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 Schneeweß, 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).
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), Drexel 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. 81ff.) and Meyr (1999, p. 11ff.), regarding the classification of planning levels.
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.
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...
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
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
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 fulfill 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

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

Fig. 2.9 Interrelation of flexible BOMs and flexible production sequences

Stages 1
(a) Serial resource structure

Stages 1
(b) Parallel machines

R1 R2 R3
R4 R5 R6
R7

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
2.1 Dynamic Lot-Sizing

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 \( st_{i,k} \) denoting the setup time and \( f_{i,k} \) the setup cost for a changeover from product \( i \) to \( k \):

\[
\begin{align*}
st_{AB} + st_{BC} & \geq st_{AC} \\
f_{AB} + f_{BC} & \geq f_{AC}
\end{align*}
\]  

(2.1)  

(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.
2.1 Dynamic Lot-Sizing

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.
2.1 Dynamic Lot-Sizing

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 imprecise 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

<table>
<thead>
<tr>
<th>macro-periods</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>T=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>micro-periods</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>state changes bound</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td>P2</td>
</tr>
<tr>
<td>state changes within micro-periods</td>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td>P4</td>
</tr>
</tbody>
</table>

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
2.1 Dynamic Lot-Sizing

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 microperiods 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 microperiods 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 fulfillment, 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 prod-
ucts, 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 cor-
crectly 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
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

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

subject to

\[
\begin{align*}
I_t &= I_{t-1} + q_t - d_t & t = 1, \ldots, T \\
I_0 &= 0 \\
qu_t &\leq M \cdot x_t & t = 1, \ldots, T \\
qu_t, I_t &\geq 0 & t = 1, \ldots, T \\
x_t &\in \{0, 1\} & t = 1, \ldots, T
\end{align*}
\]

Equations (2.4)–(2.8) define the variable domains.

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.
2.1 Dynamic Lot-Sizing

- 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:

\[
\text{Minimize } F(q, x, I) = \sum_{i \in P} \sum_{t=1}^{T} (p_i q_{it} + h_t I_{it} + f_t x_{it})
\]  

subject to

\[
I_{it} = I_{i,t-1} + q_{it} - d_{it} \quad i \in P, \ t = 1, \ldots, T
\]  

\[
I_{i0} = 0 \quad i \in P
\]  

\[
\sum_{i \in P} (\kappa_i^P q_{it} + s_t x_{it}) \leq K_t \quad t = 1, \ldots, T
\]  

\[
q_{it} \leq M \cdot x_{it} \quad i \in P, \ t = 1, \ldots, T
\]  

\[
q_{it}, I_{it} \geq 0 \quad i \in P, \ t = 1, \ldots, T
\]  

\[
x_{it} \in \{0, 1\} \quad i \in P, \ t = 1, \ldots, T
\]
Table 2.2 Notations for CLSP

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m )</td>
<td>Number of products</td>
</tr>
<tr>
<td>( T )</td>
<td>Number of periods</td>
</tr>
</tbody>
</table>

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 \( t \)
- \( s_{it} \): Setup time for product \( i \) (in capacity units)
- \( k^p \): 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.
### 2.1 Dynamic Lot-Sizing

- 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

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

\[
\sum_{i \in P} \left( k_i^t q_{it} + \sum_{k \in P} s_{ik} x_{ikt} \right) \leq K_t \quad t = 1, \ldots, T \tag{2.18}
\]
Table 2.3  Notations for CLSD

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

\[
k_i^P q_{it} \leq K_i \left( \sum_{k \in P} x_{ikt} + z_{it} \right) \quad i \in P, \; t = 1, \ldots, T \quad (2.19)
\]
\[
\sum_{i \in P} z_{it} = 1 \quad t = 1, \ldots, T \quad (2.20)
\]
\[
\sum_{i \in P} x_{hit} + z_{lt} = \sum_{k \in P} x_{ikt} + z_{i,t+1} \quad i \in P, \; t = 1, \ldots, T \quad (2.21)
\]
\[
v_{kt} \geq v_{it} + 1 - |P| (1 - x_{ikt}) \quad i, k \in P, \; i \neq k, \; t = 1, \ldots, T \quad (2.22)
\]
\[
q_{it}, I_{it}, v_{it} \geq 0 \quad i \in P, \; t = 1, \ldots, T \quad (2.23)
\]
\[
I_{it} \geq 0 \quad i \in P, \; t = 1, \ldots, T \quad (2.24)
\]
\[
v_{it} \geq 0 \quad i \in P, \; t = 1, \ldots, T \quad (2.25)
\]
\[
x_{ikt} \in \{0, 1\} \quad i, k \in P, \; t = 1, \ldots, T \quad (2.26)
\]
\[
z_{it} \in \{0, 1\} \quad i \in P, \; t = 2, \ldots, T + 1 \quad (2.27)
\]

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.

\[ \text{2.1.5 The General Lot-Sizing and Scheduling Problem} \]

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:

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

subject to

\[
i_{it} = i_{i,t-1} + \sum_{r \in R} \sum_{s \in S_r} q_{ris} - d_{it} \quad i \in P \setminus \{0\}, \; t = 1, \ldots, T \tag{2.29}
\]

\[
\sum_{i \in P} \sum_{s \in S_r} \left( k_{ri}^p q_{ris} + \Psi_{ris} + \sum_{k \in P} s_{ri k} x_{ri k} \right) = K_{ri} \tag{2.30}
\]

\[ r \in R, \; t = 1, \ldots, T \]
Table 2.4 Notations for GLSP

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Constants</strong></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>Number of products</td>
</tr>
<tr>
<td>T</td>
<td>Number of macro-periods</td>
</tr>
<tr>
<td>S</td>
<td>Number of micro-periods</td>
</tr>
<tr>
<td>n_r</td>
<td>Number of resources</td>
</tr>
<tr>
<td><strong>Indices and sets</strong></td>
<td></td>
</tr>
<tr>
<td>i ∈ P = {0, …, m}</td>
<td>Products including dummy product 0</td>
</tr>
<tr>
<td>0 ∈ P</td>
<td>Dummy product for modeling time during which a resource is not set up for any product</td>
</tr>
<tr>
<td>t = 1, …, T</td>
<td>Macro-periods</td>
</tr>
<tr>
<td>s = 1, …, S</td>
<td>Micro-periods</td>
</tr>
<tr>
<td>r ∈ R = {1, …, n_r}</td>
<td>Resources</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td></td>
</tr>
<tr>
<td>d_{it}</td>
<td>Demand for product i in macro-period t</td>
</tr>
<tr>
<td>h_i</td>
<td>Non-negative holding cost per macro-period for storing one unit of product i</td>
</tr>
<tr>
<td>p_{ris}</td>
<td>Unit production cost of product i on resource r in micro-period s</td>
</tr>
<tr>
<td>Ψ_ris</td>
<td>Cost for preserving setup state of product i on resource r in micro-period s</td>
</tr>
<tr>
<td>I_{i0}</td>
<td>Initial inventory of product i</td>
</tr>
<tr>
<td>K_{rt}</td>
<td>Capacity of resource r available in macro-period t</td>
</tr>
<tr>
<td>f_{ik}</td>
<td>Setup cost that is incurred when the setup state of resource r changes from product i to k</td>
</tr>
<tr>
<td>s_{t(rk)}</td>
<td>Setup time for changeover from product i to k on resource r</td>
</tr>
<tr>
<td>C_{ri}</td>
<td>Capacity required for manufacturing one unit of product i on resource r</td>
</tr>
<tr>
<td>m_{ri}</td>
<td>Minimum production quantity for product i after changeover on resource r</td>
</tr>
<tr>
<td>z_{ris}</td>
<td>Binary parameter that indicates whether resource r is set up for i at the beginning of the first micro-period</td>
</tr>
<tr>
<td><strong>Variables</strong></td>
<td></td>
</tr>
<tr>
<td>q_{ris}</td>
<td>Production quantity of product i on resource r in micro-period s</td>
</tr>
<tr>
<td>I_{it}</td>
<td>Inventory of product i at the end of macro-period t</td>
</tr>
<tr>
<td>x_{r(k)is}</td>
<td>(Binary) variable that indicates whether a changeover from product k to product i is performed on resource r in micro-period s</td>
</tr>
<tr>
<td>z_{ris}</td>
<td>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</td>
</tr>
<tr>
<td>Ψ_{ris}</td>
<td>Time during which setup state of product i on resource r is preserved in micro-period s without production (in capacity units)</td>
</tr>
</tbody>
</table>
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 \( r_{r|s} \) 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 \( s \) and thus at the beginning of each micro-period \( s \). Equation (2.34) ensures that the changeover variable \( x_{r|s} \) 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 \( x_{r|s} \) 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>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_i \subseteq P$</td>
<td>Set of successor products of $i$</td>
</tr>
<tr>
<td>$g_{ik}$</td>
<td>Number of units of product $i$ required for producing one unit of its successor product $k$ (Gozinto factor)</td>
</tr>
<tr>
<td>$l_i \geq 1$</td>
<td>Lead time of product $i$ (time unit: number of periods)</td>
</tr>
<tr>
<td>$st_i \geq 1$</td>
<td>Production setup time for product $i$</td>
</tr>
</tbody>
</table>
times are measured in capacity units, whereas lead times are measured in the number of periods.

\[
\text{Minimize } F(q, x, I) = \sum_{j \in P} \sum_{t=1}^{T} \left( h_j I_{jt} + f_j x_{jt} \right) \quad (2.39)
\]

subject to

\[
I_{it} = I_{i,t-1} + q_{it} - d_{it} - \sum_{k \in P_+} g_{ik} q_{k,t+l_i} \quad i \in P, \ t = 1, \ldots, T - l_i \quad (2.40)
\]

\[
I_{it} = I_{i,t-1} + q_{it} - d_{it} \quad i \in P, \ t = T - l_i + 1, \ldots, T \quad (2.41)
\]

\[
\sum_{i \in P_i} \left( \nu_i^p q_{it} + s_i x_{it} \right) \leq K_{rt} \quad r \in R, \ t = 1, \ldots, T \quad (2.42)
\]

\[
q_{it} \leq M_x x_{it} \quad i \in P, \ t = 1, \ldots, T \quad (2.43)
\]

\[
q_{it}, I_{it} \geq 0 \quad i \in P, \ t = 1, \ldots, T \quad (2.44)
\]

\[
x_{it} \in \{0, 1\} \quad i \in P, \ t = 1, \ldots, T \quad (2.45)
\]

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.
2.1 Dynamic Lot-Sizing

- 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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Indices and sets</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>e(t)</strong></td>
<td>Last micro-period of macro-period t</td>
</tr>
<tr>
<td><strong>Φ</strong></td>
<td>Set of micro-periods</td>
</tr>
<tr>
<td><strong>Φ\text{fix}</strong></td>
<td>Set of micro-periods with fixed beginnings</td>
</tr>
<tr>
<td><strong>Φ^\text{r}</strong></td>
<td>Set of micro-periods during which production on r is forbidden</td>
</tr>
<tr>
<td><strong>Φ^l</strong></td>
<td>Set of micro-periods that are the last micro-period in a macro-period</td>
</tr>
<tr>
<td></td>
<td><strong>Parameters</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>d_{is}</strong></td>
<td>Demand for product i in micro-period s ((= 0 ) for all ( s \in Φ\setminusΦ^l ))</td>
</tr>
<tr>
<td><strong>I_{imax}</strong></td>
<td>Upper inventory limit for product i</td>
</tr>
<tr>
<td><strong>w_s</strong></td>
<td>Fixed beginning for micro-period s</td>
</tr>
<tr>
<td><strong>I_{w_s}^{r, max}</strong></td>
<td>Upper inventory limit for units of product i produced on resource r that are not consumed before the next micro-period</td>
</tr>
<tr>
<td><strong>z_{is1} := x_{ri1}</strong></td>
<td>Binary parameter that indicates whether resource r is set up for i at the beginning of the first micro-period</td>
</tr>
<tr>
<td></td>
<td><strong>Variables</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>x_{riks}</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>z_{is}</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>I_{is}</strong></td>
<td>Inventory of product i at the end of micro-period s</td>
</tr>
<tr>
<td><strong>q_{is}^{r}</strong></td>
<td>Quantity of product i produced on r in micro-period s available for consumption in the same period</td>
</tr>
<tr>
<td><strong>q_{is}^{r+1}</strong></td>
<td>Quantity of product i produced on r in micro-period s available for consumption in the next period s + 1</td>
</tr>
<tr>
<td><strong>w_s</strong></td>
<td>Beginning of micro-period s (on continuous time axis)</td>
</tr>
<tr>
<td><strong>y_{irs}^{r,1}</strong></td>
<td>Setup time consumed on resource r at beginning of micro-period s</td>
</tr>
<tr>
<td><strong>y_{irs}^{r,s}</strong></td>
<td>Setup time consumed on resource r at end of micro-period s</td>
</tr>
</tbody>
</table>

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 \( I_{is} \) 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 \( x_{rik, s} \) 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).
2.1 Dynamic Lot-Sizing

Minimize $F(q, x, z, I, \Psi, q^0, q^{+1}, w, y^b, y^c)$

$$
= \sum_{i \in P} \sum_{t=1}^T h_i \left( I_{i,s(t)} + \sum_{r \in R} q_{r,i,s(t)}^{+1} \right) \\
+ \sum_{i \in P} \sum_{r \in R} \sum_{s=1}^S \left( p_{ris} q_{ris} + p_{ris} \Psi r_{ris} + \sum_{k \in P} f_{rik} x_{rik} \right) 
$$

(2.46)

subject to

$$
I_{is} = I_{is-1} + \sum_{r \in R} (q_{ris}^b + q_{ris}^{+1}) \\
- d_{is} = \sum_{r \in R} \sum_{k \in P} g_{ikh} q_{eks} i \in P \setminus \{0\}, s = 1, \ldots, S 
$$

(2.47)

$$
I_{is} \leq I_{is}^{\text{max}} \\
w_s = \overline{w}_s \\
w_{s+1} = w_s \\
= y^b_{rs} + \sum_{i \in P} (\Psi_{ris} + k^b_{ri} q_{ris}) + y^c_{rs} r \in R, s = 1, \ldots, S 
$$

(2.50)

$$
q_{ris} = 0 \\
q_{ris} \leq M \cdot z_{ris} \\
q_{ris} \geq m_{ri} \sum_{k \in P^{\backslash\{i\}}} x_{rik} \\
\sum_{i \in P} z_{ris} = 1 \\
x_{rik} \geq z_{ri,s-1} + z_{eks} - 1 \\
y^b_{rs} + y^b_{s-1} = \sum_{i,k \in P} s_{rik} x_{rik} \\
q^{0}_{ris} + q^{+1}_{ris} = q_{ris} \\
q^{+1}_{ris} \leq q_{ris}^{\text{max}} \\
\sum_{i,k \in P} x_{rik} = 1 \\
w_s - w_{s-1} \geq y^b_{rs} - 1 \\
+ k^b_{ri} q_{rik} + y^c_{rs-1} \\
w_s - w_{s-1} \geq y^b_{rs} - 1 \\
+ k^b_{rik} q_{rik} + y^c_{rs} - 1 \\
g_{ik} > 0, g_{ik} k^p_{ri} > k^p_{rik} \\
g_{ik} > 0, g_{ik} k^p_{ri} < k^p_{rik}
$$

(2.51) (2.52) (2.53) (2.54) (2.55) (2.56) (2.57) (2.58) (2.59) (2.60) (2.61)
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 that originates from production in the same micro-period and a part that originates from production in the previous micro-period. 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 \( q_{ris}^0 \) 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 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 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 (\( y_{rs}^b \) and \( y_{rs}^e \), respectively). The time during which the setup state is preserved without production is captured explicitly by the \( y_{ris} \) 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 $x_{rik}$ 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 $y^{r}_{s,s-1}$ of the setup time is scheduled at the end of $s-1$, the remaining part $y_{s,s}^b$ at the beginning of $s$. Note that all $y_{s,s}^b$ and $y_{s,s}^e$ should be fixed to 0. The splitting of production quantities into a part $q_{r}^{b}$ that can be used in the same micro-period $s$ where the production takes place and another part $q_{r}^{e}$ 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 $x_{riki} = 1$, 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 $g_{ik}^{p}k_{rij} > k_{rijk}^{p}$, i.e., the production time

![Diagram](image-url)

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 \( y_{r2,s-1}^i \) 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 \( q_{r1,s-1}^i \) 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 \( s-1 \). Here, only the production \( q_{r1,s-1}^i \) 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 \( y_{r2,s-1}^i \) 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 \( s-1 \).

---

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 \( w_s - w_{s-1} \geq y_{r1,s-1}^i + y_{r2,s-1}^i \), 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 \( y^h \) and \( y^j \) variables. These variables \( y_{r1}^{ij} \) and \( y_{r2}^{ij} \) 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 \( y_{r1}^{ij} \) and \( y_{r2}^{ij} \) mentioned in footnote 5 (F. Seeanner 2009, personal communication). A schedule with the property enforced by (2.61) is shown in Fig. 2.13b.
2.1 Dynamic Lot-Sizing

(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 MLCLSP (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 \( g_{ik} \) induced by a
successor product \( k \in P_i^t \) 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 \( q_{rs}^0 \) and \( q_{rs}^{+1} \) 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 \( \bar{w}_s \) and \( \bar{w}_{s+1} \), adding them to the set \( \Phi^{fix} \), and adding \( s \) to the set \( \Phi^p \).

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
2.2 Solution Techniques for Dynamic Lot-Sizing

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)\footnote{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.}
- 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)\footnote{For additional literature see, e.g., Diaby et al. (1992a,b); Thizy and van Wassenhove (1985).}
- 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.
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):

\[
\sum_{t \in S} q_t \leq \sum_{l \in S} \sum_{t' = t}^T d_{t'} x_{t'} + I_l \quad l = 1, \ldots, T, \ S \subseteq \{1, \ldots, l\} \tag{2.70}
\]

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:

\[
q_t \leq d_l x_t + I_l \quad t = 1, \ldots, T \tag{2.71}
\]

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

\[
I_{t-1} \geq d_t (1 - x_t) \quad t = 1, \ldots, T \tag{2.72}
\]

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):

\[
I_{t-1} \geq \sum_{t = 1}^k d_t \left( 1 - \sum_{u = t}^T x_u \right) \quad t = 1, \ldots, T, k = t, \ldots, T \tag{2.73}
\]

### 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:

\[
\text{Minimize } F(x, y) = \sum_{t_1 = 1}^T \sum_{t_2 = t_1}^T \left( f_{t_1, t_2} x_{t_1} + \sum_{t_3 = t_1}^T c_{t_1, t_3} y_{t_1, t_3} \right) \tag{2.74}
\]
### Table 2.7 Additional notations for SPL-based reformulation of CLSP

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td></td>
</tr>
<tr>
<td>$c_{i,t_d}$</td>
<td>Costs per unit to satisfy demand for product $i$ in period $t_d$ from a production lot in period $t_s$</td>
</tr>
<tr>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td>$y_{i,t_d}$</td>
<td>Demand quantity for product $i$ in period $t_d$ that is satisfied from a production lot in period $t_s$</td>
</tr>
</tbody>
</table>

Subject to

$$\sum_{i=1}^{I} y_{i,t_d} = d_{t_d} \quad i \in P, \ t_d = 1, \ldots, T$$

$$\sum_{i=1}^{I} y_{i,t_d} \leq \sum_{i=1}^{I} d_{i,t} \cdot x_{i,t} \quad i \in P, \ t_s = 1, \ldots, T$$

$$y_{i,t_d} \leq d_{i,t} \cdot x_{i,t} \quad i \in P, \ 1 \leq t_s \leq t_d \leq T \quad (2.75)$$

$$\sum_{i \in P} k_{i}^{P} \sum_{i=1}^{I} y_{i,t_d} \leq K_{t_s} \quad t_s = 1, \ldots, T$$

$$x_{i,t_s} \in \{0, 1\} \quad i \in P, \ t_s = 1, \ldots, T$$

$$y_{i,t_d} \in \{0, 1\} \quad i \in P, \ 1 \leq t_s \leq t_d \leq T$$

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

$$c_{i,t_d} = p_{i,t_s} + (t_d - t_s) h_i \quad (2.76)$$

The lot-size variables $q_{i,t}$ and inventory variables $I_{i,t}$ 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 $x_{i,t_s}$ 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 $x_{i,t_s}$ 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 $y_{i,t_d}$ variables.
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)
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, \( P_3 \) 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. 2.15 Example – Relax&Fix
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 heterogeneous parallel machines, setup times, backlogging, minimum lot-sizes and safety stock level violations. 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
The generation of MILP subproblems that are solved sequentially is performed using three types of decompositions:

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 a 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.
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 formulation 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.
2.3 Transshipment Problems

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. heterogeneous
- **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 fulfillment**: All-or-nothing vs. partial order fulfillment 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.
2.3 Transshipment Problems

- 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 $\tau$-heuristic transships in order of increasing transshipment cost from locations from which a transshipment would be optimal in
the isolated two-location system. The $\alpha$-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 $\omega$-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) focuses 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 $\delta$ 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 heterogeneous 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).

Simon 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.
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 $\omega$ 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:

\[
x^* = \arg\min_{x \in X} J(x)
\]  

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.
2.4 Solution Techniques for Transshipment Problems

- **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.
“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 than 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. 577ff.). 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 \( \text{Var}(R_1 - R_2) = \text{Var}(R_1) + \text{Var}(R_2) - 2 \cdot \text{Cov}(R_1, R_2) \).

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 \( \text{Cov}(R_1, R_2) \) 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": 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., $f(x'_i) < f(x_k)$. 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.
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 \( \delta_k \) and only finds one better solution \( x'_1 \) by the move (“right”) that increases variable 1. This solution is assigned to \( x_{k+1} \), and the new step size \( \delta_{k+1} \) is obtained by doubling \( \delta_k \).

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 $\mu$ and $\sigma^2$)
- 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.
Production and Inventory Management with Substitutions
Lang, J.C.
2010, XX, 260 p. 56 illus., Softcover
ISBN: 978-3-642-04246-1