Università degli Studi di Roma Tre

Dipartimento di Informatica e Automazione

Via della Vasca Navale, 79 – 00146 Roma, Italy

Models and methods for production scheduling in the pharmaceutical industry

Dario Pacciarelli[1], Carlo Meloni[2], Marco Pranzo[3]

RT-DIA-129-2008 **Giugno 2008**

(1) Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, via della vasca navale, 79 - 00146 Roma, Italy.

(2) Dipartimento di Elettrotecnica ed Elettronica, Politecnico di Bari, via E. Orabona, 4 - 70125 Bari, Italy.

(3) Dipartimento di Ingegneria dell'Informazione, Università di Siena, via Roma, 56 - 53100 Siena, Italy.

This work is partially supported by the Italian Ministry of Research, Grant number RBIP06BZW8, project FIRB "Advanced tracking system in intermodal freight transportation"

ABSTRACT

In this report we study models and methods for production scheduling, and discuss the relationships between scheduling and planning activities with special reference to pharmaceutical manufacturing systems. In common industrial practice, planning and scheduling interact as a loosely coordinated control chain in which the plan determines input data and constraints to be satisfied by the schedule. We report on a practical case study showing that closer coordination can be attained using fast and effective automated scheduling tools. Such tools can be used in the planning activity to generate reliable estimates of the shop floor capacity and thus to produce better production plans. The state of the art and new research directions in this context are discussed.

1 Introduction

The pharmaceutical marketplace is dominated by large multinational companies, competing worldwide, with a global presence in branded products. Retaining marketshare requires standards of product quality and reliability close to 100%, attained at sustainable cost. Wholesalers and final customers expect reliability and quality from pharmaceutical companies, which face an increasing challenge to achieve such standards. Reliable production plans are critical to this aim. In fact, in order to achieve 100% availability of final products, it is not sufficient to attain excellence in each phase of the planning process from strategic planning to real time scheduling, but it is also important to effectively manage the coordination between these different phases.

As observed by several authors, e.g., McKay and Wiers (2003) [55, 56], many computerized tools for planning and scheduling have had limited success in practice. Possible reasons include the lack of flexibility of planning and scheduling systems [53, 54], as well as the mismatch between the key aspects of the scheduling problem to be solved and the system implemented in practice, which tends to reflect more the perceived than the real problem.

Kempf et al. (2000) [43] present a lucid overview of the many issues involved in assessing the quality of a production schedule. Most of them are also addressed in this chapter, in order to develop effective models and methods for production scheduling and to investigate the interaction between planning and scheduling functions. The latter topic is very under-researched in the academic literature, but is critical to the successful implementation of planning activities in pharmaceutical industry. To this aim, we use the generic term *planning activity* to denote any activity from demand management to production planning, except finite capacity scheduling, which is called the *scheduling activity*. In a commonly adopted classification, the planning function determines the quantities of products and components to be produced in a given time horizon, while the scheduling function involves the release

of work orders to the shop floor, specifying the allocation of shop resources to the different orders over time [71, 87, 89]. Following the notation of [43], a *predictive schedule* describes the designed system behavior over the schedule horizon, while a *historical schedule* describes the operation start times and resource assignment actually executed on the shop floor. The usual way planning and scheduling interact in the pharmaceutical industry, at least in the short-term, is as an open loop control chain, in which the plan determines input data and constraints to be satisfied by the predictive schedule.

To a large extent, the customer-facing end of the pharmaceutical supply chain is driven by orders. Demand management is usually based on historical data and statistical considerations, and planning decisions are driven by the need to meet market demand dates. Allocating a product less production capacity than needed may result in significant lost revenue (see, e.g. Mallik *et al.* (2002) [50]). On the other hand, scheduling decisions are driven by the availability of shop floor resources. Hence, when the quantities required by the plan are excessive relative to the available capacity, the schedule cannot comply with the plan, causing delays in the delivery of final products. However, wholesaler orders are typically the result of a negotiation in which order quantities, due dates and penalties for late delivery are regulated by contracts [61, 75]. Hence, the reliability of the whole process is closely related to the ability to negotiate realistic delivery dates for all orders or, alternatively, evaluating the impact of a new order on the delivery dates of those currently in the system. In other words, managing the demand based on historical data cannot guarantee 100% reliability, which requires closing the control loop from production scheduling to the sales department to provide real time information on the future production capacity which is available for new demand. Effective tools are needed for this purpose that can generate and maintain effective predictive schedules and automatically evaluate alternative schedules as new market scenarios materialize.

Producing reliable estimates of future capacity in the pharmaceutical industry is not an easy task. One reason is the complexity of manufacturing processes and

the impact of contamination issues on the factory throughput. Another reason is related to the issue of on-time delivery. It is generally recognized that in the vast majority of manufacturing systems the two main objectives to be pursued in production planning and scheduling are on-time delivery of the final products and maximization of the total value produced by the plant. These two objectives are often conflicting, since the former hampers the organization of production schedules with large lots, increasing the number of setups and idle time and reducing the factory throughput. Relative to other manufacturing processes, the pharmaceutical industry gives higher importance to on-time delivery over throughput maximization, due to the economical and legal implications of late deliveries and stock-outs at the final customers [39, 47, 59, 74].

The above facts have significant consequences for both real time operations management and planning activities. On the real time side, the complexity of production processes and the need for meeting short-term due dates lead to frequent under-utilization of shop resources and disregarding of long-term due dates. On the planning side, the lack of reliable information on the actual plant capacity makes estimating reliable delivery dates to negotiate with the wholesalers difficult, thus causing frequent urgent orders, which further increase the short-term pressure on the operations managers [7, 14]. The best utilization of production resources and on-time delivery of the final products can be achieved by using scheduling support systems both in real time, to organize the short-term production, and in the planning phase, to carefully estimate actual resource capacity and reliable delivery dates for final products [6, 15, 71]. This is the subject of this chapter.

The following sections are organized as follows. In Section 2 we briefly review the most common aspects of pharmaceutical production processes that are relevant to planning/scheduling activities. We do not deal here with primary pharmaceutical production, in which active ingredients and other basic components of final products are produced, but rather focus on secondary pharmaceutical production, in which the components are dispensed, blended, processed and packed. Secondary

production is more critical from the viewpoint of coordination between planning and scheduling. In Section 3 we discuss several optimization models that can be used to formulate scheduling problems arising in secondary pharmaceutical production.

Section 4 deals with solution techniques. To a large extent, scheduling and related issues are still carried out by human schedulers, who develop feasible predictive schedules based on their past experience and intuition. The development and implementation of effective computerized systems for such operational problems require paying attention to a number of aspects that are rarely taken into account simultaneously in scheduling theory or in practice. The complexity and variety of constraints arising from the shop floor characteristics and from organizational issues require powerful scheduling algorithms. The quality of a schedule may typically involve several indices, such as the utilization level of shop floor resources, production costs in terms of personnel and energy consumption, the fulfillment of production targets and others. More importantly, resources and scheduling priorities change over time, which necessitates adapting the scheduling algorithms continuously to the new situation. Finally, no decision maker would unconditionally accept a predictive schedule that they do not consider convincing. Therefore, at least in the transition from manual to computerized scheduling, it is necessary for the computer to generate schedules akin to those produced manually.

From the above issues, it follows that effective computerized scheduling algorithms must be modular, easy to implement and to maintain, able to deal with many constraints and objective functions, and able to incorporate observations and suggestions arising from the scheduling practice, i.e., to incorporate user intuition and experience in the scheduling system. In Section 4 we discuss these issues and describe several algorithmic techniques that can be successfully applied to face such challenge.

In Section 5 we report on a practical experience in three departments of a pharmaceutical company. We chose two small, non-critical departments and a typical complex manufacturing department that allow us to describe in sufficient detail the

whole process of modeling the scheduling problems, developing solution algorithms and discussing results and feedback from the field. Some conclusions and directions for future work follow in Section 6.

2 Pharmaceutical manufacturing systems

In this section we briefly review those characteristics of pharmaceutical production processes that are most relevant to the planning/scheduling decision activities.

Typical pharmaceutical supply chains contain at least two manufacturing stages: primary and secondary manufacturing [23, 83]. The former is dedicated to the production of active ingredients and other basic components through complex chemical and biochemical processes. Production is typically a *push* process organized in long campaigns, driven by forecast demand, to reduce the impact of long cleaning and setup times that are necessary to ensure quality and avoid cross-contamination. Primary manufacturing is therefore not very sensitive to short-term demand fluctuation, and the main issue here is careful lot sizing to avoid shortages of active ingredients.

Secondary manufacturing is usually a *pull* process, driven by wholesaler orders, in which active ingredients and other components are dispensed, blended, processed and packed to produce the final products. Primary and secondary manufacturing are typically decoupled by relatively large stocks of components. Since, in general, secondary production is very sensitive to market demand, scheduling and coordination between planning and scheduling are critical issues to guarantee 100% availability of final products at sustainable costs.

In the remainder of this chapter we focus on secondary manufacturing only, which is the most relevant stage as far as the coordination between planning and scheduling is concerned.

Secondary pharmaceutical manufacturing systems consist of a set of multi-purpose production facilities that produce a variety of intermediate and finished products

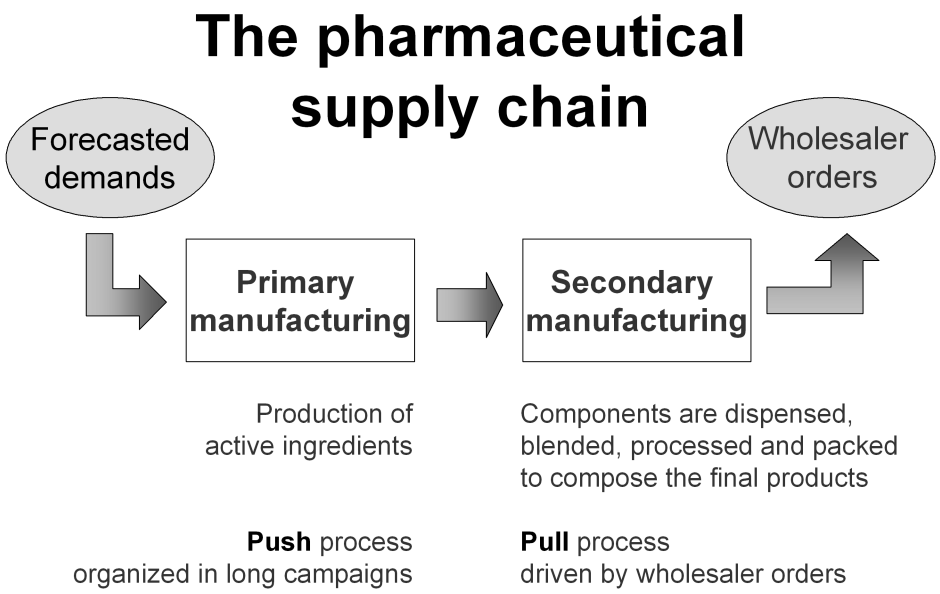

Figure 1: The pharmaceutical supply chain.

through multi-stage production processes. Facilities are linked by supplier-customer relations, i.e., one facility produces intermediate goods that are processed further by other facilities, reflecting the material flow relationships given by the recipes of the final products. Furthermore, each facility may interact with external (e.g. suppliers) and/or internal (e.g. warehouses) entities.

Common production processes are devoted to producing solid, liquid, aerosol or powdered items according to a family of similar recipes [23]. For example, one of the main process families is devoted to Solid Dosage Manufacturing (SDM), which includes, among others, the production of tablets. Each common production process consists of a set of self-contained activities. For example, SDM includes:

1. raw material handling, in which the availability of all materials required by the receipt is checked;
2. dispensing activities, in which materials are weighed according to the batch recipes and stored in sealed bins;
3. binder preparation, where specific agents are prepared to be used to wet powders in the granulation;

4. granulation, where dry materials from the bins are passed into a granulator and mixed with specific quantities of binder agents to produce granules. This is passed through a wet mill into a fluid bed drier. After drying the granule is transferred in a handling bin;
5. blending, where bin contents are blended. This can occur before or after the granulation and may include the addition of new materials;
6. compression, where granules are fed into tablet presses;
7. coating, in which a solution is prepared and sprayed over the tablets;
8. counting, where packages are prepared according to the different orders and market places;
9. packaging, in which tablets and packages are transferred to the packaging line and processed. It includes individual specific packages and bulk packs when the products are sent to other sites for further processing;
10. quality assurance and control activities are distributed throughout the whole the manufacturing process.

Between coating and packaging, tablets are typically stored in sealed bins in order to decouple the two activities. A similar separation occurs between dispensing and granulation. Hence, the production is typically organized with three or four main departments, as in Figure 2, which to a certain extent can operate independently of each other. The first two activities are performed in the *dispensing* department, activities from binder preparation to coating are performed in the *manufacturing* department, and counting and packaging are performed in the *packaging* department. When counting activities require a significant amount of work, *counting* and packaging are performed in different departments. This is a common situation in Europe, where the many different national regulations and languages require the handling of a huge number of different packages in the same plant [23, 73, 83].

The typical layout of a dispensing department consists of one or more rooms connected to a warehouse, in which raw materials are weighed. Cross-contamination issues require processing of one product at a time in each room and cleaning the room

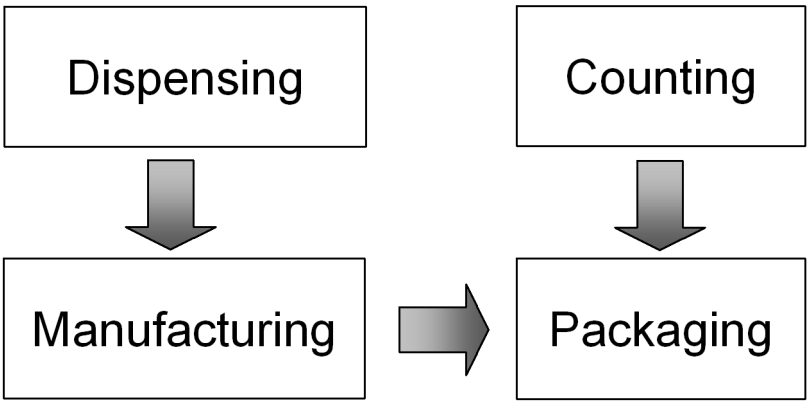

Figure 2: The typical layout of a secondary pharmaceutical manufacturing plant.

when switching from one product to another. Minor cleaning is sufficient when two consecutive products need the same raw materials, while major cleaning is necessary when the raw materials change. Hence, the production is organized in groups of products requiring the same raw materials to be scheduled consecutively, which are called *campaigns*. Major cleaning is however necessary after a maximum number of products of the same type, called the *size of a campaign*. Hence, from the scheduling point of view, each room acts as a single machine with sequence-dependent setup time and campaigns. The layout of a counting department is very similar to that of a dispensing department, but its production process is characterized by the absence of significant setups and by a larger number of lots per week.

The layout of a typical manufacturing department is a complex flow shop with a number of specialized machines (e.g. mixers, reactors, dryers, etc.), in which each product is processed on a subset of machines only. Even though there may be different copies of the same machine, production is often organized by assigning a specific machine to each product type in order to reduce cross-contamination risks. Hence routing flexibility is very limited in practice. As in the dispensing department, the production is organized in campaigns to reduce setups.

Packaging departments usually contain one or more packaging lines, which can process one product at a time. Hence, this department can be viewed as a set

of parallel machines. The number of lots per week processed by the packaging department is usually much larger than for the manufacturing department. In fact, for example, a single lot of tablets can be divided into many different lots of final products, differing from each other only in the package.

Planning and scheduling activities are usually loosely integrated in secondary manufacturing industrial practice. In their simplest form, planning and scheduling activities strive to meet delivery dates for all demands subject to constraints on sequencing, resource capacity, and process production [5, 6, 22].

Sequencing constraints specify a partial ordering among the operations for a set of tasks. These constraints are typically dictated by each specific recipe, but sequencing constraints may be required also by the production process, by quality control or by specific management policies. For example, in some cases a task can be started only after some equipment preventive maintenance or calibration task is finished.

Resource constraints limit the number of resources that can be employed for a specific activity at any time. For example, manpower resources are categorized by their respective skills and low skilled operators can perform only a subset of simple tasks. This fact may limit, for example, the number of tool changes that can be performed simultaneously in a shift, when these activities involve complex mechanical operations.

Besides such issues, pharmaceutical production processes are characterized by extremely strict requirements of chemical composition and operations execution to guarantee the quality of the final products. Process constraints are dictated by the production process established in the plant. These constraints generally aim to avoid the risk of cross-contamination between different products, or between different lots of the same product. Good manufacturing practices prescribe cleaning equipment before, and sometimes after, processing a new material. Therefore, setup times can be large with respect to the processing time of a lot.

Some features require incorporation of a number of relevant details into the

scheduling models. For example, when switching from a product to another, it may be necessary to clean a machine and to change some tools, which involve mechanical operations. While cleaning operations can be performed by low skilled operators, mechanical operations require specialized operators. Moreover, different mechanical operations can be performed in parallel, and therefore a careful model of the setup time should take into account the number of operators involved and their respective skills, besides the possible need for minor/major cleaning.

Constraints may derive from specific plant management policies. For example, a department may prefer to organize production such that the starting/completion time of some particular operation is aligned with the beginning or end of a shift. Other departments may constrain each setup operation to be completely executed within the same shift, although there are no specific technological reasons for such requirement.

A good scheduling decision support system (DSS) in this context should provide a feasible predictive schedule which simultaneously ensures that:

1. enough material at each bill of materials level is produced to satisfy the demands;
2. early warnings of shortages of intermediates and raw materials are provided;
3. all production constraints, resource availabilities, and business requirements are respected;
4. good solutions in a practical sense with regard to the multiple conflicting objectives are obtained.

Besides these main requirements, the DSS should enable management to:

1. reduce the amount of work of the human scheduler;
2. permit every department to operate independently to a certain extent, while collaborating with the others for increased transparency and information sharing;
3. reduce response time to changes;
4. monitor and manage conflicts among departments.

3 Optimization models

The construction of optimization models for planning and scheduling purposes is necessarily the result of an iterative process, during which all different aspects of the problem are taken into account and formalized progressively with the agreement of all stakeholders. It is possible to distinguish at least two different methods for formally defining a problem. *Descriptive models* provide a formal way to list the relevant aspects of the problem, while *formulations* explicitly define all the variables of the problem as well as their logical and mathematical relations.

While descriptive models are more suitable for defining the problem in a preliminary phase, when interacting with the stakeholders, formulations enable the construction of solution algorithms. In other words, descriptive models are necessary to define all the aspects of the problem to be solved, and formulations are necessary to design solution methods.

In the next subsection we give a brief description of the most frequent aspects arising in pharmaceutical industries using a descriptive model. In the subsequent subsection, we introduce some formulations that have been shown to be particularly effective in solving practical production scheduling problems.

3.1 Descriptive models

A common classification scheme for scheduling problems focuses on the characteristics of the storage facilities. From this point of view, at least four relevant settings can be distinguished (see, for example [84]): Unlimited Intermediate Storage (UIS), Finite Intermediate Storage (FIS), No Intermediate Storage (NIS), and Zero Wait (ZW). Hall and Sriskandarajah [35] model the absence of intermediate buffers (NIS) as a *blocking* constraint. In this case a job, having completed processing on a machine, remains on it until the next machine becomes available for processing. The FIS case can be viewed as NIS problem by modeling each position of an intermediate buffer as a dummy machine with zero processing time [52]. In ZW settings a job,

having completed processing on a machine, must immediately start processing on the subsequent machine without waiting.

Other relevant settings are related to scheduling problems with setups (see, e.g. the extensive review of Allahverdi *et al.*, 1999, [4]) and to the management of perishable items. A commodity is said to be perishable if some of its characteristics are subject to deterioration over time with respect to customer/producer requirements. The perishability issue is approached in various ways in the scheduling literature. A common approach when scheduling perishable goods with high decay rate consists of introducing tight zero wait constraints in order to avoid product degradation (see for example [34, 35, 70]).

A more detailed descriptive model for scheduling problems is the well known $\alpha/\beta/\gamma$ classification scheme of Graham *et al.* (1979) [33]. With this notation, α indicates the scheduling environment, β describes the job characteristics or restrictive requirements, and γ defines the objective function to be minimized.

3.2 Problems Formulations

In general, most production scheduling problems can be formulated as mixed integer linear programs by defining suitable variables and constraints. However, as observed by Jain and Meeran (1999) [41] and Pinedo (1995) [70], mathematical programming techniques have so far been of limited practical use, at least for job shop scheduling problems of practical size. Their observation is particularly valid for complex production scheduling problems in which a solution has to be found within strict time limits, or when the models can change due to re-configurations of the production environment, rendering the mathematical properties that can be exploited to speed up solution algorithms less and less effective [1, 20].

A different approach to the formulation of scheduling problems is based on the observation that the basic variables to be considered are the starting times of the operations to be scheduled, while the constraints can be viewed as time relations among

pairs of variables. Within this stream of research, graph models have been demonstrated to be particularly effective for modeling and solving very general scheduling problems.

The first successful model in this context is the disjunctive graph formulation for the job shop scheduling problem proposed by Roy and Sussman (1964) [77]. The job shop scheduling problem is the problem of allocating machines to competing jobs over time, subject to the constraint that each machine can handle at most one job at a time. With this first formulation, a job consists of a sequence of operations, each to be processed without interruption on a given machine. This version of the job shop scheduling problem is clearly a simplification of many relevant problems that can be encountered in the industrial practice.

The disjunctive graph does not take into account a number of additional constraints arising from scheduling practice. However, several authors have observed that it can be easily adapted to deal with many practical issues (see, for example, White and Rogers, 1990 [92], and Schutten, 1998 [82]) such as assembly and disassembly sequences, setups, due dates, release times, maintenance operations, material handling delays and other operational side constraints. Different graph formulations have been proposed in the last two decades to overcome the limitations that remain in these models. Mc Cormick *et al.* [52] introduce the so-called Precedence Constraints Graph to study a flow shop scheduling problem in an assembly line with finite capacity buffers between machines (FIS). They model the positions of the intermediate buffers as machines with zero processing time and show that, once a sequence for the jobs has been found, the starting time of the jobs on all machines can be easily computed on a Precedence Constraints Graph. Sanmartí *et al.* [81] introduce the *Schedule Graph* representation to study a production scheduling problem arising in a multipurpose batch plant. The Schedule Graph generalizes the Precedence Constraints Graph to represent general job shop problems with both Unlimited Intermediate Storage (UIS) and Non Intermediate Storage (NIS) situations. Romero *et al.* [76] combine the Schedule Graph representation, or S-graph,

with a feasibility check based on linear programming to deal with different intermediate storage policies, including zero wait (ZW), Finite Intermediate Storage (FIS) and Common Intermediate Storage (CIS). Mascis and Pacciarelli [51] introduce the *Alternative Graph* model, which generalizes the previous formulations. An advantage of the Alternative Graph with respect to the Schedule Graph is that zero wait and perishability constraints can be represented directly within the graph, without the need for additional feasibility checks. This compact representation enables the development of more effective optimization procedures, and it has been successfully applied to the solution of complex production scheduling problems [58, 67, 68, 72].

A different stream of research aims at formulating integrated routing and scheduling problems. The most common production environment in this context is the parallel machines case, when the problem is to decide when to start an operation and on which machine. Examples in this context can be found, for example, in Ovacik and Uzsoy (1997) [66]. This problem has also been extensively studied in the form of a constraint satisfaction problem by several researchers from the field of Artificial Intelligence [80]. Finally, the field of Resource Constrained Project Scheduling investigates the case occurring when the operations may require a general amount of some production resources, which are available in a given quantity [18].

4 Solution methods

In this section we describe a general algorithmic approach to design scheduling algorithms that are easily adaptable, modular and suitable for incorporating human experience in the computerized methods. In these methods, the search process is performed by several heuristic procedures guided by a general optimization strategy. These features are particularly useful when facing practical problems, where models suffer from a lack of strong mathematical properties that can be used to design effective solution algorithms. Moreover, simple heuristics are more easily accepted and trusted by the human schedulers, who can understand their principles and

suggest modifications to improve their performance over time.

In this section we describe the algorithmic building blocks: constructive algorithms, mainly used to obtain initial feasible predictive schedules, and iterative algorithms, which can be used to improve the initial solution. Subsection 4.1 focuses on greedy methods, which can be used to find a feasible solution for a large variety of practical scheduling problems. Subsection 4.2 describes simple and general local search techniques. Subsection 4.3 deals with strategies to improve the performance of a heuristic or to combine different algorithms. Finally, Subsection 4.4 illustrates methods for improving the effectiveness of simple local search strategies.

4.1 Constructive algorithms

Among simple constructive techniques for scheduling problems we focus on greedy algorithms such as *list scheduling* and *insertion* algorithms.

List scheduling algorithms typically sort the operations to be performed according to a given criterion and then, starting from an empty schedule, build a complete predictive schedule by adding one operation at a time to the end of the partial schedule, in the order induced by the sorting criterion. When routing and scheduling decisions have to be made simultaneously, as in the parallel machines environment, the added operation is assigned to a machine according to a given dispatching rule. The sorting criterion and the dispatching rule specify the list scheduling algorithm. Several studies of greedy algorithms can be found in [11, 13, 21, 28, 38, 62, 69], among others.

Sorting criteria can be broadly classified according to different dimensions, such as static or dynamic, deterministic or randomized criteria and more. A static criterion sorts the operations according to information available in the initialization phase of the algorithm. A dynamic criterion sorts the operations according to information that changes during the run of the algorithm, thus requiring the recalculation of the criterion for each unscheduled operation every time an operation is added to the

partial schedule. Deterministic criteria always select the next operation according to a certain sorting criterion, whereas stochastic criteria may choose randomly from a subset of candidate operations.

More sophisticated constructive algorithms are the so-called insertion algorithms. An insertion algorithm repeatedly selects a promising operation and then chooses its position in the partial schedule, according to a given policy. A common choice consists of tentatively inserting the selected operation in all the possible feasible positions in the partial schedule, and choosing the locally best position. Insertion algorithms are among the best performing simple constructive algorithms for scheduling problems [63, 64], but typically require larger computation times than list scheduling algorithms.

4.2 Iterative algorithms

Iterative algorithms are based on the concepts of move and neighborhood. A *move* is a rule for transforming a solution s into a different solution s'. A solution s' that can be reached from s using a single move is called a *neighbor* of s. The set of all the neighbors of s is called the *neighborhood* of s. A *neighborhood structure* $\mathcal{N}$ associated with a given move defines, for each solution s, its neighborhood $\mathcal{N}(s)$.

Given an initial solution s, there are three basic elements that compose an iterative algorithm: (i) the neighborhood structure $\mathcal{N}$, (ii) a procedure for selecting a new solution in $\mathcal{N}(s)$, called the selection criterion, (iii) a method for terminating the procedure, called the stopping criterion.

Common neighborhood structures for scheduling problems are based on three kind of moves, called *swap*, *exchange* and *insert* [16, 17, 91]. The swap move consists of exchanging the position of two adjacent operations in s, while leaving the rest of the schedule unchanged. The exchange move considers two operations i and j in s, and produces a new sequence s' in which i replaces j and vice versa. The swap move is therefore a particular type of exchange move. The insert move consists of

choosing an operation i and producing a new schedule in which i is removed from its position and inserted before or after another operation j. Figure 3 provides an illustration of the three moves.

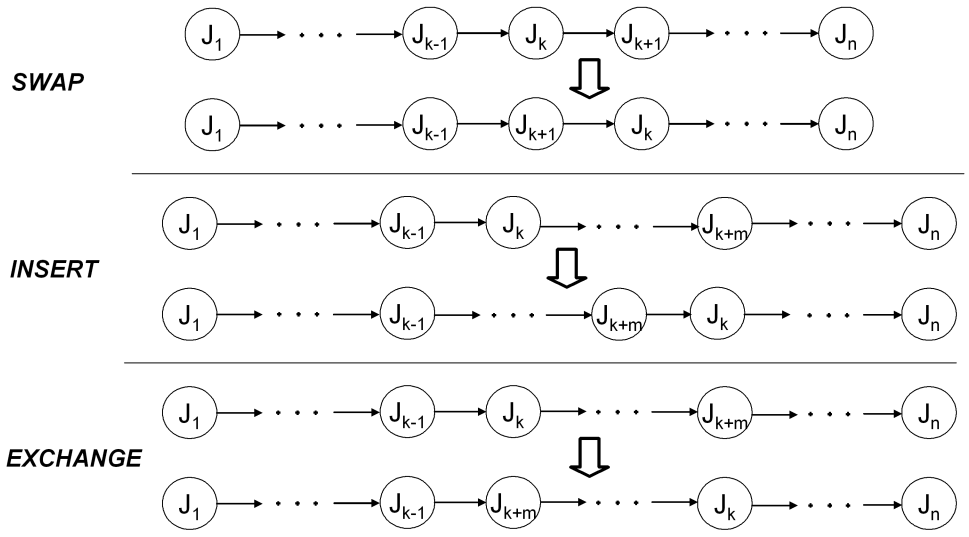

Figure 3: An illustration of the three moves.

The selection of a new solution within the neighborhood is typically performed by evaluating each solution $s' \in \mathcal{N}(s)$ with a *scoring function* $f(s')$. The scoring function depends on the problem under study and can be an approximation or a surrogate of the value of the objective function to be minimized in s'. Common approaches to choosing the new solution $s' \in \mathcal{N}(s)$ consist in selecting a solution $s' \in \mathcal{N}(s)$ such that $f(s') < f(s)$. Algorithms based on this approach are called local search algorithms. The most common choices are to choose s' such that $f(s') = min\{f(x) : x \in \mathcal{N}(s)\}$, called the *steepest descent* strategy, or the first solution $s' \in \mathcal{N}(s)$ such that $f(s') < f(s)$, called the *first improvement* strategy.

The iterative algorithm consists of replacing s with s' and continuing until the stopping criterion is satisfied. Common stopping criteria consist of stopping the algorithm when a time limit is reached or if there are no improving moves in the neighborhood of the current solution, i.e., when $f(s) \leq min\{f(x) : x \in \mathcal{N}(s)\}$. If the latter case holds, s is called a *local minimum* with respect to $f(x)$ and $\mathcal{N}(x)$.

4.3 Constructive metaheuristics

As observed, e.g., by Lawrence [45], greedy heuristics may exhibit erratic behavior, possibly because (*i*) locally promising decisions may not lead to good global solutions, and (*ii*) poor choices can not be changed once made. In order to overcome these limitations, general strategies have been developed to improve the reliability and efficiency of greedy heuristics, incorporating them into more sophisticated constructive metaheuristics. Common strategies are based on the following ideas:

- Maintain a set of partial promising solutions. This is the basic idea of the *beam search* technique [62], which consists of a limited breadth first visit of the branching tree. At each step of the search process only the most promising β (*beam width*) candidates are maintained, and all the other partial solutions are discarded. The beam search procedure has been applied by Kanakamedala *et al.* [42] to a rescheduling problem arising in a multi-purpose batch chemical plant, and to job shop problems by Sabuncuoglu and Bayiz [79] using dispatching rules, by Werner and Winkler [91] using an insertion heuristic and a local search procedure, and by McMullen and Tarasewich (2005) [57] for mixed-model scheduling with setups.

- Restart and randomize the construction. The mechanism of the *Greedy Randomized Adaptive Search Procedure* (GRASP) [26] consists of repeatedly constructing a solution using a greedy algorithm with a randomized criterion, and then improving the solution with a local search. If the new solution improves the current best solution, the new solution is stored as the current best. Binato *et al.* [12] develop a GRASP algorithm for the job shop scheduling problem.

- Backtrack mechanism. An effective strategy to guide the exploration of the search tree is *Limited Discrepancy Search* [37], with its variants and extensions [44, 90]. In Limited Discrepancy Search, a discrepancy is defined when the child chosen is not the best one according to the heuristic criterion. The

underlying idea is that solutions with small discrepancy are more promising and worth visiting early in the search.

- Reiterated construction. The *Iterated Greedy* approach iterates a Destruction-Construction cycle to improve the performance of a greedy heuristic. The algorithm consists of three different phases, destruction (some components of the current solution are removed), construction (the greedy algorithm is applied to the partial solution to build a new complete solution) and the acceptance criterion (the new solution is evaluated). The iterated greedy approach has been demonstrated to be very successful for solving the flow shop scheduling problem [78].

- Look-ahead strategy. This is the main idea of the *Rollout* algorithm [8, 9] or *Pilot* method [25, 88] to overcome the myopic effect greedy heuristics. A master program enlarges a partial solution by using one or more greedy algorithms as a look-ahead strategy. Meloni *et al.* (2004) [58] applied this approach to solve several versions of the job shop problem.

We next illustrate the rollout/pilot method. The basic idea of this metaheuristic is to consider a solution s to an optimization problem as a collection of m components, $s = (s_1, s_2, \ldots, s_{m-1}, s_m)$, e.g. a sequence of jobs to be defined. The problem is then solved sequentially by fixing the components $s_1, s_2, \ldots, s_{m-1}, s_m$ one at a time. A partial solution S_k in which the value of k components is fixed is called a *k-state*. For example, $S_k = (s_1, s_2, \ldots, s_{k-1}, s_k)$ may correspond to the first k jobs in a schedule. Starting from $k = 0$ with no fixed components, the k-th iteration consists in evaluating a *scoring function* $p(e)$ for each seemingly feasible component e which can be added to the current state S_{k-1} and choosing the most promising component, i.e., the one having the smallest scoring function. The iteration is repeated for $k = 1$ to m, when a complete solution is found.

The scoring function $p(\cdot)$ is a lookahead strategy, which in the rollout method is guided by one or more sub-heuristics, called *pilot heuristics*. More precisely, consider

Algorithm Pilot/Rollout
begin
for $k = 1 \ldots, |M|$ **do**
  begin
  $p(e_0) = +\infty$
  for all $e \in M$ **do**
    begin
    if $p(e) = \min_i\{H^i(S_{k-1}, s_k = e)\} < p(e_0)$ **then**
      $e_0 = e$
    end
  $s_k = e_0$, $M = M \setminus \{e_0\}$
  end
end

Figure 4: Pseudocode of Pilot/Rollout Algorithm.

a state S_{k-1} and let e be a possible value for the s_k-th component. The i-th pilot heuristic $H^i(\cdot)$ is a constructive algorithm that, starting from the partial solution $S_k = (S_{k-1}, s_k = e)$, produces a complete solution with objective function value $H^i(S_{k-1}, e)$. The value of the scoring function $p(e)$ for $(S_{k-1}, s_k = e)$ is then based on the values $H^i(S_{k-1}, e)$ of all the pilot heuristics, e.g. the minimum value for all pilot heuristics. Then, if $p(e_0) = \min\{p(e)\}$, then $s_k = e_0$. Figure 4 shows a sketch of the rollout algorithm applied to the problem of sequencing a set M of jobs.

4.4 Iterative improving metaheuristics

Iterative improving heuristics typically converge to local minima. To overcome this limitation, a variety of strategies have been proposed to escape from local minima. These can be broadly classified as iterative improving metaheuristics. Among the metaheuristic schemes that appear more suitable for applications in industrial planning and scheduling are Iterated Local Search (ILS), Tabu Search (TS), Variable Neighborhood Search (VNS) and Simulated Annealing (SA). Several appli-

cations of these techniques to planning and scheduling problems are reported in [16, 17, 19, 31, 62]. In this section we give a brief description of these methods, and then discuss in more detail the VNS scheme.

- *Tabu Search* (TS) metaheuristic [30, 32] is an iterative algorithm with steepest descent criterion, which accepts non-improving moves and uses a tabu list to restrict the neighborhood at each step, based on the past moves. Non-improving moves allow the algorithm to escape from local minima, while the tabu list prevents the search returning to solutions already selected in recent iterations. More precisely, at each step, the inverse of the last applied move is added to the tabu list. The moves in the tabu list are then forbidden in the next steps. A move remains on the tabu list for a limited number of iterations, called the length of the tabu list, which can be fixed or variable. This mechanism can be overruled when a solution associated with a tabu move satisfies an aspiration criterion by leading to a solution with a good objective function value. In a basic TS algorithm the only parameter to set is the length of the tabu list. More sophisticated TS schemes have been proposed in the literature, and this method is certainly among the most successful heuristics for a large number of planning and scheduling problems (e.g. [27, 65]).

- *Simulated Annealing* (SA) is an optimization technique based on the analogy with cooling solids (e.g. metals) [2, 3]. The basic idea is that when solids cool, the particles behave in a way that can be viewed as a local search for the minimum energy configuration. The literature on the SA is quite rich in successful applications of the technique to planning and scheduling problems [86]. SA escapes from local minima by using a non-deterministic move selection. At each iteration of a SA algorithm, a candidate move is randomly selected and the corresponding change in the objective function value Δ_f is evaluated. The candidate move is always accepted if it leads to an improvement, and is accepted with probability $e^{-\frac{\Delta_f}{T}}$ if the move is not improving (i.e., Δ_f is pos-

itive). In SA terminology, the parameter T is referred to as the temperature. When T assumes high values, the probability of accepting a non-improving move becomes close to one for all moves and the algorithm acts as a random search procedure. On the other hand, for very low values of T, only improving moves will be accepted. The SA algorithm starts with a high value of T which slowly decreases over a large number of iterations. The detailed specification of this mechanism is called the cooling schedule.

- *Iterated Local Search* (ILS) performs a local search in a neighborhood structure until a local minimum is found. Then a perturbation mechanism allows the procedure to escape from local minima by generating a new starting point and restarting the local search phase. Finally, an acceptance criterion decides whether to accept the new local minimum or not [48, 49]. A perturbation can be implemented with a number of randomly selected moves, called kick moves, or with a problem specific strategy, e.g., as a short search in a secondary neighborhood, pursuing a secondary evaluation function. There are several variants of this metaheuristic and several applications to planning and scheduling in production and supply chain systems are presented in the literature [10, 24, 85].

- *Variable Neighborhood Search* (VNS) is a local search technique focused on systematic neighborhood changes [36, 60]. The main idea of this algorithm is to exploit several different neighborhood structures. It is based on the observation that a local minimum for a given neighborhood is not necessarily a local minimum for different neighborhoods. The sketch of the algorithm is given in Figure 5.

  Let $\{\mathcal{N}_1 \ldots \mathcal{N}_k\}$ be a set of neighborhood structures. The VNS starts the search process from an initial solution in the first neighborhood $i = 1$, i.e., $\mathcal{N}_1$. In the generic iteration, the algorithm scans the i-neighborhood looking for an improving move; once an improving move is detected it is performed and the VNS starts searching in the first neighborhood ($\mathcal{N}_1$). If no such improv-

Variable Neighborhood Search (VNS)
Input: a set of neighborhoods: $\{\mathcal{N}_1 \ldots \mathcal{N}_k\}$, and an initial solution;
$i := 1$
repeat
  search for a profitable move in $\mathcal{N}_i$
  if a move is found **then**
    apply the move
    $i := 1$
  else $i := i + 1$
until $i \geq k$

Figure 5: Algorithmic scheme of Variable Neighborhood Search (VNS).

ing move exists, the algorithm starts searching in the $(i + 1)$-neighborhood. The search terminates when no improving moves are available in any of the considered neighborhoods. Since the VNS accepts only improving moves, the final solution of a VNS algorithm is a local minimum for all the considered neighborhoods $\{\mathcal{N}_1 \ldots \mathcal{N}_k\}$.

5 Case study

In this section we report on our experience with a practical implementation of a decision support system for production scheduling at a pharmaceutical production plant located in Italy. The plant supplies different European countries and the production flow is organized into the four main phases of Figure 2. However, while there are only one dispensing and one counting department in the plant, manufacturing and packaging activities are organized with several departments.

The plantwide planning organization follows a 4-week rolling horizon strategy in which departments are called to solve their specific scheduling problems. The plant's ERP system defines a set of due dates for the final products to be delivered in weeks 3 and 4. These become due dates for the packaging department, which are

propagated backward to the counting and manufacturing departments, by assuming approximately one week of lead time for each product, and to the dispensing department, by assuming approximately two weeks of lead time for each product.

The impact of a late delivery can be minor, as when the delay is absorbed by the wholesaler inventory system, or major, when it may cause a stock-out at final customers. In the latter case, a hard deadline is associated to the product besides the due date. A deadline can be viewed as a due date whose violation causes an infinite penalty, and therefore it should not be violated by the scheduler.

At the beginning of week 1, the dispensing department schedules the production orders for weeks 1 and 2 and implements the predictive schedule for week 1. If some product is delivered late with respect to the due date defined by the plan, its scheduled delivery time is used as a release time for the subsequent department. At the beginning of week 2, the manufacturing and counting departments schedule the production orders for weeks 2 and 3 and implement the schedule for week 2. Similarly, at the beginning of week 3 the packaging department schedules the production orders for weeks 3 and 4 and implements the schedule for week 3. The whole process is repeated every week, i.e., every week each department schedules the production for the next two weeks, to recover possible differences between the historical schedule and the predictive schedule. At the end of each week there may be production orders in some department that have not been delivered on schedule, or the production of one or more urgent orders is required by the planner, e.g., in case of stock-outs for some products. In such cases it may be necessary to re-schedule the production in different departments, since late deliveries at a department may cause lack of materials at the subsequent department, while urgent orders cause extra requirements at the corresponding department. Late and urgent orders are managed as orders with strict deadlines to be processed as soon as possible. Clearly, deadlines make it difficult to organize long campaigns, and thus reduce the actual capacity of the departments. This reduction may cause, in turn, late deliveries at the end of the week and such negative effects may propagate over several weeks.

All these problems motivate the need for more coordination among the departments and with the planner.

In this section, we consider the production scheduling of three departments: dispensing, counting and manufacturing. The first two are the simplest and least critical departments, which can be described quite easily. The third department is more complex, and represents the typical complexity encountered in secondary manufacturing. A description of the considered departments is given in the next subsections, after which we describe the models of the scheduling problems and the solution algorithms. Computational experience and feedback from the field conclude this section.

5.1 Dispensing department

The dispensing department is in charge of checking the availability of all raw materials required by each recipe and preparing sealed bins containing the needed amount of raw material according to the batch recipe. The sealed bins are then put in a buffer waiting for processing in the manufacturing department.

The weighing operation is carried out in two independent rooms, where the raw materials picked up from a warehouse are weighed and put in a bin. The remainders are sent back to the warehouse, unless the following job requires the same components. Cross-contamination issues require the department to process one product at a time and to clean the room when switching from one product to another. Minor cleaning is sufficient when two consecutive products need the same raw materials, while major cleaning is required when the raw materials change or when the maximum number of consecutive products of the same type, called the size of a campaign, is reached.

From the scheduling point of view each room acts as a single machine with sequence-dependent setup times and campaigns. There may be planned temporary room unavailability, mainly due to lack of personnel, which must be taken into

account when scheduling production. Note that a job can start processing before an unavailability and complete after the interruption, since the weighed materials can be stored in sealed bins which are re-opened after the interruption. Hence, according to Lee [46], the unavailability constraint classifies as resumable. For each production order there may be a release time and a due date or a deadline, when there is a risk of stock-out.

The primary objective in evaluating the quality of a predictive schedule is to respect all the deadlines and, as far as possible, all the due dates. Secondary objectives are makespan minimization, minimization of the number of late jobs and maximization of the total value produced in the scheduling horizon. In the first implementation of the system, the primary objective was the minimization of the maximum lateness, i.e., the maximum over all products of the difference between the product completion time and its due date. As secondary objectives, in lexicographic order, we chose makespan minimization and the minimization of the number of late jobs.

The discussion of the results with plant managers led us to some changes. First, the primary objective was modified to the minimization of the maximum tardiness, i.e., the maximum between zero and the maximum lateness. In fact, since negative values of the lateness correspond to zero tardiness, there can be many schedules with different (negative) lateness, that are all optimal in terms of the tardiness. Among this larger number of solutions we are more likely to find schedules with smaller makespan values, i.e., there is more margin for the minimization of this secondary objective. The second change was the introduction of priorities among products to reflect product value and company preferences. Priorities are used if the total amount of time necessary to complete all production orders exceeds some resource capacity. In such cases, several production orders have to be postponed to the subsequent weeks, and this task is carried out by taking into account the priorities among products.

5.2 Counting Department

The counting department is where materials required for packaging are prepared. Packages, labels and information leaflets are taken from a warehouse, counted and prepared for the subsequent packaging operations. Although each counting operation is relatively simple, the counting department serves several areas of the plant. Moreover, due to the number of different packages that must be used in different countries, this department typically deals with a much larger number of lots than the dispensing department.

The counting department is composed of three independent rooms, although some room may be temporarily unavailable, and there is no significant setup between two consecutive operations. In the counting department, there are also release times (when the packages becomes available), deadlines and due dates (propagated backward by the packaging department). The objectives are, as in the dispensing department, minimizing tardiness, makespan and number of late jobs. In this department, there are also priorities among products when the demand exceeds the capacity.

5.3 Manufacturing Department

In the plant there are several manufacturing departments, each devoted to a particular line of products, in which raw materials are processed to form finished drugs. We focus on a department with 24 different machines, where the presence of several constraints makes the scheduling task particularly difficult.

Some machines may be unavailable due to preventive maintenance or, if available, can be shared with other manufacturing departments. In the latter case, the production is organized with reserved time intervals for each department. In our model a machine is considered unavailable when busy with maintenance or when reserved to other departments.

The sequences of machines traversed by different production orders are quite

different, and range from very simple (a single operation) to complex, including job recirculation and assembly of different sub-products. Operations cannot be preempted, i.e., no machine can stop processing an operation to execute a different one. However, some machines can interrupt processing an operation for a while to resume its execution later, for example during unattended time intervals between two shifts. Processing times range from a fraction of hour to several shifts. Buffer space between machines is not an issue, therefore it is considered of infinite capacity in our model.

Each job has a release time (the completion time at dispensing department) and a due date or a deadline. Cross-contamination issues require cleaning the machine when switching from one product to another. Minor cleaning is required when consecutive products belong to the same campaign, major cleaning is required otherwise. Machine tooling issues imply the presence of sequence-dependent setup times, since changing different tools requires different amounts of time. These constraints can be summarized as sequence-dependent setup times with maximum campaign sizes.

A further constraint concerns personnel availability. Workers are needed to supervise machines during processing and setup operations, and their number may vary from shift to shift, the night shift being less supervised. The number of workers needed to supervise a machine varies from zero (for completely automated machines) to two (for complex setup operations), and typically their number is not sufficient to allow all the machines to operate simultaneously. In addition, workers have different skills, and each one is able to supervise a limited set of machines.

As in the counting and dispensing departments, the objective functions are minimization of maximum tardiness, makespan and number of late jobs.

5.4 Descriptive models

In this subsection we formally define the two scheduling problems with the $(\alpha/\beta/\gamma)$ classification scheme of Graham *et al.* [33] and then describe our scheduling algo-

rithms. We use the following notation.

- P_k, identical parallel machines production environment with k machines
- J_k, job shop production environment with k machines
- r_i, release times
- d_i, due dates
- D_i, deadlines
- s_{ij}, sequence-dependent setup times
- MCS, maximum campaign size
- RA, resumable availability constraints on the machines
- RC, resource availability constraints (personnel, tools, machines, ...)
- T_{max}, minimization of the maximum tardiness
- C_{max}, minimization of the makespan
- U, minimization of the number of tardy jobs

With this notation, the Dispensing department can be classified as

$$P_2|r_i, d_i, D_i, s_{ij}, MCS, RA|T_{max}, C_{max}, U,$$

the Counting department as

$$P_3|r_i, d_i, D_i, RA|T_{max}, C_{max}, U,$$

and the Manufacturing department as

$$J_{24}|r_i, d_i, D_i, s_{ij}, MCS, RC|T_{max}, C_{max}, U.$$

We notice that, although the dispensing and counting departments are considered simple departments from the practical point of view, since they are not critical and by far less complicated than the manufacturing department, the resulting scheduling problems are quite difficult NP-hard problems [29, 70]. Moreover, some constraints like the maximum size of a campaign and resumable machine availability are not frequently addressed in the scheduling literature.

5.5 Algorithms for dispensing and counting

In this section we describe the algorithms developed for the first two, non-critical departments. Following the discussion of Section 4, we focus on solution algorithms that are modular, easy to implement and modify, and suitable for incorporating human experience in the computerized methods.

Modified Jackson Schedule (MJS)
Input: a set P of production orders;
$t = \min\{r_i : i \in P\}$, $S = \emptyset$
repeat
  $R = \{i \in P : r_i \leq t\}$
  if there is a job in R with a deadline
  then select a job $j \in R$ with the smallest deadline
  else **if** there is a job in R with a due date
    then select a job $j \in R$ with the smallest due date
    else $t = \min\{r_i : i \in P\}$, $R = \{i \in P : r_i \leq t\}$
  if a job has been selected
  then assign j to the machine able to complete it first, taking into account (if any) setups, campaigns and machine availability, $S = S \cup \{j\}$, $P = P \setminus \{j\}$
    update t to the smallest completion time of all available machines
until $P = \emptyset$

Figure 6: Pseudocode of the Modified Jackson Schedule.

In a first step of the algorithm development phase, we implemented several simple greedy procedures. Some algorithms were simple list schedules such as the Earliest Due Date, Shortest Processing Time and the like. Other greedy heuristics were insertion heuristics or more sophisticated algorithms. The purpose of the first phase was to verify the correctness of the models with the users, and to seek useful properties of the problem instances to exploit in the algorithms. Discussion with the users allowed us to validate the models described in the previous section and to a description of the behavior of human scheduler when building a feasible schedule. The human schedulers in the plant did not follow any formal procedure to schedule

production orders and did not even use any formal definition of the quality of a predictive schedule. The schedules were simply the result of schedulers intuition and past experience. However, the schedules produced by hand in the counting department were quite similar to those from the MJS algorithm illustrated in Figure 6, a modified version of the Jackson Schedule [40]. In the Dispensing department, the schedulers strove to obtain large campaigns in addition to respecting the due dates. Algorithm Delta in Figure 7 is a surrogate of their behavior. In a preprocessing step the processing horizon is divided into intervals of length Δ, and the due dates in the interval $[k\Delta, (k+1)\Delta]$ are replaced with $k\Delta$. Then, when scheduling jobs according to the MJS algorithm, the production orders of the same type having the same due dates are scheduled to form campaigns. Hence, a larger value of Δ favors the formation of larger campaigns over the tardiness minimization objective.

Once an initial schedule was built, small changes were made in order to improve the overall quality of the schedule. We implemented the heuristics of Figures 6 and 7, among others, and the results were considered feasible, although not particularly successful, by the schedulers. Hence, we used these simple procedures as a good starting point for further algorithmic development.

In a second step of the algorithm development phase we used the rollout method of Figure 4 to improve the performance of the basic heuristics. Finally, the variable neighborhood search approach of Figure 5 was applied to further improve the solutions of the rollout algorithm.

We considered five neighborhood structures associated with the following moves. The first three moves disregard the parallel machine environment and aim at improving the single machine schedules independently of each other. The latter two moves involve two machines.

- SWAP: Two adjacent jobs assigned to the same machine M_x are swapped.
- INSERT: A job is removed from its current position in the schedule of M_x and is inserted on the same machine, m positions forward or backward.

Algorithm Delta (Δ)

Input: a set P of production orders;

Forall jobs $i \in P$ compute modified due dates and deadlines: $\bar{d}_i = \lfloor \frac{d_i}{\Delta} \rfloor \Delta$; $\bar{D}_i = \lfloor \frac{D_i}{\Delta} \rfloor \Delta$

$t = \min\{r_i : i \in P\}$, $S = \emptyset$

repeat

$R = \{i \in P : r_i \leq t\}$

if there is a job in R with a deadline

then select a job $j \in R$ with the smallest modified deadline,

in case of tie select a job with the smallest setup

else **if** there is a job in R with a due date

then select a job $j \in R$ with the smallest modified due date,

in case of tie select a job with smallest the setup

else $t = \min\{r_i : i \in P\}$, $R = \{i \in P : r_i \leq t\}$

if a job has been selected

then assign j to the machine able to complete it first, taking into account setups, campaigns and machine availability, $S = S \cup \{j\}$, $P = P \setminus \{j\}$

update t to the smallest completion time of all available machines

until $P = \emptyset$

Figure 7: Pseudocode of the Algorithm Delta (Δ).

- EXCHANGE: Two (non-adjacent) jobs assigned to the same machine M_x are swapped.
- M-MOVE: A job assigned to machine M_x is removed from M_x and inserted in the sequence assigned to machine M_y.
- M-SWAP: A job assigned to machine M_x and a job assigned to machine M_y are swapped.

5.6 Algorithms for manufacturing

In this section, we describe the algorithm implemented for the manufacturing department. Due to the complexity of the production process, the common practice of this department was to schedule production orders with a time horizon of a few days. Therefore, no practical experience was available to design efficient algorithms for this department. We first developed a constructive heuristic able to generate feasible solutions for the problem. We then embedded the algorithm in a rollout scheme followed by a local search phase, based on the ILS framework.

The constructive algorithm for the manufacturing department is composed of two phases as shown in Figure 8. In the first phase, called *backward*, a provisional schedule σ is built backwards in time, by taking into account only some constraints. In the second phase, called *forward*, a detailed schedule Σ is obtained starting from σ and assigning a feasible starting time to all operations. The aim of the latter phase is to correctly evaluate setups and resource availability constraints, which are taken into account only approximately in the backward phase. Specifically, in the backward phase, workers are all viewed as having the same skill and campaigns are not considered, i.e., the cleaning operations between consecutive operations of the same type are always minor. The forward procedure assigns starting times to operations and skilled workers to machines, updating the worker availability for the shifts in which the operation is executed. This operation is made more difficult due to the availability of workers with different skills.

In a second step of the algorithms development phase we use the rollout method to improve the performance of the constructive heuristic. The algorithmic scheme is detailed in Figure 9.

Algorithm Manufacture

1. **Backward phase.** Construction of a provisional schedule σ.
   Let O be the set of operations, $O' = O$,
   while $O' \neq \emptyset$ **do**
      Update the set E of eligible operations
      while $\exists x \in E$ **do**
         Schedule x last in O', i.e., add x at the beginning of σ
         Remove x from O' and E
      Decrease T
2. **Forward phase.** Timing of operations in σ and construction of a final schedule Σ.
   for $i = 1$ to $|O|$ **do**
      Schedule σ_i, the i-th operation of σ, at the minimum feasible starting time, considering all the constraints and the previously scheduled operations $\sigma_1, \ldots, \sigma_{i-1}$
      Update resource availability
   return Σ

Figure 8: Pseudocode of the constructive algorithm for the manufacture department.

Finally, the iterated local search approach shown in Figure 10 is applied to further improve the solution from the rollout algorithm. The ILS applies two moves in tandem:

- SWAP: Two adjacent jobs in σ are swapped;
- SWAP-M: A job j_a assigned to machine M_x is swapped with job j_b, where j_b is the next operation in σ assigned to machine M_x.

These two moves are defined on the provisional schedule σ. Therefore, in order to calculate the corresponding solution, the forward phase of the algorithm must be

executed to obtain a new schedule Σ. Note that a move may also lead to infeasible schedules.

Algorithm Rollout Manufacture

repeat

Compute the set of eligible operations E

for all ($i \in E$)

Reset Σ_{best}

Apply the Backward phase starting from the partial provisional schedule $i + \sigma$

Apply the Forward phase and obtain Σ'

if Σ' is better than $\Sigma_{best} = i$

then $best = i$ and $\Sigma_{best} = \Sigma'$

Add to the schedule $\Sigma = \Sigma + best$ and $\sigma = best + \sigma$

until not a complete schedule Σ

Figure 9: Pseudocode of the Rollout procedure for the manufacturing department.

Algorithm ILS Manufacture

while stopping criterion **do**

x_1 = Perturbation(x)

x_2 = LocalSearch[SWAP](x_1)

x_3 = LocalSearch[SWAP-M](x_2)

x = Acceptance Criterion(x, x_3)

Figure 10: Algorithmic scheme of the ILS procedure in the manufacturing department.

5.7 Results and feedback from the field

The algorithms described in the previous section have been tested on a set of instances for each department, each representing from two to three weeks of planned production. However, we notice that the evaluation of its performance is limited to a short time horizon and is related to the transition from the manual to the

automated scheduling system. More quantitative and accurate conclusions can only be drawn after an extensive observation of the system, carried on in a more stable situation. Moreover, the lack of detailed information on the performance of the previous scheduling process does not allow a direct and complete comparison with the new system. Nevertheless, from these early results several conclusions can be drawn.

As far as the dispensing and the counting departments are concerned, the assessment of the new system was based on the feedback from the department managers. The solutions provided by the stand-alone greedy algorithms were not considered satisfactory, since it was simple for the users to improve the solutions with local changes in the schedules. On the other hand, the schedules obtained after the roll-out and VNS phases were rarely improved by users. The increase in capacity for the dispensing department has been estimated as up to one hour per day by the department manager, which approximately corresponds to a 2% productivity increase. Similar results were obtained for the counting department.

In the manufacturing department, a more systematic comparison was carried out by asking the human schedulers to produce the predictive schedules by hand for several weeks. These schedules were then compared to those produced automatically. In all cases the computerized schedules outperformed the manual schedules, also when taking into account Key Performance Indicators (KPI) that differed from those explicitly included in the objective functions and listed in Section 5.4.

Figure 11 reports the average improvement attained by computerized schedules over manual ones, as far as several KPI are concerned. Specifically, under T_{max}, C_{max} and U we report the improvement of the maximum tardiness, the makespan and the number of tardy jobs, respectively. Under *Setup Time* we report the decrease of the cumulative time required by the machines to switch from a product type to another. T_{tot} denotes the total tardiness improvement, i.e., the reduction of the sum of the tardiness of all jobs. Finally, under *Lead Time* we indicate the decrease in the average time elapsed from the jobs' release to their completion. We notice that the

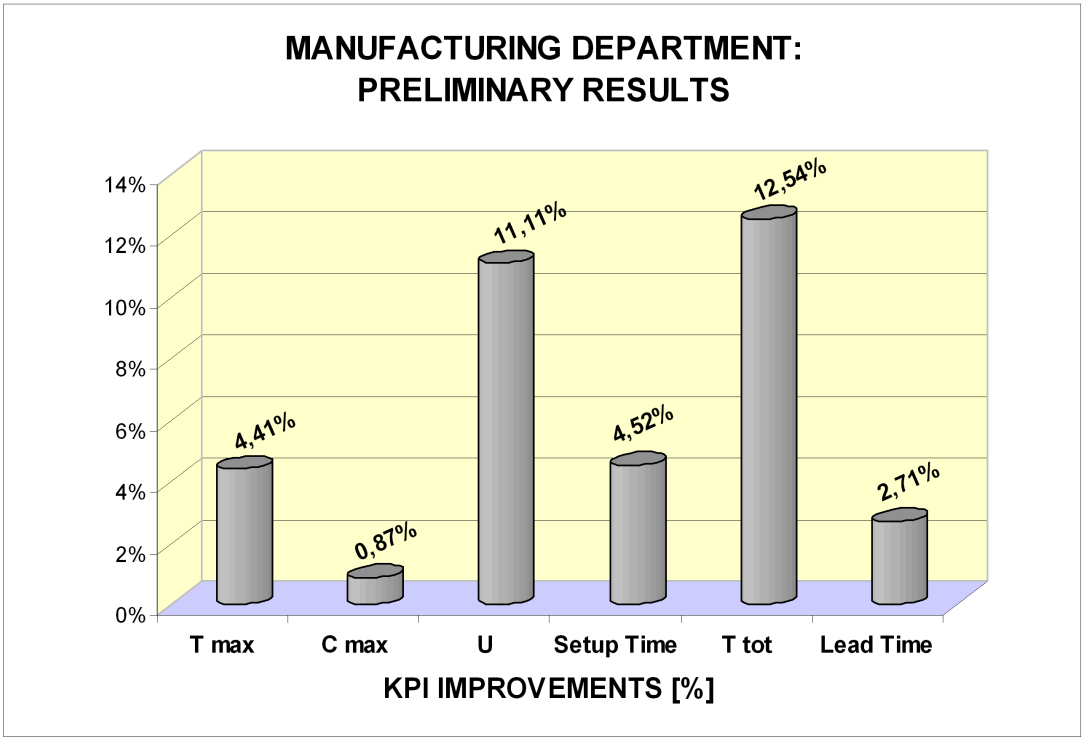

Figure 11: Preliminary results.

automated system is able to outperform the manual one from the viewpoint of conflicting objective functions, i.e., it is able to improve the products on-time delivery while increasing department productivity. Moreover, while a computerized schedule for the next two or three weeks is produced within a few minutes, human schedulers need several hours of work to produce a satisfactory schedule. The common practice at the manufacturing department was actually to schedule production orders every day, just for the subsequent few days, without constructing a complete two-week schedule.

The qualitative feedback from the field is highly positive. The system has been approved by the production managers and accepted by the schedulers. Key success factors of the project included the following facts: (i) the solutions provided by the system were considered convincing by the human schedulers, (ii) the algorithms used by the system were understood by the schedulers, who participated actively in

their tuning, (*iii*) the system allowed the actual planning horizon for the manufacturing area to be extended from a few days to three weeks, (*iv*) the system allows a significant reduction of the time needed to generate new schedules, and (*v*) it allows for negotiation of deadlines for urgent orders with the planner, based on the quantitative impact of urgent orders on department productivity. Other strengths include the improved traceability allowed by the computerized system, which in turn enables better production activity control, communication and information sharing among departments.

Nevertheless, there are aspects of the systems that need further development. Optimizing the production at the four departments led other problems to emerge that were not perceived as relevant with the manual scheduling system. The transportation system can become a bottleneck, in particular when the predictive schedules at dispensing and manufacturing are very different, which requires storing and handling a large number of bins of dispensed materials. Hence the management expressed the need to coordinate the schedules at dispensing and manufacturing by taking into account the material handling system. A further need deals with human resource management. In the current version of the scheduling system, each unit of personnel must be preliminarily assigned to a certain shift in each working day, while the department managers prefer to manage the assignment of persons to shifts and machines with the largest flexibility, to better cope with specific needs of each worker, besides the production process. In fact, the computerized scheduler is currently used by taking into account only the amount of workers needed in each shift, while the actual assignment to shifts and resources is performed manually. Finally, there is a need to measure the difference between predictive and historical schedules, to evaluate the reliability of the schedules. This ongoing process will be important to detect systematic discrepancies and their causes, and consequently to update models, algorithms and production data.

6 Discussion and conclusions

In this chapter we described several algorithmic techniques that can be effectively used to rapidly automate the production scheduling process in a pharmaceutical manufacturing plant and presented a practical implementation of these techniques. Most of the discussion also applies to a large number of manufacturing systems. Clearly, when changing the context, key variables and attributes of the problem will change, as well as key performance indicators. However, for the techniques to be successfully applied, there are some characteristics that must be retained by the production process. In fact, the appropriateness of any technique depends on the structure of the problem and on the manner in which the technique is used. This holds in particular for the content of this chapter.

To the success of the techniques described in this chapter, the reliability and stability of the data set, at least within the scheduling horizon, is important. In the pharmaceutical industry, a large amount of production data is deterministic, being subject to national or international regulations, and only weakly exposed to daily changes. In different production environments, when the data set is very uncertain, or subject to frequent changes, then it may not make sense to implement this kind of techniques to produce schedules that would be in any case unreliable or sub-optimal. If the weekly bucket of work changes everyday, scheduling on a weekly horizon is unlikely to meet company expectations. Similarly, since computerized systems work with numbers, if the processing times loaded in the computer differ from the actual processing times in the shop floor, the performance promised by the predictive schedule will not be met by the historical one. In such cases, more suitable techniques can be used to deal with uncertainties, such as stochastic programming, fuzzy systems and simulation-based methods.

In any case, important features of a good computerized scheduling system include the modularity of the algorithms, the ease of implementation and maintenance operations, the possibility to easily incorporate in the scheduling algorithms observations

and suggestions arising from the scheduling practice. All the methods described in this chapter exhibit these features.

Automated scheduling systems can increase plant productivity, but they must be accepted by the human schedulers, who must understand the principles of the algorithms in order to manage possible unexpected events. However, there are always problems that are better faced by humans than by computers, such as disruption management, personnel management and more in general the management of undefined or ambiguous issues.

A number of problems remain that need further research. The coordination between planning and scheduling, as well as the interaction between human and automated scheduling systems, are critical to the success of planning and scheduling activities. Despite this fact, they have not received enough attention in the academic literature, and need further development. Automated planning and scheduling tools offer new opportunities to improve productivity, but the current interaction mechanisms between planning and scheduling do not seem adequate to fully exploit their potential. New concepts are necessary in this field. A number of issues also remain to be addressed to improve the effectiveness and reliability of models and algorithms. In recent years, there is a clear trend in the academic literature towards richer scheduling models, but practical scheduling problems are still significantly more difficult than the typical academic ones. This is a research direction that needs further attention in the coming years. On the other hand, there are obvious limits to the performance attainable by simple metaheuristic techniques. Simple and general algorithms are useful when prototyping a scheduling system, but when dealing with well-established problems, *ad hoc* algorithms can obtain better performance. Deeper analysis of heuristic algorithms and new mathematical properties for practical problems will be useful from an academic as well as from an industrial point of view.

References

[1] Alle, A., Papageorgiou, L. G. and J. M. Pinto (2004). "A mathematical programming approach for cyclic production and cleaning scheduling of multistage continuous plants." *Computers and Chemical Engineering* **28**: 3–15.

[2] Aarts, E. and J. Korst (1989). *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons.

[3] Aarts, E., and P. J. M. VanLaarhoven (1988). *Simulated Annealing: Theory and Applications*, Kluwer Academic Press.

[4] Allahverdi, A., Gupta, J. N. D. and T. Aldowaisan (1999). "A review of scheduling research involving setup considerations." *Omega* **27**: 219–239.

[5] Askin, R. G. and C. R. Standridge (1993). *Modeling and analysis of manufacturing systems*, John Wiley & Sons.

[6] Bauer, A., Bowden, R., Browne, J., Duggan, J. and G. Lyons (1994). *Shop floor control systems. From design to implementation*, Chapman & Hall.

[7] Berning, G., Brandenburg, M., Gürsoy, K., Kussi, J. S., Mehta, V. and F-J. Tölle (2004). "Integrating collaborative planning and supply chain optimization for the chemical process industry (I) - methodology." *Computers and Chemical Engineering* **28**: 913–927.

[8] Bertsekas, D. P. and J. N. Tsitsiklis (1996). *Neuro-Dynamic Programming*, Athena Scientific.

[9] Bertsekas, D. P., Tsitsiklis, J. N. and C. Wu (1997). "Rollout Algorithms for Combinatorial Optimization." *Journal of Heuristics* **3**: 245–262.

[10] den Besten, M., Stützle, T. and M. Dorigo (2001). "Design of Iterated Local Search Algorithms: An Example Application to the Single Machine Total Weighted Tardiness Problem." *Lecture Notes in Computer Science* **2037**: 441–451.

[11] Bhaskaran, K. and M. Pinedo (1992). Dispatching. *Handbook of Industrial Engineering.* G. Salvendy. John Wiley and Sons: 2184–2198.

[12] Binato, S., Hery, W. J., Loewenstern, D. and M. G. C. Resende (2001). A GRASP for the job shop scheduling. *Essays and surveys on metaheuristics.* C. C. Ribeiro and P. Hansen P. Kluwer Academic Publishers: 59–79.

[13] Blackstone, J. H., Phillips, D. T. and G. L. Hogg (1982). "A state-of-the-art survey of dispatching rules for manufacturing job shop operations." *International Journal of Production Research* **21**: 27–45.

[14] Bollapragada, R. and N. M. Sadeh (2004). "Proactive Release Procedures for Just-in-Time Job Shop Environments Subject to Machine Failures." *Naval Research Logistics* **51**: 1018–1044.

[15] Brown D. E. and W. T. Scherer (1994). *Intelligent Scheduling Systems*, Kluwer Academic Publishers.

[16] Brucker, P., Hurink, J. and F. Werner (1996). "Improving local search heuristics for some scheduling problems - part I." *Discrete Applied Mathematics* **65**: 97–122.

[17] Brucker P., Hurink J. and F. Werner (1997). "Improving local search heuristics for some scheduling problems - part II." *Discrete Applied Mathematics* **72**: 47–69.

[18] Brucker, P., Drexl, A., Möhring, R., Neumann, K. and E. Pesch (1999). "Resource-constrained project scheduling: Notation, classification, models, and methods." *European Journal of Operational Research* **112**: 3–41.

[19] Brucker, P. and J. Hurink (2000). "Solving a chemical batch scheduling problem by local search." *Annals of Operations Research* **96**: 17–38.

[20] Burkard, R. E., and J. Hatzl (2006). A complex time based construction heuristic for batch scheduling problems in the chemical industry. *European Journal of Operational Research* **174**: 1162–1183..

[21] Chang, Y. L., Sueyoshi, T. and R. S. Sullivan (1996). "Ranking dispatching rules by data envelopment analysis in a job-shop environment." *IIE Transaction* **28***: 631–642.*

[22] Clement J., Coldrick, A. and J. Sari (1992). *Manufacturing Data Structures.* John Wiley & Sons.

[23] Cole G. C. (1998). *Pharmaceutical production facilities. Design and applications*, 2nd edition, CRC Press.

[24] Detti, P., Meloni, C. and M. Pranzo (2006). "Minimizing and balancing setups in a serial production system." *International Journal of Production Research* to appear.

[25] Duin, C., and S. Voß(1999). "The pilot method: a strategy for heuristic repetition with application to the Steiner problem in graphs." *Networks* **34**: 181–191.

[26] Feo, T. A. and G. C. Resende (1995). "Greedy randomized adaptive search procedures." *Journal of Global Optimization* **6**: 109–133.

[27] França, P. M., Gendreau, M., Laporte, G. and F. M. Müller (1996). "A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times." *International Journal of Production Economics* **43**: 79–89.

[28] Geiger, C. D., Uzsoy, R. and H. Aytuğ (2006). "Rapid modeling and discover of priority dispatching rules: An autonomous learning approach." *Journal of Scheduling* **9**: 7–34.

[29] Garey, M. R. and D. S. Johnson (1979). *Computers and intractability: a guide to the theory of NP completeness*, Freeman.

[30] Glover, F. (1990). "Tabu Search: A tutorial." *Interfaces* **20**: 74–94.

[31] Glover, F. and G. Kochenberger (2003). *Handbook of Metaheuristics*, Kluwer.

[32] Glover, F. and M. Laguna (1997). *Tabu Search*, Kluwer.

[33] Graham, R. L., Lawler, E. L., Lenstra, J. K. and A. H. G. Rinnooy Kan (1979). "Optimization and approximation in deterministic machine scheduling: a survey." *Annals of Discrete Mathematics* **5**: 287–326.

[34] Grabowski, J., Pempera, J. and C. Smutnicki (1997). Scheduling in production of concrete wares. Operations Research Proceedings 1996. Springer: 192–196.

[35] Hall, N. J. and C. Sriskandarajah (1996). "A survey on machine scheduling problems with blocking and no-wait in process." *Operations Research* **44**: 510–525.

[36] Hansen, P. and N. Mladenović (2001). "Variable neighborhood search: Principles and applications." *European Journal of Operational Research* **130**: 449–467.

[37] Harvey, W. D. and M. L. Ginsberg (1995). Limited discrepancy search, International Joint Conference on Artificial Intelligence (IJCAI'95), Montreal, Canada.

[38] Haupt, R. (1989). "A survey of priority rule-based scheduling." *OR Spektrum*, **11**: 3–16.

[39] Huang, W. and B. Chen (2006). "Scheduling of batch plants: Constraint-based approach and performance investigation." *International Journal of Production Economics* **105**: 425–444.

[40] Jackson, J. R. (1955). Scheduling a production line to minimize maximum tardiness, Research Report 43, Management Science Research Project, University of California, Los Angeles.

[41] Jain, A. S. and S. Meeran (1999). "Deterministic job-shop scheduling: Past, present and future." *European Journal of Operational Research* **113**: 390–434.

[42] Kanakamedala, K. B., Reklaitis G. V. and V. Venkatasubramanian (1994). "Reactive schedule modification in multipurpose batch chemical plants." *Industrial & Engineering Chemistry Research* **33**: 77–90.

[43] Kempf, K., Uzsoy R., Smith S. and K. Gary (2000). "Evaluation and comparison of production schedules." *Computers in Industry* **42**: 203–220.

[44] Korf, R. E. (1996). Improved limited discrepancy search, Proceedings of the thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland, OR, 286–291.

[45] Lawrence, S. (1984). Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques. Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh.

[46] Lee, C. Y. (1996). "Machine scheduling with an availability constraints." *Journal of Global Optimization* **9**: 363–382.

[47] Lin, X., Floudas, C. A., Modi, S. and N. M. Juhasz (2002). "Continuous-Time Optimization Approach for Medium-Range Production Scheduling of a Multiproduct Batch Plant." *Industrial & Engineering Chemistry Research* **41**: 3884–3906.

[48] Lorenço, H. R., Martin, O. and T. Stützle (2001). A Beginner's introduction to Iterated Local Search, Proceedings of MIC'01 Metaheuristics International Conference, Porto, Portugal.

[49] Lourenço, H. R., Martin, O. and T. Stützle (2002). Iterated Local Search. *Handbook of Metaheuristics*. F. Glover and G. Kochenberger. Kluwer: 321–353.

[50] Mallik, A., Pinkus, G. S., and S. Sheffer (2002). "Biopharma's capacity crunch." McKinsey Quarterly **2**: 9–11.

[51] Mascis, A. and D. Pacciarelli (2002). "Job shop scheduling with blocking and no-wait constraints." *European Journal of Operational Research* **143**: 498–517.

[52] McCormick, S. T., Pinedo, M. L., Shenker, S., and B. Wolf (1989). "Sequencing in an assembly line with blocking to minimize cycle time." *Operations Research* **37**: 925–935.

[53] McKay, K. N., Pinedo, M. and S. Webster (2002). "Practice-focused research issues for scheduling systems." *Production and Operations Management* **11**: 249–258.

[54] McKay, K. N., Safayeni, F. R. and J. A. Buzacott (1988). "Job-Shop scheduling theory: what is relevant?" *Interfaces* **18**: 84–90.

[55] McKay, K. N. and V. C. S. Wiers (2003). "Integrated Decision Support for Planning, Scheduling, and Dispatching Tasks in a Focused Factory." *Computers In Industry* **50** (1): 5–14.

[56] McKay, K. N. and V. C. S. Wiers (2003). "Planners, Schedulers and Dispatchers: a description of cognitive tasks in production control." *Cognition, Technology and Work* **5** (2): 82–93.

[57] McMullen, P. R. and P. Tarasewich (2005). "A beam search heuristic method for mixed-model scheduling with setups." *International Journal of Production Economics* **96**: 273–283.

[58] Meloni, C., Pacciarelli, D. and M. Pranzo (2004). "A Rollout Metaheuristic for Job Shop Scheduling Problems." *Annals of Operations Research* **131**: 215–235.

[59] Mishra, B. V., Mayer, E., Raisch, J. and A. Kienle (2005). "Short-Term Scheduling of Batch Processes. A Comparative Study of Different Approaches." *Industrial & Engineering Chemistry Research* **44**: 4022–4034.

[60] Mladenović, N. and P. Hansen (1997). "Variable neighborhood search." *Computers & Operations Research* **24**: 1097–1100.

[61] Mockus, L., Vinson, J. M. and K. Luo (2002). "The integration of production plan and operating schedule in a pharmaceutical pilot plant." *Computers and Chemical Engineering* **26**: 697–702.

[62] Morton, T. E. and D. W. Pentico (1993). *Heuristic Scheduling Systems*, John Wiley & Sons.

[63] Nawaz, M., Enscore, E. E. and I. Ham (1983). "A heuristic algorithm for the m-machines and n-jobs flow-shop sequencing problem." *Omega* **11**: 91–95.

[64] Nowicki, E. and C. Smutnicki (1996). "A fast taboo search algorithm for the job shop scheduling problem." *Management Science* **42**: 797–813.

[65] Nowicki, E. and C. Smutnicki (2005). "An advanced tabu search algorithm for the job shop problem." *Journal of Scheduling* **8**: 145–159.

[66] Ovacik, I. M. and R. Uzsoy (1997). *Decomposition methods for complex factory scheduling problems*, Kluwer.

[67] Pacciarelli, D. (2002). "The alternative graph formulation for solving complex factory scheduling problems." *International Journal of Production Research* **40**: 3641–3653.

[68] Pacciarelli, D. and M. Pranzo (2004). "Production scheduling in a steelmaking-continuous casting plant." *Computers and Chemical Engineering* **28**: 2823–2835.

[69] Panwalkar, S. S. and W. Iskander (1977). "A survey of scheduling rules." *Operations Research* **25**: 45–61.

[70] Pinedo M. (1995). *Scheduling. Theory, algorithms, and systems.* Prentice-Hall.

[71] Pinedo M. (2005). *Planning and Scheduling in Manufacturing and Services.* Springer.

[72] Pranzo, M., Meloni, C. and D. Pacciarelli (2003). "A New Class of Greedy Heuristics for Job Shop Scheduling Problems." *Lecture Notes in Computer Science* **2647**: 223–236.

[73] Piachaud B. (2005). *Outsourcing of R&D in the Pharmaceutical Industry : From Conceptualization to Implementation of the Strategic Sourcing Process*, Palgrave Macmillan.

[74] Romero, J., Espuña, A., Friedler, F. and L. Puigjaner (2003). "A New Framework for Batch Process Optimization Using the Flexible Recipe." *Industrial & Engineering Chemistry Research* **42**: 370-379.

[75] Romero, J., Badell, M., Bagajewicz, M. and L. Puigjaner (2003). "Integrating Budgeting Models into Scheduling and Planning Models for the Chemical Batch Industry." *Industrial & Engineering Chemistry Research* **42**: 6125–6134.

[76] Romero, J., Puigjaner, L., Holczinger, T. and F. Friedler (2004). "Scheduling intermediate storage multipurpose batch plants using the S-graph." *AIChE Journal* **50**: 403–417.

[77] Roy, B. and R. Sussman (1964). Les problèmes d'ordonnancement avec contraintes disjonctives, Note DS No. 9bis, SEMA, Paris.

[78] Ruiz, R. and T. Stützle (2006). "A simple and effective Iterated Greedy algorithm for permutation flow shop scheduling problem." *European Journal of Operational Research* **177**: 2033–2049.

[79] Sabuncuoglu, I. and M. Bayiz (1999). "Job shop scheduling with beam search." *European Journal of Operational Research* **118**: 390–412.

[80] Sadeh, N., Sycara, K. and Y. Xiong (1995). "Backtracking techniques for the job shop scheduling constraints satisfaction problem." *Artificial Intelligence* **76**: 455–480.

[81] Sanmartí, E., Friedler, F. and L. Puigjaner (1998). "Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation." *Computers and Chemical Engineering* **22**: 847–850.

[82] Schutten, J. M. J. (1998). "Practical job shop scheduling." *Annals of Operations Research* **83**: 161–177.

[83] Shah, N. (2004). "Pharmaceutical supply chains: key issues and strategies for optimisation." *Computers and Chemical Engineering* **28**: 929–941.

[84] Schwindt, C. and N. Trautmann (2002). Storage Problems in Batch Scheduling. Operations Research Proceedings 2001, Springer: 213–217.

[85] Stützle, T. (1998), Applying iterated local search to the permutation flow shop problem, Technical Report AIDA-98-04, FG Intellektik, TU Darmstadt.

[86] Van Laarhoven, P. J. M., Aarts E. H. L. and J. K. Lenstra (1992). "Job Shop Scheduling by Simulated Annealing." *Operations Research* **40**: 112–129.

[87] Vollmann, T. E., Berry, W. L. and D. C. Whybark (1997). *Manufacturing Planning and Control Systems*, Irwin McGraw-Hill.

[88] Voß, S., Fink, A. and C. Duin (2005). "Looking Ahead with the Pilot Method." *Annals of Operations Research* **136**: 285–302.

[89] Voß, S. and D. L. Woodruff (2003). *Introduction to computational optimization models for production planning in a supply chain*, Springer.

[90] Walsh, T. (1997). Depth-bounded discrepancy search, International Joint Conference on Artificial Intelligence (IJCAI'97), Nagoya, Japan: 1388–1393.

[91] Werner, F. and A. Winkler (1995). "Insertion techniques for the heuristic solution of the job-shop problem." *Discrete Applied Mathematics* **58**: 191–211.

[92] White, K. P. and R. V. Rogers (1990). "Job-shop scheduling: limits of the binary disjunctive formulation." *International Journal of Production Research* **28**; 2187–2200.