-sizing with substitutions. The new models and solution methods that his thesis provides have promising potential for successful use in practice. Hence, his work represents a significant scientific advance and will serve as a valuable resource for researchers and practitioners.

Darmstadt Wolfgang Domschke
July 2009

Preface

This thesis is the result of three years of research conducted at the chair of Operations Research of the Department of Law, Business and Economics at Technische Universität Darmstadt, Germany. It would not have been possible without the backing of numerous people.

First of all, I deeply thank my advisor Wolfgang Domschke for his continuous support and mentorship. He has always granted my colleagues and me an extraordinarily high degree of freedom in pursuing our research interests. I learned a lot from his broad expertise, way of thinking and highly precise style of writing, and received helpful and constructive advice and feedback from him.

I am very grateful to Herbert Meyr for serving as co-advisor and the second referee of my thesis. In addition, I would like to thank him for the inspiring suggestions he made while I was working on my PhD project and his highly constructive and detailed feedback that helped to improve the content and presentation of my dissertation.

I thank Alvaro da Costa for coincidentally inspiring the idea for my PhD research. Also, I would like to thank Robert Klein and Uwe Probst for sparking my interest in research.

Furthermore, I thank my colleagues Gabriela Mayer, Anita Petrick, and Daniel Scholz at the chair of Operations Research, our teaching and research assistants, as well as all colleagues on our floor for creating a warm, familial atmosphere and enjoyable research environment. Especially, I would like to thank Florian Seeanner, Daniel Scholz, and Thomas Widjaja for fruitful discussions and feedback. I am particularly grateful for the ideas, contributions and feedback of Ole-Björn Baasch, Anna-Lena Beutel, Saskia Kauder, Baptiste Lebreton, Raissa Sachs, Florian Sahling, Björn-Ragnar Weber, and Wolfram Wiesemann. In addition, I would like to thank the entire QBWL research community for providing plenty of challenging and inspiring feedback.

I am further indebted to Kai Sachs and the other members of the TUD-Aktiv party – our joint dedication to academic politics was a welcome change in day-to-day research and indeed a valuable experience.

Yet there is a life besides academia. I am thankful to my parents for providing the educational background that enabled me to write this dissertation. Also, I am deeply grateful to my brother Benjamin for the many animating discussions and

joyful times. Finally, I would like to thank my parents, my brother, and my friends for enriching my life throughout times of joy and pain during this project.

Thank you all.

Darmstadt Jan Christian Lang
July 2009

Contents

List of Figures

List of Tables

Acronyms

LSP-SI	Lot-sizing problem with substitution and initial inventory
BOM	Bill-of-materials
WWP	Wagner–Whitin problem
CLSP	Capacitated lot-sizing problem
CLSD	Capacitated lot-sizing problem with sequence-dependent setups
GLSP	General lot-sizing and scheduling problem
MLCLSP	Multi-level capacitated lot-sizing problem
GLSPMS	General lot-sizing and scheduling problem for multiple production stages
CLSP-S	Capacitated lot-sizing problem with substitution
CLSD-S	Capacitated lot-sizing problem with sequence-dependent setups and substitutions
RPS	Requirements planning problem with substitutions
MR-CLSP-S	Multi-resource capacitated lot-sizing problem with substitution
MILP	Mixed-integer linear programming
MIP	Mixed-integer programming
APS	Advanced planning system
SCP	Supply chain planning
ATP	Available-to-promise
CTP	Capable-to-promise
MTS	Make-to-stock
MTO	Make-to-order
STB	Small time bucket
LTB	Large time bucket
SP	Stochastic programming
RO	Robust optimization
DCF	Discounted cash flow
CLSPL	Capacitated lot-sizing problem with linked lot-sizes
SBO	Simulation-based optimization

NPV	Net present value
GPS	Generalized pattern search
RSM	Response surface methodology
GA	Genetic algorithm
PS	Pattern search
DS	Direct search
TS	Tabu search
MCNFP	Minimum cost network flow problem
CPU	Central processing unit
GHz	Gigahertz
CWH	Car wiring harness
SOA	Service-oriented architecture
STN	State-task network
RTN	Resource-task network
SPL	Simple plant location
ECU	Electronic control unit
WLP	Warehouse location problem
ISM	Integrated steel manufacturer
SWCP	Substitution with conversion problem
MLFP	Multi-level lot-sizing problem with flexible production sequences
EOQ	Economic-order-quantity
SWOP	Substitution without conversion problem
ZIP	Zero inventory-production
HPL	Homogeneous product lots
MRU	Most recent usage
IU	Immediate usage
MLCLSP-S	Multi-level capacitated lot-sizing problem with substitutions
GLSPMS-FPS	General lot-sizing and scheduling problem for multiple production stages with flexible production sequences
VMI	Vendor managed inventory
RBC	Regional blood center
PPM	Production process model
SNO	(Oracle®) Strategic Network Optimization
APO	(SAP®) Advanced Planner and Optimizer
ATO	Assemble-to-order
CRN	Common random numbers

Chapter 1
Introduction

Striving for operational and strategic excellence, companies have continuously been trying to improve their business processes in the past years. As sophisticated planning processes are often seen as a key enabler for efficient business processes, this has led to an increased focus on developing methods to optimize processes using mathematical methods in cases where decision problems are too complex to be solved by a human decision maker. In order to transfer such methods into practice, IT-based systems such as *Advanced Planning Systems (APS)* that make use of these optimization methods to support and automate planning processes and execute the resulting plans are essential.

In companies handling physical goods, the planning processes for production and logistics often involve a high amount of complexity. This complexity results on the one hand from the fact that companies frequently have multiple locations in various countries and offer a large product variety associated with high manufacturing complexity. On the other hand, companies are increasingly embedded in complex global supply networks with a large number of actual or potential suppliers and customers. From the viewpoint of an individual company, its suppliers and customers and the actors further upstream or downstream are referred to by the term Supply Chain (SC). Thus, each company may be part of multiple "subjective" supply chains of other companies (Bretzke, 2006).

The notion of *Supply Chain Management (SCM)* promises to tackle the decision problems in today's complex environment by a holistic approach for designing and steering transportation, inventory and production processes in a network with suppliers, locations and customers. The ideal goal in this approach would be an optimization of the overall system – i.e., a company, its suppliers and customers – e.g., with respect to profit or service level goals, in contrast to the isolated optimization of subsystems frequently leading to suboptimal solutions for the entire system. However, the term SCM is often used in a less revolutionary way, referring to the design and control of the supply network of a multi-division multi-location company with actual and potential suppliers and customers, where supplier and customer decision processes are not included explicitly. Including these raises profit distribution questions and, from a scientific view, often leads to game-theoretic approaches. For literature on these aspects see, e.g., Bretzke (2006).

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_1,

The concept of *flexibility* in logistics and production has recently attracted increasing attention. Various types of flexibility can be distinguished (Naim et al., 2006). Flexibility in this context refers to certain instruments by which a company can perform logistics and production processes in a slightly different way than usual. Exploiting flexibility instruments can help the decision maker to increase system performance with regard to a chosen goal.

Substitution of products by compatible products is one such flexibility instrument. E.g., a manufacturer that produces parts for ship and locomotive motors from steel cylinders (unfinished parts) using a milling machine can use a longer steel cylinder as a substitute if the required shorter length is not in stock and reduce the length by a conversion step. Similarly, when handling a large variant number of electronic control units (ECU) that cover different sets of features (e.g., in the automotive sector), an ECU variant can often be substituted by another variant that covers a superset of its features. The products among which substitution takes place could be different products manufactured by the company itself as well as products sourced from one or more suppliers.

In multi-level production settings, substitutions correspond with *flexible bills-of-materials (BOMs)* for discrete goods and with alternative recipes for continuous goods, as the usage of alternative BOMs / recipes equals substitutions of input or intermediate goods. For example, if a component of an electronic assembly can be substituted by another one, the BOM associated with that assembly is flexible.

Substitutions are particularly useful in cases where capacitated production resources are tight or the supply side and thus the inventory of goods is scarce. Classical deterministic production planning optimization models do not include product substitutions. The same holds true for the most stochastic inventory control models. However, neglecting substitution options might result in potentials for increased efficiency that remain unused.
Five possible benefits of substitutions can be distinguished:

- *Increased service level*: Stock-outs due to supply or production bottlenecks can be avoided by performing substitutions.
- *Reduction of holding costs*: As substitutions lead to a "risk pooling" effect between products in a stochastic setting, they might reduce the required level of safety stocks.
- *Reduction of setup costs and times*: By producing larger lot sizes of a smaller number of products (that can substitute for other products not produced), the total setup costs and times can be reduced.
- *Exploitation of unit cost variations*: If a substitute product becomes cheaper, one can "switch" to this substitute and use it to replace the previously used product.
- *Reduction of wastage*: If the considered products are perishable, substitutions can be used to reduce the amount of outdated inventory, e.g., by consuming substitute stocks first if they have an earlier expiry date.

1.1 Definitions

Product substitution means that demand for a certain quantity of a product is fulfilled using another product. Usually, there is a *preferred* product for satisfying a specific demand, which can optionally be *substituted* by certain alternative products (*substitutes*).

We distinguish product substitution from *resource substitution*, where resources – e.g., employees or machines – can be substituted by certain others regarding a specific activity to be performed.

Customer-driven substitution means that the customer substitutes one product by another, whereas *firm-driven substitution* means that the supplier decides on using a substitute (Hale et al., 2000).

These categorizations are shown in Fig. 1.1, which depicts the different substitution types together with two flexibility instruments, namely transshipments and emergency orders. The latter term refers to orders for express deliveries from a location at an upper echelon to a lower echelon.

Substitution between products may either take place by directly replacing product B by product A or by performing a *conversion* activity on A so that it can replace B. If product A can substitute product B, we say that there is a *substitution option* from A to B. Product A is then called a *substitute* for product B.

Note that in the following, we use the term substitution referring both to substitution in single-level production structures and to flexible BOMs in multi-level production structures. We do so because substitution in single-level production structures can always be seen as a special case of flexible BOMs.

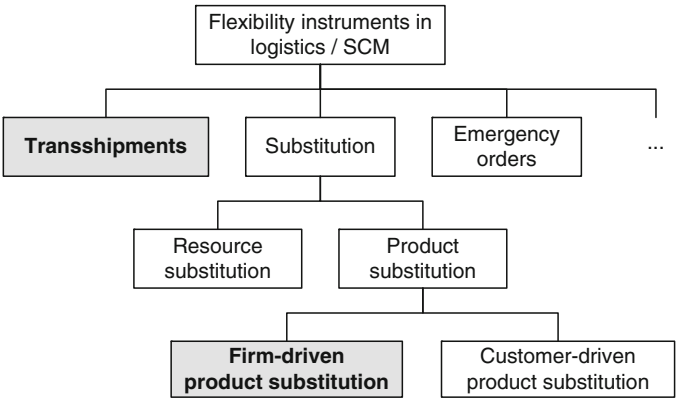


Fig. 1.1 Substitution and other flexibility instruments – classification

1.2 Motivation

Existing production planning and inventory control optimization models, integrated in APS and other IT-based planning systems, already represent a significant improvement to manual planning processes that have been in place in companies previously. However, though product substitution options exist in various real-world production planning and inventory management problems and promise raising efficiency, it seems that software solutions taking product substitution into account in optimization models are rarely used in industry and, concerning standard software, in the stage of being developed.

Only a very small number of publications on dynamic lot-sizing models *with substitutions* exists. The existing models and the corresponding solution approaches lack certain assumptions required to make them realistic:

- They assume that there are no initial inventories at the beginning of the planning horizon. Making this assumption in lot-sizing models without substitutions did not cause any loss of generality as one could simply transform gross into net demands. However, as we will illustrate later, it does cause a loss of generality if substitutions are possible.
- Production resources are assumed to be uncapacitated in all models known to the author, except for the model considered by Begnaud et al. (2006). However, production bottlenecks often exist in real-world production planning.
- Setup costs and times are assumed to be sequence-independent in all models, though the costs and duration of setup activities often depend on the sequence of products in practical problems.
- To the best of our knowledge, the model of Begnaud et al. (2006) is the only generic multi-stage lot-sizing and scheduling model *with flexible BOMs*. However, it cannot map exogenous resource downtime and assumes sequence-independent setups.

In the area of stochastic inventory control *with substitution*, a considerable amount of literature has been published, but often with very restrictive model assumptions that make it difficult to apply the solution approaches in practice:

- Most models assume a single period, there are only few models for inventory control with substitution assuming an arbitrary number of periods.
- Also, most models are intended for a special case of substitution options: They assume a downward substitution structure, where products can be ordered according to their quality and a product can substitute all other products with lower quality.
- Several models describe a decision problem involving only two products.

Two additional motives for this work are that the terminology used in literature with respect to product substitution is not sufficiently standardized and a classification of existing models by the characteristics of substitution options is missing. Hence, we intend to standardize the terminology regarding substitution and flexible BOMs and develop a classification scheme to categorize the existing models.

Regarding stochastic inventory control with substitutions, the *combination* with *lateral stock transshipments* in multi-location inventory systems has rarely been considered in literature, though this combination occurs in several real-world optimization problems. Transshipments are a flexibility instrument just as substitutions: This term refers to movements of stocks between locations on the same echelon. A typical example for companies that perform transshipments are shoe retailers: If a pair of shoes of a certain type and size that a customer wants to buy is out of stock at one outlet, the retailer transships the pair from a nearly outlet where it is still in stock.

1.3 Goals

The goal of this thesis is to contribute to the integration of substitutions and flexible BOMs into dynamic lot-sizing and inventory control models: We aim at integrating product substitution in some of these models and stimulating further research in this area by providing a unifying conceptual framework and classification. In addition, we intend to develop efficient solution approaches for solving the decision problems mapped in these models. This is done by using and adapting available methodology from the research fields of mixed-integer programming and simulation-based optimization.

Concerning dynamic lot-sizing models, we aim at integrating product substitution in well-known capacitated lot-sizing models. This is done by successively considering more complicated models, starting off from a simple uncapacitated model. The situation assumed in one of the capacitated models of this work is as follows: A company produces various goods in larger quantities on several production lines (e.g., machines). Each product is assigned to exactly one production line. It is assumed that the sequence of activities on a certain production line is the same for all products (flow shop). Thus, each production line can be modeled as a single capacitated production resource. As these resources are limited and running almost at full capacity, each of them poses a possible production bottleneck.

In order to start producing a certain product on a resource, setup activities (e.g., cleaning a machine) have to be performed. These incur *setup costs* and require a certain *setup time*. We will also consider the case where the setup costs and times for a product depend on the product that was previously produced on the resource (so-called *sequence-dependent setups*).

The production type is *make-to-stock*, i.e., the production planning is mainly performed based on demand forecasts, not primarily based on customer orders. The production takes place in larger lots to reduce the total setup costs and times, and thus to have more capacity available for production. Hence, produced goods often have to be kept in stock because parts of the produced goods are not immediately sold and/or shipped to customers, but stocked for demands at future points in time. This incurs *holding costs*, which represent capital lockup, storage costs and other aspects. Producing goods incurs variable *production costs*, among others the costs

of input products. If substitutions are performed, these may cause *substitution (conversion) costs*, e.g., the costs of labor time of manual activities necessary to use one product as a substitute for another.

The decision problem is now composed of the following questions:

- Which products should be set up at which points in time in which quantities, while observing the constrained production capacities?
- How should the produced units be used to satisfy demands, i.e., should substitutions be performed and if yes, how many units of which product should be used to replace another product at which point in time?

This problem is a combined lot-sizing and scheduling problem, as its solution determines both the lot-sizes and the production sequences on the resources.

With respect to stochastic inventory control, we focus on multi-location blood bank inventory management as one specific area of application where both substitutions and transshipments are relevant. Here, blood transfusion units are produced and stored at multiple locations and can be transshipped between these. If transfusion units of a patient's ABO/Rhesus blood type are not available in sufficient quantity or out of stock at a hospital, transfusion units with a compatible blood type can be used as a substitute. In practice, such blood bank inventory systems are often run with manual inventory control performed by human decision makers.

The idea is now to develop an inventory policy for the system that can be configured by setting various parameters, such as target inventory levels and critical levels. Such a policy could be implemented in a real-world setting using an IT solution. Our goal is to develop a simulation-based optimization algorithm that automatically improves the parameters of the policy by evaluating various configurations by means of discrete-event simulation. We intend to devise this solution approach sufficiently generic so that it can be adapted as well to other applications.

1.4 Outline

2 starts with an explanation of how the models considered in this work fit into the conceptual framework of the Supply Chain Planning matrix and how they could be integrated into Advanced Planning Systems. After this, we develop a classification of dynamic lot-sizing and scheduling problems by aggregating several existing classifications and adding further criteria for substitutions, flexible bills-of-materials and flexible production sequences, intending to make it as comprehensive as possible. This classification scheme is used to categorize several known dynamic lot-sizing and scheduling problems, which we explain in detail as they are the basis of the models with substitution options described in Chaps. 4–7. Subsequently, we give a compact overview of available methods for solving these deterministic dynamic lot-sizing problems by means of algorithms for mixed-integer linear programming (MILP).

As Chap. 8 deals with a stochastic transshipment problem that includes substitutions, we explain transshipment problems and present a classification scheme for transshipment models. We review selected existing approaches for such models, and afterwards focus on the methodology of simulation-based optimization used in Chap. 8.

The purpose of Chap. 3 is to explain in a clear and illustrative way how product substitution and flexible BOMs can be modeled. First, we introduce basic terms regarding product substitution. After this, a number of real-world examples for product substitution are given, showing the evident practical relevance of the topic. We then present four modeling approaches, namely blending models, substitution graphs, substitution hypergraphs, and task-oriented modeling and compare them. After developing classification criteria for substitution models complementary to the classifications in Chap. 2, we finally focus on important aspects to be considered when performing product substitutions in practice: We examine conditions where substitution can be beneficial, describe requirements for organizationally implementing substitutions, and potential pitfalls that should be kept in mind.

Chapter 4 reviews the literature on production planning and inventory control with substitution and related fields of research, with a focus on dynamic lot-sizing problems. We briefly review the literature on assortment problems, which can be seen as a tactical counterpart to dynamic lot-sizing problems with substitutions. Three existing dynamic lot-sizing models – two with substitutions, one with flexible BOMs and production sequences – are described and classified using the criteria developed in the previous chapters. Also, we explain how two of these models can be transformed into the more general third model. In the remainder of the chapter, we shortly discuss related topics, amongst others stochastic inventory control with substitutions and flexible bills-of-materials, and point out their relations to the topic of this work.

In Chap. 5, we develop two single-level lot-sizing models with substitutions. One of them is uncapacitated, the other one assumes capacitated production resources. Both of them incorporate initial inventories, allow for general substitution structures, and use the concept of demand classes. We formulate the problems as mixed-integer linear programs and develop Simple Plant Location-based reformulations as well as new valid inequalities. Also, we develop approximate extended formulations that only contain a subset of the disaggregated constraints. These various formulations are compared in extensive computational experiments on generated problem instances.

Chapter 6 deals with a single-level capacitated lot-sizing problem that assumes substitution options as well as sequence-dependent setup costs and times. The model is motivated from a real-world application described in Chap. 3. First, we formulate the model as an MILP model. Subsequently, we devise MIP-based heuristics by adapting the principles of Relax&Fix as well as Fix&Optimize heuristics to the model. Solution quality and running times of different variants of the developed heuristics are compared using generated problem instances.

In Chap. 7, we develop two lot-sizing models for multi-level production with flexible BOMs. One of them is an extension of the multi-level capacitated

lot-sizing problem and uses hypergraphs to model substitutions. It is appropriate for applications with discrete products and assembly structures, but cannot map flexible production sequences. We show that it can be transformed into one of the models in Chap. 4, and examine the ambiguity of echelon stocks arising in the model.

The second model, which is an extension of the general lot-sizing and scheduling problem for multiple production stages, uses state-task networks to model flexible BOMs. It is more general than the former model, as it can also map by-products (also: co-products) and flexible production sequences.

Chapter 8 considers a stochastic inventory model that incorporates both transshipments and substitutions. It uses a simulation-based optimization approach for determining parameters of a critical-level inventory policy. The approach is illustrated with the practical application in multi-location blood bank inventory management mentioned in Sect. 1.3. We model the blood bank system by discrete-event simulation and develop a novel pattern search SBO algorithm for improving parameters of the inventory policy. Using this algorithm, we improve inventory policies with and without transshipments and/or substitutions and compare the quality of the resulting solutions.

Chapter 9 completes this work by summarizing its results and highlighting potential avenues for future research.

Chapter 2
Production and Operations Management: Models and Algorithms

This chapter intends to give an overview of the literature on dynamic lot-sizing models and stochastic transshipment models. These two types of models are used as a basis for developing models with substitution in the following chapters. Section 2.1 contains a classification of models for dynamic lot-sizing / production planning, and selected models. In Sect. 2.2, we give a brief overview of available methods for solving deterministic dynamic lot-sizing problems modeled using mixed-integer linear programming (MILP). Section 2.3 introduces transshipment problems and presents a classification scheme for transshipment models. Section 2.4 reviews selected solution approaches that can be applied to stochastic inventory control models such as transshipment problems.

Dynamic lot-sizing models and transshipment models are linked to certain planning tasks in an *Advanced Planning System (APS)*: In the conceptual framework of *Advanced Planning* and the *Supply Chain Planning (SCP) matrix* (Fleischmann et al., 2005, p. 87) that is shown in Fig. 2.1, the combined lot-sizing and scheduling models considered in this work cover the planning tasks *lot-sizing* and *machine scheduling*. In addition, they are linked to the topic *short-term sales planning*, as a *Capable-To-Promise (CTP)* logic (Fleischmann et al., 2005, p. 91, Kilger and Schneeweiß, 2005, p. 185) could make use of the models to check whether customer orders could be fulfilled. Also, some lot-sizing models include *supplier selection*, *capacity planning* and other mid-term/tactical planning tasks in addition to short-term planning tasks. Transshipment models are used to optimize short-term planning tasks related to *warehouse replenishment*, *transport planning*, and *short-term sales planning*: Transshipments are executed to fulfill customer demands in case of local stock-outs. These replenishments from warehouses on the same echelon have to be implemented using available transportation capacities.

Thus, in the software architecture of an APS (a generic, idealized architecture is shown in Fig. 2.2), the lot-sizing models considered in this work will most likely be used for optimization in a *Production Planning* and/or *Scheduling* software module. Transshipment models would be used for the optimization of operational decision-making in the *Transport Planning* and *Demand Fulfilment & Available-To-Promise (ATP)* software modules. For details on the functionalities and architectures of APS, the reader is referred to Meyr et al. (2005a,b).

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_2, © Springer-Verlag Berlin Heidelberg 2010

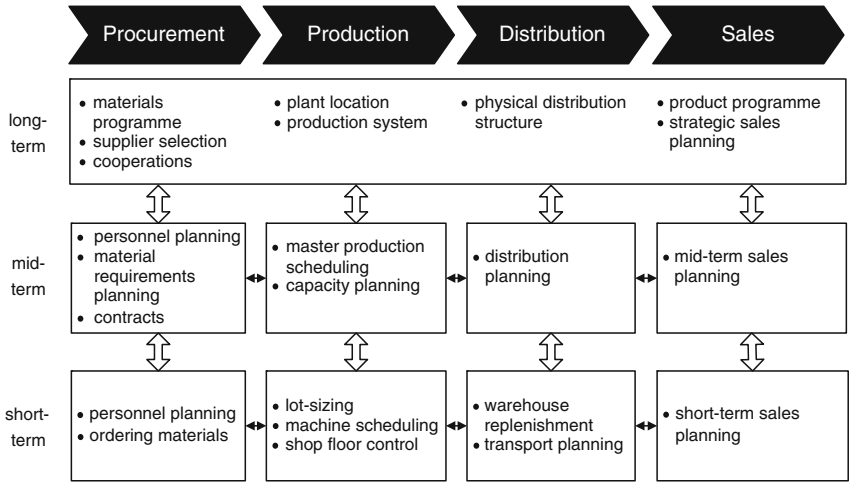


Fig. 2.1 Supply Chain Planning (SCP) matrix (Fleischmann et al., 2005, p. 87)

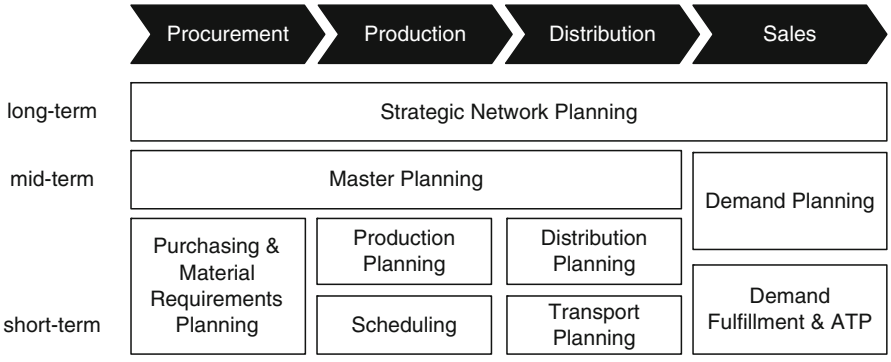


Fig. 2.2 Idealized APS software module architecture covering the SCP matrix (Meyr et al., 2005b, p. 109)

2.1 Dynamic Lot-Sizing

A vast amount of literature on production planning problems, especially on dynamic lot-sizing problems and simultaneous lot-sizing and scheduling problems, has been published.[1] This is due to the ubiquity of lot-sizing and scheduling decision problems in manufacturing firms and the large variety of production types in these firms, which often require specialized models. Most of the publications use MILP for

[1] We subsume dynamic lot-sizing problems as well as simultaneous lot-sizing and scheduling problems under the term "production planning problem". Note that in literature, "(aggregate) production planning" may also refer to Master Planning (Rohde and Wagner, 2005) models.

modeling and solving these problems. The models found in the literature can be distinguished according to the following groups of classification criteria (similarly to Meyr, 1999, p. 45):

1. *Context of model*
2. *Production system characteristics*
3. *Modeling technique*
4. *Decision variables*
5. *Objective(s)*

The classification criteria of these groups will be described in detail in the following section.

2.1.1 Classification of Models

The classification framework presented here is based on classification criteria contained in Domschke et al. (1997), Drexl and Kimms (1997), Meyr (1999), Jans and Degraeve (2006), and Quadt and Kuhn (2008). The list of classification criteria is indeed not exhaustive, but gives a sufficient overview to serve as a basis for classifying the dynamic lot-sizing models with substitutions presented in this work. Figures 2.3–2.6 summarize them in an abbreviated form.

2.1.1.1 Context of Model

Tactical vs. Operational Models

Production planning models differ in the planning horizon and level of aggregation that they use: Some models are meant to be used with a long *planning horizon* (e.g., 1 year), others with a short planning horizon (e.g., 1 week). The former models belong to the group of *tactical* or *strategic production planning models* and can include strategic decisions such as capacity expansions that have a *long-term* impact on the production system. The latter models are used for routine *short-term* production planning decisions and are termed *operational production planning models*.[2]

Centralized vs. Decentralized Production Planning

There is a general difference between production planning models that assume a single central decision maker and models that assume multiple actors/agents who

[2] Also see Fleischmann et al. (2005, p. 81f.) and Meyr (1999, p. 11ff.). regarding the classification of planning levels.

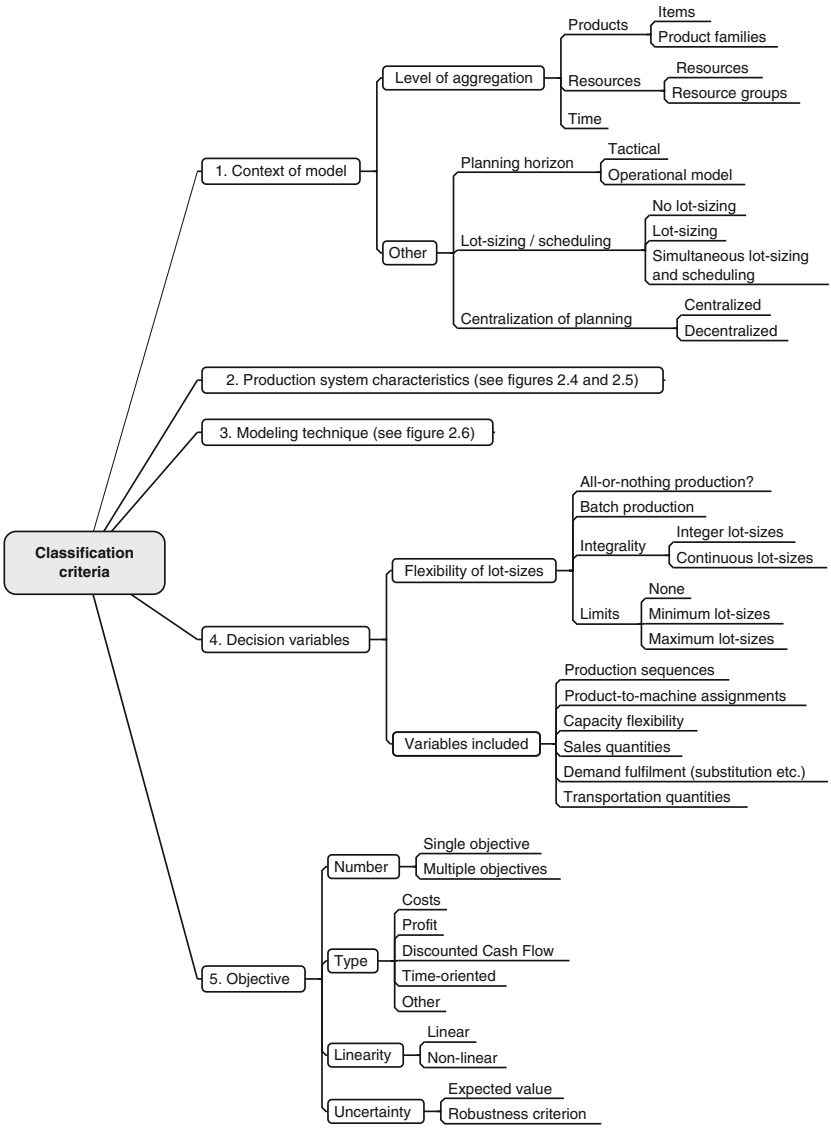


Fig. 2.3 Classification criteria for production planning models – 1/4

steer subsystems of the entire production system, e.g., subsidiary companies of a corporate group or plant managers with some autonomy. The former case allows for a centralized optimization of the system, whereas the latter case, which we will not consider in this work, requires the design of coordination mechanisms (see, e.g., Ertogral and Wu, 2000). For example, Drechsel and Kimms (2008) consider a capacitated lot-sizing model with transshipments and multiple players using a game-theoretic approach.

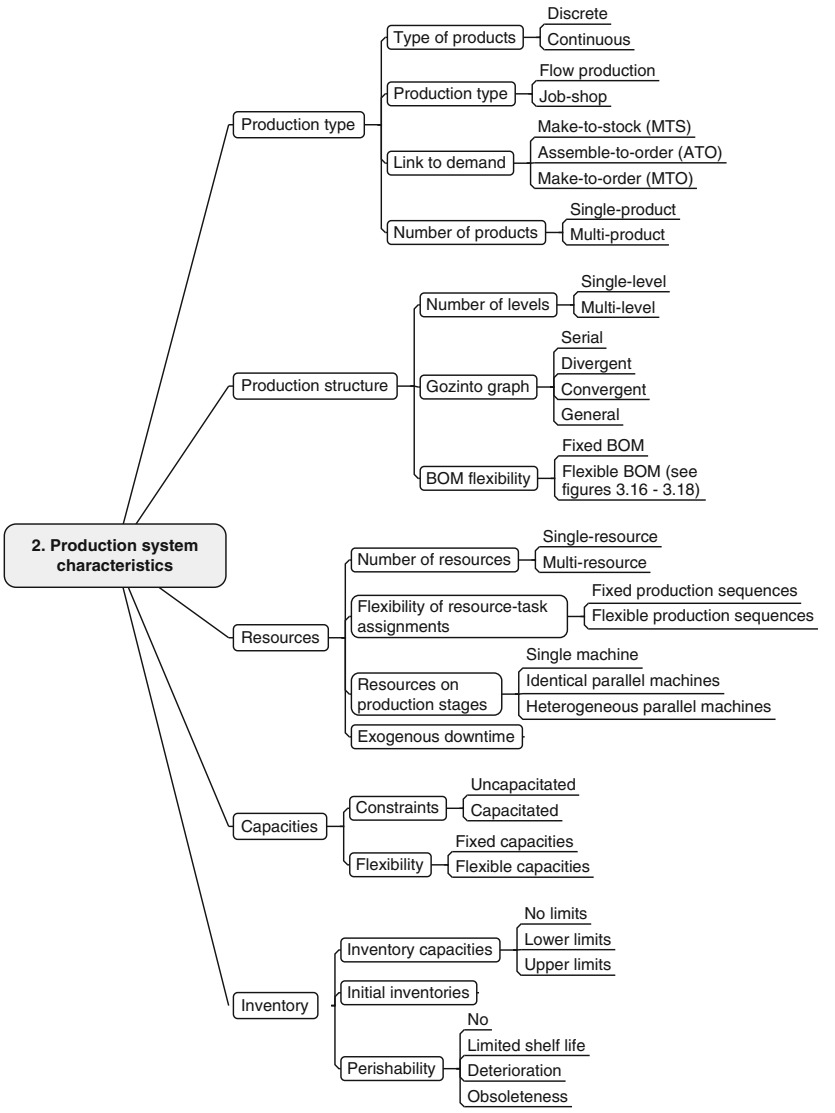


Fig. 2.4 Classification criteria for production planning models – 2/4

Aggregation

In tactical/strategic models, it is often necessary to **aggregate** certain entities (e.g., products or machines) of the production system (Fleischmann et al., 2005, p. 85). Aggregation means, e.g., that the model plans production quantities for product types instead of individual products and considers constrained capacities of entire production lines or groups of machines instead of individual machines. One important reason for aggregation is that demand forecasts in medium- or long-term

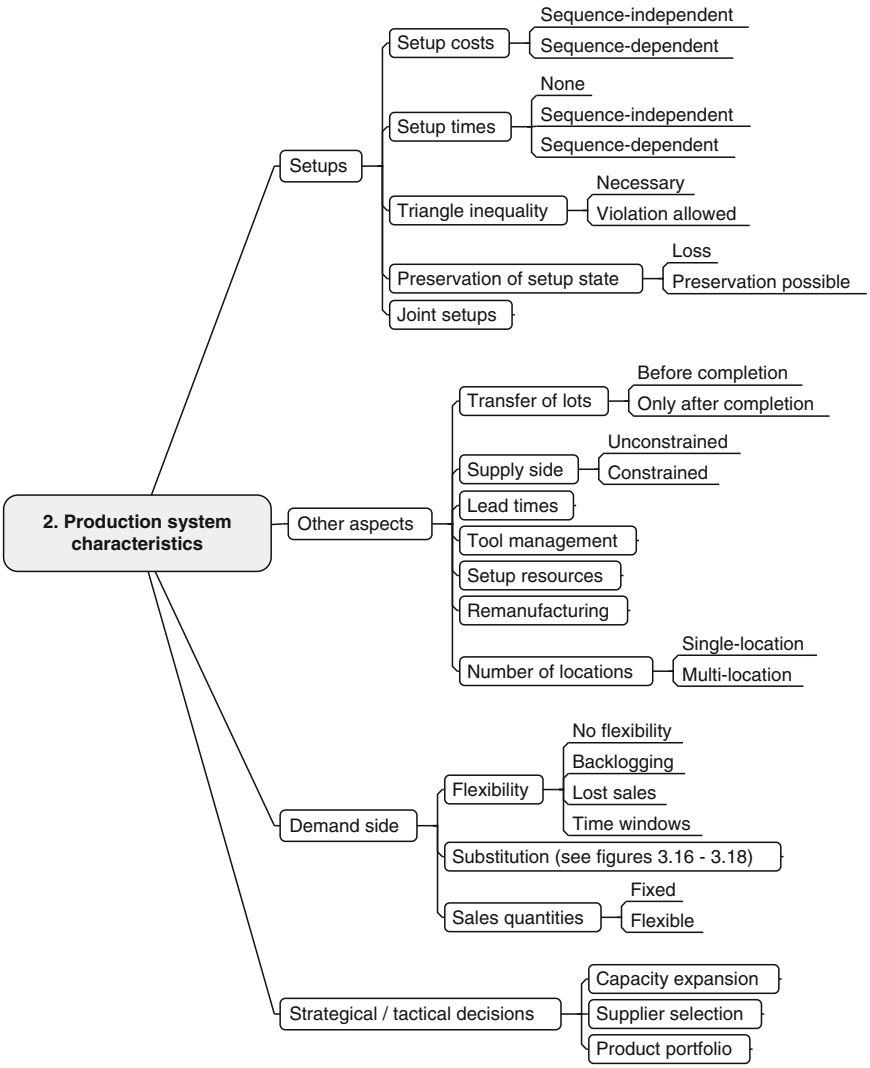


Fig. 2.5 Classification criteria for production planning models – 3/4

models on the level of individual products could involve too much uncertainty, whereas forecasts on the level of groups of products will presumably have smaller errors. Also, detailed and accurate data on products and resources might not be available and expensive to obtain, especially in large companies with complex product portfolios and manufacturing systems. Thus, it often makes sense to consider products and resources on an aggregated level. Another reason is that the model size would explode when considering the manufacturing system on the finest, most disaggregated level. Such a large model is difficult to solve, it might take a

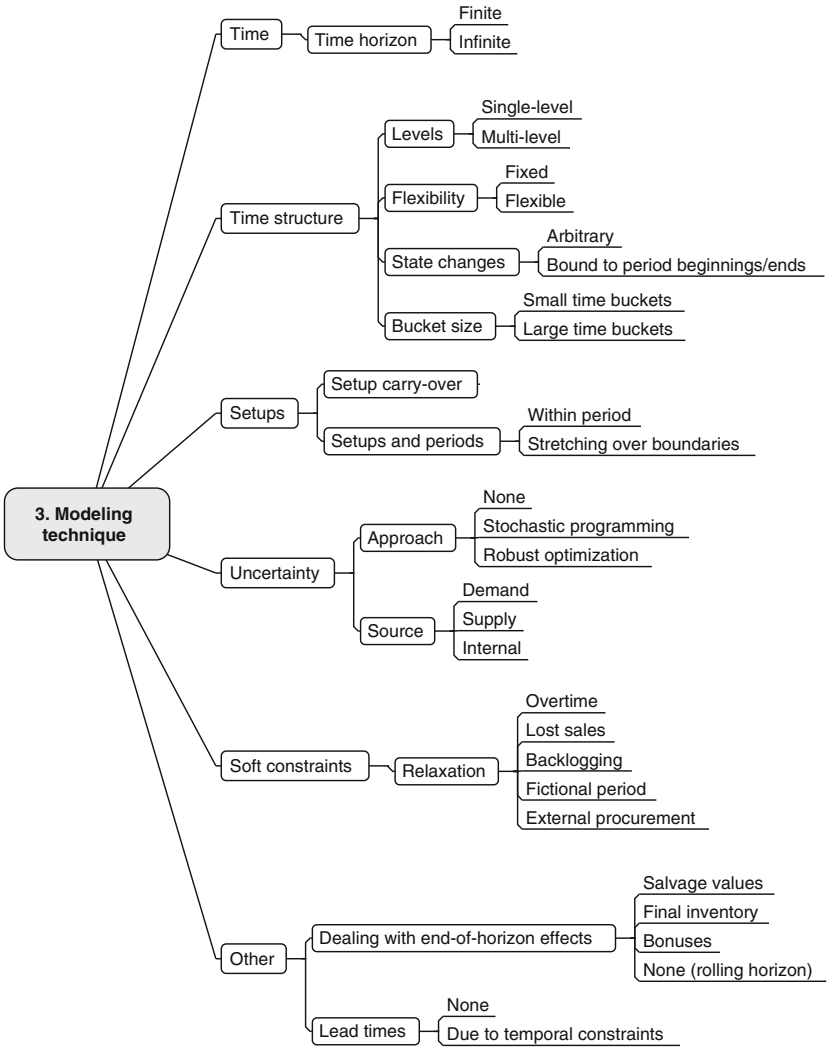


Fig. 2.6 Classification criteria for production planning models – 4/4

prohibitively long time to be solved and require more memory than available even on high-end computers.

Another dimension of aggregation – in addition to products and resources – is time: The longer the periods considered in a production model (e.g., days, weeks, or months), the higher the level of aggregation of time. Also, one could use a fine time grid in the earlier part of the model's planning horizon (e.g., hours or days) and a coarser time grid (e.g., weeks) in the later part of the time horizon, as data regarding this part is very uncertain anyway. Furthermore, by interpreting the elements of a

model in different ways, e.g., "product" as an individual product in one case and as a setup family in the other case, the same mathematical model can sometimes be used for different levels of aggregation.

Mixed Aggregation Levels

Multiple levels of aggregation might also be combined in a single model. For example, if there are joint setups for product families on a capacitated resource, i.e., no or a negligibly short setup activity is necessary when changing over between products in the same family, two levels of aggregation (individual products and setup families) of a certain entity type (products) appear in a single production planning model that maps such joint setups.

Choice of Aggregations

The number of aggregation levels for products is often three, namely individual products, setup families, and product types (also: items, families, and types). However, depending on the application, a larger or smaller number of aggregation levels might be more appropriate. The same holds true for aggregation of resources. Here, the resources (workers, machines, etc.) of an entire production line can often be considered as a single aggregated resource in case of a flow production type. When constructing aggregations of products or resources, the question is which attribute(s) should be used as a criterion for aggregation if several are available. Aggregation implicitly contains *clustering* decisions that cluster entities on a lower level into aggregates on a higher level. In practice, these aggregations might already be preset by the terminology used in a company. Yet, it could be useful to reconsider these clustering decisions.

Decomposition

As the size and complexity of a complete model of the production system might be prohibitively high, production planning problems are often *decomposed* into various subproblems. These subproblems could, e.g., refer to the production planning at different plants or to certain subsets of resources.

Hierarchical Production Planning

The idea of *hierarchical production planning* (Hax and Meal, 1975) is to sequentially solve production planning models on different levels of aggregation, starting from the most strategic, most aggregated model (also see Jans and Degraeve, 2006). A solution to a model of this type results in decisions that set limitations for

decisions in the less aggregated models solved subsequently. E.g., the "higher-level" model sets production quantities for product types, which then limit the quantities for subtypes and individual products in lower-level models. Each model approximately anticipates the decisions that will be taken in lower-level models. As this anticipation makes some simplifying assumptions and the models are interwoven, hierarchical production planning yields solutions that are suboptimal in most cases. In each step from a higher-level to a lower-level model that is more disaggregated, the data given and decisions taken in the level above have to be disaggregated, which is a non-trivial task that can be performed in different ways, influencing the quality of the overall production plan obtained. The "right" number of levels (and thus models) in a hierarchical production planning approach might depend on the characteristics of the considered setting.

2.1.1.2 Production System Characteristics

Discrete vs. Continuous Products

The products handled in a production system can be *discrete* (indivisible) or *continuous* (divisible). If discrete products are produced in large quantities, they are often modeled as continuous products (Meyr, 1999, p. 27), presumably without much loss of planning accuracy (also see de Araujo et al., 2007). Continuous products are typical for the process industry, whereas discrete products are typical for manufacturing. In some companies, both continuous and discrete products are present at different stages in the production system (e.g., in the food industry).

Production Type

Regarding the layout type and process structure of a production system, lot-sizing models can be differentiated into models designed for *job-shop production* and *flow production systems* (Buschkühl et al., 2008). The term flow production systems refers to cases where the production layout and activities are organized into a series of production stages: On each production stage, certain intermediate goods are produced using a single or multiple resources available on that stage and then transferred to the next stage. Intermediate goods can also denote certain *states* of an item that goes through the production process. Flow production models assume that each intermediate and finished good and each resource belongs to exactly one production level/stage. An example of a flow production type is given in Fig. 2.7: It shows a system with two production stages, on each of which tasks performed on certain machines (used exclusively on that stage) produce intermediate and finished goods, respectively. In contrast, the term job-shop production refers to settings where no clear – physical and organizational – serial order of production stages exists: Instead, several resources are situated at different locations within a plant without a flow structure, and the resource sequences of the intermediate goods

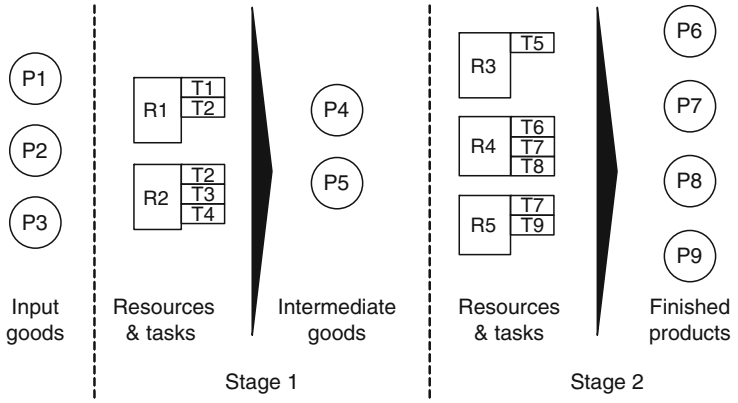


Fig. 2.7 Example – stages, resources, tasks, and products in a flow production system

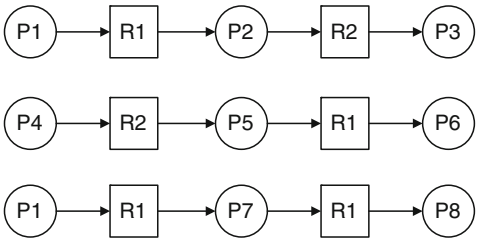


Fig. 2.8 Example – resource sequences in a job-shop production environment

required for a finished good may differ among the finished products. In addition, the same resource could be used for producing a product and another product that contains this product as a predecessor. Figure 2.8 illustrates possible resource sequences that can occur in a job-shop environment: Resources 1 and 2 are used in a different order in the resource sequence for product 6, compared to product 3. Also, in the resource sequence for product 8, resource 1 manufactures both intermediate product 7 and its successor product 8.

Referring to the relation between production and demand, one can distinguish a *make-to-order (MTO)* from a *make-to-stock (MTS)* production approach. In the former approach, production is triggered by confirmed customer orders. Thus, the inventories of finished goods are usually low in an MTO environment. In the latter approach, production quantities are mainly based on demand forecasts, they are not linked to individual customer orders. Thus, products are stocked by the manufacturer until required to fulfil customer demand in an MTS environment. In practice, also hybrid (combined) MTO and MTS production approaches occur (Denton and Gupta, 2004).

Number of Products

Another classification criterion is the *number of products* considered in the model. Some models only consider a single product, others a fixed number of products or, in the most general case, an arbitrary number of products.

Production Structure

Regarding the production structure (Gozinto graph), models can be classified into *single-level* models, models with a fixed number of levels (e.g., two) or an arbitrary number of levels (*multi-level* models). Multi-level (also: *multi-stage*) production structures can in addition be differentiated into *serial*, *convergent*, *divergent* (with by-products, e.g., in the chemical industry), and *general structures*. Production structures can also be *cyclic*.

The majority of models assumes *fixed bills-of-materials (BOMs)*, where a list with unique quantities of input goods required for one unit of a finished product exists. However, in practice, BOMs are sometimes *flexible*, i.e., alternative combinations of input goods can be used to manufacture a product. In the process industry, *alternative recipes* (which correspond to *flexible BOMs* in manufacturing) are sometimes available for producing a product (Crama et al., 2001; Kallrath, 2005). Flexible BOMs are closely related to substitution because the usage of alternative BOMs corresponds to substitutions of input or intermediate goods.

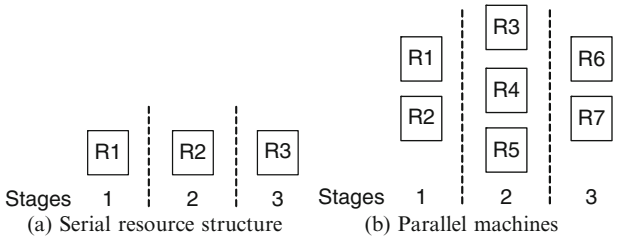
For an in-depth discussion of the topic flexible BOMs and recipes, see Sects. 3.2.3 and 3.2.4. A compact summary of classification criteria for dynamic lot-sizing models with flexible BOMs / recipes developed in these sections is contained in Fig. 3.17.

Resources

Regarding the number of resources, models can be classified into *single-* and *multi-resource* models. Note that "resource" could refer to a worker, group of workers, a single machine, a group of machines, a production line, a reactor, or a group of reactors. In most models, it is predetermined which production *task* for which product is performed on which resource, i.e., the assignment of tasks (and thus intermediate/finished products) to resources is assumed to be unique. In contrast, some models also contain decisions on *product-resource assignments*. The flexibility regarding these assignments is a special case of *alternative* (also: *flexible*) *production sequences*. E.g., in the process industries multiple production sequences on different sets of resources are often available for producing a certain product. These production sequences are frequently also linked to differing BOMs. In these cases, flexible production sequences coincide with flexible BOMs. However, the classification criteria fixed vs. flexible BOMs and fixed vs. flexible production sequences can be seen as orthogonal: In some applications, production sequences

		production sequences	
		fixed	flexible
bill of materials / recipe	fixed	standard fixed production structure	flexible production sequences, fixed BOM
	flexible	flexible BOM, resource assignments fixed	flexible production sequences and flexible BOM

Fig. 2.9 Interrelation of flexible BOMs and flexible production sequences



(a) Serial resource structure (b) Parallel machines

Fig. 2.10 Example – serial resource structure vs. parallel machines

could be flexible but the BOMs fixed, or vice versa. The four possible resulting combinations are illustrated by Fig. 2.9. A simple example of flexible production sequences would be that two alternative resource sequences exist for producing a certain product P1, e.g., the resource sequences R1–R2–R3 and R4–R2–R5.

In multi-level models with exactly one resource per level that is only used on this level, the resource structure is termed *serial*. The case where multiple resources are available on a production level of a flow production system is termed lot-sizing with *parallel machines*. Such settings with parallel machines are a special case of alternative production sequences. If the parallel machines have the same characteristics (costs, capacities and capacity consumption), this is termed *identical parallel machines*. Otherwise, i.e., if the characteristics of resources on the same level differ, this is named *heterogeneous parallel machines*. Figure 2.10 shows a case with a serial resource structure as well as another case with parallel machines on each of three stages of a flow production system.

Some models assume that production resources are *uncapacitated*, i.e., production times are zero and production quantities unlimited. This assumption is valid if the production resources are not scarce at all, if there are no production bottlenecks. However, this assumption often does not correspond to practical production planning problems. Hence, in most cases it is more realistic to assume *capacitated resources*, with non-zero production times and limited production quantities. Regarding the flexibility of resources capacities, one can distinguish models with *fixed capacities* from models with some *capacity flexibility* (e.g., by allowing

overtime production). The available capacities are either *constant* over the time horizon or *time-varying*.

Resource downtime can be classified into exogenous and endogenous downtime (Meyr, 1999, p. 48), where exogenous downtime is caused by technical and other restrictions, whereas endogenous downtime is decided on in the production planning model, e.g., downtime caused by low order volumes. Times for recurring *maintenance* that can be scheduled with some flexibility, downtime, e.g., due to external legal restrictions, and setup times are subsumed under the term *exogenous downtime*. Some lot-sizing models allow for modeling exogenous downtime.

Inventory

One can distinguish models with *limited inventory capacities* (*upper limits for inventory*) from models with unlimited inventory. In practice, storage space for input goods, semi-finished and finished products is usually limited. This limitation could be a real constraint in some cases whereas in other cases, storage space is limited but still ample, so that this constraint can be neglected. In the process industries, *lower limits for inventory* (e.g., contents of tanks) are sometimes necessary due to technical restrictions. In addition, some multi-level models assume that there are *work-in-progress (WIP) buffers* on all stages, whereas others assume that no buffer inventories are allowed between levels.

Especially in short-term production planning, *initial inventories* that are in stock at the beginning of the planning horizon cannot be neglected. Initial inventories can, in most cases, be eliminated from mathematical models by transforming gross into net demands by subtracting initial inventories.

Another aspect that significantly complicates production planning problems is *perishability*: Some (input and output) goods only have a limited shelf life, after which they expire (e.g., blood transfusions). Others have a quality that deteriorates over time, influencing the purpose for which the products can be used. Also, stocked products might get *obsolete* due to technological advances or market changes.

Setups

In order to produce a certain product on a resource, *setup* activities could be necessary before a *lot* (in the process industry also: *campaign*) of the product can be started. These activities can incur *setup costs* and require a *setup time* during which the resource is not available for production. Both setup costs and times for a product can be *sequence-dependent*, which means that they depend on the product previously manufactured on the resource. Lot-sizing decisions are usually only included in operational production planning models, but also in tactical production planning models if lot-sizes are large and take a long time (e.g., months) to be produced.

The so-called *triangle inequality* (Haase, 1996; Meyr, 1999, p. 48) for sequence-dependent setup cost and times is fulfilled if it is never faster or cheaper to perform

two subsequent changeovers from product A to B and B to C than to perform a single changeover from A to C. Mathematically, this can be expressed as follows with st_{ik} denoting the setup time and f_{ik} the setup cost for a changeover from product i to k:

$$st_{AB} + st_{BC} \geq st_{AC} \tag{2.1}$$

$$f_{AB} + f_{BC} \geq f_{AC} \tag{2.2}$$

This triangle inequality is violated in some cases in the process industries, e.g., if there is a product B that cleans the resource while being produced. Some models assume that the triangle inequality has to be fulfilled and thus cannot map these cases, e.g., the capacitated lot-sizing problem with sequence-dependent setups (CLSD) (Haase, 1996) in which each product can be set up at most once per period.

If the setup state can remain as before during idle time segments (without production on the resource), this is termed *preservation of setup state*. In other models, a *loss of setup state* occurs as soon as the production resource is idle or after a certain time. "Joint setups" for product families are included in some models (Anily et al., 2005): Here, only negligible or no setup activities are required when changing over from one product of a product family to another of the same family. Setup activities are only necessary if a change to a product belonging to another family occurs.

Transfer of Lots

Furthermore, some multi-level models assume that units of a product produced in a lot are not transferred to the next production activity until this production lot has been completed, whereas other models assume that parts of a lot can already be transferred to the next production activity before the lot has been completed.

Supply Side

Production planning models usually assume that the supply side is unconstrained, i.e., input goods are available in unlimited quantities, but one could also assume that the procurement quantities for input goods are limited to certain maxima. The latter assumption will likely coincide with settings where substitutions of input goods come into consideration. In addition, note that lot-sizing models can also be applied to procurement *order lot-sizing* instead of production lot-sizing.

Lead Times

If production planning models also map purchasing/order decisions or intra-company transportation aspects, *lead times* become relevant. In addition, minimum and maximum waiting times between stages can be necessary in multi-level systems, e.g., due to technical restrictions such as ensuring a certain temperature/state of a product.

Tool Management

In various production systems, *tools* (e.g., milling cutters) are required and often shared among products (Jans and Degraeve, 2006). Tools are only available in limited quantities and might have a limited lifetime. As they are interrelated with products and machines – production downtime occurs if required tools are not available at the right time –, some approaches have been developed to include tool management decisions in lot-sizing and scheduling models (Jans and Degraeve, 2006).

Setup Resources

The concept of *common setup resources* (Tempelmeier and Buschkühl, 2008) is related to tool management: While tools are required for production activities, common setup resources are required for setup activities. In practice, setup personnel are often responsible for performing setups at more than one machine. If the limited availability of personnel for setup activities is not included in a model, this might lead to unnecessary downtime if setup activities are scheduled to be performed at overlapping times and no sufficient setup resources are available.

Remanufacturing

The topic *reverse logistics*, especially *remanufacturing*, enjoys growing interest in the production planning and inventory control literature (Jans and Degraeve, 2006): In addition to regular production of new products, there is a return flow of used products from customers that can be reused either immediately or after reconditioning/repair. Usually, testing procedures are performed to check whether a used product is suitable for remanufacturing. Remanufacturing frequently also involves disassembly and reassembly operations. Instead of reusing a product as a whole, one could also reuse only certain components of a product or materials contained in it. A number of lot-sizing models that include remanufacturing options in addition to regular production have been developed (see, e.g., Bayindir et al., 2007; Inderfurth, 2004; Li et al., 2007). Remanufacturing decisions are also combined with *final order lot-size* decisions for *spare parts* in some models (Kleber and Inderfurth, 2007). The so-called final order is the last regular production lot of a product before the end of its production life-cycle.

Demand Side

Regarding the demand side of production planning models, the most rigid assumption is that all occurring demand has to be met immediately at fixed points in time without any delay, and the problem becomes infeasible if this is not possible with

the given production capacities. This assumption can be softened by allowing for *backorders* (also: *backlogging*), *lost sales* or specifying *delivery time windows* during which demand can be satisfied (also see Jans and Degraeve, 2006). Backordering means that demand can be fulfilled by production after the due date at a specific penalty cost that increases the longer the delay is. Lost sales denote that demand which cannot be fulfilled at the due date is lost entirely, i.e., it cannot be fulfilled later, and a certain penalty cost is incurred. Another option is to assume delivery time windows for demands (Lee et al., 2001; Brahimi et al., 2006; Wolsey, 2006): For each customer order, an earliest and latest admissible delivery date is given. Demand can be fulfilled without penalty between these dates. Note that also combinations of these assumptions are possible, e.g., delivery time windows combined with backlogging costs that are incurred for deliveries after the "latest" delivery date.

Another possibility is to specify *service level constraints* for certain types of demand: E.g., one could add a constraint that 99% of the demand of a certain high-priority customer group for a specific product should be satisfied on time.

The inclusion of backorders, lost sales, or similar softening assumptions implicitly requires that the period demands can be fulfilled *partially*. This might not be the case in practice if each of the customer orders that ultimately represent the demand in a period has to be fulfilled completely or gets canceled (*all-or-nothing order fulfillment*).

Most typical production planning models assume that the sales/demand quantities are predetermined, apart from models that allow for lost sales. Those contain a downward flexibility of the sales quantities as all quantities between zero and an upper limit are possible. Another smaller group of models assumes that sales quantities for various demand classes / market segments are not fixed, but flexible in a certain range, which leads to a profit (margin) maximization instead of a cost minimization objective (also see Jans and Degraeve, 2006). Using this approach, it is possible to optimize production systems with high- and low-margin products that compete for the same scarce resources.

Substitution

The majority of models assume that demands refer to precisely specified products, and can only be satisfied by production of these. In various practical cases, this assumption is too rigid, as the demand for a certain product can also be fulfilled by *substitute* products. E.g., demand for a low-quality product can sometimes be met by supplying a similar product with higher quality at the same sales price. In such cases, the unit costs of the low-quality product are usually lower than those of the high-quality product. Such *product substitutions* may require that the substitute is converted into the substitutable product, which may incur additional *conversion* costs. Also, applications exist where substitutions can be performed immediately without any conversions. Conversion costs can include actual transformation costs as well as opportunity costs of substitutions. For details on their semantics see Sect. 3.3.4.1.

Substitution of input and intermediate goods corresponds to flexible BOMs. Only few papers on lot-sizing with substitutions have been published (see, e.g., Balakrishnan and Geunes, 2000; Geunes, 2003; Hsu et al., 2005, which we review in Chap. 4).

For the sake of understanding, we develop our comprehensive taxonomy for lot-sizing with substitutions in Sect. 3.3 *after* introducing the required modeling framework in Sect. 3.2. This classification scheme is condensed in Figs. 3.15–3.17.

Multi-location Models

Lot-sizing models are usually applied to production planning at a single location. However, models have been developed that integrate production planning and inventory management with transportation decisions, i.e., inbound logistics and distribution, for multi-location manufacturing companies. Such models that coordinate production, inventory and transportation planning and control are called *supply chain optimization* models (also see Jans and Degraeve, 2006). According to the terminology used in literature on Advanced Planning and APS (see, e.g., Rohde and Wagner, 2005), such models belong to the category of Master Planning models, rather than Production Planning models.

The locations in these models could be plants, warehouses, retail outlets, locations of suppliers, or locations of customers. The multi-player capacitated lot-sizing model of Drechsel and Kimms (2008) belongs to the category of such multi-location models. It assumes that transshipments between the locations of the players are possible. Additional aspects that can be incorporated in multi-location models are multiple transportation modes, e.g., cheap but slower truck or train transportation vs. costly but fast air freight service, and carrier selection decisions (for a recent survey, see Meixell and Norbis, 2008).

Strategical and Tactical Decisions

Another area of research are production planning models that include *strategical* and/or *tactical* decisions such as *capacity expansion*, *acquisition* or *subcontracting* and *supplier selection* (Jans and Degraeve, 2006): In addition to the short-term capacity flexibility provided by overtime options, one can allow for expansions of production capacities by adding new production resources, upgrading existing resources, or by externally obtaining additional capacities. Such capacity expansions have a longer time range and are thus tactical actions. In terms of APS (see, e.g., Meyr et al., 2005b), models containing such decisions would be employed in the modules Master Planning or Strategic Network Planning.

In production planning models with supplier selection (Aissaoui et al., 2007), several suppliers are available for supplying input goods. The decision on selecting a combination of suppliers for the input goods is performed simultaneously with

production planning decisions. The suppliers differ in various characteristics, e.g., pricing, quality, production capacities, reliability, and lead time. Supplier selection models can be differentiated into *single-sourcing* models where exactly one supplier is selected for each input good, and *multiple-sourcing* models where one or more suppliers are selected for each input good. *Make-or-buy* decisions can also be included in lot-sizing models and combined with supplier selection decisions. Some models that include external sourcing also consider *quantity discounts* (Haase, 2001; Xia and Wu, 2007). In addition, *product portfolio* decisions could be included if sales quantities are flexible and/or substitutions possible (also see Sect. 4.1 on assortment problems).

2.1.1.3 Modeling Technique

In practice, the *state* of a manufacturing system changes over time. Some of these state changes are *exogenous*, e.g., input good deliveries from suppliers or arrivals of customer orders, some of them *endogenous*, e.g., product changeovers on machines that result from the production plan (Meyr, 1999, p. 49). Exogenous state changes result from external impacts on the manufacturing system that cannot be influenced by the planner's decisions. Endogenous state changes follow from implemented decisions made by the planner. Due to the commonly high interaction between a manufacturing system and its environment, it seems useful to distinguish between the *internal* and *external dynamics* of a system based on this difference between endogenous and exogenous state changes (Meyr, 1999, p. 49). State changes of a system either happen as a *discrete* event at a single point in time (e.g., the setup of a machine if the setup time is zero) or as a *continuous* process over a time interval (e.g., the inventory reduction of a tank with a liquid). Frequently, continuous state changes are modeled as discrete state changes because these are easier to model with MILP. Note that the terminological boundary between exogenous and endogenous might be blurred because, e.g., stochastic input good delivery arrivals from suppliers would not happen if the decision maker had not placed the corresponding order beforehand.

Time Horizon

The *time horizon* in production planning models is either assumed to be *infinite* or *finite* (Domschke et al., 1997, p. 70f.). An infinite time horizon mostly coincides with the assumption that manufacturing system and its environment is *static*, i.e., the system of course changes over time, but with a recurring pattern. A finite time horizon is mostly combined with the assumption that the system is *dynamic*, i.e., the external dynamics do not show a simple recurring pattern. In the following, we will focus on dynamic models with a finite time horizon.

Time Structure

When developing production planning models, the time horizon is usually segmented into a number of *periods* (also: *time buckets*) with a certain fixed, identical duration. Both exogenous and endogenous state changes are attached to the beginnings or ends of these periods. That is, the same time structure is used to map internal and external dynamics. As the lengths of all periods are identical, such models cannot map state changes at arbitrary points in time but only at the beginnings or ends of periods, which might result in unprecise planning results unless a large number of short periods is introduced.

However, some newer lot-sizing and scheduling models distinguish an exogenous from an endogenous time structure, e.g., the Discrete Lot-sizing and Scheduling Problem (DLSP) with sequence-dependent setup costs (Fleischmann, 1994) and the General Lot-sizing and Scheduling Problem (GLSP) (Fleischmann and Meyr, 1997). So-called *macro-periods* map the exogenous time structure, *micro-periods* the endogenous time structure. In contrast to other models with a *single-level time structure*, these models thus have a *multi-level time structure*. Each macro-period contains one or more micro-periods. Thus, the lengths of micro-periods are assumed to be shorter than the lengths of macro-periods. The underlying assumption is that endogenous state changes are required more frequently than occurring exogenous state changes. Both exogenous and endogenous *time structure* are either *fixed* (i.e., state changes can only occur at certain fixed points in time) or *flexible* (i.e., state changes can happen at arbitrary points in time). Here, "fixed points in time" refers to beginnings or ends of time periods that are predetermined. The lengths of periods might be non-identical. Arbitrary points in time for state changes can be modeled by treating period lengths and beginnings or ends of periods as decision variables instead of parameters, and by allowing state changes that are not bound to beginnings and ends of periods. An example of a two-level time structure is given in Fig. 2.11. Each of the four macro-periods contains a certain set of micro-periods. The beginnings and ends of micro-periods may be flexible, i.e., decision variables. Two examples of production plans for a resource are shown below the macro- and micro-periods, assuming given beginnings and durations of micro-periods: One where the state changes are bound to beginnings and ends of micro-periods and

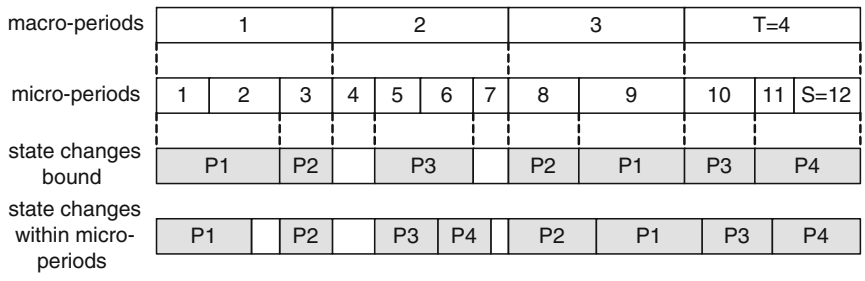


Fig. 2.11 Example of a two-level time structure

another plan where also state changes within micro-periods are possible. Such state changes within a micro-period could for instance be as follows:

- A changeover started in the previous micro-period ends after some time in the micro-period, and production of a product begins.
- Production of a product ends, following by some idle time.
- A changeover to another product starts, either immediately after production or after some idle time.

If models have time buckets with rather short durations (e.g., hours), these are called *small time bucket (STB)* models. Models with time buckets that have a longer duration (e.g., weeks or months) are called *large time bucket (LTB)* models. Note that this classification criterion refers to the macro-periods when considering models with a multi-level time structure. A closely related classification criterion is the *maximum number of lots* or *maximum number of endogenous state changes* within a period. In some models, only one product can be produced per period, others allow for two or an arbitrary but fixed number of products. STB models only allow for a small number (e.g., 1 or 2), whereas LTB models allow a large number of lots / endogenous state changes per period. In addition, many models only allow that each product is set up maximally once per period, whereas others allow multiple setups for the same product within a period. The latter might be advantageous in the process industries in cases where the triangle inequality is violated.

Dealing with End-of-Horizon Effects

So-called *end-of-horizon effects* can occur in the last periods of a production planning model: As final inventories are assumed to be worthless in a large number of lot-sizing models, there is no tendency in their optimal solutions to produce units to stock in the last periods. This leads to solutions where the production plan empties the inventory towards the end of the planning horizon, and no lots are produced in the last periods. This might yield production plans that are nonsensical from a practical point of view. Several means to counteract these end-of-horizon effects have been developed, such as defining *salvage values* for products or *final inventory* (also: *terminal inventory*) targets, or rewarding lots in the last periods with *bonuses*, i.e., by decreasing the setup costs for these lots (Stadtler, 2000). Production planning models are often used in a *rolling-horizon* environment where, as new data becomes available, the model is solved repeatedly with updated data (Chand et al., 2002) in a *replanning cycle* of a certain duration. Only the decisions in the first period(s) of the optimal solution are actually implemented. Hence, end-of-horizon effects could be alleviated by the usage of a model in a rolling-horizon environment.

Setups and Periods

Another modeling aspect is whether setup states are assumed to remain preserved between periods (so-called *setup carry-over* or *linked lot-sizes*) or get lost (Suerie

and Stadtler, 2003). Oftentimes, the assumption that setup states get lost is only a simplification for modeling reasons.

Models also differ in another aspect: Some assume that each setup has to fit into one period, which requires that the maximum setup time is shorter than a chosen fixed period length, whereas others assume that a setup activity can stretch over period boundaries, e.g., a setup activity starts somewhere in period t and ends in period $t + 1$.

Lead Times

In some cases, lead times between production stages are assumed for modeling reasons: For example, in the multi-level capacitated lot-sizing problem (MLCLSP) (see, e.g., Stadtler, 1996; Buschkühl et al., 2008) there is a minimum lead time of one period before produced units can be used on the next production stage to avoid violations of temporal constraints.

Stochastic Lot-Sizing

In most manufacturing companies, there is some degree of uncertainty in production planning, both with respect to the environment (suppliers, customers, etc.) and the manufacturing system itself. However, most dynamic lot-sizing models make the simplifying assumption that demands, processing times and all other data are *deterministic*. Some research has been performed to overcome this limitation by developing *stochastic lot-sizing models* and appropriate solution techniques (see, e.g., Huang, 2005; Raa and Aghezzaf, 2005; Beraldi et al., 2006; Brandimarte, 2006; Guan et al., 2006; Snyder, 2006; Leung et al., 2007a,b; Tempelmeier, 2007). Such models often assume that only some parameters are stochastic, e.g., the demands, and all other data deterministic. Different approaches for modeling uncertainty can be used, e.g., two-stage and multi-stage *Stochastic Programming (SP)* or *Robust Optimization (RO)*.

Soft Constraints

When mathematically modeling dynamic lot-sizing problems using MILP, constraints are usually mapped as *hard constraints*, and a problem is considered infeasible if not all of these can be fulfilled. However, in practice constraints are often not as hard as specified in a model. For instance, a machine might continue with a task for another 10 min even if the planned daily operating time of x hours has already been exceeded. Also, even if a machine runs 24 h per day, it might be possible to slightly increase its throughput/capacity because it is by default run below its technical limits.

For these reasons, rather than specifying all constraints as hard constraints, it is often more realistic to convert some of them into *soft constraints*. Violations of those

soft constraints are not assumed to result in infeasibility of a solution, but associated with certain penalty costs.

In the following, we name several constraints types of dynamic lot-sizing models that are typical candidates for being relaxed into soft constraints:

- Capacity constraints can be relaxed by allowing for overtime production. A similar approach is to introduce an *fictional period* 0 at the beginning of the planning horizon with unlimited production capacities and high penalty costs for production (Fleischmann and Meyr, 1997). Also, one could assume that *external procurement* of input, intermediate and finished goods is possible at a certain penalty cost (Meyr, 2004a).
- Demand satisfaction constraints can, e.g., be relaxed by allowing for lost sales or backlogging.
- Other possible relaxations include penalties for minimum safety stock violations and exceedance of perishable product shelf lives.

Another reason for converting hard into soft constraints could be that, when solving MILP lot-sizing models with heuristic or exact algorithms, it is sometimes very difficult to find an initial feasible solution that satisfies constraints. In such cases, penalty costs can serve to guide the algorithm to solutions that preferably do not violate the soft constraints. Another potential advantage of using soft constraints is to trace data errors in problem instances that seem infeasible.

2.1.1.4 Decision Variables

Flexibility of Lot-Size Decisions

If the endogenous time structure of the model is fixed, i.e., the durations of micro-periods are fixed and endogenous state changes can only occur at beginnings or ends of periods, there are only two options for the production on a resource in a certain period: Either, one product is produced throughout the entire period, or no production at all takes place (so-called *all-or-nothing production*) (Meyr, 1999, p. 53). In this context, a lot consists of the total production of several subsequent micro-periods during which the same product is produced. Thus, only integer multiples of the production quantity of a single period can occur, given that all periods have an identical length.

Another field of research are production planning models with *batch production*, where lots can only be created by combining several production orders for the same product. *Family batching* means that lots can also be formed by combining orders for products belonging to the same product family (Meyr, 1999, p. 53).

If the products are continuous or discrete but the production quantities rather large, one can use continuous decision variables for lot-sizes. However, if a manufacturing system with rather small production quantities and discrete products is considered, it might make sense to use integer variables for lot-sizes to avoid a loss of planning accuracy and ambiguity of solutions depending on the rounding procedure used.

In some applications, especially in the process industries, technical restrictions might require to specify minimum and maximum lot-sizes or that lot-sizes always have to be a multiple of a certain base size (Meyr, 1999, p. 54).

Other Variables

Depending on the assumptions for the scope of the model and the flexibility of the manufacturing system, the model might contain (amongst others) the following decision variables in addition to setup variables, inventory levels and production quantities:

- Substitution quantities for possible substitutions
- Variables describing demand fulfilment, e.g., lost sales or backlogging variables
- Additional variables describing production sequences if the model includes scheduling decisions
- Variables for product-to-machine assignments if these are not fixed
- Variables for capturing capacity flexibility (e.g., overtime variables)
- Sales quantities if these are flexible in the model
- Transportation quantities if it is a multi-location model

2.1.1.5 Objective(s)

The common objective in deterministic dynamic lot-sizing models is to minimize the total cost over the entire time horizon. Other objectives, such as time-related objectives, e.g., the minimization of deviations from due dates, are often mapped to monetary objectives by including them using penalty costs. The cost minimization objective usually contains variable holding costs for inventory that represent capital lockup and storage costs, and fixed setup costs. These two cost categories are partially conflicting objectives, as a reduction of the total setup costs often leads to an increase of the holding costs due to lot-size-induced inventories. In addition to holding and setup costs, various other cost and revenue categories can be included in the objective function, e.g.:

- Variable production costs have to be included if one or more of the following cases apply:
  - Alternative production sequences.
  - Heterogeneous parallel machines.
  - Flexible BOMs exist.
  - Product substitutions are possible.
  - The variable production costs are time-varying.

  In these cases, variable production costs are not irrelevant any more as they were in basic lot-sizing models.
- Conversion costs (also: substitution costs) that might be incurred by substitutions. These could, e.g., represent the costs of labor time of manual activities

necessary to use one product as a substitute for another. Also, if the objective does not explicitly contain sales prices or unit costs of substitutable products, they may include opportunity costs of substitutions. The interpretation of conversion costs is treated in detail in Sect. 3.3.4.1.
- Sales prices have to be included if sales quantities are flexible, which leads to a profit (margin) maximization objective.
- Overtime and other capacity flexibility costs.
- Backlogging and lost sales costs.
- Transportation costs.
- Costs for preserving the setup state of a resource if applicable.

Commonly, the cost minimization objective is modeled as a linear function, but also non-linear costs could be considered, e.g., to model quantity discounts in models with suppliers selection.[3] As an alternative to cost-oriented objectives, models with *Discounted Cash Flow (DCF)* oriented objectives have been developed that correctly map financial aspects in tactical/strategic models with a longer time horizon (Helber, 1994; Fleischmann, 2001).

Regarding stochastic lot-sizing models, the question is whether taking the expected total cost or revenue as an objective is the right approach, because expected cost minimization or revenue maximization might lead to production plans that yield poor results in some scenarios. Hence, robust optimization approaches for finding solutions that perform well even in adversarial scenarios might be more suitable (Scholl, 2001; Gebhard and Kuhn, 2007).

In the following subsections, we review selected dynamic lot-sizing models:

- The simple, uncapacitated, single-product Wagner–Whitin problem (WWP)
- The capacitated lot-sizing problem (CLSP)
- The capacitated lot-sizing problem with sequence-dependent setups (CLSD)
- The general lot-sizing and scheduling problem (GLSP)
- The multi-level capacitated lot-sizing problem (MLCLSP)
- The general lot-sizing and scheduling problem for multiple production stages (GLSPMS)

[3] Note that the assumption of linear holding costs underlying most dynamic lot-sizing models is only a simplification: Usually, "snapshots" of the current inventory of a product are taken at the ends/beginnings of (macro-)periods to approximate the actual average inventory in each (macro-) period. For an exact calculation of holding costs based on (marginal) inventory changes within each period, it would be necessary to multiply the holding costs per quantity and time unit with the integral over the inventory level as a function of the current time. This is because the inventory of a product can change within a period if the production speed is finite. For example, considering the CLSD that assumes fixed production speeds (see Sect. 2.1.4), the inventory level as a function of time would be a piecewise linear function. However, the common approach of approximating it seems sufficiently precise for practical purposes. Beyond this, the general idea of using holding costs can be criticized because it is difficult to measure these opportunity costs due to capital lockup in practice. Also, it should be noted that lot-sizing problem data like demand forecasts are subject to uncertainty anyway, which might render the mentioned imprecision of holding costs calculations insignificant. On the accurate calculation of holding costs in dynamic lot-sizing models, also see Stammen-Hegener (2002, p. 143f.).

Though being too simplistic to capture the complexity of real-world lot-sizing problems, the WWP is the root of the more sophisticated dynamic lot-sizing models, and is hence included here. CLSP and CLSD are single-level big-bucket models, the latter with sequence-dependent setups. The GLSP is a single-level model with a two-level time structure. The MLCLSP is a straightforward extension of the CLSP for multi-level production structures. A more realistic and detailed model is the GLSPMS, a multi-level model that enhances the GLSP.

Note that different publications often use the same name (e.g., CLSP) for slightly different models: Frequently, there are subtle or also larger differences in the model assumptions: Oftentimes, one paper assumes time-varying whereas the other assumes time-invariant values for certain parameters (e.g., production costs). Sometimes setup times are included, sometimes not. Also, the models can differ as to the usage of a relaxation (e.g., backlogging, overtime) and the assumption of initial inventories.

2.1.2 The Wagner–Whitin Problem

The most basic dynamic lot-sizing problem is the *Wagner–Whitin Problem* (WWP) (Wagner and Whitin, 1958). Its assumptions are as follows (also see Domschke et al., 1997, p. 115):

- Lot-sizing for a single, continuous product (can also be interpreted as order instead of production lot-sizing).
- All parameters are deterministic.
- Finite time horizon with T periods.
- Time-varying demand that has to be satisfied at the beginning of each period.
- Production takes place at the beginning of each period (and can also fulfill the demand of that period).
- Uncapacitated production with infinite speed.
- No setup carry-over.
- Time-varying fixed setup costs and linear holding costs.
- Time-invariant variable production costs that are thus irrelevant for finding the optimal solution.
- No lead times.
- All occurring demand has to be fulfilled immediately (no relaxation).
- No initial inventories (without loss of generality).
- Cost minimization objective.
- Continuous variables for lot-sizes.

Using the notation given in Table 2.1, the WWP can be formulated as follows:

$$\text{Minimize } F(q, x, I) = \sum_{t=1}^{T} (f_t x_t + h_t I_t) \tag{2.3}$$

Table 2.1 Notations for WWP

Symbol	Definition
Constants	
T	Number of periods
Indices and sets	
$t = 1, \ldots, T$	Periods
Parameters	
d_t	Demand for product in period t
h_t	Non-negative holding cost for storing one unit of product in period t
I_0	Initial inventory of product
f_t	Fixed setup or order cost for product in period t
M	Sufficiently large number
Variables	
q_t	Production or order quantity of product in period t
I_t	Inventory of product at the end of period t
x_t	Binary variable that indicates whether a setup for the product occurs in period t

subject to

$$I_t = I_{t-1} + q_t - d_t \qquad t = 1, \ldots, T \tag{2.4}$$

$$I_0 = 0 \tag{2.5}$$

$$q_t \leq M \cdot x_t \qquad t = 1, \ldots, T \tag{2.6}$$

$$q_t, I_t \geq 0 \qquad t = 1, \ldots, T \tag{2.7}$$

$$x_t \in \{0, 1\} \qquad t = 1, \ldots, T \tag{2.8}$$

The objective (2.3) is to minimize the sum of setup and holding costs. The inventory balance equations are given by (2.4). Equation (2.5) specifies that the initial inventories are zero. Equation (2.6) enforces that the setup variable x_t is one if production takes place in period t, i.e., if $q_t > 0$. Equations (2.7)–(2.8) define the variable domains.

2.1.3 The Capacitated Lot-Sizing Problem

The *capacitated lot-sizing problem (CLSP)* (see, e.g., Karimi et al., 2003) is a well-known big-bucket single-resource capacitated lot-sizing model that allows to produce all products in each period. Its assumptions can be summarized as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.
- Single-level time structure.
- Finite time horizon with T periods.

- An arbitrary number of products can be set up in each period (big bucket model).
- No scheduling (sequencing) of products within periods (resulting from the model assumptions, all permutations of the products' lots within a period result in the same objective value).
- Time-varying demand for products that has to be satisfied at the end of each period.
- Demand always refers to exactly specified products (no substitution).
- Single capacitated production resource with finite speed, all products share this resource.
- Capacity consumption per unit produced differ among products.
- No setup carry-over.
- Single production level.
- Time-varying sequence-independent setup costs.
- Time-invariant linear holding costs.
- No setup times.
- Time-varying variable production costs.
- No lead times.
- All occurring demand has to be fulfilled immediately (no relaxation).
- No initial inventories (without loss of generality).
- Cost minimization objective.
- Continuous variables for lot-sizes.

It can easily be extended to a multi-resource version or a version with setup carry-overs (the Capacitated Lot-Sizing Problem with Linked Lot-Sizes (CLSPL), Suerie and Stadtler, 2003).

Using the notation given in Table 2.2, a CLSP formulation is given by:

$$\text{Minimize } F(q, x, I) = \sum_{i \in P} \sum_{t=1}^{T} (p_{it} q_{it} + h_i I_{it} + f_{it} x_{it}) \tag{2.9}$$

subject to

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{2.10}$$

$$I_{i0} = 0 \qquad i \in P \tag{2.11}$$

$$\sum_{i \in P} \left(\kappa_i^p q_{it} + st_i x_{it} \right) \leq K_t \qquad t = 1, \ldots, T \tag{2.12}$$

$$q_{it} \leq M \cdot x_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{2.13}$$

$$q_{it}, I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{2.14}$$

$$x_{it} \in \{0, 1\} \qquad i \in P,\ t = 1, \ldots, T \tag{2.15}$$

Table 2.2 Notations for CLSP

Symbol	Definition
Constants	
m	Number of products
T	Number of periods
Indices and sets	
$i \in P = \{1, \ldots, m\}$	Products
$t = 1, \ldots, T$	Periods
Parameters	
d_{it}	Demand for product i in period t
h_i	Non-negative holding cost per period for storing one unit of product i
p_{it}	Unit production cost of product i in period t
I_{i0}	Initial inventory of product i
f_{it}	Fixed setup or order cost for product i in period t
K_t	Capacity of resource available in period t
st_i	Setup time for product i (in capacity units)
κ_i^p	Capacity required for manufacturing one unit of product i
Variables	
q_{it}	Production quantity of product i in period t
I_{it}	Inventory of product i at the end of period t
x_{it}	Binary variable that indicates whether a setup for product i occurs in period t

The objective (2.9) is composed of setup costs, variable production costs, and holding costs. The inventory balance equations are given by (2.10), analogously to the WWP. Equation (2.11) specifies that the initial inventories are zero. The constrained capacity of the production resource is modeled by (2.12): It ensures that the capacity consumption by setup and production activities never exceeds the available capacity in a period. Equation (2.13) enforces that the setup variable x_{it} is one if production of product i takes place in period t, i.e., if $q_{it} > 0$. Equations (2.14)–(2.15) define the variable domains.

2.1.4 The Capacitated Lot-Sizing Problem with Sequence-Dependent Setups

The *capacitated lot-sizing problem with sequence-dependent setups (CLSD)* (Haase, 1996) is an extension of the CLSP by sequence-dependent setup costs and times. As the objective value of a solution also depends on scheduling decisions within a period due to this differing assumption, the CLSD is a combined lot-sizing and scheduling model. Its assumptions can be summarized as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.

- Single-level time structure.
- Finite time horizon with T periods.
- An arbitrary number of products can be set up in each period, provided that the durations of required changeovers fit into the capacity (big bucket model).
- Scheduling (sequencing) of products within periods.
- Time-varying demand for products that has to be satisfied at the end of each period.
- Demand always refers to exactly specified products (no substitution).
- Single capacitated production resource with finite speed, all products share this resource.
- Capacity consumption per unit produced differ among products.
- Setup carry-over is possible.
- Single production level.
- Time-varying sequence-dependent setup costs and setup times (these consume capacity).
- Each product can be set up at most once per period, thus the setup costs and times need to fulfill the triangle inequality (see Sect. 2.1.1.2).
- Setups have to be completed within a single period.
- Time-invariant linear holding costs.
- Time-invariant production costs that are thus irrelevant for finding the optimal solution.
- No lead times.
- All occurring demand has to be fulfilled immediately (no relaxation as, e.g., backlogging).
- No initial inventories (without loss of generality).
- Initial setup state is given (machine set up for a certain product).
- Cost minimization objective.
- Continuous variables for lot sizes.

Using most of the CLSP notation given in Table 2.2 and several additional symbols introduced in Table 2.3, the CLSD can be formulated as follows:

$$\text{Minimize } F(q,x,z,I,v) = \sum_{i\in P}\sum_{t=1}^{T}\left(h_i I_{it} + \sum_{k\in P} f_{ik}x_{ikt}\right) \tag{2.16}$$

subject to

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \qquad i \in P,\ t = 1,\ldots,T \tag{2.17}$$

$$\sum_{i\in P}\left(\kappa_i^p q_{it} + \sum_{k\in P} st_{ik}x_{ikt}\right) \le K_t \qquad t = 1,\ldots,T \tag{2.18}$$

Table 2.3 Notations for CLSD

Symbol	Definition
Indices and sets	
$i \in P = \{0, \ldots, m\}$	Products including dummy product 0
$0 \in P$	Dummy product for modeling time during which a resource is not set up for any product
Parameters	
f_{ik}	Setup cost that is incurred when the setup state of the machine changes from product i to k
st_{ik}	Setup time for changeover from product i to k
z_{i1}	Binary parameter that indicates whether the resource is already set up for i at the beginning of the first period
Variables	
x_{ikt}	Binary variable that indicates whether product k is set up immediately after product i in period t
z_{it}	Binary variable that indicates whether the machine is already set up for product i at the beginning of period t
v_{it}	Auxiliary variable: the larger it is, the later product i is scheduled in period t

$$\kappa_i^p q_{it} \leq K_t \left(\sum_{k \in P} x_{kit} + z_{it} \right) \qquad i \in P,\ t = 1, \ldots, T \tag{2.19}$$

$$\sum_{i \in P} z_{it} = 1 \qquad t = 1, \ldots, T \tag{2.20}$$

$$\sum_{h \in P} x_{hit} + z_{it} = \sum_{k \in P} x_{ikt} + z_{i,t+1} \qquad i \in P,\ t = 1, \ldots, T \tag{2.21}$$

$$v_{kt} \geq v_{it} + 1 - |P|\,(1 - x_{ikt}) \qquad i, k \in P,\ i \neq k,\ t = 1, \ldots, T \tag{2.22}$$

$$q_{it}, I_{it}, v_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{2.23}$$

$$I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{2.24}$$

$$v_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{2.25}$$

$$x_{ikt} \in \{0, 1\} \qquad i, k \in P,\ t = 1, \ldots, T \tag{2.26}$$

$$z_{it} \in \{0, 1\} \qquad i \in P,\ t = 2, \ldots, T + 1 \tag{2.27}$$

The objective (2.16) is to minimize the sum of holding costs and sequence-dependent setup costs. A dummy product 0 is introduced for modeling the state where the resource is not set up for any product: If a changeover to this product occurs, this means that the previous setup state for another product gets lost and the resource is not set up for any real product. The inventory balance equations are given by (2.17). The constrained capacity of the production resource is modeled by (2.18): It ensures that the capacity consumption by production activities and sequence-dependent setup times never exceeds the available capacity in a period.

Equation (2.19) enforces that production of product i only takes place in period t if the resource was already set up for i at the end of period $t-1$ or a changeover to i is performed in t. Equation (2.20) means that exactly one product is set up on the resource at the end of each period $t-1$ and thus at the beginning of each period t. The so-called "setup state flow preservation / sub-tour elimination" is modeled by (2.21). It maps the following aspects:

- If a product was set up at the beginning of a period t and no changeover takes place in the period, it is still set up at the beginning of the next period $t+1$.
- If a changeover to a product i occurs in a period t and no further changeover from i to another product is performed in the period, i is still set up at the beginning of the next period $t+1$.
- If a changeover to a product i occurs in a period t and a further changeover from i to another product is performed in the period, i is not set up at the beginning of the next period $t+1$.

Equation (2.22) creates a production sequence for the products within each period using the auxiliary variables v_{it}. If $v_{kt} > v_{it}$, this means that product k is scheduled for production later than product i in period t, i.e., immediately after i or with some other products in between.[4] The production sequence associated with a CLSD solution can easily be determined by sorting the product indices in P in ascending order of the v_{it} values. Equations (2.23)–(2.27) define the variable domains.

2.1.5 The General Lot-Sizing and Scheduling Problem

The *general lot-sizing and scheduling problem (GLSP)* (Fleischmann and Meyr, 1997; Meyr, 1999, 2000, 2002) is a single-level lot-sizing and scheduling problem with a two-level time structure. The assumptions of its most general version with sequence-dependent setup costs and times, multiple heterogeneous parallel machines, setup state preservation as well as the possibility of setup loss can be summarized as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.
- Two-level time structure.
- Finite time horizon with T macro-periods and S micro-periods.
- Each macro-period contains a predetermined set of micro-periods, the number of micro-periods can differ among macro-periods.
- The beginning and length of each macro-period are fixed by the capacities of the preceding macro-periods and its own capacity.
- Micro-period beginnings and lengths are flexible, except for the beginnings of micro-periods that are the first micro-period within a macro-period.

[4] This constraint group is based on the same idea as the Miller–Tucker–Zemlin subtour elimination constraints for the Asymmetric Traveling Salesman Problem (ATSP) (Domschke, 1997, p. 106f.).

- Only a single product can be set up and produced in each micro-period.
- Scheduling (sequencing) of products within macro-periods.
- Time-varying demand for products that has to be satisfied at the end of each macro-period.
- Demand always refers to exactly specified products (no substitution).
- Multiple capacitated production resources with finite speed.
- Heterogeneous parallel resources.
- Capacity consumption per unit produced differ among products.
- Setup carry-over is possible.
- Single production level.
- Time-invariant sequence-dependent setup costs and setup times (these consume capacity).
- Changeovers are performed at beginnings of micro-periods.
- Setups have to be completed within a single period.
- Time-invariant linear holding costs that are incurred for inventory at the end of each macro-period.
- Time-invariant production costs that differ among the parallel resources and thus have to be included.
- Time-invariant costs for preservation of setup state for a product on a certain resource (per time unit).
- No lead times.
- All occurring demand has to be fulfilled immediately (no relaxation).
- Initial inventories.
- Minimum production quantity after changeover.
- Minimum time for resource staying in idle state (without production).
- Initial setup state given for each resource.
- Cost minimization objective.
- Continuous variables for production quantities.

The GLSP is formulated using the notations contained in table 2.4:

$$\text{Minimize } F(q,x,z,I,\Psi) = \sum_{i\in P\setminus\{0\}}\sum_{t=1}^{T} h_i I_{it} + \sum_{i\in P}\sum_{r\in R}\sum_{s=1}^{S}\left(p_{ris}q_{ris} + \overline{p}_{ris}\Psi_{ris} + \sum_{k\in P} f_{rik}x_{riks}\right) \quad (2.28)$$

subject to

$$I_{it} = I_{i,t-1} + \sum_{r\in R}\sum_{s\in S_t} q_{ris} - d_{it} \qquad i \in P\setminus\{0\},\ t = 1,\ldots,T \quad (2.29)$$

$$\sum_{i\in P}\sum_{s\in S_t}\left(\kappa^{p}_{ri}q_{ris} + \Psi_{ris} + \sum_{k\in P} st_{rik}x_{riks}\right) = K_{rt} \qquad r \in R,\ t = 1,\ldots,T \quad (2.30)$$

Table 2.4 Notations for GLSP

Symbol	Definition
Constants	
m	Number of products
T	Number of macro-periods
S	Number of micro-periods
n_r	Number of resources
Indices and sets	
$i \in P = \{0, \ldots, m\}$	Products including dummy product 0
$0 \in P$	Dummy product for modeling time during which a resource is not set up for any product
$t = 1, \ldots, T$	Macro-periods
$s = 1, \ldots, S$	Micro-periods
$r \in R = \{1, \ldots, n_r\}$	Resources
P_r	Set of products that are manufactured using resource r
S_t	Set of micro-periods belonging to macro-period t ($\subseteq S$)
$t(s)$	Macro-period t to which micro-period s belongs
Parameters	
d_{it}	Demand for product i in macro-period t
h_i	Non-negative holding cost per macro-period for storing one unit of product i
p_{ris}	Unit production cost of product i on resource r in micro-period s
$\overline{p}_{ris}$	Cost for preserving setup state of product i on resource r in micro-period s per capacity unit
I_{i0}	Initial inventory of product i
K_{rt}	Capacity of resource r available in macro-period t
f_{rik}	Setup cost that is incurred when the setup state of resource r changes from product i to k
st_{rik}	Setup time for changeover from product i to k on resource r
κ^p_{ri}	Capacity required for manufacturing one unit of product i on resource r
m_{ri}	Minimum production quantity for product i after changeover on resource r
z_{ri1}	Binary parameter that indicates whether resource r is set up for i at the beginning of the first micro-period
Variables	
q_{ris}	Production quantity of product i on resource r in micro-period s
I_{it}	Inventory of product i at the end of macro-period t
x_{riks}	(Binary) variable that indicates whether a changeover from product k to product i is performed on resource r in micro-period s
z_{ris}	Binary variable that indicates whether resource r is already set up for product i at the beginning of micro-period s or a changeover to it is completed in s
Ψ_{ris}	Time during which setup state of product i on resource r is preserved in micro-period s without production (in capacity units)

$$\kappa_{ri}^{p} q_{ris} + \Psi_{ris} \leq K_{r,t(s)} \cdot z_{ris} \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{2.31}$$

$$q_{ris} \geq m_{ri} \left(z_{ris} - z_{ri,s-1} \right) \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{2.32}$$

$$\sum_{i \in P} z_{ris} = 1 \qquad s = 1, \ldots, S \tag{2.33}$$

$$x_{riks} \geq z_{ri,s-1} + z_{rks} - 1 \qquad r \in R,\ i,k \in P,\ s = 1, \ldots, S \tag{2.34}$$

$$q_{ris}, \Psi_{ris} \geq 0 \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{2.35}$$

$$I_{it} \geq 0 \qquad i \in P \backslash \{0\},\ t = 1, \ldots, T \tag{2.36}$$

$$x_{riks} \geq 0 \qquad r \in R,\ i,k \in P,\ s = 1, \ldots, S \tag{2.37}$$

$$z_{ris} \in \{0, 1\} \qquad r \in R,\ i \in P,\ s = 2, \ldots, S \tag{2.38}$$

The objective (2.28) is to minimize the sum of holding costs, variable production costs, sequence-dependent setup costs and setup state preservation costs. As in the CLSD, a dummy product 0 is introduced for modeling the state where the resource is not set up for any product: If a changeover to this product occurs, this means that the previous setup state for another product gets lost and the resource is not set up for any real product. The inventory balance equations are given by (2.29). The constrained capacity of each production resource is modeled by (2.30): It ensures that the capacity consumption by production activities and sequence-dependent setup times never exceeds the available capacity in a period. The time during which the setup state is preserved without production is captured explicitly by the Ψ_{ris} variables because setup state preservation incurs costs in the GLSP. This is also the reason why the capacity constraint is an equation instead of an inequality as in the CLSD. Equation (2.31) enforces that production of product i on a resource r only takes place in micro-period s if the resource was already set up for i at the end of micro-period $s - 1$ or a changeover to i is performed in s. Minimum production quantities after the changeover to a product are enforced by (2.32). Equation (2.33) means that exactly one product is set up on the resource at the end of each micro-period $s - 1$ and thus at the beginning of each micro-period s. Equation (2.34) ensures that the changeover variable x_{riks} becomes one if i was set up at the end of the previous micro-period $s - 1$ and k is set up at the end of s, which implies that a changeover must have been performed. Equations (2.35)–(2.38) define the variable domains. The variables x_{riks} do not need to be defined as binary variables because their value will always be 0 or 1 in an optimal solution and should as well be binary in every good solution. This is due to (2.37) and their nonnegative coefficients in the objective.

2.1.6 The Multi-Level Capacitated Lot-Sizing Problem

The *multi-level capacitated lot-sizing problem (MLCLSP)* (see, e.g., Stadtler, 1996; Buschkühl et al., 2008) is an extension of the CLSP by multiple production levels.

Its assumptions can be summarized as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.
- Single-level time structure.
- Finite time horizon with T periods.
- An arbitrary number of products can be set up in each period, provided that their setup times fit into the capacity (big bucket model).
- No scheduling (sequencing) of products within periods (resulting from the model assumptions, all permutations of the products' lots within a period result in the same objective value).
- Time-varying demand for products that has to be satisfied at the end of each period.
- Demand always refers to exactly specified products (no substitution).
- Multiple capacitated production resources with finite speed, multiple products may share a resource.
- No parallel (alternative) resources.
- Capacity consumption per unit produced differ among products.
- No setup carry-over.
- Multiple production levels, Gozinto factors are given (units of direct predecessor product required per unit of successor product).
- Predecessor and successor products may share the same resources.
- Time-invariant sequence-independent setup costs and times.
- Setups have to be completed within a single period.
- Time-invariant linear holding costs.
- Time-invariant production costs that are thus irrelevant for finding the optimal solution.
- Lead times between production stages, minimum of one period.
- All occurring demand has to be fulfilled immediately (no relaxation).
- Initial inventories.
- Unlimited work-in-progress buffers for intermediate goods.
- Cost minimization objective.
- Continuous variables for lot-sizes.

Using the CLSP notation given in Table 2.2 and the additional symbols introduced in Table 2.5, the MLCLSP can be formulated as given below. Note that setup

Table 2.5 Notations for MLCLSP

Symbol	Definition
Indices and sets	
$P_i^s \subseteq P$	Set of successor products of i
Parameters	
g_{ik}	Number of units of product i required for producing one unit of its successor product k (Gozinto factor)
$l_i \geq 1$	Lead time of product i (time unit: *number of periods*)
st_i	Production setup time for product i

times are measured in capacity units, whereas lead times are measured in the number of periods.

$$\text{Minimize } F(q, x, I) = \sum_{i \in P} \sum_{t=1}^{T} (h_i I_{it} + f_i x_{it}) \tag{2.39}$$

subject to

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} - \sum_{k \in P_i^s} g_{ik} q_{k,t+l_i} \qquad i \in P,\ t = 1, \ldots, T - l_i \tag{2.40}$$

$$I_{it} = I_{i,t-1} + q_{it} - d_{it} \qquad i \in P,\ t = T - l_i + 1, \ldots, T \tag{2.41}$$

$$\sum_{i \in P_r} \left(\kappa_i^p q_{it} + st_i x_{it}\right) \leq K_{rt} \qquad r \in R,\ t = 1, \ldots, T \tag{2.42}$$

$$q_{it} \leq M \cdot x_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{2.43}$$

$$q_{it}, I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{2.44}$$

$$x_{it} \in \{0, 1\} \qquad i \in P,\ t = 1, \ldots, T \tag{2.45}$$

The objective (2.39) is to minimize the sum of holding and setup costs. The inventory balance equations are given by (2.40) and (2.41): Due to the multi-level production structure, they include the consumption of units of a product by successor products with the respective Gozinto factors. Equation (2.41) is required because there is no inventory consumption caused by production of successor products in the last l_i periods due to the lead time between production stages. The constrained capacity of each production resource is modeled by (2.42): It ensures that the capacity consumption by setup and production activities never exceeds the available capacity in a period. Equation (2.43) enforces that the setup variable x_{it} is one if production of product i takes place in period t, i.e., if $q_{it} > 0$. Equations (2.44)–(2.45) define the variable domains.

2.1.7 The General Lot-Sizing and Scheduling Problem for Multiple Production Stages

The *general lot-sizing and scheduling problem for multiple production stages (GLSPMS)* (Meyr, 2004a) is a multi-level lot-sizing and scheduling problem with a two-level time structure. It enhances the GLSP and is intended for a flow shop production environment, where each intermediate or finished good and each resource belongs to exactly one production stage and the stages have a unique order. The assumptions of its most general version with production quantity splitting and setup time splitting can be summarized as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.
- No lead times.

- Two-level time structure.
- Finite time horizon with T macro-periods and S micro-periods.
- Each macro-period contains a predetermined set of micro-periods, the number of micro-periods can differ among macro-periods.
- The beginning and duration of each macro-period are fixed.
- Micro-period beginnings and lengths are flexible, apart from some micro-periods with explicitly fixed beginnings; these fixed beginnings are required for micro-periods that are the first micro-period in a macro-period and for modeling predetermined fixed exogenous downtime of resources.
- Only a single changeover is possible between each pair of micro-periods.
- Scheduling (sequencing) of products within macro-periods.
- Time-varying demand for products that has to be satisfied at the end of each macro-period.
- Demand always refers to exactly specified products (no substitution).
- Multiple capacitated production resources with finite speed.
- Multiple production levels, Gozinto factors are given (units of direct predecessor product required per unit of successor product).
- Heterogeneous parallel resources on each production level.
- Resources may be shared between production levels, i.e., between predecessor and successor products.
- Capacity consumption per unit produced differ among products.
- Setup carry-over is possible.
- Time-invariant sequence-dependent setup costs and setup times (these consume capacity).
- Changeovers are started towards the end of a micro-period and may continue into the subsequent micro-period (setup time splitting).
- Time-invariant linear holding costs that are incurred for inventory at the end of each macro-period.
- Time-invariant production costs that differ among the parallel resources and thus have to be included.
- Time-invariant costs for preservation of setup state for a product on a certain resource (per time unit).
- All occurring demand has to be fulfilled immediately (no relaxation).
- Initial inventories.
- Quantities produced in a micro-period can be used for satisfying (successor) demand in the same micro-period (after they have been produced).
- Limited work-in-progress buffers for intermediate goods within micro-periods, thus production on a predecessor stage is possible while a setup is performed on the successor stage (production quantity splitting is necessary to model this and the previous aspect).
- Minimum production quantity after changeover.
- Minimum time for resource staying in idle state (without production).
- Initial setup state given for each resource.
- Cost minimization objective.
- Continuous variables for production quantities.

Table 2.6 Notations for GLSPMS

Symbol	Definition
Indices and sets	
$e(t)$	Last micro-period of macro-period t
Φ	Set of micro-periods
Φ^{fix}	Set of micro-periods with fixed beginnings
Φ_r^o	Set of micro-periods during which production on r is forbidden
Φ^l	Set of micro-periods that are the last micro-period in a macro-period
Parameters	
d_{is}	Demand for product i in micro-period s ($= 0$ for all $s \in \Phi \backslash \Phi^l$)
I_i^{max}	Upper inventory limit for product i
$\overline{w}_s$	Fixed beginning for micro-period s
$I_{ri}^{w,max}$	Upper inventory limit for units of product i produced on resource r that are not consumed before the next micro-period
$z_{ri1} = x_{rii1}$	Binary parameter that indicates whether resource r is set up for i at the beginning of the first micro-period
Variables	
x_{riks}	Binary variable that indicates whether a changeover from product k to product i is performed on resource r starting at the end of micro-period $s-1$ and continuing into micro-period s
z_{ris}	Binary variable that indicates whether resource r is already set up for product i at the beginning of micro-period s or a changeover to it started in $s-1$ is completed in s
I_{is}	Inventory of product i at the end of *micro*-period s
q_{ris}^0	Quantity of product i produced on r in micro-period s available for consumption in the same period
q_{ris}^{+1}	Quantity of product i produced on r in micro-period s available for consumption in the next period $s+1$
w_s	Beginning of micro-period s (on continuous time axis)
y_{rs}^b	Setup time consumed on resource r at beginning of micro-period s
y_{rs}^e	Setup time consumed on resource r at end of micro-period s

Using the GLSP notation given in Table 2.4 and the additional symbols introduced for MLCLSP and GLSPMS in Tables 2.5 and 2.6, the GLSPMS can be formulated as given in the following. Note that in contrast to the GLSP that uses inventory variables I_{it} on the macro-period level, the GLSPMS keeps track of the inventory on the micro-period level and hence requires I_{is} inventory variables for each micro-period. Also, the x_{riks} variables have a different meaning than in the GLSP: They indicate that the changeover from product i to k is already started towards the end of the previous micro-period $s-1$, not exactly at the beginning of s. In addition, note that we assume for the sake of generality that setup state preservation costs may differ among the products. Hence, we added the index i to $\overline{p}_{ris}$, in contrast to $\overline{p}_{rs}$ defined in the original formulation of Meyr (2004a).

$$\begin{aligned}\text{Minimize } & F(q, x, z, I, \Psi, q^0, q^{+1}, w, y^b, y^e) \\ &= \sum_{i \in P\setminus\{0\}} \sum_{t=1}^{T} h_i \left(I_{i,e(t)} + \sum_{r \in R} q^{+1}_{ri,e(t)} \right) \\ &+ \sum_{i \in P} \sum_{r \in R} \sum_{s=1}^{S} \left(p_{ris} q_{ris} + \overline{p}_{ris} \Psi_{ris} + \sum_{k \in P} f_{rik} x_{riks} \right) \end{aligned} \tag{2.46}$$

subject to

$$\begin{aligned} I_{is} &= I_{i,s-1} + \sum_{r \in R} \left(q^0_{ris} + q^{+1}_{ri,s-1} \right) \\ &- d_{is} - \sum_{r \in R} \sum_{k \in P_i^s} g_{ik} q_{rks} \; i \in P\setminus\{0\},\; s = 1, \ldots, S \end{aligned} \tag{2.47}$$

$$I_{is} \le I_i^{max} \qquad i \in P\setminus\{0\},\; s = 1, \ldots, S \tag{2.48}$$

$$w_s = \overline{w}_s \qquad s \in \Phi^{fix} \tag{2.49}$$

$$\begin{aligned} & w_{s+1} - w_s \\ & \quad = y^b_{rs} + \sum_{i \in P} \left(\Psi_{ris} + \kappa^p_{ri} q_{ris} \right) + y^e_{rs} \end{aligned} \qquad r \in R,\; s = 1, \ldots, S \tag{2.50}$$

$$q_{ris} = 0 \qquad r \in R,\; i \in P,\; s \in \Phi^o_r \tag{2.51}$$

$$q_{ris} \le M \cdot z_{ris} \qquad r \in R,\; i \in P_r,\; s = 1, \ldots, S \tag{2.52}$$

$$q_{ris} \ge m_{ri} \sum_{k \in P\setminus\{i\}} x_{rkis} \qquad r \in R,\; i \in P_r,\; s = 1, \ldots, S \tag{2.53}$$

$$\sum_{i \in P_r} z_{ris} = 1 \qquad r \in R,\; s = 1, \ldots, S \tag{2.54}$$

$$x_{riks} \ge z_{ri,s-1} + z_{rks} - 1 \qquad r \in R,\; i, k \in P_r,\; s = 2, \ldots, S \tag{2.55}$$

$$y^e_{r,s-1} + y^b_{rs} = \sum_{i,k \in P_r} st_{rik} x_{riks} \qquad r \in R,\; s = 2, \ldots, S \tag{2.56}$$

$$q^0_{ris} + q^{+1}_{ris} = q_{ris} \qquad r \in R,\; i \in P_r,\; s = 1, \ldots, S \tag{2.57}$$

$$q^{+1}_{ris} \le I^{w,max}_{ri} \qquad r \in R,\; i \in P_r,\; s = 1, \ldots, S \tag{2.58}$$

$$\sum_{i,k \in P_r} x_{riks} = 1 \qquad r \in R,\; s = 2, \ldots, S \tag{2.59}$$

$$\begin{aligned} & w_s - w_{s-1} \ge y^b_{r_1,s-1} \\ & \quad + \kappa^p_{r_1 i} q^0_{r_1,i,s-1} + y^e_{r_2,s-1} \end{aligned} \qquad \begin{aligned} & s \ge 2,\; r_1, r_2 \in R,\; i \in P_{r_1},\; k \in P_{r_2}, \\ & g_{ik} > 0,\; g_{ik} \kappa^p_{r_1 i} > \kappa^p_{r_2 k} \end{aligned} \tag{2.60}$$

$$\begin{aligned} & w_s - w_{s-1} \ge y^b_{r_1,s-1} \\ & + \kappa^p_{r_2 k} q_{r_2,k,s-1} + y^e_{r_2,s-1} \end{aligned} \qquad \begin{aligned} & s \ge 2,\; r_1, r_2 \in R,\; i \in P_{r_1},\; k \in P_{r_2}, \\ & g_{ik} > 0,\; g_{ik} \kappa^p_{r_1 i} < \kappa^p_{r_2 k} \end{aligned} \tag{2.61}$$

$$y^b_{r1} = y^e_{rS} = 0 \qquad r \in R \tag{2.62}$$

$$q_{ris}, \Psi_{ris} \geq 0 \qquad r \in R,\ i \in P_r,\ s = 1, \ldots, S \tag{2.63}$$

$$I_{is} \geq 0 \qquad i \in P \backslash \{0\},\ s = 1, \ldots, S \tag{2.64}$$

$$x_{riks} \geq 0 \qquad r \in R,\ i,k \in P_r,\ s = 2, \ldots, S \tag{2.65}$$

$$z_{ris} \in \{0, 1\} \qquad r \in R,\ i \in P_r,\ s = 2, \ldots, S \tag{2.66}$$

$$w_s \geq 0 \qquad s = 1, \ldots, S + 1 \tag{2.67}$$

$$q^0_{ris}, q^{+1}_{ris} \geq 0 \qquad r \in R,\ i \in P_r,\ s = 1, \ldots, S \tag{2.68}$$

$$y^b_{rs}, y^e_{rs} \geq 0 \qquad r \in R,\ s = 1, \ldots, S \tag{2.69}$$

The cost minimization objective (2.46) is composed of holding costs, variable production costs, sequence-dependent setup costs and setup state preservation costs. As in the CLSD and GLSP, a dummy product 0 is introduced for modeling the state where the resource is not set up for any product: If a changeover to this product occurs, this means that the previous setup state for another product gets lost and the resource is not set up for any real product. The inventory balance equations are given by (2.47): Due to the multi-level production structure, they include the consumption of units of a product by successor products with the respective Gozinto factors. In addition, the inventory increase is split in a part q^0_{ris} that originates from production in the same micro-period s and a part that originates from production in the previous micro-period $s - 1$. This production quantity splitting is necessary to ensure that the usage of units produced in a micro-period by a successor product in the same micro-period is always temporally feasible in the real-world application, i.e., no units are "consumed before they have been produced". Note that q^{+1}_{ri0} has to be defined as a constant with value 0. Equation (2.48) limits the inventory of each product to a certain maximum level. The constraints (2.49) fix the beginnings of a subset of the micro-periods to certain points in time.

The constrained capacity of each production resource is modeled by (2.50): It ensures that the capacity consumption by production activities and sequence-dependent setup times never exceeds the effective duration of each micro-period. This duration of a micro-period s is the difference $w_{s+1} - w_s$ of the beginning of the subsequent period and its own beginning. It is necessary to introduce w_{S+1} because the duration of the last micro-period S has to be fixed as well. As setup activities overlap micro-period boundaries in the GLSPMS, setup time may lie both at the beginning and the end of each micro-period (y^b_{rs} and y^e_{rs}, respectively). The time during which the setup state is preserved without production is captured explicitly by the Ψ_{ris} variables because setup state preservation incurs costs in the GLSPMS. Hence, the capacity constraint is an equation as in the GLSP. Equation (2.51) ensures that no production takes place on r in a micro-period s during which production is forbidden. Equation (2.52) enforces that production of product i on a resource r only takes place in micro-period s if the resource is already set up for i at the beginning of s or a changeover to i is performed in s. Minimum production quantities after the changeover to a product are enforced by (2.53).

Equation (2.54) means that exactly one product is set up on the resource at the beginning of each micro-period s. Equation (2.55) ensures that the changeover variable x_{riks} becomes one if i was set up in micro-period $s-1$ and k is set up in s, which implies that a changeover must have been performed. Equation (2.56) splits up the setup time of a setup activity that starts in $s-1$ and continues into s, and thus overlaps the micro-period boundaries: A part $y^e_{r,s-1}$ of the setup time is scheduled at the end of $s-1$, the remaining part y^b_{rs} at the beginning of s. Note that all y^b_{r1} and y^e_{rS} should be fixed to 0. The splitting of production quantities into a part q^0_{ris} that can be used in the same micro-period s where the production takes place and another part q^{+1}_{ris} that cannot be used before the next period $s+1$ is implemented by (2.57). Equation (2.58) specifies upper limits for the work-in-progress buffers after resources between successive production stages. Equation (2.59) makes sure that exactly one changeover occurs in each micro-period. Note that if $x_{riis}=1$, this denotes that product i remains set up. If we omitted (2.59) in the model, it would still be fulfilled in every optimal solution, but not in every integer feasible solution to the GLSPMS. Equations (2.63)–(2.69) define the variable domains.

Examples that show the necessity of (2.60) and (2.61) are given by Meyr (2004a): Certain cases exist where feasible solutions to the GLSPMS with these two constraints omitted would be inapplicable to the real-world problem: If the linkage between production stages was ignored, temporal constraints could be violated within micro-periods though the production and setups on r_1 and r_2 seem capacity-feasible.

Figure 2.12 illustrates why (2.60) is required: The predecessor i is assumed to be slower than its successor k. In this context, the proposition that a predecessor i is slower than its successor k denotes that $g_{ik}\kappa^p_{r_1 i} > \kappa^p_{r_2 k}$, i.e., the production time

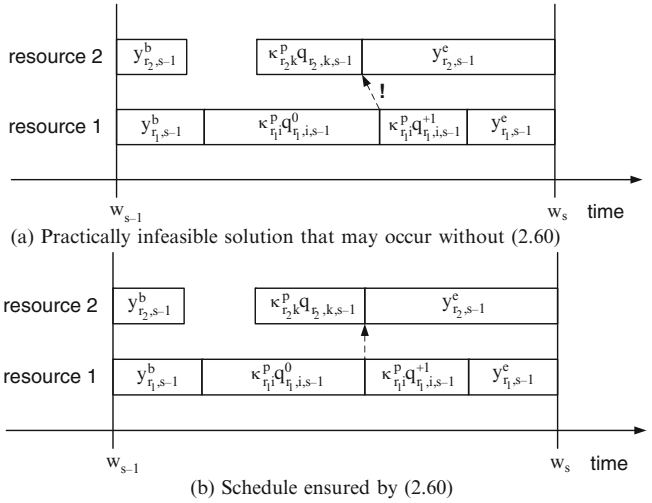


Fig. 2.12 Example showing necessity of (2.60), adapted from Meyr (2004a)

for producing the number of units of i required for one unit of k is longer than the production time for one unit of k (excluding the production time for predecessors). Without (2.60), the schedule shown in Fig. 2.12a could be contained in a "feasible" solution. In the schedule, the split changeover time $y^e_{r_2,s-1}$ at the end of $s-1$ is so long that the production of the faster successor k would have to stop before the production of the required quantity $q^0_{r_1 i,s-1}$ of its predecessor i has been completed, assuming that there are no stocks of i at the beginning of $s-1$. Thus, it might not be possible to implement the resulting schedule due to its violation of temporal constraints.

Equation (2.60) ensures that the changeover to and production of a slower predecessor product i on resource r_1 and the split changeover from a faster successor product k on resource r_2 (on the next production stage) to another product all fit into the duration $w_s - w_{s-1}$ of a micro-period $s-1$.[5] Here, only the production $q^0_{r_1 i,s-1}$ of i that can be used in $s-1$ is considered. A schedule with the property enforced by (2.60) is shown in Fig. 2.12b.

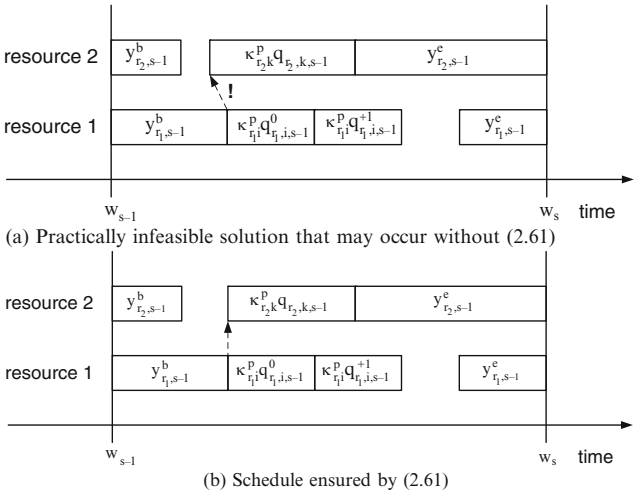
The necessity of (2.61) is exemplified by Fig. 2.13: The predecessor product i is now assumed to be *faster* than its successor product k. Without (2.61), the schedule shown in Fig. 2.13a could be contained in a "feasible" solution. Due to the slowness of the successor product k and the duration $y^e_{r_2,s-1}$ of the split changeover on its resource, one would have to start the production of k before the production of its predecessor i has started, which is impossible if we again assume that there are no stocks of i at the beginning of $s-1$. Thus, even with (2.60) added to the model, it might not be possible to implement the resulting schedule as it might still violate temporal constraints.

Hence, (2.61) is required. It enforces that the split changeover to a faster predecessor product i on resource r_1 and the production of a slower successor product k on resource r_2 (on the next production stage) as well as the split changeover from k to another product all fit into the duration $w_s - w_{s-1}$ of a micro-period $s-1$.[6]

---

[5] The constraint (2.60) is slightly more restrictive than necessary. It always ensures temporal feasibility of a solution, but excludes some feasible solutions (F. Seeanner 2009, personal communication): (2.60) is always enforced, no matter whether both the slow predecessor i and its successor k are actually produced on resources r_1 and r_2, respectively, in $s-1$. If i is not produced on r_1 in $s-1$, (2.60) still enforces that $w_s - w_{s-1} \geq y^b_{r_1,s-1} + y^e_{r_2,s-1}$, which unnecessarily limits the duration of split setups on r_1 and r_2 involving other products. Considering the case that k is not produced on r_2 in $s-1$, (2.60) is still enforced although the solution shown in Fig. 2.12a would not violate temporal constraints. Equation (2.60) can be formulated in a less restrictive way by introducing setup splitting variables for each possible changeover from one product h to another product j, i.e., by introducing additional indices h and j for the y^b and y^e variables. These variables y^b_{rhjs} and y^e_{rhjs} denote the setup time consumed by a changeover from product h to j on resource r at the beginning and end of micro-period s, respectively (F. Seeanner 2009, personal communication). With these variables, one can reformulate (2.60) so that only relevant durations of changeovers involving the products i and k are included.

[6] Just as (2.60), (2.61) is more restrictive than necessary and can be reformulated in a less restrictive way using the variables y^b_{riks} and y^e_{riks} mentioned in footnote 5 (F. Seeanner 2009, personal communication). A schedule with the property enforced by (2.61) is shown in Fig. 2.13b.



(a) Practically infeasible solution that may occur without (2.61)

(b) Schedule ensured by (2.61)

Fig. 2.13 Example showing necessity of (2.61) adapted from Meyr (2004a)

2.1.8 Comparison of Models

The CLSP and CLSD can be mapped to the GLSP by introducing additional constraints (Meyr, 1999, p. 82f.). The same holds true for the Proportional Lot-Sizing and Scheduling Problem (PLSP) (Meyr, 1999, p. 83). Also, the Discrete Lot-Sizing and Scheduling Problem (DLSP) and the Continuous Lot-Sizing and Scheduling Problem can immediately be mapped to special cases of the GLSP (Meyr, 1999, p. 82f.). The GLSP itself could be mapped to a single production level version of the GLSPMS where no production quantity splitting and setup time splitting are allowed.

Comparing the (multi-resource version of) CLSD and the GLSP, a main difference is that the CLSD cannot deal with the case where multiple setups of the same product within a (macro-)period would be efficient due to a violation of the triangle inequality in the problem instances (Meyr, 1999, p. 76): It only allows a single setup of each product in a period. In contrast, the GLSP allows multiple setups of a product within a macro-period.

The MLCLSP and GLSPMS compare as follows: First, the MLCLSP does not consider sequence-dependent setups and setup carry-overs. However, the MLCLSPL (Suerie and Stadtler, 2003) includes setup carry-overs, and one could also develop a multi-level formulation of the CLSD. Second, a critical assumption of the MLCLSP is that there is a minimum lead time of an entire period ($l_i \geq 1$) between production stages to ensure feasibility of the resulting solution. This assumption is made solely for modeling reasons and might result in production plans that are suboptimal from a practical point of view. The requirement that $l_i \geq 1$ can be explained as follows: When setting $l_i = 0$, the entire production q_{it} of a predecessor product i in a period t could immediately be used for satisfying the demand $g_{ik}q_{kt}$ induced by a

successor product $k \in P_i^s$ in the same period. This might not be feasible in practice because the units of i have to be produced within t before they can be used by k later on in that period. The GLSPMS handles this difficulty by introducing the production quantity splitting variables q_{ris}^0 and q_{ris}^{+1} as well as the constraints (2.60) and (2.61). Thus, no minimum lead time between production stages is required. Third, the GLSPMS allows for modeling exogenous resource downtime, e.g., due to predetermined scheduled machine maintenance on resource r: This can be included by fixing the beginnings of two adjacent micro-periods s and $s+1$ to values $\overline{w}_s$ and $\overline{w}_{s+1}$, adding them to the set Φ^{fix}, and adding s to the set Φ_r^0.

2.2 Solution Techniques for Dynamic Lot-Sizing

In this section, we provide a high-level overview of solution methods available for solving deterministic dynamic lot-sizing problems modeled as mixed-integer linear programs (MILP).

2.2.1 Overview

Generally, *exact algorithms* that yield optimal solutions and guarantee their optimality can be distinguished from *heuristics* that generate feasible, preferably good solutions. Oftentimes, the running times of exact algorithms are too high to use these algorithms, as the user of an algorithm wants to obtain a good solution after a limited and ideally short amount of time. Hence, heuristics are preferable for many real-world problems because they frequently yield good solutions within a short computation time. Before even starting to solve an MILP formulation of a lot-sizing problem with an algorithm, one often tries to *reformulate* the problem so that it can be solved more efficiently.

Reformulations of lot-sizing problems are often based on analogies to other MILP problems or on analogies of their LP relaxations to certain other linear programming models, e.g., Warehouse Location Problems, Shortest-Path Problems, or Minimum-Cost Network Flow Problems. Another way of reformulating a problem is to add certain *valid inequalities* to the mathematical model a priori. These valid inequalities cut off certain non-integer solutions from the feasible solution space, and ideally provide the convex hull. Their purpose is also to sharpen the bounds obtained from solving the LP relaxation of the problem. However, when adding certain groups of valid inequalities to a lot-sizing model a priori, one should trade off the exclusion of non-integer solutions and improvement of the bounds against the increased model size: By adding additional valid inequalities, the number of constraints increases, and thus the time for solving the LP relaxation presumably increases. Also, *decompositions* of a problem can be used for solving a problem more efficiently, e.g., the *Dantzig–Wolfe decomposition* (see, e.g., Degraeve and Jans, 2007). In addition, in some approaches the solution space is reduced by

considering only solutions that fulfill certain properties (see, e.g., Haase and Kimms, 2000).

Typical exact algorithms for MILP are:

- Pure *Branch&Bound (B&B)* algorithms
- *Branch&Cut (B&C)* algorithms that combine B&B with the generation of cutting planes
- *Branch&Price (B&P)* algorithms that combine B&B with column generation (Barnhart et al., 1998)[7]
- *Branch&Cut&Price (B&C&P)* algorithms that combine B&B with the generation of cutting planes and column generation (Ralphs et al., 2001)

Heuristic algorithms can be categorized into *construction heuristics* that generate initial feasible solutions and *improvement heuristics* that start from a feasible solution and try to find better solutions. However, the terminological boundary between construction and improvement heuristics is blurred because improvement heuristics could also start from a solution that is feasible for a relaxation of the original lot-sizing problem (e.g., with backlogging or lost sales) but infeasible for the original problem. Heuristic approaches to lot-sizing MILP models can be classified into the following categories (also see Silver, 2004; Jans and Degraeve, 2007; Buschkühl et al., 2008):

- "Pure" *constructive heuristics* that generate feasible solutions
- *Exact algorithms* that are *truncated* after a certain time before reaching the optimum
- "Pure" *meta-heuristics* (Jans and Degraeve, 2007) (e.g., Tabu Search, Simulated Annealing, Genetic Algorithms, Scatter Search, Particle Swarm Optimization, Ant Colony Optimization, Threshold Accepting as well as countless hybrids of these algorithms, see, e.g., Caserta and Rico, 2009)
- *Decomposition and aggregation heuristics* (Buschkühl et al., 2008) (e.g., time-, product-, or resource-based decomposition, product- or resource-based aggregation)
- *Lagrangean (relaxation or decomposition) heuristics* (Fisher, 1981), see, e.g., Tempelmeier and Derstroff (1996); Toledo and Armentano (2006)[8]
- *Combinations of (meta-) heuristics and exact algorithms*
  (Puchinger and Raidl, 2005), e.g., Relax&Fix heuristics (Stadtler, 2003) or combinations of local search heuristics with a minimum cost network flow solver (Meyr, 2000)

In the following subsections, we briefly review selected solution approaches for lot-sizing: Valid inequalities are considered in Sect. 2.2.2, reformulations in Sect. 2.2.3, combinations of exact algorithms and heuristics in Sect. 2.2.4, and selected other approaches for lot-sizing with sequence-dependent setups in Sect. 2.2.5.

---

[7] See Desrosiers and Lübbecke (2005); Huisman et al. (2005); Lübbecke and Desrosiers (2005); Ralphs et al. (2001, 2003); Vanderbeck (2003, 2005); Wilhelm (2001) for additional literature.

[8] For additional literature see, e.g., Diaby et al. (1992a,b); Thizy and van Wassenhove (1985).

2.2.2 Valid Inequalities

Valid inequalities (also: *cuts* or *cutting planes*) are inequalities that are valid for all *integer*-feasible solutions of an MILP. They are often violated by solutions feasible for the LP relaxation that do not fulfill the integrality constraints for some integer and binary variables. Cuts are added to improve the bounds yielded from the LP relaxations, which helps to prune B&B nodes, and ideally to effect that the optimal solution of a node's LP relaxation becomes integer-feasible.

There are three common approaches for using valid inequalities when practically solving lot-sizing problems:

1. Add valid inequalities to the model formulation a priori even before starting an algorithm (standard MIP solver or specialized algorithm). This results in a larger (possibly huge) model formulation whose LP relaxation usually yields tighter bounds.
2. Add valid inequalities to a so-called pool of user cuts, a feature that is offered by some standard MIP solvers (e.g., CPLEX®). The user can insert cuts of which he expects that they speed up the B&C procedure into a set ("pool"). The solver uses these cuts during B&C by adding them to the LP relaxations of B&B nodes where they are violated. In this approach, the solver might spend a significant amount of time on checking whether user cuts are violated without gaining any advantage from the cuts, especially if the cut pool contains a large number of "weak" valid inequalities: As the solver uses no specialized *separation heuristic* for the cuts, a "brute-force" enumeration of all cuts in the cut pool might be necessary.
3. Implement a specialized separation heuristic for each group of valid inequalities and integrate such separation heuristics into a B&C algorithm: This heuristic is applied at each node of the B&B tree to find cuts belonging to a certain group of valid inequalities that are violated by the optimal solution of the node's LP relaxation. This approach critically depends on the computation time required for the separation heuristic and the number and "strength" of the cut it returns.

A general difficulty is how to select groups of valid inequalities that could be effective for a certain lot-sizing problem. Since no structured approach for this selection exists, empirical analysis of algorithm performance on instances of the problem class to be solved is often the only choice. Also, as some groups of valid inequalities contain an exponential number of cuts, it is not practicable to add all of these in approaches (1) and (2). A vast amount of literature on valid inequalities for lot-sizing models has been published (see, e.g., Pochet and Wolsey, 1991, 2006; Marchand et al., 2002; Belvaux and Wolsey, 2000, 2001; Pochet, 2001; Wolsey, 1997, 1998, 2003a,b; Pochet et al., 2005).

As a comprehensive overview of valid inequalities for lot-sizing models would go beyond the scope of this work, we only give examples of valid inequalities for the

uncapacitated WWP: The so-called (l, S)-cuts can be formulated as follows (Pochet and Wolsey, 2006, p. 218):

$$\sum_{t \in S} q_t \leq \sum_{t \in S} \sum_{t'=t}^{l} d_{t'} x_t + I_l \qquad l = 1, \ldots, T,\ S \subseteq \{1, \ldots, l\} \tag{2.70}$$

These valid inequalities can easily be generalized to uncapacitated multi-product lot-sizing problems. Taking the subset of these cuts with $l = 1, \ldots, T$ and $S = \{l\}$ results in:

$$q_t \leq d_t x_t + I_t \qquad t = 1, \ldots, T \tag{2.71}$$

By inserting the inventory balance equation (2.4) for I_t, we see that (2.71) is equivalent to the setup/inventory carryover cuts

$$I_{t-1} \geq d_t (1 - x_t) \qquad t = 1, \ldots, T \tag{2.72}$$

In Sect. 5.3.3, we will define valid inequalities for uncapacitated lot-sizing with substitutions that are based on (2.70)–(2.72). Another group of valid inequalities for the WWP that contains (2.72) as a special case is (Vyve and Wolsey, 2006):

$$I_{t-1} \geq \sum_{\tau=t}^{k} d_\tau \left(1 - \sum_{\upsilon=t}^{\tau} x_\upsilon\right) \qquad t = 1, \ldots, T, k = t, \ldots, T \tag{2.73}$$

2.2.3 Reformulations

Reformulations of lot-sizing problems are frequently based on analogies to other MILP models, e.g., the Simple Plant Location (SPL) problem or the network flow formulation of Shortest Route (SR) [also: Shortest Path (SP)] problems. Other reformulations, e.g., substitute inventory variables in order to eliminate minimum inventory levels from the model, or introduce echelon stock variables in multi-level models (Belvaux and Wolsey, 2001). In addition, also models to which valid inequalities are added a priori are termed reformulations. As one example of a reformulation, we present an SPL-based CLSP reformulation using the notations contained in Tables 2.2 and 2.7:

$$\text{Minimize } F(x, y) = \sum_{t_s=1}^{T} \sum_{i \in P} \left(f_{it_s} x_{it_s} + \sum_{t_d=t_s}^{T} c_{it_s t_d} y_{it_s t_d} \right) \tag{2.74}$$

Table 2.7 Additional notations for SPL-based reformulation of CLSP

Symbol	Definition
Parameters	
$c_{it_st_d}$	Costs per unit to satisfy demand for product i in period t_d from a production lot in period t_s
Variables	
$y_{it_st_d}$	Demand quantity for product i in period t_d that is satisfied from a production lot in period t_s

subject to

$$\begin{aligned}
&\sum_{t_s=1}^{t_d} y_{it_st_d} = d_{it_d} && i \in P,\ t_d = 1,\ldots,T \\
&\sum_{t_d=t_s}^{T} y_{it_st_d} \le \sum_{\tau=t_s}^{T} d_{i\tau} \cdot x_{it_s} && i \in P,\ t_s = 1,\ldots,T \\
&y_{it_st_d} \le d_{it_d} \cdot x_{it_s} && i \in P,\ 1 \le t_s \le t_d \le T \\
&\sum_{i\in P} \kappa_i^p \sum_{t_d=t_s}^{T} y_{it_st_d} \le K_{t_s} && t_s = 1,\ldots,T \\
&x_{it_s} \in \{0,1\} && i \in P,\ t_s = 1,\ldots,T \\
&y_{it_st_d} \in \{0,1\} && i \in P,\ 1 \le t_s \le t_d \le T
\end{aligned} \tag{2.75}$$

In this reformulation, transportation variables $y_{it_st_d}$ are introduced that denote the demand quantity for product i in period t_d satisfied from a production lot in t_s. The costs $c_{it_st_d}$ associated with these variables include the corresponding variable production costs and holding costs:

$$c_{it_st_d} = p_{it_s} + (t_d - t_s)\, h_i \tag{2.76}$$

The lot-size variables q_{it} and inventory variables I_{it} are no longer necessary. Equation (2.75) ensures that the demand quantity for each product in each period is satisfied. The model includes both aggregated and disaggregated setup forcing constraints: The aggregated variant (2.75) forces the setup variable x_{it_s} for product i in period t_s to be 1 in the optimal solution of the LP relaxation if all demand in t_s and consequent periods that could be satisfied from the lot is fulfilled by it. The disaggregated variant (2.75) is redundant to (2.75), but tightens the lower bound obtained by the LP relaxation: It ensures that if the entire demand for product i in a period t_d is fulfilled using only units produced in period t_s, the corresponding setup variable x_{it_s} becomes 1 in the optimal solution of the LP relaxation. The meaning of (2.75) equals the capacity constraint (2.12) in the CLSP, but is now formulated using the $y_{it_st_d}$ variables.

Eppen and Martin (1987) develop a shortest-route reformulation for a CLSP variant. Tempelmeier and Helber (1994) extend this reformulation to the MLCLSP. Stadtler (1996) compares the performance of several formulations for the MLCLSP with initial inventories and overtime, including an SPL-based reformulation and a shortest-route reformulation. *Approximate extended formulations* (Vyve and Wolsey, 2006; Stadtler, 1997) only add a subset of valid inequalities of a certain class to an extended model formulation a priori. By varying a control parameter, the user can determine the size of this subset and thus find a good tradeoff between the size of the approximate extended formulation and the strength of its LP relaxation. Denizel et al. (2008) show the linear equivalence of an SPL and SP reformulation of the CLSP with setup times: They prove that every feasible solution to the LP relaxation of the SPL reformulation is also feasible for the LP relaxation of the SP reformulation and has the same objective value, and vice versa. For further literature on reformulations, see, e.g., Belvaux and Wolsey (2001); Pochet (2001); Wolsey (2003a,b); Pochet et al. (2005); Pochet and Wolsey (2006).

2.2.4 Combinations of Heuristics and Exact Algorithms

Puchinger and Raidl (2005) classify combinations of (meta-)heuristics and exact algorithms as shown in Fig. 2.14.

Combinations of exact algorithms and metaheuristics are termed collaborative if the algorithms exchange information, but are not part of each other. In this case, the exact algorithm and metaheuristic are either executed one after another, in parallel, or intertwined. In integrative combinations of exact algorithms and metaheuristics, either an exact algorithm is the "master" algorithm and invokes a metaheuristic or vice versa.

A recently popular combination of exact algorithms and heuristics for lot-sizing are *Relax&Fix (R&F)* [also: *Fix&Relax (F&R)*] *heuristics*. They belong to the class

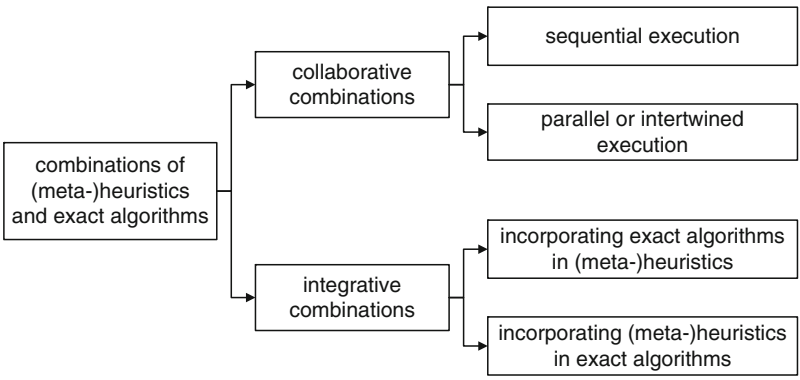


Fig. 2.14 Possible combinations of exact and heuristic algorithms (Puchinger and Raidl, 2005)

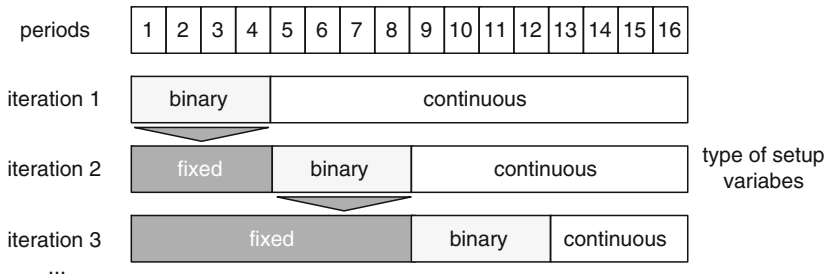


Fig. 2.15 Example – Relax&Fix

of integrative combinations where an exact algorithm (mostly B&C) is integrated into a heuristic: The idea of R&F heuristics is to solve a series of MILP problems in each of which the integrality constraints of some of the binary decision variables of the problem (i.e., a subset of the setup variables) are *relaxed*, i.e., these variables can take arbitrary continuous values between 0 and 1. Thus, R&F heuristics belong to the class of *MIP-based heuristics*. Only the setup variables in a certain time window are treated as binary decision variables.

In the following, we exemplary describe the procedure of a R&F algorithm, which is also visualized in Fig. 2.15: Considering a CLSP with $T = 16$ periods, a (simplified) R&F heuristic that uses a time windows size of four periods would in a first step solve an MILP P_1 where only the setup variables x_{it} with $t \in \{1, 2, 3, 4\}$ are binary and $0 \leq x_{it} \leq 1$ for $t \geq 5$. In the next iteration, a new MILP P_2 is created for which the values x_{it} with $t \in \{1, 2, 3, 4\}$ are *fixed* to the "optimal" values x_{it}^* obtained by solving P_1 to optimality. In P_2, only the x_{it} with $t \in \{5, 6, 7, 8\}$ are binary and $0 \leq x_{it} \leq 1$ for $t \geq 9$. Again, the "optimal" values x_{it}^* obtained from P_2 are used for fixing the values of x_{it} with $t \in \{5, 6, 7, 8\}$. In addition, the variables x_{it} with $t \in \{1, 2, 3, 4\}$ remain fixed to the values obtained from P_1. The *R&F* heuristic proceeds in an analogous way for a problem P_3. After that, P_4 is reached, where all x_{it} with $t \leq 12$ are already fixed and x_{it} with $t \in \{13, 14, 15, 16\}$ binary. When solving this last problem of the series of MILP problems, two cases can occur:

1. A feasible solution is returned in which all setup variables are fixed. This solution is returned as the result of the R&F algorithm.
2. The subproblem (P_4) has no feasible solution. In this case, repair mechanisms could be used that "unfix" setup variables – which had already been fixed in a previous iteration – to find a feasible solution. However, these repair mechanisms might fail.

The time windows of subsequent MILP problems solved during R&F can also overlap, e.g., the first time windows could consist of periods 1, 2, 3 and 4, the second time windows of periods 3, 4, 5 and 6, etc. In this case, only the setup variables that do not overlap with the next time windows are fixed in each of the problems solved sequentially. An example for such overlapping time windows in R&F is shown in

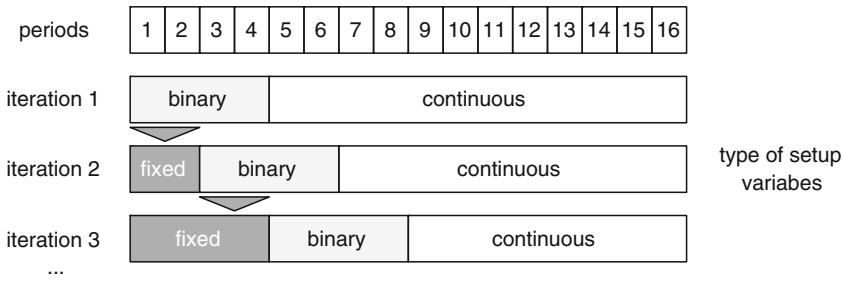


Fig. 2.16 Example – Relax&Fix with overlapping time windows

Fig. 2.16. Note that instead of relaxing the domain of certain setup variables to continuous values between 0 and 1, one could also fix all or some of them to 0 or 1.

Stadtler (2003) describes a R&F heuristic for the MLCLSP. Federgruen et al. (2007) develop a heuristic for a multi-product lot-sizing problem with a joint setup for a single product family incurring setup costs that is a generalization of R&F. The model is a big-bucket model resembling the CLSP that additionally contains minimum inventory level constraints. In addition, they extend the algorithm to include setup costs for individual products in addition to the joint setup costs. Akartunalı and Miller (2009) apply a R&F algorithm to an MLCLSP variant with setup times that assumes zero lead times between production stages and explicitly distinguishes between finished products and intermediate goods. Absi and Kedad-Sidhoum (2007) describe R&F heuristics for a multi-resource CLSP with heterogenous parallel machines, setup times, backlogging, minimum lot-sizes and safety stock level violation penalties. Beraldi et al. (2008) develop a R&F heuristic for a small-bucket multi-resource capacitated lot-sizing model with identical parallel machines and sequence-dependent setup costs. The model is derived from applications in textile and fiberglass industries where a large number (hundreds) of machines is present. Hence, it does not use separate variables for each machine. Instead, its decision variables count the number of machines set up for a certain product and also the number of changeovers on machines from a certain product to another in each period. Interestingly, Beraldi et al. (2006) develop a R&F heuristic for a *stochastic* dynamic lot-sizing problem. The model assumes multiple capacitated resources (identical parallel machines), sequence-dependent setup costs, and no holding costs. Demands are deterministic, whereas the capacity consumption factors (processing times) are stochastic. De Araujo et al. (2007) propose a R&F heuristic for a GLSP variant with a single resource and backlogging. Förster et al. (2006) describe a R&F heuristic for solving a real-world tactical production planning problem of a brewery.

Another type of MIP-based heuristics is introduced by Helber and Sahling (2008); Sahling et al. (2009): In the *Fix&Optimize (F&O)* heuristic that they apply to the MLCLSP, a series of MILP is solved in each of which most of the binary variables are tentatively *fixed* to 0 or 1. Only a selected subset of binary variables of the original model is treated as decision variables and *"optimized"* by a run of

an MIP solver. The generation of MILP subproblems that are solved sequentially is performed using three types of decompositions:[9]

1. Product-oriented decomposition: In each subproblem, only the setup variables of a single product are binary, all other setup variables are fixed to 0 or 1.
2. Resource-oriented decomposition: In each subproblem, only the setup variables of product that are manufactured on the same resource are binary (and also only those of a subset of periods), all other setup variables are fixed to 0 or 1.
3. Process-oriented decomposition: In each subproblem, only the setup variables of a pair of products where one product is a successor of the other are binary (and also only those of a subset of periods, e.g., the first or second half of the planning horizon), all other setup variables are fixed to 0 or 1.

Initially, all setup variables are fixed to 1. In order to ensure that each of the generated subproblems has a feasible solution, a highly penalized overtime option is added to the MLCLSP model. "Optimal" setup variable values obtained from the optimal solution to a subproblem are used to tentatively fix these variables in successive subproblems. The three decompositions can be employed alternatively or also serially, e.g., first subproblems are generated with the product-oriented decomposition, second with the resource-oriented decomposition, and third with the process-oriented decomposition. Figure 2.17 exemplifies the principle of F&O with an example showing the iterations (subproblems) of the product-oriented decomposition for a small example with three products. The F&O heuristic seems to outperform Tempelmeier and Derstroff (1996) and Stadtler (2003) with respect to solution quality. Its key advantage is that the algorithm scheme could easily be applied to extensions of the MLCLSP, e.g., by minimum lot-sizes or parallel machines. In addition, other decompositions could be integrated in the algorithm.

Another group of combinations between exact algorithms and heuristics are *LP-based rounding heuristics*: Roughly speaking, they solve LP relaxations of lot-sizing problems and try to construct a feasible solution for the problem by rounding fractional values of setup variables in the optimal solution to the LP relaxation. They belong to the class of integrative combinations where an exact algorithm (for solving the LP relaxations) is integrated into a heuristic. Alfieri et al. (2002) describe LP-based rounding heuristics for the CLSP without setup times which are combined with an SPL-based or shortest-path reformulation and a primal/dual simplex or interior point algorithm for solving the LP relaxation. Computational experiments show that the reformulations yield much better lower bounds, the heuristics

[9] The usage of decompositions in R&F and F&O resembles the decompositions methods in SAP® APO: The Supply Network Planning (SNP) Optimizer offers decompositions by time, product, and resource (Kallrath and Maindl, 2006, pp. 84 and 89). The subproblems into which the problem is decomposed are solved sequentially. The time decomposition uses time windows as in R&F. The product decomposition optimizes the variables associated with a certain subset of products in each subproblem. It thus differs from the product decomposition in F&O that only considers *one* product per subproblem. So-called SNP priority profiles can be specified to provide the SNP Optimizer with information that helps decompose the problem using the product or resource decomposition in an appropriate way.

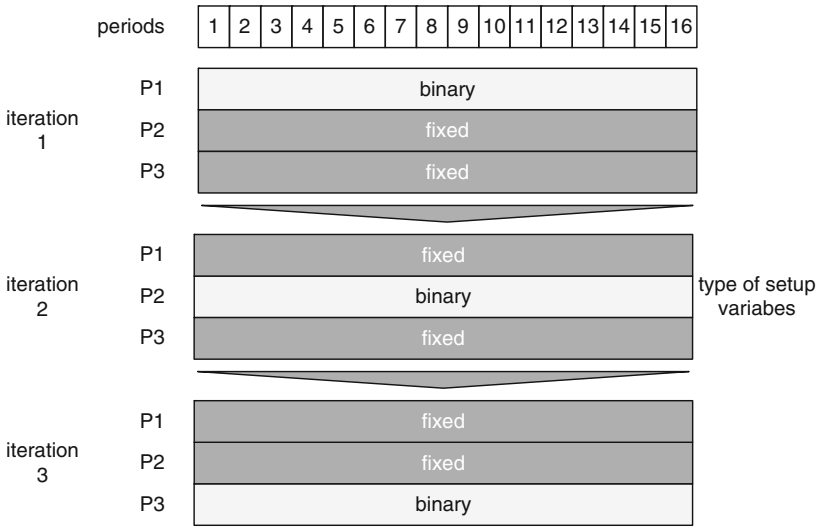


Fig. 2.17 Example – Fix&Optimize with product-oriented decomposition

generate near-optimal solutions, and none of the two reformulations clearly dominates the other with respect to running times of the heuristics. Denizel and Süral (2006) describe reformulations for the CLSP with setup times and develop LP-based heuristics based on the reformulations. In their computational analysis, the SPL-based and shortest-path reformulation perform better for the LP-based heuristics than a standard formulation with a set of cuts added a priori. Pochet and Van Vyve (2004) apply a generic iterative production estimate (IPE) heuristic to a multi-level capacitated lot-sizing problem with setup times. It belongs to the category of LP-based heuristics. The approach is based on the observation that the fractional values in the optimal LP relaxation solution are oftentimes far from the optimal values of the MIP, i.e., rounding and similar heuristics might produce rather bad solutions: Their idea is to modify the formulation in a way that the fractional values in the optimal solution to the LP relaxation are closer to the values of the optimal MIP solution. The IPE heuristic seems to perform well, it quickly finds feasible solutions for all considered instances and the solution quality outperforms other examined heuristics, except for a R&F heuristic that proved better on small and medium-size instances and instances with a strong formulation.

2.2.5 Selected Approaches for Lot-Sizing with Sequence-Dependent Setups

An overview of scheduling, lot-sizing and combined lot-sizing and scheduling models with sequence-dependent setups is given by Zhu and Wilhelm (2006). They

also categorize publications on combined lot-sizing and scheduling models with sequence-dependent setups by the solution approaches used.

Suerie and Stadtler (2003) develop a reformulation and valid inequalities for the CLSP with setup carry-overs that assumes sequence-independent setups.

Haase and Kimms (2000) devise a reformulated CLSD variant that only considers efficient sequences, where a sequence (of products manufactured within a period) is called efficient if it is not dominated by any other sequence. A sequence A is said to dominate another sequence B if its total setup cost is less than that of B, it contains the same products as B, the first product in A is identical with the first product in B, and the last product in A is identical with the last product in B.

Gupta and Magnusson (2005) develop a three-stage heuristic for a CLSD variant with sequence-dependent setup costs and sequence-independent identical setup times. Their MILP formulation contains more binary variables than the formulation of Haase (1996) and, as Almada-Lobo et al. (2008) show, requires an additional set of constraints in order to avoid disconnected subtours in the production sequences. The heuristic is composed of an initial step for finding a feasible solution, a sequencing step that tries to find a better production sequence within each period, and an improvement step in which a backward-oriented heuristics attempts to move and combine production lots of different periods to reduce the total cost. Almada-Lobo et al. (2007) develop another CLSD reformulation as well as a specialized heuristic for the problem, but do not directly compare their approaches with those of Gupta and Magnusson (2005) on the same test instances.

2.3 Transshipment Problems

This section gives a condensed overview of the literature on transshipment problems. We use the term "transshipment problem" referring to stochastic inventory models with transshipments between locations. However, note that the term is *homonymous* in literature, as it may also stand for deterministic linear minimum-cost network flow problems. The content of this section is presented in Lang (2008) in an abbreviated form. Most transshipment models belong to the class of stochastic inventory control models. For basics on inventory control policies such as the (R, Q) and (s, S) policies, the reader is referred to, e.g., Domschke et al. (1997, p. 166ff.) and Tempelmeier (2006, p. 61ff.).

2.3.1 Basics

The goal in so-called *transshipment problems* (Archibald, 2007; Herer et al., 2006; Minner et al., 2003) is to decide for a multi-location inventory system whether one or more transshipment(s) between pairs of locations should be performed at a certain point in time, and which quantities should be transshipped between the locations.

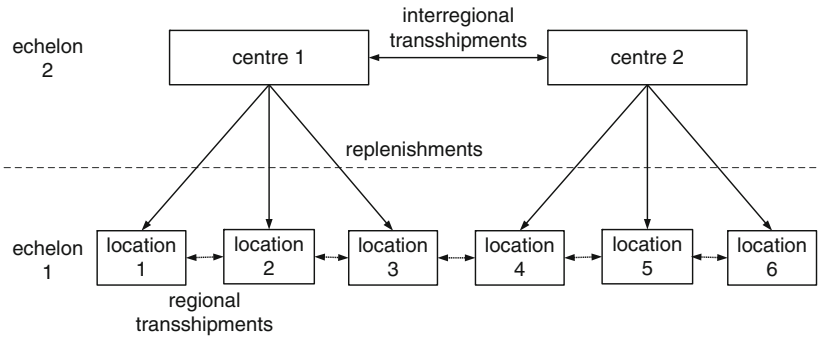


Fig. 2.18 Example of a two-echelon transshipment network

Such transshipments are primarily an emergency recourse to preserve a certain service level despite local stock-outs that are caused, e.g., by low safety stocks or irregular demand behavior. In this context, a *transshipment* is a delivery of units of one or more products from one location to another location on the *same* echelon. The term *replenishment* is used to refer to shipments from a location to another location at a *lower* echelon. An example of a two-echelon transshipment network structure is depicted in Fig. 2.18. Usually, a trade-off has to be made between differing lost sales / backlogging costs at the locations as well as the transshipment costs.

The reasons for performing transshipments are often similar to those for substitutions. Analogously to the reasons for substitutions given in Chap. 1, five benefits of / reasons for transshipments can be distinguished:

- *Increased service level*: Local stock-outs due to supply or production bottlenecks can be avoided by performing transshipments from other locations.
- *Reduction of holding costs*: As transshipments lead to a "risk pooling" effect between locations in a stochastic setting, they might reduce the required level of safety stocks.
- *Reduction of fixed costs*: It might be possible to reduce the total fixed and variable transportation cost by joint replenishments for neighboring locations.
- *Exploitation of unit cost variations*: If unit costs of an input product differ between locations, one could buy it where it is the cheapest and transship it.
- *Reduction of wastage*: If the considered products are perishable, transshipments can be used to reduce the amount of outdated inventory, e.g., by transshipping and consuming stocks at another location first if they have an earlier expiry date.

2.3.2 Classification of Models

Manifold variations of the transshipment problem exist. Hence, we will list several criteria for classifying transshipment models, some of which are also contained in Bhaumik and Kataria (2006):

- *Uncertainty*: Deterministic vs. stochastic (demand and/or lead times)[10]
- *Number of products*: Single-product vs. multi-product
- *Number of periods*: Single-period vs. multi-period
- *Number of locations*: Two-location vs. multi-location
- *Number of echelons*: Single-echelon vs. multi-echelon
- *Cost/demand characteristics*: Identical for all locations vs. heterogenous
- *Correlation of demands of locations*: Correlated vs. uncorrelated
- *Inventory review*: Period vs. continuous review
- *Demand observation*: Observation of total period demand vs. demand stream (e.g., Poisson process)
- *Time of transshipments*: Preventive vs. reactive transshipments (before vs. after demand observation)
- *Lead times*: Non-zero lead times for replenishments/transshipments?
- *Feasibility of transshipments*: Is there always sufficient time for a transshipment, or would a transshipment require more time than allowable in some cases?
- *Flexibility on demand side*: Backlogging vs. lost sales
- *Replenishment policy*: Replenishment policy given vs. to be determined (most models assume that the parameters of the replenishment policy have been set in advance)
- *Cost types*: Only variable vs. also fixed replenishment/transshipment costs
- *Capacities*: Uncapacitated vs. capacitated production / transshipments / inventory
- *Objective*: Expected cost minimization vs. service level target vs. robustness measure
- *Partial order fulfilment*: All-or-nothing vs. partial order fulfilment allowed?
- *Initial inventories*: Does the model assume that initial inventories are in stock at the beginning of the planning horizon?
- *Substitution options*: Are substitute products available for the considered products?
- *Multiple transportation modes*: Are multiple transportation modes available that differ w.r.t. lead time, costs, and/or capacity?
- *Due dates*: Do all customer orders arrive with the characteristic that they should be fulfilled as soon as possible, or rather with specified due dates?
- *Perishable products*: Are the considered products perishable?
- *Multiple demand classes*: Is the demand differentiated into classes with differing priorities?

The following cost types could be included in transshipment models:

- Variable replenishment / production / procurement costs
- Variable transshipment costs
- Fixed (joint) replenishment costs, where "joint" refers to a group of products
- Fixed (joint) transshipment costs

[10] All transshipment models considered in this work are stochastic.

- Holding costs
- Lost sales / shortage costs
- Backlogging costs

In terms of the transshipment policy used, one can distinguish between *partial* and *complete pooling* of the locations' stocks: If the locations share their entire inventory with other locations (by transshipments) and do not reserve a part of the stocks for local demand, this is termed *complete pooling*. In contrast, if each location reserves a proportion of the inventory for demand occurring at that location and does not transship this proportion, this is called *partial pooling*. The term *no pooling* refers to multi-location inventory policies without transshipments.

A stream related to transshipment problems is the research on inventory rationing and inventory control with multiple demand classes with differing priorities (Kleijn and Dekker, 1998; Kranenburg and van Houtum, 2007). E.g., in an inventory system with one product and two demand classes, the basic idea is to ration the product as follows: If its inventory falls below a *critical level*, only high-priority demand is fulfilled, low-priority demand is backlogged or not fulfilled at all. Transshipment policies with partial pooling resemble *critical-level policies*.

Another stream of research connected with transshipment models is the literature on *Vehicle Routing Problems with Pickups and Deliveries (VRPPD)* (see, e.g., Berbeglia et al., 2007; Desaulniers et al., 2002; Parragh et al., 2008b). VRPPD are related to transshipment problems insofar as the latter in practice involve vehicle routing decisions, because transshipments have to be performed with a vehicle fleet. However, standard transshipment models abstract from routing aspects. The problem setting in *Pickup and Delivery Problems (PDPs)*, which are a subclass of VRPPD, is as follows (Parragh et al., 2008b): Given are a number of transportation requests for a single product, each of which specifies that a certain quantity has to be transported from a location A to another location B. The goal is to minimize the total transportation costs by finding the best vehicle routing that executes all transportation requests. Both single- and multi-vehicle PDP variants are described in the literature.

The replenishment and transshipment policy combination can be categorized as follows:

- *Type of replenishment policy*: order-up-to vs. (s,S) vs. (R,Q) vs. other policy
- Transshipment policy: *fixed critical levels* vs. *remaining-time policy* in which the transshipment policy's decisions depend on the remaining time until the next replenishment from an upper echelon arrives
- *Partial transshipments* from multiple locations possible vs. only *full transshipments* of the entire transshipment quantity from single location
- Are transshipments confined to be within location *subgroups* / regions?
- Transshipment policy considers total stock of a product in all locations vs. only stock at receiving and potential transshipping location
- *No vs. partial vs. complete pooling* (limit for maximum proportion of inventory at potential transshipping location to be shared, critical level policy?)
- "Collect" orders to see whether a transshipment makes sense (*batch order processing*) instead of making the transshipment decision right away for each customer demand arriving?

The operational transshipment problems commonly considered in the literature assume that the transshipment network structure is already given. Another group of problems worth investigating could be tactical/strategic *transshipment network design problems* that determine the optimal transshipment links between locations, e.g., by clustering locations into subgroups.

2.4 Solution Techniques for Transshipment Problems

This section reviews selected approaches for solving the transshipment problems described in the previous chapter. First, we review selected existing solution approaches for stochastic transshipment models in Sect. 2.4.1. Section 2.4.2 introduces into the methodology of simulation-based optimization. This methodology will be used in Chap. 8 for solving an optimization problem in multi-location blood bank inventory management with substitutions and transshipments. In addition, a brief introduction to robust optimization is given in Sect. 2.4.3.

2.4.1 Existing Approaches for Transshipment Problems

Existing solution approaches for stochastic transshipment problems can be classified as follows:

- Does it determine replenishment policy parameters in addition to transshipment policy parameters?
- Does it return an optimal or heuristic replenishment policy?
- Does the procedure for determining a replenishment policy take transshipments into account?
- Does it return an optimal vs. heuristic transshipment policy?
- Which approach for determining the transshipment policy is used?
  - Network-flow based approach
  - Newsvendor problem-based approach
  - Other (e.g., simple heuristic rule)

Archibald (2007) considers a transshipment problem with periodic review, zero transshipment lead times, and only variable replenishment and transshipment costs. In addition to transshipments, the model allows for emergency orders that are fulfilled from the supplier. The transshipment policies used are remaining-time policies. The general idea of the solution approach is as follows: In case of a stock-out, it considers all pairs of the stock-out location and another location, and treats them as an isolated two-location system, for which an optimal transshipment policy is known. The so-called τ-heuristic transships in order of increasing transshipment cost from locations from which a transshipment would be optimal in

the isolated two-location system. The α-heuristic transships to all locations if the considered source location would transship to at least one location (considering isolated two-location systems) and is thus comparatively "aggressive". In contrast, the ω-heuristic is more "conservative": It only transships to a location if the considered source location would transship to all locations (considering isolated two-location systems).

Axsäter (2003b) focusses on a transshipment problem with continuous review. The key idea of his approach is to choose the best transshipment option assuming that no further transshipments take place after the transshipment to be performed next. If no value $\delta > 0$ exists, no transshipment is performed, where δ is calculated as the expected cost if no transshipment is performed minus the transshipment cost and expected cost after the transshipment.

Minner et al. (2003) consider a transshipment model with an (s,Q) replenishment policy, fixed replenishment costs, non-zero replenishment lead times, lost sales, and fixed as well as variable transshipment costs. Transshipment lead times are assumed to be 0 in the model, i.e., there is always sufficient time for a transshipment. In contrast to the assumptions of Evers (2001), the model additionally allows for partial instead of full transshipments, includes fixed transshipment costs, and takes replenishments into account that are in transit from the central warehouse. The developed heuristic uses cost trade-offs and is based on the ideas employed in the heuristic devised by Evers (2001).

Chou et al. (2006) develop a robust optimization approach for a transshipment problem with only variable replenishment and transshipment costs and zero transshipment lead times.

Herer et al. (2006) consider a transshipment model with heterogenous locations and periodic review. Demand distributions are assumed to be stationary, and the model only includes variable replenishment and transshipment costs. Replenishments with a lead time of one period are placed after the demand in the previous period has been observed. Transshipments are performed after demand observation and have no lead time. The model in addition assumes complete pooling among the locations, and an order-up-to replenishment policy for each location. They propose an algorithm that is a combination of a heuristic and an exact algorithm: A gradient algorithm heuristically improves the order-up-to levels of the replenishment policy and internally solves minimum cost network flow problems (MCNFP) to determine the optimal transshipment policy for each period. Opportunity costs obtained from the optimal MCNFP solution are used to estimate the gradient for the gradient algorithm. Two variants of this model with capacitated transportation (Özdemir et al., 2006a) and capacitated production and lost sales (Özdemir et al., 2006b) have been developed. Zhao and Sen (2006) compare a stochastic decomposition approach with the algorithm of Herer et al. (2006).

Other recent publications on transshipment problems include Axsäter (2003a); Cheung and Lee (2002); Comez et al. (2006); Iravani et al. (2005); Lee et al. (2007); Nonås and Jörnsten (2005, 2007); Wee and Dada (2005); Zhang (2005); Banerjee et al. (2003).

2.4.2 Simulation-Based Optimization

This subsection gives a condensed introduction to simulation-based optimization methods, based on the content of the more elaborate and in-depth overview of Lang (2005). Many stochastic optimization problems encountered in real-world applications are too complex to be described in closed-form mathematical models and solved analytically. One approach for tackling such problems is to use simulation. The purpose of simulation models is to forecast the behavior of complex, stochastic, real-world systems. Usually, simulation models are used to evaluate the consequences of single decision alternatives without actually implementing these in the real-world system, as this might result in negative effects (Domschke and Scholl, 2005, p. 31).

Simulation models can be categorized along the following three dimensions (Law, 2006, p. 5f.):

- *Static vs. dynamic simulation models*: A static simulation model represents a system by a "snapshot" at a certain point in time, whereas a dynamic simulation model maps a system's behavior over time.
- *Deterministic vs. stochastic simulation models*: If a simulation model does not contain any random influences, it is termed deterministic, and otherwise stochastic.
- *Continuous vs. discrete simulation models*: Continuous simulation models map a system that changes continuously over time, whereas discrete simulation models map systems in which state changes only happen at certain points of time.

A frequently considered case are dynamic, stochastic, discrete simulation models, which are also named *discrete-event simulation* models (Law, 2006, p. 6).

In most cases, only a comparatively small number of alternatives is evaluated using simulation software, and one of these alternatives is selected using a certain decision criterion. Thus, the classical approach in simulation is to perform simulations automatically using software, whereas the choice of an alternative happens manually.

In the approach of *Simulation-Based Optimization (SBO)*, this choice of the "best" alternative is performed automatically by an algorithm that uses a simulation model to compute objective values for solutions that are generated in an iterative heuristic SBO algorithm. Many SBO algorithms can be seen as local search algorithms (for stochastic local search algorithms see, e.g., Hoos and Stützle, 2005). A SBO algorithm repeatedly varies variable values of a solution, evaluates the new solution by simulation, and after a certain number of iterations, returns the best solution found. SBO is mainly used for discrete-event simulation models, but can also be applied to other simulation models. The typical architecture of SBO software systems consists of two components – a simulation model and an optimization component (algorithm) that uses this simulation model (April et al., 2003). Whenever the optimization component evaluates a solution, it forwards the solution's variable values to the simulation model. This in turn executes one or more simulation runs

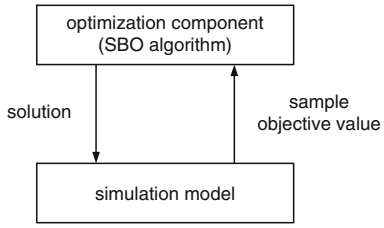


Fig. 2.19 Idealized SBO system architecture

and returns an objective value to the optimization component, which uses the value as an information in the search for good solutions. This basic principle is illustrated by Fig. 2.19.

Analogously to "classical" optimization (e.g., MILP), the objective estimated by simulation in SBO corresponds to the objective function of an optimization problem and the variables of the simulation model to the decision variables of the problem. Unlike it is the case for many common optimization problems, "true" objective values cannot be calculated exactly in SBO, but only be approximated by simulation.

Section 2.4.2.1 gives several examples for real-world applications of SBO. The elements and mathematical formulation of an SBO model are describes in Sect. 2.4.2.2. As many models with differing assumptions are subsumed under the term SBO, we provide a classification of SBO models in Sect. 2.4.2.3. Section 2.4.2.4 focusses on specific aspects of SBO problems that make them in general hard to solve. Common Random Numbers, an approach for dealing with these aspects, are explained in Sect. 2.4.2.5. A taxonomy of SBO algorithms is developed in Sect. 2.4.2.6 and complemented by several selection criteria. Finally, the principle of Pattern Search, one specific class of SBO algorithms, that will be used in Chap. 8 is illustrated in Sect. 2.4.2.7.

2.4.2.1 SBO Applications

Real-world applications for SBO can be found in various areas. Generally, one can distinguish applications of SBO into two categories (Fu, 2001):

- *Design* of the structure of a system: In this case, SBO is used to support singular, non-recurring decisions with a long-term impact.
- *Operation* of a system: Here, SBO is used to optimize operational aspects of a system.

However, the boundary between SBO models for design and operation of systems is not clear: E.g., a supply chain SBO model for determining parameters of replenishment policies neither clearly deals with design nor operation of the system, as these parameters are updated in a certain cycle and kept constant in between in order to avoid planning nervousness. In the following, we give several examples for possible applications of SBO in Operations Research, naming possible objectives and

decision variables of the SBO problems (for additional examples see Fu, 2001; April et al., 2006).

- *Inventory control*: Given a simple single-product inventory system operated using a (s,S) policy, determine good policy parameters regarding the total average costs composed of ordering, holding and backlogging costs that are to be minimized (Fu, 2001).
- *Production system design*: Considering a simulation model of a semiconductor production system, maximize the number of produced wafers by choosing the best configuration of the production system (Fu, 2001). Another example is the choice of the optimal number of machines on multiple levels of a flow production system with buffers (Law and McComas, 2002).
- *Call centers*: Given a simulation model of a call center, minimize the total operating costs while maintaining a certain service level, where the call center head count could be a decision variable of the SBO problem (Fu, 2001).
- *Financial portfolio optimization*: Maximize the expected Return on Investment (ROI) of a portfolio of stocks while ensuring that a maximum acceptable risk level is not exceeded, with the proportions of different stocks in the portfolio used as decision variables in the SBO problem (Fu, 2001).
- *Project portfolio optimization*: Maximize the Net Present Value (NPV) of a project portfolio, with a constraint on the standard deviation of the NPV. The SBO decision variables could describe the subset of potential projects to be implemented (April et al., 2004).
- *Machine maintenance*: Minimize the average of the sum of maintenance and downtime opportunity costs of machines by appropriately choosing a maintenance schedule (Gosavi, 2003).
- *Revenue management*: Choose booking limits for various flight ticket booking classes in a way that the airline's marginal profit is maximized (Bertsimas and de Boer, 2005).

2.4.2.2 Elements of an SBO Model

A *SBO model* is an optimization model that is based on a simulation model and describes an optimization problem (the *SBO problem*). It consists of variables with specified domains, a *result function* that can be estimated by simulation runs, an *objective function* that is either identical with the result function estimation or derived from it, and may also contain *constraints* on the variables. The decision alternatives in an SBO problem are usually not given explicitly, but specified implicitly by the variable domains and constraints. A *solution* (also: *design*, *configuration*) to an SBO problem completely describes a decision alternative for the problem, and consists of values for one or more *variables*. These variables could be variables with a continuous, integer, binary or mixed domain. The *dimension* of an SBO model is defined as the number of variables of the model.

In SBO, uncertainty is usually modeled by assuming certain probability distributions for all uncertain factors in the model. Using random number generators,

simulation software (standard software or specialized code) then generates various *scenarios* by sampling values for all random variables required for the simulation. In this context, the term scenario refers to the values of various exogenous factors in a certain situation/setting that could occur in the modeled system. A scenario is not necessarily a snapshot at a single point in time, but rather the realization of a process. In terms of statistics and computer science, a scenario can be understood as a series of random numbers and is usually implemented by pseudo random number generators that are initialized with certain random seeds. It is a description of all simulated stochastic events, that are relevant for simulation the effects of an SBO solutions. Note that there is a mutual dependency between the individual action performed during the simulation resulting from a certain SBO solution and the scenario: A single action might trigger certain stochastic events (e.g., an order that triggers deliveries with stochastic lead times). In order to simulate those events, further random numbers are required. In a *simulation run*, the implementation of a simulation model is executed for a single SBO solution and a single scenario.

A *simulation result* is a value that is returned from a single simulation run from a virtual *"result function"*. The simulation result could be scalar, or also a more complex data type, e.g., a vector or matrix. In the latter case, the entries of the vector or matrix are referred to as components. One should distinguish between the sampled value of a result function that is returned from a single simulation run, and its expected value, i.e., the "true" value of the result function. The sampled value of the result function is only an estimate of this true value.

The *(sample) objective function* (also: *(sample) performance measure*, *loss function*, *utility function*) has a real-valued domain and can be determined by mathematically combining multiple components of the result function. Also, it might be necessary to perform more than one simulation run for calculating the objective, e.g., one might want to simulate an SBO solution on multiple scenarios in order to examine the robustness of the solution quality under these different scenarios. However, in the most simple case that is usually assumed in the literature, the result function is a simple scalar with real-valued domain and identical with the objective function, and only a single scenario is used for evaluating a solution. Similarly as for the result function, one should differentiate the sample value of the objective function resulting from specific scenarios used from the expected value of the objective function, i.e., its "true" value.

Formally, the standard SBO optimization problem can be described as follows (Fu, 2002, p. 195): The decision maker's goal is to minimize the expected value of an objective function. A set of feasible solutions $\mathcal{X}$ is given. A single solution is described by a variable vector $x \in \mathcal{X}$ with p variables. E.g., with respect to a transshipment problem modeled as an SBO problem, the variable vector could contain order-up-to levels for the locations' replenishment policy as well as critical levels for a transshipment policy. The result function is represented by $r(x, \omega)$, where ω represents a scenario that can be described by values for several random variables. A sample objective value is denoted by $f(x, \omega_1, \ldots, \omega_s)$, where $\omega_1, \ldots, \omega_s$ are the scenarios used for calculating it. In the simplest case, the number of scenarios used for calculating the objective value is $s = 1$. In order to

evaluate $f(x, \omega_1, \ldots, \omega_s)$, the system is sequentially simulated for the solution x on all scenarios $\omega_1, \ldots, \omega_s$, and the objective function estimate is calculated from the simulation results $r(x, \omega_1), \ldots, r(x, \omega_s)$. $f(x)$ denotes the objective value estimate resulting from simulating the solution with arbitrarily chosen scenarios.

The expected value of the objective function for a solution x is denoted by $J(x) = E(f(x))$. The ideal goal of the SBO problem is to find a solution x^* that minimizes the "true" objective $J(x)$, i.e., we search for:

$$x^* = argmin_{x \in \mathcal{X}}(J(x)) \tag{2.77}$$

However, the difficulty is that this true objective $J(x)$ cannot be calculated exactly if there is no finite number of scenarios. Instead, one has to revert to noisy samples of the objective function obtained from simulation runs that only estimate the true value.

2.4.2.3 Classification of SBO Models

In the literature, a large variety of stochastic optimization models with differing assumptions are subsumed under the term "SBO model". In the following, we propose a taxonomy for categorizing SBO models. They can be classified along the following dimensions that include modeling criteria as well as technical aspects related to the implementation of the simulation model:

- *Domain of the result function*: A simulation model returns either a scalar or a more complex data type. E.g., a transshipment simulation model could return a service level measure in addition to the total costs incurred by the replenishment and transshipment policy.
- *Variance of the objective function*: The variance of the objective function (w.r.t. to different random seeds used in simulation runs) could be relatively low, or high, so that a larger number of simulation runs has to be performed to obtain a sufficient approximation of the "true" objective.
- *Relation between result and objective function*: The objective function either immediately equals the result function or is calculated from (possibly multiple) components of the result function.
- *Information on objective function*: The common case is that the objective function is only given as a "black box" by the simulation model, but it could also be available in closed mathematical form in some cases.
- *Information on gradient of objective function*: One can differentiate between (seldom) cases where the partial derivatives of the SBO objective function can be calculated exactly and other cases where these can only be approximated.
- *Number of variables (dimension)*: The number of variables of an SBO problem can range from one or two up to hundreds of decision variables. However, typical SBO algorithms seem no appropriate method for solving large-scale optimization problems with tens of thousands of variables.

- *Type of variables*: The SBO decision variables could be variables with a continuous, integer, binary or mixed domain. A SBO model can contain variables with heterogeneous types.
- *Variable bounds*: Some SBO models specify lower and/or upper bounds for the decision variables.
- *Constraints*: Constraints could be specified on the variables – e.g., that $s < S$ in an (s,S) replenishment policy – as well as the components of the result function – e.g., that the service level is $\geq 95\%$.
- *Description of search space*: The most common case in SBO is that the search space is specified implicitly by the variable domains and constraints. However, in some SBO approaches, the search space is given explicitly as a set of solutions.
- *Duration of a simulation run*: One simulation run on a single solution and scenario could take only few milliseconds, but depending on the complexity of the simulation model, it might take several minutes or even hours to complete.
- *Applicability of variance reduction techniques*: Some simulation models are more suitable for applying *Variance Reduction Techniques (VRT)* such as *Common Random Numbers (CRN)* (see Sect. 2.4.2.5) than others, because it is difficult to synchronize the random numbers when simulation different solutions in some cases.

2.4.2.4 Specifics of SBO

When choosing, designing and implementing SBO algorithms, the following specific traits of SBO should be considered:

Determining Initial Solutions

There is no generic heuristic for finding good initial SBO solutions: As most SBO algorithms are heuristic improvement algorithms, they need an initial solution to start from. However, there is no generic procedure for finding a good initial solution, and in some cases it might even be hard to find an initial *feasible* SBO solution. Possible approaches are to create the initial solution manually, generate it randomly, or use a problem-specific construction heuristic.

"Expensive" Evaluation of the Objective Function

In contrast to other methodologies in optimization, the computation of objective values by simulation consumes the better part of the total computation time. Hence, SBO algorithms have to use simulations runs sparingly in order to be efficient and not to waste a large portion of the computational budget on simulating the "wrong" solutions.

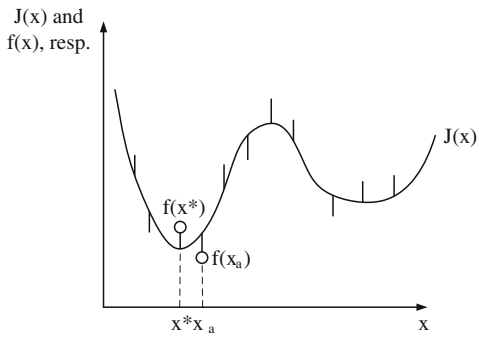


Fig. 2.20 Effect of objective function "noise" in SBO

"Noisy" Objective Function

As mentioned before, only estimates of the "true" objective function are available in SBO due to the random sampling of scenarios. These estimates contain random noise that often distorts the objective value into a certain direction. This problem is illustrated for an SBO minimization problem with a single variable in Fig. 2.20: The variable values are shown on the x-axis. The curve represents the unknown expected value of the objective function, and the vertical lines show the deviations of objective value estimates (obtained by simulation) from their expected value. One can see that the suboptimal solution x_a is chosen as the "best" solution although the optimal solution w.r.t. the true objective is x^*. Thus, there is always a non-zero probability of error in SBO that one solution is considered to be better than another solution though this is not the case. The noise in the objective can be reduced by calculating each objective value estimate using a larger number of scenarios. However, when averaging over scenarios, the error only reduces by factor $1/\sqrt{N}$ when increasing the number of scenarios by factor N (Spall, 2003, p. 14). Typically, SBO algorithms generate a large number of scenarios while running and do not evaluate all solutions on the same scenarios. Another possible approach is to use the same, fixed set of scenarios for evaluating all SBO solutions. A danger in the former approach is that if the objective variance is too high, the SBO algorithm might become almost "blind", i.e., it cannot properly distinguish whether a solution is actually better that another or this observation is caused by distortions due to the noise in the objective function. The latter approach in turn might lead "over-fitting" of a solution to the subset of solutions if this subset is not representative for the distributions of the uncertain factors.

Curse of Dimensionality

The size of the solution space of combinatorial SBO problems (i.e., with integer and binary variables) increases exponentially with the number of variables (so-called "curse of dimensionality", see Spall (2003, p. 14). This, together with the noisy objective function, further complicates an efficient search for good solutions.

"No Free Lunch" Theorems

Roughly speaking, the so-called *"no free lunch" theorems* (Wolpert and Macready, 1997; Spall, 2003, pp. 18ff., 273ff.) express that averaging over all possible instances of optimization problems, no algorithm performs better than another one. Improved performance of an algorithm on a certain class of problems due to specialization for these problems implicates that the algorithm performs worse on other problem classes.

Referring to SBO, the "no free lunch" theorems denote that averaging over all possible types of objective functions, no SBO algorithm performs better than another one w.r.t. the quality of returned solutions. Thus, they suggest that generic SBO algorithms that perform well on all types of SBO problems cannot exist. However, one cannot conclude from their propositions that there is no SBO algorithm that performs better on typical SBO problems than random search, as the search landscape belonging to a specified local search neighborhood for an SBO problem usually has some "structure" that can be utilized: This structure can be described by criteria such as the *fitness-distance correlation*, which describes the relation between the quality of a solution and its distance to a local optimum, and the *ruggedness*, which measures the correlation of the quality of neighbor solutions (Hoos and Stützle, 2005, Chap. 5). The latter can, e.g., be measured by sampling neighbor solutions of an SBO problem. So-called *plateaus* in the search landscapes are areas, i.e., sets of (indirectly) neighboring solutions, for which the objective function has the same value, making it difficult for an SBO algorithm to find better solutions.

2.4.2.5 Common Random Numbers

Simulation results contain stochastic elements if the simulation runs for evaluating solutions are based on randomly generated scenarios: Depending on the scenario(s) used for calculating the objective value estimate for a solution, different values are returned. Due to this noise in the objective function, solutions cannot be evaluated exactly, and a precise comparison of solutions is not possible. So-called *Variance Reduction Techniques (VRT)* aim at reducing the variance of the objective function (Law, 2006, p. 577f.). One such approach is called *Common Random Numbers (CRN)* (Law, 2006, p. 578ff.):

Here, the term variance refers to the variance of the difference between the sample objective functions of two solutions. The idea of CRN is to use completely or partially identical scenarios for calculating the objective values of a pair of solutions instead of using a different solution for evaluating each scenario. The reason why this reduces the variance of the objective value difference is as follows: Each of the sample objective functions $f(x_1)$ and $f(x_2)$ of the two solutions x_1 and x_2 can be interpreted as a random variable $R_1 = f(x_1)$ and $R_2 = f(x_2)$, respectively. From statistics, it is known that $Var(R_1 - R_2) = Var(R_1) + Var(R_2) - 2 \cdot Cov(R_1, R_2)$. Thus, the higher the positive correlation of the objective estimates, the lower the variance of the difference $R_1 - R_2$. CRN try to ensure that the correlation $Cov(R_1, R_2)$ actually becomes positive.

In order to use CRN in implementations of SBO algorithms, the random number generators that create the scenarios for multiple simulation runs on differing solutions have to be *synchronized* (Law, 2006, p. 582ff.). "Synchronization" means that ideally, each individual random number generated and used with a certain semantics in a simulation run is used with the very same semantics in another simulation run. Solely using the same random seed for two simulation runs on different solutions is usually insufficient for ensuring a proper synchronization. Instead, more sophisticated techniques have to be used, which we motivate from the following example:

Assuming a (s, S) inventory system with random lead times, multiple types of random values are required for simulating the system: Demand quantities, demand inter-arrival times, and lead times. These random value types are needed intermittently during the simulation for simulating demand and replenishment arrival events. If a single random number stream (generator) would be used for simulating all three random value types for different solutions, the scenarios would differ unintendedly: The number of orders triggered depends on the choice of s and S. Hence, for one solution, a particular single random value would be interpreted as an order quantity, whereas it would be interpreted as a stochastic lead time for another solution. This problem can be circumvented by using a separate random number stream with its own random seed for each type of random values.

If a complete synchronization of the random numbers is technically impossible or very difficult, already a partial synchronization (so-called partial CRN) of some of the random value types can reduce the variance of the objective value difference significantly (Spall, 2003, p. 396ff.).

2.4.2.6 SBO Algorithms: Taxonomy and Selection Criteria

Based on the classifications contained in (Tekin and Sabuncuoglu, 2004; Fu, 2002; Swisher et al., 2000), SBO algorithms can be categorized as follows:

- *Algorithms for SBO problems with a finite search space*, explicitly defined by a set of solutions: Ranking & Selection, Multiple Comparisons (Tekin and Sabuncuoglu, 2004) and Ordinal Optimization (Fu, 2002)
- *Stochastic Approximation algorithms (SA)*, e.g., Finite Differences Stochastic Approximation (FDSA) and Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 1998; Fu, 2006)
- *Response Surface Methodology (RSM)* algorithms (Hood and Welch, 1993), e.g., Sequential RSM with linear regression or neural networks
- *Sample Path Optimization (SPO)* (Gürkan et al., 1994)
- *Metaheuristics*, e.g., Genetic Algorithms (GA), Tabu Search (TS), Scatter Search (SCS), or Particle Swarm Optimization (PSO)
- *Direct Search (DS)* algorithms such as Pattern Search (PS) (Lewis et al., 2000)
- *Random Search* (Andradóttir, 1998)

A large number of SBO algorithms exist. When facing the question which of those should be chosen for solving a specific SBO problem, the following criteria

can be used to compare the algorithms:

- *Effectivity*: How good are the solutions yielded by the SBO algorithm?
- *Efficiency*: How many simulation runs does the SBO algorithm require to find a solution with a pre-specified minimum quality? How much computation time does the SBO algorithm itself consume, excluding the computation time for simulation runs?
- *Versatility*: Is the algorithm only suitable for a special class of SBO problems? (e.g., only problems with unbounded continuous variables).
- *"Robustness"*:[11] Does the algorithm only perform well on SBO problems with a certain special structure, or does it lend itself for a broader class of SBO problems?
- *Need for manual configuration*: Is extensive parameter tuning required to make the SBO algorithm work well for a specific type of SBO problems?
- *Implementation effort*: How complex and time-consuming is an implementation of the SBO algorithm?

In the literature, only few empirical comparisons of SBO algorithms have been performed, see, e.g., Tekin and Sabuncuoglu (2004) for selective comparisons of some SBO algorithms.

2.4.2.7 Direct Search: Pattern Search

The notion *Direct Search* (Lewis et al., 2000) refers to a class of gradient-free algorithms, which were among the first algorithms applied to SBO problems (Bowden and Hall, 1998). Direct Search algorithms are iterative local search algorithms that start off from an initial solution. In each iteration, a step to another solution contained in a specified neighborhood of the current solution is performed. The most common type of Direct Search algorithms are so-called *Pattern Search (PS)* algorithms. We illustrate the principle of Pattern Search by describing a basic version of a Pattern Search SBO algorithm (Lewis et al., 1998):

Given are a stochastic objective function $f(x)$ to be minimized (implemented as a simulation model), d real-valued variables, and an initial solution x_0. The solution chosen in iteration k of the algorithm is denoted by x_k. $\delta_k > 0$ is a step size parameter and e_i the i-th standard basic vector with a 1 as the i-th entry and 0 for all other entries.

The algorithm successively considers all solutions $x_i' = x_k \pm \delta_k e_i$ for $i = 1, \ldots, n$ – this is a so-called search pattern – until a solution x_i' is found that seems better than the current solution x_k, i.e., x_i' with $f(x_i') < f(x_k)$. If no such solution exists, the step size is halved by setting $\delta_{k+1} = 1/2 \cdot \delta_k$. If a better solution was found, it is used as x_{k+1} and the step size is doubled by setting it to $\delta_{k+1} = 2\delta_k$. This loop is repeated until a termination criterion (e.g., maximum number of iterations) is

[11] This criterion refers to the behavior of an algorithm, and should not be confused with the robustness criteria for solutions used in Robust Optimization.

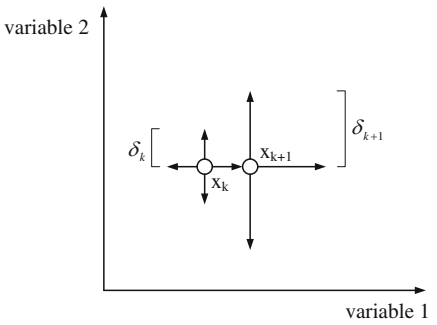


Fig. 2.21 Example of pattern search steps for an SBO problem with two variables

met. Thus, the algorithm increases its step size if it was "successful" and decreases it if no better solutions were found in the neighborhood defined by the search pattern and the step size. The described pattern search algorithm can be interpreted as a first-improvement local search algorithm. Figure 2.21 illustrates two successive iterations of the algorithm for an SBO problem with two variables: At the beginning of iteration k, the current solution is x_k. Assume that the algorithm then considers the four moves ("left, right, up, down") with the current step size δ_k and only finds one better solution x_i' by the move ("right") that increases variable 1. This solution is assigned to x_{k+1}, and the new step size δ_{k+1} is obtained by doubling δ_k.

Lewis et al. (1998) describe a Generalized Pattern Search (GPS) scheme that assumes a "custom" search pattern and procedure for updating the step size. This scheme underlies modern GPS, which were, e.g., applied to an SBO problem by Sriver and Chrissis (2004). Several other PS algorithms based on the GPS scheme have been developed, e.g., a PS algorithm for problems with continuous bounded variables (Lewis and Torczon, 1999) and another PS algorithm for problems with linear constraints (Lewis and Torczon, 2000).

If the search pattern and step size updating procedure meet certain conditions, global convergence of PS algorithms can be proven. Here, "global convergence" does not denote convergence to a global optimum, but to a local optimum from an arbitrary initial solution (Kolda et al., 2004). Roughly speaking, one has to ensure that every solution can be reached from every other solution in a finite number of steps. For details, see, e.g., Lewis et al. (1998).

2.4.3 Robust Optimization

Robust Optimization (RO) (Mulvey et al., 1995; Kouvelis and Yu, 1997; Ben-Tal and Nemirovski, 2002; Scholl, 2001) approaches for decision under uncertainty employ risk-averse objectives, e.g., the minimization of the maximum regret or an optimization of the result occurring for a solution in the *worst-case scenario*. In contrast, most models in classical stochastic optimization pursue a risk-neutral optimization of the expected value of, e.g., a cost function. Both approaches have shortcomings: Risk-neutral stochastic optimization might yield solutions that seem excellent on average, but lead to extremely poor results in scenarios that could

eventuate with a non-negligible probability. Robust optimization might in turn generate solutions that are too conservative and miss out on opportunities for economic benefits. This is especially the case if the decision maker using an RO approach does not fully understand it, and unintentionally chooses a *robustness criterion* that is more risk-averse than his/her own preferences.

In the RO literature, the chosen techniques of modeling uncertainty differ: Some RO approaches assume scenarios with given probabilities (Mulvey et al., 1995), some scenarios without probabilities, others that solely interval data for uncertain parameters are known (Kouvelis and Yu, 1997). Using a more general approach, one could assume probability distributions for random variables. Also, in the literature various, completely different concepts are subsumed under the homonymous term *robustness*, amongst others *feasibility robustness* (also: *solution robustness)* and *optimality robustness* (also: *model robustness*) of solutions. Feasibility robustness criteria measure how likely it is that a solution is feasible to the problem, which is not known in advance if this depends on the eventuating scenario, e.g., due to constraints that include uncertain factors. Optimality robustness criteria measure how much the quality (result value) of a solution varies in the different scenarios that can occur. Even within the concept of optimality robustness, the existing mathematical definitions for operationalizing it are diverse, e.g.:

- Expected value-variance criterion (e.g., linear combination of μ and σ^2)
- Mini-max/Maxi-min criterion (worst-case optimization for minimization and maximization problem, resp.)
- Minimization of the maximum regret (the maximum deviation of a solution's result in a scenario from the scenario-optimal solution)
- Expected value-semi-variance criterion
- Quantile-based criteria, e.g., a Value-at-Risk (VaR) measure such as the 95% quantile of a cost function

Feasibility robustness can, e.g., be included by adding *chance constraints* to a model or adding penalty costs for violations of these constraints to the objective (*"compensation"*) (see, e.g., Scholl, 2001, p. 105). In addition to such generic robustness criteria, one can also use robustness or flexibility criteria that are specific to a certain optimization problem or application (*problem-specific robustness criteria*).[12]

As a review of the vast amount of literature on RO would go beyond the scope of this work, we refer the reader to the excellent overview given and framework developed by Scholl (2001). Note that in the optimization literature, the word "robust" is used very ambiguously and can also refer to certain properties of *algorithms*, rather than solutions.

[12] E.g., Liao and Rittscher (2007b) consider a supplier selection problem where the flexibility provided by options for increasing or reducing supply quantities and for reducing supplier lead times is valued in the objective function. Such options that give additional flexibility can increase the (problem-specific) robustness of a solution to the problem.

Chapter 3
Graphical Modeling of Substitutions and Flexible Bills-of-Materials

This chapter deals with the modeling of product substitution and flexible BOMs. In Sect. 3.1, we describe several real-world applications where product substitution occurs. Section 3.2 presents four approaches for modeling substitution: Blending models, substitution graphs, substitution hypergraphs, and task-oriented modeling. Complementary classification criteria for product substitution models are developed in Sect. 3.3. Finally, Sect. 3.4 focusses on conditions where substitution can be beneficial, requirements for organizationally implementing substitutions, and potential pitfalls that should be kept in mind.

3.1 Applications

Product substitution occurs in several real-world applications:

- *Production of aluminium tubes*: Balakrishnan and Geunes (2000) describe a production planning problem occurring at a aluminium-tube manufacturer: Blooms are intermediate products that are used to produce finished tubes. These tubes are ordered by customers in variants that differ in their dimensions (outer diameter, wall thickness, and length) and material specifications. When producing a tube from a bloom, its outer diameter and wall thickness are reduced and its length is increased ("drawing"). This tube-drawing process is somewhat flexible, so that different blooms can be used to produce a tube with specific attributes. Thus, substitutions among the blooms are possible and the BOM for a certain tube is flexible. Computational results show for the considered case that manufacturing costs can be reduced by approximately 8.7% if BOMs flexibility is exploited.
- *Steel parts for ship and locomotive motors*: In another real-world application at a manufacturer of metal parts for ship / locomotive motors known to the author, substitutions are performed if there is a supply bottleneck for the preferred input product: The metal parts are produced with a CNC turning machine from steel cylinders. These steel cylinders, the raw parts, have a diameter that varies over its length, a certain length and material. If a required steel cylinder is not in stock, a set of rules can be used to determine other steel cylinders that are

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_3,

compatible for producing the required end product: E.g., one can use another steel cylinder whose diameter is higher than that of the preferred product over the entire length, and use it as a substitute, of course increasing the amount of wastage in the turning process. These substitutions can help reduce delayed deliveries significantly.

- *Microchip production*: In the semiconductor industry, there is often a random yield in production processes. E.g., when producing CPUs, some of the produced CPUs can run at 2.4 GHz frequency, some of them only at 2.2 GHz due to the random quality of the products. If there is a higher demand for 2.2 GHz CPUs, one could sell CPUs that would be capable of running at 2.4 GHz as 2.2 GHz CPUs. Thus, the products of the semiconductor manufacturer can be ordered by ascending quality, and higher-quality products can substitute lower-quality products. For product substitution in the semiconductor industry, see, e.g., Gallego et al. (2006).
- *Retail computer manufacturers* such as Dell™ or Lenovo® offer a variety of different PC/notebook configurations in their portfolio. If there is a supply bottleneck for a certain component (e.g., graphics card, display type, or CPU) contained in a configuration, the manufacturer could offer a slightly different configuration with a cheaper, equivalent, or more expensive substitute (*down-selling*, *alternative-selling* and *up-selling*, resp.) for that component to the customer. Either, the manufacturer charges the same price for the configuration, or forwards the change of the component price to the customer. Optimization models for this industry that include substitution are considered by Ettl et al. (2006a,b); Ervolina et al. (2009).
- *Spare parts*: In the inventory management of spare parts, sometimes more than one spare part type would be a feasible replacement for the broken part (Kennedy et al., 2002). In such cases, substitutions can be advantageous to shorten machine downtime (if a substitute can be delivered faster than the preferred spare part) and also reduce holding costs due to demand pooling among substitutable spare parts.
- *Blood transfusions*: As already mentioned in Chap. 1, substitutions by a compatible blood type can be performed if a patient's blood type is not in stock at a hospital (Katsaliaki and Brailsford, 2007). This is also done in practice: In the UK, this "mismatching" is performed in approx. 5% of all cases (BSMS, 2003).
- *Car wiring harnesses*: At a car manufacturer known to the author, a large number of variants of car wiring harnesses (CWHs) exist (more than 2,000). Each of these variants supports a certain set of car features. A variant X that supports a superset of the features supported by another variant Y may be used as a substitute for Y. The situation occurring at the car manufacturer is as follows: Sometimes, a CWH already assigned to a specific customer car order is found to be defective. The CWHs have a rather long lead time. Hence, in that case, one could take another, compatible CWH as a substitute that was previously already assigned to another customer order and is at a later position in the "queue" of CWHs. However, such substitutions might cause a "domino effect"

by triggering additional substitutions for the CWHs that were detached from their assigned customer order to substitute a defective CWH.

- *Windshield interlayer production*: Car windshields contain an interlayer plastic "sheet" between two layers of glass. Several variants of these interlayers exist that are produced on the same production line and sold to customers on rolls. They differ in various attributes: Width and thickness of the foil, length of the foil on a roll, color and material, surface of the foil (coarse vs. even), and cooling (either the foil has to be cooled or an "anti-adhesive paper" has to be added on the roll to prevent the foil from adhering). Non-negligible sequence-dependent setup times and costs occur when changing over from one foil variant to another. Some substitution options exist for these rolls: Foil with a larger width can replace a smaller width, which requires cutting activities and increases wastage. Slightly thicker foil can substitute thinner foil, even foil can substitute coarse foil. Also, one could deliver uncooled foil to a customer who requested cooled foil, given that the customer agrees with it. In this production system, substitutions can help reduce the total setup costs and the total duration of setups, which increases the capacity available for production and thus the effective output of the production system.
- *Medicine*: Pharmacies often offer an identical or similar drug from another pharmaceutical company to the customer if the drug specified on the prescription is not in stock. Also, they might offer several smaller packages if the requested, larger packaging size is not available.
- *Online grocery stores* (also: e-groceries) (Boyer et al., 2003; Boyer and Hult, 2005; Delaney-Klinger et al., 2003; Scott and Scott, 2006; Sleight, 2001; Woudhuysen, 2001): e-groceries such as Tesco.com[1] provide online stores where customers can shop groceries. The groceries are directly shipped to the customers' homes, either from dedicated warehouses or from supermarkets. If some of the products contained in a customer order are not in stock at the warehouse/supermarket from which the customer is served, e-groceries sometimes perform substitutions (Apte and Viswanathan, 2007): E.g., if the ordered orange juice brand is currently out of stock, an orange juice of another brand is delivered. These substitutes are usually sold for the same price and have a higher quality or larger quantity. However, as profit margins are usually low in this type of business, an efficient inventory and substitution policy is crucial to avoid losses due to substitutions.
- *Software components*: Substitutions also occur in the context of tangible, non-physical products such as software components: If a certain software component A covers all functionalities that another component B offers and, in addition, runs on a compatible platform, it might be possible to substitute B by A. Substitutions of software components are only feasible if the architecture of a software system is flexible, which is, e.g., promised by the Service-Oriented Architecture (SOA) paradigm. Web service composition models that contain substitution

[1] http://www.tesco.com/.

aspects are, e.g., developed by Canfora et al. (2005); Chang et al. (2005); Berbner et al. (2006); Di Penta and Troiano (2005). Lang et al. (2008) develop an optimization model for software supplier selection and product portfolio planning that includes substitution of software components.

3.2 Modeling Approaches

Different ways of modeling substitution and flexible BOMs in production planning and inventory control models exist. In this section, we describe the following approaches:

1. Blending models
2. Substitution graphs
   (a) Standard substitution graphs for single-level production structures
   (b) Substitution hypergraphs for multi-level production structures
3. Task-oriented modeling
   (a) State-Task Networks (STN)
   (b) Resource-Task Networks (RTN)

In general, one can distinguish approaches that model flexible BOMs *explicitly* from approaches that model them *implicitly*. The first and third approach model BOMs flexibility implicitly, whereas the second models substitution options explicitly. Similarly, one can distinguish models where an *infinite number of alternative BOMs* exists for fulfilling a certain demand from other models where a *finite* number of alternative BOMs is given.

Another approach for implicitly modeling substitution would be to define a problem-specific function based on the attribute values of two products A and B that returns 1 if A may substitute B and 0 otherwise. The relations defined by such functions implicitly represent substitution graphs, which we will introduce later.

An additional aspect is whether substitution options can be described by *clear-cut* yes–no data, or are modeled in a *soft* way, e.g., by similarity measures for products: In this work, we only consider models of the former type, where one can exactly say whether one product can substitute another and all substitutes are equally suitable, but may differ regarding their costs. In the latter approach, one could introduce a distance metric between products that is calculated from attribute values of the products and measures the (dis-)similarity of two products. The smaller the distance from product A to B, the more suitable is product A as a substitute for B. Instead of measuring distances between products, one could also measure the distance between a customer's requirements and various products that could be offered to the customer.

3.2.1 Blending Models

In industrial *blending models* (Crama et al., 2001), the goal is to determine quantities of several continuous input products that are *blended* to obtain an output product that fulfills certain properties while minimizing a cost objective. Each production plan with specified quantities for the input products that fulfills these properties is considered feasible. The properties could, e.g., be that the output product contains exactly $x\%$ of substance A, at least $y\%$ of substance B, and at most $z\%$ of substance C. The compositions of the input products, i.e., the quantities of the substances contained in them, and their (usually) differing unit costs are given. Also, co-products may result when blending the input products. Note that in most blending models, there is an infinite number of admissible production plans because the quantity variables are continuous.

Blending models are directly related to product substitution, as the input products of the blending process are to some extent substitutable by each other. According to Crama et al. (2001), blending models can be categorized along two dimensions:

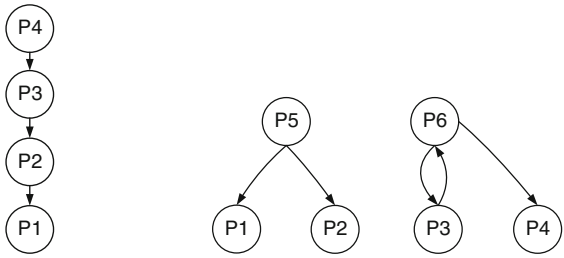
- *Type of industry*: Food, oil (with subcategories), steel industry, steel industry, chemical, energy, agriculture, and miscellaneous applications
- *Planning level* / degree of integration:
  - *Product design models* that solve the blending problem separately, without integration with other production planning models
  - *Long- or mid-term planning* models where the blending problem is integrated in a long- or mid-term (master) planning model
  - *Short-term planning and scheduling* models that contain blending decisions

Another modeling approach is based on an analogy to set covering problems, and resembles blending models: The decision maker has to compose a product from various components, each of which covers a certain set of features. These feature sets may overlap between components. Overall, the product has to fulfil a set of features demanded by a customer. In such models (e.g., considered by Lang et al., 2008), a component A can substitute another component B if A covers all features of B, given that no incompatibilities between components exist.

3.2.2 Substitution Graphs

The substitution options for a specific set of products can be represented and visualized by a *substitution graph*.[2] It is a directed graph that contains one vertex for each product and one (directed) arc for each substitution option. An arc from product A to product B denotes that A is a substitute for B. The *substitution ratio* could be 1:1, e.g., one unit of product A substitutes one unit of product B, or in the more general case x:1, e.g., x units of A substitute one unit of B.

[2] The content of this section is partly presented in Lang and Domschke (2008).



(a) Downward substitution structure (b) General substitution structure

Fig. 3.1 Substitution structures

3.2.2.1 Substitution Structures

Downward (also: *one-way*) *substitution* graphs have a chain structure with the highest-quality product at the tail and the lowest-quality product at the head of the chain. An example is given in Fig. 3.1a: Here, product P4 can substitute all other products, product P3 only P1 and P2, P2 only P1, and the lowest-quality product P1 cannot replace any other product. Downward substitution graphs occur in settings where products are *downward compatible*, where cheaper products can be *upgraded* to better, more expensive products. *General substitution* graphs may consist of more than one connected component and may contain cycles. Figure 3.1b shows a general substitution graph with two connected components and a cycle: Substitution options exist only within, not across the two product groups P1, P2, P5 and P3, P4, P6. In the latter group, P3 can substitute P6 and the other way around.

3.2.2.2 Demand Class-Specific Substitution Options

In practical applications, it might happen that product P1 is a feasible substitute for product P2 if customer X has ordered P2, but not if customer Y has ordered P2, e.g., due to additional compatibility constraints of customer Y. The *demand class* concept (Hsu and Bassok, 1999) can be used to incorporate substitution options of a product that depend on its intended usage (*demand class-specific substitution options*). Figure 3.2 illustrates how the two differing substitution graphs for the customers X and Y and products P1 and P2 of the example introduced above can merged by introducing demand classes. For both customers X and Y, the demand for P1 can only be satisfied by P1 itself. Hence, one demand class D1 that refers to P1 is introduced that contains the demand of X and Y for P1. In contrast, demand for P2 of customer X can also be satisfied by substituting with P1, whereas customer Y does not accept P1 as a substitute for P2. Thus, we have to add two demand classes for P2: One that represents demand of X for which substitution by P1 is allowed (D2X), another one that represents demand of Y which can only be satisfied by P2 itself (D2Y).

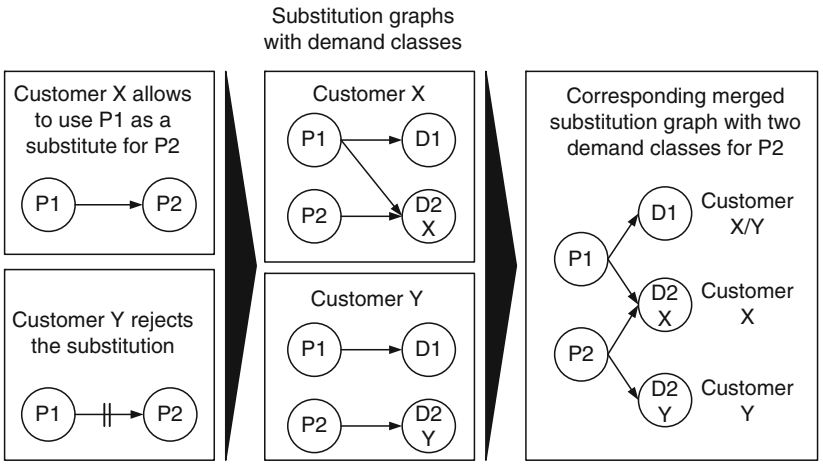


Fig. 3.2 Example – substitution graphs with demand classes and merging substitution graphs for two customers

A substitution graph with demand classes contains a vertex for each (input) product and for each demand class. Arcs from products A and B to demand class D denote that both of them may fulfill demand of class D, i.e., they are mutually substitutable regarding this specific demand class. Demand quantities are associated with demand classes only. Substitution graphs containing only products can be mapped to substitution graphs with demand classes by introducing one demand class for each product, whose demand can be fulfilled by the product itself and all substitutes of the product.

3.2.2.3 Modeling Transitive Substitution Options

Substitution options are either *transitive*, i.e., if product A is a substitute for B and B is a substitute for C, A can also substitute C, or *intransitive*, i.e., A is not necessarily a feasible substitute for C. Intransitive substitution options could, e.g., occur due to technical restrictions or "soft" constraints specified internally or by a customer. Examples for both transitive and intransitive substitution options are given in Fig. 3.3. In the case of transitive substitution options, the model could be limited to a maximum of *one conversion step* per item or allow *multiple conversions* of an item at successive points in time.

Three alternative ways of representing substitution graphs can be distinguished in the case of transitive substitutability:

1. One could explicitly include arcs for both direct and indirect substitution options, where the latter refer to such substitutions that are made possible by performing multiple conversion steps sequentially. This representation would obviously contain several redundant arcs.

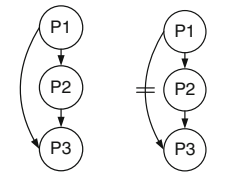


Fig. 3.3 Transitivity of substitution options

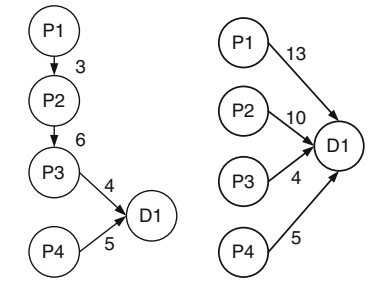


Fig. 3.4 Flattening transitive substitutability

2. One could omit those arcs corresponding to indirect substitution options, as they are redundant due to the transitivity (e.g., the arc from A to C). An example for this is shown in Fig 3.4a, where the conversion cost of each substitution option is given next to the corresponding arc.
3. One could introduce arcs for all indirect substitution options from products to demand classes and remove all arcs between products. This alternative, where the substitution options implied by the transitivity have been "flattened" and the cost of required conversion steps added up, is illustrated with an example in Fig. 3.4b.

In the following, substitution graphs will always be interpreted in the way that all substitution options contained in them are transitive, i.e., if a substitution graph contains arcs (A,B) and (B,C), this means that A can substitute C.

3.2.2.4 Substitution Ratio

In the simplest case, the quantity ratio in substitution options (**substitution ratio**) is assumed to be 1:1. That is, one quantity unit of the substitute can substitute for exactly one quantity unit of the original product. However, consider the example where an online grocery customer orders a package of eight chocolate bars, but the warehouse only has separate chocolate bars in stock and hence delivers eight separate chocolate bars instead of one package. In that case, the substitution ratio is 8:1.

3.2.2.5 Substitution Triangle Inequality

Similarly to the triangle inequality properties introduced for sequence-dependent setup times and costs, one can categorize substitution structures by asking whether they fulfil the *substitution triangle inequality*: It means that the costs of converting product A into product B and successively B into C are the same as or higher than those of directly converting A into C. With conversion costs for substituting product j by i denoted by r_{ij}, it can be formulated as follows:

$$r_{AB} + r_{BC} \geq r_{AC} \tag{3.1}$$

The models with multiple conversion steps described in this work do not assume that the substitution triangle inequality holds true. The inequality is irrelevant for models that only allow for a single conversion step, or one could interpret this single step in a way that it stands for the "shortest path" of multiple conversion steps.

3.2.3 Substitution Hypergraphs

In this section, we focus on an approach for modeling flexible BOMs in multi-level production structures and develop corresponding model classification criteria.

3.2.3.1 Modeling Approach

We begin by defining the terms we use in the context of multi-level substitution models: By **component**, we name an indivisible, atomic good that is produced or purchased. An **assembly** consists of specified quantities of one or more of the following entities: (1) components, (2) other assemblies, which we refer to as **sub-assemblies**, and/or (3) so-called **abstract products**, which are "placeholders" for parts of the assembly that can be implemented by several substitutable components. An abstract product can be interpreted as an "internal demand class". The demand for an abstract product can be satisfied by two or more components or assemblies, these are substitutable w.r.t. this abstract product. In the following, we use the term *internal demand class* synonymously with *abstract product*.

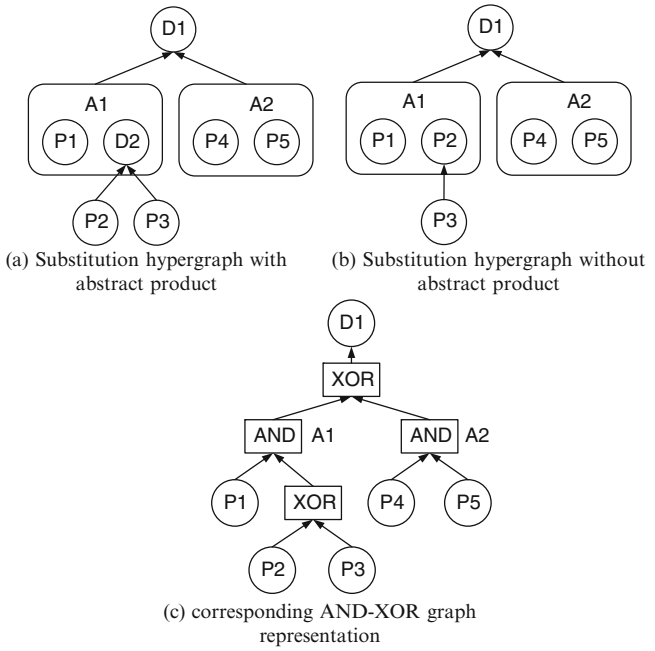
To visualize some exemplary multi-level substitution structures in the following subsections, we use so-called **directed hypergraphs** (see, e.g., Ausiello et al., 2001; Gallo and Pallottino, 1992; Gallo and Scutella, 1998a,b). In contrast to "normal" graphs, hypergraphs do not contain arcs from one vertex to another vertex, but **hyperedges** that go from one **set of vertices** to another set of vertices. In a *directed* hypergraph, the arcs between sets of vertices are directed.

In multi-level substitution models, a directed **substitution hypergraph** is constructed as follows to model the substitution options: We introduce one vertex for each component and each internal or external demand class. In addition, for each

assembly we introduce a set of vertices containing all entities required for that assembly. For each possible way to fulfil demand of a demand class using some components or assemblies, we introduce an arc from the vertex or set of vertices that correspond to these components or assemblies to the vertex corresponding to the demand class.

Note that the resulting substitution hypergraph is a special directed hypergraph whose arcs originate from a set of one or more vertices and end in a set with *exactly one* vertex. The Gozinto factors for entities, i.e., components, subassemblies and abstract products, that are part of an assembly are added to the hypergraph as weights of entity-assembly pairs. The weights of arcs from components or assemblies to an abstract product represent substitution ratios.

An example of a simple substitution hypergraph with five products, two assemblies, one external demand class, and Gozinto factors assumed to be 1 is depicted in Fig. 3.5a. The rounded boxes represent assemblies. Demand of the demand class D1 can either be satisfied by assembly A1 or A2. Assembly A1 consists of products P1 and P2, but product P2 may be substituted by product P3. This is modeled by introducing an abstract product D2 that can be implemented either by product P2 or P3. Alternatively to introducing abstract products, one could add an arc for each substitution option between two products from the substitute to the substitutable product. This is illustrated by Fig. 3.5b. However, this method has the drawback that it cannot model demand-class specific demand for components. Assembly 2 consists of products P4 and P5, for which no substitutes exist.



(a) Substitution hypergraph with abstract product

(b) Substitution hypergraph without abstract product

(c) corresponding AND-XOR graph representation

Fig. 3.5 Substitution hypergraph vs. AND-XOR graph representation for multi-level models

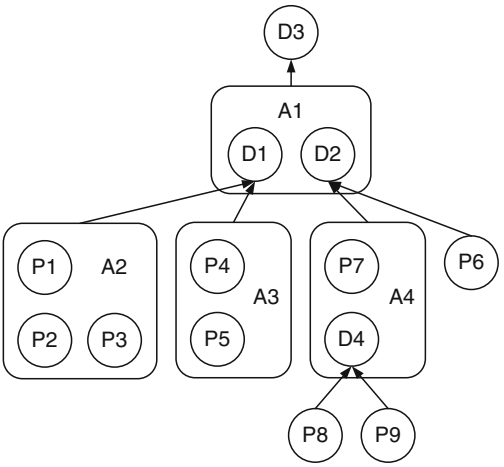


Fig. 3.6 Example of a substitution hypergraph for a multi-level model

An example of a more complicated substitution hypergraph is given in Fig. 3.6. Here, the external demand (demand class 3) is for an assembly A1 that consists of two parts, namely the abstract products D1 and D2. The first abstract product either needs subassembly A2 or A3, the second either subassembly A4 or product P6, which could, e.g., be a finished assembly purchased from a supplier. In subassembly A4, product P8 can be substituted by product P9.

One can always transform a directed hypergraph into a so-called *AND-XOR graph* representation (Ozturan, 2004). This is done for the first example in Fig. 3.5c. The AND-XOR graph contains all vertices of the hypergraph. In addition, an AND vertex is added for each hypergraph vertex set that is the source of a hyperedge. All vertices that are contained in the vertex set have an arc to the AND vertex. One XOR vertex is introduced for each vertex or vertex set of the hypergraph in which multiple hyperedges end. For each hyperedge, we introduce an arc going from the "normal" or AND vertex corresponding to the source vertex or vertex set of the hyperedge to the "normal" or XOR vertex corresponding to the destination vertex or vertex set of the hyperedge.

With regard to substitution graphs, the AND and XOR vertices of this graph can be interpreted as follows:[3] An AND vertex corresponds to an assembly (rounded box). Edges from vertices to an AND vertex indicate that the components or (sub-)assemblies corresponding to these vertices are required for this assembly. An XOR vertex denotes several options to satisfy demand of a certain demand class or abstract product. Each arc ending in an XOR vertex stands for one way of

[3] Wedekind and Müller (1981) developed an approach for modeling *variant* BOMs that is similar to AND-XOR-graph representations of substitution hypergraphs.

satisfying the demand of the corresponding demand class or abstract product. The AND-XOR representation of a substitution hypergraph is more suitable to include Gozinto factors, since they can be added as weights of the arcs that end in AND vertices.

3.2.3.2 Classification Criteria

Number of Components and Assemblies

Models may be restricted to a certain number of components and/or assemblies. E.g., Thomas and Warsing (2007) consider a model with two components and one assembly.

Demand Level

We distinguish models where external demand only occurs for **final assemblies** from others where external demand may also occur for **subassemblies and individual components**.

Level of Substitution

Substitution options in multi-level models can exist for both components and subassemblies. An example with substitutions only on the **component level** is depicted in Fig. 3.7, an example where also substitutions of subassemblies are allowed, i.e., **assembly level** substitutions, is shown in Fig. 3.5a.

Interacting Substitution

Whether a certain substitute can be used within an assembly may depend on its compatibility with other parts of the assembly. This setting in multi-level substitution models, where compatibility issues between components have to be considered, is

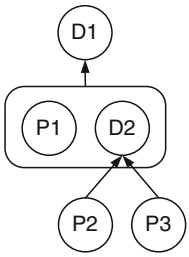
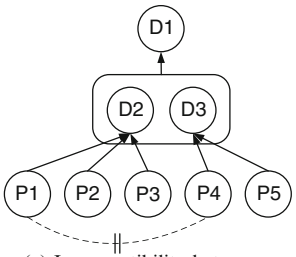


Fig. 3.7 Substitution of components only



(a) Incompatibility between components

D2/D3 implemented by	P4	P5
P1	✗	✓
P2	✓	✓
P3	✓	✓

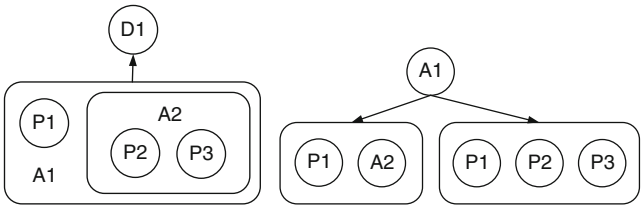
(b) Corresponding compatibility matrix

Fig. 3.8 Interacting substitutions

termed **interacting substitution** in Balakrishnan and Geunes (2000). Chen (2003) uses the synonym "linked substitutes". The aspect of material compatibility is also addressed in Ball et al. (2003). In the automotive industry, so-called *code-rules* describe interdependencies between potential features of a car (Meyr, 2004b). Such code-rules implicitly define the set of feasible BOMs by excluding infeasible combinations of features using Boolean logical expressions (Röder and Tibken, 2006). Code-rules are related to interacting substitutions as they could specify feasible substitutions in applications with interacting substitution. An example of **incompatible components** is given in Fig. 3.8a . Here, if product 2 is used for the internal demand class 1, product 4 cannot be used for internal demand class 2 and vice versa. For assemblies with two parts, such incompatibilities can also be represented in a **compatibility matrix** as shown in Fig. 3.8b. The reasons for incompatibility may either be functional (e.g., a technical restriction) or non-functional (e.g., a customer requirement).

Disassembly Options

The subject areas of remanufacturing and **disassembly** are related to multi-level substitution models, as parts of used products returned by customers can be reused and hence function as substitutes for new parts. In addition, substitution and disassembly could also be combined in the domain of service parts inventory control, as it could be expedient to disassemble a certain service part if only a component of it is needed that is currently not individually in stock. Disassembly options can be represented using disassembly graphs (see, e.g., Schultmann et al., 2002). An example of a disassembly graph and the corresponding Gozinto (hyper-)graph are depicted are Fig. 3.9. In this example, there are two alternative ways of disassembling A1: One can either disassemble it into product P1 and subassembly A2, or subsequently disassemble this subassembly A2 into products P2 and P3 if a unit of P2 or P3 is needed.



(a) Assembly graph (b) Disassembly graph

Fig. 3.9 Disassembly in substitution models

3.2.4 Task-Oriented Modeling

State-Task Networks (STN) (Kondili et al., 1993; Crama et al., 2001) are a modeling framework for production processes (primarily in the process industries) that uses directed graphs with two types of nodes:[4]

- *State nodes*: They represent raw, intermediate and finished products within the production process. In practice, these "products" are often different *states* of an item that goes through the production process.
- *Task nodes*: These represent process activities. Each task consumes certain quantities of input products and produces quantities of one or more output products by an operation that transforms the input products. Co-production can be modeled by tasks that produce more than one output product.

These two types of nodes are illustrated by Fig. 3.10. An arc from a state to a task node denotes that the state is an input product for the task, an arc from a task to a state node that the state is an output product of that task. An example of an STN is given in Fig. 3.11: Task 1 transforms state 1 (input product) into another state 2 (output product). Task 2 uses state 2 to produce state 3. Task 3 is a production activity with co-products, it produces both state 4 and 5.

STN can model product substitution (and flexible BOMs) by introducing several tasks that produce the same product but use different input products. This is illustrated by Fig. 3.12: Task 1 uses products (states) 1 and 2 to produce product (state) 4, which can be interpreted as a demand class). In contrast, task 2 uses products 2

[4] STN are related to the *Production Process Model (PPM)* concept described, e.g., by Stadtler (2005, p. 202f.) and Richter and Stockrahm (2005, p. 443f.): A PPM consists of one or more *operations* (manufacturing stages), each of which consists of several *activities* and is associated with a primary resource. Each of the activities may use one or more secondary resources, consume certain input materials, and produce certain output materials. Precedence relations are defined for activities belonging to the same operation, and visualized as arcs between activity nodes. Also, minimal and maximal lead times between activities can be specified. In contrast to STN, in which task nodes are always connected by a state node in between, there are direct arcs between activities in PPM. Operations are connected by *pegging arcs* from an output material of one operation to an input material of another operation. PPMs seem suitable to model flexible production sequences similarly to STN by defining alternative routings (Stadtler, 2005, p. 205).

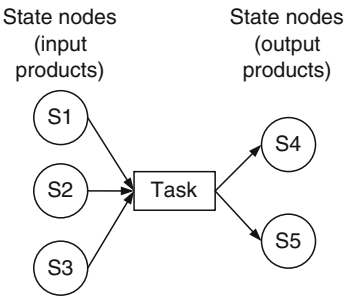


Fig. 3.10 STN task node with state nodes belonging to its input and output products

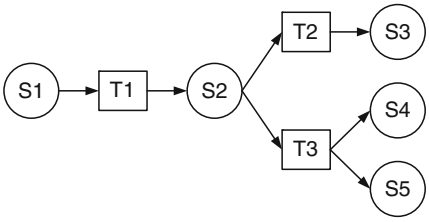


Fig. 3.11 STN example with co-products

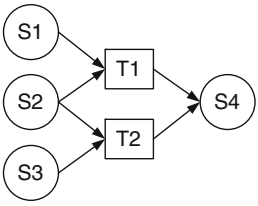


Fig. 3.12 STN example with product substitution

and 3 to produce the same product 4, i.e., it substitutes product 1 by 3. Also, STN can model flexible production sequences by multiple "paths" of tasks that use the same input products. Gozinto factors can be included in STN as weights for arcs that go into task nodes. Additionally, data on the output quantities of a task (per unit) is given that can be added to the arcs originating from task nodes.

Resource-Task Networks (RTN) (Pantelides, 1994) are another modeling framework proposed as an alternative to STN:[5] RTN are directed graphs that contain

[5] There are certain analogies between STN, RTN and the graphical modeling language of Oracle® Strategic Network Optimization (SNO) (formerly: PeopleSoft SNO). The entities in an SNO model are time periods, commodities, nodes, and arcs (Wagner and Meyr, 2005, p. 377f.). "Commodity" either refers to a (physical) good or consumed time. Thus, an SNO commodity representing a physical good corresponds to an STN state node or an RTN resource node that stands for a product/state. Considering the various types of nodes in SNO, SNO process nodes roughly correspond to STN/RTN tasks, and SNO machine nodes to RTN resource nodes that represent processing equipment items. Interestingly, SNO automatically transforms models developed with its graphical language into LP/MILP models.

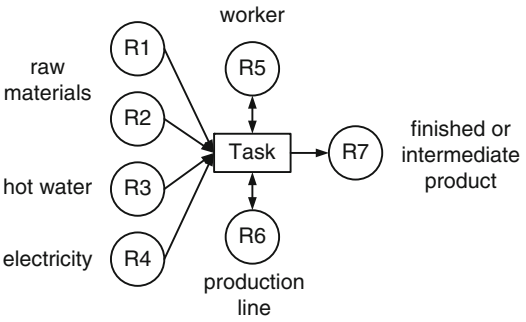


Fig. 3.13 RTN example

task nodes and, in contrast to STN, *resource nodes* instead of state nodes. Products (states), processing equipment items (reactors, machines, etc.), workers and utilities (hot water, steam, etc.) are modeled as resource nodes. Each processing equipment item is modeled as a resource node by treating it as if it was "consumed" at the beginning of the usage by a task and "produced" after completion of the task (Pantelides, 1994, p. 267). For that reason, Figs. 3.13 and 3.14 contain arcs both from and to each resource representing a processing equipment item.[6] An arc from a resource to a task denotes that the task consumes that resource, and an arc from a task to a resource that the task produces the resource. A task may require and produce multiple resources. An example of an RTN is shown in Fig. 3.13.

RTN can model product substitution similarly to STN by introducing several tasks that produce the same output product but use different input products. In contrast to STN, input and output products are modeled by resource nodes instead of state nodes. In addition, RTN allow for modeling alternative production sequences that also differ regarding the resources (i.e., workers, reactors, or machines) they use: For each alternative set of resources available to perform it, a separate task node is introduced (Pantelides, 1994). Such resource substitutability aspects can also be added to STN-based models, but in STN, there is usually a single task node representing all alternative sets of resources available for performing the task. A simple example of an RTN with both substitution and flexible resource assignments is shown in Fig. 3.14: Task 1 uses products R1 and R2 to produce product R4 on machine R5. In contrast, task 2 uses products R2 and R3 to produce the same product R4 on machine R5, i.e., it substitutes product R1 by R3 but uses the same resource. Task 3 consumes the same products as task 1, but is performed on another machine R6.

[6] This differs from the approach chosen in the Oracle® SNO modeling framework: Here, the usage of a machine/reactor would be modeled as a commodity flow (of the "commodity machine time") from an SNO machine node to an SNO process node. Thus, there would only be *one* arc – from the resource to the task, not reversely – in the corresponding graph.

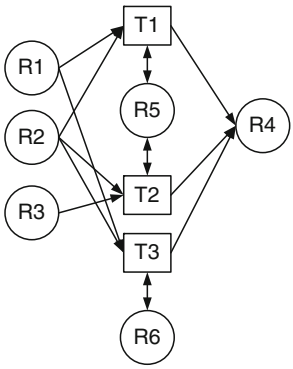


Fig. 3.14 RTN example with product substitution and flexible resource assignments

Table 3.1 Comparison of substitution modeling approaches

Criterion/approach	Blending	Su. graphs	Su. hyper-graphs	STN	RTN
Modeling of su.	Implicit	Explicit	Explicit	Implicit	Implicit
# BOMs	Infinite	Finite	Finite	Finite	Finite
Discrete products	✗	✓	✓	✓	✓
Continuous products	✓	✓	(✓)	✓	✓
Manufacturing	✗	✓	✓	(✓)	(✓)
Process industry	✓	✓	(✓)	✓	✓
MTS	(✓)	✓	✓	(✓)	(✓)
ATO	✗	✗	✓	✗	✗
MTO	✓	✓	✓	✓	✓
Single-level production	✓	✓	✗	✓	✓
Multi-level production	(✓)	✗	✓	✓	✓
Co-products	(✓)	✗	✗	✓	✓
Multi-resource usage	✓	✗	✗	✗	✓

3.2.5 Comparison of Modeling Approaches

When encountering a production or inventory management problem with substitutions or flexible BOMs in practice, the question is which of the approaches presented in Sects. 3.2.1–3.2.4 should be used to model it. Table 3.1 intends to serve as a guideline for choosing the most suitable approach. The primary decision whether to consider substitutions at all depends on the level of aggregation of products in a model: For example, if the level of aggregation of products into groups/families in an aggregated mid-term Master Planning model is so high that substitutions are only possible within the same product group, substitutions options might become insignificant (see Sect. 3.4).

Blending models, STN and RTN model substitution implicitly, whereas substitution graphs and hypergraphs model it explicitly. None of these two options is clearly better than the other. Blending models are especially appropriate for applications with continuous products and an infinite number of feasible BOMs. “Normal” substitution graphs are suitable for applications with single-level production structures,

a finite number of feasible BOMs and no blending. Also, they seem the best choice for cases where substitution decisions are made after production in inventory and transportation planning. Substitution hypergraphs appear predominantly suitable for discrete products and multi-level production, and for ATO manufacturing environments. The task-based STN and RTN modeling approaches are appropriate for both continuous and discrete products, and, at least in their basic version, cannot model blending. They can also map flexible production sequences in addition to flexible BOMs. Also, they can model co-products. Note that substitution graphs and hypergraphs can be transformed into STN/RTN models (see Sect. 4.2.3).

3.3 Model Classification Criteria

In this section, we introduce a classification framework that complements the taxonomy of production planning / dynamic lot-sizing models in Sect. 2.1.1. The framework is primarily intended for classifying dynamic lot-sizing models with substitutions and flexible BOMs / recipes. However, a number of the criteria can also be applied to stochastic inventory control models with substitution or to models in the area of demand fulfillment. The classification criteria are summarized in the Figs. 3.15–3.17. The framework contains the criteria developed in Sect. 3.2 as well as further criteria delineated in the following. The criteria are grouped into seven categories:

1. General modeling aspects
2. Demand
3. Model context
4. Substitution characteristics
5. Conversions
6. Production
7. Special criteria for multi-level production/assembly

In the following, we develop additional criteria for the categories 2–6.

3.3.1 Demand

3.3.1.1 Information and Activeness of Customers Regarding Substitutions

In this work, we only consider models with firm-driven substitution. Concerning the activeness and information of customers w.r.t. firm-driven substitution, four cases can be differentiated:

1. The customer knows that the supplier may perform substitutions and explicitly decides to allow certain substitutions when buying/ordering. This is, e.g., the case for *flexible products* (Gallego and Phillips, 2004). Another example are cases where the customer wants a product that fulfils his/her functionality requirements and does not care *how* this functionality is implemented.

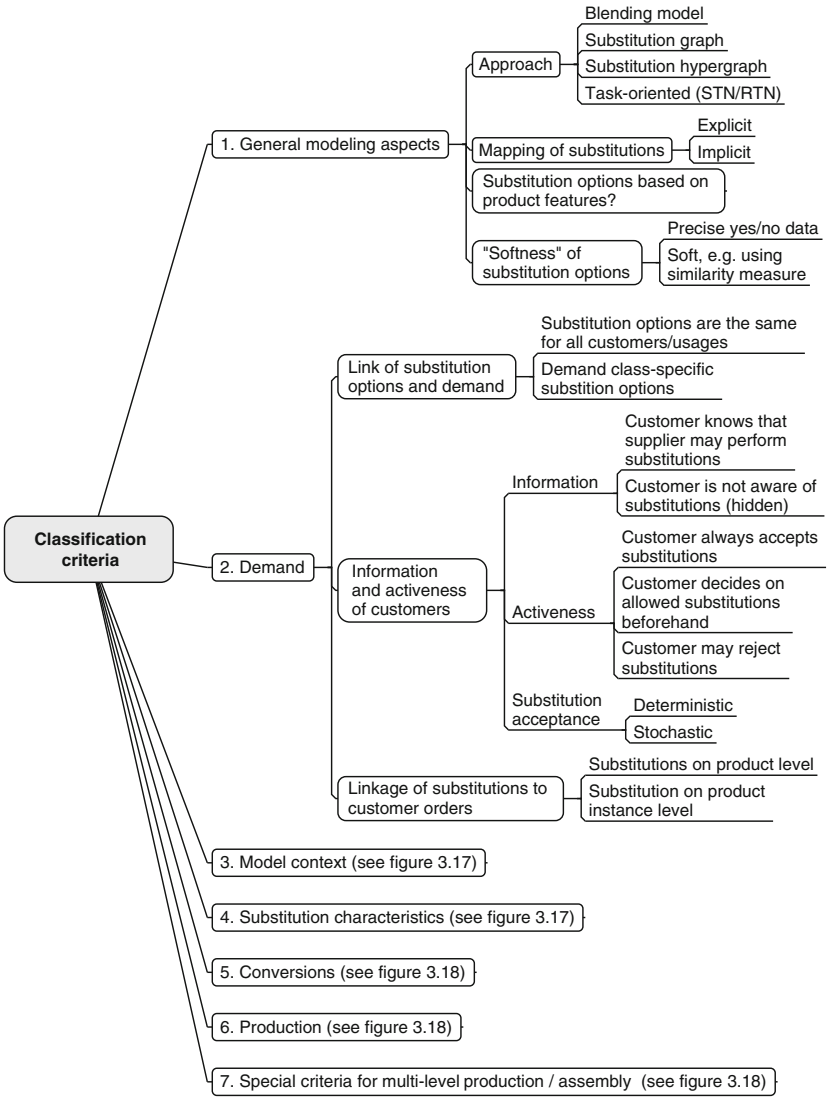


Fig. 3.15 Classification criteria – substitutions and flexible BOMs / recipes – 1/3

2. The customer is not aware of performed substitutions, which is, e.g., possible if "hidden" components of an assembly are substituted. Also, cases where the customer neither determines nor knows how his functionality requirements are implemented belong to this category.
3. The customer does not explicitly decide to buy a product that the supplier may substitute, but will be aware of substitutions performed by the supplier. Rejections of substitutions by customers may occur on delivery. One example for this case is the substitution procedure of some online groceries.

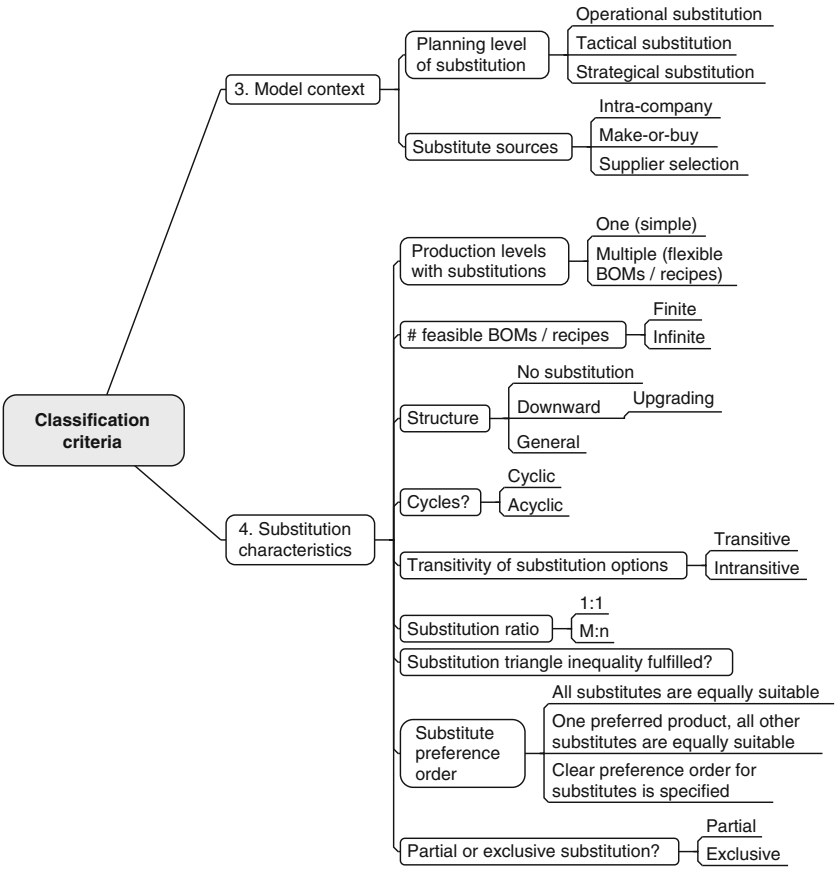


Fig. 3.16 Classification criteria – substitutions and flexible BOMs / recipes – 2/3

4. The firm proposes one or more substitutions when the customer is ordering, e.g., interactively in an online shop. The customer may then either decide for a specific substitution, accept a longer lead time for the original product, or choose not to order the product or a substitute from the firm. This case can be considered a hybrid of firm- and customer-driven substitution.

Cases 1 and 2 are similar from a modeling point of view because the substitution acceptance rate (see Sect. 3.3.1.2) is 100%. Cases 3 and 4 have to be modeled differently as the substitution acceptance is stochastic.

3.3.1.2 Substitution Acceptance

A simplification often made is to assume that the customers accept 100% of the substitutions performed, i.e., to assume *deterministic substitution acceptance*. However,

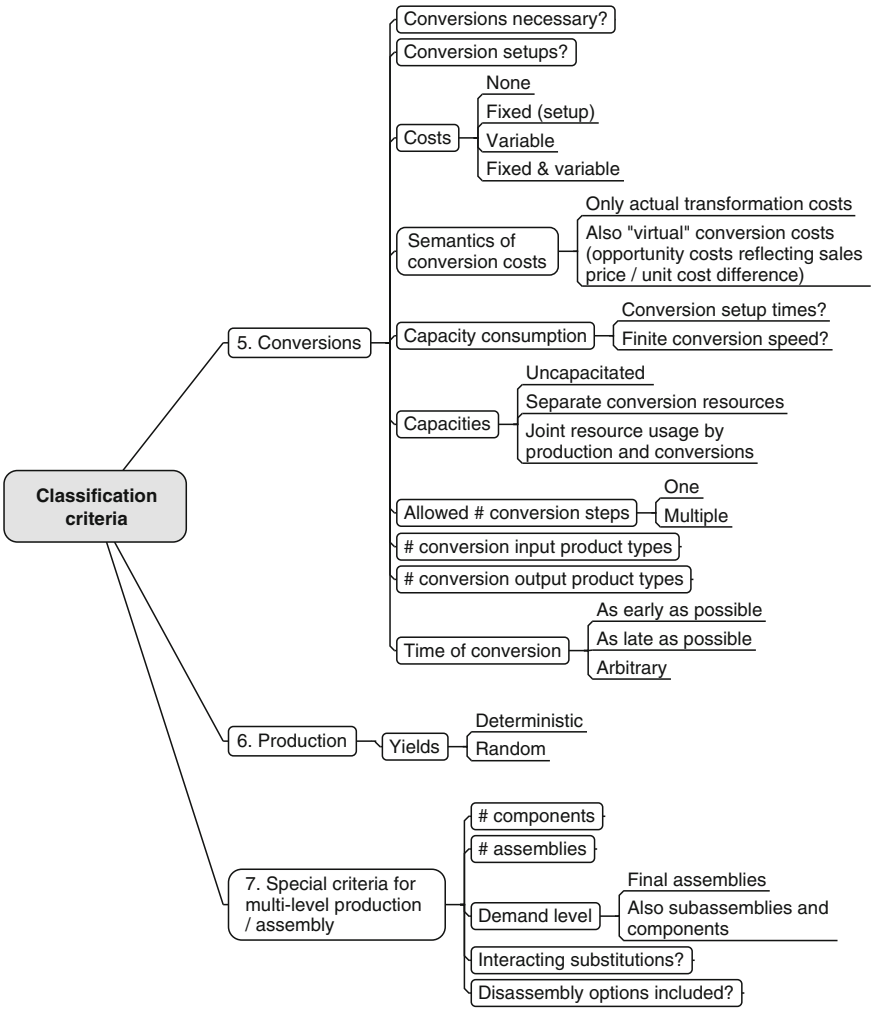


Fig. 3.17 Classification criteria – substitutions and flexible BOMs / recipes – 3/3

though we are considering the case of firm-driven substitution, customers might in practice reject some of the substitutions. This case is called *stochastic substitution acceptance*. The substitution acceptance behavior could be modeled as a "black box", or dependent on type, price, quantity or other attributes of the substitute offered.

In this context, note that the company could either sell the substitute at the same price as the original product or offer a higher or lower price. A higher price could, e.g., be offered if the substitute is a superior "upgrade", a lower price if the substitute is an inferior "downgrade".

3.3.1.3 Linkage of Substitutions to Customer Orders

One can differentiate substitutions on the product level from substitutions on the product instance level: If produced units were not yet assigned to customer orders when they are employed as substitutes, we term this substitution on the *product level*. The case where produced units have to be detached from customer orders for which they were already set apart in order to substitute another product is called substitution on the *product instance level*.

3.3.2 Model Context

3.3.2.1 Planning Level of Substitution

Firm-driven product substitution can be triggered on different planning levels. We distinguish operational, tactical and strategical product substitution: By *operational substitution*, we mainly refer to substitutions that are performed in operational, short-term planning models, i.e., in *short-term production planning and scheduling* and *demand fulfilment* models. *Tactical substitution* characterizes substitutions that are performed, e.g., due to temporary scarcity of a product, caused either by supply or production bottlenecks, or to respond to price changes of input products. Substitutions that take place as a result of product life-cycle management of the company or its suppliers are assigned to the category of *strategical substitution*. The models considered in this work are predominantly tailored to operational and tactical substitution.

3.3.2.2 Substitute Sources

The *substitution decisions* can have various interpretations as products interrelated by substitution options can either be purchased externally, produced internally or both. We distinguish the following three main cases of substitute sources. These cases are not mutually exclusive because combinations are possible as well:

1. The substitution options are *intra-company*, i.e., between products manufactured by the company.
2. Substitution decisions imply repetitive *make-or-buy* decisions, as substitution options between internally manufactured products and purchasable products exist.
3. Substitution decisions imply repetitive *supplier selection* decisions, as certain demands of the company can be satisfied using either of several products – substitutable among each other – from external suppliers.

3.3.3 Substitution Characteristics

3.3.3.1 Substitute Preference Order

Concerning the existence of a preference order among the substitutes that could fulfil the demand of a certain class, one can distinguish three cases:

1. All feasible substitutes are equally suitable for fulfilling demand of the class.
2. There is a preferred product for satisfying the demand, but all other products are considered equally suitable.
3. A clear preference order among the substitutes exists, i.e., they can be ordered descendingly by their suitability.

3.3.3.2 Partial vs. Exclusive Substitution

Sometimes, it might be advantageous from a cost perspective to substitute only a part of the product quantity ordered by a customer. This case, where, e.g., for an order of 100 quantity units of a product, 70 quantity units of exactly this product and 30 quantity units of a substitute are delivered, is called *partial substitution*. The opposite, where it is only feasible to either substitute 100% of a customer order by exactly one product or not to substitute at all, is termed *exclusive substitution*. Examples for both cases are given in Fig. 3.18. They assume the substitution graph shown in Fig. 3.18a. Note that partial substitution of divisible goods has an analogy to the blending models mentioned in Sect. 3.2.1: Frequently, multiple input products are blended that partly contain the same substances, which can be interpreted as partial substitution.

3.3.4 Conversions

3.3.4.1 Interpretations of Conversion Costs

In Sect. 2.1.1.2, p. 24 and Sect. 2.1.1.5, we already introduced the notion of conversion costs. These can contain one or more of the following elements, depending on the production planning model under consideration:

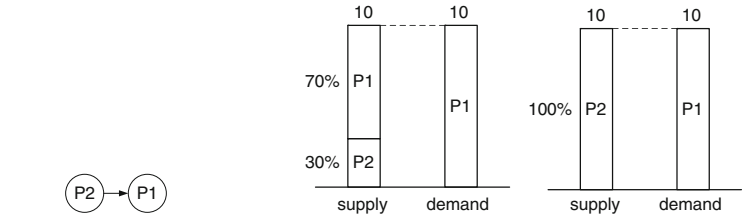


Fig. 3.18 Partial vs. exclusive substitution

1. Conversion costs in the narrow sense:
   (a) Variable transformation costs that are caused by performing physical conversion activities, e.g., labor or machine operating costs for manually or automatically performing conversion activities, repackaging converted products; or costs of auxiliary materials required for conversions.
   (b) Fixed costs of setup activities required for starting physical conversions.
2. "Virtual conversion costs" that represent *opportunity costs* of substitutions:
   (a) If the objective does not contain the unit costs p_1 and p_2 of a product 1 and its substitute 2, the conversion costs should include the unit cost difference $p_2 - p_1$.
   (b) If the objective does not incorporate the sales prices sp_1 and sp_2 of a product 1 and its substitute 2, the sales price difference $sp_2 - sp_1$ should be added to the conversion costs.
   (c) Another possibility could be to include approximate expected costs caused by product returns due to rejected substitutions.

   In a model with profit maximization objective and variable sales quantities that contains unit costs and sales prices, it is unnecessary to include the respective opportunity costs in the conversion costs, as they would otherwise be counted twice. Also, if a cost minimization objective contains unit costs and lost sales costs, there is no need to include opportunity costs in the conversion costs.
3. Variable and/or fixed costs that quantify the administrative overhead caused by substitutions.
4. In addition, the conversion costs could, e.g., include transfer payments or discounts given to customers that aim at increasing the substitution acceptance rate.

In the models developed in this work, we assume that conversion costs data only include conversion costs in the *narrow* sense.

3.3.4.2 Setup Times

In models with substitution, conversion activities might require non-zero setup times.

3.3.4.3 Production and Conversion Capacities

As mentioned in Sect. 2.1.1.2, lot-sizing models in general can be grouped into capacitated and uncapacitated models. The term capacity usually refers to *production* capacities. Regarding substitution models, a possible extension is to assume certain capacities for *conversion (substitution)* activities, so-called *conversion capacities*. Incorporating these conversion capacities, one can distinguish three classes of capacitated models: (1) models with limited *production capacities*, but

unlimited conversion capacities, (2) models with limited *separate production and conversion capacities*, where "separate" means that production and conversion do not share any resources, and (3) models with limited *joint production and conversion capacities*, where production and conversion share some or all resources.

3.3.4.4 Product Type Quantities

The *numbers of input and output product types* of conversion activities may either be identical or different. The numbers are identical if conversion activities only generate products that are already contained in the set of input products. They differ if conversion activities generate some new products (e.g., by certain finalizing or specializing steps) that do not equate to any of the input products.

3.3.4.5 Time of Conversion

Regarding the time of conversion, three modeling options are distinguishable: (1) conversion activities are performed *as early as possible*, i.e., as soon as a decision has been made into which product a quantity unit should be transformed, (2) conversion activities are performed *as late as possible*, i.e., at the beginning of the period in which a converted quantity unit of a product is to be used to fulfil demand, or (3) the time of conversion can be chosen *arbitrarily*. The latter option will likely be used in models with limited conversion capacities to ensure that a feasible solution observing the capacity limits in every period exists. Simple examples for the three options mentioned above are given in Fig. 3.19. They assume the same substitution graph as depicted in Fig. 3.18a. The figures show the inventory levels over time for a product P2 and its substitute P1. The examples assume that initial inventories of P1 and P2 are zero, P2 is set up at time $t = 2$ (with infinite production speed), and a demand for P1 occurs at $t = 4$.

Due to the substitution options, an ambiguity regarding the calculation of the holding costs arises: Assume that the unit holding costs per time unit of P1 are lower than those of a higher-quality substitute P2. If the decision maker decides at

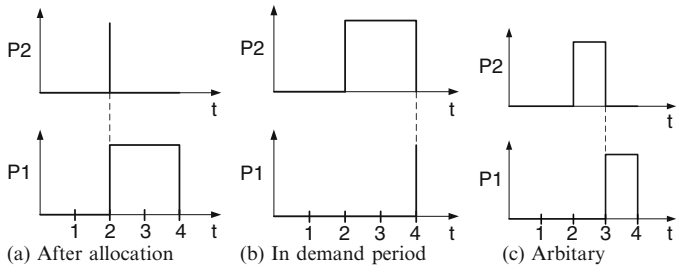


Fig. 3.19 Time of conversion

some point in time to produce units or use stocked units of product P2 to substitute demand for P1 at a later point in time, possibly requiring a conversion, how do we calculate the holding costs for the units of P2 that are used to substitute P1? Three possible ways of calculating them are:

1. Calculate them using the higher holding costs of P2 for the entire time
2. Calculate them using the higher holding costs of P2 until the decision is made that the units are used to substitute for P1, and use the lower holding costs of P1 after that point
3. Calculate them using the higher holding costs of P2 until the units of P2 are actually converted into P1, and use the lower holding costs of P1 after that point

The first option might overestimate the true holding costs. The second one would lead to incorrect results in a rolling horizon planning environment where preliminary substitution decisions could be undone in successive periods. From an academic point of view, the calculation of the holding costs seems problematic in the context of substitution. However, if the difference between the holding costs of P1 and P2 is relatively small, it appears questionable whether the optimal plans resulting from the mentioned options will differ significantly.

3.3.5 Production

3.3.5.1 Yield

Several substitution models originating from applications in the semiconductor industry incorporate *random yields* (see, e.g., Hsu and Bassok, 1999) of production processes in contrast to *deterministic yields*. Uncertain yields occur, e.g., in the production of microprocessors, where the maximum speed of a microprocessor depends on some random quality attributes that can only be determined after production (also see the example in Sect. 3.1). In this setting, substitution is possible because microprocessors are also suitable for running at lower speeds than their maximum feasible speed, i.e., the substitution graph has a downward structure. In addition to random yields of internal production activities, one could also assume random quality of goods purchased from external suppliers and output products of conversion activities.

3.4 Implementing Product Substitution

In this section, we examine the conditions under which substitution can be beneficial, requirements for implementing substitutions in terms of business processes and IT systems, and potential pitfalls that should be considered.

Note that, in practice, product substitution options are often just one out of many aspects to be considered when modeling production and inventory systems (referring to the process industry, see, e.g., Kallrath, 2005). There might be other potential process improvements with higher priority, whose implementation promises a higher gain of efficiency.

One should ensure that substitutions are worth the effort before implementing them. Before contemplating to implement substitutions in a production or inventory system, one should check which the possible benefits of substitutions mentioned in Chap. 1 may apply in the considered case:

- *Increased service level*: Is the service level already sufficient without substitutions? If yes, there might not be a need for performing them.
- *Reduction of holding costs*: Are holding costs a significant cost factor in the considered case?
- *Reduction of setup costs and times*: Presume that substitutions can help reduce the total setup costs and the total duration of setups. This increases the capacity available for production and thus the effective output of the production system. Even if this is the case, is the reduction of total setup costs and times by substitutions big enough to compensate the effort for performing them?
- *Exploitation of unit cost variations*: Do substitute products in a price range similar to the preferred product exist?
- *Reduction of wastage*: Are the considered products perishable?

The key precondition for performing substitutions is that substitution options (including flexible BOMs) exist and they are feasible. Three types of feasibility can be distinguished:

1. It has to be *technically* possible to substitute one product by another (technical feasibility). Sometimes, it may at the first look seem trivial to substitute one product by another product, but unexpected problems in production processes may occur. E.g., if a very similar powder from another supplier is used to substitute the default product in a chemical production process, it might happen that due to slight differences between the products, it lumps in the production line and causes a failure. Thus, substitutes have to be chosen carefully.
2. Substitutions have to conform to *legal* requirements in the country where the production takes place and/or the products are sold (legal feasibility). E.g., legal restrictions may prohibit the exploitation of flexible BOMs in pharmaceutical companies due to regulations for approval of drugs.
3. It is necessary that the *customers* to whom the products are sold accept the substitutions (customer feasibility).

However, even if all these feasibility conditions are fulfilled, product substitutions are not necessarily useful: Two additional conditions are that the business processes and IT systems of the company can handle substitutions and the substitutions are economically beneficial (or regarding another objective). The former condition can be operationalized as follows:

- Data on substitution options (and flexible BOMs) can easily be determined or have already been captured in the IT systems: Which are the feasible substitutes for each product? How high are the conversion costs?
- The production and logistics processes are flexible enough to implement substitutions, i.e., the personnel and production resources can actually implement them.
- The business divisions and/or functional units (logistics, production, sales, etc.) of the company are willing to cooperate in implementing substitutions and/or there is sufficient top management support. This is especially important if substitutable products are manufactured by different divisions of a company who might not have an incentive to give parts of their production away as substitutes for another division's products.

The following points are indicators that the latter condition is fulfilled:

- Conversions are not required for substitutions or only cause low costs, short expenditures of time and low resource usage.
- Production resources are scarce.
- The product lead times are long and/or stochastic.
- The demand quantities are uncertain and subject to high variation.
- The fixed production setup costs are relatively high compared to the conversion costs, holding costs, and unit cost differences of substitutable products.
- The lost sales or backlogging costs are high.
- The one-time costs of creating the conditions for implementing substitutions are acceptable:
  - Costs for training of workers.
  - Costs for adapting machines.
  - Costs for adapting IT systems.
- The additional overhead costs caused by substitutions (and flexible BOMs) are acceptable.

Note that adaptations for substitutions may cause a temporary loss of efficiency of production processes due to learning curve effects.

Another point to be considered is the planning level on which substitutions should be incorporated. In some cases, it might be useful to include them in (mid-term) Master Planning models, in other cases only in short-term lot-sizing and scheduling models or in short-term transportation and inventory planning.

Substitutions might be irrelevant in mid-term production planning if the aggregation of products is chosen in a way that products and their substitutes always belong to the same product groups: If substitutions are only possible among products of the same group and demand forecasts are on the aggregation level of product groups, substitutions will not be considered here, but potentially in short-term models with a lower level of aggregation.

Also, it might make sense to only consider a subset of the feasible substitutions in an application, e.g., only those that are economic, critical for increasing the service level, or easy to implement.

Substitutions can cause several negative effects:

- Substitutions by higher, more expensive substitutes (upgrades) reduce profit margins. Thus, a too high number of substitutions might indicate that demand forecasts lack accuracy, safety stock levels are set too low, demand fulfillment rules need improvements, or measures for counteracting supply bottlenecks are indicated.
- Substitution on a component level of assemblies might significantly increase the number of product variants and complicate error analysis and maintenance. E.g., if dozens of variants of a certain assembly were delivered to customers due to component substitutions, this makes it harder to trace technical weaknesses of the product.
- Myopic consumption of products designated for high-priority, high-margin demand may occur if the stocks are inconsiderately used as substitutes for low-priority, low-margin demand. For example, a successive customer order for a high-margin product might get lost if production resources are scarce and production times rather long, and the stocks of the product were previously consumed as a substitute for a low-margin product.
- Substitutions that are performed on the product instance level, i.e., among units that were already assigned to customer orders, can cause domino effects by triggering additional substitutions for the substitute products. This can cause organizational overhead, and in some cases it might be better to delay the delivery to the first customer instead of performing a chain of substitutions and delaying the delivery to another, perhaps more important customer.
- Customers might get annoyed by substitutions if they perceive the substitute as worse than their preferred product. E.g., customers of e-groceries might decide to stop buying from the e-grocery if they receive unwanted substitutes too frequently.
- If customers start to realize that they frequently receive a better substitute when they try to buy a certain cheaper product, they might decide to always buy the cheaper product although they want the better one. This results in lost revenue for the company, and can be termed *cannibalization* of substitute products (also see Gallego and Phillips, 2004). E.g., customers of a rental car agency might decide to always reserve a cheap compact car as they know that this car will likely not be available at the station and they will get a larger one in that case (which is an example of resource substitution).

Chapter 4
Literature Review

This chapter reviews the literature on lot-sizing with substitution and related fields of research. In Sect. 4.1, we briefly review the literature on and highlight applications of assortment problems, which can be seen as a tactical/medium-term counterpart to dynamic lot-sizing problems with substitution. Sect. 4.2 describes three existing production planning models with substitution/flexible BOM. Related topics, amongst others stochastic inventory control with substitution and flexible bills-of-materials, are briefly discussed in Sect. 4.3.

4.1 Assortment Problems

The early roots of the literature on lot-sizing with product substitution can be seen in the so-called *assortment problem* (Sadowski, 1959). The goal of the basic assortment problem is to determine, given a set of products with a downward substitution structure, the subset of products to be produced or stocked and their respective production/order quantities. Its cost minimization objective is composed of production/stocking costs and substitution costs. The survey paper of Pentico (2008) gives a comprehensive overview of the assortment problem and introduces the following criteria for classifying its manifold variations:

- Deterministic vs. stochastic demand
- Finite vs. infinite number of products/sizes ("discrete vs. continuous demand pattern")
- One or multiple product dimensions
- Number of products to stock/produce: fixed vs. to be determined
- Linear vs. non-linear substitution cost structure
- Stationary vs. non-stationary stocking pattern

Basic assortment problems correspond to single-period lot-sizing problems with product substitution, most of them with downward substitution. Pentico (1988) and Jones et al. (1995) describe reductions of assortment problems to SPL problems: Pentico (1988) solves an assortment problem by reducing it to an SPL problem and using the algorithm of Erlenkotter (1978). Jones et al. (1995) describe a

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_4,

single-period deterministic lot-sizing problem with downward substitution whose objective is composed of fixed plus variable production cost, which is equivalent to an assortment problem. It is solvable in polynomial time, as it is reduced to an SPL problem with a special structure that can be solved using a shortest-path algorithm.

Hardung and Kollert (2005) describe a real-world assortment problem in the automotive sector: A car manufacturer has to decide which variants of an electronic control unit (ECU) it should produce. Each variant covers a certain set of features, and it is possible to install a variant in a car that supports some features not required for this car. As each additional ECU variant increases the overall handling costs, it is not desirable to manufacture all variants for which demand might occur. Instead, some variants are substituted by others that contain a superset of their features. The resulting decision problem of finding the optimal "variant combination", i.e., the set of variants to produce and stock, is transformed into a Warehouse Location Problem (WLP).

A stochastic assortment problem of an Integrated Steel Manufacturer (ISM) is modeled and solved by Denton and Gupta (2004). The goal is to determine the semi-finished products that should be made to stock and the corresponding production quantities. Demands and yields are stochastic. Semi-finished products, e.g., steel slabs, can be used to fulfill demand for more than one finished product. Conversion steps are necessary in some cases: E.g., the with of a slab can be reduced by roughing. Also, often more than one semi-finished product is applicable for fulfilling demand for a certain finished product. Only a part of the demand is fulfilled by semi-finished products made to stock, make-to-order production is used for other demand. The decision problem is decomposed into a strategic assortment problem and an operational problem that determines production quantities for the individual periods of the planning horizon. Both problems are modeled as two-stage stochastic linear programs.

4.2 Lot-Sizing with Substitutions

In the following, we present three production planning models with substitutions / flexible BOMs from the literature:

- We first consider the *Requirements Planning problem with Substitutions (RPS)* (Balakrishnan and Geunes, 2000), a dynamic uncapacitated multi-product lot-sizing model with product substitution, in Sect. 4.2.1.
- The *Substitution With Conversions Problem (SWCP)* (Hsu et al., 2005), another dynamic uncapacitated multi-product lot-sizing model with product substitution, is summarized in Sect. 4.2.2.
- The *Multi-Level lot-sizing problem with Flexible Production sequences (MLFP)* (Begnaud et al., 2006), a big-bucket capacitated multi-level lot-sizing problem that allows for flexible BOMs and production sequences, is described in Sect. 4.2.3. It uses a modeling approach that resembles State-Task Networks (STN).

All those lot-sizing models assume that lot-sizing decisions are made simultaneously with the substitution decisions. However, one can distinguish two additional cases in addition to this one:

- Substitution and lot-sizing decisions are made simultaneously.
- Substitution decisions are made before lot-sizing decisions (pre).
- Substitution decisions are made after lot-sizing decisions (post).

Another production planning model with substitution is developed by Chen (2003): It is a deterministic capacitated multi-period multi-level assembly model with general substitution, but no fixed setup costs. Order quantities for components are already preset. Holding costs of components are zero. The model incorporates earliness and tardiness costs, and assumes that the conversion costs are zero. Also, it includes interacting substitutions (see Sect. 3.2.3.2), i.e., compatibility aspects between components.

Static Economic Order Quantity (EOQ) type lot-sizing models with substitution are considered by Drezner et al. (1995) and Gurnani and Drezner (2000).

4.2.1 The Requirements Planning Problem with Substitutions

The *Requirements Planning problem with Substitutions (RPS)* (Balakrishnan and Geunes, 2000) is a dynamic uncapacitated multi-product lot-sizing model with product substitution. Its assumptions can be summarized as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.
- Single-level time structure.
- Finite time horizon with T periods.
- An arbitrary number of products can be set up in each period (big bucket model).
- Demand refers to demand classes (set of demand classes D).
- Time-varying demand of demand classes that has to be satisfied at the end of each period.
- Each demand class can be satisfied by certain products (demand-class specific substitution options).
- General substitution structure.
- Substitution ratios are arbitrary.
- Uncapacitated production.
- No setup carry-over.
- Single production level.
- Time-varying sequence-independent setup costs.
- Time-varying linear holding costs.
- Time-varying variable production costs.

- Time-invariant variable conversion costs for satisfying demand of a certain class with a specific product.
- All occurring demand has to be fulfilled immediately (no relaxation).
- No initial inventories.
- Cost minimization objective.
- Continuous variables for lot-sizes.

Using the notation given in Table 4.1, the RPS can be formulated as follows:

$$\text{Minimize } F(q,x,I,s) = \sum_{i\in P}\sum_{t=1}^{T}\left(f_{it}x_{it} + p_{it}q_{it} + h_{it}I_{it} + \sum_{j\in D_i} r_{ij}s_{ijt}\right) \tag{4.1}$$

Table 4.1 Notations for RPS

Symbol	Definition
Constants	
m	Number of products
n	Number of demand classes
T	Number of periods
Indices and sets	
$i \in P = \{1,\ldots,m\}$	Products
$j \in D = \{1,\ldots,n\}$	Demand classes
$t = 1,\ldots,T$	Periods
$V = P \cup D$	Vertex set of substitution graph
$E \subseteq P \times D$	Arcs of substitution graph denoting feasible substitutions: $(i,j) \in E$ if product i can fulfil demand of class j
$G = (V,E)$	Substitution graph
$D_i = \{j \mid (i,j) \in E\}$	Set of demand classes whose demand can be fulfilled by product i
$P_j = \{i \mid (i,j) \in E\}$	Set of products that can fulfil demand of class j
Parameters	
d_{jt}	Demand of class j in period t
h_{it}	Non-negative holding cost for storing one unit of product i in period t
p_{it}	Unit production cost of product i in period t
r_{ij}	Substitution cost for fulfilling demand class j by product i per unit
f_{it}	Fixed setup or order cost for product i in period t
a_{ij}	Number of units of product i that substitute for one unit of demand class j
Variables	
q_{it}	Production or order quantity of product i in period t
s_{ijt}	Quantity of product i used to fulfil demand of class j in period t
I_{it}	Inventory of product i at the end of period t
x_{it}	Binary variable that indicates whether a setup for product i occurs in period t

subject to

$$I_{it} = I_{i,t-1} + q_{it} - \sum_{j \in D_i} s_{ijt} \qquad i \in P,\ t = 1, \ldots, T \tag{4.2}$$

$$I_{i0} = 0 \qquad i \in P \tag{4.3}$$

$$d_{jt} = \sum_{i \in P_j} a_{ij}^{-1} s_{ijt} \qquad j \in D,\ t = 1, \ldots, T \tag{4.4}$$

$$q_{it} \leq M \cdot x_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{4.5}$$

$$q_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{4.6}$$

$$I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{4.7}$$

$$x_{it} \in \{0, 1\} \qquad i \in P,\ t = 1, \ldots, T \tag{4.8}$$

$$s_{ijt} \geq 0 \qquad (i, j) \in E,\ t = 1, \ldots, T \tag{4.9}$$

The objective (4.1) is to minimize the sum of setup, unit order/production, holding and substitution costs. Equation (4.2) represents the *inventory balances*. Equation (4.3) denotes that there are no initial inventories. Constraint (4.4) enforces that demand is always satisfied completely using the available substitution options (*product usage and substitution*). It takes the substitution ratios a_{ij} between products and demand classes into account. Due to constraint (4.5), quantity units of a product can only be purchased or produced in a period if setup takes place for the product in that period (*setup forcing*). Equations (4.6)–(4.9) define the domains of the variables.

Balakrishnan and Geunes (2000) reformulate the RPS as a generalized fixed-charge minimum cost network flow problem with arc gains and losses. Based on this reformulation, they show that every RPS instance has at least one optimal solution with the following properties:

- *Zero Inventory-Production (ZIP)* property: For every product and period, the product is either produced in that period or stocks of the product are carried over from the previous period, but not both.
- *Homogeneous Product Lots (HPL)* property: For every demand class and period, the solution fulfills the entire demand of the demand class in the period using only one product (that is a feasible substitute for the demand class).
- *Most Recent Usage (MRU)* property: The entire demand of a demand class in a period is fulfilled using only units from the most recent batch of exactly one product.

Assuming that setup and variable production costs are constant or decrease over time, also the *Immediate Usage (IU)* property holds for at least one optimal solution: If a product is produced in a period, it is used to fulfill demand of at least one demand class in that period, i.e., no stockpiling due to increasing future production costs takes place. In addition, Balakrishnan and Geunes (2000) develop dynamic programming algorithms with exponential worst-case complexity for the RPS and two extensions with backlogging and a two-level assembly structure. Geunes (2003) reduces the RPS to an SPL problem and solves it using the algorithm of Erlenkotter (1978).

4.2.2 Substitution with/without Conversion Problems

The *Substitution With Conversion Problem (SWCP)* (Hsu et al., 2005) is another dynamic uncapacitated multi-product lot-sizing model with product substitution. It differs from the RPS in three assumptions: First, it does not distinguish demand classes from products. Second, it allows for multiple conversion steps. Third, it assumes fixed 1:1 substitution ratios. Its assumptions can thus be summarized as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.
- Single-level time structure.
- Finite time horizon with T periods.
- An arbitrary number of products can be set up in each period (big bucket model).
- Time-varying demand for products that has to be satisfied at the end of each period.
- Multiple conversion steps in successive periods are allowed.
- Feasible substitutes are specified for each product.
- Substitution ratios are 1:1.
- Uncapacitated production.
- No setup carry-over.
- Single production level.
- Time-varying sequence-independent setup costs.
- Time-varying linear holding costs.
- Time-varying variable production costs.
- Time-invariant variable conversion costs for satisfying demand of a certain class with a specific product.
- All occurring demand has to be fulfilled immediately (no relaxation).
- No initial inventories.
- Cost minimization objective.
- Continuous variables for lot-sizes.

Using the same notation as for RPS (see Table 4.1) and setting $D = P$, SWCP can be formulated as:

$$\text{Minimize } F(q,x,I,s) = \sum_{i\in P}\sum_{t=1}^{T}\left(f_{it}x_{it} + p_{it}q_{it} + h_{it}I_{it} + \sum_{j\in D_i} r_{ij}s_{ijt}\right) \tag{4.10}$$

subject to

$$I_{it} = I_{i,t-1} + q_{it} + \sum_{j\in P_i} s_{jit} - \sum_{j\in D_i} s_{ijt} - d_{it} \qquad i \in P,\ t = 1,\ldots,T \tag{4.11}$$

$$I_{i0} = 0 \qquad i \in P \tag{4.12}$$

$$q_{it} \leq M \cdot x_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{4.13}$$

$$q_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{4.14}$$

$$I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{4.15}$$

$$x_{it} \in \{0, 1\} \qquad i \in P,\ t = 1, \ldots, T \tag{4.16}$$

$$s_{ijt} \geq 0 \qquad (i, j) \in E,\ t = 1, \ldots, T \tag{4.17}$$

The objective (4.10) is to minimize the sum of setup, unit order/production, holding and substitution costs. The SWCP allows for multiple conversion steps in contrast to the RPS and does not distinguish products from demand classes. Hence, the inventory balances (4.11) additionally contain the inventory increase resulting from conversions of a substitute into the product as well as the primary demand for the product. Equation (4.12) denotes that there are no initial inventories. Due to constraint (4.13), quantity units of a product can only be purchased or produced in a period if setup takes place for the product in that period (*setup forcing*). Equations (4.14)–(4.17) define the domains of the variables.

Hsu et al. (2005) also consider another model called *Substitution WithOut conversion Problem (SWOP)*, which only allows a single conversion step and assumes that the substitution costs are 0, and show that it can be reduced to a special case of SWCP. They reformulate SWCP as a minimum concave-cost network flow problem, and show that SWOP and SWCP are NP-hard. In addition, they develop dynamic programming algorithms with exponential/factorial worst-case complexity as well as an extension of the Silver-Meal heuristic. Their computational experiments should be interpreted with care, as they generate instances with the same normal distribution for the unit production cost of all (high-quality and low-quality) products, irrespective of their position in the downward substitution structure. This results in instances where producing high-quality products can be cheaper than producing low-quality products, which might be an unrealistic assumption and, according to our computational experiences, makes the problem instances significantly easier to solve.

4.2.3 The Multi-level Lot-Sizing Problem with Flexible Production Sequences

The *Multi-Level lot-sizing problem with Flexible Production sequences (MLFP)* (Begnaud et al., 2006) is a big-bucket capacitated multi-level lot-sizing problem that allows for flexible BOMs and production sequences. It uses a modeling approach with tasks that resembles State-Task Networks (STN) (see Sect. 3.2.4): Instead of production quantity variables for products, the model contains variables that specify how many times / for how long a task is executed. Begnaud et al. (2006) develop a single-resource as well as a multi-resource version of the MLFP.

4.2.3.1 Assumptions

The assumptions of the more general multi-resource MLFP can be summarized as follows:

- Lot-sizing for multiple, continuous tasks (set of tasks A).
- Executing one time unit of a task requires certain quantities of one or multiple input products and (simultaneously) produces one or multiple output products.
- All parameters are deterministic.
- Single-level time structure.
- Finite time horizon with T periods.
- An arbitrary number of tasks can be set up in each period (big bucket model).
- Time-varying demand for products that has to be satisfied at the end of each period.
- Multiple capacitated production resources.
- Each task can be executed on every resource.
- Executing one time unit of a task consumes a certain amount of capacity of exactly one resource.
- Execution of one unit of a task does not simultaneously consume multiple resources, but the same task can simultaneously be executed on multiple resources.
- No setup carry-over.
- Multiple production levels.
- Time-varying sequence-independent setup costs for tasks.
- Time-varying linear holding costs.
- Time-varying variable production costs for tasks.
- All occurring demand has to be fulfilled immediately (no relaxation).
- No initial inventories.
- Cost minimization objective.
- *Integer* variables for lot-sizes of tasks.

4.2.3.2 Formulation

Using the notation given in Table 4.2, the multi-resource MLFP can be formulated as follows:

$$\text{Minimize } F(q,x,I) = \sum_{i\in P}\sum_{t=1}^{T} h_{it} I_{it} + \sum_{a\in A}\sum_{r\in R}\sum_{t=1}^{T} (f_{rat} x_{rat} + p_{rat} q_{rat}) \quad (4.18)$$

subject to

$$I_{it} = I_{i,t-1} + \sum_{a\in A}\sum_{r\in R} (\sigma_{ia} - \rho_{ia}) q_{rat} - d_{it} \qquad i \in P,\ t = 1,\ldots,T \quad (4.19)$$

$$I_{i0} = 0 \qquad i \in P \quad (4.20)$$

Table 4.2 Notations for MLFP

Symbol	Definitions
Constants	
m	Number of products
n	Number of demand classes
T	Number of periods
n_r	Number of resources
n_a	Number of tasks
Indices and sets	
$i \in P = \{1, \ldots, m\}$	Products
$j \in D = \{1, \ldots, n\}$	Demand classes
$t = 1, \ldots, T$	Periods
$r \in R = \{1, \ldots, n_r\}$	Resources
$a \in A = \{1, \ldots, n_a\}$	Tasks
Parameters	
d_{it}	Demand for product i in period t
h_{it}	Non-negative holding cost for storing one unit of product i in period t
p_{rat}	Unit production cost of task a on resource r in period t
f_{rat}	Fixed setup cost for task a on resource r in period t
ρ_{ia}	Number of units of product i required for executing one unit of task a
σ_{ia}	Number of units of product i produced by executing one unit of task a
K_{rt}	Capacity of resource r available in period t
κ^p_{ra}	Capacity required for executing one unit of task a on resource r *Variables*
q_{rat}	Units of task a performed on resource r in period t
I_{it}	Inventory of product i at the end of period t
x_{rat}	Binary variable that indicates whether a setup for task a on resource r is performed in period t

$$\sum_{a \in A} \kappa^p_{ra} q_{rat} \leq K_{rt} \qquad r \in R,\ t = 1, \ldots, T \tag{4.21}$$

$$q_{rat} \leq M \cdot x_{rat} \qquad a \in A,\ r \in R,\ t = 1, \ldots, T \tag{4.22}$$

$$q_{rat} \in \mathbb{Z}^+_0 \qquad a \in A,\ r \in R,\ t = 1, \ldots, T \tag{4.23}$$

$$I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{4.24}$$

$$x_{rat} \in \{0, 1\} \qquad a \in A,\ r \in R,\ t = 1, \ldots, T \tag{4.25}$$

The objective (4.18) is to minimize the sum of holding, setup, and unit production costs. The inventory balances (4.19) consider the inventory changes caused by the execution of tasks as well as those caused by primary demand. Equation (4.20) denotes that there are no initial inventories. The capacity constraints (4.21) ensure that the tasks executed on each resource in a period do not exceed the resource capacity. The q_{rat} variables describe "execution quantities", i.e., they indicate how often activity a is repeated / for how long it is executed. The multiplication with κ^p_{ra} in (4.21) converts these execution quantities into time/capacity units. Due to constraints (4.22), units of a task can only be executed in a period if setup takes

place for the task in that period. Equations (4.23)–(4.25) define the domains of the variables.

4.2.3.3 Transformation of RPS and SWCP into Special Case of MLFP

The RPS can be transformed into a special case of the MLFP as follows:

- Introduce one MLFP product for each RPS product and demand class.
- Introduce one task for each RPS product. This task requires no input product and produces one unit of the corresponding product per unit. The setup costs and variable production costs of that task equal those of the corresponding RPS product.
- Introduce one task for each feasible conversion $(i, j) \in E$ of an RPS product i into an RPS demand class j. This task requires one unit of i and produces a_{ij}^{-1} units of j. Its setup costs are 0, and its variable production costs equal the RPS conversion costs r_{ij}.
- The holding costs of MLFP products corresponding to RPS products equal the holding costs of those. However, the holding costs of MLFP products corresponding to RPS demand classes have to be set to a high penalty value M, as the RPS assumes that no storage takes place after conversion of a product for a demand class.
- As the RPS is uncapacitated and we need at least one MLFP resource, introduce one resource $r = 1$, set all $\kappa_{a1}^{p} = 0$ and all $K_{1t} = 0$. Thereby, the resource effectively becomes unlimited.

Similarly, the SWCP can be transformed into a special case of the MLFP:

- Introduce one MLFP product for each SWCP product.
- Introduce one task for each SWCP product. This task requires no input product and produces one unit of the corresponding product per unit. The setup costs and variable production costs of that task equal those of the corresponding SWCP product.
- Introduce one task for each feasible conversion $(i, j) \in E$ of an SWCP product i into another SWCP product j. This task requires one unit of i and produces one unit of j. Its setup costs are 0, and its variable production costs equal the SWCP conversion costs r_{ij}.
- The holding costs of MLFP products equal those of the corresponding SWCP products.
- As the SWCP is uncapacitated and we need at least one MLFP resource, introduce one resource $r = 1$, set all $\kappa_{a1}^{p} = 0$ and all $K_{1t} = 0$. Thereby, the resource effectively becomes unlimited.

Note that instead of allowing each task to be executed on an arbitrary resource as done in the MLFP, one could also specify a set of tasks A_r that can be executed using resource r for each resource. This would be useful in cases where some tasks can only be executed on specified subsets of resources.

The approach chosen in the MLFP is to "decouple" tasks from resources, i.e., interpret activities with the same input and output quantities on different resources as the same task. Alternatively, one could always interpret activities on different machines as different tasks, as done when using RTN.

A possible extension of the MLFP would be to allow that the execution of a task can simultaneously consume multiple resources, which would be equivalent to a more general RTN-based lot-sizing model.

4.3 Related Aspects

Numerous topics in supply chain management, logistics, production, operations management and economics are related to product substitution and flexible bills-of-materials. As a complete review of all related literature would go beyond the scope of this work, we only describe the relation of each of the topics to substitution and point the reader to some literature. The references are summarized in Table 8.8 contained in the appendix, grouped by the topics. Parts of the content of this section are presented in Lang and Domschke (2008) in an abbreviated form.

One related stream of research is the literature on *stochastic inventory control with substitutions*, which mostly analyzes single-period models with downward substitution (see, e.g., Bassok et al., 1999; Hale et al., 2000; Rao et al., 2004; Liu and Lee, 2007; Yao and Zheng, 2003; Gallego et al., 2006). Some of the models assume that the downward substitutable products are manufactured by production processes with *random yields*, i.e., processes that output products with varying quality (Gerchak and Grosfeld-Nir, 1999; Bitran and Dasu, 1992; Hsu and Bassok, 1999; Duenyas and Tsai, 2000), which are typical for the semiconductor industry.

Inventory control for *perishable products* (Nahmias, 1982) is linked to substitution because blood transfusions, which are substitutable products, are one of the typical examples of perishable products.

As already indicated in Sect. 1.1, substitution options correspond to *flexible bills-of-materials* in multi-level assembly/production structures. Ram et al. (2006) describe a "Material Requirements Planning (MRP) problem with flexible BOM" which is a blending problem as it can be found in the process industry (see Crama et al. (2001)). Depending on the application, *material compatibility* has to be ensured when employing flexible BOMs (Ball et al., 2003). I.e., incompatible combinations of components / materials for producing a product have to be excluded (also see Sect. 3.2.3.2).

Customer-driven product substitution typically occurs in retailing: For instance, if a supermarket customer cannot buy his/her desired orange jam as it is out of stock, he/she might instead buy another type of jam.

Component commonality means that multiple products in a multi-level production system have certain components / intermediate products in common. This topic is linked to *postponement*, which denotes that certain finalizing production steps of an intermediate product are postponed. Postponing the decision into which finished

product an intermediate product is converted provides more flexibility than making the decision as early as possible. Postponement in turn is related to substitution: Assuming that there are some substitution options and, due to demand uncertainty, the decision maker does not want to make the substitution decisions right now, he/she can postpone them for as long as possible.

Substitution might also play a role in *assembly-to-order (ATO)* and *build-to-order (BTO)* systems (DeCroix and Zipkin, 2005; Gunasekaran and Ngai, 2005), but no literature considering flexible BOMs in that context is known to the author.

The relation between flexible BOMs and the topics *remanufacturing* and *disassembly* has already been discussed in Sect. 3.2.3.2.

Lot-sizing models that incorporate *supplier selection* and *multiple sourcing* (Aissaoui et al., 2007) are also related to product substitution, as they contain substitution options between similar or equivalent products of various suppliers. Similarly, lateral *transshipments* (Minner et al., 2003; Axsäter, 2006; Archibald, 2007) between locations bear an analogy to product substitution (also see Chap. 1): Stocks of a products stored at different locations can be interpreted as substitutable if transshipment links exist.

Emergency orders (see Sect. 1.1) can be seen as a flexibility instrument complementary to substitutions and transshipments.

The literature on production planning with *resource substitution* (see, e.g., Henrich et al., 2007) is closely related to what was referred to by alternative production sequences in Sect. 2.1.1.2: Resources can be termed substitutable if they can perform the same activities or output the same products.

Another wide field of research is the literature on *product design* as well as *product variant, line, family and portfolio design*. Assuming that the considered products are to some extent substitutable or have flexible BOMs, product variant and portfolio design can be seen as the long-term decision problem to which substitution models are the corresponding short-term model. This research field is also related to assortment problems and component commonality (see, e.g., Boysen and Scholl, 2008). Product substitution could already be considered in the product design stage, e.g., by designing products in a way that they become downward compatible to another product or use components for which substitutes exist. The anticipation of substitution decisions in the future could make sense in such strategic models, e.g., for spare parts inventory management.

As mentioned in Sect. 2.3.2, Vehicle Routing Problems with Pickups and Deliveries (VRPPD) are related to transshipments.

Another class of models related to substitution are *cutting stock problems* (see, e.g., Poltroniere et al., 2008): The case where finished products can be produced from more than one type of rolls by employing cutting activities can be interpreted as a flexible BOM. Decisions have to be made that specify which roll is used to produce which product using which cutting steps. The former type of decisions can be interpreted as substitution decisions.

The research on *inventory rationing* and *inventory control with multiple demand classes* with differing priorities (Kleijn and Dekker, 1998; Kranenburg and van Houtum, 2007) is related to transshipments (see Sect. 2.3.2), and due to the analogy

of transshipments and substitutions, also with substitution: The usage of a high-quality product as a substitute for a lower-quality product could be limited by a critical level policy.

The topic of substitution also occurs in the field of *revenue management* (Birge et al., 1998; Shumsky and Zhang, 2007; Gallego and Phillips, 2004; Karaesmen and van Ryzin, 2004): E.g., if an airline upgrades an economy class customer to a business class seat, this can be interpreted as downward substitution of resources. Products for which substitution is possible are called *flexible products* in revenue management.

Substitution is also considered in some publications on *demand fulfillment*, *ATP* and *CTP* models, e.g., by Fleischmann and Meyr (2003); Chen et al. (2001). According to Quante et al. (2009), one can distinguish revenue management and demand fulfillment models by two criteria:

1. Replenishment consideration
   (a) No replenishments take place during the entire planning horizon.
   (b) Replenishments are exogenous, i.e., given as data.
   (c) Replenishments can be decided upon.
2. Demand / price consideration
   (a) Demands and prices are exogenous, i.e., fixed.
   (b) Prices are fixed, sales quantities (customer order acceptance) are decision variables.
   (c) Prices are decision variables, sales quantities (customer order acceptance) exogenous.

The lot-sizing models developed in this work belong to the categories 1(c) and 2(a) / 2(b). The blood bank simulation model belongs to categories 1(b) / 1(c) and 2(a).

Another literature stream deals with the *coordination of pricing, production, and procurement* decisions. In such models, prices are not given as data, but decisions variables. Karakul and Chan (2008) consider a joint pricing and procurement model that includes substitution.

Being one example of an *Advanced Planning System (APS)*, SAP® Advanced Planner and Optimizer (SAP® APO) incorporates product substitutions (in SAP® APO documentation also: *product interchangeability*) to some extent: Substitute products and substitute preference orders can be specified in rules-based ATP (Dickersbach, 2006, p. 120ff.), while considering stocks of a product at different locations as different virtual products ("locationproducts"). Product interchangeability can also be modeled for products that substitute a discontinued product (Dickersbach, 2006, p. 421ff.), and by adding products that are substitutable among each other into the same *form-fit-function class*. According to the author's knowledge, support for substitutions has been integrated in the heuristic algorithm for Production Planning and Detailed Scheduling (PP/DS). For details on substitutions in SAP® APO, also see its online documentation (SAP, 2008).

The idea for using *hypergraphs* to model flexible BOMs (see Sect. 3.2.3) emerged from the publication of Ozturan (2004) on optimization models for *electronic barter* exchanges, which are however insignificant in practice. On the online platform of an electronic barter exchange, participants can offer an item and specify which other items they would accept in exchange for that item. Such offers describe the "acceptable substitutes" for a certain item.

The concept of substitution also occurs in economics and especially production theory (Varian, 2007; Domschke and Scholl, 2005, p. 85f.): Good 2 is said to be a *perfect substitute* for another good 1 w.r.t. to a certain consumer if the actor is willing to substitute 1 by 2 at a constant substitution ratio (Varian, 2007, p. 38). Referring to the substitutions graphs described in Sect. 3.2.2, substitute products in a substitution graph are always perfect substitutes. The *marginal rate of substitution (MRS)* is defined as the slope of the indifference curve of the quantities of two goods (Varian, 2007, p. 48), i.e., it describes all quantity pairs of goods 1 and 2 that have the same utility for the consumer. It is -1 if one product is a perfect substitute for the other. The concept of the *cross price elasticity of demand* is related to customer-driven product substitution and describes how strongly the demand for good 1 changes as the price of good 2 changes. *Production functions* (Varian, 2007, p. 323) can model that one production factor can partially or totally be substituted by other production factors. For instance, a production function can describe a setting with perfect substitutes. Other examples of substitutive production functions are Cobb–Douglas production functions and production functions describing linear technologies (Domschke and Scholl, 2005, p. 85f.). The *technical rate of substitution (TRS)* is the slope of the isoquant curve of the quantities of two production factors (input goods), i.e., it describes all quantity pairs of goods 1 and 2 that lead to the same output (Varian, 2007, p. 328). It is thus analogous to the MRS, but refers to production output instead of consumer utility.

Chapter 5
Efficient Reformulations for Uncapacitated and Capacitated Lot-Sizing with Substitutions and Initial Inventories

5.1 Introduction

This section considers extensions of two well-known single-level lot-sizing models, namely the Wagner–Whitin Problem (WWP) and the Capacitated Lot-Sizing Problem (CLSP), that incorporate *product substitution options*.[1]

The literature published on lot-sizing models with substitution until now does not cover two aspects that are important in real-world production planning problems: *Initial inventories* are not taken into account. While these can be neglected easily without loss of generality in standard lot-sizing models by netting demands, this cannot be done if substitutions are possible, as the net demands depend on substitution decisions which are part of the optimization problem. E.g., consider a lot-sizing problem with two products A and B whose initial inventory is 60 and 20 units, respectively. In addition, assume that A can substitute B, and the gross demand for A and B in period 1 is 40 and 30, respectively. In this case one cannot say that the net demand of B in period 1 is $30 - 20 = 10$, because it could be optimal due to the cost parameters and demand in subsequent periods to partially substitute B by A in period 1, so that B is not set up in period 1. In addition, no models and algorithms for lot-sizing with substitution and *capacitated resources* have been developed. If production bottlenecks exist, it is necessary to consider production capacities in combination with substitutions: Capacitated resources can on the one hand be the reason for substitutions, on the other hand limit the amount of substitutions (e.g., if a machine that could produce substitutes is working almost to full capacity).

Our uncapacitated model differs from the RPS (Balakrishnan and Geunes, 2000) in the following assumptions: Initial inventories are allowed to be $\neq 0$, and the substitution ratios are 1:1 instead of x:1. The difference between the SWCP and SWOP (Hsu et al., 2005) and our model is that the SWCP and SWOP are restricted to downward substitution structures and assume no initial inventories. In addition, the SWCP permits multiple conversion steps in successive periods in contrast to our

[1] This section is an extended version of the publication Lang and Domschke (2008).

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_5, © Springer-Verlag Berlin Heidelberg 2010

model, whereas the SWOP only allows a single conversion step and assumes that the substitution costs are 0.

5.2 Outline

We focus on dynamic deterministic lot-sizing models with firm-driven product substitution. The models that we formulate allow for general substitution structures and use the concept of demand classes. Additional assumptions are a single-level production structure, multiple products and demand classes, a maximum of one conversion step, and initial inventories. The first model that we formulate – named Lot-Sizing Problem with Substitution and Initial Inventory (LSP-SI) – is an uncapacitated Wagner–Whitin-type model with product substitution, whereas the second model – referred to as Multi-Resource Capacitated Lot-Sizing Problem with Substitution (MR-CLSP-S) – is an extension of the multi-resource CLSP with production setup times and lost sales / overtime by product substitution options. The cost minimization objective function contains variable substitution costs as an additional cost type. The remainder of this section is structured as follows: We first formulate the LSP-SI and describe how to reduce it to a special case of the Capacitated Facility Location Problem (CFLP). Based on this transformation, we develop an SPL-based extended formulation of the model. In addition, we devise valid inequalities for the original formulation. After introducing the MR-CLSP-S, we extend the SPL-based reformulation of the LSP-SI to include the additional assumptions of the MR-CLSP-S. The efficiency of the reformulations and valid inequalities is examined in extensive computational experiments. In these experiments, we also compare the complete SPL-based reformulations to approximate extended formulations.

5.3 The Lot-Sizing Problem with Substitution and Initial Inventory

In this section, we consider an uncapacitated, deterministic, single-level, multi-product, multi-period dynamic *Lot-Sizing Problem with Substitution and Initial Inventory (LSP-SI)*. Its detailed assumptions are as follows: Initial inventories of the products are in stock at the beginning of the first period. One quantity unit of a product satisfies demand of exactly one quantity unit of a demand class, i.e., the substitution ratio is 1:1. Only certain products can be used to satisfy demand of a specific demand class. The model embraces general substitution graphs. Conversion of units of a product is carried out immediately before shipping to the customer, thus no converted goods are kept in stock. The decision variables model setup and lot size decisions and the allocation of stocks to demands, i.e., the substitution decisions. The objective is to minimize the sum of fixed setup cost and variable order/production cost, holding cost and substitution cost.

5.3.1 Model Formulation

The notation for the LSP-SI is given in Table 5.1. For the ease of notation, we do not explicitly distinguish the indices of products and demand classes. However, note that in the vertex set V, we have to draw a distinction between product and demand class vertices with identical indices. Using this notation, the LSP-SI can be formulated as a linear mixed-integer programming model:

$$\text{Minimize } F(q, x, I, s) = \sum_{i \in P} \sum_{t=1}^{T} \left(f_{it} x_{it} + p_{it} q_{it} + h_{it} I_{it} + \sum_{j \in D_i} r_{ij} s_{ijt} \right) \quad (5.1)$$

subject to

$$I_{it} = I_{i,t-1} + q_{it} - \sum_{j \in D_i} s_{ijt} \qquad i \in P,\ t = 1, \ldots, T \quad (5.2)$$

Table 5.1 Notations for LSP-SI model

Symbol	Definition
Constants	
m	Number of products
n	Number of demand classes
T	Number of periods
Indices and sets	
$i \in P = \{1, \ldots, m\}$	Products
$j \in D = \{1, \ldots, n\}$	Demand classes
$t = 1, \ldots, T$	Periods
$V = P \cup D$	Vertex set of substitution graph
$E \subseteq P \times D$	Arcs of substitution graph denoting feasible substitutions: $(i, j) \in E$ if product i can fulfil demand of class j
$G = (V, E)$	Substitution graph
$D_i = \{j \mid (i, j) \in E\}$	Set of demand classes whose demand can be fulfilled by product i
$P_j = \{i \mid (i, j) \in E\}$	Set of products that can fulfil demand of class j
Parameters	
d_{jt}	Demand of class j in period t
h_{it}	Non-negative holding cost for storing one unit of product i in period t
p_{it}	Unit production cost of product i in period t
r_{ij}	Substitution cost for fulfilling demand class j by product i per unit
I_{i0}	Initial inventory of product i
f_{it}	Fixed setup or order cost for product i in period t
Variables	
q_{it}	Production or order quantity of product i in period t
s_{ijt}	Quantity of product i used to fulfil demand of class j in period t
I_{it}	Inventory of product i at the end of period t
x_{it}	Binary variable that indicates whether a setup for product i occurs in period t

$$d_{jt} = \sum_{i \in P_j} s_{ijt} \qquad j \in D,\ t = 1, \ldots, T \tag{5.3}$$

$$q_{it} \leq M \cdot x_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{5.4}$$

$$q_{it} \geq 0, I_{it} \geq 0, x_{it} \in \{0, 1\} \qquad i \in P,\ t = 1, \ldots, T \tag{5.5}$$

$$s_{ijt} \geq 0 \qquad (i, j) \in E,\ t = 1, \ldots, T \tag{5.6}$$

This formulation is referred to as LSP-SI$_{ORIG}$. The objective (5.1) is to minimize the sum of setup, unit order/production, holding and substitution cost. Equation (5.2) represents the *inventory balances*. Constraint (5.3) enforces that demand is always satisfied completely using the available substitution options (*product usage and substitution*). Due to constraint (5.4), quantity units of a product can only be purchased or produced in a period if setup takes place for the product in that period (*setup forcing*). Equations (5.5)–(5.6) define the domains of the variables. The LSP-SI is NP-hard. This can be shown easily by reducing the NP-hard *Uncapacitated Facility Location Problem (UFLP)* to a special case of the LSP-SI with only a single period (i.e., $T = 1$), zero initial inventory, no holding cost and no variable production cost. Also, Hsu et al. (2005) proved that SWOP, a special case of LSP-SI, is NP-hard (for a proof see Sect. 5.8).

5.3.2 Facility Location Based Reformulation

Reformulations based on analogies to facility location problems have been developed for various lot-sizing problems, see, e.g., (Bowman, 1956; Krarup and Bilde, 1977). We develop a stronger SPL-based reformulations for the LSP-SI which is based on the fact that the LSP-SI can be transformed into a *Capacitated Facility Location Problem (CFLP)* (Sridharan, 1995). This transformation resembles the reduction of RPS to an SPL problem in Geunes (2003), and is described in detail in Sect. 5.7. The reformulation enables us to solve the LSP-SI using standard MIP solvers because it significantly improves the lower bounds obtained by the LP relaxation and thereby reduces running times to an acceptable duration.

The following variable redefinitions are required for the SPL-based reformulation: We introduce decision variables y^0_{ijt} that denote the amount of initial inventory of product i used to fulfill demand of class j in period t. In addition, we need a transportation variable $y_{ijt_st_d}$ for each possible substitution $(i, j) \in E$ that denotes the quantity of product i produced in period t_s used to fulfill demand of class j in period t_d. Additional notation required for the formulation is described in Table 5.2. The values C_{it}, $c^0_{ijt_d}$, and $c_{ijt_st_d}$ are calculated as follows:

$$C_{it} = \sum_{j \in D_i} \sum_{\tau=t}^{T} d_{j\tau} \tag{5.7}$$

Table 5.2 Additional notations for SPL-based reformulation of LSP-SI

Symbol	Definition
Indices and sets	
t_s	Setup period
t_d	Demand period
Parameters	
C_{it}	Maximum total demand for product i in periods $t, \ldots, T$
$c^0_{ijt_d}$	(Negative of the) savings per unit caused by using initial inventory of product i to fulfil demand of class j in period t_d
$c_{ijt_st_d}$	"Transportation cost" per unit for fulfilling demand of class j in period t_d from a replenishment of product i in period t_s

$$c^0_{ijt_d} = \begin{cases} r_{ij} - \sum_{t=t_d}^{T} h_{it} & \text{if } (i,j) \in E \\ M & \text{otherwise} \end{cases} \tag{5.8}$$

$$c_{ijt_st_d} = \begin{cases} p_{it_s} + r_{ij} + \sum_{t=t_s}^{t_d-1} h_{it} & \text{if } t_d \geq t_s \wedge (i,j) \in E \\ M & \text{otherwise} \end{cases} \tag{5.9}$$

The objective function (5.10) of the reformulation is composed of the fixed cost, the (negative of the) savings caused by using initial inventory, and the cost for fulfilling demand by replenishments. $c^0_{ijt_d}$, the negative of the savings, is calculated by (5.8). We calculate the "transportation cost" $c_{ijt_st_d}$ for fulfilling demand of class j in period t_d from a replenishment of product i in period t_s as given in (5.9). The complete CFLP/SPL-based reformulation – referred to as LSP-SI$_{SPL}$ – is:

$$\text{Minimize } F(x, y^0, y) = \sum_{i \in P} \left(\sum_{t_s=1}^{T} f_{it_s} x_{it_s} + \sum_{j \in D_i} \sum_{t_d=1}^{T} \left(c^0_{ijt_d} y^0_{ijt_d} + \sum_{t_s=1}^{t_d} c_{ijt_st_d} \cdot y_{ijt_st_d} \right) \right) \tag{5.10}$$

subject to

$$I_{i0} \geq \sum_{j \in D_i} \sum_{t=1}^{T} y^0_{ijt} \qquad i \in P \tag{5.11}$$

$$\sum_{i \in P_j} \sum_{t_s=1}^{T} y_{ijt_st} + y^0_{ijt} = d_{jt} \qquad j \in D,\ t = 1, \ldots, T \tag{5.12}$$

$$y_{ijt_st_d} \leq d_{jt_d} \cdot x_{it_s} \qquad (i,j) \in E,\ 1 \leq t_s \leq t_d \leq T \qquad (5.13)$$

$$y^0_{ijt} \geq 0 \qquad (i,j) \in E,\ t = 1,\ldots,T \qquad (5.14)$$

$$y_{ijt_st_d} \geq 0 \qquad (i,j) \in E,\ 1 \leq t_s \leq t_d \leq T \qquad (5.15)$$

$$x_{it} \in \{0,1\} \qquad i \in P,\ t = 1,\ldots,T \qquad (5.16)$$

Here, constraint (5.11) limits the usage of initial inventory of a product i to the available quantity. Constraint (5.12) enforces that all demand is fulfilled either by initial stocks or replenishments. The disaggregated setup forcing constraints (5.13) significantly tightens the lower bound obtained by the LP relaxation: Referring to the LP relaxation of (5.10)–(5.16), it ensures that if the entire demand of a specific demand class j in a period t_d is fulfilled using only units of product i produced in period t_s, the corresponding setup variable x_{it_s} will be 1 in the optimal solution of the LP relaxation. Alternatively, one can use the weaker aggregated variant (5.17) of the setup forcing constraints:

$$\sum_{j \in D_i} \sum_{t_d = t_s}^{T} y_{ijt_st_d} \leq C_{it_s} \cdot x_{it_s} \qquad i \in P,\ t_s = 1,\ldots,T \qquad (5.17)$$

It forces the setup variable x_{it_s} for product i in period t_s to be 1 in the optimal solution of the LP relaxation if all demand in the current and consequent periods that could be covered by the lot is effectively fulfilled by it.

Instead of including all $|E|\ T^2$ disaggregated setup forcing constraints into the model, one may opt for inserting only some of them to reduce the number of constraints in the model. We will examine two such approximate extended formulations, with one and two lookahead periods, respectively (also see Stadtler, 1997):

1. An SPL-based reformulation named LSP-SI$_{SPL-A1}$ which only contains the valid inequalities of (5.13) with $t_d = t_s$. I.e., a setup variable is forced to be 1 if the production quantity fulfills the entire demand of at least one demand class in the *setup* period.
2. An SPL-based reformulation named LSP-SI$_{SPL-A2}$ which only contains the valid inequalities of (5.13) with $t_d = t_s$ and $t_d = t_s + 1$. I.e., a setup variable is forced to be 1 if the production quantity fulfills the entire demand of at least one demand class in the *setup* period or in the *consequent* period.

In these reformulations, all constraints of the aggregated variant (5.17) have to be added to the model to ensure that setup variables become 1 whenever corresponding production takes place. Otherwise, it could happen that a setup variable x_{it} is 0 though the production quantity is > 0, e.g., considering LSP-SI$_{SPL-A1}$ if $y_{ij,t,t} = 0$ but $y_{ij,t,t+1} > 0$.

5.3.3 Valid Inequalities

In this section, we introduce valid inequalities for the LSP-SI that are generalizations of the known (l, S)-cuts and setup/inventory carryover cuts for uncapacitated lot-sizing (Pochet and Wolsey, 2006, p. 218). The following valid inequalities are a generalization of the (l, S)-cuts:

$$\sum_{t \in S} q_{it} \leq \sum_{t \in S} \sum_{j \in D_i} \sum_{t'=t}^{l} d_{jt'} x_{it} + I_{il} \qquad i \in P,\ l = 1, \ldots, T,\ S \subseteq \{1, \ldots, l\} \tag{5.18}$$

A proof is given in Sect. 5.9. Taking the subset of these cuts with $l = 1, \ldots, T$ and $S = \{l\}$ results in:

$$q_{it} \leq \sum_{j \in D_i} d_{jt} x_{it} + I_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{5.19}$$

By inserting the inventory balance equation (5.2) for I_{it}, we see that (5.19) is equivalent to

$$I_{i,t-1} \geq \sum_{j \in D_i} \left(s_{ijt} - d_{jt} x_{it} \right) \qquad i \in P,\ t = 1, \ldots, T \tag{5.20}$$

Another group of valid inequalities is (5.21). It is based on the fact that, if none of the substitutes $i \in P_j$ for demand class j is set up in a period t, the total inventory carried over from the previous period of these substitutes has to cover the entire demand d_{jt}.

$$\sum_{i \in P_j} I_{i,t-1} \geq d_{jt} \left(1 - \sum_{i \in P_j} x_{it} \right) \qquad j \in D,\ t = 1, \ldots, T \tag{5.21}$$

Note that, if we consider a special case of the LSP-SI "without substitution", i.e., with $P = D$, $P_j = \{j\}$, and $D_i = \{i\}$, we see that (5.18) and (5.21) contain the well-known (l, S)-cuts and setup/inventory carryover cuts, respectively, for uncapacitated lot-sizing as a special case (see Sect. 2.2.2).

5.4 The Multi-Resource Capacitated Lot-Sizing Problem with Substitution

The Multi-Resource Capacitated Lot-Sizing Problem with Substitution (MR-CLSP-S) extends the LSP-SI by introducing multiple capacitated production resources with time-varying capacities and fixed production setup times. We develop two

Table 5.3 Notations for MR-CLSP-S

Symbol	Definition
Constants	
n_r	Number of resources
Indices and sets	
$r \in R = \{1, \ldots, n_r\}$	Resources
P_r	Set of products that are manufactured using resource r
Parameters	
K_{rt}	Capacity of resource r available in period t
l_i^p	Capacity required for setup of product i (production setup time)
κ_i^p	Capacity required for manufacturing one unit of product i
oc_{rt}	Overtime production cost for resource r in period t per unit
g_{jt}	Lost sales cost for demand class j per unit in period t
Variables	
O_{rt}	Overtime production on resource r in period t
o_{jt}	Lost sales of demand class j in period t

versions of the model, one where overtime production is possible, one where lost sales are allowed. Both the overtime and lost sales assumption avoid infeasibility of instances and make the problem easier to solve. Due to the inclusion of setup times, the feasibility problem of a MR-CLSP-S without overtime / lost sales would be NP-complete, because MR-CLSP-S contains CLSP as a special case (a proof for CLSP is given by Maes et al., 1991). Each product requires exactly one of the resources for production. We assume that only production activities consume resources. Substitution activities are assumed to be uncapacitated. The notation given in Table 5.3 is used to model the production capacity constraints.

We obtain a formulation of the MR-CLSP-S by adding the following capacity constraints to the original LSP-SI model:

$$K_{rt} + O_{rt} \geq \sum_{i \in P_r} \left(l_i^p x_{it} + \kappa_i^p q_{it} \right) \qquad r \in R,\ t = 1, \ldots, T \tag{5.22}$$

Lost sales can be included by replacing (5.3) by (5.23):

$$d_{jt} - o_{jt} = \sum_{i \in P_j} s_{ijt} \qquad j \in D,\ t = 1, \ldots, T \tag{5.23}$$

$$o_{jt} \geq 0 \qquad j \in D,\ t = 1, \ldots, T \tag{5.24}$$

A SPL-based reformulation of the MR-CLSP-S can easily be developed by adding the following capacity constraints to the SPL-based LSP-SI reformulation:

$$K_{rt_s} + O_{rt_s} \geq \sum_{i \in P_r} \left(l_i^p x_{it_s} + \kappa_i^p \sum_{j \in D_i} \sum_{t_d = t_s}^{T} y_{ijt_s t_d} \right) \qquad r \in R,\ t_s = 1, \ldots, T \tag{5.25}$$

Lost sales can also be included in the SPL-based formulation by replacing (5.12) by:

$$d_{jt} - o_{jt} = \sum_{i \in P_j} \sum_{t_s=1}^{T} y_{ijt_s t} + y_{ijt}^0 \qquad j \in D,\ t = 1, \ldots, T \tag{5.26}$$

In the original and SPL-based formulation, the overtime and lost sales cost, respectively, has to be added to the objective function:

$$F_{\text{MR-CLSP-S,Overtime}} = F_{\text{LSP-SI}} + \sum_{r \in R} \sum_{t=1}^{T} oc_{rt} O_{rt} \tag{5.27}$$

$$F_{\text{MR-CLSP-S,Lost sales}} = F_{\text{LSP-SI}} + \sum_{j \in D} \sum_{t=1}^{T} g_{jt} o_{jt} \tag{5.28}$$

5.5 Computational Experiments

In this section, we compare the original formulations of the LSP-SI and MR-CLSP-S with (a) SPL-based (approximate) extended formulations, (b) formulations with valid inequalities added a priori, and (c) formulations with valid inequalities added as cuts during the branch-and-cut algorithm. The comparison is performed by analyzing the running times of an MIP solver (ILOG CPLEX® 10.2) on various types of problem instances. The instances are not solved to optimality, but a relative MIP gap tolerance of 10^{-4} (CPLEX® default setting) is used. We examine the following research questions: (1) Which model formulation is the most efficient for the LSP-SI? What is the influence of the amount of initial inventories and the substitution structure (downward vs. general) on the hardness of the problem instances? (2) How do the running times scale for larger instances depending on the chosen formulation? (3) With regard to the MR-CLSP-S, do running times differ significantly between the model variants with lost sales vs. overtime? In addition, what is the influence of (4) the "scarcity" of resources and (5) the number of resources on the running times?

The common setup for our experimental designs is as follows: As real-world instances of the LSP-SI and MR-CLSP-S were not available, an instance generator is used to generate synthetic instances. We simultaneously vary multiple factors of this instance generator (e.g., number of products, number of periods, initial inventories and substitution structure) and then analyze the impact of these factors on the MIP solver running times. The problem instances are generated using a Common Random Numbers (CRN) technique (see Sect. 2.4.2.5), e.g., to compare running times on instances with and without initial inventories, we use the same random seeds for generating demand data and cost parameters for both types of instances. For each instance generator configuration (number of products and periods, substitution

structure, etc.), 20 replications (problem instances) are generated to ensure that the results are reliable and meaningful. ILOG CPLEX® 10.2 is used for solving the problem instances, with a time limit of 10 min and default values for all other settings. The experiments were implemented in Java™ and run on a computer with 2.4 GHz Pentium® 4 CPU, 2 GB memory and Windows® XP.

5.5.1 Problem Instances

The characteristics of the generated problem instances (such as demand distribution, substitution structure, cost parameters, and resource capacities) are chosen in a way that the instances are difficult: The substitution decisions are nontrivial, substitutions do happen in the optimal solutions and, in the case of the MR-CLSP-S, the limited production capacities influence substitutions.

We generate instances with two types of substitution structures: downward and general substitution structure. In addition, we generate instances with zero as well as positive initial inventories and uncapacitated as well as capacitated instances. The distribution assumptions of the test instances are given in Table 5.4, grouped into generator settings that all instances have in common and settings that are specific to downward/general substitution and capacitated instances.

Every instance with downward substitution has the same number of products and demand classes and every demand class corresponds to a product, i.e., $m = n$ and $P = D$. Demand class j can be fulfilled by product j and all "better" products i with $i \geq j$, i.e., $P_j = \{j, \ldots, m\}$. The unit cost is set in a way that the higher a product is in the hierarchy, the more expensive it is. In preliminary experiments, we found that this assumption makes instances significantly more difficult compared to normally distributed unit costs that were assumed by Hsu et al. (2005). The substitution cost for using a product i to fulfil demand of class j has a mean that increases with the number of "steps" in the downward substitution hierarchy, which we define as $i - j$.

For the general substitution instances, the substitution structures are created assuming that the products have two attributes (e.g., width and height). Let a_{id} denote the value of attribute d for product i. Substituting a product j by a product i is allowed if product i has a value greater than or equal to product j for both attributes, i.e., if $a_{i1} \geq a_{j1}$ and $a_{i2} \geq a_{j2}$. A practical example for such substitution structures are metal plates whose width and height can be reduced by processing steps. Thus, we assume that the unit cost is a linear function of the product of the two attributes' values, i.e., of the product's surface size if the attributes represent width a_{i1} and height a_{i2}. The numeric values in the unit cost formula were chosen in such a way that the resulting unit costs were between 25 and 40. We assume that the mean of the substitution cost increases with the "distance" between the product and its supposed substitute. We define this distance as the absolute value of the difference of the products of the attributes' values of the two products. The reasoning behind this choice of the substitution cost is that, with the attributes representing

Table 5.4 Instance generator settings

Parameter	Assumption
Common	
Demand	Normally distributed, stationary over time, but different mean μ and variation coefficient (standard deviation divided by mean) σ/μ for each demand class. Values < 0 are cut off, i.e., we repeatedly sample values until a value ≥ 0 is returned. $\mu \sim N\,(50, 20)$, $\sigma/\mu \sim N\,(0.2, 0.1)$
Setup cost	$\sim N\,(1000, 500)$ with values < 0 cut off
Holding cost	2% of the unit cost (specified below for the two instance types)
Initial inventory	0 or normal distribution with $\mu = 5\% \cdot \sum_{j \in D} \sum_{t=1}^{T} d_{jt}/\lvert P\rvert$ and $\sigma/\mu = 0.4$. Values < 0 are mapped to 0
Downward substitution	
Unit cost	$20 + 2i$, where the product index i corresponds to the position of the product in the downward substitution hierarchy
Substitution cost	Cost for using product i to fulfil demand of class $j \sim N(5, 3 + (i - j))$
General substitution	
Product width	$a_{i1} \sim U(100, 200)$
Product height	$a_{i2} \sim U(10, 20)$
Unit cost	$20 + a_{i1}a_{i2}/200$
Substitution cost	Cost for using product i to fulfil demand of class $j \sim N(5, 3 + d\,(i, j)\,/60)$, where $d\,(i, j) = \lvert a_{i1}a_{i2} - a_{j1}a_{j2}\rvert$
Capacitated instances	
Resource consumption	$\kappa_i^p \sim N\,(20, 5)$
Setup times	$l_i^p \sim N\,(200, 50)$
Capacities	$K_{rt} \sim N(\mu, \sigma)$ with $\mu = \bar{K}_r^{min} \cdot$ "capacity availability" factor, e.g., 105%, and $\sigma/\mu = 0.1$, where $\bar{K}_r^{min} = \frac{1}{T} \sum_{t=1}^{T} K_{rt}^{min}$ and K_{rt}^{min} is the minimum resource capacity required to ensure feasibility of the solution where all demand is fulfilled using its preferred product by production in the demand period
Lost sales cost	10 times as high as the production cost of the preferred product of the demand class
Overtime cost	100 per capacity unit and period

width and height, we assume that the duration and resource consumption of the conversion step are correlated with the difference in surface size of product and substitute. The numeric values in the substitution cost formula were chosen such that the resulting substitution costs were between 5 and 55.

Problem instances with capacitated production, i.e., instances of the MR-CLSP-S, are generated as follows: In instances with multiple resources, each product is randomly assigned to a resource. To generate the resource capacities, we first determine minimum capacities K_{rt}^{min} required to ensure feasibility when all demand is fulfilled using its preferred product by production in the demand period. After that, we calculate the mean $\bar{K}_r^{min}$ of K_{rt}^{min} for every resource. Then, the capacity K_{rt} is generated with a normal distribution whose mean is $\bar{K}_r^{min}$ multiplied by a "capacity availability" factor.

5.5.2 Solution Approaches

In our computational experiments for the LSP-SI, we compare the formulations LSP-SI$_{ORIG}$, LSP-SI$_{SPL}$, LSP-SI$_{SPL-A1}$, LSP-SI$_{SPL-A2}$, as well as the following alternatives with certain groups of valid inequalities added to the formulation LSP-SI$_{ORIG}$: LSP-SI$_{O-AP1}$ with (5.20) added a priori, LSP-SI$_{O-AP2}$ with (5.21) added a priori, LSP-SI$_{O-UC1}$ with (5.20) added as CPLEX® user cuts, and LSP-SI$_{O-UC2}$ with (5.21) added as CPLEX® user cuts. The CPLEX® branch-and-cut algorithm checks user cuts for violation in the optimal LP relaxation solution of every subproblem and adds them if violated. In the computational experiments for the MR-CLSP-S, we denominated the formulations analogously, i.e., MR-CLSP-S$_{ORIG}$, MR-CLSP-S$_{SPL}$, etc.

5.5.3 Experimental Designs, Results and Interpretation

In the following, we examine five experimental designs: Sect. 5.5.3.1 describes a design with small uncapacitated instances, Sect. 5.5.3.2 a design with larger uncapacitated instances. Section 5.5.3.3 contains a design with capacitated instances, one group of them with lost sales, the other one with overtime. The design in Sect. 5.5.3.4 serves to analyze the effect of the tightness of production capacities on the hardness of instances. Finally, the design in Sect. 5.5.3.5 compares the running times for instances with 1, 2, and 3 resources.

5.5.3.1 LSP-SI: Influence of Substitution Structure, Initial Inventories and Problem Size

In experiment 1 we generated downward and general substitution LSP-SI instances. We analyzed both instances with initial inventories $= 0$ and $= 5\%$ of total demand (InI = "No" and "Yes", respectively). The number of periods was varied from 8, 10, to 12. For downward substitution instances, we varied the number of products (demand classes) from 6, 8, to 10, for general substitution instances from 12 to 16. We chose to consider general substitution instances with a larger number of products than the downward substitution instances for two reasons: First, general substitution instances turned out to be somewhat easier to solve, presumably because their substitution graphs often consist of several (unconnected) components. Second, our opinion was that settings with downward substitution and more than a medium number of products, e.g., 10, are infrequent in practice. In both experiments, we compared all eight solution approaches for the LSP-SI mentioned above.

We observed that the SPL-based formulations were clearly superior to all other formulations, regardless of problem size, substitution structure, and initial inventories. A comparison of the running times on downward and general substitution instances is given in Table 5.5. Here, the abbreviation "DW" stands for downward

Table 5.5 Median running times on LSP-SI instances in seconds (experiment 1)

Type	InI	$m = n$	T	ORIG	O-AP1	O-UC1	O-AP2	O-UC2	SPL	SPL-A1	SPL-A2
DW	Yes	6	8	0.37	0.70	0.42	0.34	0.38	0.20	0.38	0.36
			10	1.11	1.73	1.25	0.97	1.22	0.20	0.52	0.48
		8	8	1.90	1.84	1.94	1.88	1.93	0.37	0.53	0.54
			10	51.22	47.40	51.26	26.55	47.68	0.30	0.84	0.77
		10	8	14.95	13.29	16.07	11.73	16.38	0.33	0.59	0.57
			10	>600	>600	>600	>600	>600	0.76	1.23	1.16
	No	6	8	0.23	0.26	0.27	0.23	0.24	0.06	0.20	0.20
			10	0.61	1.09	0.70	0.58	0.60	0.08	0.30	0.30
		8	8	0.72	1.15	0.77	0.76	0.81	0.09	0.27	0.27
			10	11.91	12.21	12.48	11.19	14.38	0.11	0.48	0.44
		10	8	5.07	3.74	5.90	3.91	4.95	0.08	0.27	0.27
			10	359.40	186.68	453.70	313.58	527.04	0.12	0.51	0.55
GE	Yes	12	8	0.51	0.45	0.49	0.45	0.46	0.22	0.33	0.34
			10	52.73	16.63	51.86	22.17	42.97	0.30	0.53	0.54
		16	8	43.33	21.77	37.99	27.97	37.46	0.51	0.69	0.70
			10	>600	>600	>600	>600	>600	0.78	0.90	1.09
	No	12	8	0.23	0.22	0.24	0.22	0.23	0.06	0.16	0.16
			10	10.55	3.44	12.21	5.21	10.65	0.10	0.29	0.30
		16	8	5.49	3.06	5.48	2.88	5.31	0.13	0.29	0.30
			10	408.20	359.24	386.61	475.04	407.43	0.20	0.51	0.55

Table 5.6 Percentage of LSP-SI instances solved within 10 min (experiment 1)

Type	InI	$m = n$	T	ORIG (%)	O-AP1 (%)	O-UC1 (%)	O-AP2 (%)	O-UC2 (%)
DW	Yes	10	10	20	20	20	20	20
	No	10	10	65	85	55	65	55
GE	Yes	12	8	100	100	100	100	100
			10	90	95	85	85	90
		16	8	80	100	80	95	80
			10	30	25	30	25	25
	No	12	8	100	100	100	100	100
			10	90	95	90	95	90
		16	8	100	100	100	100	100
			10	55	55	55	55	60

substitution and "GE" for general substitution instances. Each table cell contains the median calculated over 20 replications of the instance type. We chose to calculate median values because we could not determine average running times in those cases where some instances were not solved within the time limit. Because the number of replications is even, the median is calculated as $med = 1/2\left(rt_{(10)} + rt_{(11)}\right)$, where $rt_{(k)}$ is the running time at position k in the list of running times in ascending order. Thus, we can only calculate the median exactly if $rt_{(10)}$ and $rt_{(11)}$ are below the time limit. Otherwise, we only know that the true median without time limit would be $>med$. Table 5.6 contains the percentages of instances solved within the time limit of 10 min. Here, we only report percentages for downward substitution instances with 10 products and 10 periods, as all smaller downward substitution instances were solved within the time limit by all formulations. The average optimality gap for instances that could not be solved within 10 min was 0.82%. The percentages of the SPL-based formulations were excluded as these solved all instances within the time limit.

In general, running times on instances with initial inventories were somewhat higher than those on instances with no initial inventories. One possible explanation for this effect is that the existence of initial inventories might make it harder for the MIP solver to generate effective cuts. Referring to the SPL-based reformulation, it is obvious that less setup variables are fixed to 1 by the disaggregated setup forcing constraints if parts of period demands are fulfilled from initial inventories. Adding the valid inequalities to the original formulation (LSP-SI$_{O-AP1}$ and LSP-SI$_{O-AP2}$) seems to improve running times slightly compared to LSP-SI$_{ORIG}$. Yet LSP-SI$_{ORIG}$, LSP-SI$_{O-AP1}$, and LSP-SI$_{O-AP2}$ could not compete with the SPL-based formulations: Their running times quickly exceeded the time limit for instances with 10 or more periods and a moderate number of products (10 in the case of downward substitution, 16 in the case of general substitution). In contrast, most of the running times of LSP-SI$_{SPL}$ were below 1 s. In addition, we found that the running times for the approximate extended formulations LSP-SI$_{SPL-A1}$

and LSP-SI$_{SPL-A2}$ were between two and four times worse than those of the full SPL-based formulation LSP-SI$_{SPL}$, but most of them were also below 1 s and thus by far better than those of LSP-SI$_{ORIG}$. In the next experiment, we examined whether this also holds true for larger problem instances.

5.5.3.2 Larger LSP-SI Instances: Approximate Extended Formulations

In the second experiment, we compared the running times of LSP-SI$_{SPL}$, LSP-SI$_{SPL-A1}$, and LSP-SI$_{SPL-A2}$ on larger instances with general substitution structure, both with and without initial inventories. The number of periods was varied from 10, 15, to 20 and the number of products (demand classes) from 20, 30, to 40.

A comparison of the running times on these instances is given in Table 5.7. The percentages of instances solved within the time limit are stated in Table 5.8. Here, we only report percentages for instances with 30 and 40 products and 20 periods, as all smaller instances were solved within the time limit by the considered formulations. The average optimality gap for instances that could not be solved within 10 min was 0.12%. We found that the running times of LSP-SI$_{SPL}$ were the shortest. Those of LSP-SI$_{SPL-A1}$ and LSP-SI$_{SPL-A2}$ were in most cases slightly worse, in some cases marginally better for instances with initial inventory, while these approximate extended formulations contained a much smaller number of (disaggregated)

Table 5.7 Median running times on large general substitution LSP-SI instances in seconds (experiment 2)

InI	$m=n$	T	SPL	SPL-A1	SPL-A2
Yes	20	10	1.39	1.90	1.95
		15	4.73	5.91	6.22
		20	9.48	11.22	12.50
	30	10	4.41	5.85	5.92
		15	15.85	19.98	22.26
		20	38.84	31.77	36.96
	40	10	9.16	13.12	13.67
		15	39.41	38.66	35.73
		20	129.86	127.63	150.58
No	20	10	0.3	0.85	0.86
		15	0.77	2.76	2.95
		20	1.45	5.32	5.61
	30	10	0.72	2.19	2.38
		15	1.81	6.85	7.25
		20	3.34	12.11	13.78
	40	10	1.38	4.44	4.71
		15	3.68	12.31	13.59
		20	7.73	26.5	27.7

Table 5.8 Percentage of larger general substitution LSP-SI instances solved within 10 min (experiment 2)

InI	$m = n$	T	SPL (%)	SPL-A1 (%)	SPL-A2 (%)
Yes	30	20	100	95	95
	40	20	90	90	90
No	30	20	100	100	100
	40	20	100	100	100

constraints. For instances without initial inventory, the formulations LSP-SI$_{SPL-A1}$ and LSP-SI$_{SPL-A2}$ were approximately four times slower than LSP-SI$_{SPL}$, but all instances could be solved within the time limit using either of the three formulations. Both on instances with and without initial inventory, LSP-SI$_{SPL-A2}$ seems to perform slightly worse than LSP-SI$_{SPL-A1}$.

5.5.3.3 MR-CLSP-S: Lost Sales vs. Overtime

In experiment 3, we focussed on capacitated instances. We generated one group of MR-CLSP-S instances with lost sales and another group with overtime. Within each of the two groups, we created instances with downward as well as general substitution structure. We then compared the running times of MR-CLSP-S$_{ORIG}$, MR-CLSP-S$_{SPL}$, MR-CLSP-S$_{SPL-A1}$, and MR-CLSP-S$_{SPL-A2}$ for these instances. Common settings for all instances were a number of eight periods, initial inventories $= 5\%$ of total demand, a single production resource ($n_r = 1$) and a capacity availability factor of 120%. For downward substitution instances, the number of products (demand classes) was varied from 6, 8, to 10, for general substitution instances from 8, 12, to 16.

A comparison of the running times on these instances is given in Table 5.9. The SPL-based formulations solved all instances within the time limit. The percentages of instances solved within the time limit by the original formulation are stated in Table 5.10. The average optimality gap for instances that could not be solved within 10 min was 1.73%. We observed that the SPL-based formulations MR-CLSP-S$_{SPL}$, MR-CLSP-S$_{SPL-A1}$, MR-CLSP-S$_{SPL-A2}$ did not dominate the original formulation MR-CLSP-S$_{ORIG}$ as clearly as it was the case for the (uncapacitated) LSP-SI. On downward substitution instances with six products, the running times of the SPL-based formulations and MR-CLSP-S$_{ORIG}$ were almost the same. Yet on instances with more products, the SPL-based formulations performed better than MR-CLSP-S$_{ORIG}$. This observation is in line with Denizel and Süral (2006) and Alfieri et al. (2002), who observed that CLSP reformulations performed better than standard formulations for LP-based heuristics. The running times for instances with lost sales and overtime were very similar.

Table 5.9 Median running times on MR-CLSP-S instances with lost sales / overtime in seconds (experiment 3)

Type	Lost sales	Overtime	$m = n$	ORIG	SPL	SPL-A1	SPL-A2
DW	No	Yes	6	4.50	2.20	2.28	2.45
			8	40.27	3.22	3.38	3.91
			10	>600	6.44	7.05	6.34
	Yes	No	6	3.20	2.27	2.38	2.61
			8	42.14	3.45	3.41	4.06
			10	>600	6.09	8.40	8.66
GE	No	Yes	8	8.73	2.21	2.77	2.70
			12	>600	12.39	13.06	12.31
			16	>600	19.78	18.20	18.75
	Yes	No	8	8.78	2.71	3.32	2.88
			12	>600	13.19	14.55	13.63
			16	>600	16.23	22.87	22.74

Table 5.10 Percentage of MR-CLSP-S instances with lost sales / overtime solved within 10 min (experiment 3)

Type	Lost sales	Overtime	$m = n$	ORIG (%)
DW	No	Yes	6	100
			8	95
			10	25
	Yes	No	6	100
			8	100
			10	40
GE	No	Yes	8	100
			12	35
			16	0
	Yes	No	8	100
			12	20
			16	0

5.5.3.4 MR-CLSP-S: Scarcity of Production Capacity

In experiment 4, we generated MR-CLSP-S instances with three levels of the capacity availability factor, namely 105%, 120%, and 150%. We compared the running times of MR-CLSP-S$_{ORIG}$, MR-CLSP-S$_{SPL}$, MR-CLSP-S$_{SPL-A1}$, and MR-CLSP-S$_{SPL-A2}$ on MR-CLSP-S instances with downward and general substitution structure. We only analyzed instances with overtime, as their running times were very similar to those of instances with lost sales in experiments 4 and 5. Common settings for all instances were a number of eight periods, a single production resource and initial inventories $=$ 5% of total demand. For downward substitution instances, the number of products (demand classes) was varied from 6, 8, to 10, for general substitution instances from 8, 12, to 16.

Table 5.11 Median running times on MR-CLSP-S instances with different capacity availability levels in seconds (experiment 4)

Type	Capacity availability	$m = n$	ORIG	SPL	SPL-A1	SPL-A2
DW	1.05	6	3.91	1.78	1.74	1.77
		8	132.72	4.92	5.25	5.07
		10	>600	15.91	12.55	14.45
	1.2	6	3.09	1.43	1.69	1.61
		8	91.95	4.02	4.05	4.12
		10	>600	7.34	8.31	7.95
	1.5	6	1.20	0.89	0.98	0.98
		8	28.34	1.73	1.80	1.96
		10	>536.73	3.34	3.07	3.23
GE	1.05	8	19.37	3.52	3.67	3.37
		12	>600	12.88	15.73	13.36
		16	>600	27.73	24.03	24.17
	1.2	8	8.81	2.13	2.76	2.70
		12	>600	12.28	14.90	12.35
		16	>600	19.66	18.22	18.63
	1.5	8	2.28	0.98	1.23	1.20
		12	140.22	2.41	3.63	3.73
		16	>600	7.69	7.79	8.28

A comparison of the running times on these instances is given in Table 5.11. The SPL-based formulations solved all instances within the time limit. The percentages of instances solved within the time limit by the original formulation are stated in Table 5.12. The average optimality gap for instances that could not be solved within 10 min was 1.67%. Our finding was that running times of all considered formulations were significantly higher for instances with scarcer production capacity, i.e., with lower capacity availability value. Compared to MR-CLSP-S$_{ORIG}$, the running times of the SPL-based formulations grew slower depending on the scarcity of the production resource.

5.5.3.5 MR-CLSP-S: Number of Resources

In experiment 5, we generated MR-CLSP-S instances with 1, 2, and 3 resources. The running times of MR-CLSP-S$_{ORIG}$, MR-CLSP-S$_{O-AP1}$, MR-CLSP-S$_{O-AP2}$, MR-CLSP-S$_{SPL}$, and the approximate extended formulations MR-CLSP-S$_{SPL-A1}$ and MR-CLSP-S$_{SPL-A2}$ were compared on MR-CLSP-S instances with downward and general substitution structure. Common settings for all instances were a number of eight periods, initial inventories = 5% of total demand, and a capacity availability factor of 120%. We only analyzed instances with overtime. For downward substitution instances, the number of products (demand classes) was varied from 6, 8, to 10, for general substitution instances from 8, 12, to 16.

Table 5.12 Percentage of MR-CLSP-S instances with different capacity availability levels solved within 10 min (experiment 4)

Type	Capacity availability	$m = n$	ORIG (%)
DW	1.05	6	100
		8	90
		10	20
	1.2	6	100
		8	100
		10	30
	1.5	6	100
		8	100
		10	50
GE	1.05	8	100
		12	0
		16	0
	1.2	8	100
		12	35
		16	0
	1.5	8	100
		12	80
		16	5

A comparison of the running times on these instances is given in Table 5.13. The percentages of instances solved within the time limit are stated in Table 5.14. The average optimality gap for instances that could not be solved within 10 min was 1.29%. With regard to downward substitution instances, we made the following observation: When the number of resources was 1, the SPL-based formulations were clearly better than MR-CLSP-S$_{ORIG}$, MR-CLSP-S$_{O-AP1}$, and MR-CLSP-S$_{O-AP2}$. Yet, when we considered instances with more resources (e.g., 3), the running times of MR-CLSP-S$_{ORIG}$, MR-CLSP-S$_{O-AP1}$, and MR-CLSP-S$_{O-AP2}$ decreased, while the median running times of the SPL-based formulations increased. The running times of MR-CLSP-S$_{ORIG}$, MR-CLSP-S$_{O-AP1}$, and MR-CLSP-S$_{O-AP2}$ were similar to or better than those of the SPL-based formulations for $n_r > 1$ in most cases.

This seemingly surprising observation is caused by the automatic cut generation of CPLEX®: On instances with multiple resources, CPLEX® is able to generate a large number of cuts for MR-CLSP-S$_{ORIG}$ already at the branch-and-bound root node that significantly sharpen the lower bound (LB). In contrast, CPLEX® generates only few cuts for MR-CLSP-S$_{SPL}$. In fact, the LB obtained for MR-CLSP-S$_{ORIG}$ after cut generation was better that the LB obtained from the LP relaxation of MR-CLSP-S$_{SPL}$ in some cases. E.g., when examining an instance with $n_r = 3$ and $m = 8$, we observed the following (the data is summarized in Table 5.15): With MR-CLSP-S$_{ORIG}$, the optimal LP relaxation objective at the root node was 80,063.6 before and 88,172.0 after cut generation, i.e., the LB was significantly sharpened by the cuts added. With MR-CLSP-S$_{SPL}$, the optimal

Table 5.13 Median running times on MR-CLSP-S instances with different numbers of resources in seconds (experiment 5)

Type	n_r	$m = n$	ORIG	O-AP1	O-AP2	SPL	SPL-A1	SPL-A2
DW	1	6	3.22	2.43	2.71	1.45	1.70	1.63
		8	92.44	56.15	58.48	4.05	4.11	4.05
		10	>600	>600	>594.73	6.45	7.08	6.55
	2	6	2.00	2.21	2.51	8.58	8.67	10.32
		8	56.64	43.21	40.77	21.47	27.05	23.48
		10	>589.62	309.91	>495.66	35.91	39.78	36.93
	3	6	1.32	1.36	1.18	9.29	9.09	10.77
		8	27.16	21.24	33.44	46.97	54.59	51.38
		10	247.68	203.11	201.91	186.00	383.70	350.49
GE	1	8	9.03	3.73	5.79	2.27	2.80	2.71
		12	>600	415.97	481.62	12.41	12.92	12.25
		16	>600	>600	>600	17.05	18.30	19.70
	2	8	9.51	7.18	8.45	8.08	9.20	8.30
		12	>600	>600	>600	32.05	39.97	41.16
		16	>600	>600	>600	194.18	235.38	239.40
	3	8	10.09	7.37	11.71	22.75	22.82	25.03
		12	>600	>600	>600	178.13	131.50	188.81
		16	>600	>600	>600	>600	>600	>600

Table 5.14 Percentage of MR-CLSP-S instances with different numbers of resources solved within 10 min (experiment 5)

Type	n_r	$m = n$	ORIG (%)	O-AP1 (%)	O-AP2 (%)	SPL (%)	SPL-A1 (%)	SPL-A2 (%)
DW	1	6	100	100	100	100	100	100
		8	100	100	100	100	100	100
		10	25	45	50	100	100	100
	2	6	100	100	100	100	100	100
		8	100	95	100	100	100	100
		10	50	60	50	100	100	100
	3	6	100	100	100	100	100	100
		8	100	100	100	90	90	95
		10	60	65	65	70	65	65
GE	1	8	100	100	100	100	100	100
		12	35	65	55	100	100	100
		16	0	0	0	100	100	100
	2	8	100	100	100	100	100	100
		12	25	35	45	100	100	100
		16	0	0	0	85	80	85
	3	8	100	100	100	100	90	95
		12	25	30	35	85	80	75
		16	0	0	0	40	40	35

Table 5.15 Effect of MIP solver cut generation on tightness of lower bounds and running times of original and SPL formulation, examined on two instances with $n_r = 1$ and 3

n_r	Data	ORIG	ORIG, no cuts	SPL	SPL, no cuts
1	Running time (s)	48.5	>600	2.9	2.5
	LB before cuts	79,985.7	79,985.7	87,309.5	87,309.5
	LB after cuts	84,532.7	-	87,447.6	-
	# cuts at root	326	0	37	0
3	Running time (s)	10.8	>600	97.4	112.4
	LB before cuts	80,063.6	80,063.6	86,816.5	86,816.5
	LB after cuts	88,172.0	-	87,574.6	-
	# cuts at root	300	0	55	0

LP relaxation objective at the root node was 86,816.5 before and 87,574.6 after cut generation, i.e., the cuts added at the root node did not sharpen the LB as much as for MR-CLSP-S$_{ORIG}$. The LB obtained for MR-CLSP-S$_{ORIG}$ was even better than the LB obtained for MR-CLSP-S$_{SPL}$. For instances with $n_r = 1$, we did not observe this effect. Considering an instance with $n_r = 1$ and the same CRN as the previous instance, we found that the cuts added by CPLEX® improved the LB significantly for MR-CLSP-S$_{ORIG}$, but the LB for MR-CLSP-S$_{SPL}$ was better. With cut generation turned off, (i.e., with a pure branch&bound algorithm), MR-CLSP-S$_{SPL}$ performed better than MR-CLSP-S$_{ORIG}$. Using CPLEX® with cuts turned off, MR-CLSP-S$_{ORIG}$ exceeded the time limit for both $n_r = 1$ and 3, whereas running times changed only slightly for MR-CLSP-S$_{SPL}$. These results suggest that, though reformulations like the SPL or shortest-path formulation of lot-sizing problems usually yield sharper LBs, it depends on the cut generation and other features of the MIP solver used whether the usage of reformulations actually decreases running times. Similarly, Stadtler (1996) observed that the original inventory and lot-size ML-CLSP formulation works better than proposed reformulations on one set of test instances.

When considering instances with general substitution and more than one resource, we found for $n_r = 2$ and $m = n = 8$ that the SPL-based formulations yielded running times similar to LSP-SI$_{ORIG}$, LSP-SI$_{O-AP1}$, and LSP-SI$_{O-AP2}$. However, on general substitution instances with a larger number of products ($m = n = 12$), the SPL-based formulations were clearly better than LSP-SI$_{ORIG}$, LSP-SI$_{O-AP1}$, and LSP-SI$_{O-AP2}$, in contrast to our observations for downward substitution instances.

5.6 Conclusions

In this section, we described SPL-based reformulations for two dynamic lot-sizing problems with product substitution – the uncapacitated LSP-SI and the capacitated MR-CLSP-S. We exemplified the problem that net demands cannot be calculated exactly if product substitutions are possible. In addition, we developed two classes

of valid inequalities for uncapacitated lot-sizing with product substitution. We compared the original formulations of the LSP-SI and MR-CLSP-S to SPL-based (approximate extended) reformulations and formulations to which the new classes of valid inequalities were added. The comparison was performed by computational experiments with an MIP solver on generated instances. A main finding was that the SPL-based formulations of the LSP-SI dominated the original formulation as well as the formulations with valid inequalities added in terms of running times in most cases. Adding the valid inequalities to the model seems to slightly improve running times. In our experiments on the MR-CLSP-S, we found that instances with lost sales and overtime required very similar running times. The scarcer the production capacity was, the higher the running times. For all instances with one production resource, the SPL-based formulations clearly dominated the original formulation. Yet, when increasing the number of resources, the distance between the SPL-based formulations and the original formulation shortened: On instances with downward substitution and more than one resource, the running times of the SPL-based reformulations were similar to or even worse than those of the original formulation. Both for the LSP-SI and MR-CLSP-S, running times on SPL-based approximate extended formulations – which only contain a subset of the disaggregated constraints and thus are significantly smaller – could compete with those of complete SPL-based reformulations.

5.7 Transformation of LSP-SI into a CFLP

We transform the LSP-SI into a *Capacitated Facility Location Problem (CFLP)* (Sridharan, 1995). This transformation illustrates the SPL-based reformulation of the problem and enables us to solve the LSP-SI using CFLP algorithms. The notation given in Table 5.16 is used for the CFLP (for a formulation see, e.g., Sridharan, 1995). To distinguish the CFLP symbols from LSP-SI symbols, the superscript w was added to the former.

The analogies between the entities and model elements in the LSP-SI and the CFLP resulting from the transformation are summarized in Table 5.17.

CFLP Structure, Capacities and Demands

We create the corresponding CFLP instance for a given LSP-SI instance as follows: In the CFLP, each pair of demand class j and period t forms a customer k with

Table 5.16 Notations for LSP-SI → CFLP transformation

Symbol	Definition
Parameters	
a_h^w	Capacity of warehouse h

Table 5.17 Analogies between LSP-SI and CFLP elements in the transformation

LSP-SI	CFLP
Product setup in a period	Opening of a warehouse
Fixed setup cost of a product in a specific period	Fixed opening cost of a warehouse
Demand for a product in a specific period	Demand of a customer
Initial inventory of a product	Capacity of a warehouse
Unit conversion cost − holding cost saved (for initial inventory); Unit production + conversion + holding cost (for replenishments)	Unit transportation cost from a warehouse to a customer

index $k(j,t) = (t-1) \cdot n + j$, whose demand $d^w_{k(j,t)}$ is equal to the demand d_{jt} of class j in period t. The CFLP contains a warehouse h with index $h(i) = i$ for each product i representing the initial stock of the product, with a capacity a^w_i equal to the initial inventory I_{i0}. If a warehouse $i \in P$ delivers $y^w_{i;t\cdot n+j}$ quantity units to customer $t \cdot n + j$, this means for the LSP-SI that the same quantity of initial inventory of product i is used to satisfy demand of class j in period t. Moreover, we introduce a warehouse for each pair of product i and period t, whose index is $h(i,t) = t \cdot m + i$. Its opening decision corresponds to the decision whether product i should be set up in period t. As there is no limit on lot sizes in the LSP-SI, we assign a sufficiently large M to its capacity $a^w_{h(i,t)}$, e.g., $M = C_{it}$ [see (5.7)]. A transportation quantity $y^w_{h(i,t_s);k(j,t_d)}$ in the CFLP denotes for the LSP-SI that this quantity from the lot of product i set up in period t_s is used to satisfy demand of class j in period t_d.

Cost Parameters

The CFLP cost data are chosen in such a way that the CFLP objective equals the LSP-SI objective minus a constant. This constant is the sum of holding cost H_{I_0} that would be caused by keeping all initial inventory in stock without consuming any of it till the end of the planning horizon:

$$H_{I_0} = \sum_{i \in P} \sum_{t=1}^{T} h_{it} I_{i0} \tag{5.29}$$

To ensure that $F_{CFLP} = F_{LSP-SI} - H_{I_0}$, we choose the warehouses' opening cost vector f^w and the transportation cost matrix c^w as follows: The opening cost of the first m warehouses $i \in P$, whose capacities correspond to the products' initial inventories, is set to 0. The unit transportation cost of the warehouses $i \in P$ to a customer $k(j,t_d)$ equals the negative of the saving that we achieve by taking one unit of initial inventory from stock in period t_d to satisfy demand of class j. For feasible conversions, this saving is the holding cost per unit of product i from period

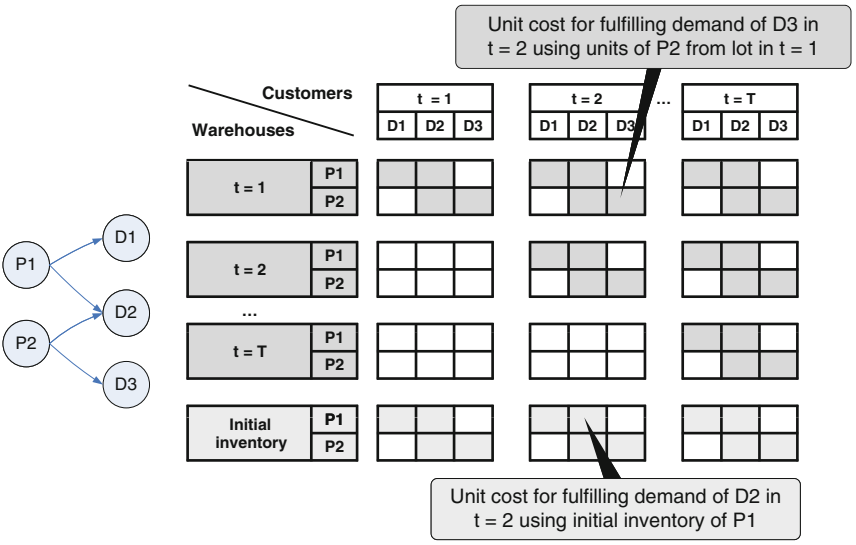


Fig. 5.1 Example for the transformation of the LSP-SI into a CFLP

t_d to T minus the additional cost r_{ij} for converting the item to accommodate demand of class j. The corresponding formula is given by (5.8), i.e., we set $c^w_{i;k(j,t_d)} = c^0_{ijt_d}$.

Considering the choice of the transportation cost from warehouse $t_s \cdot m + i$ to customer $k(j, t_d)$, we have to forbid (a) infeasible conversions with $(i, j) \notin E$ and (b) fulfillment of demand from future periods to previous periods ($t_d < t_s$) since backlogging is not allowed. This explains the two cases in (5.9). We set $c^w_{h(i,t_s);k(j,t_d)} = c_{ijt_st_d}$. Thus, the transportation cost is the sum of production/order unit cost p_{it_s}, substitution cost r_{ij} and the total holding cost from the replenishment arrival period t_s till the demand occurrence period t_d.

The LSP-SI $\rightarrow$ CFLP transformation is illustrated with an example in Fig. 5.1. The example is based on a substitution graph with two products and three demand classes. The matrix on the right side of Fig. 5.1 depicts the cost matrix of the corresponding CFLP with $2 \cdot (T + 1)$ warehouses (rows) and $3 \cdot T$ customers (columns). White cells correspond to forbidden transportation links, gray cells to allowed transportation links that denote feasible substitutions.

5.8 Proof that LSP-SI is NP-Hard

The LSP-SI is strongly NP-hard. We prove this by reducing the *Uncapacitated Facility Location Problem (UFLP)* [also: *Simple Plant Location Problem (SPLP)*], which is known to be strongly NP-hard, to a special case of the LSP-SI. Given an instance of UFLP, we reduce it to a special case of LSP-SI with only a single period (i.e., $T = 1$), zero initial inventory, no holding cost and no variable production cost.

We show that this special case is strongly NP-hard using the following mapping: Products of the LSP-SI represent warehouses and demand classes represent customers of the UFLP, i.e., $P = W$ and $D = C$. The setup of products corresponds to the opening of warehouses in the UFLP. Substitution decisions correspond to transportation decisions in the UFLP.

The parameters of the LSP-SI are set as follows: All initial inventories I_{h0}, holding cost h_{h1} and unit order/production cost p_{h1} are 0. The number of products m equals the number of warehouses m^w, the number of demand classes n is identical with the number of customers n^w. The product setup cost is equal to the fixed cost for the corresponding warehouse, i.e., $f_{h1} = f_h^w$. Substitution from each product h to all demand classes in D has to be feasible because each warehouse can deliver to any of the customers. Hence, $E = P \times D$. The unit cost for converting product h to demand class k equals the transportation cost from warehouse h to customer k.

To transform a LSP-SI solution into a solution for the respective UFLP, the setup variable value x_{h1} of product h in period 1 is assigned to x_h^w denoting the construction of warehouse i. Furthermore, we obtain the transportation quantity y_{hk}^w from s_{hk1}. It is easy to see that the optimal LSP-SI solution yields a corresponding optimal UFLP solution, as the constraints, variable domains and objective functions are equivalent. Since the depicted reduction of UFLP to a special LSP-SI is obviously polynomial and UFLP is strongly NP-hard, this shows that LSP-SI is NP-hard in the strong sense.

5.9 Proof for Product Substitution (l,S)-Cuts

To prove that (5.18) is class of valid inequalities, we show that

$$\sum_{t \in S} q_{it} \leq \sum_{t \in S} \sum_{j \in D_i} \sum_{t'=t}^{l} d_{jt'} x_{it} + I_{il} \tag{5.30}$$

holds for all $i \in P$, $l = 1, \ldots, T$, $S \subseteq \{1, \ldots, l\}$ if (x, q, I, s) is an integer feasible solution of a LSP-SI instance. Considering a certain integer feasible solution (x, q, I, s), we define T_i as the set of periods in which product i is produced: $T_i = \{t \mid x_{it} = 1\}$. As $q_{it} = 0$ and $x_{it} = 0$ for all $t \notin T_i$, (5.30) is equivalent to

$$\sum_{t \in S \cap T_i} q_{it} \leq \sum_{t \in S \cap T_i} \sum_{j \in D_i} \sum_{t'=t}^{l} d_{jt'} + I_{il} \tag{5.31}$$

If $S \cap T_i = \emptyset$, (5.31) obviously holds true because the left side of the inequality is 0 as all q_{it} are 0. For the case that $S \cap T_i \neq \emptyset$, with t_i defined as the first period in S when a lot of product i is produced, i.e., $t_i = \min\{t \mid t \in S \cap T_i\}$, we obtain by recursively inserting the inventory balance (5.2) into itself $l - t_i$ times that:

$$I_{il} = I_{i,t_i-1} + \sum_{t=t_i}^{l} q_{it} - \sum_{t=t_i}^{l} \sum_{j \in D_i} s_{ijt} \tag{5.32}$$

Inserting (5.32) for I_{il} in (5.31) yields:

$$\sum_{t \in S \cap T_i} q_{it} + \sum_{j \in D_i} \sum_{t=t_i}^{l} s_{ijt} \leq \sum_{t=t_i}^{l} q_{it} + \sum_{j \in D_i} \sum_{t \in S \cap T_i} \sum_{t'=t}^{l} d_{jt'} + I_{i,t_i-1} \tag{5.33}$$

We prove (5.33) by showing that the following two inequalities hold true ($0 \leq I_{i,t_i-1}$ obviously applies because the variables are nonnegative):

$$\sum_{t \in S \cap T_i} q_{it} \leq \sum_{t=t_i}^{l} q_{it} \tag{5.34}$$

$$\sum_{j \in D_i} \sum_{t=t_i}^{l} s_{ijt} \leq \sum_{j \in D_i} \sum_{t \in S \cap T_i} \sum_{t'=t}^{l} d_{jt'} \tag{5.35}$$

(5.34) holds true because $S \cap T_i \subseteq \{t_i, \ldots, l\}$. Equation (5.35) is fulfilled because $t_i \in S \cap T_i$ holds by definition and $s_{ijt} \leq d_{jt}$ due to (5.3), as these propositions imply the first $\leq$ in:

$$\sum_{j \in D_i} \sum_{t=t_i}^{l} s_{ijt} \leq \sum_{j \in D_i} \sum_{t \in S \cap T_i} d_{jt} \leq \sum_{j \in D_i} \sum_{t \in S \cap T_i} \sum_{t'=t}^{l} d_{jt'} \tag{5.36}$$

The second $\leq$ in (5.36) holds true because the left side is contained in the right side and all addends are positive. Thus, (5.33) holds true which proves that (5.18) is a class of valid inequalities for the LSP-SI.

Chapter 6
MIP-Based Heuristics for Capacitated Lot-Sizing with Sequence-Dependent Setups and Substitutions

6.1 Introduction

In this section, we consider a single-level capacitated lot-sizing problem with substitutions and sequence-dependent setups. The model was designed to map the industrial optimization problem in windshield interlayer production planning described in Sect. 3.1. Rather than building a specific model for this application, we aimed at devising a model and appropriate solution approach for a more general model that can capture the characteristics of this application as well as those of similar production planning problems. Why should it make sense to consider substitutions and sequence-dependent changeovers in one model, rather than treating each of the two aspects in separate subproblems that decompose the overall problem? The idea is that substitutions affect the optimal production sequencing and vice versa, as it might be beneficial to save setup times by refraining from producing certain products and substituting them by others, at least in some settings. This reduces the time spent with "worthless" changeovers, and thereby increases the total capacity available for production.

Figure 6.1 illustrates this idea: Assume that an initial production sequence as shown exists. As visible, the remaining unused capacity in period shown as a dark gray box is rather small. Setup times are assumed to be sequence-dependent. Also, we assume that product 4 can be substituted by product 1. The box $3 \rightarrow 2$ denotes that a changeover from product 3 to product 2 takes place. In a first step, a better sequence can be obtained by moving product 2, which was the second product in the sequence in period t, to the end of the sequence. Due to the sequence-dependent setup times, this could result in a solution with a smaller total setup time and larger remaining unused capacity as shown in the second sequence. In a second step, one could exploit the substitution option by removing the setup of product 4 and instead producing more of product 1 to fulfill the demand for product 4. However, this change might increase the total cost due to substitution costs. If the substitution ratio is 1:1 and the setup time between products 1 and 4 greater than 0, the change further decreases the total setup time and increases the capacity available for production. One could now, for instance, initiate a setup to another product and start producing it at the end of period t, as the resource would otherwise be idle in this time window.

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_6,

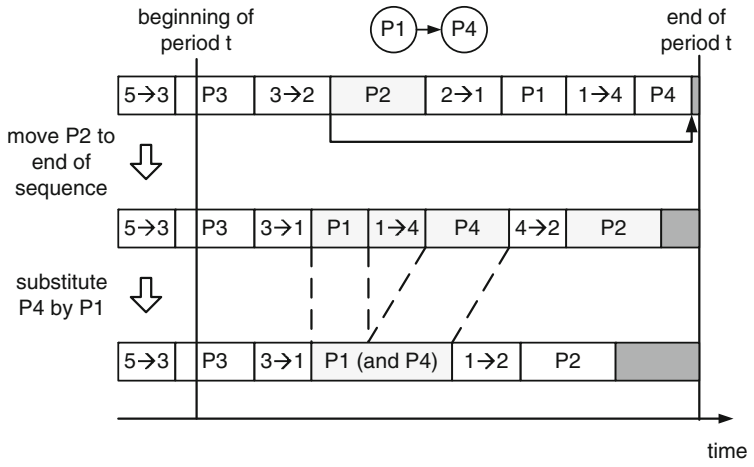


Fig. 6.1 Example – considering substitutions when choosing production sequences

6.2 Outline

The remainder of this chapter is structured as follows: The assumptions and mathematical formulation of the considered model – the Capacitated Lot-sizing problem with Sequence-Dependent setups and Substitutions (CLSD-S) – are contained in Sect. 6.3. Section 6.4 deals with the task of obtaining efficient or at least good production sequences for the model. Such sequences will be used to construct initial solutions for F&O heuristics. Section 6.5 describes several variants of R&F and F&O heuristics for the CLSD-S. The chapter starts by introducing notations for defining subproblems of several CLSD-S decompositions in Sect. 6.5.1. Time-oriented, product-oriented and substitute-oriented CLSDS-S decompositions are described in Sect. 6.5.2. Sections 6.5.3 and 6.5.4 explain the mechanics of a R&F and F&O implementation developed for the model. After that, we introduce a two-stage Relax&Fix&Optimize algorithm in Sect. 6.5.5 that combines the two approaches by executing R&F first and then running F&O for post-optimization. We compare several algorithm variants by testing them on generated test instances, with assumptions following the structure of the real-world optimization problem in windshield interlayer production. Section 6.6 describes the design, results and interpretations of these computational experiments.

6.3 The Capacitated Lot-Sizing Problem with Sequence-Dependent Setups and Substitutions

6.3.1 Assumptions

The model developed in the following – termed Capacitated Lot-sizing problem with Sequence-Dependent setups and substitutions (CLSD-S) – is an extension of the *Capacitated Lot-sizing problem with Sequence-Dependent setups (CLSD)* (Haase, 1996) by product substitution options. Its assumptions are as follows:

- Lot-sizing for multiple, continuous products (set of products P).
- All parameters are deterministic.
- Single-level time structure, big bucket model.
- Finite time horizon with T periods.
- An arbitrary number of products can be set up in each period, provided that the durations of required changeovers fit into the capacity (big bucket model).
- Scheduling (sequencing) of products within periods.
- Demand refers to demand classes (set of demand classes D).
- Time-varying demand of demand classes that has to be satisfied at the end of each period.
- Each demand class can be satisfied by certain products (demand-class specific substitution options).
- General substitution structure.
- Substitution ratios are arbitrary.
- Single capacitated production resource with finite speed, all products share this resource.
- Capacity consumption per unit produced differ among products.
- Setup carry-over is possible.
- Single production level.
- Time-invariant sequence-dependent setup costs and setup times (the latter consume capacity).
- Each product can be set up at most once per period, thus the setup costs and times need to fulfill the triangle inequality (see Sect. 2.1.1.2).
- Setups have to be completed within a single period.
- Time-invariant linear holding costs.
- Time-invariant production costs.
- Time-invariant variable conversion costs for satisfying demand of a certain class with a specific product.
- No lead times.
- Lost sales and overtime production are allowed, with high penalty costs.
- Initial inventories.
- Initial setup state is given (machine set up for a certain product).
- Final inventory targets specifying the minimum final inventory quantity to be built up for each demand class.

- Cost minimization objective.
- Continuous variables for lot sizes.

We chose to assume that both lost sales and overtime are allowed. The reasoning behind these two assumptions is as follows:

(1) First, production capacities are frequently "soft" constraints as machines could run longer than the planned daily operating time or their throughput could be increased if they are by default run below their technical limits. Second, the overtime assumption is required for F&O (see section 6.5.4): With overtime allowed, even a production sequence containing *all* products can fulfill the capacity constraint, as the overtime amount may be set arbitrarily high. Thus, this simplifies determining an initial feasible solution. The overtime penalty costs are set very high to effect that F&O later finds a solution without overtime.

(2) If finding a feasible plan that fulfills all demand on time is impossible and later delivery dates cannot be agreed upon with certain customers, these potential sales get lost. In addition, we wanted to make it easier for R&F (see section 6.5.3) to find integer-feasible solutions to subproblems, hoping that R&F will later find a solution without lost sales due to the associated penalty costs.

6.3.2 Formulation

Using the notation given in Table 6.1, the CLSD-S can be formulated as given below. As in the CLSD, a dummy product 0 is introduced for modeling initial setup states where the resource is not set up for any product.

Minimize $F(q, x, z, I, s, v, o, O, s^f) =$

$$\sum_{t=1}^{T} \sum_{i \in P} \left(p_i q_{it} + h_i I_{it} + \sum_{k \in P, k \neq i} f_{ik} x_{ikt} + \sum_{j \in D_i} r_{ij} s_{ijt} \right) + \sum_{t=1}^{T} \sum_{j \in D} g_j o_{jt} + \sum_{t=1}^{T} oc\, O_t \quad (6.1)$$

subject to

$$I_{it} = I_{i,t-1} + q_{it} - \sum_{j \in D_i} s_{ijt} \quad i \in P, \qquad t = 1, \ldots, T \quad (6.2)$$

$$d_{jt} - o_{jt} = \sum_{i \in P_j} a_{ij}^{-1} s_{ijt} \quad j \in D, \qquad t = 1, \ldots, T \quad (6.3)$$

$$\sum_{i \in P} \left(\kappa_i^p q_{it} + \sum_{k \in P, k \neq i} st_{ik} x_{ikt} \right) \leq K_t + O_t \qquad t = 1, \ldots, T \quad (6.4)$$

Table 6.1 Notations for CLSD-S

Symbol	Definition
Constants	
m	Number of products
n	Number of demand classes
T	Number of periods
Indices and sets	
$i \in P = \{0, \ldots, m\}$	Products including dummy product 0
$j \in D = \{1, \ldots, n\}$	Demand classes
$t = 1, \ldots, T$	Periods
$V = P \cup D$	Vertex set of substitution graph
$E \subseteq P \times D$	Arcs of substitution graph denoting feasible substitutions: $(i, j) \in E$ if product i can fulfil demand of class j
$G = (V, E)$	Substitution graph
$D_i = \{j \mid (i, j) \in E\}$	Set of demand classes whose demand can be fulfilled by product i
$P_j = \{i \mid (i, j) \in E\}$	Set of products that can fulfil demand of class j
Parameters	
d_{jt}	Demand of class j in period t
h_i	Non-negative holding cost for storing one unit of product (per period)
p_i	Unit production cost of product i
r_{ij}	Conversion cost for fulfilling demand class j by product i per unit
I_{i0}	Initial inventory of product i
f_{ik}	Setup cost that is incurred when the setup state of the resource changes from product i to k
K_t	Capacity of resource available in period t
st_{ik}	Setup time for changeover from product i to k
κ_i^p	Capacity required for manufacturing one unit of product i
oc	Overtime production cost per capacity unit
g_j	Lost sales cost for demand class j per unit
a_{ij}	Number of units of product i that substitute one unit of demand class j
z_{i1}	Binary parameter that indicates whether the resource is already set up for i at the beginning of the first period
I_j^f	Final inventory target for demand class j
M_{it}	Large number
Variables	
q_{it}	Production quantity of product i in period t
s_{ijt}	Quantity of product i used to fulfil demand of class j in period t
I_{it}	Inventory of product i at the end of period t
x_{ikt}	Binary variable that indicates whether product k is set up immediately after product i in period t
z_{it}	Binary variable that indicates whether the resource is already set up for product i at the beginning of period t
v_{it}	Auxiliary variable: the larger it is, the later product i is scheduled in period t
O_t	Additional capacity required for overtime production in period t
o_{jt}	Lost sales of demand class j in period t
s_{ij}^f	Quantity of product i used to build up final inventories for demand of class j at the end of period t

$$\kappa_i^p q_{it} \leq M_{it} \left(\sum_{h \in P, h \neq i} x_{hit} + z_{it} \right) i \in P, \qquad t = 1, \ldots, T \tag{6.5}$$

$$\sum_{i \in P} z_{it} = 1 \qquad t = 1, \ldots, T \tag{6.6}$$

$$\sum_{h \in P, h \neq i} x_{hit} + z_{it} = \sum_{k \in P, k \neq i} x_{ikt} + z_{i,t+1} i \in P, \qquad t = 1, \ldots, T \tag{6.7}$$

$$v_{kt} \geq v_{it} + 1 - |P| (1 - x_{ikt}) \qquad i, k \in P,\ i \neq k,\ t = 1, \ldots, T \tag{6.8}$$

$$I_j^f \leq \sum_{i \in P_j} a_{ij}^{-1} s_{ij}^f \qquad j \in D \tag{6.9}$$

$$I_{iT} \geq \sum_{j \in D_i} s_{ij}^f \qquad i \in P \tag{6.10}$$

$$q_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{6.11}$$

$$I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{6.12}$$

$$v_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{6.13}$$

$$x_{ikt} \in \{0, 1\} \qquad i \neq k \in P,\ t = 1, \ldots, T \tag{6.14}$$

$$z_{it} \in \{0, 1\} \qquad i \in P,\ t = 1, \ldots, T + 1 \tag{6.15}$$

$$s_{ijt} \geq 0 \qquad (i, j) \in E,\ t = 1, \ldots, T \tag{6.16}$$

$$o_{jt} \geq 0 \qquad j \in D,\ t = 1, \ldots, T \tag{6.17}$$

$$O_t \geq 0 \qquad t = 1, \ldots, T \tag{6.18}$$

$$s_{ij}^f \geq 0 \qquad (i, j) \in E \tag{6.19}$$

The objective (6.1) is to minimize the sum of holding costs, unit production costs, substitution costs, sequence-dependent setup costs, overtime costs, and lost sales costs. The inventory balance equations are given by (6.2). Constraint (6.3) enforces that demand is either satisfied using the available substitution options, taking the substitution ratios a_{ij} into account, or counted as lost sales otherwise. The constrained capacity of the production resource is modeled by (6.4): It ensures that the capacity consumption by production activities and sequence-dependent setup times never exceeds the available capacity in a period. Equation (6.5) enforces that production of product i only takes place in period t if the resource was already set up for i at the end of period $t - 1$ or a changeover to i is performed in t. As the value M_{it} has to incorporate the final inventory targets so that they can be built up in any case, we set it as follows:

$$M_{it} = \sum_{j \in D_i} \left(\sum_{\tau = t}^{T} d_{j\tau} + I_j^f \right) \tag{6.20}$$

Equation (6.6) means that exactly one product is set up on the resource at the end of each period $t-1$ and thus at the beginning of each period t. The so-called "setup state flow preservation / sub-tour elimination" is modeled by (6.7), which corresponds to (2.21) in the CLSD. Equation (6.8) creates a production sequence for the products within each period using the auxiliary variables v_{it}, analogously to (2.22). Equation (6.9) ensures that at least the minimum required final inventory is built up using feasible substitutes for each demand class. Equation (6.10) determines that not more than the available final inventory of each product is allocated to demand classes that can be satisfied by the product.

We introduced final inventory targets in the model to deal with end-of-horizon effects (also see Sect. 2.1.1.3). The reason why we assumed specified final inventory targets for demand classes rather than products is as follows: If we chose the latter approach, it would not be possible to eliminate a substitutable product from the portfolio of manufactured products because production would still be necessary to build up the specified final inventory. A substitution for the final inventory of the considered product would not be possible. This is turn might lead to solutions that cannot realize savings by substitutions. Equations (6.11)–(6.19) define the variable domains.

In the following, a solution that fulfills all constraints (6.2)–(6.19) but might still require overtime or lost sales is termed "feasible". We say that a solution is "capacity-feasible" if it does not use overtime (i.e., $O_t = 0 \,\forall\, t$), and name it "demand-feasible" if the β service level is 100% (i.e., $o_{jt} = 0 \,\forall\, j,t$).

6.4 Determining Efficient or Good Sequences

A sequence (of products manufactured within a period) is called efficient if no other sequence containing the same set of products and having the same first and last product exists that has a lower total setup cost than the considered sequence (also see Sect. 2.2.5 and Haase and Kimms, 2000). If setup times are assumed to be greater than zero, the efficiency concept for sequences can alternatively be based on setup *times*. However, note that a sequence that is efficient regarding the setup costs might be inefficient regarding the setup times and vice versa. Haase and Kimms (2000) assume that the setup costs of changeovers are a linear function of the setup times, i.e., they assume a close connection between setup costs and times.

We will need efficient or at least good sequences as a sequence containing all products is required for the initial fixation of binary variables in the Fix&Optimize algorithm described in Sect. 6.5.4. We begin with a sequence containing all products to ensure that every possible subset of the products can be selected for production by removing products from this initial sequence.

Given a set of products P^s that the sequence should contain and the pre-specified first and last product b and l of the sequence, the problem of determining an efficient sequence can be reduced to an Asymmetric Traveling Salesman Problem (ATSP) (for ATSP textbook literature see, e.g., Lawler et al., 1985; Reinelt, 1994; Gutin

and Punnen, 2002): This ATSP contains one corresponding city for each product $i \in P^s$. The transportation costs c_{ik} from a city i to another city k equal the setup cost f_{ik} (or setup time st_{ik}) for changing over from the product corresponding to city i to the product corresponding to city k. If the first and last product b and l are not identical, one can deal with this by merging the first and last product into one ATSP city h, setting $c_{hi} = f_{bi}$ and $c_{ih} = f_{il}$, and introducing no separate cities for i and k.

We will use an effective implementation of the Lin–Kernighan heuristic called LKH (Helsgaun, 2000, 2007) for solving these ATSPs. LKH is known to produce high-quality solutions even for very large TSP instances, so that we consider it sufficient for determining good sequences, rather than choosing an exact algorithm.

6.5 MIP-Based Heuristics

As the CLSD-S seems too hard to be solved by using standard MIP solvers, we devise and analyze several variants of MIP-based heuristics for the CLSD-S in this section. These heuristics – Relax&Fix (see Sect. 2.2.4) and Fix&Optimize (see Sect. 2.2.4 and Sahling et al., 2009) – are based on decompositions of the original problem that result in multiple subproblems with a smaller number of binary variables.

The structure of this section is as follows: In order to describe time-, product-, and substitute-oriented CLSD-S decompositions in Sect. 6.5.2, we introduce a formal notation for expressing their subproblems in Sect. 6.5.1. The Relax&Fix and Fix&Optimize algorithm schemes are presented in Sects. 6.5.3 and 6.5.4. A simple hybrid of these heuristics called Relax&Fix&Optimize is described in Sect. 6.5.5. Finally, Sect. 6.5.6 explains the time limits and stopping criteria used when solving subproblems within these heuristics.

6.5.1 Subproblems with Relaxed or Fixed Binary Variables

In each of the CLSDS-S subproblems of a decomposition used in R&F or F&O, some of the binary variables x_{ikt} and z_{it} are fixed to certain values or relaxed to continuous variables with fractional values between 0 and 1 allowed. Only a subset of binary variables is actually optimized as such in the subproblem. The notation used for defining the sets of fixed and relaxed variables is given in Table 6.2.

The sets $\mathcal{V}_x$ and $\mathcal{V}_z$ are defined as:

$$\mathcal{V}_x = \{(i,k,t) \mid i \neq k \in P,\ t = 1,\ldots,T\} \tag{6.21}$$

$$\mathcal{V}_z = \{(i,t) \mid i \in P,\ t = 1,\ldots,T+1\} \tag{6.22}$$

Table 6.2 Notations for CLSD-S subproblems

Symbol	Definition
Indices and sets	
$d \in D$	Ordered list of decompositions
$s \in S_d$	Ordered list of subproblems of decomposition d
S_s^p	Set of subproblems of considered decomposition that were solved before subproblem s
$\mathcal{V}_x$	Set of all index combinations (i,k,t) referring to x_{ikt} variables
$\mathcal{V}_{x,s}^{rel} \subseteq \mathcal{V}_x$	Set of index combinations referring to x_{ikt} variables relaxed continuously in subproblem s
$\mathcal{V}_{x,s}^{fix} \subseteq \mathcal{V}_x$	Set of index combinations referring to x_{ikt} variables that are fixed to the value $\overline{x}_{ikt}$ in subproblem s
$\mathcal{V}_{x,s}^{opt} \subseteq \mathcal{V}_x$	Set of index combinations referring to x_{ikt} variables that are neither fixed nor relaxed and can thus be optimized in subproblem s
$\mathcal{V}_z$	Set of all index combinations (i,t) referring to z_{it} variables
$\mathcal{V}_{z,s}^{rel} \subseteq \mathcal{V}_z$	Set of index combinations referring to z_{it} variables relaxed continuously in subproblem s
$\mathcal{V}_{z,s}^{fix} \subseteq \mathcal{V}_z$	Set of index combinations referring to z_{it} variables that are fixed to the value $\overline{x}_{ikt}$ in subproblem s
$\mathcal{V}_{z,s}^{opt} \subseteq \mathcal{V}_z$	Set of index combinations referring to z_{it} variables that are neither fixed nor relaxed and can thus be optimized in subproblem s
Parameters	
$t(s)$	Period corresponding to subproblem s in time-oriented decompositions
$p(s)$	Product corresponding to subproblem s in product-oriented decompositions
$\overline{x}_{ikt}$	Value to which binary variable x_{ikt} is fixed
$\overline{z}_{it}$	Value to which binary variable z_{it} is fixed
$\widetilde{x}_{ikt}$	Value of x_{ikt} in current solution
$\widetilde{z}_{it}$	Value of z_{it} in current solution
δ	Parameter for substitute-oriented decomposition

Note that the following conditions should hold:

$$\mathcal{V}_x = \mathcal{V}_{x,s}^{rel} \cup \mathcal{V}_{x,s}^{fix} \cup \mathcal{V}_{x,s}^{opt} \qquad s \in S \tag{6.23}$$

$$\mathcal{V}_z = \mathcal{V}_{z,s}^{rel} \cup \mathcal{V}_{z,s}^{fix} \cup \mathcal{V}_{z,s}^{opt} \qquad s \in S \tag{6.24}$$

Also, the sets of relaxed, fixed and optimized variables of course must not overlap in a subproblem.

Using the above notations, a subproblem s is defined by (6.1)–(6.18), with (6.14) and (6.15) replaced by:

$$x_{ikt} = \overline{x}_{ikt} \qquad (i,k,t) \in \mathcal{V}_{x,s}^{fix} \tag{6.25}$$

$$z_{it} = \overline{z}_{it} \qquad (i,t) \in \mathcal{V}_{z,s}^{fix} \tag{6.26}$$

$$0 \leq x_{ikt} \leq 1 \qquad (i,k,t) \in \mathcal{V}_{x,s}^{rel} \tag{6.27}$$

$$0 \leq z_{it} \leq 1 \qquad (i,t) \in \mathcal{V}_{z,s}^{rel} \tag{6.28}$$

$$x_{ikt} \in \{0,1\} \qquad (i,k,t) \in \mathcal{V}_{x,s}^{opt} \tag{6.29}$$

$$z_{it} \in \{0,1\} \qquad (i,t) \in \mathcal{V}_{z,s}^{opt} \tag{6.30}$$

6.5.2 Decompositions

In this section, we describe time-oriented, product-oriented, and substitute-oriented decompositions for the CLDS-S. The product- and substitute-oriented decompositions will only be used for F&O, whereas some of the time-oriented decompositions (TD_1, TD_2, and TD_4) can be used in R&F as well as F&O. Note that we only specify the sets $\mathcal{V}_{x,s}^{opt}$ and $\mathcal{V}_{z,s}^{opt}$ for the decompositions. When using a decomposition for R&F, the other variable sets for a certain subproblem are derived from the specified sets as follows:

$$\mathcal{V}_{x,s}^{fix} = \left(\bigcup_{\varsigma \in S_s^p} \mathcal{V}_{x,\varsigma}^{opt} \right) \setminus \mathcal{V}_{x,s}^{opt} \tag{6.31}$$

$$\mathcal{V}_{z,s}^{fix} = \left(\bigcup_{\varsigma \in S_s^p} \mathcal{V}_{z,\varsigma}^{opt} \right) \setminus \mathcal{V}_{z,s}^{opt} \tag{6.32}$$

$$\mathcal{V}_{x,s}^{rel} = \mathcal{V}_x \setminus \left(\mathcal{V}_{x,s}^{opt} \cup \mathcal{V}_{x,s}^{fix} \right) \tag{6.33}$$

$$\mathcal{V}_{z,s}^{rel} = \mathcal{V}_z \setminus \left(\mathcal{V}_{z,s}^{opt} \cup \mathcal{V}_{z,s}^{fix} \right) \tag{6.34}$$

In contrast, F&O does not relax any variables not contained in $\mathcal{V}_{x,s}^{opt}$ and $\mathcal{V}_{z,s}^{opt}$, but fixes all of them:

$$\mathcal{V}_{x,s}^{fix} = \mathcal{V}_x \setminus \mathcal{V}_{x,s}^{opt} \tag{6.35}$$

$$\mathcal{V}_{z,s}^{fix} = \mathcal{V}_z \setminus \mathcal{V}_{z,s}^{opt} \tag{6.36}$$

$$\mathcal{V}_{x,s}^{rel} = \emptyset \tag{6.37}$$

$$\mathcal{V}_{z,s}^{rel} = \emptyset \tag{6.38}$$

6.5.2.1 Time-Oriented Decompositions

We developed several variants of time-oriented decompositions (TD) of the CLSD-S. In each of these, the number of subproblems equals the number of periods, i.e., $|S_d| = T$. Considering the most basic TD variant $\boldsymbol{TD_1}$, in the subproblem $s \in S$ corresponding to period $t = t(s)$ only the variables x_{ikt} are optimized as binary

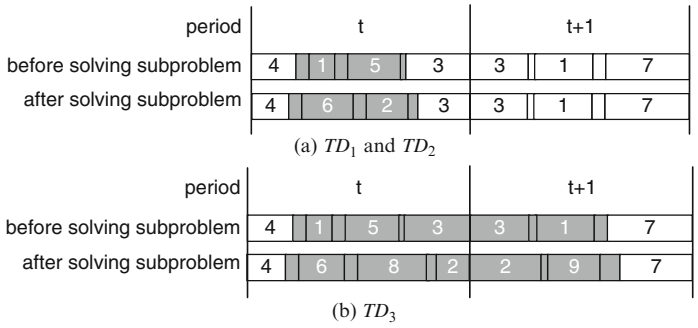


Fig. 6.2 Examples of possible "moves" in F&O using the different time-oriented decompositions

variables. All other variables are either relaxed continuously (in the case or R&F) or fixed (in the case of F&O):

$$\mathcal{V}_{x,s}^{opt} = \{(i,k,\tau) \in \mathcal{V}_x \mid \tau = t(s)\} \tag{6.39}$$

$$\mathcal{V}_{z,s}^{opt} = \emptyset \tag{6.40}$$

Thus, using TD_1 in F&O, the production sequence in period t can be changed by solving the corresponding subproblem. However, the first and last product in the period remain fixed. This is illustrated by Fig. 6.2(a). In the figure, white boxes stand for fixed products and changeovers, whereas gray boxes represent production and changeovers that can be "optimized". Assuming that the initial sequence is as shown in the figure, the initial setup of product 4 and the changeover to 3 at the end of t are fixed in the subproblem for period t, but the setups in between products 4 and 3 are variable. For instance, instead of setting up products 1 and 5, the solution obtained by solving the subproblem might set up products 6 and 2 in between 4 and 3 if this yields a better objective value.

In variant $\boldsymbol{TD_2}$, additionally the setup carryover variables $z_{i,t+1}$ are optimized:

$$\mathcal{V}_{x,s}^{opt} = \{(i,k,\tau) \in \mathcal{V}_x \mid \tau = t(s)\} \tag{6.41}$$

$$\mathcal{V}_{z,s}^{opt} = \{(i,\tau) \in \mathcal{V}_z \mid \tau = t(s)+1\} \tag{6.42}$$

The actual difference between TD_1 and TD_2 is rather small, as fixed values of x_{ikt} automatically imply the value of $z_{i,t+1}$. However, we define TD_2 as we use it for R&F, whereas we use TD_1 in our variants of the F&O heuristic.

Variant $\boldsymbol{TD_3}$ additionally allows to optimize all changeover variables of the subsequent period $t+1$, and thus optimizes all x_{ikt}, $z_{i,t+1}$, and $x_{ik,t+1}$:

$$\mathcal{V}_{x,s}^{opt} = \{(i,k,\tau) \in \mathcal{V}_x \mid \tau = t(s) \vee \tau = t(s)+1\} \tag{6.43}$$

$$\mathcal{V}_{z,s}^{opt} = \{(i,\tau) \in \mathcal{V}_z \mid \tau = t(s)+1\} \tag{6.44}$$

TD_3 allows to change the entire production sequence in period $t+1$, except for the last product in the sequence which is fixed. Referring to the example in Fig. 6.2(b), the changeover to product 3 in period t and the changeover to product 1 in $t+1$ are no longer fixed, but can be replaced by setups of other products. In the example, products 2 and 9 are set up instead of 3 and 1. The changeover to product 7 in $t+1$ remains fixed. TD_3 represents an overlapping time decomposition, where the binary variables optimized in adjacent subproblems overlap.

The time decomposition variants differ in the number of binary variables in each subproblem. TD_1 requires the smallest number of binary variables, whereas TD_3 requires the highest. However, TD_3 also promises to lead to better solutions as it optimizes more solution variables simultaneously. When deciding between these variants, there is thus a tradeoff between the computation time for solving a subproblem and the solution quality improvements achieved by solving it.

6.5.2.2 Product-Oriented Decompositions

In the product-oriented decompositions (PD) of the CLSD-S, there is one subproblem for each product $i \in P$. All product decompositions described in the following are based on an obtained integer-feasible solution. In the basic product decomposition ***PD*$_1$**, we optimize all variables required for removing those setups of product $i = i(s)$ in the considered solution whose production does not overlap period boundaries. We let $h(t)$ denote the product that is produced immediately before i in period t and $k(t)$ the product produced immediately after i in period t. PD_1 allows to remove the production of i in t and directly change over from $h(t)$ to $k(t)$. The set $\widetilde{T}_i \subseteq \{1, \ldots, T\}$ contains all periods in which i is neither set up at the beginning nor the end of the period in the considered solution. Thus, the variables to be optimized are as follows:

$$\mathcal{V}_{x,s}^{opt} = \left\{(h,i,\tau) \in \mathcal{V}_x \mid i = i(s) \wedge \tau \in \widetilde{T}_i \wedge h = h(\tau)\right\} \cup$$
$$\left\{(i,k,\tau) \in \mathcal{V}_x \mid i = i(s) \wedge \tau \in \widetilde{T}_i \wedge k = k(\tau)\right\} \cup$$
$$\left\{(h,k,\tau) \in \mathcal{V}_x \mid \tau \in \widetilde{T}_i \wedge h = h(\tau) \wedge k = k(\tau)\right\} \tag{6.45}$$
$$\mathcal{V}_{z,s}^{opt} = \emptyset \tag{6.46}$$

An illustration of PD_1 is given in Fig. 6.3(a): Assuming that the initial F&O sequence is as indicated, PD_1 allows to remove the setup of product 1 in period 1 by immediately changing over from product 4 to 2.

PD_1 does not enable F&O to remove production lots of a product that stretch over two or more periods. For this purpose, one can add additional variables to the sets of variables to be optimized. This is done in the product decomposition ***PD*$_2$**, which was developed by (F. Sahling 2008, personal communication) for an MLCLSP extension with sequence-dependent setups: Consider each production of a product i that starts in a period t_s with a changeover from another product h to i,

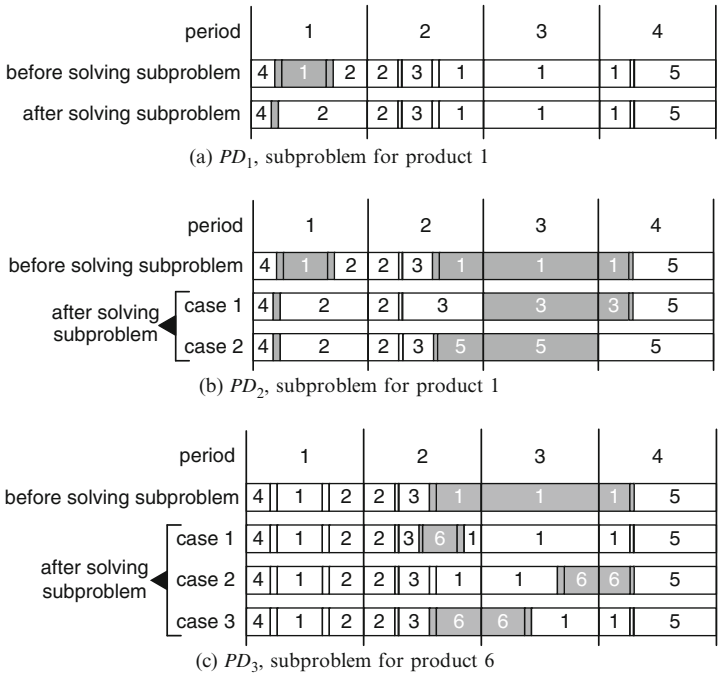


Fig. 6.3 Examples of possible "moves" in F&O using the different product-oriented decompositions

stretches through potentially multiple periods $t_s, \ldots, t_e$, and ends in period t_e with a changeover from i to another product k. t_s and t_e may also be successive periods, i.e., the case where $t_e = t_s + 1$ is included. PD_2 ensures that F&O can do the following two changes to the solution:

1. Entirely remove the production of i in periods $t_s, \ldots, t_e$, and instead let h remain set up until the beginning of t_e, where a changeover from h to k occurs.
2. Entirely remove the production of i in periods $t_s, \ldots, t_e$, and instead immediately change over from h to k in t_s, and let k remain set up until t_e or even later.

These two changes are demonstrated in Fig. 6.3(b), referring to the subproblem corresponding to product $i = 1$: When removing the production of product 1 in the periods $t_s = 2, \ldots, t_e = 4$, one can either let product $h = 3$ remain set up until period $t_e = 4$ where a fixed changeover to product 5 has to be performed (case 1), or immediately change over from $h = 3$ to $k = 5$ in period $t_s = 2$ (case 2).

In addition, for the case that $h = k$, we want to allow F&O a third type of change that lets $h = k$ remain set up from $t_s, \ldots, t_e$, without any changeovers in between. Note that change 1 is not possible if a changeover to h from k or a product scheduled after k in t_e is contained in the considered solution. Also, note that change 2 is

not possible if a changeover from k to h or a product scheduled before h in t_s is contained in the considered solution.

In order to allow change 1, we add the index combinations of the following variables to $\mathcal{V}_{x,s}^{opt}$ and $\mathcal{V}_{z,s}^{opt}$: $x_{hit_s}, x_{ikt_e}, x_{hkt_e}$ as well as z_{it} and z_{ht} for all $t = t_s+1, \ldots, t_e$. For change 2, we have to add the index combinations of the following variables to the sets: $x_{hit_s}, x_{hkt_s}, x_{ikt_e}$, as well as z_{it} and z_{kt} for all $t = t_s + 1, \ldots, t_e$. For change 3, we need to add x_{hit_s}, x_{ikt_e}, as well as z_{it} and z_{ht} for all $t = t_s + 1, \ldots, t_e$.

Compared with PD_1, PD_2 additionally includes all index combinations to $\mathcal{V}_{x,s}^{opt}$ and $\mathcal{V}_{z,s}^{opt}$ required for allowing feasible changes 1 and 2.

In another product decomposition named $\boldsymbol{PD_3}$, we also allow to insert production lots into periods where product i is not produced in the current, integer-feasible solution. The insertion position in the current sequence of a period is not allowed to be arbitrary, but only the cheapest insertion is allowed. This cheapest insertion is determined based on the sequence-dependent setup costs in a manner similar to the cheapest insertion heuristic for the TSP: Considering a product i and period t, we insert i between a product h and product k that is immediately produced after h in the production sequence in period t, i.e., $\widetilde{x}_{hkt} = 1$. We chose those products $h(i,t)$ and $k(i,t)$ for which the setup cost $\Delta(h,i,k)$ increase is the lowest:

$$\Delta(h,i,k) = f_{hi} + f_{ik} - f_{hk} \tag{6.47}$$

$$(h(i,t),k(i,t)) = argmin_{h,k \in P \mid \widetilde{x}_{hkt}=1} \{\Delta(h,i,k)\} \tag{6.48}$$

In the subproblem s for product $i = i(s)$, we iterate over all periods and, for each period t where i is not produced in the current solution, add the index combinations of the variables $x_{h(i,t),k(i,t),t}$, $x_{h(i,t),i,t}$, and $x_{i,k(i,t),t}$ to $\mathcal{V}_{x,s}^{opt}$. Setup carry-over variables are not optimized, i.e., $\mathcal{V}_{z,s}^{opt} = \emptyset$. An example of the effect of PD_3 is shown in Fig. 6.3(c): As to the subproblem for product $i = 6$, PD_3 allows to insert product 6 at the cheapest position in period 2, which we assume to be the position between products 3 and 1 (case 1).

In PD_3, we also incorporate the case where a product h is already set up at the beginning of a period t during which i is not set up at any point and no changeover occurs, and h remains set up until period $t + 1$ where a changeover to k occurs: We allow to insert i between h and k by changing over from h to i in t and from i to k in $t + 1$. That is, we allow to insert a product "into the carryover" between periods t and $t + 1$. This is done by adding the index combinations of the variables x_{hit}, $x_{h,k,t+1}$, and $x_{i,k,t+1}$ to $\mathcal{V}_{x,s}^{opt}$, and adding $z_{h,t+1}$ and $z_{i,t+1}$ to $\mathcal{V}_{z,s}^{opt}$. This is illustrated in the case 2 in Fig. 6.3(c): Product 6 is inserted into the carryover from period 3 to 4, between products 1 and 5.

Also, we allow to insert i between two products h and k if a changeover from h to k occurs in a period t and k remains set up in the entire period $t + 1$: The index combinations of the variables x_{hit}, x_{hkt}, and $x_{i,k,t+1}$ are added to $\mathcal{V}_{x,s}^{opt}$ and those of $z_{k,t+1}$ and $z_{i,t+1}$ to $\mathcal{V}_{z,s}^{opt}$. This change is named case 3 in Fig. 6.3(c): Product 6 is inserted into the carryover from period 2 to 3, between products 3 and 1.

In addition, let PD_4 denote a decomposition that combines PD_2 and PD_3, i.e., for PD_4 $\mathcal{V}_{x,s}^{opt}$ and $\mathcal{V}_{z,s}^{opt}$ are the union of the corresponding sets of PD_2 and PD_3.

6.5.2.3 Substitute-Oriented Decompositions

It might be useful to optimize the setups of a product and its substitutes simultaneously in order to properly incorporate substitutions in the solution process, rather than optimizing the setup variables of the products sequentially, separately for each product. Hence, we decided to implement and test decompositions that are based on the substitution structure of the problem: For each demand class that can be satisfied by $\geq \delta$ products, we define a subproblem that simultaneously optimizes the setups of all substitutes for the demand class. Thus, the set of demand classes for which a subproblem will be solved is defined as follows:

$$D^{sd,\delta} = \left\{ j \in D \mid |P_j| \geq \delta \right\} \tag{6.49}$$

δ is a parameter with a default value of 2, which implies that subproblems are only created for demand classes for which substitutions are actually possible. Instead, one could also set $\delta = 1$ so that subproblems are solved for *all* demand classes. This could be useful if we want to entirely replace the product-oriented by a substitute-oriented decomposition, and wish to optimize the binary setup variables of every product in at least one subproblem.

This type of substitute-oriented decomposition can be based on either of the product-oriented decompositions presented in the previous section. We use the name $SD_{\delta,\epsilon}$ to refer to the substitute-oriented decomposition that is based on PD_ϵ and uses the parameter value δ. In the subproblem associated with $j \in D^{sd,\delta}$, $\mathcal{V}_{x,s}^{opt}$ and $\mathcal{V}_{z,s}^{opt}$ are the union of those sets of all subproblems of PD_ϵ that correspond to substitute products $i \in P_j$ of the considered demand class j.

6.5.3 Relax&Fix

The control flow of the implemented Relax&Fix heuristic is depicted in Algorithm 6.1. In the following, we drop the indices x and z in symbols like $\mathcal{V}_{x,s}$ and $\mathcal{V}_{z,s}$ to simplify the notation: $\mathcal{V}$ refers to the index combinations of all binary variables (both x_{ikt} and z_{it}), $\mathcal{V}_s^{opt}$ to the index combinations of all binary variables to be optimized in a certain subproblem, etc. Also, we let $\overline{\mathcal{V}}^{opt}$ denote the index combinations of binary variables that are optimized in the current state of the problem, and let $\overline{\mathcal{V}}^{rel}$ denote the index combinations of binary variables that are relaxed continuously.

In all its variants, R&F is based on a time-oriented decomposition. The algorithm starts by calling function `RelaxContinuous`, which manipulates the problem

so that all binary variables are relaxed continuously, i.e., added to $\overline{\mathcal{V}}^{rel}$. The subproblems $s \in S_{TD}$ are solved in ascending order of the indices of the corresponding periods $t(s)$. To this end, function `OrderByPeriodAscending` returns the subproblems of the specified time-oriented decomposition in ascending order of the period indices. The subproblems are saved in that order in list LS. The algorithm iterates over all subproblems $d \in S_{TD}$.

In every iteration, it first calls function `ClearChanges` to clear all changes that were performed on the problem by removing relaxations or fixations of binary variables, i.e., by adding all binary variables to the set $\overline{\mathcal{V}}^{opt}$, resp. After that, it calls function `Solve` to run an MIP solver on the problem with the current fixations and relaxations of binary variables. If no feasible solution to the subproblem can be found within a specified time limit, the algorithm stops and returns without a feasible solution. Otherwise, function `FixToCurrentValues` is used to fix all binary variables optimized in the subproblem to their current values, i.e., either 0 or 1.

The algorithm terminates normally after all subproblems of the time-oriented decomposition have been solved. The R&F heuristic may be combined with either of the time-oriented decompositions TD_1, TD_2, an TD_3.

Algorithm 6.1: Relax&Fix heuristic

```
   Input  : Problem instance, time-oriented decomposition TD
   Output: Feasible solution if found
 1 RelaxContinuous(V);
 2 LS ← OrderByPeriodAscending(S_TD);
 3 foreach s ∈ LS do
 4 |   ClearChanges(V_s^opt);
 5 |   (status, solution) ← Solve();
 6 |   if status = TimeLimitExceeded then
 7 |   |   return (status, ∅);
 8 |   end
 9 |   FixToCurrentValues(V_s^opt);
10 end
11 return status, solution;
```

6.5.4 Fix&Optimize

This section describes the general control flow of the implemented Fix&Optimize and also delineates priority rules for the decompositions given in Sect. 6.5.2.

6.5.4.1 Algorithm Scheme

Algorithm 6.2 outlines the control flow of Fix&Optimize. The algorithm starts by generating an initial feasible solution using a specified function `GenerateInitialSolution` as described in Sect. 6.4. If this function succeeds in finding a solution, all binary variables are fixed to their values in that solution. After that, the algorithm applies several decompositions in their order in the list LD. For each decomposition $d \in LD$, it iterates over all subproblems $s \in S_d$. The order in which the subproblems are considered is determined by a priority rule PR_d. The subproblems are solved in descending order of their priority values. This order is determined and saved in the list L.

For each subproblem s, the following steps are performed: First, save the current fixations of binary variables. Second, clear all fixations of variables contained in $\mathcal{V}_s^{opt}$. Third, solve the subproblem using an MIP solver. If no feasible subproblem solution is found within a specified time limit, the algorithm stops. Otherwise, it checks whether the found solution should be accepted using a criterion specified in a function `Accept`. If yes, the optimized variables are fixed to their values in the new solution. If not, the solution is discarded and the variable fixations that were cleared for solving the subproblem are restored. The algorithm terminates normally after all subproblems of all decompositions have been solved.

For the sake of simplicity, we will use an `Accept` function that accepts all new solutions. As a "warm start" from the current solution is performed for every subproblem, the new solution will be identical to or better than the current solution. In contrast, Sahling et al. (2009) use a more complex `Accept` function: As long as no capacity-feasible solution (i.e., with zero overtime in all periods; see Sect. 6.3.2) has been found, every better new solution is accepted. As soon as a first capacity-feasible solution has been found, it only accepts better capacity-feasible solutions. However, there might not be a need to do so if the overtime costs are set sufficiently high, as this acceptance criterion makes the F&O implementation more complex.

Note that F&O can be interpreted as a *local search* algorithm (Hoos and Stützle, 2005) in which the neighborhood of a solution is determined by the variables in $\mathcal{V}_s^{opt}$ to be optimized in the current subproblem of the used decomposition: The neighborhood of a solution is implicitly given by the integer-feasible solutions to the current subproblem. "Moves" are performed by fixing binary variables to their new values after solving a subproblem. If each of its subproblems is solved to optimality, F&O can be seen as a local search with best-fit neighbor selection.

F&O has a certain analogy to the metaheuristic Variable Neighborhood Search (VNS) (Hansen et al., 2008): Each of the subproblems of each of the decompositions $d \in LD$ implicitly defines a neighborhood, and moves using these neighborhoods are executed in sequential order. Similarly, VNS uses multiple neighborhoods and changes between these to "escape" local optima. However, the described F&O algorithm scheme does not repeatedly switch back and forth between neighborhoods as possible in VNS.

Algorithm 6.2: Fix&Optimize heuristic

Input : Problem instance, list of decompositions LD, subproblem priority rule PR_d for each decomposition d
Output: Feasible solution if found

```
(status, solution) ← GenerateInitialSolution();
if status = TimeLimitExceeded then
    return (status, ∅);
end
FixToCurrentValues(V);
foreach d ∈ LD do
    L ← OrderUsingPriorityRule(S_d, PR_d);
    foreach s ∈ L do
        SaveFixedValues(V_s^opt);
        ClearChanges(V_s^opt);
        (status, solution) ← Solve();
        if status = TimeLimitExceeded then
            return (status, ∅);
        end
        if Accept(solution) then
            FixToCurrentValues(V_s^opt);
        else
            RestoreFixedValues(V_s^opt);
        end
    end
end
return status, solution;
```

6.5.4.2 Priority Rules

For each of the decompositions, several potential priority rules come into question. We decided to use/compare the following priority rules:

- Time-oriented decompositions: Always consider the subproblems in ascending order of the period indices $1, \ldots, T$.
- Product-oriented decompositions
  - Consider subproblems in ascending order of the product numbers $i \in P$ ($PR_{i,ASC}$). This rule that only depends on the (arbitrary) numbering of the products serves as a comparison to assess the following two rules.
  - Consider subproblems in descending order of the number of demand classes $|D_i|$ that the corresponding product i can fulfill ($PR_{|D_i|,DESC}$).
- Substitute-oriented decompositions
  - Consider subproblems in ascending order of the demand class numbers $j \in D$ ($PR_{j,ASC}$).

– Consider subproblems in descending order of the number of substitutes $|P_j|$ that are available for the corresponding demand class j ($PR_{|P_j|,DESC}$). This rule is based on the idea that substitutions should first be considered for demand classes with a high number of substitutes.

6.5.4.3 Generating Initial Feasible Sequences

An implementation of the function `GenerateInitialSolution` is required for the F&O heuristic. In contrast to the ML-CLSP where a trivial initial fixation with all binary setup variables fixed to 1 always has a feasible solution, this approach is not practicable for the CLSD: If we set all x_{ikt} and z_{it} to 1, the resulting subproblem will be infeasible. Hence, it is required to generate an initial production sequence containing all products for every period to ensure that the problem has a feasible solution. A trivial yet suboptimal way of doing so would be to randomly generate a production sequence for period 1 that starts with the product set up initially and contains all products, then carry over the last product in that sequence to the second period, again create a random sequence with all products in period 2, etc.

Another approach is to generate a good production sequence with all products (a cycle) by solving an ATSP as described in Sect. 6.4. After obtaining a cycle with all products by running the LKH heuristic (Helsgaun, 2000, 2007) one can easily generate a sequence as follows (F. Sahling 2008, personal communication): Start with the product set up initially (with $z_{i1} = 1$), and then repeat the setup cycle again and again throughout all periods – setting up each product exactly once in each period.

6.5.5 Two-Stage Relax&Fix / Optimize Algorithm

Alternatively to using the ATSP approach sketched in Sect. 6.5.4.3 for generating initial sequences in F&O, one can use R&F for generating an initial feasible solution. This can easily be done by inserting the R&F heuristic into the function `GenerateInitialSolution`. This results in a two-stage algorithm, which first generates an initial feasible solution using R&F and then uses F&O for further improving the solution. We will refer to this combination of the two approaches by the term *Relax&Fix&Optimize (R&F&O)*.

6.5.6 Time Limit and Stopping Criterion for Subproblems

When solving medium- or large-size CLSD-S instances with either of the MIP heuristics described in this chapter, it might not be advisable to solve each of the subproblems to optimality because this could take hours just for a single instance.

Hence, it makes sense to specify a different stopping criterion for R&F/F&O subproblems. One simplistic approach is to set an overall, absolute time limit TL for the MIP heuristic (e.g., 5 min), and divide this time equal among all subproblems: Assuming that there are in total n_s subproblems to be solved, the time limit for each subproblem is set to TL/n_s. However, in this way, one would risk that the heuristic terminates after TL/n_s time units without a feasible solution because no feasible solution to the first subproblem has been found within this time limit, even though the absolute time limit TL has not been exceeded yet.

Hence, we try to invest this computational budget in a smart way, Again, an overall time limit TL is given. At any point, the heuristic aims at allocating the remaining time T^r equally to the n_s^r remaining subproblems to be solved. Based on this idea, it determines a specific subproblem time limit T^r/n_s^r for each subproblem before solving it. If no feasible solution to the current subproblem has been found by the MIP solver within T^r/n_s^r, the algorithm keeps solving this subproblem until TL is exceeded or a feasible solution to the subproblem has been found. If, instead, a feasible solution to the subproblem is found by the MIP solver earlier than T^r/n_s^r, the heuristic lets the MIP solver keep running to improve the solution until the subproblem time limit T^r/n_s^r is reached. If an optimal solution to the subproblem is found earlier than T^r/n_s^r, the leftover time budget is divided among the remaining subproblems.

6.6 Computational Experiments

In this section, we analyze the solution quality and performance of several R&F and F&O variants and compare them with a standard MIP solver. In detail, we examine the following research questions:

1. Do substitutions occur in the solutions yielded by the algorithms?
2. Do the MIP heuristics work better than a standard MIP solver?
3. Which R&F or F&O variant performs best in terms of solution quality and/or performance?

In the following, we first describe the employed test instance generator in Sect. 6.6.1. Section 6.6.2 explains the design of our computational experiments. Section 6.6.3 specifies the algorithm variants that we compare in the experiments. The computational results are presented and interpreted in Sect. 6.6.4.

6.6.1 Problem Instances

We could not obtain sufficient problem data for solving the real-world CLSD-S application in windshield interlayer production planning (see Sect. 3.1) from the company. This was on one hand due to difficulties of gathering the required data,

on the other hand due to reasons of confidentiality. Hence, we decided to generate problem instances artificially, while trying to make their structure as realistic as possible. We made the instances difficult by setting capacities rather tight and ensuring that substitution decisions are nontrivial.

The notation used for describing the instance generator is given in Table 6.3. Most distribution assumptions of the test instances are contained in Table 6.4, the more complex assumptions are delineated in the following.

Every instance has the same number of products and demand classes and every demand class corresponds to a product, i.e., $m = n$ and $P = D$. Each product can be described by values of several features $f \in F$. We assume that feature values are integer and ≥ 0, and distinguish three groups of features:

1. Features for which substitutions are not allowed, in the sense that a feasible substitute k has to have the same value for each of these features of product i: $v_{fk} = v_{fi} \; \forall \; F_{NO} \subseteq F$.
2. Features for which downward substitution is possible, i.e., it is required that a feasible substitute k has to have the same or a higher value for each of these features than product i: $v_{fk} \geq v_{fi} \; \forall \; F_{DW} \subseteq F$. Higher values of a feature mean that the feature value is higher in downward substitution hierarchy.
3. Features for which downward substitution of a product i is possible, but only by a substitute product k whose corresponding feature value is maximally 1 higher: $v_{fk} = v_{fi} \vee v_{fk} = v_{fi} + 1 \; \forall \; F_{DW+1} \subseteq F$.

Note that the LSP-SI and MR-CLSP-S instances with general substitution described in Sect. 5.5.1 also assume a substitution structure based on product features: In these instances, one product can substitute another if its width and height are greater than those of the other product.

Based on the application in windshield interlayer production planning mentioned in Sect. 3.1, we assume that there are $n_f = 3$ features, namely the mix, thickness, and width of the foil. We excluded the attributes roughness, cooling, and packaging in order to get medium-size instances where substitution options exist. If we had included those in addition, only few substitution options would be generated, as the chance that a product can substitute another gets smaller with an increasing number of attributes. The distribution assumptions and substitution characteristics for the features are contained in Table 6.5. $U^{int}(a, b)$ stands for a uniform distribution with integer values between a and b.

The idea for generating the unit cost data is as follows: We assume that the unit cost p_i of a product i is a linear function of its feature values v_{fi} except for feature 1 (mix), which has a special meaning: A certain unit cost base value p^b_v is associated with each of its possible values. For all other features, a weight w^p_f (which could also be negative) is multiplied with the value v_{fi}. The overall unit cost of a product i is obtained by summing up the unit cost base and all of the weighted feature values:

$$p_i = p^b_{v_{1i}} + \sum_{f \in F \setminus \{1\}} w^p_f v_{fi} \tag{6.50}$$

The unit cost base values for feature 1 are set as follows: $p^b_0 = 5$ and $p^b_1 = 10$.

Table 6.3 Notations for CLSD-S instance generator

Symbol	Definition
Constants	
n_f	Number of features
Indices and sets	
$f \in F = \{1, \ldots, n_f\}$	Features
$F_{NO} \subseteq F$	Features for which substitutions are only possible by products with an identical value for this feature
$F_{DW} \subseteq F$	Features for which downward substitution is possible
$F_{DW+1} \subseteq F$	Features for which downward substitutions of a product i are possible, but only by a substitute product k whose corresponding feature value is maximally 1 higher, i.e., $v_{fk} = v_{fi}$ or $v_{fk} = v_{fi} + 1$
Parameters	
$p^b_{v_{1i}}$	Unit cost base for products with value v_{1i} for feature 1
w^p_f	Cost weight for feature f
w^c_f	Conversion effort weight for feature f
w^s_f	Changeover effort weight for feature f
α	Production times weight for generating capacities
β	Setup times weight for generating capacities
γ	"Capacity availability" factor
Random values	
v_{fi}	Value of feature f of product i
$N^{\geq 0}(\mu, \sigma)$	Normal distribution with values <0 are cut off, i.e., it repeatedly samples values from a $N(\mu, \sigma)$ distribution until a value ≥ 0 is returned
Derived values	
ω_{st}, ω_f	Auxiliary values
st_{cycle}	Total setup time of setup cycle corresponding to TSP tour that contains all products
$K^{min,p}$	Total capacity required for producing the approximate net demand (incorporating initial and final inventories) for all products excluding setup times
$K^{min,s}$	Approximate total capacity required for changeovers if each product is produced in every period
$\overline{K}^{min}$	Approximate required capacity per period

Table 6.4 Instance generator settings

Parameter	Assumption
Demand	Normally distributed, stationary over time, but different mean μ and variation coefficient σ/μ for each demand class: $\sim N^{\geq 0}(\mu, \sigma)$ with $\mu \sim N\,(50, 20)$ and $\sigma/\mu \sim N\,(0.2, 0.1)$
Setup cost	See (6.55)
Unit cost	See (6.50)
Substitution cost	See (6.51)
Holding cost	2% of the unit cost
Initial setup state	Resource is set up for randomly chosen product
Initial inventory	$\sim U^{int}(1.5\iota_i, 4\iota_i)$ with $\iota_i = \frac{1}{T}\sum_{t=1}^{T} d_{it}$
Final inventory	$4\iota_i$ with ι_i as defined above
Resource consumption	$\kappa_i^p \sim N\,(20, 5)$
Setup times	See (6.54)
Capacities	See (6.58)
Lost sales cost	500 times as high as the production cost of the preferred product of the demand class
Overtime cost	100 per capacity unit and period

Table 6.5 Assumptions for distributions and characteristics of product features

Feature	Name	Distribution	Substitution	w_f^p	w_f^c	w_f^s
1	Mix	$U^{int}(0, 1)$	No substitution	–	–	0.5
2	Thickness	$U^{int}(0, 2)$	Downward, diff. ≤ 1	1.5	0.5	0.5
3	Width	$U^{int}(70, 100)$	Downward	0.3	0.2	0.05

The product substitution ratios a_{ij} are assumed to be 1 in all cases. We calculate the conversion cost for feasible substitutions using feature weights w_f^c that indicate the conversion effort when substituting a product by another product that has a different value for feature f (see Table 6.5). The underlying assumption is that the more the features of a product and its substitute differ, the higher the conversion effort. The conversion cost for feasible conversions $(i, k) \in E$ is calculated by the following equation, where X is a random variable with the distribution $N(5, 3)$:

$$r_{ik} = \left(\frac{1}{\sum_{f \in F} w_f^c} \cdot \sum_{f \in F} \begin{cases} w_f^c & \text{if } i \neq k \\ 0 & \text{if } i = k \end{cases} \right) \cdot X \tag{6.51}$$

The sequence-dependent setup times and costs are generated using changeover effort weights w_f^s that denote how much the setup cost and time for changing from one product i to another product k depends on whether k has a different value for feature f than i (see Table 6.5). We assume that the setup costs components sum up if i and k differ in multiple features, whereas the setup time only depends on the differing feature that causes the highest effort. That is, we assume that the setup

activities for changing multiple features can be parallelized. Hence, a sum is used in (6.53) and a maximum operator in (6.52).

In order to generate st_{ik} and f_{ik}, we first sample random numbers Y from a normal distribution $N^{\geq 0}(1, 0.2)$ with values < 0 cut off. Using these random numbers, both st_{ik} and f_{ik} are calculated using the same random value, so that they are correlated:

$$\omega_{st} = \frac{1}{max\left\{w^s_f \mid f \in F\right\}} max\left\{w^s_f \mid f \in F, v_{fi} \neq v_{fk}\right\} \tag{6.52}$$

$$\omega_f = \frac{1}{\sum\limits_{f \in F} w^s_f} \sum_{f \in F \mid v_{fi} \neq v_{fk}} w^s_f \tag{6.53}$$

$$st_{ik} = \omega_{st} \cdot 4000 \cdot Y \tag{6.54}$$

$$f_{ik} = \omega_f \cdot 40000 \cdot Y \tag{6.55}$$

Resource capacities are set in a way that they are, averaged over all periods, correlated with the capacities required for a feasible production plan without overtime. The tightness of the capacities can be adjusted by instance generator parameters.

The procedure for generating the capacities is as follows: We first calculate the minimum resource capacity required for producing sufficient quantities of each product to fulfill the net demand, while considering initial and final inventories in an approximate way that ignores potential substitutions:

$$K^{min,p} = \sum_{i \in P} \kappa^p_i \left(\sum_{t=1}^{T} d_{it} - I_{i0} + I^f_i \right) \tag{6.56}$$

The corresponding capacity consumption by setups is estimated by the total setup time of a good setup cycle containing all products. This is done by solving an ATSP with the LKH heuristic where, roughly speaking, one city exists for each product and the transportation costs equal the sequence-dependent changeover times (for details see Sect. 6.4). With st_{cycle} denoting the total length of the obtained tour and assuming that each product is produced in every period, the capacity consumption by setups is estimated as follows:

$$K^{min,s} = T \cdot st_{cycle} \cdot \frac{m-1}{m} \tag{6.57}$$

The term $(m-1)/m$ is required because, if every product is set up in every period, there are on average $(m-1)/m$ changeovers per period. $K^{min,s}$ overestimates the capacity consumption by setups because, usually, only a subset of the products will be set up in each period. Hence, we weight $K^{min,p}$ with a factor $\alpha = 1.0$ and $K^{min,s}$ with a factor $\beta = 0.4$ and weight the sum with the capacity availability factor $\gamma = 1.2$ to obtain an approximate required capacity for product i. We then divide

by the number of periods to obtain an approximation of the required capacity $\overline{K}^{min}$ per period:

$$\overline{K}^{min} = \frac{1}{T} \cdot \gamma \cdot \left(\alpha \cdot K^{min,p} + \beta \cdot K^{min,s}\right) \tag{6.58}$$

Using these values, we generate the resource capacities K_t by setting all of them to $\overline{K}^{min}$.

6.6.2 Experimental Design

The setup for our experimental design is as follows: As real-world instances of the CLSD-S were not available, we use the instance generator described in Sect. 6.6.1 to generate synthetic instances. The generated instances are of medium size, they contain 20 products and 10 periods. We generate 20 problem instances to ensure that the comparison of algorithm variants is not skewed due to a non-representative sample of instances. This is done using the base case parameters of the instance generator. Several factors of this instance generator could be varied in further experiments, e.g., the tightness of capacities and the setting whether overtime and/or lost sales are allowed.

ILOG CPLEX® 11.2 is used for solving CLSD-S instances and subproblems in R&F and F&O. All problems and subproblems are not solved to optimality, but a relative MIP gap tolerance of 10^{-4} (default setting) is used. The overall, absolute time limit for each MIP heuristic is 5 min. The stopping criterion described in Sect. 6.5.6 is used to determine subproblem time limits based on the remaining time and remaining number of subproblems to be solved: CPLEX® is configured in a way that after this subproblem time limit, it stops solving a subproblem as soon as an integer-feasible solution to it has been found. If no feasible solution to a subproblem has been found until the absolute time limit is reached, the heuristic returns without a feasible solution.

The running times reported in the following include everything, i.e., the running times of CPLEX® used for solving subproblems as well as those of our heuristics and those of the LKH heuristic used for solving ATSPs to generate initial sequences in F&O. The algorithms and experiments were implemented in Java™ and run on a computer with 2.0 GHz Intel® Pentium® Dual-Core CPU, 2 GB memory and Windows Vista™.

6.6.3 Solution Approaches

In our computational experiments, we compare the following algorithms:

- CPLEX® B&C with the standard time limit of 5 min (termed CPX_{std})
- CPLEX® B&C with a longer time limit of 1 h to obtain a good lower bound (referred to by CPX_{1h})

- R&F with decompositions TD_2 and TD_3 (termed RF_{TD_2} and RF_{TD_3})
- Several variants of F&O and R&F&O, which either use the LKH heuristic to generate an initial fixation or R&F with TD_2

Tables 6.6 and 6.7 describe all of these algorithm variants in detail, along with the names used for referring to them in the following. Note that the substitute decomposition of an algorithm variant is always based on its specified product decomposition. Thus, if the column "SD" contains the value 2 and the column "PD" the value 4, this means that the used substitute decomposition is based on PD_4 and uses $\delta = 2$, i.e., we use $SD_{2,4}$.

We will explain the content of the table by exemplarily describing algorithm variant $\text{RFO}_{TPS,B}$: The strings "RFO" and "TPS" in its name mean that it is an R&F&O heuristic that first runs R&F, then F&O with a time-oriented decomposition, after that with a product-based decomposition, and finally with a substitute-oriented decomposition. "B" indicates that the algorithm configuration is variant "B" in the group of R&F&O algorithm variants with time-, product- and substitute-oriented decomposition that we run in our experiments. $\text{RFO}_{TPS,B}$ is run with a 5 min time limit. It uses TD_2 in R&F and TD_3 in the F&O stage. Decomposition PD_4 is used with priority rule $PR_{|D_i|,DESC}$, and finally, decomposition $SD_{2,4}$ is used with priority rule $PR_{|P_j|,DESC}$.

The reasoning behind the choice of algorithm variants in Tables 6.6 and 6.7 is as follows: Algorithm CPX_{std} is used as a comparison to see whether the heuristics actually return better solutions than CPLEX® within the same computational

Table 6.6 Algorithm variants (1/2)

Name	Type	Time limit (min)	R&F TD	F&O TD	PD	PD priority rule	SD	SD priority rule	Decomposition order
CPX_{std}	B&C	5	–	–	–	–	–	–	–
CPX_{1h}	B&C	60	–	–	–	–	–	–	–
RF_{TD_2}	R&F	5	2	–	–	i, ASC	–	–	–
RF_{TD_3}	R&F	5	3	–	–	i, ASC	–	–	–
$\text{FO}_{P,A}$	F&O	5	–	–	1	i, ASC	–	–	P
$\text{FO}_{P,B}$	F&O	5	–	–	2	i, ASC	–	–	P
$\text{FO}_{P,C}$	F&O	5	–	–	4	i, ASC	–	–	P
$\text{FO}_{P,D}$	F&O	5	–	–	4	$\|D_i\|, DESC$	–	–	P
$\text{FO}_{S,A}$	F&O	5	–	–	1	i, ASC	2	j, ASC	S
$\text{FO}_{S,B}$	F&O	5	–	–	1	i, ASC	1	j, ASC	S
$\text{FO}_{S,C}$	F&O	5	–	–	1	i, ASC	1	$\|P_j\|, DESC$	S
$\text{FO}_{S,D}$	F&O	5	–	–	4	i, ASC	2	j, ASC	S
$\text{FO}_{S,E}$	F&O	5	–	–	4	i, ASC	1	j, ASC	S
$\text{FO}_{S,F}$	F&O	5	–	–	4	i, ASC	1	$\|P_j\|, DESC$	S
$\text{FO}_{T,A}$	F&O	5	–	1	–	–	–	–	T
$\text{FO}_{T,B}$	F&O	5	–	3	–	–	–	–	T

Table 6.7 Algorithm variants (2/2)

Name	Type	Time limit (min)	R&F TD	F&O TD	PD	PD priority rule	SD	SD priority rule	Decomposition order
$FO_{TP,A}$	F&O	5	–	3	4	i, ASC	–	–	T→P
$FO_{TP,B}$	F&O	5	–	3	4	$\|D_i\|, DESC$	–	–	T→P
FO_{TS}	F&O	5	–	3	4	i, ASC	1	$\|P_j\|, DESC$	T→S
$FO_{TPS,A}$	F&O	5	–	3	4	i, ASC	2	$\|P_j\|, DESC$	T→P→S
$FO_{TPS,B}$	F&O	5	–	3	4	$\|D_i\|, DESC$	2	$\|P_j\|, DESC$	T→P→S
$FO_{TPS,C}$	F&O	5	–	3	3	i, ASC	2	$\|P_j\|, DESC$	T→P→S
$FO_{TPS,D}$	F&O	5	–	3	3	$\|D_i\|, DESC$	2	$\|P_j\|, DESC$	T→P→S
$FO_{TPS,E}$	F&O	5	–	3	1	i, ASC	2	$\|P_j\|, DESC$	T→P→S
$FO_{TPS,F}$	F&O	5	–	3	1	$\|D_i\|, DESC$	2	$\|P_j\|, DESC$	T→P→S
RFO_T	R&F&O	5	2	3	–	i, ASC	–	–	T
RFO_{TS}	R&F&O	5	2	3	4	i, ASC	1	$\|P_j\|, DESC$	T→S
$RFO_{TPS,A}$	R&F&O	5	2	3	4	i, ASC	2	$\|P_j\|, DESC$	T→P→S
$RFO_{TPS,B}$	R&F&O	5	2	3	4	$\|D_i\|, DESC$	2	$\|P_j\|, DESC$	T→P→S

budget. CPX_{1h} serves to yield good lower bounds and to obtain a good solution to judge the solutions returned by the MIP heuristics.

The R&F variants RF_{TD_2} and RF_{TD_3} are executed to see whether they can compete with the F&O heuristics.

After that, we examine several variants of F&O that only use one of the three types of decompositions, i.e., either a time-, product- or substitute-oriented decomposition ($FO_{T,\circ}$, $FO_{P,\circ}$, and $FO_{S,\circ}$, where $\circ$ stands for an arbitrary variant of the algorithm type). For each of the decomposition types, multiple configurations with different variants of the decomposition type and different priority rules are considered.

Also, we examine F&O variants in which either a product- or substitute-oriented decomposition is executed after a time-oriented decomposition ($FO_{TP,A}$, $FO_{TP,B}$, and FO_{TS}). As a further variant, we consider F&O variants in which the time-oriented decomposition is run first, after that a product-oriented and finally a substitute-oriented decomposition ($FO_{TPS,\circ}$).

Based on the experiences regarding the F&O variants, we then compare four R&F&O variants that are based on settings which performed well for F&O.

6.6.4 Results and Interpretation

Table 6.8 describes the symbols used for presenting the computational results. We define the β service level as follows:

$$\overline{\beta\text{-SL}} = 1 - \frac{\sum_{t=1}^{T} \sum_{j \in D} o_{jt}}{\sum_{t=1}^{T} \sum_{j \in D} d_{jt}} \tag{6.59}$$

Table 6.8 Notations for computational results

Symbol	Definition
rt_{med}	Median running time (minutes)
rt_{min}	Minimum running time (minutes)
rt_{max}	Maximum running time (minutes)
r_f	Proportion of instances for which a feasible solution (may require overtime or backlogging) has been found by the algorithm (in %)
r_{df}	Proportion of instances for which a feasible solution without backlogging has been found by the algorithm (in %)
r_{cf}	Proportion of instances for which a feasible solution without overtime has been found by the algorithm (in %)
r_{cdf}	Proportion of instances for which a feasible solution with neither overtime nor backlogging has been found by the algorithm (in %)
$\overline{r}_{subst}$	Average proportion of substituted demand in feasible solutions (in %)
$\overline{gap}$	Average relative MIP gap of feasible solutions to best known lower bound (in %)
$\overline{dev}^{best}$	Average deviation from best feasible solution found by any of the algorithms (in %)
$\overline{\beta\text{-SL}}$	Average β service level in feasible solutions (in %)
$\overline{r}_{ot}$	Average proportion of periods with overtime in feasible solutions (in %)

The results of the experimental design are summarized in Tables 6.9 and 6.10. The findings w.r.t. the behavior of CPLEX® were as follows: When run with a 5 min time limit (CPX_{std}), CPLEX® returned a feasible solution for only 13 out of 20 instances. The solution quality of CPX_{std} was rather poor, with an average relative MIP gap of 1,794.19% in feasible solutions. Also, most of the solutions (17 out of 20) still contained overtime or lost sales. Using a 1 h time limit (CPX_{1h}), the solution quality was much better. The average gap to the best solution found for an instance (by any of the algorithm variants) was only 1.10%, i.e., compared with all other algorithm variants, CPX_{1h} found the best solution in the majority of cases. However, the average gap was 29.24%, and CPX_{1h} did not even find a single feasible solution for 4 out of 20 instances. Also, CPX_{1h} could not find an optimal solution for any of the instances. Hence, all reported relative MIP gaps are based on the best known lower bound, not on the optimal objective value. Thus, the exact optimality gaps are likely smaller than the gaps stated in the following.

Comparing the average substitution ratio in solutions yielded by CPX_{std} and CPX_{1h}, one can see that it drops from 10.48% in the bad solutions obtained by CPX_{std} to 3.87% in the better solutions of CPX_{1h}. This might indicate that a high substitution ratio is correlated with lower solution quality for the considered type of instances.

Regarding R&F, variants RF_{TD_2} and RF_{TD_3} returned feasible solutions for all instances. RF_{TD_2} with time decomposition TD_2 yielded better solutions than variant RF_{TD_3} that uses the overlapping time decomposition TD_3: Its running times were somewhat lower, and its average gap of 185.40% was better than the average gap of 498.12% of RF_{TD_3}. This might be due to the increased number of binary variables

Table 6.9 Computational results – experiment 1 (1/2)

Algorithm	rt_{med}	rt_{min}	rt_{max}	$\overline{gap}$	$\overline{dev}_{best}$	$\overline{r}_{ot}$	$\overline{\beta\text{-SL}}$	r_f	r_{cf}	r_{df}	r_{cdf}	$\overline{r}_{subst}$
CPX_{std}	>300	>300	>300	1,794.19	1,371.95	14.62	96.97	65	15	25	15	10.48
CPX_{1h}	>3,600	>3,600	>3,600	29.24	1.10	0.00	100.00	80	80	80	80	3.87
RF_{TD_2}	167.2	91.2	275.4	185.40	116.26	10.00	99.95	100	40	55	20	8.67
RF_{TD_3}	248.7	184.3	>300	498.12	361.34	9.50	99.43	100	70	45	25	7.59
$\text{FO}_{P,A}$	2.4	2.1	2.8	1,558.96	1,171.76	37.00	99.98	100	0	90	0	9.99
$\text{FO}_{P,B}$	2.3	2.1	2.8	1,558.96	1,171.76	37.00	99.98	100	0	90	0	9.99
$\text{FO}_{P,C}$	2.3	2.1	2.9	1,558.96	1,171.76	37.00	99.98	100	0	90	0	9.99
$\text{FO}_{P,D}$	2.4	2.1	3.3	1,567.67	1,176.33	38.50	99.99	100	0	95	0	10.05
$\text{FO}_{S,A}$	2.6	2.2	3.1	1,537.97	1,155.40	40.00	99.98	100	0	90	0	10.11
$\text{FO}_{S,B}$	2.5	2.2	3.2	1,537.97	1,155.40	40.00	99.98	100	0	90	0	10.11
$\text{FO}_{S,C}$	2.4	2.1	3.2	1,494.68	1,122.51	41.50	99.99	100	0	95	0	10.35
$\text{FO}_{S,D}$	2.5	2.0	3.1	1,537.97	1,155.40	40.00	99.98	100	0	90	0	10.11
$\text{FO}_{S,E}$	2.5	2.1	3.1	1,537.97	1,155.40	40.00	99.98	100	0	90	0	10.11
$\text{FO}_{S,F}$	2.5	2.0	3.8	1,494.68	1,122.51	41.50	99.99	100	0	95	0	10.35
$\text{FO}_{T,A}$	102.4	15.1	181.0	191.85	122.91	0.50	99.65	100	95	0	0	3.84
$\text{FO}_{T,B}$	161.7	40.0	242.4	73.42	32.36	0.00	99.91	100	100	45	45	5.26

Table 6.10 Computational results – experiment 1 (2/2)

Algorithm	rt_{med}	rt_{min}	rt_{max}	$\overline{gap}$	$\overline{dev}_{best}$	$\overline{r}_{ot}$	$\overline{\beta\text{-SL}}$	r_f	r_{cf}	r_{df}	r_{cdf}	$\overline{r}_{subst}$
CPX_{std}	>300	>300	>300	1,794.19	1,371.95	14.62	96.97	65	15	25	15	10.48
CPX_{1h}	>3,600	>3,600	>3,600	29.24	1.10	0.00	100.00	80	80	80	80	3.87
$\text{FO}_{TP,A}$	79.3	41.0	100.8	81.54	38.51	0.00	99.99	100	100	75	75	3.68
$\text{FO}_{TP,B}$	73.9	41.9	90.3	72.65	32.22	0.00	99.99	100	100	65	65	3.76
FO_{TS}	79.1	39.5	100.4	84.20	40.49	0.00	99.97	100	100	60	60	3.70
$\text{FO}_{TPS,A}$	67.5	44.3	88.9	77.58	35.81	0.50	99.98	100	95	70	70	3.32
$\text{FO}_{TPS,B}$	68.1	44.5	88.1	75.83	34.38	0.00	100.00	100	100	70	70	3.82
$\text{FO}_{TPS,C}$	67.7	42.3	82.2	74.74	33.46	0.50	100.00	100	95	85	80	2.97
$\text{FO}_{TPS,D}$	70.0	38.3	96.3	83.38	39.64	0.00	99.98	100	100	70	70	2.60
$\text{FO}_{TPS,E}$	69.6	43.0	87.8	169.72	107.23	1.50	99.78	100	85	35	35	6.00
$\text{FO}_{TPS,F}$	67.0	36.5	84.3	140.53	83.72	0.50	99.84	100	95	35	35	4.80
RFO_{T}	197.4	81.4	>300	496.09	344.63	1.50	98.84	100	90	85	80	4.18
RFO_{TS}	128.9	83.9	173.4	1,419.26	1,031.31	4.50	96.66	100	75	40	40	5.42
$\text{RFO}_{TPS,A}$	119.1	72.5	164.3	924.34	665.34	2.00	97.79	100	85	55	55	4.98
$\text{RFO}_{TPS,B}$	112.5	81.6	164.6	1,224.01	895.20	2.00	97.00	100	80	50	50	3.81

in subproblems of TD_3, which seems to make them harder to solve and not worth the effort. Hence, we decided to use time decomposition TD_2 for the R&F stage of all R&F&O variants.

Considering the F&O variants that only use the product-oriented decomposition ($\text{FO}_{P,\circ}$), we found that the running times were very low for all of them (below 4 s). However, the resulting solution quality was rather poor, with all average gaps greater than 1,500%. None of the obtained solutions were capacity-feasible, i.e., all of them required overtime. The variants $PR_{i,ASC}$ and $PR_{|D_i|,DESC}$ did not differ much. The observations for F&O variants that only use the substitute-oriented decomposition ($\text{FO}_{S,\circ}$) were similar: They only required a few seconds of running time, but all average gaps were around 1,500%, and all solutions still required overtime.

In contrast, the two F&O variants $\text{FO}_{T,A}$ and $\text{FO}_{T,B}$ that only use a time-oriented decomposition performed better in terms of solution quality: $\text{FO}_{T,A}$ required median running times around 100 s, and returned feasible solutions for all instances, with an average gap of 191.85%. For 19 out of 20 instances, the solution contained no overtime. However, the average β service level of 99.65% was rather poor, i.e., significant lost sales occurred in the solutions. $\text{FO}_{T,B}$ was clearly better than $\text{FO}_{T,A}$: It returned solutions with an average relative MIP gap of 73.42% and an average gap of 32.36% to the best found solution, while requiring slightly higher running times (median: 161.7 s). Thus, the overlapping time decomposition TD_3 seems to work better with F&O than the non-overlapping TD_1. This is contrast to R&F, where the non-overlapping TD_2 yielded better results than TD_3. For F&O, the increased number of binary variables in subproblems of TD_3 compared to TD_1 appears to be rewarded by higher solution quality. Hence, we decided to employ TD_3 in the F&O stage of all F&O variants that use a time-oriented decomposition.

When running a product-oriented after a time-oriented decomposition (variants $\text{FO}_{TP,A}$ and $\text{FO}_{TP,B}$), we observed that the resulting solution quality of both of them was similar to $\text{FO}_{T,B}$, with the main difference, that the percentage of demand-feasible solutions (i.e., solutions without lost sales) was higher for $\text{FO}_{TP,\circ}$ than for $\text{FO}_{T,B}$. The running times of $\text{FO}_{TP,\circ}$ were somewhat lower than those of $\text{FO}_{T,B}$ (a median of about 80 s for $\text{FO}_{TP,\circ}$ vs. about 160 s for $\text{FO}_{T,B}$). This is presumably because the time allocated to each subproblem is smaller in $\text{FO}_{TP,\circ}$ due to the higher number of subproblems, and some of the subproblems of the product-oriented decomposition could be solved quickly. $\text{FO}_{TP,B}$ on average returned better solutions than $\text{FO}_{TP,A}$, which suggests that $PR_{|D_i|,DESC}$ works better than $PR_{i,ASC}$ here.

For an F&O variant that runs a substitute-oriented decomposition after a time-oriented decomposition (FO_{TS}), we found that the running times were similar to those of $\text{FO}_{TP,A}$ and $\text{FO}_{TP,B}$, and the solution quality was minimally worse.

As to the F&O variants that sequentially run a time-, product- and substitute-oriented decomposition ($\text{FO}_{TPS,\circ}$), we made the following observations: They clearly work better with product- and substitute-oriented decomposition based on PD_3 or PD_4 (which combines PD_2 and PD_3) instead of PD_1. This indicates that the additional solution changes enabled by PD_2 and PD_3 are of use in the

algorithm. $PR_{|D_i|,DESC}$ does not clearly perform better than $PR_{i,ASC}$ and vice versa.

For $FO_{TPS,A}$–$FO_{TPS,D}$, the solution quality was comparable to $FO_{T,B}$ and $FO_{TP,\circ}$. Though, one positive difference was that $FO_{TPS,C}$ yielded a higher percentage of both capacity- and demand-feasible solutions than $FO_{T,B}$ and $FO_{TP,\circ}$ (80% vs. 45% / 75% / 65%).

When comparing $FO_{T,\circ}$, $FO_{P,\circ}$, and $FO_{S,\circ}$, it appears that the time-oriented decomposition is the most effective one for the considered CLSD-S instances. The solutions get slightly better w.r.t. to the β service level and required overtime when executing a product- and/or substitute-oriented decomposition after the time-oriented decomposition (see, e.g., $FO_{TPS,C}$).

None of the R&F&O variants ($RFO_\circ$) appears competitive with the good F&O variants. In fact, all of them worked even worse than the simple RF_{TD_2} regarding both running times and solution quality. This is likely caused by the fact that the number of subproblems is very high in $RFO_\circ$, especially in $RFO_{TPS,\circ}$, which contains $T + T + |P| + |D^{sd,\delta}|$ subproblems. Therefore, the total time budget has to be divided among a large number of R&F and F&O subproblems, so that the time budget allotted to each subproblem might get too small to find good solutions. However, RFO_T returned a high percentage of both capacity- and demand-feasible solutions (80%).

6.7 Summary and Conclusions

In this chapter, we considered a dynamic capacitated lot-sizing and scheduling problem with sequence-dependent setups and substitutions, the CLSD-S. The model assumptions were derived from a practical application in windshield interlayer production planning. After formulating the problem as an MILP model, we dealt with the topic of determining efficient or good production sequences. These are required for generating initial feasible solutions for Fix&Optimize, which is one of the CLSD-S MIP heuristics developed in Sect. 6.5. The described Fix&Optimize algorithm can make use of three types of decompositions: Time-, product-, and substitute-oriented decompositions. In addition to Fix&Optimize, we also devised Relax&Fix heuristics, which use only a time-oriented decomposition, and a hybrid of the two heuristic types named Relax&Fix&Optimize.

A main finding was that the R&F and F&O heuristic principles could easily be adapted for incorporating substitutions as they are very generic, and the inclusion of substitution options does not change the structure of the lot-sizing problem concerning its *binary* variables.

We conducted computational experiments by running several variants of the MIP heuristics on synthetic instances created by an instance generator, which is documented in Sect. 6.6.1. Summing up, the main conclusions from the computational

experiments are as follows:

- The best variants of the MIP heuristics were able to find feasible solutions with an average relative MIP gap of circa 75% for about 80% of the generated instances, which contain 20 products and 10 periods.
- The solutions found by CPLEX® within 1 h are on average better than those of all MIP heuristics, with an average relative MIP gap of 29.24%. Compared with all MIP heuristic variants, CPX_{1h} found the best solution in the majority of cases. However, CPX_{1h} could not find a feasible solution for 20% of the instances.
- F&O with time-oriented decomposition seems superior to R&F and R&F&O in terms of running times and solution quality.
- The relative MIP gaps of the solutions returned by well-performing F&O variants are still rather high for the considered type of instances (e.g., 72.65% for $\text{FO}_{TP,B}$).
- R&F works better with the non-overlapping time decomposition TD_2 than with the overlapping time decomposition TD_3.
- Regarding the time-oriented decomposition in F&O, the overlapping time decomposition TD_3 performs better than the non-overlapping time decomposition TD_1.
- As to the product-oriented decomposition in F&O, PD_3 and PD_4 seem to yield better results than the simple PD_1. Concerning the priority rules, $PR_{|D_i|,DESC}$ and $PR_{i,ASC}$ lead to similar results.
- The considered variants of F&O with only a substituted-oriented decomposition did not differ much in terms of solution quality and performance.
- In general, the time-oriented decomposition in F&O seems to have the highest positive impact on solution quality regarding the CLSD-S. Adding the product- and/or substitute-oriented decomposition apparently increases the probability that a solution without overtime and lost sales will be found.
- The F&O variants found to perform best seem able to generate feasible solutions for medium-sized instances in less than 100 s, with a solution quality that is not outstanding, but at least acceptable.

Chapter 7
Multi-Level Lot-Sizing Models with Flexible Bills-of-Materials

In this chapter, we develop two lot-sizing models for multi-level production with flexible BOMs:

- The Multi-Level Capacitated Lot-Sizing Problem with Substitution (MLCLSP-S), which is an extension of the MLCLSP (see Sect. 2.1.6) and uses hypergraphs to model substitution
- General Lot-sizing and Scheduling Problem for Multiple production Stages with Flexible Production Sequences (GLSPMS-FPS), which is an extension of the GLSPMS (see Sect. 2.1.7) that uses STN (see Sect. 3.2.4) to model flexible production sequences and flexible BOMs

The purpose of the former model, the MLCLSP-S, is to show how flexible BOMs modeled by hypergraphs can be integrated in multi-level lot-sizing models. The MLCLSP-S would be suitable for applications with discrete products and assembly structures, but is less appropriate for flexible production structures as they can be found in the process industries. The model cannot map flexible production sequences.

The latter model, the GLSPMS-FPS, uses a more general approach for modeling both flexible BOMs and flexible production sequences. It can also map by-products and offers all advantages that the GLSPMS has compared to the MLCLSP (see Sect. 2.1.8).

7.1 The Multi-Level Capacitated Lot-Sizing Problem with Substitutions

The *Multi-Level Capacitated Lot-Sizing Problem with Substitutions (MLCLSP-S)* is an extension of the MLCLSP (see Sect. 2.1.6) by flexible BOMs. The model was developed with an application with discrete products, an assembly structure and flexible BOMs in mind.

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_7,

7.1.1 Assumptions

The assumptions of the MLCLSP-S can be summarized as follows:

- Lot-sizing for multiple, discrete products (set of products P).
- The set of products contains components as well as assemblies.
- All parameters are deterministic.
- Single-level time structure.
- Finite time horizon with T periods.
- An arbitrary number of products can be set up in each period, provided that their setup times fit into the capacity (big bucket model).
- No scheduling (sequencing) of products within periods (resulting from the model assumptions, all permutations of the products' lots within a period result in the same objective value).
- Demand refers to demand classes (set of demand classes D).
- The set of demand classes contains internal demand classes (abstract products) as well as external demand classes (see Sect. 3.2.3).
- Time-varying demand of demand classes that has to be satisfied at the end of each period.
- Each demand class can be satisfied by certain products (demand-class specific substitution options).
- Assemblies are composed of certain quantities of multiple (concrete) products and abstract products.
- Substitution hypergraph is given implicitly as an AND-XOR-graph.
- Substitution ratios are arbitrary.
- Multiple capacitated production resources with finite speed, multiple products may share a resource.
- No parallel (alternative) resources.
- Capacity consumption per unit produced differ among products.
- No setup carry-over.
- Multiple production levels, Gozinto factors for assemblies are given.
- Predecessor and successor products may share the same resources.
- Time-variant sequence-independent setup costs and times.
- Setups have to be completed within a single period.
- Time-invariant linear holding costs.
- Time-varying variable production costs.
- Lead times between production stages, minimum of one period.
- Time-invariant variable conversion costs for satisfying demand of a certain class with a specific product.
- All occurring demand has to be fulfilled immediately (no relaxation).
- Initial inventories.
- Unlimited work-in-progress buffers for intermediate goods.
- Cost minimization objective.
- Integer variables for lot-sizes.

7.1.2 Formulation

Using the notations introduced in Table 7.1, the MLCLSP-S can be formulated as given below. The substitution hypergraph is specified as an AND-XOR graph (see Sect. 3.2.3) in the model: AND nodes of the corresponding AND-XOR graph correspond to assemblies and XOR nodes to internal or external demand classes. Edges from products $i \in P$ to assemblies $k \in A$ correspond to positive g^c_{ik} values, arcs from internal demand classes $j \in I$ to assemblies $k \in A$ correspond to positive g^a_{jk} values. Edges from products $i \in P$ to demand classes $j \in D$ correspond to positive a_{ij} values.

Minimize $F(q, x, I, s) =$

$$\sum_{t=1}^{T} \left(\sum_{i \in P} (f_{it} x_{it} + p_{it} q_{it} + h_i I_{it}) + \sum_{(i,j) \in E} r_{ij} s_{ijt} \right) \tag{7.1}$$

subject to

$$I_{it} = I_{i,t-1} + q_{it} - \sum_{j \in D_i} s_{ijt} - \sum_{k \in A_i} g^c_{ik} q_{k,t+l_i} \qquad i \in P,\ t = 1, \ldots, T - l_i \tag{7.2}$$

$$I_{it} = I_{i,t-1} + q_{it} - \sum_{j \in D_i} s_{ijt} \qquad i \in P,\ t = T - l_i + 1, \ldots, T \tag{7.3}$$

$$\sum_{i \in P_j} a^{-1}_{ij} s_{ijt} = d_{jt} \qquad j \in D,\ t = 1, \ldots, T \tag{7.4}$$

$$\sum_{i \in P_j} a^{-1}_{ij} s_{ijt} = \sum_{k \in A} g^a_{jk} q_{kt} \qquad j \in I,\ t = 1, \ldots, T \tag{7.5}$$

$$q_{it} \leq M \cdot x_{it} \qquad i \in P,\ t = 1, \ldots, T \tag{7.6}$$

$$\sum_{i \in P_r} \left(st_i x_{it} + \kappa^p_i q_{it} \right) \leq K_{rt} \qquad r \in R,\ t = 1, \ldots, T \tag{7.7}$$

$$q_{it} \in \mathbb{Z}^+_0 \qquad i \in P,\ t = 1, \ldots, T \tag{7.8}$$

$$I_{it} \geq 0 \qquad i \in P,\ t = 1, \ldots, T \tag{7.9}$$

$$x_{it} \in \{0, 1\} \qquad i \in P,\ t = 1, \ldots, T \tag{7.10}$$

$$s_{ijt} \in \mathbb{Z}^+_0 \qquad (i, j) \in E,\ t = 1, \ldots, T \tag{7.11}$$

In the MLCLSP-S, variable production costs have to be included in the objective (7.1) due to the substitution options, in contrast to the standard MLCLSP objective. For the sake of generality, we additionally assume that the variable production costs are time-varying. The inventory balance equations are given by (7.2) and (7.3): (7.2)

Table 7.1 Notations for MLCLSP-S

Symbol	Definition
Constants	
$t = 1, \ldots, T$	Periods
n_r	Number of resources
Indices and sets	
$i \in P$	(Concrete) products
$i \in C \subseteq P$	Components
$k \in A = P \backslash C$	Assemblies
$j \in B$	Demand classes
$j \in I \subseteq B$	Internal demand classes (abstract products)
$j \in D = B \backslash I$	External demand classes
$t = 1, \ldots, T$	Periods
$V = P \cup B$	Vertex set of substitution graph
$E \subseteq P \times B$	Arcs of substitution graph denoting feasible substitutions: $(i, j) \in E$ if product i can fulfil demand of internal or external demand class j
$G = (V, E)$	Substitution graph
$D_i = \{j \in B \mid (i, j) \in E\}$	Set of demand classes whose demand can be fulfilled by product i
$P_j = \{i \in P \mid (i, j) \in E\}$	Set of products that can fulfil demand of class j
$r \in R = \{1, \ldots, n_r\}$	Resources
P_r	Set of products that are manufactured using resource r
Parameters	
d_{jt}	Demand of class j in period t
h_i	Non-negative holding cost per period for storing one unit of product i
p_{it}	Unit production cost of product i in period t
r_{ij}	Substitution cost for fulfilling demand class j by product i per unit
I_{i0}	Initial inventory of product i
f_{it}	Fixed setup or order cost for product i in period t
$l_i \geq 1$	Lead time of product i (time unit: *number of periods*)
K_{rt}	Capacity of resource r available in period t
st_i	Production setup time for product i
κ_i^p	Capacity required for manufacturing one unit of product i
g_{jk}^a	Number of units of internal demand class j required for one unit of assembly k (Gozinto factor)
g_{ik}^c	Number of units of concrete component or assembly i required for one unit of assembly k (Gozinto factor)
$A_j = \left\{k \mid g_{jk}^a > 0\right\}$	Set of assemblies that require the internal demand class j
a_{ij}	Number of units of component or assembly i that substitute for one unit of internal or external demand class j
Variables	
q_{it}	Production or order quantity of product i in period t
I_{it}	Inventory of product i at the end of period t
x_{it}	Binary variable that indicates whether a setup for product i occurs in period t
s_{ijt}	Quantity of product i used to fulfil demand of class j in period t

denotes that the inventory of component or assembly i at the end of period t is the inventory at the end of the previous period plus the production quantity minus the demand of internal and external demand classes fulfilled by units of i and minus the units of i consumed for producing successor products. Equation (7.3) is required analogously to (2.41) as there is no inventory consumption by production of successor products in the last l_i periods due to the lead time between production stages. The demand of internal or external demand classes fulfilled by the product i is represented by the sum of conversion quantities s_{ijt} over all demand classes that can be satisfied by the product. The quantity of i consumed for producing successor products is given by multiplying the production quantities of successor products with the respective Gozinto factors and summing these terms up.

Equation (7.4) ensures that the external demand d_{jt} is always satisfied completely using the available substitution options. It is assumed that all g^a_{jk} are zero for all external demand classes $j \in D$. Equation (7.5) ensures that the demand of an internal demand class $j \in I$ caused by the production of assemblies that require units of j are fully satisfied by feasible products. It is assumed that all d_{jt} are zero for all internal demand classes. The setup forcing constraints (7.6) and capacity constraints (7.7) equal those of the MLCLSP. Equations (7.8)–(7.11) define the variable domains.

Alternatively to distinguishing between the sets P and D in the model, one could also define the sets in the model in a way that the set P contains partly overlapping sets: A set C of components, a set B of abstract products, a set D of external demand classes, and a set A of assemblies.

7.1.3 Example of Substitution Hypergraph

Figure 7.1 illustrates an exemplary AND-XOR substitution graph for the MLCLSP-S: It is assumed that the problem instance has three demand classes and 12 products. Demand class 1 is an external demand class, i.e., $D = \{1\}$, whereas 2 and 3 are internal demand classes, i.e., $I = \{2, 3\}$. The set of products $P = \{1, \ldots, 12\}$ can be partitioned into the set of assemblies $A = \{1, 2, 6, 9\}$ and the set of components $C = \{3, 4, 5, 7, 8, 10, 11, 12\}$.

In the graph, the products are labeled $P1, \ldots, P12$ and the demand classes $D1, D2, D3$. The single arc $(P1, D1)$ going into $D1$ denotes that the external demand class $D1$ can only be fulfilled by assembly $P1$. $P1$ consists of $g^c_{21} = 3$ units of subassembly $P2$ and $g^a_{31} = 2$ units of another "flexible" part $D3$. Assembly $P2$ consists of $g^c_{32} = 4$ units of a component $P3$ and $g^a_{22} = 1$ unit of a flexible part $D2$. This internal demand class $D2$ can either be satisfied by $a_{42} = 2$ units of component $P4$ or $a_{52} = 1$ unit of component $P5$. The flexible part $D3$ may either be $a_{63} = 1$ unit of subassembly $P6$ or $a_{93} = 1$ unit of another compatible subassembly $P9$. Assembly $P6$ is composed of $g^c_{76} = 4$ units of $P7$ and $g^c_{86} = 1$ unit of $P8$. Assembly $P9$ is composed of $g^c_{10,9} = 2$ units of $P10$ and $g^c_{11,9} = g^c_{12,9} = 1$ unit each of $P11$ and $P12$.

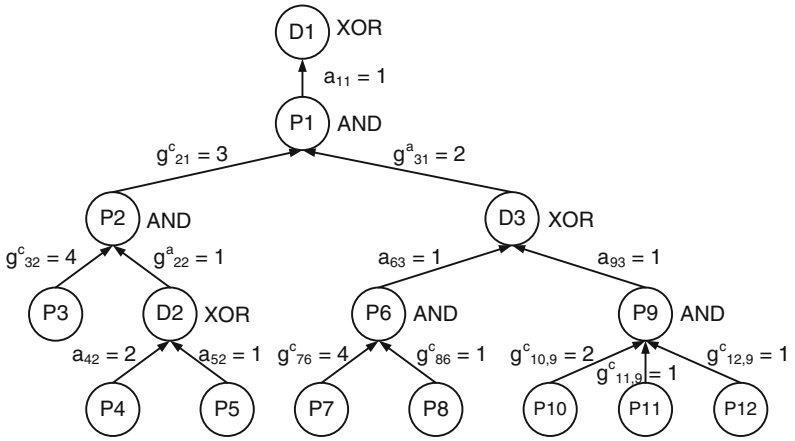


Fig. 7.1 Example of an AND-XOR substitution graph for the MLCLSP-S

7.1.4 Transformation into Special Case of MLFP

Note that it is possible to transform the MLCLSP-S into a special case of an extension of the MLFP (see Sect. 4.2.3) with setup times:

- Introduce one MLFP product for each MLCLSP-S product (component or assembly) and demand class.
- Introduce one MLFLP resource for each MLCLSP-S resource with the same capacity values as the corresponding MLCLSP-S resource.
- Introduce one task for each MLCLSP-S component $i \in C$. This task requires no input product and produces one unit of the corresponding component per unit. The setup costs and variable production costs of that task equal those of the corresponding MLCLSP-S component. It has the same setup time and capacity consumption per unit as the corresponding MLCLSP-S component.
- Introduce one task for each feasible conversion $(i, j) \in E$ of an MLCLSP-S product i into an MLCLSP-S demand class j. This task requires one unit of i and produces a_{ij}^{-1} units of j. Its setup costs are 0, and its variable production costs equal the MLCLSP-S conversion costs r_{ij}.
- Introduce one task for each MLCLSP-S assembly $k \in A$. This task requires g_{hk}^{a} units of each internal demand class h as well as g_{ik}^{c} units of each product i and produces one unit of the corresponding assembly per unit. The setup costs and variable production costs of that task equal those of the corresponding MLCLSP-S assembly. It has the same setup time and capacity consumption per unit as the corresponding MLCLSP-S assembly.
- The holding costs of MLFP products corresponding to MLCLSP-S products equal the holding costs of those. However, the holding costs of MLFP products corresponding to internal and external MLCLSP-S demand classes have to be set to a high penalty value M, as the MLCLSP-S assumes that no storage takes place after conversion of a product for a demand class.

7.1.5 Echelon Stocks and Flexible BOMs

An obvious consequence of flexible BOMs is that echelon stocks cannot be specified exactly as the Gozinto factors are ambiguous. Hence, standard cuts for multi-level capacitated lot-sizing that use the concept of echelon stocks cannot be applied in the usual way. One approach for circumventing this problem developed by Begnaud et al. (2006) is to work with "minimal values" for Gozinto factors, i.e., to calculate the minimum quantity g_{ik}^{min} of product i that will be contained in product k considering all possible production sequences for k. The Gozinto factors are "replaced" by these minimal values. However, this approach only works if all possible ways of producing k use at least one unit of i. This property is violated as soon as it is possible to fully substitute i by another product j. In that case, g_{ik}^{min} becomes 0, which might render the corresponding valid inequalities useless.

7.2 The General Lot-Sizing and Scheduling Problem for Multiple Production Stages with Flexible Production Sequences

The *General Lot-sizing and Scheduling Problem for Multiple production Stages with Flexible Production Sequences (GLSPMS-FPS)* extends the GLSPMS (see Sect. 2.1.7) by an STN-based approach (see Sect. 3.2.4) for modeling flexible production sequences and flexible BOMs. The model can be seen as a combination of the GLSPMS and the MLFP (see Sect. 4.2.3).

7.2.1 Assumptions

The assumptions of the GLSPMS-FPS are as follows:

- Lot-sizing for multiple, continuous tasks (set of tasks A).
- Executing one time unit of a task requires certain quantities of one or multiple input products and (simultaneously) produces one or multiple output products.
- All parameters are deterministic.
- No lead times.
- Two-level time structure.
- Finite time horizon with T macro-periods and S micro-periods.
- Each macro-period contains a predetermined set of micro-periods, the number of micro-periods can differ among macro-periods.
- The beginning and duration of each macro-period are fixed.
- Micro-period beginnings and lengths are flexible, apart from some micro-periods with explicitly fixed beginnings; these fixed beginnings are required for micro-periods that are the first micro-period in a macro-period and for modeling predetermined fixed exogenous downtime of resources.

- Only a single changeover is possible between each pair of micro-periods.
- Scheduling (sequencing) of products within macro-periods.
- Time-varying demand for products that has to be satisfied at the end of each macro-period.
- Demand refers to products, some or all of which are substitutable.
- Multiple capacitated production resources with finite speed.
- Each task can only be executed on a specified subset of resources.
- Executing one time unit of a task consumes a certain amount of capacity of exactly one resource.
- Execution of one unit of a task does not simultaneously consume multiple resources, but the same task can simultaneously be executed on multiple resources.
- Multiple production levels.
- Heterogeneous parallel resources.
- Resources may be shared between production levels, i.e., between predecessor and successor products.
- Capacity consumption per execution unit differ among tasks and also among resources regarding a certain task.
- Setup carry-over is possible.
- Time-invariant sequence-dependent setup costs and setup times (these consume capacity) for *tasks*.
- Changeovers are started towards the end of a micro-period and may continue into the subsequent micro-period (setup time splitting).
- Time-invariant linear holding costs that are incurred for inventory at the end of each macro-period.
- Time-invariant production costs for tasks that differ among the parallel resources and thus have to be included.
- Time-invariant costs for preservation of setup state for a task on a certain resource (per time unit).
- All occurring demand has to be fulfilled immediately (no relaxation).
- Initial inventories.
- Quantities produced in a micro-period can be used for satisfying (successor) demand in the same micro-period (after they have been produced).
- Limited work-in-progress buffers for intermediate goods within micro-periods, thus production on a predecessor stage is possible while a setup is performed on the successor stage (production quantity splitting is necessary to model this and the previous aspect).
- Minimum task execution quantity after changeover.
- Minimum time for resource staying in idle state (without production).
- Initial setup state given for each resource.
- Cost minimization objective.
- Continuous variables for task execution quantities.

7.2.2 Formulation

Table 7.2 contains the notations used for formulating the GLSPMS-FPS. We added dimension units to parameters and variables in the table to ease understanding. MU stands for monetary units, TU for time/capacity units, AU_a for task execution

Table 7.2 Notations for GLSPMS-FPS

Symbol	Definition
Constants	
m	Number of products
T	Number of macro-periods
S	Number of micro-periods
n_r	Number of resources
n_a	Number of tasks
Indices and sets	
$i \in P = \{1, \ldots, m\}$	Products
$t = 1, \ldots, T$	Macro-periods
$s = 1, \ldots, S$	Micro-periods
S_t	Set of micro-periods belonging to macro-period t ($\subseteq S$)
$e(t)$	Last micro-period of macro-period t
Φ	Set of micro-periods
Φ^{fix}	Set of micro-periods with fixed beginnings
Φ_r^o	Set of micro-periods during which production on r is forbidden
Φ^l	Set of micro-periods that are the last micro-period in a macro-period
$r \in R = \{1, \ldots, n_r\}$	Resources
$a \in A = \{0, \ldots, n_a\}$	Tasks including dummy task 0
$0 \in A$	Dummy task for modeling time during which a resource is not set up for any task
A_r	Set of tasks that can be executed using resource r
Parameters	
d_{is}	Demand for product i in micro-period s (= 0 for all $s \in \Phi \backslash \Phi^l$) $[QU_i]$
I_i^{max}	Upper inventory limit for product i $[QU_i]$
$\overline{w}_s$	Fixed beginning for micro-period s $[TU]$
$I_{ri}^{w,max}$	Upper inventory limit for units of product i produced on resource r that are not consumed before the next micro-period $[QU_i]$
h_i	Non-negative holding cost per macro-period for storing one unit of product i $[MU/QU_i]$
I_{i0}	Initial inventory of product i $[QU_i]$
p_{ras}	Unit production cost of task a on resource r in micro-period s $[MU/AU_a]$
$\overline{p}_{ras}$	Cost for preserving setup state of task a on resource r in micro-period s per capacity unit $[MU/TU]$
f_{rab}	Setup cost that is incurred when the setup state of resource r changes from task a to b $[MU]$
st_{rab}	Setup time for changeover from task a to b on resource r $[TU]$
κ_{ra}^p	Time required for executing one unit of task a on resource r $[TU/AU_a]$

(*continued*)

Table 7.2 (continued)

Symbol	Definition
m_{ra}	Minimum execution quantity for task a after changeover on resource r $[AU_a]$
ρ_{ia}	Number of units of product i required for executing one unit of task a $[QU_i/AU_a]$
σ_{ia}	Number of units of product i produced by executing one unit of task a $[QU_i/AU_a]$
$z_{ra1} = x_{raa1}$	Binary parameter that indicates whether resource r is already set up for task a at the beginning of the first micro-period
g_{ik}^{min}	Minimum quantity of product i required for producing one unit of its successor product k (lower bound for Gozinto factor) $[QU_i/QU_k]$
g_{ik}^{max}	Maximum quantity of product i required for producing one unit of its successor product k (upper bound for Gozinto factor) $[QU_i/QU_k]$
Variables	
I_{is}	Inventory of product i at the end of *micro*-period s $[QU_i]$
q_{ris}^{0}	Quantity of product i produced on r in micro-period s available for consumption in the same period $[QU_i]$
q_{ris}^{+1}	Quantity of product i produced on r in micro-period s available for consumption in the next period $s+1$ $[QU_i]$
w_s	Beginning of micro-period s (on continuous time axis) $[TU]$
y_{rs}^{b}	Setup time consumed on resource r at beginning of micro-period s $[TU]$
y_{rs}^{e}	Setup time consumed on resource r at end of micro-period s $[TU]$
q_{ras}^{θ}	Units of task a performed on resource r in micro-period s $[AU_a]$
q_{ris}	Production quantity of product i on resource r in micro-period s (auxiliary variable) $[QU_i]$
x_{rabs}	Binary variable that indicates whether a changeover from task a to task b is performed on resource r starting at the end of micro-period $s-1$ and continuing into micro-period s
z_{ras}	Binary variable that indicates whether resource r is already set up for task a at the beginning of micro-period s or a changeover to it started in $s-1$ is completed in s
Ψ_{ras}^{θ}	Time during which setup state of task a on resource r is preserved in micro-period s without production $[TU]$

units of task a, and QU_i for quantity units of product i. The GLSPMS-FPS can be formulated as given in the following:

$$\begin{aligned}\text{Minimize } F(q, x, z, I, \Psi^{\theta}, q^{0}, q^{+1}, w, y^{b}, y^{e}) \\ = \sum_{t=1}^{T} \sum_{i \in P} h_i \left(I_{i,e(t)} + \sum_{r \in R} q_{ri,e(t)}^{+1} \right) \\ + \sum_{s=1}^{S} \sum_{r \in R} \sum_{a \in A_r} \left(p_{ras} q_{ras}^{\theta} + \overline{p}_{ras} \Psi_{ras}^{\theta} + \sum_{b \in A_r} f_{rab} x_{rabs} \right) \end{aligned} \quad (7.12)$$

subject to

$$q_{ris}^{0} + q_{ris}^{+1} = q_{ris} \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{7.13}$$

$$q_{ris} = \sum_{a \in A_r} \sigma_{ia} q_{ras}^{\theta} \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{7.14}$$

$$I_{is} = I_{i,s-1} + \sum_{r \in R} \left(q_{ris}^{0} + q_{ri,s-1}^{+1} \right) - d_{is} - \sum_{r \in R} \sum_{a \in A_r} \rho_{ia} q_{ras}^{\theta} \qquad i \in P,\ s = 1, \ldots, S \tag{7.15}$$

$$I_{is} \le I_{i}^{max} \qquad i \in P,\ s = 1, \ldots, S \tag{7.16}$$

$$w_s = \overline{w}_s \qquad s \in \Phi^{fix} \tag{7.17}$$

$$w_{s+1} - w_s = y_{rs}^{b} + \sum_{a \in A_r} \left(\Psi_{ras}^{\theta} + \kappa_{ra}^{p} q_{ras}^{\theta} \right) + y_{rs}^{e} \qquad r \in R,\ s = 1, \ldots, S \tag{7.18}$$

$$q_{ras}^{\theta} = 0 \qquad r \in R,\ a \in A_r,\ s \in \Phi_r^{o} \tag{7.19}$$

$$\Psi_{ras}^{\theta} + \kappa_{ra}^{p} q_{ras}^{\theta} \le M \cdot z_{ras} \qquad r \in R,\ a \in A_r,\ s = 1, \ldots, S \tag{7.20}$$

$$q_{ras}^{\theta} \ge m_{ra} \sum_{b \in A_r \setminus \{a\}} x_{rbas} \qquad r \in R,\ a \in A_r,\ s = 1, \ldots, S \tag{7.21}$$

$$\sum_{a \in A_r} z_{ras} = 1 \qquad r \in R,\ s = 1, \ldots, S \tag{7.22}$$

$$x_{rabs} \ge z_{ra,s-1} + z_{rbs} - 1 \qquad r \in R,\ a, b \in A_r,\ s = 2, \ldots, S \tag{7.23}$$

$$y_{r,s-1}^{e} + y_{rs}^{b} = \sum_{a,b \in A_r} st_{rab} x_{rabs} \qquad r \in R,\ s = 2, \ldots, S \tag{7.24}$$

$$q_{ris}^{+1} \le I_{ri}^{w,max} \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{7.25}$$

$$\sum_{a,b \in A_r} x_{rabs} = 1 \qquad r \in R,\ s = 2, \ldots, S \tag{7.26}$$

$$w_s - w_{s-1} \ge y_{r_1,s-1}^{b} + \kappa_{r_1 a}^{p} \frac{q_{r_1,i,s-1}^{0}}{\sigma_{ia}} + y_{r_2,s-1}^{e} \qquad s \ge 2,\ r_1, r_2 \in R,\ a \in A_{r_1}, b \in A_{r_2},\ i, k \in P,\ \sigma_{ia} > 0,\ \rho_{kb} > 0,\ g_{ik}^{max} > 0,\ \frac{g_{ik}^{max}}{\sigma_{ia}} \kappa_{r_1 a}^{p} > \frac{1}{\rho_{kb}} \kappa_{r_2 b}^{p} \tag{7.27}$$

$$w_s - w_{s-1} \ge y_{r_1,s-1}^{b} + \kappa_{r_2 b}^{p} q_{r_2,b,s-1}^{\theta} + y_{r_2,s-1}^{e} \qquad s \ge 2,\ r_1, r_2 \in R,\ a \in A_{r_1}, b \in A_{r_2},\ i, k \in P,\ \sigma_{ia} > 0,\ \rho_{kb} > 0,\ g_{ik}^{max} > 0,\ \frac{g_{ik}^{min}}{\sigma_{ia}} \kappa_{r_1 a}^{p} < \frac{1}{\rho_{kb}} \kappa_{r_2 b}^{p} \tag{7.28}$$

$$y^b_{r1} = y^e_{rS} = 0 \qquad r \in R \tag{7.29}$$

$$q^\theta_{ras}, \Psi^\theta_{ras} \geq 0 \qquad r \in R,\ a \in A_r,\ s = 1, \ldots, S \tag{7.30}$$

$$I_{is} \geq 0 \qquad i \in P,\ s = 1, \ldots, S \tag{7.31}$$

$$x_{rabs} \geq 0 \qquad r \in R,\ a, b \in A_r,\ s = 2, \ldots, S \tag{7.32}$$

$$z_{ras} \in \{0, 1\} \qquad r \in R,\ a \in A_r,\ s = 2, \ldots, S \tag{7.33}$$

$$w_s \geq 0 \qquad s = 1, \ldots, S + 1 \tag{7.34}$$

$$q_{ris} \geq 0 \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{7.35}$$

$$q^0_{ris}, q^{+1}_{ris} \geq 0 \qquad r \in R,\ i \in P,\ s = 1, \ldots, S \tag{7.36}$$

$$y^b_{rs}, y^e_{rs} \geq 0 \qquad r \in R,\ s = 1, \ldots, S \tag{7.37}$$

The GLSPMS-FPS introduces a dummy *task* 0 for modeling the state where the resource is not set up for any task: If a changeover to task 0 occurs, this means that the previous setup state for another task gets lost and the resource is not set up for any real task. This dummy task serves the same purpose as the dummy product introduced in the GLSPMS. No dummy *product* is required in the GLSPMS-FPS.

The q^θ_{ras} variables describe "execution quantities", i.e., they indicate how often activity a is repeated / for how long it is executed. The multiplication with κ^p_{ra} in (4.21) converts these execution quantities into time/capacity units. The minimum execution quantity m_{ra} ensures a minimum duration of a production lot and a minimum lot size for task a on resource r in combination with the parameters κ^p_{ra} and σ_{ia}.

The objective (7.12) consists of holding costs, variable task execution (production) costs, setup state preservation costs, and sequence-dependent setup costs. The splitting of production quantities into a part q^0_{ris} that can be used in the same micro-period s where the production takes place and another part q^{+1}_{ris} that cannot be used before the next period $s + 1$ is implemented by (7.13). This splitting is necessary for formulating the constraints (7.27) and (7.28), which we explain in Sect. 7.2.3. Equation (7.14) is required to set the correct values of the auxiliary variables q_{ris}.

The inventory balance equations are given by (7.15): As the model is based on STN, the equations contain the inventory changes caused by the execution of tasks as well as those caused by primary demand. In addition, the inventory increase by production is split in a part q^0_{ris} that originates from production in the same micro-period s and a part that originates from production in the previous micro-period $s - 1$. As in the GLSPMS, this production quantity splitting is necessary to ensure that the usage of units produced in a micro-period by a successor product in the same micro-period is always temporally feasible in the real-world application. Again, q^{+1}_{ri0} is a constant with value 0. Equation (7.16) limits the inventory of each product to a certain maximum level. The constraints (7.17) fix the beginnings of a subset of the micro-periods to certain points in time.

The constrained capacity of each production resource is modeled by (7.18): It strongly resembles (2.50) and ensures that the capacity consumption of tasks and their sequence-dependent setup times never exceed the effective duration of each micro-period. The time during which the setup state of a certain task a is preserved without production is captured explicitly by the Ψ^θ_{ras} variables because setup state

preservation incurs costs as in the GLSPMS. Hence, the capacity constraint is an equation as in the GLSP and GLSPMS. Equation (7.19) ensures that no production takes place on r in a micro-period s during which production is forbidden. Equation (7.20) enforces that execution of task a on a resource r only takes place in micro-period s if the resource is already set up for a at the beginning of s or a changeover to a is performed in s. Minimum execution quantities after the changeover to a task are enforced by (7.21).

Equation (7.22) means that exactly one task a is set up on the resource at the beginning of each micro-period s. Equation (7.23) ensures that the changeover variable x_{rabs} becomes one if task a was set up in micro-period $s-1$ and b is set up in s, which implies that a changeover must have been performed. Equation (7.24) splits up the setup time of a setup activity that starts in $s-1$ and continues into s, and thus overlaps the micro-period boundaries: A part $y^e_{r,s-1}$ of the setup time is scheduled at the end of $s-1$, the remaining part y^b_{rs} at the beginning of s. Note that all y^b_{r1} and y^e_{rS} should be fixed to 0 or omitted from the model.

Equation (7.25) specifies upper limits for the work-in-progress buffers after resources between successive production stages. Equation (7.26) makes sure that exactly one changeover occurs in each micro-period. Note that if $x_{raas} = 1$, this denotes that task a remains set up. Equations (7.27) and (7.28) will be explained in Sect. 7.2.3. Equations (7.30)–(7.37) define the variable domains.

Substitution options can be mapped in the GLSPMS-FPS by introducing two or more tasks that consume different input products but have the same output product(s) (also see Sect. 3.2.4). The substitution options are implicitly described by the ρ_{ia} and σ_{ia} parameters, and substitution decisions can be derived from the q^{θ}_{ras} variable values of a solution.

An obvious consequence of flexible production sequences is that echelon stocks cannot be specified exactly (see Sect. 7.1), which complicates the development of valid inequalities for the model.

7.2.3 Ensuring Temporal Feasibility within Micro-Periods

In addition, the GLSPMS-FPS requires two additional constraint groups that correspond to (7.27) and (7.28) in the GLSPMS. However, developing these constraints is not as straightforward as it might seem at first glance: As the GLSPMS-FPS allows for flexible production sequences, various ways can be feasible for producing a certain product, each potentially corresponding to a different BOM containing different predecessor products. Hence, three problems arise:

1. In many cases, one cannot state for all possible BOMs of a certain product k whether another product i is a (direct or indirect) predecessor of it. E.g., there might be one task a that produces k from an intermediate good i and another task b that produces k from another intermediate good j. In this example, i is predecessor of k in the former case, whereas it is no predecessor in the latter case. This is illustrated by Fig. 7.2(a).

(a) Product i is only actual predecessor of k if a used

(b) Product i is predecessor of k independent of task

Fig. 7.2 Examples of possible STNs in GLSPMS-FPS

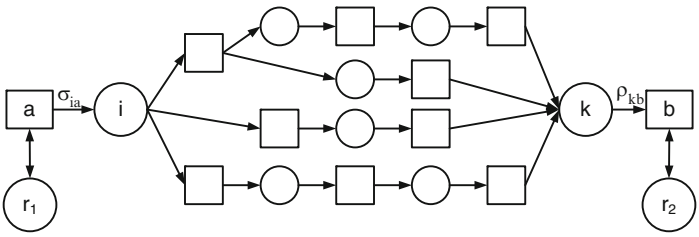


Fig. 7.3 STN example where i is an indirect predecessor of k

2. Even if all of several (alternative) tasks producing a certain product k consume one or more units of i, the Gozinto factor g_{ik} (as it was used in the MLCLSP) becomes ambiguous as soon as the ratios ρ_{ia}/σ_{ka} differ between various tasks $a \in A$. Each of these ratios denotes the (direct) Gozinto factor for the pair of products with respect to a certain task a. Fig. 7.2(b) illustrates this in an example with two tasks a and b.
3. The multi-level production structure of the model further complicates the situation, as there might be a myriad of ways of producing a certain product, each using a different flow of tasks and potentially a different BOM. E.g., even if we know for sure that all possible ways of producing k require units of i, it depends on the actual way used whether the predecessor i is "slower" or "faster" than its (direct or indirect) successor k because the ways might use different resources and the (indirect) Gozinto factor might be ambiguous. Fig. 7.3 demonstrates this with an example where product k is used by a task b on resource r_2 and always requires units of i which is produced by task a on resource r_1, no matter which of the shown alternative ways of producing k is actually used.

In the following, we say that product i is a direct predecessor of product k and k is a direct successor of i if a task $a \in A$ exists with $\rho_{ia} > 0$ and $\sigma_{ka} > 0$. We define the set of input products of a task a as $P_a^i = \{i | i \in P \wedge \rho_{ia} > 0\}$, and analogously the set of output products of a as $P_a^o = \{i | i \in P \wedge \sigma_{ia} > 0\}$. We define the set of tasks that consume i as $A_i^i = \{a | a \in A \wedge \rho_{ia} > 0\}$, and analogously the set of tasks that produce i as $A_i^o = \{a | a \in A \wedge \sigma_{ia} > 0\}$. A product i is termed an indirect predecessor of product k and k an indirect successor of i if a sequence $p_1, p_2, \ldots, p_{n-1}, p_n$ of products exists so that i is a direct predecessor of p_1, p_1 a direct predecessor of p_2, ..., p_{n-1} a direct predecessor of p_n, and p_n a direct predecessor of k. Direct and indirect successors are defined analogously. Note

that all these definitions of direct and indirect predecessor and successor concepts only refer to *potential* relationships. Even if one product is *potential* predecessor of another product, some ways of producing the latter might exist that do not consume its potential predecessor.

7.2.3.1 Calculating Minimal and Maximal Values for Gozinto Factors

In Sect. 7.1, we already mentioned the approach of calculating the minimum quantity g_{ik}^{min} of a product i that will doubtless be required for producing one unit of a product k. In order to develop the constraints corresponding to (2.60) and (2.61), we will need this value as well as the maximum quantity g_{ik}^{max} of i that can be required for producing one unit of k.

A dynamic programming approach for calculating all g_{ik}^{min} values with $O(|P|^2|A|)$ complexity is given by Begnaud et al. (2006): Assuming that the production structure is acyclic, the products in the set P can be renumbered in a way that each product has a lower index than its successors. With g_{ii}^{min} set to 1 for all $i \in P$, the g_{ik}^{min} values can then be calculated recursively using the following formula by iterating from $k = 1, \ldots, |P|$ and $i = k + 1, \ldots, |P|$:

$$g_{ik}^{min} = \min_{a \in A_k^o} \left\{ \frac{\sum_{j \in P} g_{ij}^{min} \rho_{ja}}{\sigma_{ka}} \right\} \tag{7.38}$$

Analogously, with g_{ii}^{max} set to 1 for all $i \in P$, the g_{ik}^{max} values can be calculated recursively with $O(|P|^2|A|)$ complexity as follows by iterating from $k = 1, \ldots, |P|$ and $i = k + 1, \ldots, |P|$:

$$g_{ik}^{max} = \max_{a \in A_k^o} \left\{ \frac{\sum_{j \in P} g_{ij}^{max} \rho_{ja}}{\sigma_{ka}} \right\} \tag{7.39}$$

The meaning of (7.38) is illustrated by Fig. 7.4: Assume that the minimum quantities of i contained in each of the intermediate products j_1, j_2, and j_3 have already been calculated, with "best" production sequences with respect to the consumption of i corresponding to them. The minimum quantity of i that can be contained in k if produced by task a_1 can be determined by summing up the minimum quantities of i contained in the intermediate products required for producing one unit of k with task a_1. The reason why this sum is necessary is exemplified by Fig. 7.4: If i is potentially used in more than one input product $j \in P_a^i$ of task a, this might increase the minimum quantity contained in k. The same calculation is performed for the alternative task a_2. Comparing the minimum required quantities of i of all alternative tasks that output k then yields the sought-after value for g_{ik}^{min}.

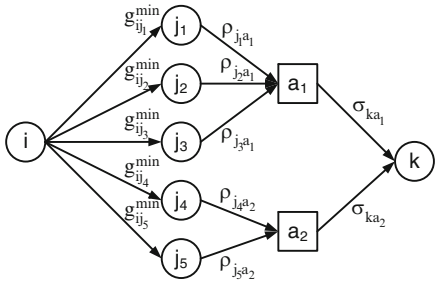


Fig. 7.4 Illustration of recursion (7.38) for determining g_{ik}^{min} values

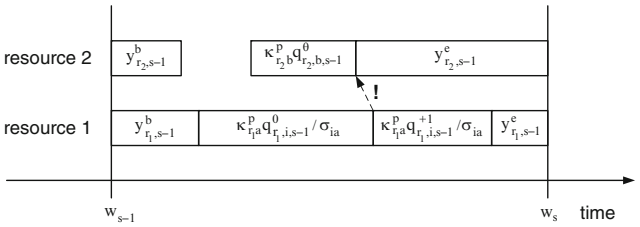
Correspondingly, the idea contained in (7.39) is as follows: Suppose that the maximum quantities of i contained in each of the intermediate products j_1, j_2, and j_3 are already known, with "worst" production sequences with respect to the consumption of i corresponding to them. The maximum quantity of i contained in k if produced by task a_1 is given by the sum of the maximum quantities of i contained in the intermediate products required for producing one unit of k with task a_1. The maximum quantity of i in k when using the alternative task a_2 can be calculated analogously. The correct value for g_{ik}^{max} can be obtained by comparing the maximum required quantities of i of all alternative tasks that output k.

7.2.3.2 Constraints for Potentially Faster Successors

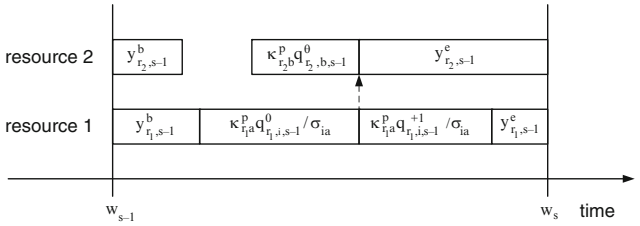
The constraint (7.27) described in the following corresponds to (2.60) in the GLSPMS. Figure 7.5 illustrates why (7.27) is required: Without (7.27), the schedule shown in Fig. 7.5a could be contained in a "feasible" solution. The task b (executed on resource r_2) consuming units of a product k is assumed to be potentially faster with respect to the consumption/production of k than another task a (executed on resource r_1) that could produce the potential direct or indirect predecessor i of k. In this context, the statement that b is potentially faster than a w.r.t. k and its predecessor i denotes that at least one production sequence with corresponding (fixed) BOM exists where k is faster than i according to the definition of "faster" for the fixed-BOMs case in the standard GLSPMS. Mathematically this can be expressed as follows:

$$\frac{g_{ik}^{max}}{\sigma_{ia}} \kappa_{r_1 a}^{p} > \frac{1}{\rho_{kb}} \kappa_{r_2 b}^{p} \tag{7.40}$$

In (7.40), g_{ik}^{max} can be interpreted as the "worst-case" consumption of i for producing k, i.e., the highest possible consumption with a feasible BOM. The left-hand side thus represents the maximum production time of task a on resource r_1 to output units of i required for supplying predecessor units for one unit of k. The right-hand side denotes the production time of task b on resource r_2 during which exactly one unit of k is consumed.



(a) Practically infeasible solution that may occur without (7.27)



(b) Schedule ensured by (7.27)

Fig. 7.5 Example showing necessity of (7.27)

In the schedule shown in Fig. 7.5, the split changeover time $y^e_{r_2,s-1}$ at the end of $s-1$ is so long that the faster execution of task b consuming k would have to stop before a sufficient quantity of task a has been executed to output $q^0_{r_1 i,s-1}$ units of i. If less than $q^0_{r_1 i,s-1}$ units of i are produced, it is not guaranteed that the maximum quantity of predecessor i consumed in $s-1$ can be satisfied, assuming that there are no stocks of i at the beginning of $s-1$. Thus, it might not be possible to implement the resulting schedule due to its violation of temporal constraints.

Equation (7.27) ensures that the changeover to and execution of a task a producing predecessor i on resource r_1 and the split changeover from task b producing successor k on resource r_2 (on a subsequent production stage) to another task all fit into the duration $w_s - w_{s-1}$ of a micro-period $s-1$.[1] Here, only the execution quantity of a required for the production of $q^0_{r_1 i,s-1}$ of i that can be used in $s-1$ is considered. The term $\kappa^p_{r_1 a} \frac{q^0_{r_1,i,s-1}}{\sigma_{ia}}$ represents the capacity consumption of the execution quantity of task a required for outputting $q^0_{r_1 i,s-1}$ units of i.

Note that (7.27) also considers the case where two tasks a and b are directly linked by a product i, i.e., $\sigma_{ia} > 0$ and $\rho_{ib} > 0$: This setting is described by the constraints of (7.27) with $i = k$. The corresponding STN fragment is visualized in

[1] Analogously to (2.60) (see footnote 5 on p. 50), (7.27) is more restrictive than required for ensuring temporal feasibility. It can be reformulated by introducing variables y^b_{rabs} and y^e_{rabs} that denote the setup time consumed by a changeover from task a to b on resource r at the beginning and end of micro-period s, respectively.

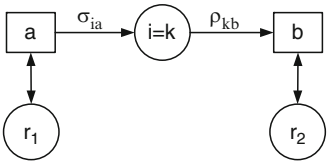
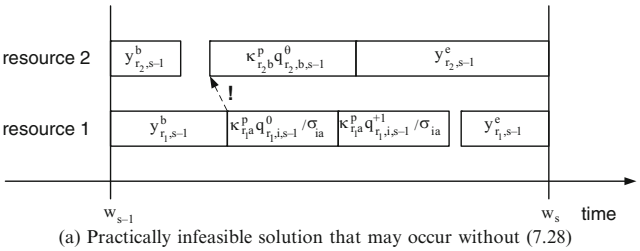
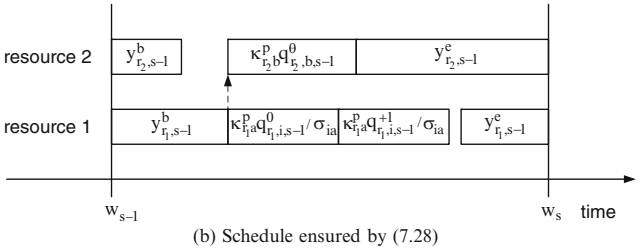


Fig. 7.6 STN example where tasks a and b are directly linked by a product $i = k$



(a) Practically infeasible solution that may occur without (7.28)



(b) Schedule ensured by (7.28)

Fig. 7.7 Example showing necessity of (7.28)

Fig. 7.6. As mentioned earlier, g_{ii}^{min} and g_{ii}^{max} are simply set to 1 for all $i \in P$. A schedule with the property enforced by (7.27) is shown in Fig. 7.5(b).

7.2.3.3 Constraints for Potentially Slower Successors

We will now consider constraint (7.28) that corresponds to (2.61) in the GLSPMS: The necessity of (7.28) is exemplified by Fig. 7.7: Without (7.28), the schedule shown in Fig. 7.7(a) could be contained in a "feasible" solution. The task b (executed on resource r_2) consuming units of a product k is now assumed to be *potentially slower* with respect to the consumption/production of k than another task a (executed on resource r_1) that could produce the potential direct or indirect predecessor i of k. In this context, the statement that b is potentially slower than a w.r.t. k and its predecessor i denotes that at least one production sequence with corresponding (fixed) BOM exists where k is slower than i according to the definition of "slower" for the fixed-BOMs case in the standard GLSPMS. Mathematically this can be expressed as follows:

$$\frac{g_{ik}^{min}}{\sigma_{ia}}\kappa_{r_1a}^{p} < \frac{1}{\rho_{kb}}\kappa_{r_2b}^{p} \tag{7.41}$$

Note that comparing (7.41) to (7.40), not only the $>$ sign is replaced by $<$, but (7.41) also contains g_{ik}^{min} instead of g_{ik}^{max}. g_{ik}^{min} can be interpreted as the "best-case" consumption of i for producing k, i.e., the lowest possible consumption with a feasible BOM. The left-hand side thus represents the *minimum* production time of task a on resource r_1 to output units of i required for supplying predecessor units for one unit of k. The right-hand side denotes the production time of task b on resource r_2 during which exactly one unit of k is consumed.

Regarding the schedule shown in Fig. 7.7 that may result without (7.28), the situation is as follows: Due to the relative slowness of the task b compared to a and the duration $y_{r_2,s-1}^{e}$ of the split changeover on resource r_2, one would have to start the execution of task b consuming k before the production of its (potential) predecessor i by task a has started, which is impossible if we again assume that there are no stocks of i at the beginning of $s-1$. Thus, even with (7.27) added to the model, it might not be possible to implement the resulting schedule as it might still violate temporal constraints.

Hence, (7.28) is required. It enforces that the split changeover to a task a producing predecessor i on resource r_1 and the execution time of a slower task b producing of a slower successor k on resource r_2 as well as the split changeover from b to another task on r_2 all fit into the duration $w_s - w_{s-1}$ of micro-period $s-1$.[2] A schedule with the property enforced by (7.28) is shown in Fig. 7.7(b). Note that (7.28) also considers the case where two tasks a and b are directly linked by a product i analogously to (7.27) due to the constraints (7.28) with $i = k$.

7.2.4 Transformation of GLSPMS into Special Case of GLSPMS-FPS

The GLSPMS can be transformed into a special case of the GLSPMS-FPS. The basic idea of this transformation is as follows:

- Introduce one GLSPMS-FPS product for each GLSPMS product.
- Introduce one GLSPMS-FPS resource for each GLSPMS resource with the same capacity values as the corresponding GLSPMS resource.
- Introduce one GLSPMS-FPS task for each GLSPMS product. Task k requires g_{ik} units of product i and produces one unit of product k per execution unit. The variable production costs of that task equal those of the corresponding GLSPMS product. The changeover time and sequence-dependent setup costs between two

[2] Just as (7.27), (7.28) excludes some temporally feasible solutions and can be reformulated using the variables y_{rabs}^{b} and y_{rabs}^{e} explained in footnote 1.

tasks are the same as the changeover time and sequence-dependent setup costs between the corresponding GLSPMS products.
- In addition, convert the dummy product 0 into a dummy task 0.
- The holding costs of GLSPMS-FPS products equal the holding costs of the corresponding GLSPMS products.
- Set $g_{ik}^{min} = g_{ik}^{max} = g_{ik}$.

One can see that (2.60) is a special case of (7.27) and (2.61) a special case of (7.28).

Chapter 8
Blood Bank Inventory Control with Transshipments and Substitutions

8.1 Introduction

In this section, we focus on the combination of two flexibility instruments (also see Chap. 1) that are available for the control of multi-location multi-product inventory systems:[1]

1. *Lateral stock transshipments*
2. *Product substitutions*

As mentioned in Sect. 2.3, the reasons for performing transshipments and substitutions (see Chap. 1) are often similar. They are summarized in Table 8.1.

Both transshipments and substitutions can be differentiated into two types: *preventive* and *reactive*. *Preventive* (also: *proactive, planned*) *transshipments* are performed before a stock-out actually occurs, whereas *reactive* (also: *emergency*) *transshipments* are initiated after the location has run out of stock for a product (Herer et al., 2006). The latter require that the transshipment lead time is short enough to be able to fulfill the demand. *Preventive substitution* means that we start using substitutes before a stock-out of the requested product occurs, e.g., to reserve some stocks for high-priority demand. *Reactive substitutions* are performed if the requested product is already out of stock. These aspects are summarized in Table 8.2.

8.1.1 Analogy Between Transshipments and Substitutions

A close analogy exists between T&S, which is also indicated in Axsäter (2003a): Considering an inventory system with multiple locations, transshipments and a single product, we can reinterpret stocks of the product at each location as a different "virtual product". With this reinterpretation, transshipments correspond to substitutions between the virtual products. Transshipment links can be interpreted as

[1] This section is an extended version of the working paper Lang (2008).

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_8,

Table 8.1 Benefits of transshipments and substitutions

Benefit	Substitutions	Transshipments
Increased service level	Avoid shortages due to supply or production bottlenecks of a product	Avoid shortages due to supply or production bottlenecks at a location
Reduction of fixed costs	Produce larger lot sizes of a smaller number of products	Joint replenishments for neighboring locations
Exploitation of unit cost variations	"Switch" to cheaper substitutes	Buy product where it is the cheapest and transship it if unit costs differ between locations
Reduction of holding costs (in stochastic settings)	Lower safety stocks due to "risk pooling" between products	Lower safety stocks due to "risk pooling" between locations
Reduction of wastage (in case of perishability)	Consume substitutes with shorter expiry date first	Transship and consume stocks from other location with shorter expiry date first

Table 8.2 Preventive vs. reactive transshipments and substitutions

Type	Preventive	Reactive
Transshipments	Transship from location with excess inventory if another location will run out of inventory soon	Transship to location when stock-out occurs
Substitutions	Start using substitutes before stock-out of preferred product occurs	Use substitutes when stock-out occurs

Table 8.3 Analogies between transshipment and substitution model entities

Substitutions	Transshipments
Substitution graph	Transportation network
Substitution option	Transportation link
Substitution/conversion cost	Transshipment cost
Conversion time	Transshipment lead time
Conversion capacity	Transportation capacity

substitution options between the virtual products, transshipment costs from location A to B as substitution/conversion costs of using virtual product A to substitute B. Similarly, transshipment lead times correspond to durations of conversions between virtual products and maximum capacities of transshipment links to capacitated production resources for converting virtual products. The other way around, considering an inventory system with a single location, multiple products and substitution options, each product can be interpreted as a "virtual location", and substitutions between products as transshipments between virtual locations. These analogies between transshipments and substitutions are summarized in Table 8.3.

8.1.2 Outline

The remainder of this section is structured as follows: After describing how to combine T&S in a multi-location inventory model, we highlight several cases where this combination occurs in practice. Subsequently, we focus on the inventory control of blood banks as one practical example. We describe the characteristics of blood bank systems, briefly review relevant literature and develop a discrete-event simulation model of a multi-location blood bank system with T&S. Simulation-based optimization is used to improve the parameters of a replenishment, transshipment and substitution policy for the system, which is compared to other policies in computational experiments.

8.2 Combining Transshipments and Substitutions

Instead of studying transshipments and substitutions in separate models, one can consider multi-location inventory models where both T&S are possible. In such models, four ways of fulfilling an order for a product at a certain location can be distinguished:

1. The order is fulfilled using stocks of the product from this location (no substitution, no transshipment).
2. The order is fulfilled using stocks of a substitute at this location (substitution).
3. The order is fulfilled using stocks of the product from another location (transshipment).
4. The order is fulfilled using stocks of a substitute from another location (substitution and transshipment).

These four options are illustrated with an example that contains two products and locations in Fig. 8.1. Options (2)–(4) can serve as an emergency recourse in case of a local shortage of the ordered product.

By considering each location-product pair as a separate virtual product respectively location, models with both T&S can be mapped to substitution models (and consequently to transshipment models based on their analogy).

Inventory systems with a combination of transshipments and substitutions occur in various practical settings:

- When managing the usage of *empty containers* for freight transportation, one has to ensure that enough empty containers are available at each location to transport commodities. This can be done by transshipping empty containers from other locations if there is a shortage at a location. Alternatively, compatible empty containers with differing attributes (e.g., dimensions) stocked at the location could be used as substitutes, as a certain product to be shipped would often fit into various container types (Chang et al., 2008).
- In the *e-grocery* business, customers order grocery products online and get them delivered to their homes. If a customer orders a product that is not in

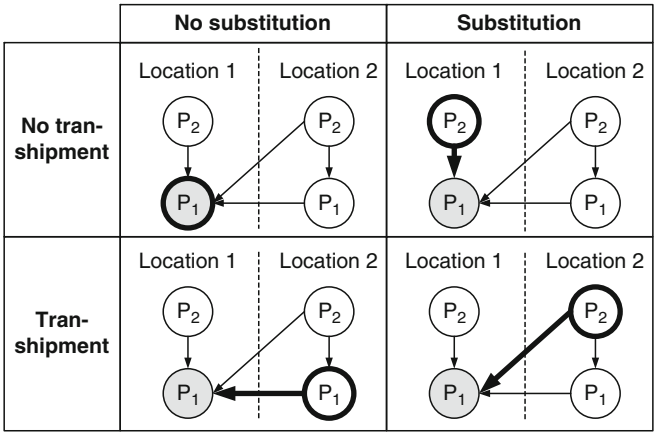


Fig. 8.1 Combining substitutions and transshipments

stock at the warehouse/supermarket from which the order will be fulfilled, e-groceries sometimes perform substitutions by shipping a similar (substitute) product to the customer. Alternatively, transshipments from other locations could be performed (Scott and Scott, 2006).

- *Car rental* companies frequently face the problem that customers want to rent a car type at a location for a specific time window where no car of that type is available at the location. In such situations, customers often get upgrades to larger and better cars without paying more, which is a form of resource substitution. Alternatively, a car of the requested type could be transshipped from a nearby location given that enough time is left (Fink and Reiners, 2006).

Another application area where both T&S are relevant is blood bank inventory control.

8.3 Blood Bank Inventory Control

In *blood bank* systems, blood is collected from eligible donors and processed into various (perishable) blood products at Regional Blood Centers (RBC). There are three main blood products: *Erythrocytes (red blood cells)*, *thrombocytes (blood platelets)* and *blood plasma*. The blood transfusions are distributed to hospitals, where they are stored in Hospital Blood Banks (HBB). Blood transfusions are needed for routine and emergency surgeries as well as other treatments. Usually, blood with the same ABO/Rhesus group is given to a patient. If there is a shortage of transfusions of that blood group, a compatible blood group can be used as a substitute (so-called "*mismatching*"): E.g., red blood cells (erythrocytes) with blood group O− can substitute all other blood groups (Katsaliaki and Brailsford, 2007). A substitution graph describing the substitution options between red blood cell transfusions

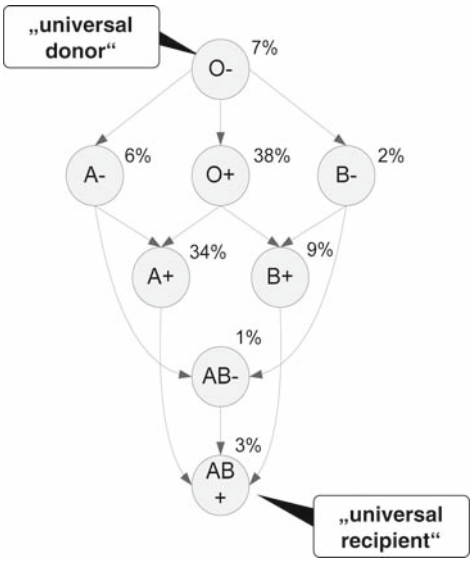


Fig. 8.2 Red blood cell product compatibility – transitive substitution graph (including blood group distribution in the USA)

of the 8 main blood groups is shown in Fig. 8.2. It also contains the percentage blood group distribution in the USA population (Edwards, 2006). In the UK, mismatching is performed in approx. 5% of all cases (BSMS, 2003). In addition, transshipments of blood from other hospitals (on the regional level) or regions (on the interregional level) can help prevent shortages (Prastacos, 1984). An aspect not considered in this section is that there might be competing suppliers from which RBCs or HBBs can purchase blood products. The typical structure of a blood supply chain consisting of several RBCs and hospitals is illustrated by Fig. 8.3.

In addition to the substitutability between blood groups, there are further characteristic aspects of blood bank inventory control:

Blood products are a typical example for perishable products (Nahmias, 1982): Red blood cells have a lifetime of 35 days, blood platelets a lifetime of only 5 days.

The supply of fresh blood is stochastic and constrained, as it depends on the number of people who donate and on the donors' blood groups. The number of potential donors can be influenced by advertisements that aim at motivating people to donate blood and by special blood donation events, but only to some extent. The eligibility to donate depends on age, health status, disease risk assessment (e.g., HIV) and time since last donation. Tests performed on donated blood may yield that the blood cannot be used.

The actual usage of blood transfusions does not coincide with the demand: Before a scheduled surgical procedure, doctors request a quantity for the patient that is usually higher than the expected consumption, as shortages of blood during surgeries might put lives at risk. Moreover, for several types of surgeries, blood transfusions are only required with a low probability.

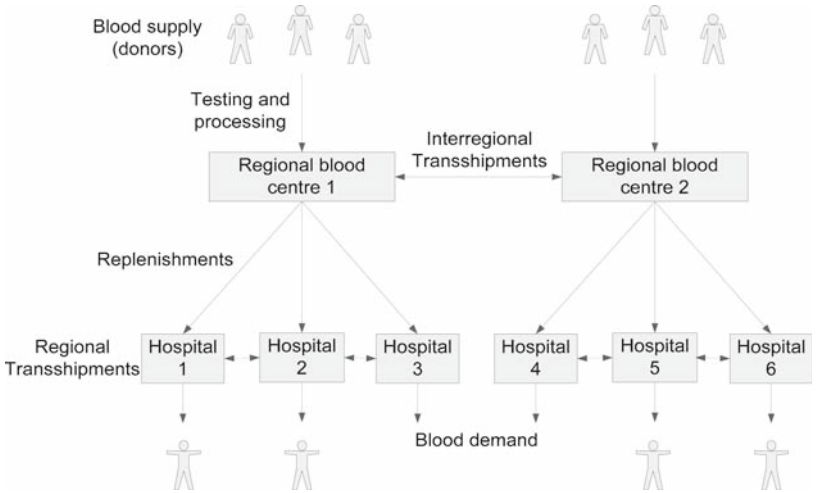


Fig. 8.3 The blood supply chain

In some cases, blood of donor and patient is incompatible due to irregular antibodies although it would be compatible according to the ABO/Rhesus system. Hence, it is common practice to perform compatibility tests of patient blood and donor blood product by mixing samples (so-called *serological crossmatching*) before using blood transfusions. Blood that has been tested for compatibility usually stays in the HBB but is assigned to a particular patient (*assigned inventory*). If the assigned blood has not been used within a certain period (so-called *crossmatch-release period*), it goes back into the (virtual) pool of *unassigned inventory*. The temporary reservation of blood stocks for individual patients impedes risk pooling and thus raises the level of required safety stocks: Safety stocks are kept individually for each patient who might need a blood transfusion, though it is unlikely that many of the patients will actually receive the entire quantity assigned to them. Because of the difference between blood demand and actual usage, the *crossmatch-to-transfusion ratio* – defined as the quantity assigned to particular patients divided by the quantity actually transfused – is significantly greater than 1.

In practice, blood bank systems are often decentralized systems with several decision makers that may have conflicting objectives. E.g., hospitals usually want a minimum number of shortages (a high service level) of blood transfusions while being cost-efficient. Hence, they tend to avoid stocking rare groups such as AB+ or B+ because these are substitutable by more frequent groups, which can lead to a scarcity of other groups, especially of O−, and often causes stocks of these rare groups to perish at RBCs. The latter in turn incurs avoidable costs for the RBCs, and might lead to shortages in other locations if blood is scarce. In addition, the actors in blood bank systems often do not sufficiently share information on inventory levels, demand forecasts, shortages, etc.

Due to recent progress in medical research, the proportion of inventory assigned to particular patients might be reduced significantly: In the so-called *Type & Screen* procedure (Georgsen and Kristensen, 1998; Chapman et al., 2000; Pereira, 2005), no serological crossmatching is performed. Instead, only the blood group of the patient's blood is determined (*Type*) and a screening for antibodies is performed (*Screen*). If the antibody screen returns positive (usually in $< 1\%$ of the cases), an identification of the antibodies is conducted. The same is done for the blood of all donors. The results are stored in an information system, so that *computer* (also: *electronic*) *crossmatching* between stocked blood transfusions and a patient's blood is possible, which removes a possible source of human errors. Computer crossmatching with blood group determination and antibody screen could be used as a standard procedure for compatibility testing. Yet, there is a residual risk of incompatibility. Hence, one option is to perform a serological crossmatching in advance in addition to blood group determination and antibody screen if the probability that a patient will require a transfusion is high (*Type, Screen & Crossmatch*).

The survey paper of Prastacos (1984) gives an overview of the decision fields in blood bank management. Pierskalla (2005) covers various aspects of blood supply chain management. Cohen and Pierskalla (1979) used simulation to model hospital blood bank inventory control and applied a simulation-based optimization approach. Substitutability between blood groups has been considered by Kopach et al. (2003) and Katsaliaki and Brailsford (2007). BSMS (2003) contains the results of a survey among UK hospitals regarding mismatching practice. Jennings (1973) described policies for initiating transshipments between hospital blood banks. Hemmelmayr et al. (2007) developed a solution approach for determining blood delivery strategies including vehicle routing decisions, based on a deterministic integer programming model. They compared a (centralized) vendor managed inventory (VMI) strategy (Campbell et al., 2002) to decentralized inventory management by hospitals.

8.4 Blood Bank Simulation Model

8.4.1 Assumptions

The assumptions of the discrete-event simulation model are as follows: We simulate a regional blood bank system with one RBC and multiple hospitals, i.e., a two-echelon system. The system is operated using a vendor managed inventory (VMI) strategy: The RBC centrally manages the blood stocks of the regional hospitals. Only red blood cells are considered, as this is the most common blood product. There are eight products: red blood cell transfusions of each of the blood groups. They are perishable and expire after 35 days. Blood transfusions are assumed to be indivisible. The blood supply at the RBC and demands at the hospitals are stochastic with stationary distributions. The RBC blood product supply is modeled as a Poisson process, the demands at the hospitals as Compound-Poisson processes with log-normally distributed quantities (Prastacos, 1984; Katsaliaki and Brailsford,

2007). The cost and demand characteristics differ between the hospitals. There are some large and several smaller hospitals (with regard to the mean blood demand). Small hospitals have a larger demand variability than larger hospitals. The demand processes of the hospitals are uncorrelated.

Hospitals can place orders for regular replenishments from the RBC at fixed points (e.g., once daily), the replenishments have a deterministic lead time that is less than the interval between orders. It is assumed that the RBC operates vehicles that drive fixed delivery routes covering all hospitals every day. Only preventive transshipments are allowed, transshipments in response to stock-outs are not possible. Transshipments can be initiated at any time and have a deterministic lead time. Blood demand is either fulfilled immediately or counted as "lost sales". Partial fulfillment of orders is allowed. In case of a stock-out of a certain blood group, it can be substituted by ABO/Rhesus-compatible blood (reactive substitution). We assume that all hospitals use the Type & Screen approach for compatibility testing and thus do not keep any assigned inventories. For simplicity, we assume that there are no capacity limits for transshipments, substitutions and hospital inventories. Initial inventories are stocked at the RBC and the hospitals at the beginning of the simulation.

8.4.2 Replenishment, Substitution and Transshipment Policies

We use the notation given in Table 8.4 to describe the replenishment, substitution and transshipment policies. Note that the set P_i of substitutes for blood group i also contains i itself. Older units of a blood group (with earlier expiry date) are consumed and delivered/transshipped first, i.e., a FIFO strategy is used for consumption and transshipments, but only if the time until expiry is more than a certain number of days. A periodic review order-up-to replenishment policy (an (r,S)-policy) is used. If supply is scarce, it is rationed according to the hospitals' order-up-to levels, taking current inventory levels and transshipments in transit into account. q_{ih} quantity units of blood group i are delivered to hospital h:

$$q_{ih} = \min \left\{ S_{ih} - I_{ih} - I_{ih}^{tr}, \frac{S_{ih} - I_{ih} - I_{ih}^{tr}}{\sum_{g \in H} \left(S_{ig} - I_{ig} - I_{ig}^{tr} \right)} I_i^{RBC} \right\} \tag{8.1}$$

$I_{ih}^{s,tr}$ is defined as:

$$I_{ih}^{s,tr} = \sum_{\iota \in P_j} I_{\iota h}^{tr} \tag{8.2}$$

We implemented the following critical-level policy for controlling substitutions: When an order for a certain quantity d of transfusions of red blood cells with group i arrives at hospital h, it is fulfilled using transfusions with group i stocked at

Table 8.4 Notations for simulation model

Symbol	Definition
Indices and sets	
$i, j \in P$	Products (red blood cells of each blood group)
$E \subseteq P \times P$	Arcs of substitution graph denoting feasible substitutions: $(i, j) \in E$ if blood group i can substitute blood group j
$G = (V, E)$	Substitution graph
D_i	Set of blood groups that can be substituted by blood group i: $D_i = \{j \mid (i, j) \in E\}$
P_i	Set of blood groups that can substitute blood group i: $P_i = \{j \mid (j, i) \in E\}$
$g, h \in H$	Hospitals
Parameters	
S_{ih}	Order-up-to level for blood group i at hospital h
CL^s_{ih}	Critical level for substitutions using blood group i at hospital h
CL^t_{ih}	Critical level for transshipments of blood group i at hospital h
L^t_{ih}	Level for initiating transshipments of blood group i to hospital h
S^t_{ih}	Transshipment order-up-to level for blood group i at hospital h
State	
d	Requested quantity of a demand event
I_i^{RBC}	Inventory of blood group i at RBC
I_{ih}	Inventory of blood group i at hospital h
I^s_{ih}	Inventory of blood group i and substitutes at hospital h
I^{tr}_{ih}	Quantity of transshipments of blood group i in transit to hospital h (not including deliveries from RBC)
$I^{s,tr}_{ih}$	Quantity of transshipments of blood group i and substitutes in transit to hospital h (not including deliveries from RBC)
q_{ih}	Quantity of blood group i delivered to hospital h in replenishment
q^t_{ih}	Intended transshipment quantity of blood group i for hospital h

hospital h. If there is a stock-out of i at h or the remaining inventory I_{ih} is less than the ordered quantity d, we look for an ABO/Rh-compatible transfusion $j \in P_i \backslash i$ (see Fig. 8.2) stocked at h for which the total inventory of substitutes I^s_{jh} is greater than a critical level CL^s_{jh}. A substitute preference order is given for each group i (see Table 8.6). I^s_{jh} is defined as:

$$I^s_{jh} = \sum_{\iota \in P_j} I_{\iota h} \tag{8.3}$$

We take at most $I^s_{jh} - CL^s_{jh}$ quantity units of j, and if the demand d is still not fulfilled completely, we look for other substitutes $j' \in P_i \backslash i$ with $I^s_{j'h} > CL^s_{j'h}$ and proceed in the same way as for j. The critical levels, e.g., serve to protect AB− from being taken as a substitute for AB+ if AB− is scarce, so that A− or B− is used. Also, the critical level of B− redirects substitute demand of B+ to the other substitute O+ which is less rare than B−. If no substitute can be found this way and a portion of the order is still unfulfilled, we try to fulfill the remaining demand using substitutes in the preference order ignoring the critical levels. The reasoning behind this is to fulfill the demand for blood using hospital stocks whenever possible.

The implemented transshipment policy works as follows: If the sum of inventory I^s_{ih} and quantity in transit $I^{s,tr}_{ih}$ of substitutes for i becomes $\leq L^t_{ih}$, we try to initiate emergency transshipments from other hospitals. Every other hospital k is only "willing" to transship a blood group i as long as the stocks I^s_{ik} of substitutes for i remain greater than a critical level CL^t_{ik}. In a first attempt, we try to transship i. If it is also scarce at the other hospitals, we try to transship substitutes. The intended transshipment quantity q^t_{ih} of i and/or substitutes is determined using a transshipment order-up-to level S^t_{ih}: $q^t_{ih} = S^t_{ih} - I^s_{ih} - I^{s,tr}_{ih}$. We transship from other hospitals in decreasing order of the maximum quantities of i and substitutes for i that they are willing to transship.

8.5 Simulation-Based Optimization Approach

In this section, we first introduce the considered simulation-based optimization problem. After this, we describe the SBO algorithm used for solving the problem in a generic form as well as its adaption to the problem.

8.5.1 Optimization Problem

The optimization problem based on our simulation model is as follows: We want to determine parameters of the T&S policy as well as the replenishment policy, i.e., the variables of the optimization problem are the parameters S_{ih}, CL^s_{ih}, CL^t_{ih}, L^t_{ih}, and S^t_{ih} for all blood groups $i \in P$ and all hospitals $h \in H$.

The objective function that we chose is to weight the number of occurred shortages and the number of transshipments that were performed, which results in a certain tradeoff between service level and transshipment costs. Note that this weighting of quality of medical service – and possibly loss of human lives – and costs reflects the ethical problems of applying OR in health care. However, this weighting is only used as a means to obtain solutions that are better than the initial solutions with regard to both objectives.

The optimization problem contains four groups of linear constraints and the nonnegativity constraints (8.8):

$$CL^s_{ih} \leq S_{ih} \qquad i \in P,\ h \in H \tag{8.4}$$

$$CL^t_{ih} \leq S_{ih} \qquad i \in P,\ h \in H \tag{8.5}$$

$$S^t_{ih} \leq S_{ih} \qquad i \in P,\ h \in H \tag{8.6}$$

$$L^t_{ih} \leq S^t_{ih} \qquad i \in P,\ h \in H \tag{8.7}$$

$$S_{ih}, CL^s_{ih}, CL^t_{ih}, L^t_{ih} \geq 0 \qquad i \in P,\ h \in H \tag{8.8}$$

Equations (8.4)–(8.6) ensure that the (critical) levels never exceed the order-up-to level S_{ih}. Equation (8.7) excludes solutions where the transshipment order-up-to level S^t_{ih} is less than the point L^t_{ih} for initiating transshipments.

8.5.2 Pattern Search Algorithm

For an overview of Simulation-Based Optimization (SBO) methods, see Fu (2002). We use an algorithm that resembles *Pattern Search (PS)* algorithms for linearly constrained continuous optimization (Lewis et al., 1998; Lewis and Torczon, 2000). Alternatively, one could apply metaheuristics like genetic algorithms (GA), but it seems questionable whether these will have acceptable running times: Objective function evaluations are very "expensive" in SBO in terms of computation time. GA require many objective function evaluations to assess each solution in a new population, whereas PS uses those sparingly. PS is a heuristic iterative local search algorithm that starts from an initial feasible solution and, in every iteration, explores a neighborhood that is defined by a certain *pattern* of search directions and a step size vector.

E.g., a pattern could contain search directions for each variable that decrease/increase it while leaving all other variables unchanged. The steps that the algorithm performs are multiplicative, i.e., it increases/decreases variables by a certain percentage. We found that when using multiplicative (relative) instead of additive (absolute) steps, it is easier to ensure that steps are in due proportion to the current variable values. This becomes especially important when changing variables simultaneously. If no feasible neighbor solution (that fulfills all variable bounds and constraints) is better than the current solution, this solution remains current solution in the next iteration. The step size vector is updated after every iteration by a procedure specified below. We use the notation in Table 8.5 to describe the pattern search algorithm.

The algorithm for a minimization problem can be described as follows:

An initial solution x_0 and step size vector Δ_0 are given. For iteration $k = 0, 1, \ldots$:

1. Compute objective value $f(x_k)$ of current solution.
2. Determine a step $s_k = \text{scale}(\Delta_k) d_r$ using a linearly constrained exploratory moves algorithm with a problem-specific pattern of search directions.
3. If a step s_k with $f(\text{perform\_step}(x_k, s_k)) < f(x_k)$ to a feasible neighbor has been found, perform this step: $x_{k+1} = \text{perform\_step}(x_k, s_k)$. Otherwise set $x_{k+1} = x_k$. The function perform_step(x_k, s_k) returns a new solution x_{k+1} with

$$x_{k+1,v} = \begin{cases} x_{kv} \cdot s_{kv} & \text{for } s_{kv} > 0 \\ \frac{x_{kv}}{|s_{kv}|} & \text{for } s_{kv} < 0 \\ x_{kv} & \text{for } s_{kv} = 0 \end{cases} \tag{8.9}$$

Table 8.5 Notations for PS algorithm

Symbol	Definition
Indices and sets	
$v = 1, \ldots, p$	Variables
k	Iteration counter
$r = 1, \ldots, n_d$	Search directions
H_s, H_l	Set of small resp. large hospitals
State	
Δ_k	Step size in iteration k (vector)
Δ_{kv}	Step size of variable v in iteration k
Variables	
x_k	Solution vector in iteration k
x_{kv}	Variable v in iteration k
Other	
S^0_{ih}	Order-up-to level for group i at hospital h in initial solution
x_0	Initial solution
d_r	rth direction under consideration for exploratory move
s_k	Step in iteration k (vector)
$f(x_k)$	Estimate of objective function value by simulation run for solution x_k
α	Initial step size parameter
β, γ	Parameter for updating step size after successful resp. unsuccessful iterations
δ, ϵ	Minimum/maximum step size
ζ,η	Parameters for upper variable bounds

4. Determine new step size vector Δ_{k+1}.
5. Stop if termination criterion is met (time limit, maximum number of iterations, or local optimum).

The function $scale(\Delta_k)$, if necessary, scales the current step size vector by a factor that is chosen large enough so that for each variable v changed in this iteration (i.e., with $d_{rv} \neq 0$), the new rounded value $\lfloor x_{k+1,v} + 0.5 \rfloor$ differs from the old rounded value $\lfloor x_{kv} + 0.5 \rfloor$. It is necessary because the simulation model rounds variables internally to integers because of the indivisibility assumption for blood transfusions. Without scaling of steps, the algorithm would often perform small steps that would result in only few or no effective changes in the solution.

All entries in the initial step size vector Δ_0 are set to α. The step size vector is updated as follows: If the iteration was successful, subtract β from each entry of the step size vector that belongs to a variable changed in this iteration, i.e., if $f(x_k + s_k) < f(x_k)$, set $\Delta_{k+1,v} = \Delta_{kv} - \beta$ for all v with $s_{kv} \neq 0$. Otherwise, set $\Delta_{k+1,v} = \Delta_{kv} + \gamma$. Ensure that step sizes do not get too small or large by applying $\Delta_{k+1,v} = \max\{\delta, \min\{\Delta_{k+1,v}, \epsilon\}\}$. The idea of this procedure is to decrease the step size after successful moves to intensify the search in promising regions and to increase it after unsuccessful moves to escape local optima. The progressive decrease of the step size resembles the cooling process in simulated annealing algorithms.

8.5.3 Adaption of PS Algorithm

Referring to the blood bank SBO problem, a solution vector x_k contains the following $p = 5 \cdot |P| \cdot |H|$ variables: S_{ih}, CL^s_{ih}, CL^t_{ih}, L^t_{ih}, and S^t_{ih} for all blood groups $i \in P$ and all hospitals $h \in H$. In addition to the constraints (8.4)–(8.8), we add upper variable bounds to restrict the search space: For each order-up-to level S_{ih} respectively (critical) level CL^s_{ih}, CL^t_{ih}, L^t_{ih}, and S^t_{ih}, the upper bound is set to a multiple ζ respectively $\eta > 1$ of the order-up-to level S^0_{ih} of blood group i at hospital h in the initial solution:

$$S_{ih} \leq \zeta S^0_{ih} \qquad i \in P,\ h \in H \tag{8.10}$$

$$CL^s_{ih}, CL^t_{ih}, L^t_{ih}, S^t_{ih} \leq \eta S^0_{ih} \qquad i \in P,\ h \in H \tag{8.11}$$

Neighbor solutions that violate (8.10) or (8.11) are repaired by setting variables that cause infeasibility to their respective upper bounds. In addition to patterns of search directions that only manipulate variables individually while leaving the other variables unchanged, we also examine patterns with search directions that simultaneously vary multiple variables: We group the order-up-to and critical level variables of a blood group by the size of hospitals. For this purpose, we partition the hospitals into a set of small hospitals H_s and a set of large hospitals H_l according to their average blood demand. This distinction is made because we assume that the traits of good/optimal blood group order-up-to and critical levels differ depending on the average demand and demand variability of a hospital (see Pereira, 2005), while good/optimal levels are similar for hospitals that have almost the same size. The idea is to speed up the algorithm by simultaneously changing the order-up-to or critical levels of a blood group i for all hospitals with the same size in a single iteration, rather than only manipulating single variables. This is important for algorithm performance, as the evaluation of a single solution on one scenario takes several seconds.

We used patterns of search directions for the blood bank SBO problem that contain the following sets of search directions:

(a) Increase or decrease the order-up-to level S_{ih} for a single blood group i and hospital h, while leaving all other variables unchanged
(b) Increase or decrease either the (critical) level CL^s_{ih}, CL^t_{ih}, L^t_{ih}, or S^t_{ih} for a single blood group i and hospital h, while leaving all other variables unchanged
(c) Simultaneously increase or decrease the order-up-to levels S_{ih} for a single blood group i and all hospitals with a certain hospital size (either all $h \in H_s$ or $h \in H_l$), while leaving all other variables unchanged
(d) Simultaneously increase or decrease either the (critical) levels CL^s_{ih}, CL^t_{ih}, L^t_{ih}, or S^t_{ih} for a single blood group i and all hospitals with a certain hospital size (either all $h \in H_s$ or $h \in H_l$), while leaving all other variables unchanged

The corresponding vectors d_r for the search directions contain a value of $+1$ for variables that are increased, -1 for variables that are decreased and 0 for variables

that remain unchanged. The linearly constrained exploratory moves algorithm in step (2) of the PS algorithm examines the (feasible) neighbors in a generated order and performs a step to the first neighbor found with a better objective (first improvement). In every iteration, the order is generated randomly using weights for the above sets of search directions: First, the order of search directions in each of the sets is shuffled randomly. Then, the search directions of the sets are combined by iteratively taking the next search direction from a set that is selected randomly using a discrete probability distribution based on the weights. Thus, the higher the weight for a set of search directions, the earlier they are considered when examining the neighborhood. The procedure terminates when all search directions of the individual (shuffled) lists have been added to the combined list.

8.6 Computational Experiments

We first briefly describe the research questions that we address with our experiments. After this, we describe setup, design, and results of the experiments used to examine these questions:

1. What is the impact of performing T&S on the service level of the blood bank system? Does the combination improve system performance?
2. Which parameters of the replenishment, substitution and transshipment policies influence system performance the most? Where should the effort in the PS algorithm be put?
3. Is it more efficient to change variables simultaneously in the PS algorithm?
4. Should policy parameters be chosen differently for small and large hospitals?

8.6.1 Setup

The simulation model, SBO algorithm and experiments were implemented in Java™ using the simulation library DESMO-J (Page and Kreutzer, 2005).

8.6.1.1 Simulation Model Parameters

In the following, we describe the base case settings for the blood bank simulation model. We tried to choose these as realistic as possible, based on interviews with an RBC logistics/sales manager. We consider a system with seven hospitals, two of them large, the others small. Each run simulates the system for 365 days of simulation time. We assume that the blood groups of supply and demand are distributed as given in Fig. 8.2, i.e., for each supply/demand event, the blood group is drawn from a discrete distribution with these probabilities.

Table 8.6 Blood types preference order

Donor / Recipient	O−	O+	A−	A+	B−	B+	AB−	AB+
O−	1	2	2	4	2	4	4	8
O+		1		3		3		7
A−			1	2			3	6
A+				1				5
B−					1	2	2	4
B+						1		3
AB−							1	2
AB+								1

The average daily demand at a location is 100 and 110 for the two large and 20, 18, 25, 22, and 21 for the five small hospitals. The demand at each location is modeled as a Compound-Poisson process. The quantity of an individual demand event is generated using a log-normal distribution with $\mu = 2.0$ and $\sigma = 1.0$ by rounding it up to the next integer value. This results in a mean demand of about 2.5 units per demand event. The mean inter-arrival time of the Compound-Poisson process is set in a way that the mean daily demand equals the desired quantity.

The supply of red blood cell transfusions at the RBC is modelled as a Poisson process, i.e., new blood transfusions arrive one by one. The average supply is assumed to be 101% of the total average demand of all hospitals, i.e., we consider a situation where slightly more blood than consumed is available.

When substitutions are necessary, substitutes are selected in the preference order given in Table 8.6: The decision maker tries to take substitutes in ascending order of the preference values, i.e., substitutes with lower preference number are taken first.

We assume that the RBC has initial inventories that cover the total demand of approximately 3 days, while each hospital has initial inventories that cover the hospital's demand for about 5 days. The age of each transfusion (in days) is drawn from a $U(1, 34)$ uniform distribution. The blood type proportions in the initial inventories follow the assumed blood type distribution.

The interval between the fixed points for replenishment orders is assumed to be 1 day, the replenishment lead time from the RBC to each of the hospitals 20 h, and the transshipment lead time between each pair of hospitals 3 h. These assumptions would correspond to a setting in a rural area where the next RBC is far away from the hospitals, and the distances among a group of hospitals are smaller. No units that expire in less than 5 and 3 days respectively are delivered from the RBC and transhipped between hospitals.

In the objective function, the number of shortages is weighted with factor 0.95, the number of transshipments with 0.05. We also tried other weightings in experiments not included here, and found that with this tradeoff, the PS algorithm often returned solutions with a significantly better service level that required less or only a moderate number of additional transshipments.

8.6.1.2 PS Algorithm Parameters

The PS algorithm parameters are chosen as follows: $\alpha = 1.2$, $\beta = 0.01$, $\gamma = 0.1$, $\delta = 1.05$, $\epsilon = 2.0$, $\zeta = 4$, and $\eta = 2$. The algorithm terminates when a time limit of 16 h is reached or no improvement has been found in the last 10 iterations.

The following initial solution is used for the PS algorithm: The order-up-to levels S^0_{ih} are set to 5 days of supply, i.e., to cover the average direct demand (without potential additional demand caused by substitutions) for blood group i at hospital h of 5 days. We chose these initial order-up-to levels as such simplistic approaches seem to be in use in practice, in order to see how much improved solutions differ from the status quo. The levels CL^s_{ih} equal 2 days of supply for i at h, CL^t_{ih} 3 days of supply. The levels L^t_{ih} for initiating transshipments correspond to a half day of supply, the transshipment order-up-to levels S^t_{ih} to 1 day of supply. All variables are rounded up to obtain integer values.

During the PS algorithm, the objective value of each solution is computed by averaging over 20 fixed scenarios, where each scenario is represented by specific random seeds for supply and demand distributions. In each simulation run, all variable values of the solution are rounded internally. In order to measure the quality of returned solutions objectively, solution metrics are computed by averaging over 20 scenarios that differ from those used in the PS algorithm.

8.6.2 Experiment Designs, Results and Interpretation

8.6.2.1 Impact of Transshipments and Substitutions

In our first experiment, we analyze the impact of performing T&S (research question 1) by comparing the following policies: (1) neither substitutions nor substitutions, (2) substitutions, (3) transshipments, (4) both transshipments and substitutions are allowed. Table 8.7 contains various metrics that describe both the initial solutions and the solutions returned by the PS algorithm. A consistent numbering of algorithm and problem configurations is used to identify configurations that occur in more than one experiment. The algorithm was run with the sets of search directions (c) and (d) (simultaneous changes of order-up-to/critical levels) with weight 0.5 for each of the two sets. The search directions manipulating CL^t_{ih}, L^t_{ih}, or S^t_{ih} variables were excluded from set (d) for policies (1) and (2) as they are not required for these policies.

The results show for initial and SBO solutions that with respect to the number of shortage events, the combined T&S policy performs the best, the substitutions-only policy slightly worse, and the transshipments-only policy clearly inferior but still better than the policy without T&S. The SBO solutions have a significantly higher service level than the solutions before optimization. Comparing the transshipments-only with the T&S policy, we see that the latter performs much less transshipments.

Table 8.7 Computational results – experiment 1 (simultaneous search pattern, order-up-to and critical levels)

Configuration	1	2	3	4	5	6	7	8
Initial sol. / SBO	In.	In.	In.	In.	SBO	SBO	SBO	SBO
Substitutions	No	Yes	No	Yes	No	Yes	No	Yes
Transshipments	No	No	Yes	Yes	No	No	Yes	Yes
# Shortage events	727.35	223.50	625.00	206.60	225.30	86.60	199.65	85.20
Shortage quantity	1,394.55	439.60	1,050.00	393.10	454.60	185.05	364.50	170.60
# Transshipments	0	0	1,723.95	657.65	0	0	630.10	225.75
Quantity transshipped	0	0	2,535.80	1,386.45	0	0	1,816.30	844.75
β Service level (%)	98.79	99.62	99.09	99.66	99.61	99.84	99.68	99.85
Substitution ratio (%)	0	1.09	0	1.11	0	0.46	0	0.47
# Iterations	–	–	–	–	96	90	147	90
Running time (h)	–	–	–	–	16	16	16	16

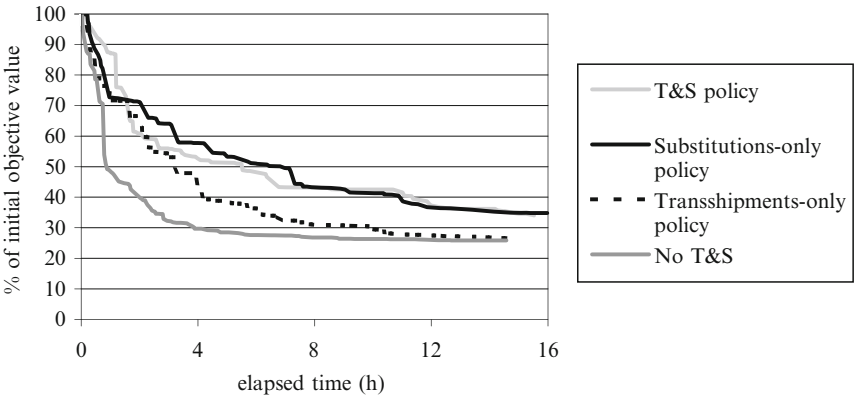


Fig. 8.4 SBO algorithm progress

The substitution ratio decreases from approx. 1% in the initial to 0.5% in the SBO solutions for substitutions-only and T&S policy.

When examining the simulation results, we found that with substitutions-only and T&S policy, demand for O− made up the biggest share (about 90%) of the total shortage quantity averaging over the scenarios. It was followed by A− and B− with shares between 3% and 6% and the other blood groups near 0%. When using the transshipments-only policy and policy without T&S, the shares were distributed differently: Almost no shortages occurred for the frequent groups O+ and A+ (due to low demand variance), whereas the share was between 10% and 25% for each of the other blood groups.

Figure 8.4 depicts the SBO algorithm progress over time for configurations 5–8. The elapsed time (in hours) is displayed on the x-axis, the objective value of the best solution found so far in percentage of the initial solution on the y-axis. Though a local optimum still has not been reached after 16 h, it seems that the algorithm progress has almost stagnated at the time limit for each the four policies.

8.6.2.2 Pattern of Search Directions

In a second experiment, we compare the solutions obtained using different sets of search directions (research questions 2 and 3) in the PS algorithm: (1) sets (c) and (d) (simultaneous changes of order-up-to / critical levels) with weight 0.5 for each of the two sets, (2) sets (a) and (b) (individual changes of order-up-to / critical levels) with weight 0.5 for each of the two sets, (3) set (c) (simultaneous changes of order-up-to levels), and (4) set (d) (simultaneous changes of (critical) levels). We optimize a policy that performs both T&S. Table 8.8 contains metrics describing the results of this experiment.

When changing variables individually instead of simultaneously (config. 9), the algorithm performed a larger number of iterations within the time limit, but the

Table 8.8 Computational results – experiment 2 (T&S)

Configuration	4	8	9	10	11
Initial sol. / SBO	In.	SBO	SBO	SBO	SBO
Order-up-to levels simult.	–	Yes	No	Yes	No
(Critical) levels simult.	–	Yes	No	No	Yes
Order-up-to levels indiv.	–	No	Yes	No	No
(Critical) levels indiv.	–	No	Yes	No	No
# Shortage events	206.60	85.20	128.20	81.85	181.00
Shortage quantity	393.10	170.60	244.25	160.10	373.15
# Transshipments	657.65	225.75	401.10	287.05	564.75
Quantity transshipped	1,386.45	844.75	914.25	786.40	2,268.85
β Service level (%)	99.66	99.85	99.79	99.86	99.68
Substitution ratio (%)	1.11	0.47	0.69	0.45	1.10
# Iterations	–	90	114	112	59
Running time (h)	–	16	16	16	16

returned solution is worse. Manipulating only the critical levels does not seem to have a high impact on the service level (config. 11). Interestingly, the PS algorithm yielded the best results with search directions that simultaneously change only the order-up-to levels (set (c), config. 10). In the returned solution, the number of shortages was lower than in the solutions returned by the algorithm using the other three sets of search directions. This result can be explained as follows: When changing both order-up-to and critical levels (set (1), config. 8), the algorithm spends a significant amount of time on examining changes of critical levels. These, as indicated by config. 11, have a lower impact on the service level than the order-up-to levels.

8.6.2.3 Difference Between Small and Large Hospitals

To address research question 4 – should policy parameters be chosen differently for small and large hospitals? – we analyze the solution for a policy that performs T&S. It is obtained from the PS algorithm in experiment 1 (configuration 8). The solution is summarized in Table 8.9, which contains the percentage change of variables in the SBO solution compared to the initial solution for small and large hospitals. It is computed after averaging the variables expressed in days of supply.

There is a clear difference between small and large hospitals in the SBO solution. The order-up-to levels for all blood groups are increased for small hospitals (the 100% changes result from the integrality of initial variable values and steps with integral changes). The order-up-to levels of large hospitals for O−, O+ and B+ are reduced slightly, presumably to release RBC stocks of these blood groups for small hospitals through the rationing mechanism in (8.1). In addition, the order-up-to levels are increased for the comparatively rare blood groups B−, AB− and AB+ in large hospitals. The critical level for substitutions using O− at small hospitals is increased, presumably to divert demand to other substitutes, e.g., to O+

Table 8.9 Computational results – SBO solution for configuration 8 (% change of average value of days of supply compared to initial solution)

Variable	Hosp. size	O− (%)	O+ (%)	A− (%)	A+ (%)	B− (%)	B+ (%)	AB− (%)	AB+ (%)
S_{ih}	Large	−4.9	−13	0	0	20	−5.5	76.2	28.80
S_{ih}	Small	74.7	15.5	56	14.3	100	66.5	100	85.5
CL^s_{ih}	Large	0	0	17.7	0	0	0	0	0
CL^s_{ih}	Small	33.3	0	0	0	100	0	0	0
CL^t_{ih}	Large	0	0	−11.5	0	13	0	0	0
CL^t_{ih}	Small	25	17	0	0	0	0	0	0
L^t_{ih}	Large	−25	−11.8	33.3	0	0	0	0	0
L^t_{ih}	Small	−100	−25	−100	0	0	0	0	0
S^t_{ih}	Large	0	0	22.5	0	0	0	0	0
S^t_{ih}	Small	50	0	100	0	0	0	0	0

as a substitute for B+ in cases where both O− and O+ are scarce. The critical levels for transshipments of O− and O+ are increased for small hospitals. The reason could be that transshipping them when the inventory level at the transshipping location is already low causes stock-outs at the small hospitals. Due to a 100% decrease of L^t_{ih}, transshipments of O− and A− to small hospitals are only initiated if the inventory is 0. It seems that transshipping these rare blood types is often too risky for the (potential) transshipping location. The transshipment order-up-to levels for O− and A− are increased for small hospitals, so that the transshipment quantities get larger. Note that the SBO solution should be interpreted with care: First, some of the solution variables might be "over-fitted" to the considered scenarios. Second, a heuristic was used and much better solutions with differing characteristics might exist. Third, the interpretation of the S_{ih} values is difficult because, due to the rationing performed in (8.1), rather the proportions between the S_{ih} values of the various hospitals than the absolute values are important.

8.7 Conclusions

We highlighted several possible applications for T&S models and then focussed on the blood bank model. The SBO algorithm returns solutions that are clearly better than the initial solutions. We found in our experiments that the combination of T&S improves the service level compared to performing only substitutions or only transshipments. The difference between a substitutions-only and a T&S policy is rather small, presumably the latter policy would perform considerably better when including emergency transshipments. Changing variables simultaneously instead of individually seems to speeds up the SBO algorithm. The choice of order-up-to levels has a higher impact on solution quality than the choice of critical levels. Policy parameters, amongst others the relative proportions of blood group order-up-to-levels, should be chosen differently for small and large hospitals.

8.8 Limitations

Model, algorithm and computational experiments have several limitations: Our model does not include demand forecasts, which could be used in practice, given that RBC / hospital software solutions can handle blood "reservations" for scheduled surgeries. Also, it does not allow fulfilling demand with emergency transshipments and/or emergency deliveries from the RBC, which would further improve the service level.

The considered decision rules for replenishments, substitutions and transshipments are rather simplistic, and could be replaced by more intelligent decision rules. Also, the used initial solution could be replaced by a better one that is derived from mean and standard deviation of demand. The assumed substitute preference order

is one out of many possible preference orders, and other preference orders might be better both from an economic and medical point of view. In addition, the current policy employing critical levels can lead to suboptimal control of the system: If the inventory of A− has fallen below the critical level for substitutions using A−, it might, e.g., happen that O− is used to substitute for A+ although A− is still in stock at the hospital. This would be unwanted as O− is the most precious blood group because it is the "universal donor", and one should thus always consume other substitutes first before falling back on O−.

In addition, the multi-objective setting could be handled using a more sophisticated approach than simply weighting the objectives, as this weighting is critical from an ethical perspective.

The computational results strongly depend on the simulation model parameters. In addition, it cannot be said with 100% confidence whether one strategy is better than another (e.g., substitutions-only vs. T&S), as the SBO algorithm does not return optimal solutions. The PS algorithm with its specific settings might not be the best approach, it could be compared to other algorithms. Also, the computational effort of the algorithm is rather high, as typical for SBO.

Chapter 9
Conclusions and Future Research

9.1 Conclusions

This section briefly recapitulates the foremost motives of this work and summarizes its main contributions and findings. It complements the conclusions already drawn in Sects. 5.6 and 8.7.

Only a limited number of publications on lot-sizing and scheduling models with substitutions / flexible BOMs exist. The terminologies pertaining to substitutions used in the various publications differed significantly. In addition, there was no classification scheme available for substitution models.

Therefore, we first developed a general classification scheme for production planning models in Chap. 2 by aggregating several existing classifications and inserting criteria for substitutions, flexible BOMs and flexible production sequences. Also, we exemplified and differentiated the terms "flexible BOMs" and "flexible production sequences". We developed a precise terminology for substitutions in Sect. 1.1. In order to illustrate the practical relevance of the topic, we described a number of practical applications in Sect. 3.1. After that, we presented four approaches for modeling substitutions / flexible BOMs: Blending models, substitution graphs, substitution hypergraphs, and task-oriented modeling. In Sect. 3.3, we developed classification criteria for substitution models supplementing those in Chap. 2. Also, we describe crucial points to be considered when implementing substitutions in practice (see Sect. 3.4).

The developed categorization framework was used in Chap. 4 to classify the existing models. In this chapter, we also described how various other aspects in SCM, logistics, production, operations management and economics are related to substitutions and flexible bills-of-materials.

The models in the existing publications on lot-sizing with substitution did not include several aspects relevant in real-world problems, amongst others initial inventories and sequence-dependent setups, and hardly any literature on capacitated and multi-level lot-sizing with substitutions was available. Hence, we decided to develop production planning models with substitutions that include these aspects: In Chap. 5, we developed an uncapacitated as well as a capacitated single-level lot-sizing model with substitutions, both of which incorporate initial inventories. We found that in

J. C. Lang, *Production and Inventory Management with Substitutions*,
Lecture Notes in Economics and Mathematical Systems 636,
DOI 10.1007/978-3-642-04247-8_9,

these models, initial inventories cannot neglected without loss of generality by netting demands, as the net demands depend on the substitution quantities chosen in a certain solution (see Sect. 5.1).

Motivated from a real-world application in windshield interlayer production (see Sect. 3.1), we developed a single-level capacitated lot-sizing model with sequence-dependent setup times and costs in Chap. 6.

Chapter 7 introduced two multi-level lot-sizing and scheduling models with flexible BOMs: The MLCLSP-S integrates the substitution hypergraphs modeling approach into the MLCLSP. The GLSPMS-FPS is a hybrid of the GLSPMS and MLFP models. Thus, it has the same advantages that the GLSP has over the CLSD – it can model exogenous resource downtime and sequence-dependent setups that violate the triangle inequality. Also, just as the MLFP, it can map flexible BOMs, flexible production sequences and by-products.

An important observation was that echelon stocks cannot be specified exactly if BOMs are flexible because the Gozinto factors become ambiguous (see Sect. 7.1), as they depend on which substitute is actually used. Hence, standard cuts for multi-level capacitated lot-sizing that use the concept of echelon stocks cannot be applied in the usual way.

In addition, we showed several relations between various lot-sizing and scheduling models with substitutions and flexible BOMs: The RPS and SWCP were transformed into special cases of the MLFP (see Sect. 4.2.3.3). Also, we described how the MLCLSP-S can be reduced to a special case of an extension of the MLFP by setup times (see Sect. 7.1.4). Moreover, we delineated a transformation of the GLSPMS into a special case of the GLSPMS-FPS in Sect. 7.2.4.

In stochastic inventory control, substitutions had – to the best of our knowledge – not been considered together with transshipment, though this case occurs in several applications, e.g., blood bank inventory management.

Therefore, we decided to develop a simulation model for a blood bank system with substitutions and transshipments, serving as an illustrative example of a multi-location transshipment and substitution problem (see Chap. 8). In Sect. 2.3, we gave an introduction to transshipment problems and developed a comprehensive model categorization to serve as a basis for describing this model.

Owing to the gap regarding production planning and inventory control models with substitutions and flexible BOMs, there was also a lack of solution approaches, of algorithms for such models.

Hence, we developed reformulations and valid inequalities for the LSP-SI and MR-CLSP-S in Chap. 5 to make the models solvable with standard MIP solvers in an acceptable time, and also tested the newer approach of approximate extended formulations in this context. We performed computational experiments on problem instances generated by a self-developed instance generator. These experiments showed that the reformulations are superior to the original formulations and those with valid inequalities added a priori, except for instances with multiple resources and downward substitution. Using approximate extended formulations, the MIP solver running times were almost as good as with complete Simple Plant Location-based reformulations.

As instances of the capacitated lot-sizing model with sequence-dependent setups developed in Chap. 6 were too hard to be solved immediately by MIP solvers, we decided to develop heuristics for this model. We chose to devise MIP-based heuristics by adapting the principles of Relax&Fix and Fix&Optimize, as the literature on those indicates that they yield good results for other lot-sizing models regarding solution time and quality. Computational experiments were performed on generated instances whose structure follows that of a practical application in windshield interlayer production planning, and which have rather tight production capacities. The instances were of medium size, with 20 products and 10 periods. We found that certain F&O variants performed best, and yielded feasible solutions without overtime and lost sales in less than 100 s for 70–80% of the instances. Yet, averaging over all instances – including those for which no solution without overtime and lost sales could be found – the gap to the best known solution was still around 35%.

Regarding the blood bank inventory control application, we decided to use discrete-event simulation to model the system (see Chap. 8). This enabled us to build a rather realistic, accurate model, compared to an analytical modeling approach that would have required a multitude of simplifying, unrealistic assumptions. We designed a critical-level-type policy for operating the inventory system. The parameters of this policy were improved using a simulation-based optimization algorithm (see Sect. 2.4.2). We assumed two objectives, the number of shortages and the number of transshipments, and, for the sake of simplicity, weighted those. An adapted pattern search algorithm was devised that automatically improves the policy parameters. We made sure to design it in a generic way so that it could be applied to other, similar SBO problems.

9.2 Future Research

In the following, we point out various opportunities for future research in the field of production planning and inventory control with substitutions and flexible BOMs.

9.2.1 Multi-Location Inventory Control with Transshipments and Substitutions

We see various opportunities for extending the blood bank simulation model of Chap. 8:

- It could be made more realistic by including knowledge about demand and supply forecasts that are based on "reservations" for surgeries and donor appointments.
- A more intelligent policy for substitutions and transshipments could be developed that remedies some of the drawbacks of the critical level policy.

- The initial solution could be generated by adapting formulas from stochastic inventory control.
- One could include emergency deliveries from the RBC.
- Emergency transshipments could be integrated into the model in addition to preventive transshipments.

On the algorithmic side, the PS algorithm could be enhanced by more intelligent neighborhood examination methods and step size updating procedures or additional strategies to escape local optima, or meta-heuristics like genetic algorithms could be used.

Regarding the considered application in blood bank inventory management, additional research questions could be examined using the model, e.g., the influence of the substitute preference order and supply/demand ratio on the system's performance. Also, the multi-objective problem with the tradeoff between shortages and transshipments and emergency deliveries could be examined by approximating the efficient frontier.

In addition, an empirical study with detailed historical blood supply, reservations and demand data of the RBC and hospitals in a region could help examine the possible benefits of combining T&S in a practical setting.

The solution approach for the multi-location blood bank model could be adapted to other cases where both transshipments and substitutions are relevant, to examine the benefits of considering them in a combined model. For example, the usage of substitutions by online groceries (see section 3.1) could be investigated in detail.

9.2.2 Production Planning with Substitution and Flexible BOMs

Regarding possible extensions of the models considered in this work, the following research directions could be of interest, provided that there are corresponding real-world applications:

- Extend the MR-CLSP-S or CLSD-S by fixed conversion costs and/or times, and constrained resources for performing substitution activities. However, note that this case can be seen as a special case of the GLSPMS-FPS.
- Develop multi-location production planning models that include both substitutions and transshipments.
- Develop production planning models with substitutions and joint/family setups.

With regard to the CLSD-S and the MIP heuristics developed for it, several research avenues could be explored, such as, for instance:

- Extend the model and heuristics by including multiple resources and/or fixed conversion costs/times.
- Formulate a profit maximization objective instead of a cost minimization objective.
- Add individual service level constraints for demand classes.

- Develop a CLSD-S reformulation with sequence variables analogously to the CLSD reformulation devised by Haase and Kimms (2000), and adapt the MIP heuristics for this reformulation.
- Conduct additional computational experiments with other instance generator settings to analyze the behavior and sensitivity of the MIP heuristics in terms of running times and solution quality (e.g., a different number of products and periods, different capacity availability values, or other relaxations, for instance only overtime, only lost sales, or backlogging).

Concerning solution methods for the models MLCLSP-S and GLSPMS-FPS, the following approaches seem promising:

- Develop specialized valid inequalities and appropriate separation algorithms.
- Investigate SPL, shortest-path, or other reformulations of the models, e.g., hybrid I&L and SPL formulations.
- As reformulation-based solution approaches will presumably have prohibitively high running times on MLCLSP-S and GLSPMS-FPS instances of realistic size, there will be a need for heuristics to solve them. To this end, the MIP-based heuristics developed for the CLSD-S could be generalized.

Case studies with real-world data could be performed to examine the benefit of substitutions in practical settings. The availability of data on substitution options / BOMs flexibility in Advanced Planning Systems (APS) would of course simplify such case studies.

9.2.3 Substitutions and Flexible BOMs in Advanced Planning Software

Another avenue for future research is to analyze the functionality that the latest versions of Advanced Planning software products – e.g., SAP® Advanced Planner and Optimizer (APO) and the Oracle® Advanced Planning Suite – offer regarding substitutions and flexible BOMs. Also, one could examine the modeling frameworks and solution approaches they use, which, of course, requires that detailed documentation is available. This, together with an empirical analysis of the requirements of industry clients, might hint to useful adaptations and extensions of the models and algorithms considered in this thesis.

Appendix A
Additional Related Literature

Table A.1 Literature on aspects related to product substitution

Aspect	
Stochastic inventory control with substitution	
Bassok et al. (1999)	Bitran and Dasu (1992)
Cai et al. (2004)	Chand et al. (1994)
Chen and Chen (2004)	Deniz et al. (2005)
Duenyas and Tsai (2000)	Gallego et al. (2006)
Hale et al. (2000)	Hsieh and Wu (2009)
Hsu and Bassok (1999)	Ignall and Veinott, Jr. (1969)
Liu and Lee (2007)	Pasternack and Drezner (1991)
Silver and Moon (2001a)	Silver and Moon (2001b)
Tang and Yin (2007)	Yadavalli et al. (2006)
Yao and Zheng (1999)	Yao and Zheng (2003)
Hsieh and Wu (2009)	
Inventory control with random yields	
Birge et al. (1998)	Bitran and Dasu (1992)
Duenyas and Tsai (2000)	Gerchak and Grosfeld-Nir (1999)
Hsu and Bassok (1999)	Liu and Lee (2007)
Ng and Lam (1998)	Rao et al. (2004)
Yadavalli et al. (2006)	
Inventory control for perishable products	
Karaesmen et al. (2008)	Lystad et al. (2006)
Nahmias (1982)	
Flexible bills-of-materials / material compatibility	
Ball et al. (2003)	Hohenegger et al. (2007)
Pels (2006)	Ram et al. (2006)
Ramdas (2003)	Woss (1997)

(*Continued*)

Table A.1 (Continued)

Aspect	
Customer-driven product substitution	
Agrawal and Smith (2003)	Bitran et al. (2005)
Chen and Plambeck (2008)	Chong et al. (2004)
Ernst and Kouvelis (1999)	Gaur and Honhon (2006)
Honhon et al. (2006)	Kok and Fisher (2007)
Li et al. (2006)	Mahajan and van Ryzin (2001)
Netessine and Rudi (2003)	Rajaram and Tang (2001)
Shah and Avittathur (2007)	Shah and Avittathur (2007)
Smith and Agrawal (2000)	Yang and Schrage (2009)
Zhang and Chen (2004)	Ganesh et al. (2008)
Component commonality	
Boysen and Scholl (2008)	Fixson (2007)
Jans et al. (2008)	Ma et al. (2000)
Silver and Minner (2005)	Swaminathan and Tayur (1998)
Swaminathan and Tayur (1999)	Thonemann and Brandeau (2000)
van Hoek (2001)	
Postponement	
Chen and Chen (2004)	Kerkkänen (2007)
Ma et al. (2000)	Silver and Minner (2005)
Swaminathan and Tayur (1998)	Swaminathan and Tayur (1999)
van Hoek (2001)	
Inventory rationing and multiple demand classes	
Arslan et al. (2007)	Benjaafar and ElHafsi (2006)
Duran (2007)	Kleijn and Dekker (1998)
Kocaga and Sen (2007)	Kranenburg and van Houtum (2007)
Moon and Kang (1998)	Teunter and Haneveld (2008)
Revenue management of substitutable/flexible products	
Birge et al. (1998)	Faber (2005)
Gallego et al. (2004)	Gallego and Phillips (2004)
Karaesmen and van Ryzin (2004)	Shumsky and Zhang (2007)
Assembly-to-order / build-to-order	
DeCroix and Zipkin (2005)	DeCroix et al. (2005)
Gunasekaran and Ngai (2005)	Lu and Song (2005)
Thomas and Warsing (2007)	
Remanufacturing and disassembly	
Bayindir et al. (2005)	Bayindir et al. (2007)
Inderfurth (2004)	Li et al. (2006)
Li et al. (2007)	Schultmann et al. (2002)
Supplier selection and multiple sourcing	
Aissaoui et al. (2007)	Basnet and Leung (2005)
Benjaafar et al. (2007)	Burke et al. (2007)
Degraeve et al. (2000)	Degraeve et al. (2004)
Demirtas and Üstün (2008)	Elmaghraby (2000)

(*Continued*)

Table A.1 (Continued)

Aspect	
Freling (2003)	Gupta and Krishnan (1999)
Hsu et al. (2006)	Huang and Keskar (2007)
Sonmat (2005)	Tempelmeier (2002)
Xia and Wu (2007)	Zhao and Klabjan (2005)
Emergency orders	
Axsäter (2007)	Axsäter (2006)
Gallego et al. (2007)	
Resource substitution	
Begnaud et al. (2006)	Henrich et al. (2007)
Klein and Luss (1991)	Klein et al. (1993)
Product, product family and portfolio design	
Balakrishnan and Chakravarty (2008)	Boysen and Scholl (2008)
de Weck et al. (2003)	D'Souza and Simpson (2003)
Gupta and Krishnan (1999)	Hardung and Kollert (2005)
Hölttä-Otto (2005)	Jiao et al. (2006b)
Jiao et al. (2006a)	Krishnan and Ulrich (2001)
Lang et al. (2008)	Nepal et al. (2009)
Ramdas (2003)	Simpson (2004)
Suh et al. (2004)	Suh (2005)
Thonemann and Brandeau (2000)	
Substitution, APS and ATP/CTP	
Chen et al. (2001)	Chen (2006)
Dickersbach (2006)	Ettl et al. (2006a)
Ettl et al. (2006b)	Faber (2005)
Zhao et al. (2005)	
Flexibility in production and logistics	
Naim et al. (2006)	Petkova and van Wezel (2006)
Pickup and delivery problems	
Berbeglia et al. (2007)	Desaulniers et al. (2002)
Parragh et al. (2008a)	Parragh et al. (2008b)
Coordination of pricing, production, and procurement	
Karakul and Chan (2008)	Kuyumcu and Popescu (2006)
Levis and Papageorgiou (2007)	Tang and Yin (2007)
Yano and Gilbert (2004)	
Hypergraphs	
Ausiello et al. (2001)	Gallo and Pallottino (1992)
Gallo et al. (1993)	Gallo and Scutella (1998a)
Gallo and Scutella (1998b)	Ozturan (2004)
Cutting stock problems	
Poltroniere et al. (2008)	

Bibliography

Absi, N., & Kedad-Sidhoum, S. (2007). MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs. *RAIRO – Operations Research – Recherche Opérationnelle, 41*, 171–192.

Agrawal, N., & Smith, S. (2003). Optimal retail assortments for substitutable items purchased in sets. *Naval Research Logistics, 50*(7), 793–822.

Aissaoui, N., Haouari, M., & Hassini, E. (2007). Supplier selection and order lot sizing modeling: A review. *Computers & Operations Research, 34*(12), 3516–3540.

Akartunalı, K., & Miller, A. (2009). A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research, 193*(2), 396–411.

Alfieri, A., Brandimarte, P., & D'Orazio, S. (2002). LP-based heuristics for the capacitated lot-sizing problem: The interaction of model formulation and solution algorithm. *International Journal of Production Research, 40*(2), 441–458.

Almada-Lobo, B., Klabjan, D., Carravilla, M., & Oliveira, J. (2007). Single machine multi-product capacitated lot sizing with sequence-dependent setups. *International Journal of Production Research, 45*(20), 4873–4894.

Almada-Lobo, B., Oliveira, J., & Carravilla, M. A. (2008). A note on "the capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times". *Computers & Operations Research, 35*(4), 1374–1376.

Andradóttir, S. (1998). A review of simulation optimization techniques. In *Proceedings of the 1998 Winter Simulation Conference* (pp. 151–158).

Anily, S., Tzur, M., & Wolsey, L. (2005). *Multi-item lot-sizing with a joint set-up cost* (Working Paper 2005/70). Louvain, Belgium: Center for Operations Research and Econometrics (CORE), Université Catholique de Louvain.

April, J., Better, M., Glover, F., & Kelly, J. (2004). New advances and applications for marrying simulation and optimization. In *Proceedings of the 2004 Winter Simulation Conference* (pp. 80–86).

April, J., Better, M., Glover, F., Kelly, J., & Laguna, M. (2006). Enhancing business process management with simulation optimization. In *Proceedings of the 2006 Winter Simulation Conference* (pp. 642–649).

April, J., Glover, F., Kelly, J., & Laguna, M. (2003). Simulation-based optimization: Practical introduction to simulation optimization. In *Proceedings of the 2003 Winter Simulation Conference* (pp. 71–78).

Apte, U., & Viswanathan, S. (2007). A proactive demand management model for controlling e-retailer inventory. In U. Apte & U. S. Karmarkar (Eds.), *Annals of Information Systems. Managing in the information economy* (Chap. 15, pp. 355–384). Berlin: Springer.

Archibald, T. W. (2007). Modelling replenishment and transshipment decisions in periodic review multilocation inventory systems. *Journal of the Operational Research Society, 58*(7), 948–956.

Arslan, H., Graves, S., & Roemer, T. (2007). A single-product inventory model for multiple demand classes. *Management Science, 53*(9), 1486–1500.

Ausiello, G., Franciosa, P., & Frigioni, D. (2001). Directed hypergraphs: Problems, algorithmic results, and a novel decremental approach. In A. Restivo, S. R. D. Rocca, & L. Roversi (Eds.), *Lecture Notes in Computer Science: Vol. 2202. 7th Italian Conference ICTCS 2001 Torino* (pp. 312–328). Berlin: Springer.

Axsäter, S. (2003a). Evaluation of unidirectional lateral transshipments and substitutions in inventory systems. *European Journal of Operational Research, 149*(2), 438–447.

Axsäter, S. (2003b). A new decision rule for lateral transshipments in inventory systems. *Management Science, 49*(9), 1168–1179.

Axsäter, S. (2006). *International Series in Operations Research & Management Science. Inventory control* (2nd edn.). Berlin: Springer.

Axsäter, S. (2007). A heuristic for triggering emergency orders in an inventory system. *European Journal of Operational Research, 176*(2), 880–891.

Balakrishnan, A., & Geunes, J. (2000). Requirements planning with substitutions: Exploiting bill-of-materials flexibility in production planning. *Manufacturing & Service Operations Management, 2*(2), 166–185.

Balakrishnan, N. R., & Chakravarty, A. K. (2008). Product design with multiple suppliers for component variants. *International Journal of Production Economics, 112*(2), 723–741.

Ball, M. O., Chen, C.-Y., & Zhao, Z.-Y. (2003). Material compatibility constraints for make-to-order production planning. *Operations Research Letters, 31*(3), 420–428.

Banerjee, A., Burton, J., & Banerjee, S. (2003). A simulation study of lateral shipments in single supplier, multiple buyers supply chain networks. *International Journal of Production Economics, 81–82*, 103–114.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., & Vance, P. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research, 46*(3), 316–329.

Basnet, C., & Leung, J. (2005). Inventory lot-sizing with supplier selection. *Computers & Operations Research, 32*(1), 1–14.

Bassok, Y., Anupindi, R., & Akella, R. (1999). Single-period multiproduct inventory models with substitution. *Operations Research, 47*(4), 632–642.

Bayindir, Z., Erkip, N., & Guellue, R. (2005). Assessing the benefits of remanufacturing option under one-way substitution. *Journal of the Operational Research Society, 56*(3), 286–296.

Bayindir, Z., Erkip, N., & Guellue, R. (2007). Assessing the benefits of remanufacturing option under one-way substitution and capacity constraint. *Computers & Operations Research, 34*(2), 487–514.

Begnaud, J., Miller, L., & Benjaafar, S. (2006). *The multilevel lot sizing problem with flexible production sequences* (Working paper). Minnesota: Industrial and Systems Engineering Department of Mechanical Engineering, University of Minnesota, MN.

Belvaux, G., & Wolsey, L. A. (2000). bc-prod: A specialized branch-and-cut system for lot-sizing problems. *Management Science, 46*(5), 724–738.

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

Ben-Tal, A., & Nemirovski, A. (2002). Robust optimization – methodology and applications. *Mathematical Programming, 92*(3), 453–480.

Benjaafar, S., Elahi, E., & Donohue, K. (2007). Outsourcing via service competition. *Management Science, 53*(2), 241.

Benjaafar, S., & ElHafsi, M. (2006). Production and inventory control of a single product assemble-to-order system with multiple customer classes. *Management Science, 52*(12), 1896–1912.

Beraldi, P., Ghiani, G., Grieco, A., & Guerriero, E. (2006). Fix and relax heuristic for a stochastic lot-sizing problem. *Computational Optimization and Applications, 33*(2), 303–318.

Beraldi, P., Ghiani, G., Grieco, A., & Guerriero, E. (2008). Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. *Computers & Operations Research, 35*(11), 3644–3656.

Berbeglia, G., Cordeau, J., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP – An Official Journal of the Spanish Society of Statistics and Operations Research, 15*(1), 1–31.

Berbner, R., Spahn, M., Repp, N., Heckmann, O., & Steinmetz, R. (2006). Heuristics for QoS-aware web service composition. In *Proceedings of the IEEE International Conference on Web Services (ICWS'06)* (pp. 72–82).

Bertsimas, D., & de Boer, S. (2005). Simulation-based booking limits for airline revenue management. *Operations Research, 53*(1), 90–106.

Bhaumik, P., & Kataria, S. (2006). Lateral transshipment for managing excesses and shortages in a multilocation inventory system: A case study of Timex Watches Ltd. *International Journal of Services Technology and Management, 7*(5), 602–614.

Birge, J. R., Drogosz, J., & Duenyas, I. (1998). Setting single-period optimal capacity levels and prices for substitutable products. *International Journal of Flexible Manufacturing Systems, 10*(4), 407–430.

Bitran, G., Caldentey, R., & Vial, R. (2005). *Pricing policies for perishable products with demand substitution* (Working paper). Cambridge, MA: Sloan School of Management, Massachusetts Institute of Technology.

Bitran, G. R., & Dasu, S. (1992). Ordering policies in an environment of stochastic yields and substitutable demands. *Operations Research, 40*(5), 999–1017.

Bowden, R., & Hall, J. (1998). Simulation optimization research and development. In *Proceedings of the 1998 Winter Simulation Conference* (pp. 1693–1698).

Bowman, E. (1956). Production scheduling by the transportation method of linear programming. *Operations Research, 4*(1), 100–103.

Boyer, K., & Hult, G. (2005). Extending the supply chain: Integrating operations and marketing in the online grocery industry. *Journal of Operations Management, 23*(6), 642–661.

Boyer, K., Hult, G., & Frohlich, M. (2003). An exploratory analysis of extended grocery supply chain operations and home delivery. *Integrated Manufacturing Systems, 14*(8), 652–663.

Boysen, N., & Scholl, A. (2008). *A general solution framework for component commonality problems* (Working Paper 11/2008). Jena Research Papers in Business and Economics (JBE). Jena, Germany: Friedrich-Schiller-Universität Jena.

Brahimi, N., Dauzere-Peres, S., & Najid, N. (2006). Capacitated multi-item lot-sizing problems with time windows. *Operations Research, 54*(5), 951–967.

Brandimarte, P. (2006). Multi-item capacitated lot-sizing with demand uncertainty. *International Journal of Production Research, 44*(15), 2997–3022.

Bretzke, W. (2006). SCM: Sieben Thesen zur zukünftigen Entwicklung logistischer Netzwerke. *Supply Chain Management, 6*, 7–15.

BSMS. (2003). *Inventory practice survey 2003* (Working paper). Blood Stocks Management Scheme, National Health Service, UK.

Burke, G. J., Carrillo, J. E., & Vakharia, A. J. (2007). Single versus multiple supplier sourcing strategies. *European Journal of Operational Research, 182*(1), 95–112.

Buschkühl, L., Sahling, F., Helber, S., & Tempelmeier, H. (2008). Dynamic capacitated lot-sizing problems: A classification and review of solution approaches. *forthcoming in OR Spectrum*.

Cai, L., Chen, J., & Yan, H. (2004). Technical note: Single-period two-product inventory model with substitution: Solution and properties. *Journal of Systems Science and Systems Engineering, 13*, 112–123.

Campbell, A., Clarke, L., & Savelsbergh, M. (2002). Inventory routing in practice. In P. Toth & D. Vigo (Eds.), *SIAM Monographs on Discrete Mathematics and Applications. The vehicle routing problem* (Chap. 12, pp. 309–330). Philadelphia: Society for Industrial & Applied Mathematics.

Canfora, G., Penta, M. D., Esposito, R., & Villani, M. L. (2005). An approach for QoS-aware service composition based on genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)* (pp. 1069–1075), Washington DC. ACM Press.

Caserta, M., & Rico, E. (2009). A cross entropy-lagrangean hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times. *Computers & Operations Research, 36*(2), 530–548.

Chand, S., Hsu, V., & Sethi, S. (2002). Forecast, solution, and rolling horizons in operations management problems: A classified bibliography. *Manufacturing & Service Operations Management, 4*(1), 25–43.

Chand, S., Ward, J., & Weng, Z. (1994). A parts selection model with one-way substitution. *European Journal of Operational Research, 73*(1), 65–69.

Chang, H., Jula, H., Chassiakos, A., & Ioannou, P. (2008). A heuristic solution for the empty container substitution problem. *Transportation Research Part E: Logistics and Transportation Review, 44*(2), 203–216.

Chang, W., Wu, C., & Chang, C. (2005). Optimizing dynamic web service component composition by using evolutionary algorithms. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence* (pp. 708–711).

Chapman, J., Milkins, C., & Voak, D. (2000). The computer crossmatch: A safe alternative to the serological crossmatch. *Transfusion Medicine, 10*(4), 251–256.

Chen, C., Zhao, Z., & Ball, M. (2001). Quantity and due date quoting available to promise. *Information Systems Frontiers, 3*(4), 477–488.

Chen, J. (2003). Component allocation in multi-echelon assembly systems with linked substitutes. *Computers & Industrial Engineering, 45*(1), 43–60.

Chen, J., & Chen, H. (2004). Coordination mechanism for postponement strategy with downward substitutable products. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics* (pp. 6003–6008).

Chen, L., & Plambeck, E. (2008). Dynamic inventory management with learning about the demand distribution and substitution probability. *Manufacturing & Service Operations Management, 10*(2), 236.

Chen, M. (2006). *Coordinating demand fulfillment with supply across a dynamic supply chain*. PhD thesis, Decision & Information Technologies Department, University of Maryland, College Park, MD.

Cheung, K., & Lee, H. (2002). The inventory benefit of shipment coordination and stock rebalancing in a supply chain. *Management Science, 48*(2), 300–306.

Chong, J., Ho, T., & Tang, C. (2004). Demand modeling in product line trimming: Substitutability and variability. In A. K. Chakravarty & J. Eliashberg (Eds.), *Managing business interfaces – Marketing and engineering issues in the supply chain and internet domains* (Chap. 2, pp. 39–62). Berlin: Springer.

Chou, M., Sim, M., & So, K. (2006). *A robust optimization framework for analyzing distribution systems with transshipment* (Working paper). Singapore: School of Business, National University of Singapore.

Cohen, M., & Pierskalla, W. (1979). Target inventory levels for a hospital blood bank or a decentralized regional blood banking system. *Transfusion, 19*(4), 444–454.

Comez, N., Stecke, K. E., & Cakanyldrm, M. (2006). *Virtual pooling considering transshipment lead time* (Working paper). Dallas, TX: School of Management, University of Texas at Dallas.

Crama, Y., Pochet, Y., & Wera, Y. (2001). *A discussion of production planning approaches in the process industry* (Working Paper 2001/42). Louvain, Belgium: Center for Operations Research and Econometrics (CORE), Université Catholique de Louvain.

de Araujo, S., Arenales, M., & Clark, A. (2007). Joint rolling-horizon scheduling of materials processing and lot-sizing with sequence-dependent setups. *Journal of Heuristics, 13*(4), 337–358.

de Weck, O., Suh, E., & Chang, D. (2003). Product family and platform portfolio optimization. In *Proceedings of the ASME International Design Engineering Technical Conference (DETC'03)*.

DeCroix, G., Song, J., & Zipkin, P. (2005). *Managing an assemble-to-order system with returns* (Working paper). Durham, NC: The Fuqua School of Business, Duke University.

DeCroix, G., & Zipkin, P. (2005). Inventory management for an assembly system with product or component returns. *Management Science, 51*, 1250–1265.

Degraeve, Z., & Jans, R. (2007). A new Dantzig-Wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research, 55*(5), 909–920.

Degraeve, Z., Labro, E., & Roodhooft, F. (2000). An evaluation of vendor selection models from a total cost of ownership perspective. *European Journal of Operational Research, 125*(1), 34–58.

Degraeve, Z., Labro, E., & Roodhooft, F. (2004). Total cost of ownership purchasing of a service: The case of airline selection at Alcatel Bell. *European Journal of Operational Research, 156*(1), 23–40.

Delaney-Klinger, K., Boyer, K., & Frohlich, M. (2003). The return of online grocery shopping: A comparative analysis of Webvan and Tesco's operational methods. *TQM Magazine, 15*(3), 187–196.

Demirtas, E. A., & Üstün, Ö. (2008). An integrated multiobjective decision making process for supplier selection and order allocation. *Omega, 36*(1), 76–90.

Deniz, B., Karaesmen, I., & Scheller-Wolf, A. (2005). *Managing inventories of perishable goods* (Working paper). Pittsburgh, PA: Tepper School of Business, Carnegie Mellon University.

Denizel, M., Altekin, F. T., Süral, H., & Stadtler, H. (2008). Equivalence of the LP relaxations of two strong formulations for the capacitated lot-sizing problem with setup times. *OR Spectrum, 30*(4), 773–785.

Denizel, M., & Süral, H. (2006). On alternative mixed integer programming formulations and LP-based heuristics for lot-sizing with setup times. *Journal of the Operational Research Society, 57*(4), 389–399.

Denton, B., & Gupta, D. (2004). Strategic inventory deployment in the steel industry. *IIE Transactions, 36*(11), 1083–1097.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M., & Soumis, F. (2002). VRP with pickup and delivery. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem* (pp. 225–242). Philadelphia: SIAM.

Desrosiers, J., & Lübbecke, M. E. (2005). A primer in column generation. In G. Desaulniers, J. Desrosiers, & M. Solomon (Eds.), *Column generation* (pp. 1–32). Berlin: Springer.

Di Penta, M., & Troiano, L. (2005). *Using fuzzy logic to relax constraints in GA-based service composition* (Working paper). Benevento, Italy: Research Centre on Software Technology, Department of Engineering, University of Sannio.

Diaby, M., Bahl, H., Karwan, M., & Zionts, S. (1992a). Capacitated lot-sizing and scheduling by lagrangean relaxation. *European Journal of Operational Research, 59*(3), 444–458.

Diaby, M., Bahl, H., Karwan, M., & Zionts, S. (1992b). A lagrangean relaxation approach for very-large-scale capacitated lot-sizing. *Management Science, 38*(9), 1329–1340.

Dickersbach, J. T. (2006). *Supply chain management with APO: Structures, modelling approaches, and implementation of MySAP SCM 4.1* (2nd edn.). Berlin: Springer.

Domschke, W. (1997). *Logistik: Rundreisen und Touren* (4th edn.). Munich: Oldenbourg Wissenschaftsverlag

Domschke, W., & Scholl, A. (2005). *Grundlagen der Betriebswirtschaftslehre* (3rd edn.). Berlin: Springer.

Domschke, W., Scholl, A., & Voß, S. (1997). *Produktionsplanung – Ablauforganisatorische Aspekte* (2nd edn.). Berlin: Springer.

Drechsel, J., & Kimms, A. (2008). *Solutions and fair cost allocations for cooperative lot sizing with transshipments and scarce capacities* (Working paper). Lehrstuhl für Logistik und Verkehrsbetriebslehre. Duisburg, Germany: Mercator School of Management, University of Duisburg-Essen.

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

Drezner, Z., Gurnani, H., & Pasternack, B. A. (1995). An EOQ model with substitutions between products. *The Journal of the Operational Research Society, 46*(7), 887–891.

D'Souza, B., & Simpson, T. (2003). A genetic algorithm based method for product family design optimization. *Engineering Optimization, 35*(1), 1–18.

Duenyas, I. P., & Tsai, C. Y. P. (2000). Control of a manufacturing system with random product yield and downward substitutability. *IIE Transactions, 32*(9), 785–795.

Duran, S. (2007). Optimal production and inventory policies of priority and price-differentiated customers. *IIE Transactions, 39*(9), 845–861.

Edwards, T. (2006). Have you ever thought about becoming a hero? *Census CounterParts, 15*(1), 6. US Census Bureau.

Elmaghraby, W. (2000). Supply contract competition and sourcing policies. *Manufacturing & Service Operations Management, 2*(4), 350–371.

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

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research, 26*(6), 992–1009.

Ernst, R., & Kouvelis, P. (1999). The effects of selling packaged goods on inventory decisions. *Management Science, 45*(8), 1142–1155.

Ertogral, K., & Wu, S. (2000). Auction-theoretic coordination of production planning in the supply chain. *IIE Transactions, 32*(10), 931–940.

Ervolina, T., Ettl, M., Lee, Y., & Peters, D. (2009). Managing product availability in an assemble-to-order supply chain with multiple customer segments. *OR Spectrum, 31*(1), 257–280.

Ettl, M., Huang, P., Sourirajan, K., & Cheng, F. (2006a). *Product offering conditioning in assemble-to-order supply chains* (Working paper). Yorktown Heights, NY: IBM Research Division, Thomas J. Watson Research Center.

Ettl, M., Huang, P., Sourirajan, K., Ervolina, T. R., & Lin, G. Y. (2006b). *Supply and demand synchronization in assemble-to-order supply chains* (Working paper). Yorktown Heights, NY: IBM Research Division, Thomas J. Watson Research Center.

Evers, P. (2001). Heuristics for assessing emergency transshipments. *European Journal of Operational Research, 129*(2), 311–316.

Faber, F. (2005). *An extensible order promising and revenue management test-bed*. Master's thesis, Decision & Information Technologies Department, University of Maryland, College Park, MD.

Federgruen, A., Meissner, J., & Tzur, M. (2007). Progressive interval heuristics for multi-item capacitated lot sizing problems. *Operations Research, 55*(3), 490.

Fink, A., & Reiners, T. (2006). Modeling and solving the short-term car rental logistics problem. *Transportation Research Part E, 42*(4), 272–292.

Fisher, M. (1981). The lagrangian relaxation method for solving integer programming problems. *Management Science, 27*(1), 1–18.

Fixson, S. (2007). Modularity and commonality research: Past developments and future opportunities. *Concurrent Engineering, 15*(2), 85.

Fleischmann, B. (1994). The discrete lot-sizing and scheduling problem with sequence-dependent setup costs. *European Journal of Operational Research, 75*(2), 395–404.

Fleischmann, B. (2001). On the use and misuse of holding cost models. In P. Kischka, U. Leopold-Wildburger, R. Möhring, & F.-J. Radermacher (Eds.), *Models, methods and decision support for management: Essays in honor of Paul Stähly* (pp. 147–164). Heidelberg: Physica.

Fleischmann, B., & Meyr, H. (1997). The general lotsizing and scheduling problem. *OR Spectrum, 19*(1), 11–21.

Fleischmann, B., & Meyr, H. (2003). Customer orientation in Advanced Planning Systems. In H. Dyckhoff, R. Lackes, & J. Reese (Eds.), *Supply chain management and reverse logistics* (pp. 297–322). Berlin: Springer.

Fleischmann, B., Meyr, H., & Wagner, M. (2005). Advanced planning. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 4, pp. 81–106). Berlin: Springer.

Förster, A., Haase, K., & Tönnies, M. (2006). Ein modellgestützter Ansatz zur mittelfristigen Produktions-und Ablaufplanung für eine Brauerei. *Zeitschrift für Betriebswirtschaft, 76*(12), 1255–1274.

Freling, R. (2003). A branch and price algorithm for the multi-period single-sourcing problem. *Operations Research, 51*(6), 922–939.

Fu, M. (2001). Simulation optimization. In *Proceedings of the 2001 Winter Simulation Conference* (pp. 53–61).

Fu, M. (2002). Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing, 14*(3), 192–215.

Fu, M. C. (2006). Stochastic gradient estimation. In S. Henderson & B. Nelson (Eds.), *Handbook on operations research and management science: Simulation*. Amsterdam: Elsevier.

Gallego, G., Iyengar, G., Phillips, R., & Dubey, A. (2004). *Managing flexible products on a network* (Working paper). New York: Department of Industrial Engineering and Operations Research, Columbia University.

Gallego, G., Jin, Y., Muriel, A., Zhang, G., & Yildiz, V. T. (2007). Optimal ordering policies with convertible lead times. *European Journal of Operational Research, 176*(2), 892–910.

Gallego, G., Katircioglu, K., & Ramachandran, B. (2006). Semiconductor inventory management with multiple grade parts and downgrading. *Production Planning & Control, 17*(7), 689–700.

Gallego, G., & Phillips, R. (2004). Revenue management of flexible products. *Manufacturing & Service Operations Management, 6*(4), 321–337.

Gallo, G., Longo, G., Nguyen, S., & Pallottino, S. (1993). *Directed hypergraphs and applications* (Working paper). Pisa, Italy: Dipartimento di Informatica, Universita di Pisa.

Gallo, G., & Pallottino, S. (1992). *Hypergraph models and algorithms for the assembly problem* (Working paper). Pisa, Italy: Dipartimento di Informatica, Universita di Pisa.

Gallo, G., & Scutella, M. (1998a). Directed hypergraphs as a modelling paradigm. *Rivista AMASES, 21*, 97–123.

Gallo, G., & Scutella, M. (1998b). *Minimum makespan assembly plans* (Working paper). Pisa, Italy: Dipartimento di Informatica, Universita di Pisa.

Ganesh, M., Raghunathan, S., & Rajendran, C. (2008). The value of information sharing in a multi-product supply chain with product substitution. *IIE Transactions, 40*(12), 1124–1140.

Gaur, V., & Honhon, D. (2006). Assortment planning and inventory decisions under a locational choice model. *Management Science, 52*(10), 1528–1543.

Gebhard, M., & Kuhn, H. (2007). Robuste hierarchische Produktionsplanung mit Bedarfsszenarien. In A. Otto & R. Obermaier (Eds.), *Logistikmanagement 2007: Analyse, Bewertung und Gestaltung logistischer Systeme* (pp. 161–183). Wiesbaden, Germany: Gabler / Deutscher Universitäts-Verlag.

Georgsen, J., & Kristensen, T. (1998). From serological to computer cross-matching in nine hospitals. *Vox Sanguinis, 74*(2), 419–25.

Gerchak, Y., & Grosfeld-Nir, A. (1999). Lot-sizing for substitutable, production-to-order parts with random functionality yields. *International Journal of Flexible Manufacturing Systems, 11*(4), 371–377.

Geunes, J. (2003). Solving large-scale requirements planning problems with component substitution options. *Computers & Industrial Engineering, 44*(3), 475–491.

Gosavi, A. (2003). *Simulation-based optimization: Parametric optimization techniques and reinforcement learning*. Boston: Kluwer.

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

Gunasekaran, A., & Ngai, E. (2005). Build-to-order supply chain management: A literature review and framework for development. *Journal of Operations Management, 23*(5), 423–451.

Gupta, D., & Magnusson, T. (2005). The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers & Operations Research, 32*(4), 727–747.

Gupta, S., & Krishnan, V. (1999). Integrated component and supplier selection for a product family. *Production and Operations Management, 8*(2), 163–181.

Gürkan, G., Özge, A., & Robinson, S. (1994). Sample-path optimization in simulation. In *Proceedings of the 1994 Winter Simulation Conference* (pp. 247–254).

Gurnani, H., & Drezner, Z. (2000). Deterministic hierarchical substitution inventory models. *Journal of the Operational Research Society, 51*(1), 129–133.

Gutin, G., & Punnen, A. (2002). *The Traveling Salesman Problem and its variations*. Boston: Kluwer.

Haase, K. (1996). Capacitated lot-sizing with sequence dependent setup costs. *OR Spectrum, 18*(1), 51–59.

Haase, K. (2001). Beschaffungs-Controlling – Kapitalwertorientierte Bestellmengenplanung bei Mengenrabatten und dynamischer Nachfrage. *Zeitschrift für Betriebswirtschaft, 71*, 19–31.

Haase, K., & Kimms, A. (2000). Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics, 66*(2), 159–169.

Hale, W., Pyke, D. F. P., & Rudi, N. (2000). *An assemble-to-order system with component substitution* (Working paper). Hanover, NH / Rochester, NY: Amos Tuck School, Dartmouth College / The Simon School, University of Rochester.

Hansen, P., Mladenović, N., & Moreno Pérez, J. (2008). Variable neighbourhood search: Methods and applications. *4OR: A Quarterly Journal of Operations Research, 6*(4), 319–360.

Hardung, B., & Kollert, T. (2005). Optimisation of the variant combination of control units considering the order history. In H.-D. Haasis, H. Kopfer, & J. Schönberger (Eds.), *Operations research proceedings 2005*. Berlin: Springer.

Hax, A., & Meal, H. (1975). Hierarchical integration of production planning and scheduling. In M. Geisler (Ed.), *TIMS studies in the management sciences* (Vol. 1, pp. 53–69). Amsterdam: North-Holland.

Helber, S. (1994). *Kapazitätsorientierte Losgrößenplanung in PPS-Systemen*. Metzler & Poeschel Verlag für Wissenschaft und Forschung, Stuttgart, Germany.

Helber, S., & Sahling, F. (2008). *A Fix-and-Optimize Approach for the multi-level capacitated lot sizing problem* (Working paper). Hannover, Germany: Institut für Produktionswirtschaft. Leibniz Universität Hannover.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research, 126*(1), 106–130.

Helsgaun, K. (2007). *An effective implementation of K-opt moves for the Lin-Kernighan TSP heuristic* (Working paper). Roskilde, Denmark: Roskilde University.

Hemmelmayr, V., Doerner, K., Hartl, R., & Savelsbergh, M. (2007). *Delivery strategies for blood products supplies* (Working paper). Vienna: Department of Business Administration, University of Vienna.

Henrich, P., Land, M., & Gaalman, G. (2007). Semi-interchangeable machines: Implications for workload control. *Production Planning & Control, 18*(2), 91–104.

Herer, Y., Tzur, M., & Yücesan, E. (2006). The multilocation transshipment problem. *IIE Transactions, 38*(3), 185–200.

Hohenegger, J., Bufardi, A., & Xirouchakis, P. (2007). A new concept of compatibility structure in new product development. *Advanced Engineering Informatics, 21*, 101–116.

Hölttä-Otto, K. (2005). *Modular product platform design*. PhD thesis, Helsinki University of Technology, Finland.

Honhon, D., Gaur, V., & Seshadri, S. (2006). *Assortment planning and inventory management under dynamic stockout-based substitution* (Working paper). New York: Department of Information, Operations and Management Science, Leonard N. Stern School of Business, New York University.

Hood, S., & Welch, P. (1993). Response Surface Methodology and its application in simulation. In *Proceedings of the 1993 Winter Simulation Conference* (pp. 115–122).

Hoos, H., & Stützle, T. (2005). *Stochastic local search: Foundations and applications*. San Francisco, CA: Morgan Kaufmann.

Hsieh, C.-C., & Wu, C.-H. (2009). Coordinated decisions for substitutable products in a common retailer supply chain. *European Journal of Operational Research, 196*(1), 273–288.

Hsu, A., & Bassok, Y. (1999). Random yield and random demand in a production system with downward substitution. *Operations Research, 47*(2), 277–290.

Hsu, C., Kannan, V., Leong, G., & Tan, K. (2006). Supplier selection construct: Instrument development and validation. *The International Journal of Logistics Management, 17*(2), 213–239.

Hsu, V. N., Li, C. L., & Xiao, W. Q. (2005). Dynamic lot size problems with one-way product substitution. *IIE Transactions, 37*, 201–215.

Huang, K. (2005). *Multi-stage stochastic programming models in production planning*. PhD thesis, Georgia Institute of Technology, Atlanta, GA.

Huang, S. H., & Keskar, H. (2007). Comprehensive and configurable metrics for supplier selection. *Int. J. Production Economics, 105*, 510–523.

Huisman, D., Jans, R., Peeters, M., & Wagelmans, A. (2005). Combining column generation and lagrangian relaxation. In D. Guy, S. Jacques, & M. Marius (Eds.), *Column generation* (pp. 247–270). Berlin: Springer.

Ignall, E., & Veinott, Jr., A. F. (1969). Optimality of myopic inventory policies for several substitute products. *Management Science, 15*(5), 284–304.

Inderfurth, K. (2004). Optimal policies in hybrid manufacturing/remanufacturing systems with product substitution. *International Journal of Production Economics, 90*(3), 325–343.

Iravani, S., Lien, R., Smilowitz, K., & Tzur, M. (2005). *Design principles for effective transshipment networks* (Working paper). Evanston, IL: IE/MS department, Northwestern University, IL.

Jans, R., & Degraeve, Z. (2006). Modeling industrial lot sizing problems: A review. *International Journal of Production Research, 46*(6), 1619–1643.

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

Jans, R., Degraeve, Z., & Schepens, L. (2008). Analysis of an industrial component commonality problem. *European Journal of Operational Research, 186*(2), 801–811.

Jennings, J. (1973). Blood bank inventory control. *Management Science, 19*(6), 637–645.

Jiao, J., Simpson, T., & Siddique, Z. (2006a). Product family design and platform-based product development: A state-of-the-art review. *Journal of Intelligent Manufacturing, 18*(1), 5–29.

Jiao, J., Zhang, Y., & Wang, Y. (2006b). A generic genetic algorithm for product family design. *Journal of Intelligent Manufacturing*.

Jones, P. C., Lowe, T. J., Muller, G., Xu, N., Ye, Y., & Zydiak, J. L. (1995). Specially structured uncapacitated facility location problems. *Operations Research, 43*(4), 661–669.

Kallrath, J. (2005). Solving planning and design problems in the process industry using mixed integer and global optimization. *Annals of Operations Research, 140*(1), 339–373.

Kallrath, J., & Maindl, T. (2006). *Real optimization with SAP APO*. Berlin: Springer.

Karaesmen, I., Scheller-Wolf, A., & Deniz, B. (2008). Managing perishable and aging invetories: Review and future research direction. In K. Kempf, P. Keskinocak, & P. Uzsoy (Eds.), *Handbook of production planning. International Series in Operations Research and Management Science, Advancing the state-of-the-art subseries.* Boston: Kluwer.

Karaesmen, I., & van Ryzin, G. (2004). Overbooking with substitutable inventory classes. *Operations Research, 52*(1), 83–104.

Karakul, M., & Chan, L. (2008). Analytical and managerial implications of integrating product substitutability in the joint pricing and procurement problem. *European Journal of Operational Research, 190*, 179–204.

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

Katsaliaki, K., & Brailsford, S. (2007). Using simulation to improve the blood supply chain. *Journal of the Operational Research Society, 58*, 219–227.

Kennedy, W. J., Patterson, J. W., & Fredendall, L. D. (2002). An overview of recent literature on spare parts inventories. *International Journal of Production Economics, 76*(2), 201–215.

Kerkkänen, A. (2007). Determining semi-finished products to be stocked when changing the MTS-MTO policy: Case of a steel mill. *International Journal of Production Economics, 108*(1–2), 111–118.

Kilger, C., & Schneeweiß, L. (2005). Demand Fulfilment and ATP. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 9, pp. 179–195). Berlin: Springer.

Kleber, R., & Inderfurth, K. (2007). A heuristic approach for inventory control of spare parts after end-of-production. In A. Otto & R. Obermaier (Eds.), *Logistikmanagement 2007:*

Analyse, Bewertung und Gestaltung logistischer Systeme (pp. 185–200). Wiesbaden, Germany: Gabler / Deutscher Universitäts-Verlag.

Kleijn, M., & Dekker, R. (1998). *An overview of inventory systems with several demand classes* (Working paper). Rotterdam, Netherlands: Erasmus Research Institute of Management (ERIM), Rotterdam School of Management / Rotterdam School of Economics, Erasmus Universiteit Rotterdam.

Klein, R., & Luss, H. (1991). Minimax resource allocation with tree structured substitutable resources. *Operations Research, 39*(2), 285–295.

Klein, R., Luss, H., & Rothblum, U. (1993). Minimax resource allocation problems with resource-substitutions represented by graphs. *Operations Research, 41*(5), 959–971.

Kocaga, Y., & Sen, A. (2007). Spare parts inventory management with demand lead times and rationing. *IIE Transactions, 39*(9), 879–898.

Kok, A., & Fisher, M. (2007). Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research, 55*(6), 1001–1021.

Kolda, T., Lewis, R., & Torczon, V. (2004). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review, 45*, 385–482.

Kondili, E., Pantelides, C., & Sargent, R. (1993). General algorithm for short-term scheduling of batch operations – I. MILP formulation. *Computers & Chemical Engineering, 17*(2), 211–227.

Kopach, R., Frances, D., & Sadat, S. (2003). Models for predicting critical blood product shortages. In *Proceedings of the 29th Meeting of the European Working Group on Operational Research Applied to Health Services, ORAHS* (pp. 77–90), Prague, Czech Republic.

Kouvelis, P., & Yu, G. (1997). *Robust discrete optimization and its applications*. Boston: Kluwer.

Kranenburg, A., & van Houtum, G. (2007). Cost optimization in the (s–1, s) lost sales inventory model with multiple demand classes. *Operations Research Letters, 35*(4), 493–502.

Krarup, J., & Bilde, O. (1977). Plant location, set covering and economic lot size: An O(mn)-algorithm for structured problems. In L. Collatz, G. Meinardus, & W. Wetterling (Eds.), *Numerische Methoden bei Optimierungsaufgaben, Band 3: Optimierung bei graphentheoretischen und ganzzahligen Problemen* (pp. 155–180). Birkhäuser: Stuttgart.

Krishnan, V., & Ulrich, K. (2001). Product development decisions: A review of the literature. *Management Science, 47*(1), 1–21.

Kuyumcu, A., & Popescu, I. (2006). Deterministic price-inventory management for substitutable products. *Journal of Revenue and Pricing Management, 4*(4), 354–366.

Lang, J. C. (2005). *Methoden der Simulationsbasierten Optimierung*. Semester thesis, Department of Law, Business Administration and Economics, Technische Universität Darmstadt, Germany.

Lang, J. C. (2008). *Multi-location transshipment and substitution problems: An application to blood banks* (Working paper). Schriften zur Quantitativen Betriebswirtschaftslehre. Darmstadt, Germany: Department of Law, Business Administration and Economics, Technische Universität Darmstadt.

Lang, J. C., & Domschke, W. (2008). Efficient reformulations for dynamic lot-sizing problems with product substitution. *OR Spectrum.* doi:10.1007/s00291-008-0148-1.

Lang, J. C., Widjaja, T., Buxmann, P., Domschke, W., & Hess, T. (2008). Optimizing the supplier selection and service portfolio of a SOA service integrator. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences* (pp. 89–98).

Law, A. (2006). *Simulation modeling and analysis* (4th edn.). New York: McGraw-Hill.

Law, A., & McComas, M. (2002). Simulation optimization: Simulation-based optimization. In *Proceedings of the 2002 Winter Simulation Conference* (pp. 41–44).

Lawler, E., Lenstra, J. K., Kan, A. H. G. R., & Shmoys, D. B. (1985). *The Traveling Salesman Problem: A guided tour of combinatorial optimization*. Wiley Series in Discrete Mathematics & Optimization. New York: Wiley.

Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research, 53*(6), 1007–1023.

Lee, C., Cetinkaya, S., & Wagelmans, A. (2001). A dynamic lot-sizing model with demand time windows. *Management Science, 47*(10), 1384–1395.

Lee, Y. H., Jung, J. W., & Jeon, Y. S. (2007). An effective lateral transshipment policy to improve service level in the supply chain. *International Journal of Production Economics, 106*(1), 115–126.

Leung, S. C., Lai, K. K., Ng, W., & Wu, Y. (2007a). A robust optimization model for production planning of perishable products. *Journal of the Operational Research Society, 58*, 413–422.

Leung, S. C., Tsang, S. O., Ng, W., & Wu, Y. (2007b). A robust optimization model for multi-site production planning problem in an uncertain environment. *European Journal of Operational Research, 182*, 224–238.

Levis, A., & Papageorgiou, L. (2007). Active demand management for substitute products through price optimisation. *OR Spectrum, 29*(4), 551–577.

Lewis, R., & Torczon, V. (1999). Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization, 9*(4), 1082–1099.

Lewis, R., & Torczon, V. (2000). Pattern search methods for linearly constrained minimization. *SIAM Journal on Optimization, 10*, 917–941.

Lewis, R., Torczon, V., & Trosset, M. (1998). *Why pattern search works* (Working paper). Hampton, VA: Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center.

Lewis, R., Torczon, V., & Trosset, M. (2000). Direct search methods: Then and now. *Journal of Computational and Applied Mathematics, 124*(1–2), 191–207.

Li, Y., Chen, J., & Cai, X. (2006). Uncapacitated production planning with multiple product types, returned product remanufacturing, and demand substitution. *OR Spectrum, 28*(1), 101–125.

Li, Y., Chen, J., & Cai, X. (2007). Heuristic genetic algorithm for capacitated production planning problems with batch processing and remanufacturing. *International Journal of Production Economics, 105*(2), 301–317.

Liao, Z., & Rittscher, J. (2007a). Integration of supplier selection, procurement lot sizing and carrier selection under dynamic demand conditions. *International Journal of Production Economics, 107*(2), 502–510.

Liao, Z., & Rittscher, J. (2007b). A multi-objective supplier selection model under stochastic demand conditions. *International Journal of Production Economics, 105*(1), 150–159.

Liu, J., & Lee, C. G. (2007). Evaluation of inventory policies with unidirectional substitutions. *European Journal of Operational Research, 182*, 145–163.

Lu, Y., & Song, J. (2005). Order-based cost optimization in assemble-to-order systems. *Operations Research, 53*(1), 151–169.

Lystad, E., Ferguson, M., & Alexopoulos, C. (2006). *Single stage heuristic for perishable inventory control in two-echelon supply chains* (Working paper). Atlanta, GA: The College of Management, Georgia Institute of Technology.

Ma, S., Wanga, W., & Liu, L. (2000). Commonality and postponement in multistage assembly systems. *European Journal of Operational Research, 142*(3), 523–538.

Maes, J., McClain, J., & Van Wassenhove, L. (1991). Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research, 53*(2), 131–148.

Mahajan, S., & van Ryzin, G. (2001). Stocking retail assortments under dynamic consumer substitution. *Operations Research, 49*(3), 334–351.

Marchand, H., Martin, A., Weismantel, R., & Wolsey, L. (2002). Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics, 123*(1–3), 397–446.

Meixell, M., & Norbis, M. (2008). A review of the transportation mode choice and carrier selection literature. *The International Journal of Logistics Management, 19*(2), 183–211.

Meyr, H. (1999). *Simultane Losgrössen-und Reihenfolgeplanung für kontinuierliche Produktionslinien: Modelle und Methoden im Rahmen des Supply Chain Management*. Wiesbaden, Germany: Gabler / Deutscher Universitäts-Verlag,

Meyr, H. (2000). Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. *European Journal of Operational Research, 120*(2), 311–326.

Meyr, H. (2002). Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research, 139*(2), 277–292.

Meyr, H. (2004a). Simultane Losgrößen-und Reihenfolgeplanung bei mehrstufiger kontinuierlicher Fertigung. *Zeitschrift für Betriebswirtschaft, 74*, 585–610.

Meyr, H. (2004b). Supply chain planning in the German automotive industry. *OR Spectrum, 26*(4), 447–470.

Meyr, H., Rohde, J., & Wagner, M. (2005a). Architecture of selected APS. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 18, pp. 341–353). Berlin: Springer.

Meyr, H., Wagner, M., & Rohde, J. (2005b). Structure of Advanced Planning Systems. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 5, pp. 109–115). Berlin: Springer.

Minner, S. (2003). Multiple-supplier inventory models in supply chain management: A review. *International Journal of Production Economics, 81*(82), 265–279.

Minner, S., Silver, E. A., & Robb, D. J. (2003). An improved heuristic for deciding on emergency transshipments. *European Journal of Operational Research, 148*(2), 384–400.

Moon, I., & Kang, S. (1998). Rationing policies for some inventory systems. *Journal of the Operational Research Society, 49*(5), 509–518.

Mulvey, J., Vanderbei, R., & Zenios, S. (1995). Robust optimization of large-scale systems. *Operations Research, 43*(2), 264–281.

Nahmias, S. (1982). Perishable inventory theory: A review. *Operations Research, 30*(4), 680–708.

Naim, M., Potter, A., Mason, R., & Bateman, N. (2006). The role of transport flexibility in logistics provision. *The International Journal of Logistics Management, 17*(3), 297–311.

Nepal, B., Lassan, G., Drow, B., & Chelst, K. (2009). A set-covering model for optimizing selection of portfolio of microcontrollers in an automotive supplier company. *European Journal of Operational Research, 193*(1), 272–281.

Netessine, S., & Rudi, N. (2003). Centralized and competitive inventory models with demand substitution. *Operations Research, 51*(2), 329–335.

Ng, K. Y. K., & Lam, M. N. (1998). Standardisation of substitutable electrical items. *The Journal of the Operational Research Society, 49*(9), 992–997.

Nonås, L., & Jörnsten, K. (2005). Heuristics in the multi-location inventory system with transshipments. In H. Kotzab & M. Westhaus (Eds.), *Research methodologies in supply chain management* (pp. 509–524). Berlin: Springer.

Nonås, L., & Jörnsten, K. (2007). Optimal solutions in the multi-location inventory system with transshipments. *Journal of Mathematical Modelling and Algorithms, 6*(1), 47–75.

Oliveira, R., & Lourenco, J. (2002). A multicriteria model for assigning new orders to service suppliers. *European Journal of Operational Research, 139*(2), 390–399.

Ozturan, C. (2004). Network flow models for electronic barter exchanges. *Journal of Organizational Computing and Electronic Commerce, 14*(3), 175–194.

Page, B., & Kreutzer, W. (2005). *The Java simulation handbook – Simulating discrete event systems with UML and Java*. Aachen, Germany: Shaker.

Pantelides, C. (1994). Unified frameworks for optimal process planning and scheduling. In *Proceedings on the Second Conference on Foundations of Computer Aided Operations* (pp. 253–274).

Parragh, S., Doerner, K., & Hartl, R. (2008a). A survey on pickup and delivery problems – part I: Transportation between customers and depot. *Journal für Betriebswirtschaft, 58*(1), 21–51.

Parragh, S., Doerner, K., & Hartl, R. (2008b). A survey on pickup and delivery problems – part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft, 58*(2), 81–117.

Pasternack, B., & Drezner, Z. (1991). Optimal inventory policies for substitutable commodities with stochastic demand. *Naval Research Logistics, 38*, 221–240.

Pels, H. (2006). Classification hierarchies for product data modelling. *Production Planning & Control, 17*(4), 367–377.

Pentico, D. W. (1988). The discrete two-dimensional assortment problem. *Operations Research, 36*(2), 324–332.

Pentico, D. W. (2008). The assortment problem: A survey. *European Journal of Operational Research, 190*(2), 295–309.

Pereira, A. (2005). Blood inventory management in the type and screen era. *Vox Sanguinis, 89*(4), 245–250.

Petkova, B., & van Wezel, W. (2006). Disentangling manufacturing flexibility. In *Proceedings of the 14th International Working Seminar on Production Economics*, Vol. 1, pp. 287–295, Innsbruck, Austria.

Pierskalla, W. (2005). Supply chain management of blood banks. In M. Brandeau, F. Sainfort, & W. Pierskalla (Eds.), *Operations research and health care: A handbook of methods and applications* (pp. 103–145). Boston: Kluwer.

Pochet, Y. (2001). Mathematical programming models and formulations for deterministic production planning problems. In M. Jünger & D. Naddef (Eds.), *Lecture Notes in Computer Science: Vol. 2241. Computational combinatorial optimization* (pp. 57–111). Berlin: Springer.

Pochet, Y., & Van Vyve, M. (2004). A general heuristic for production planning problems. *INFORMS Journal on Computing, 16*(3), 316–327.

Pochet, Y., Van Vyve, M., & Wolsey, L. A. (2005). *LS-LIB: A library of reformulations, cut separation algorithms and primal heuristics in a high-level modeling language for solving MIP production planning problems* (Working Paper 2005/47). Louvain, Belgium: Center for Operations Research and Econometrics (CORE), Université Catholique de Louvain.

Pochet, Y., & Wolsey, L. (1991). Solving multi-item lot-sizing problems using strong cutting planes. *Management Science, 37*(1), 53–67.

Pochet, Y., & Wolsey, L. (2006). *Production planning by mixed integer programming*. Berlin: Springer.

Poltroniere, S., Poldi, K., Toledo, F., & Arenales, M. (2008). A coupling cutting stock-lot sizing problem in the paper industry. *Annals of Operations Research, 157*(1), 91–104.

Prastacos, G. (1984). Blood inventory management: An overview of theory and practice. *Management Science, 30*(7), 777–800.

Puchinger, J., & Raidl, G. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In J. Mira & J. R. Alvarez (Eds.), *Lecture Notes in Computer Science: Vol. 3562. Artificial intelligence and knowledge engineering applications: A bioinspired approach* (pp. 41–53). Berlin: Springer.

Quadt, D., & Kuhn, H. (2008). Capacitated lot-sizing with extensions: A review. *4OR: A Quarterly Journal of Operations Research, 6*(1), 61–83.

Quante, R., Meyr, H., & Fleischmann, M. (2009). Revenue management and demand fulfillment: Matching applications, models, and software. *OR Spectrum, 31*(1), 31–62.

Raa, B., & Aghezzaf, E. (2005). A robust dynamic planning strategy for lot-sizing problems with stochastic demands. *Journal of Intelligent Manufacturing, 16*(2), 207–213.

Rajaram, K., & Tang, C. (2001). The impact of product substitution on retail merchandising. *European Journal of Operational Research, 135*(3), 582–601.

Ralphs, T., Ladányi, L., & Saltzman, M. (2003). Parallel branch, cut, and price for large-scale discrete optimization. *Mathematical Programming, 98*(1), 253–280.

Ralphs, T., Ladanyi, L., & Trotter, L. (2001). Branch, cut, and price: Sequential and parallel. In M. Jünger & D. Naddef (Eds.), Lecture Notes in Computer Science: Vol. 2241. Computational combinatorial optimization (pp 223–260). Berlin: Springer

Ram, B., Naghshineh-Pour, M., & Yu, X. (2006). Material requirements planning with flexible bills-of-material. *International Journal of Production Research, 44*(2), 399–415.

Ramdas, K. (2003). Managing product variety: An integrative review and research directions. *Production and Operations Management, 12*(1), 79–101.

Rao, U. N., Swaminathan, J. N., & Zhang, J. (2004). Multi-product inventory planning with downward substitution, stochastic demand and setup costs. *IIE Transactions, 36*(1), 59–71.

Reinelt, G. (1994). *Lecture Notes in Computer Science: Vol. 840. The Traveling Salesman: Computational solutions for TSP applications*. Berlin: Springer.

Richter, M., & Stockrahm, V. (2005). Scheduling of synthetic granulate. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 24, pp. 435–451). Berlin: Springer.

Röder, A., & Tibken, B. (2006). A methodology for modeling inter-company supply chains and for evaluating a method of integrated product and process documentation. *European Journal of Operational Research, 169*(3), 1010–1029.

Rohde, J., & Wagner, M. (2005). Master planning. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 8, pp. 159–177). Berlin: Springer.

Sadowski, W. (1959). A few remarks on the assortment problem. *Management Science, 6*(1), 13–24.

Sahling, F., Buschkühl, L., Tempelmeier, H., & Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a Fix-and-Optimize heuristic. *Computers & Operations Research, 36*(9), 2546–2553.

SAP (2008). SAP Advanced Planner and Optimizer (SAP APO) online documentation. Retrieved from http://help.sap.com/saphelp_apo/helpdata/en/7e/63fc37004d0a1ee10000009b38f8cf/frameset.htm.

Scholl, A. (2001). *Robuste Planung und Optimierung. Grundlagen – Konzepte und Methoden – Experimentelle Untersuchungen*. Berlin: Springer.

Schultmann, F., Fröhling, M., & Rentz, O. (2002). Demontageplanung und -steuerung mit Enterprise-Resource- und Advanced-Planning-Systemen. *Wirtschaftsinformatik, 44*(6), 557–565.

Scott, C., & Scott, J. (2006). Efficient allocation of online grocery orders. *International Journal of Productivity and Quality Management, 1*(1), 88–102.

Shah, J., & Avittathur, B. (2007). The retailer multi-item inventory problem with demand cannibalization and substitution. *International Journal of Production Economics, 106*, 104–114.

Shumsky, R. A., & Zhang, F. (2007). *Dynamic capacity management with substitution* (Working paper). Hanover, NH: Tuck School of Business, Dartmouth College.

Silver, E. (2004). An overview of heuristic solution methods. *Journal of the Operational Research Society, 55*(9), 936–956.

Silver, E., & Minner, S. (2005). A replenishment decision involving partial postponement. *OR Spectrum, 27*(1), 1–19.

Silver, E., & Moon, I. (2001a). Multi-item economic order quantity model with an initial stock of convertible units. *The Engineering Economist, 46*(2), 129–138.

Silver, E., & Moon, I. (2001b). The multi-item single period problem with an initial stock of convertible units. *European Journal of Operational Research, 132*, 466–477.

Simpson, T. (2004). Product platform design and customization: Status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 18*, 3–20.

Sleight, P. (2001). Tesco.com. *Interactive Marketing, 2*(4), 373–383.

Smith, S., & Agrawal, N. (2000). Management of multi-item retail inventory systems with demand substitution. *Operations Research, 48*(1), 50–64.

Snyder, L. (2006). Facility location under uncertainty: A review. *IIE Transactions, 38*(7), 547–564.

Sonmat, M. (2005). *A review and critique of supplier selection process and practices* (Working paper). Leicestershire, UK: The Business School, Loughborough University.

Spall, J. (1998). An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins APL Technical Digest, 19*(4), 482–492.

Spall, J. (2003). *Introduction to stochastic search and optimization*. New York: Wiley.

Sridharan, R. (1995). The capacitated plant location problem. *European Journal of Operational Research, 87*(2), 203–213.

Sriver, T., & Chrissis, J. (2004). Combined pattern search and ranking and selection for simulation optimization. In *Proceedings of the 2004 Winter Simulation Conference* (pp. 645–653).

Stadtler, H. (1996). Mixed integer programming model formulations for dynamic multi-item multi-level capacitated lotsizing. *European Journal of Operational Research, 94*(3), 561–581.

Stadtler, H. (1997). Reformulations of the shortest route model for dynamic multi-item multi-level capacitated lotsizing. *OR Spectrum, 19*(2), 87–96.

Stadtler, H. (2000). Improved rolling schedules for the dynamic single-level lot-sizing problem. *Management Science, 46*(2), 318–326.

Stadtler, H. (2003). Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research, 51*(3), 487–502.

Stadtler, H. (2005). Production planning and scheduling. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 10, pp. 197–214). Berlin: Springer.

Stammen-Hegener, C. (2002). *Simultane Losgrößen-und Reihenfolgeplanung bei ein-und mehrstufiger Fertigung*. Wiesbaden, Germany: Gabler / Deutscher Universitäts-Verlag.

Suerie, C., & Stadtler, H. (2003). The capacitated lot-sizing problem with linked lot sizes. *Management Science, 49*(8), 1039–1054.

Suh, E. (2005). *Flexible product platforms*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

Suh, E., Kim, I., de Weck, O., & Chang, D. (2004). Design for flexibility: Performance and economic optimization of product platform components. In *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*.

Swaminathan, J., & Tayur, S. (1998). Managing broader product lines through delayed differentiation using vanilla boxes. *Management Science, 44*(12), 161–172.

Swaminathan, J., & Tayur, S. (1999). Managing design of assembly sequences for product lines that delay product differentiation. *IIE Transactions, 31*(11), 1015–1026.

Swisher, J., Hyden, P., Jacobson, S., Schruben, L., Hosp, M., & Fredericksburg, V. (2000). A survey of simulation optimization techniques and procedures. In *Proceedings of the 2000 Winter Simulation Conference* (pp. 119–128).

Tang, C., & Yin, R. (2007). Joint ordering and pricing strategies for managing substitutable products. *Production and Operations Management, 16*(1), 138–153.

Tekin, E., & Sabuncuoglu, I. (2004). Simulation optimization: A comprehensive review on theory and applications. *IIE Transactions, 36*(11), 1067–1081.

Tempelmeier, H. (2002). A simple heuristic for dynamic order sizing and supplier selection with time-varying data. *Production and Operations Management, 11*(4), 499–515.

Tempelmeier, H. (2006). *Inventory management in supply networks – Problems, models, solutions*. Norderstedt, Germany: Books on Demand.

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

Tempelmeier, H., & Buschkühl, L. (2008). Dynamic multi-machine lotsizing and sequencing with simultaneous scheduling of a common setup resource. *International Journal of Production Economics, 113*(1), 401–412.

Tempelmeier, H., & Derstroff, M. (1996). A lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science, 42*(5), 738–757.

Tempelmeier, H., & Helber, S. (1994). A heuristic for dynamic multi-item multi-level capacitated lotsizing for general product structures. *European Journal of Operational Research, 75*(2), 296–311.

Teunter, R. H., & Haneveld, W. K. K. (2008). Dynamic inventory rationing strategies for inventory systems with two demand classes, Poisson demand and backordering. *European Journal of Operational Research, 190*, 156–178.

Thizy, J., & van Wassenhove, L. (1985). Lagrangean relaxation for the multi-item capacitated lot-sizing problem: A heuristic implementation. *IIE Transactions, 17*(4), 308–313.

Thomas, D. J., & Warsing, D. P. (2007). A periodic inventory model for stocking modular components. *Production and Operations Management, 16*(3), 343–359.

Thonemann, U., & Brandeau, M. (2000). Optimal commonality in component design. *Operations Research, 48*(1), 1–19.

Toledo, F., & Armentano, V. (2006). A lagrangian-based heuristic for the capacitated lot-sizing problem in parallel machines. *European Journal of Operational Research, 175*(2), 1070–1083.

van Hoek, R. (2001). The rediscovery of postponement: A literature review and directions for research. *Journal of Operations Management, 19*(2), 161–184.

Vanderbeck, F. (2003). *Automated Dantzig-Wolfe re-formulation or how to exploit simultaneously original formulation and column generation re-formulation* (Working paper). Bordeaux, France: Department of Applied Mathematics, University Bordeaux.

Vanderbeck, F. (2005). *Implementing mixed integer column generation* (Working paper). Bordeaux, France: Department of Applied Mathematics, University Bordeaux.

Varian, H. (2007). *Intermediate microeconomics: A modern approach* (7th edn.). New York: WW Norton.

Vyve, M. V., & Wolsey, L. A. (2006). Approximate extended formulations. *Mathematical Programming, 105*(2), 501–522.

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

Wagner, M., & Meyr, H. (2005). Food and beverages. In H. Stadtler & C. Kilger (Eds.), *Supply chain management and advanced planning: Concepts, models, software and case studies* (3rd edn., Chap. 20, pp. 371–388). Berlin: Springer.

Wedekind, H., & Müller, T. (1981). Stücklistenorganisation bei einer großen Variantenzahl. *Angewandte Informatik, 23*(9), 377–383.

Wee, K., & Dada, M. (2005). Optimal policies for transshipping inventory in a retail network. *Management Science, 51*(10), 1519–1533.

Wilhelm, W. (2001). A technical review of column generation in integer programming. *Optimization and Engineering, 2*(2), 159–200.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.

Wolsey, L. (1997). MIP modelling of changeovers in production planning and scheduling problems. *European Journal of Operational Research, 99*(1), 154–165.

Wolsey, L. (1998). *Integer programming*. New York: Wiley.

Wolsey, L. A. (2003a). Solving multi-item lot-sizing problems with an MIP solver using classification and reformulation. *Management Science, 48*(12), 1587–1602.

Wolsey, L. A. (2003b). Strong formulations for mixed integer programs: Valid inequalities and extended formulations. *Mathematical Programming, 97*(1), 423–447.

Wolsey, L. A. (2006). Lot-sizing with production and delivery time windows. *Mathematical Programming: Series A, 107*, 471–489.

Woss, W. (1997). A rule-driven generator for variant parts and variant bills of material. In *Proceedings of the 8th International Workshop on Database and Expert Systems Applications (DEXA '97)* (p. 556), Los Alamitos, CA. IEEE Computer Society.

Woudhuysen, J. (2001). E-fulfilment: The opportunities for the future: Part one. *Interactive Marketing, 2*(3), 219–229.

Xia, W., & Wu, Z. (2007). Supplier selection with multiple criteria in volume discount environments. *Omega, 35*, 494–504.

Yadavalli, V. S. S., de W. Van Schoor, C., & Udayabaskaran, S. (2006). A substitutable two-product inventory system with joint-ordering policy and common demand. *Applied Mathematics and Computation, 172*(2), 1257–1271.

Yang, H., & Schrage, L. (2009). Conditions that cause risk pooling to increase inventory. *European Journal of Operational Research, 192*(3), 837–851.

Yano, C., & Gilbert, S. (2004). Coordinated pricing and production/procurement decisions: A review. In A. K. Chakravarty & J. Eliashberg (Eds.), *Managing business interfaces – Marketing and engineering issues in the supply chain and internet domains* (pp. 65–103). Berlin: Springer.

Yao, D., & Zheng, S. (1999). Optimal control of a multi-product inventory system with substitution. In *Proceedings of IEEE 38th Conference on Decision and Control* (pp. 468–473), Phoenix, AZ.

Yao, D. D., & Zheng, S. (2003). Inventory with substitution: Single- and multi-period models. In J. G. Shanthikumar, W. H. M. Zijm, & D. D. Yao (Eds.), *Stochastic modeling and optimization of manufacturing systems and supply chains* (Chap. 8, pp. 177–202). Boston: Kluwer.

Özdemir, D., Yücesan, E., & Herer, Y. (2006a). Multi-location transshipment problem with capacitated transportation. *European Journal of Operational Research, 175*(1), 602–621.

Özdemir, D., Yücesan, E., & Herer, Y. (2006b). Multi-location transshipment problem with capacitated production and lost sales. In *Proceedings of the 2006 Winter Simulation Conference* (pp. 1470–1476).

Zhang, J. (2005). Transshipment and its impact on supply chain members performance. *Management Science, 51*(10), 1534–1539.

Zhang, X., & Chen, J. (2004). Joint replenishment policy for inventory system with demand substitution. In *Proceedings of the Fifth World Congress on Intelligent Control and Automation (WCICA 2004)* (Vol. 4, pp. 2918–2922).

Zhao, L., & Sen, S. (2006). A comparison of sample-path-based simulation-optimization and stochastic decomposition for multi-location transshipment problems. In *Proceedings of the 2006 Winter Simulation Conference* (pp. 238–245).

Zhao, Y., & Klabjan, D. (2005). *Lot-sizing with supplier selection* (Working paper). Urbana, IL: Department of Mathematics, University of Illinois at Urbana-Champaign.

Zhao, Z., Ball, M., & Kotake, M. (2005). Optimization-based available-to-promise with multi-stage resource availability. *Annals of Operations Research, 135*(1), 65–85.

Zhu, X., & Wilhelm, W. (2006). Scheduling and lot sizing with sequence-dependent setup: A literature review. *IIE Transactions, 38*(11), 987–1007.

Index