Roma Tre University
Ph.D. in Computer Science and Automation

Production scheduling in pharmaceutical industry

Luca Venditti

Advisor:
Prof. Dario Pacciarelli

Production scheduling in pharmaceutical industry

A thesis presented by
Luca Venditti
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering

Roma Tre University
Dept. of Computer Science and Automation

30 March 2010

Advisor:
Prof. Dario Pacciarelli

Reviewers:
Prof. Alessandro Agnetis
Prof. Rubén Ruiz Garcìa

Acknowledgments

I would like to express first of all my gratitude to Prof. Pacciarelli for his precious support and his constant helpfulness.
A particular thanks and affective regards go to my PhD colleagues and all people that has taken part of AutOrI laboratory's life during this years, from hard research activity to more recreational and relaxing moments.

Preface

Production scheduling is the phase of production management that produces a detailed description of operations to be executed in a given period of time, typically short. Production planning, instead, is characterized, compared to production scheduling, by an higher level of abstraction and a longer time period of interest. In most manufacturing systems the two main objectives to be achieved in production planning and scheduling are the maximization of the total value produced by the plant and the on-time delivery of the final products. These two objectives are often in conflict with each other. Compared 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.
Given the complexity of the production process and the issue of on-time delivery, it's difficult to have plans well balanced with the production capacity. In this contest, it is evident that scheduling is a critical operation and so, that an automated scheduling system is important, both to obtain good scheduling solutions and to have a better control of the production process. A first aim of this thesis is to give a demonstration of the improvements that may derive from an automated scheduling, by taking into account a real case study of production scheduling in a pharmaceutical industry, in the specific, in its Packaging department. At the same time, in this thesis, the issue concerning the difficulty of solving practical scheduling problems is arisen.

Besides operations management in a single stage, another interesting issue in production management, and in general for supply chain management, is the coordination between stages. In the last decades there is an increasing research activity on coordination of multiple decisions in supply chain management as well as among different stages of a production system. In the pharmaceutical supply chain, coordination issues are particularly important to

achieve the high standards of product quality and availability required by the legal and economical implications of product stock-outs at the final customers. Availability of final products requires not only to achieve excellence at each stage of the scheduling process but also in the coordination between different stages. A second aim of this thesis is to investigate on the benefits that the introduction of a centralized decision support system can bring respect to a decentralized one; the case study of coordination between Packaging department and the subsequent Distribution stage of the pharmaceutical plant is addressed.

This thesis is organized as follows:

1. In Chapter 1 the production scheduling in the pharmaceutical industry is introduced, together with a description of common features in pharmaceutical manufacturing systems and of the the real plant considered

2. In Chapter 2 an overview of some academic problems that constitute a basis for the resolution of real applications problems, is given. Classical formulations of these problems, together with models and algorithms, found in literature, are first presented; in a second analysis, some generalizations of classical formulations are considered, in order to gradually step, passing through more complex problems, into resolution of real application problems.

3. in Chapter 3 the first case study, concerning scheduling of operations in the Packaging department, is shown. A detailed graph model and a Tabu Search algorithm are proposed

4. in Chapter 4 the second case study on coordination between Packaging and Distribution departments is presented. An extension of the graph model used in the previous case study and again a Tabu Search algorithm are proposed.

Contents

List of Tables

List of Figures

Chapter 1

Production scheduling: the pharmaceutical industry

Production scheduling is the phase of production management that produces a detailed description of operations to be executed in a given period of time, typically short. This description contains, for each operation, the specification both of resources (machines, manpower, tools) to be used, and of time when they have to be executed. Production scheduling is often cited in combination with production planning, that is characterized, compared to production scheduling, by an higher level of abstraction and a longer time period of interest. In fact production planning has the aim of defining the orders to be produced in a medium or long term; the result of the production planning constitute constraints to be respected by the scheduling phase. In most manufacturing systems the two main objectives to be achieved in production planning and scheduling are the maximization of the total value produced by the plant and the on-time delivery of the final products. These two objectives are often in conflict with each other. In fact, the former requires the organization of production schedules with large lots, to reduce the number of setups and idle time; instead, for the latter objective an organization with small lots is preferable, with consequent increase of the number of setups and idle time and reduction of the factory throughput. Compared 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.
In the pharmaceutical industry, production planning and scheduling usually in-

teracts as an open loop control chain, in which the planning phase determines input data and constraints to be satisfied by the scheduling phase. In the planning phase, decisions are taken with the aim to satisfy the demand, that is not completely known at the moment of decisions. Then, in the scheduling phase, decisions have to be taken facing two kinds of constraints, i.e. the production orders released by the planning stage (in term of release and due dates) and the availability of the production resources. Hence, when the resource amount required by the plan is not sustainable by the available capacity, the schedule cannot comply with the plan, causing delays in the delivery of final products. So it is important that the production planned does not exceed the capacity of the factory. We have already stated the production planning is driven by the demand. This demand takes the form of wholesaler orders that are typically the result of a negotiation in which order quantities, due dates and penalties for late delivery are regulated by contracts. Contracts stipulations should take into account also future production capacity and evaluate the impact of new order on due dates of those currently in the system, in order to choose sustainable quantities and due dates. However, it is not easy to do this for various reasons, as the complexity of manufacturing processes (given among other factors by constraints related to contamination problems) or the issue of on-time delivery. In fact, these factors lead to frequent under-utilization of shop resources and disregarding of long-term due dates. As in a vicious circle, this difficulty in estimating reliable delivery dates to negotiate with the wholesalers, may cause frequent urgent orders, which further increase the short-term pressure on the operations managers. Given the complexity of the production process and the difficulty to have plans well balanced with the production capacity, scheduling is a critical operation in pharmaceutical industry. So, the importance of an automated scheduling system is evident, both for obtaining good scheduling solutions and for having a better control of the production process.

The remainder of this chapter presents, in Section 1.1, a description of common features in pharmaceutical manufacturing systems; then, a description of the real plant object of the case studies of this thesis, is given in Section 1.2.

1.1 Pharmaceutical manufacturing systems

Typical pharmaceutical supply chain (Figure 1.1) contains at least two stages: primary and secondary manufacturing [10, 49]. 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,

i.e. operations are pushed to next level whether needed or not, and it is organized in long campaigns, 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 a 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. Secondary pharmaceutical manufacturing systems consist of a set of multi-purpose production facilities that produce a variety of intermediate and finished products 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 entities (e.g. suppliers) and/or internal ones (e.g. warehouses). For example, the area of interest for this thesis, i.e. the packaging area, is the one mostly connected to external entities.Primary and secondary manufacturing are typically decoupled by relatively large stocks of components.

From the point of view of production scheduling, it is more interesting to focus on secondary manufacturing only, because at this stage, scheduling is a critical issues to guarantee 100% availability of final products at sustainable costs. In particular, 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.

Common production processes are devoted to producing solid, liquid, aerosol or powdered items according to a family of similar recipes [10]. 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.

Hence, the production is typically organized with four main departments (with eventually a fifth, the Distribution), as in Figure 1.2 which to a certain extent can operate independently of each other:

- Dispensing, where, with respect to the receipt, the availability of all materials required is checked (raw material handling) and materials are weighed and stored in sealed bins (dispensing).

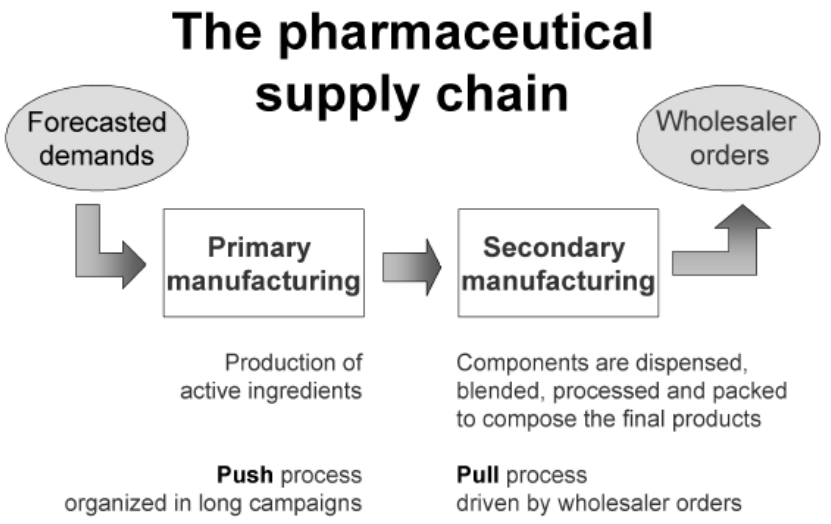

Figure 1.1: The pharmaceutical supply chain

- Manufacturing, where most of activities are performed, i.e.: specific agents are prepared to be used together to wet powders in the granulation (binder preparation); then, dry materials from the bins are passed into a granulator and mixed with specific quantities of binder agents to produce, after a drying procedure, granules (granulation), that are then stored in bins; bin contents are then blended with, eventually, the addition of new materials (blending); then granules are compressed in tablet presses (compression); finally, a solution is prepared and sprayed over the tablets (coating)
- Counting, where packages are prepared according to the different orders and market places (counting)
- Packaging, in which tablets are put into packages to form the final (or semi-final, if they are sent in bulk packages to other sites) products (packaging);

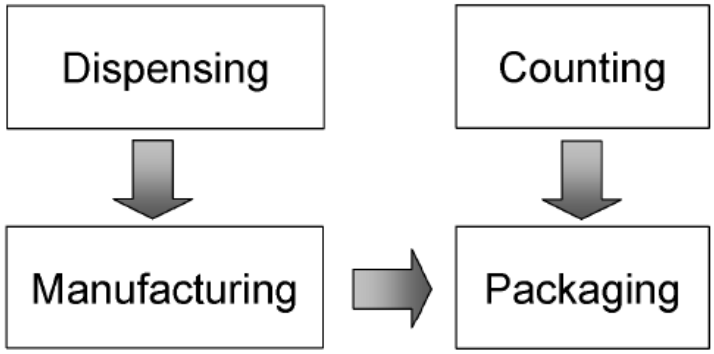

Figure 1.2: Typical layout of a secondary pharmaceutical manufacturing plant

- Distribution, that constitute the final stage, where final products are delivered to customers.

Quality assurance and control activities are distributed throughout the whole manufacturing process. Dispensing and Manufacturing, as Counting and Packaging, are typically decoupled by means of the sealed bins where tablets are typically stored. 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 [10, 39, 49].

Some common features of all departments are:

1. single-product processing, i.e. one product at a time is processed in the department (or in one of its rooms if there are more than one) to avoid cross-contamination between different kinds of products;

2. cleaning operations, that are executed when switching from one product to another with the aim, again, to avoid cross-contamination; minor cleaning is sufficient when two consecutive products belong to the same family, i.e. they need, or they are composed (relating to tablets), by com-

patible raw materials; major cleaning is necessary when the raw materials are incompatible.

3. resource constraints, that 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.

4. sequencing constraints, that 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.

These 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.

All these factors confirm the need of a scheduling decision support system (DSS) to increase the efficiency of the factory by optimizing the use of resources and monitoring the production process to be able to give a quick response to changes that can occur.

1.2 An example of pharmaceutical plant

In this section we describe the plant object of study in this thesis, and in particular the Packaging and Distribution areas. A more complete description of the plant and of the automated scheduling project for the whole plant can be found in [38]. The plant is located in Italy and it supplies different European countries. The production flow is organized into the four main phases described above. However, while there are only one dispensing and one counting department in the plant, manufacturing and packaging activities are organized with several departments. The production planning of the plant follows a 4-week rolling horizon strategy in which departments are called to solve their specific scheduling problems. The result of the planning is a set of production orders and 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. At the beginning of week 1, the dispensing department schedules the production orders for weeks 1 and 2 and implements the schedule for week 1. If some products are delivered late with respect to the due dates defined by the plan, their scheduled delivery time are used as release times 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 consider possible changes that may occur. For example, 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.

Packaging and Distribution

The Packaging department contains three packaging lines working in parallel. Each line can process one lot of identical products at a time and performs all operations from the production of blisters to the final individual specific packages. 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 (corresponding for example to different countries)

Lines are multi-purpose, i.e. they can process more families of products with supplementary tools. However, not all lines can use a specific tool, so lines can process only some families of products. Tools are shared among families of similar products and available in a limited number of copies, in most cases there is a single copy of each tool. In each line, sequence-dependent setup and removal times occur before and after the processing of a lot, in order to clean and calibrate the line and to change the tool. A setup or removal time is called *minor* when requiring no tool change and *major* in the latter case. Lines can be unavailable in given periods for preventive maintenance operations.

Cleaning operations, mechanical configurations and lot processing require a given amount of work for human operators, therefore lines cannot process lots nor execute setups or removals without human resources. The same number of operators is required by each production line. Human resources availability is constant within a shift, but it can vary from one shift to another, the night shift being typically less supervised. In order to reduce the risk of cross-contaminations, the plant policy is to assign each operator to a specific line during the whole shift. Setups and removals cannot be interrupted while the processing of a lot on a line can be interrupted and resumed later on the same line if the line becomes unavailable for planned maintenance or if the number of operators in the subsequent shift is not sufficient to process the lot.

The Distribution department is the final department of the chain. Resources are constituted by vehicles and operators. All vehicles have the same capacity and can contains every kind of package produced. Operators can be divided into internal operators, responsible for the freight of vehicles, and drivers. Deliveries are usually made during the day, except in some cases of urgent orders, in which they are made in the night.

Chapter 2

Literature on scheduling problems: from theory to practice

Models and algorithms known in literature are usually developed for solving simplified versions of real application scheduling problems. However they constitute a basis for the resolution of real world problems, that need more detailed models and more sophisticated algorithms to be faced. Some extensions of most known problems are considered in this chapter together with their models and some neighborhood definitions for Tabu Search, that can be taken as example of effective algorithm to solve practical problems.

2.1 Introduction

Most optimization algorithms from the scheduling literature are mainly concerned with the computation of optimal or near-optimal solutions to very simplified problems [40, 45]. According to the survey of Reisman, Kumar, and Motwani [42], out of 170 articles (related to flowshop scheduling/sequencing) published from 1952 to 1994 only 5 were judged to be true applications. On the other hand, in the last years an increasing number of articles focus on realistic scheduling models including more practical constraints than in the past [37, 41, 45]. In this chapter we give an overview of some academic problems that constitute a basis for the resolution of real applications problems. Classical formulations of these problems, together with models and algorithms, found

in literature, are first presented; in a second analysis, some generalizations of classic formulations are considered, in order to gradually step, passing through more complex problems, into resolution of real application problems. Concerning with algorithms, only heuristics has been considered, because to consider exact algorithms would have gone out of the aim this thesis. In particular, tabu search algorithm and related neighborhoods are treated, because it is certainly among the most successful heuristics for a large number of theoretical and practical scheduling problems [33] and it has been chosen for solving the two case studies in the this thesis. Among books on scheduling theory, in our review, we mostly refer to the classification proposed by Brucker in [5] and by Brucker and Knust in [7].

The remainder of this chapter is structured as follows. In Section 2.2 some basic definitions for scheduling problems are given. Then, Tabu Search algorithm is described in Section 2.3. In Section 2.4 we introduce shop problems, that represent the most common environments in manufacturing scheduling. Some generalizations of shop problems are considered in Section 2.5. Finally, in Section 2.6 we introduce another class of problems adapt to represent real application scheduling, the multi-resource constrained scheduling problems.

2.2 Basic definitions

A *schedule* is an allocation of resources in one or more time intervals to activities to be executed: a scheduling problem consists in finding a schedule such that some constraints are satisfied and some criteria are optimized (objective function). Usually, term *machine* is used to indicate the generic resource while the term *task* or *operation* is used to indicate an activity. A *job* is a set of tasks technologically related to each other.

Denoted with J_j a generic job and with $O_{1j}, \ldots, O_{n_j j}$ its set of tasks, a job is usually characterized by the following elements:

- a *processing time* p_{ij} required to process task i of job J_j
- a set of *eligible machines* $\mathcal{M}_{ij}$ for task O_{ij}
- a *release date* r_j indicating the time when the first task of job J_j is available for processing
- a *due date* d_j within which it is preferable to complete job J_j

- a *deadline* D_j within which job J_j must be completed

Classes of scheduling problems are specified in terms of a three-field classification $\alpha|\beta|\gamma$, introduced by Graham et al. [20], where:

- the first field α represents the shop environment. Some examples are single machine (indicated with 1), parallel machines (P) and multi-purpose machines (MPM);
- the second field represents the constraints of the problem. Some examples are release times (r_i), due dates (d_i), deadlines (D_i), precedence constraints ($prec$), preemption ($pmtn$), sequence dependent setup times (S_{sd}) and resource unavailability ($unaval_j$);
- the third field represents the optimality criteria of the problem. Some examples are: makespan (C_{max}), i.e. the maximum value, among all jobs, of their completion times C_j; maximum lateness (L_{max}), i.e. the maximum value, among all jobs, of $C_j - d_j$; maximum tardiness (T_{max}), i.e the maximum positive value, among all jobs, of $C_j - d_j$; total completion time ($\sum_1^n C_j$), i.e. the sum of completion times of jobs. Optimality criteria may be more than one (*multi-criteria* or *multi-objective* problems); in this case a priority may be given to a criterion rather than to another (for example $Lex(C_{max}, T_{max})$ indicates the minimization of the two objectives in the lexicographic order)

2.3 Tabu Search

Among the most effective algorithms for solving practical scheduling problems there is Tabu Search algorithm, that is an evolution of Local Search algorithm. *Local Search* is an algorithm that, given a solution, searches for better solutions among those near to the given one. This set of near solutions is obtained by applying an operator (*move*) to the current solution and it is called *neighborhood*; when the neighborhood does not contains better solutions than the current one, the algorithm stops. The current solution is called *local optimum*.

Tabu Search is a local search algorithm which makes extensive use of memory for guiding the search. It takes its origins from ideas of Glover [15, 16] stimulated by the need for overcoming limits of Local Search, in which the exploration of the solution space is bounded by feasibility and local optimality criteria. The key concept at the basis of Tabu Search, in fact, is that an

optimization algorithm, in order to be effective and efficient, must provide a large exploration of solutions but at the same time this exploration must be dynamically guided in an intelligent way that can be represented by two aspects: a memory of the past exploration (adaptive memory) and the ability to change strategy during the exploration (responsive exploration).In particular, the adaptive memory is the main feature of Tabu Search together with the neighborhood. The information stored in this memory is used to modify the neighborhood $N(x)$ of the current solution x to obtain a new neighborhood $N^*(x)$ that is used for the exploration. The term adaptive memory can refers to various forms of memory introduced over the years with the development of more sophisticated tabu search procedures. This memory can be *explicit*, when the entire solutions are stored, or implicit, when solution attributes changed in the recent past, usually the moves performed, are stored. A first classification in this variety can be made, for example, between short term and long term memory. A *short term* memory is usually an implicit memory where only information on the recent past of the exploration is available and typically this information is used to determine elements that have to be excluded from $N(x)$ to obtain $N^*(x)$, that consequently is a subset of $N(x)$. In a *long term* memory, instead, information on a more ancient past of the exploration is available and typically this information is used to include additional elements in $N^*(x)$ together with elements of $N(x)$. However, when we deal with Tabu Search in its basic scheme, we commonly refer to a short term memory, called *tabu list*, that keeps track of last solution explored during the recent past by storing the opposites of moves performed. A simple mechanism, combining the use of tabu list with the permission to explore non-improving solutions, gives the possibility to escape from local optima. In fact, if the current solution is a local optima the best non-improving solution can be selected as the new incumbent. At the next iteration, in a memory-less algorithm, optimality criteria would cause the algorithm to re-visit the last local optimum, thus establishing a cycle in the execution of the algorithm. This can be avoided by considering for the exploration the modified neighborhood $N^*(x)$ in which the last solutions visited are excluded. However, this mechanism does not guarantee the escape from local optima in any case. Problems are related to the tabu list length. In fact tabu list is a first-in first-out list, so a move is kept in it for a number of iterations equal to the tabu list size so in some cases this number may be not sufficiently high to avoid the algorithm to return to the last local optimum visited. To face this problem more advanced algorithms feature a dynamic tabu list size. Another problem may arise when a solution obtainable with a tabu move is better than the best solution known at the current iteration; in this

case the move can be deleted from the tabu list and the corresponding solution mantained in $N^*(x)$. In general, criteria used to delete a move from the tabu list are called *aspiration criteria*.
Concerning to long term memory, its most common use consists in storing selected elite solutions, i.e. high quality local optima; for this reason long term memory is usually explicit. These solutions are typically involved in responsive exploration strategies as diversification and intensification. *Intensification* consists in exploring regions of the solution space that are considered particularly good, while *diversification* aims to explore unvisited regions. These regions are reached by moving the search to new solutions obtained from selected elite solutions; specifically, in the case of intensification strategy, selected elite solutions with similar components are assembled together to form new solutions, while in the case of diversification strategy, elite solutions with different features are combined.
Examples in the literature of sophisticated TS schemes for scheduling problems are [13, 36].

2.4 Shop scheduling problems

In this section we discuss shop scheduling problems, such as open shop problems, flow shop problems, job shop problems, and mixed shop problems, which are widely used for modeling industrial production processes. All of these problems are special cases of the general shop problem, defined in the following.

A set $\mathcal{M}$ of m machines $M_1, \ldots, M_m$ and a set $\mathcal{J}$ of n jobs J_j with $j = 1, \ldots, n$ are given. Job J_j consists of n_j operations O_{ij} with $i = 1, \ldots, n_j$ with processing times p_{ij}. Each operation O_{ij} must be processed on a machine $\mu_{ij} \in \mathcal{M}$. There may be precedence relations between the operations of all jobs. Each job can only be processed by one machine at a time and each machine can only process one job at a time. The objective is to find a feasible schedule that minimizes some objective function of the completion times C_j of the jobs $j = 1, \ldots, n$. The objective functions are assumed to be regular, i.e. non-decreasing functions of C_j

Job shop

The Job Shop Scheduling Problem (JSSP or Job-Shop) is one of the most studied combinatorial optimization problems in literature. The classic job-shop scheduling problem may be formulated as follows. A set $\mathcal{M}$ of m machines

$M_1, \ldots, M_m$ and a set $\mathcal{J}$ of n jobs $J_1, \ldots, J_n$ are given. Job J_j consists of n_j operations O_ij with $i = 1, \ldots, n_j$ which have to be processed in the order $O_{1j} \rightarrow O_{2j}, \ldots, O_{n_j j}$. Operation O_{ij} must be processed for a processing time $p_{ij} > 0$ on a dedicated machine $\mu_{ij} \in M_1, ..., M_m$. Each machine can process only one job at a time and $\mu_{ij} \neq \mu_{i+1,j}$ for all $j = 1, \ldots, n$ and $i = 1, \ldots, n_j - 1$, i.e. no two subsequent operations of a job are processed on the same machine (otherwise the two operations may be replaced by one operation). Such a job-shop is also called a job-shop without machine repetition. In the classic formulation there is sufficient buffer space between the machines to store a job if it finishes on one machine and the next machine is still occupied by another job. A schedule is defined by the vector $S = (S_{ij})$ of the starting times of all operations O_{ij}. A schedule S is feasible if: (i) $S_{ij} + p_{ij} \leq S_{i+1,j}$ for all operations (precedence constraints between operations of the same job), and (ii) $S_{ij} + p_{ij} \leq S_{uv}$ or $S_{uv} + p_{uv} \leq S_{ij}$ for all pairs O_{ij}, O_{uv} of operations with $\mu_{ij} = \mu_{uv}$, (each machine processes only one job at a time). The objective is to determine a feasible schedule S such that the makespan $C_{max} = \max_{j=1}^{n} C_j$, where $C_j = S_{n_j,j} + p_{n_j,j}$ is the completion time of job J_j.

A feasible schedule S defines for each machine M_k a sequencing $\pi_k = (\pi_k(1), \ldots, \pi_k(m_k))$ of the m_k operations that has to be processed on M_k.

Model

A job shop can be represented by a so-called disjunctive graph, introduced by Roy and Sussman in 1964 [43]. A *disjunctive graph* $\mathcal{G} = (V, C, D)$ is a mixed graph where V is the set of nodes, C a set of directed arcs (conjunctions) and D a set of undirected arcs (disjunctions). In our case each node in V represents an operation (including two dummy operations *start* and *end*, one preceding and one following all operations. each arc in C represents a precedence constraint between two consecutive operations of the same job; finally, for each pair of operations which have to be processed on the same machine, there is an undirected arc in D representing the two possible precedence relations between the two operations. Nodes in V are weighted with the processing time of corresponding operations. For ease of notation operations can be indicated with index i, the corresponding job with $J(i)$ and the corresponding machine with $\mu(i)$, with $i = 1, \ldots, n_o$, with n_o being the number of operations.

With the disjunctive graph representation, the problem of finding a feasible schedule for the job-shop problem is equivalent to the problem of fixing a direction for each disjunction such that the corresponding graph is acyclic. A set $\mathcal{S}$ of fixed disjunctions is called a *selection*. A selection $\mathcal{S}$ is called *complete*

if a direction has been fixed for each disjunction in D; moreover, $\mathcal{S}$ is called *consistent* if the corresponding graph $\mathcal{G}(\mathcal{S}) = (V, C \cup \mathcal{S})$ is acyclic. Given a complete consistent selection $\mathcal{S}$, a corresponding feasible earliest start schedule can be easily calculated. Denoted with r_i the length of a longest path from *start* to i in $\mathcal{G}(\mathcal{S})$, where the length of a path is defined as the sum of the weights of all nodes on the path, node i excluded, this feasible earliest start schedule can be calculated by setting $S_i = r_i$. Therefore, makespan $C_{max}(S)$ of schedule S is equal to the length r_{end} of a longest path from *start* to *end*. Vice versa, for each schedule S, there is an associated complete consistent selection $\mathcal{S}$; the feasible earliest start schedule corresponding to $\mathcal{S}$ is not worse than S in terms of a regular objective functions, so only complete consistent selections may be considered.

Heuristic and meta-heuristics: local search and tabu search

Among the most effective algorithms for solving JSSP there is Tabu Search algorithm. Some definitions of problem-specific neighborhoods for local and tabu search algorithms based on the disjunctive graph are presented next.
We have stated that only complete consistent selections may be considered. A natural way to go from one complete selection $\mathcal{S}$ to another selection $\mathcal{S}'$ is to reverse certain disjunctive arcs. A first neighborhood N_{ca}, introduced by van Laarhoven et al. [52] may be defined by reversing critical arcs. In fact, as they observed, re-sequencing consecutive jobs that are not on the longest path from *start* to *end* cannot improve the makespan. More specifically, a neighbor in $N_{ca}(\mathcal{S})$ is generated by a move that reverses one oriented disjunctive arc $(i, j) \in \mathcal{S}$ on some critical path in $\mathcal{G}(\mathcal{S})$ into the arc (j, i). This neighborhood has the property that all selections $\mathcal{S}' \in N_{ca}(\mathcal{S})$ are feasible. This can be proved as follows. Assume to the contrary that a selection $\mathcal{S}' \in N_{ca}(\mathcal{S})$ is not feasible, i.e. the graph $\mathcal{G}(\mathcal{S}')$ contains a cycle. $\mathcal{G}(\mathcal{S})$ is acyclic for hypothesis. Thus, the reversed arc $(j, i) \in \mathcal{S}$ must be part of the cycle in $\mathcal{G}(\mathcal{S}')$ implying that a path from i to j in $\mathcal{G}(\mathcal{S}')$ exists which consists of at least three operations i,h,j. This path must also be contained in $\mathcal{G}(\mathcal{S})$ and must be longer than the path consisting of the single arc $(i, j) \in \mathcal{S}$ because all processing times are assumed to be positive. Since this is a contradiction to the fact that the arc (i, j) belongs to a longest path in $\mathcal{G}(\mathcal{S})$, the graph $\mathcal{G}(\mathcal{S}')$ cannot be cyclic. In the case that the neighborhood $N_{ca}(\mathcal{S})$ of a selection $\mathcal{S}$ is empty, we may conclude that $\mathcal{S}$ is optimal. In fact, in this case the chosen critical path contains no disjunctive arc, then it contains only conjunctive arcs which implies that $C_{max}(\mathcal{S})$ is equal to the total processing time of a job. Since this value is a lower bound for

the optimal makespan, $\mathcal{S}$ must be optimal. Neighborhood N_{ca} is proved to be not connected, but it is proved to be opt-connected, i.e. it guarantees the reachability of an optimal solution.

A disadvantage of N_{ca} is that several neighbors $\mathcal{S}'$ exist which do not improve the makespan of $\mathcal{S}$. In the following we introduce a more effective neighborhood, based on concepts of blocks. If P^C is a critical path in $\mathcal{G}(\mathcal{S})$, a sequence $i_1, \ldots, i_k$ of at least two successive operations on P^C is called a (machine) *block* if the operations of the sequence are processed consecutively on the same machine, and enlarging the subsequence by one operation leads to a subsequence which does not fulfill this property. A first resolution approach based on blocks can be found in [17] for a single machine scheduling problem with release dates and due dates, and later it was adopted for other problems, including job shop [12, 35]. The following property has been proved in [17]:

Theorem 2.4.1 *Let S be a complete consistent selection with makespan $C_{max}(\mathcal{S})$ and let P^C be a critical path in $\mathcal{G}(\mathcal{S})$. If another complete consistent selection $\mathcal{S}'$ with $C_{max}(\mathcal{S}') < C_{max}(\mathcal{S})$ exists, then in $\mathcal{S}'$ at least one operation of a block B on P^C has to be processed before the first or after the last operation of B.*

The proof of this property is based on the fact that if the first and the last operation of each block remain the same, all permutations of the operations in the inner parts of the blocks do not shorten the length of the critical path.

On the basis of this theorem, the following block shift neighborhood N^1_{bs}, used by Grabowski et al. [18] for flow-shop problem and later by Grabowski and Wodecki [19] for job-shop problem, may be defined. $N^1_{bs}(\mathcal{S})$ contains all feasible selections $\mathcal{S}'$ which can be obtained from $\mathcal{S}$ by shifting an operation of some block on a critical path to the beginning or the end of the block. Consequently, in this neighborhood the orientation of several disjunctive arcs may be changed. For this reason, feasibility of neighbor selections must explicitly be stated because it is not automatically fulfilled as in N_{ca}. Since infeasible moves are not allowed, it's more probable that opt-connectivity is not satisfied, in fact opt-connectivity of this neighborhood is still an open question.

An opt-connected neighborhood N^2_{bs}, extending N^1_{bs}, can be obtained by allowing some additional moves. If a move of an operation to the beginning [or the end] of a block is infeasible, then the operation is moved to the first (last) position in the block such that the resulting selection is feasible. With the inclusion in N^2_{bs} of these other moves than moves in N^1_{bs}, it can be proved that neighborhood N^2_{bs} is opt-connected. Even for neighborhood N^2_{bs}, if the neighborhood $N^2_{bs}(\mathcal{S})$ of a selection $\mathcal{S}$ is empty, then $\mathcal{S}$ is optimal.

A disadvantage of neighborhoods N^1_{bs} and N^2_{bs} is that they may be quite large. For this reason a smaller neighborhood $N^2_{ca} \subseteq (N_{ca} \cap N^1_{bs})$ was proposed [34, 36]. $N^2_{ca}(\mathcal{S})$ contains all selections $\mathcal{S}'$ which can be obtained from $\mathcal{S}$ by interchanging the first two or the last two operations of a block. Since $N^2_{ca}(\mathcal{S})$ is a subset of $N_{ca}(\mathcal{S})$, all selections in N^2_{ca} are feasible. Furthermore, due to Theorem 2.4.1 all selections $\mathcal{S}' \in N_{ca}(\mathcal{S}) - N^2_{ca}(\mathcal{S})$ satisfy $C_{max}(\mathcal{S}') \geq C_{max}(\mathcal{S})$ since only operations in the inner part of blocks are changed. Thus, only non-improving solutions are excluded in the reduced neighborhood $N^2_{ca}(\mathcal{S})$.

Concerning neighborhood evaluation, in order to determine the best neighbor for a given solution, two methods are possible: exact and approximate evaluation. Exact evaluation consists in calculating exactly the makespan while approximate evaluation consists in calculating an estimate of the makespan, that usually is a lower bound of the makespan. Exact evaluation is worth when it does not take much computational time. In the job-shop case, due to the fact that each operation has at most two predecessors (a job and a machine predecessor), makespan can be calculated in $O(n_o)$ time by longest path calculations where n_o is the number of operations. However, this is not enough to have acceptable computational times, because makespan calculation has to be performed for all solution in the neighborhood. To reduce the computational time it is needed to prove some additional properties. In general, this is possible for neighborhood obtained by not very disrupting moves (as N_{ca} and N^2_{ca}); instead, for neighborhood obtained by more disrupting moves (as N^1_{bs} and N^2_{bs}), an approximate evaluation is usually needed.

An important property for evaluating neighborhood N^2_{ca} has been proved by Nowicki and Smutnicki [35]. This property is shown in the following, after some notations. We have already noticed that a schedule can be specified by a vector $\pi = (\pi_1, \ldots, \pi_m)$ where, for $k = 1, \ldots, m$, the sequence $\pi_k = (\pi_k(1), \ldots, \pi_k(m_k))$ defines the order in which all m_k operations i with $\mu(i) = M_k$ are processed on M_k. Let $\mathcal{G}(\pi)$ be the directed graph corresponding to π; a solution π is feasible if and only if $\mathcal{G}(\pi)$ contains no cycles. Let π be a feasible solution and let π' be a neighbor of π with respect to the neighborhood N_{ca}. The solution π' is derived from π by reversing a critical oriented disjunctive arc (i, j) in $\mathcal{G}(\pi)$. Let $\pi^{(i,j)}$ be such a solution and, for a feasible schedule π and operations u, v, let $\mathcal{P}_\pi(u \vee v)$ and $\mathcal{P}_\pi(\overline{u})$ be the set of all paths in $\mathcal{G}(\pi)$ containing nodes u or v and not containing u, respectively. Furthermore, let $\mathcal{P}_\pi(\overline{u} \wedge \overline{v})$ and $\mathcal{P}_\pi(\overline{u} \wedge v)$ denote the set of all paths in $\mathcal{G}(\pi)$ not containing nodes u and v and not containing u but containing v, respectively. For a given set of paths $\mathcal{P}$ the length of a longest path in $\mathcal{P}$ is denoted by $l(\mathcal{P})$. The following

property has been proved [35]:

Property 2.4.1 *The makespan of a solution* $\pi' := \pi^{(i,j)} \in N_{ca}(\pi)$ *is given by* $C_{max}(\pi') = \max\{l(\mathcal{P}'_\pi(i \vee j)), \mathcal{P}_\pi(\bar{i})\}$

Property 2.4.1 avoid to consider other paths for calculating the makespan exactly; moreover it has been shown that it can be evaluated in an efficient way, obtaining a further reduction of computational time.

Flow shop

The flow shop problem is special case of job shop, in which:

- each job J_j consists of m operations O_{ij} with processing times p_{ij} $(i = 1, \ldots, m)$ where O_{ij} must be processed on machine M_i;
- there are precedence constraints of the form $O_{ij} \to O_{i+1,j}$ $(i = 1, \ldots, m-1)$, and $\mu_{ij} = M_i$ $(i = 1, \ldots, m)$, for each $j = 1, \ldots, n$, i.e. all jobs are processed on machines in the same order.

Thus, the problem is to find machine orders, i.e. orders of jobs to be processed on the same machine. Among problems with arbitrary processing times p_{ij}, problem $F2||C_{max}$ is the only flow shop problem which has been polynomially solved.

Open shop

The open shop problem is a shop problem more general than job-shop and flow-shop, in which:

- each job J_j consists of m operations O_{ij} $(i = 1, \ldots, m)$ where O_{ij} must be processed on machine M_i, and
- there are no precedence relations between the operations.

Thus, the problem is to find job orders (orders of operations of the same job) and machine orders (orders of operations to be processed on the same machine).

Among problems with arbitrary processing times p_{ij} and no preemption, only the problem with two machines $O2||C_{max}$ (or symmetrically the problem with two jobs $O|n = 2|C_{max}$) has been polynomially solved (in $O(n)$ time). Instead, for various problems with unitary processing time, polynomial algorithms have been found.

2.5 Generalized shop scheduling problems

Various generalizations of shop problems can be found in literature. Some among the most studied are presented next. We refer to generalizations of job-shop, but similar considerations can be done for open-shop and flow-shop

Time-lags

Time-lags are general timing constraints between the starting times of operations. They can be introduced by the start-start relations $S_i + d_{ij} \leq S_j$ where d_{ij} is an arbitrary integer number.

Time-lags d_{ij} may be incorporated into the disjunctive graph $\mathcal{G} = (V, C, D)$ by weighing the conjunctive arcs $i \to j \in C$ with the distances d_{ij} and the disjunctive arcs $i - j \in D$ with the pairs (d_{ij}, d_{ji}): when an orientation is chosen, the arc is weighed with d_{ij} or d_{ji} if it is oriented into the direction (i, j) or (j, i), respectively. Time-lags may for example be used to model the following constraints:

- Release times and deadlines: denoted with r_i a release time and with D_i a deadline for operation i, then the conditions $S_{start} + r_i \leq S_i$ and $S_i + p_i - D_i \leq S_{start}$ with $S_{start} = 0$ must be hold. Thus, release times and deadlines can be modeled by the time-lags $d_{start,i} := r_i$ and $d_{i,start} := p_i - D_i$

- Exact relative timing: if an operation j must start exactly l_{ij} time units after the completion of operation i, then the relation $S_i + p_i + l_{ij} = S_j$ must be hold, with time-lag $d_{ij} = p_i + l_{ij}$

- Setup times: if a setup time is needed between the completion of an operation i and the beginning of operation j on the same machine, one of the two relations $S_i + p_i + s_{ij} \leq S_j$ and $S_j + p_j + s_{ji} \leq S_i$ must be hold, where s_{ij} denotes the setup time between operations i and j.

- Machine unavailabilities: if a machine M_k is unavailable within time intervals $]a_u, b_u[$ for $u = 1, \ldots, v$, we may introduce v artificial operations $u = 1, \ldots, v$ with $\mu(u) = M_k$, processing times $p_u := b_u - a_u$, release times $r_u := a_u$ and deadlines $D_u := b_u$. Then, release times and deadlines can be modeled with time-lags as above

- Maximum lateness objective: problems with maximum lateness objective $L_{max} = \max_{j=1}^{n} C_j - d_j$ can be reduced to C_{max} problems by setting

$d_{l(j),end} = p_{l(j)} - d_j$ for each job j where $l(j)$ denotes the last operation of job j

Blocking

Another generalization concerns blocking operations. A *blocking operation* is an operation that remains on its processing machine M_k and blocks it when it has finished on it and the next machine where the corresponding job has to be processed is still occupied by another job: M_k remains blocked until the next machine is available. This situation can arise when jobs can not be parked in wait outside the machine (for example in a so-called buffer). A JSSP where all operations are blocking is called Blocking Job Shop Scheduling Problem.

Consider two blocking operations i and j which have to be processed on the same machine $\mu(i) = \mu(j)$. If operation i precedes operation j, the successor operation $\delta(i)$ of operation i must start before operation j in order to unblock the machine, i.e. we must have $S_{\delta(i)} \leq S_j$. On the other hand,, if operation j precedes operation i, then operation $\delta(j)$ must start before operation i, i.e. we must have $S_{\delta(j)} \leq S_i$. Thus, there are two mutually exclusive (alternative) relations in connection with i and j. These relations can be modeled by a pair of alternative arcs $(\delta(i), j)$ and $(\delta(j), i)$ with arc weights $w_{\delta(i),j} = w_{\delta(j),i} = 0$. In order to get a feasible schedule, exactly one of the two alternatives has to be chosen (selected). Note that choosing $(\delta(i), j)$ implies that i must precede j by transitivity, and choosing $(\delta(j), i)$ implies that j must precede i. Thus, if blocking arcs $(\delta(i), j)$, $(\delta(j), i)$ are introduced, there is no need to add disjunctive arcs (i, j), (j, i) between operations which are processed on the same machine $\mu(i) = \mu(j)$. Similar considerations can be done in the case in which only one between i and j is a blocking operation, and in the case in which neither i nor j are blocking operations, with the only observation that, when no blocking arc is added between $\delta(i)$ and j [or between $\delta(j)$ and i], disjunctive arc (i, j) $[(j, i)]$ is not deleted; in this case the disjunction between (i, j) $[(j, i)]$ can assume the form $(S_i + w_{ij} \leq S_j)$ $[(S_j + w_{ji} \leq S_i)$ where $w_{ij} = p_i$ $[w_{ji} = p_j]$. In the special case in which operation i is the last operation of job $J(i)$, machine $\mu(i)$ is not blocked after the processing of i, so, in this case, operation i is always assumed to be non-blocking.

Summarizing that, to represent the whole problem, we can introduce a set A of pairs of alternative arcs $\{(i, j), (f, g)\}$ corresponding to mutually exclusive pairs of relations $(S_i + w_{ij} \leq S_j)$ or $(S_f + w_{fg} \leq S_g)$. The resulting graph $G = (V, C, A)$ is called *alternative graph* [28, 29]. Similarly as in the disjunctive graph model we have to determine a selection $\mathcal{S}$ which contains exactly one

arc (i, j) or (f, g) for each pair of alternative arcs in A such that the resulting graph $\mathcal{G}(\mathcal{S}) = (V, C \cup S)$ with arc weights w does not contain a positive cycle.

Flexible Job Shop

Flexible Job Shop Scheduling Problems are problems where machine are flexible, i.e. operations can be performed non only by a one machine by a subset of machines (multipurpose machines).

Problem formulation

Differently from the classic formulation, in FJSSP, a set of machines $\mathcal{M}_i \subseteq \mathcal{M}$ that can process operation i, is associated to operation i. If operation i is executed on machine $M_k \in \mathcal{M}_i$, then its processing time is equal to p_{ik}. Hence, a further decision in FJSSP with respect to JSSP is to assign to each operation i a machine from the machine set $\mathcal{M}_i$; the objective function is the C_{max} minimization. Once a machine is assigned to each operation, the problem reduce to classic JSSP. A typical application of FJSSP can be found in manufacturing scheduling when the operations need certain tools for processing and the machines are only equipped with a few of them. Then $\mathcal{M}_i$ denotes all machines equipped with the tools needed by operation i.

Neighborhoods for local and tabu search

In the following we assume that a machine assignment μ (where $\mu(i) \in \mathcal{M}_i$ denotes the machine assigned to operation i) is given; also a corresponding complete consistent selection $\mathcal{S}$, with the associated earliest start schedule S, is given.

A first kind of neighborhood ([23]) is a natural extension for this problem of the block shift neighborhoods $\mathcal{N}_{bs}^1$ and $\mathcal{N}_{bs}^2$ for the classic job-shop problem as defined in Section 2.4. The definition of these neighborhood is based on the following theorem:

Let $(\mu, \mathcal{S})$ be a feasible solution with makespan $C_{max}(\mathcal{S})$ and let P^C be a critical path in $\mathcal{G}(\mathcal{S})$. If another feasible solution $(\mu', \mathcal{S}')$ with $C'_{max} < C_{max}$ exists, then:

- in μ' at least one critical operation on P^C has to be assigned to another machine, or
- in $\mathcal{S}'$ at least one operation of some block B on P^C has to be processed before the first or after the last operation of B.

Substantially, on the basis of this theorem, neighborhoods $\mathcal{N}_{bs}^1$ and $\mathcal{N}_{bs}^2$ can be extended by considering movement of critical operation from a machine to a certain position in another machine. Again, it can be shown that the enlarged neighborhood N_{bs}^2 is opt-connected.

A disadvantage of these neighborhoods is that they may be quite large since each critical operation may be moved to a large number of positions on other machines. The number of moves may be reduced by allowing only feasible moves and trying to calculate the best insertion position of an operation which is assigned to another machine (i.e. all other insertions of this operation on the same machine do not lead to schedules with a better makespan). In the following, let $\pi_1, \ldots, \pi_m$ be the machine sequences corresponding to the solution $(\mu, \mathcal{S})$ and let v be a given operation and $M_k \in \mathcal{M}$ a given machine (that hereinafter we will identify with its index k for ease of notation). A move that removes v from its machine sequence $\pi_{\mu(v)}$ and insert it into the machine sequence π_k on M_k (if $M_k \in \mathcal{M}(v)$) is called k-insertion. A *k-insertion* is called feasible if the resulting disjunctive graph is acyclic; a feasible k-insertion (of v) is called *optimal* if the makespan of the resulting solution is not larger than the makespan of any other solution which is obtained by a feasible k-insertion (of v). A reduced neighborhood which always contains a solution corresponding to an optimal k-insertion has been used in [30] by Mastrolilli and Gambardella.

Job-Shop Problems with Transport Times

Transportation time can be taken into account in an other generalization of JSSP. When a job is moved from one machine M_k to the next one, say M_l, where it has to be processed, a transportation time t_{jkl} is required. These transportation times may be job dependent or job-independent. In this formulation we assume that the transportation is done by transport robots which can handle at most one job at a time. We again assume that unlimited buffer space exists between the machines, where jobs processed and waiting for a robot may be stored. No further time for the transfer from machine to its buffer is considered. Similarly, each machine has an unlimited input buffer where jobs which have been transported can wait for processing. We assume that transportation times satisfy the following triangle inequality for all jobs J_j and all machines M_k,M_l,M_h: $t_{jkh} + t_{jhl} \geq t_{jkl}$. We assume that there is a set $\mathcal{R}$ of r identical transport robots $R_1, \ldots, R_r$ and each job can be transported by any of these robots. Then following two cases can be distinguished, given the number of jobs n: (i) an unlimited number of robots $r \geq n$ and (ii) a limited number of robots $r < n$. In the first case no conflicts arise between two jobs

requesting for the same vehicle; in the second case, robots are resources with limited capacity as machines, so their assignment to jobs has to be decided. In a further generalization of this model,robots may be nonidentical and may have different characteristics (so-called multi-purpose robots). As for multi-purpose machines, this similar situation is modeled by associating with each job J_j a set $\mathcal{R}_j \subseteq \mathcal{R}$ containing all robots by which J_j can be transported.

General shop problems with machine dependent removal and setup times

In general shop problems, we may have removal and setup, needed to remove a tool from a machine and to setup a different tool in it. In this case we have a partition of the set $I = \{1, \ldots, n_o\}$ of operations of all jobs into disjoint sets $I_1, \ldots, I_r$, called groups. For any two operations i,j, with $i \in I_h$ and $j \in I_l$, that have to be processed on the same machine M_k, operation j can not be started until $g_{hlk}+f_{hlk}$ time units after the completion time of operation i, or operation i can not be started until $g_{lhk} + f_{lhk}$ time units after the completion time of operation j; g_{hlk} and f_{hlk} are the removal and the setup time, respectively. The changeover times, from operation i to operation j and from operation j to operation i, can be represented in the disjunctive graph model by weighing the corresponding disjunctive arcs, (i, j) and (j, i), with $g_{hlk} + f_{hlk}$ and $g_{lhk} + f_{lhk}$, respectively.

2.6 Multi-resource constrained scheduling problems

The resource-constrained project scheduling problem (RCPSP) is a very general scheduling problem which may be used to model many applications not only in manufacturing area but in other areas like services. In fact it takes into account scarcity of resources, that is a typical constraint of real application problems. The objective is to schedule some activities over time such that some scarce resource capacities are respected and a certain objective function is optimized.

The problem may be formulated as follows: n activities $i = 1, \ldots, n$, require, to be processed, a constant amount r_{ik} of renewable resource k with $k = 1, \ldots, r$. A constant amount R_k of resource k is available at any time. Activity i occupies its necessary resources for a time period equal to its processing time p_i. Precedence constraints are defined between some activities. The objective is to determine a feasible vector of starting times $S = \{S_1, \ldots, S_n\}$ for the activities $i = 1, \ldots, n$ such that the objective function is optimized. A solution

S is feasible if the following three conditions are satisfied: (i) at each time t the total resource demand is less than or equal to the resource availability R_k of each resource $k = 1, \ldots, r$, (ii) the given precedence constraints are respected, i.e. $S_i + p_i \leq S_j$ if i precedes j.

It is useful, for the RCPSP, to represent the structure of a project by a so-called *activity-on-node network* $\mathcal{G} = (V, A)$, where the vertex set $V := \{start, 1, ..., n, end\}$ contains all activities plus two dummy ones corresponding to the start and the end of the project, and the set of arcs $A = \{(i, j)|i, j \in V; i \to j\}$ represents the precedence constraints. Each vertex $i \in V$ is weighed with the corresponding processing time p_i. Another less used representation of projects is based on so-called *activity-on-arc networks* where each activity is modeled by an arc.

Local search and tabu search

Some neighborhood definition for local and tabu search algorithms for RCPSP are presented next. Reviews and comparisons of other different heuristics can be found in Hartmann and Kolisch [22] and in Kolisch and Hartmann [25, 26]. The generic solution is represented by a sequences of the activities, the *activity list* L.

- **The adjacent pairwise interchange-neighborhood**. This neighborhood is generated by a move, defined on activitiy i_λ for $\lambda = 1, \ldots, n-1$, that interchanges the elements i and its successor in L, $i_{\lambda+1}$. This move leads to a feasible list if no precedence relation $i_\lambda \to i_{\lambda+1}$ exists. Since we have at most $n-1$ feasible moves for a list, the size of the neighborhood is bounded by $O(n)$.

- **Swap-neighborhood**. The swap-neighborhood generalizes the adjacent pairwise interchange-neighborhood and is generated by a move defined on two activities i_λ and i_μ for $\lambda, \mu = 1, \ldots, n$ with $\lambda < \mu$, that interchanges the elements i_λ and i_μ in L. Such move is only feasible if for all $\nu = \{\lambda + 1, \ldots, \mu - 1\}$ no precedence relation $i_\lambda \to i_\nu$ or $i_\nu \to i_\mu$ exists. Since we have at most $n(n-1)/2$ feasible moves for a list, the size of the neighborhood is bounded by $O(n^2)$.

- **Shift-neighborhood**. The shift-neighborhood also generalizes the adjacent pairwise interchange-neighborhood and is generated by a move defined on an activities i_λ and an index μ for $\lambda, \mu = 1, \ldots, n$, with $\lambda \neq \mu$, that shifts the element i_λ in position μ in L. For $\lambda < \mu$ we have a right

shift, symmetrically, for $\lambda > \mu$ we have a left shift. Such shifts are only feasible if for $\nu = \{\lambda + 1, \ldots, \mu\}$ no precedence relation $i_\lambda \rightarrow i_\nu$ and for $\nu = \{\mu, \ldots, \lambda - 1\}$ no precedence relation $i_\nu \rightarrow i_\lambda$ exists. Since we have at most $n(n-1)/2$ feasible shift-operators for a list, the size of this neighborhood is bounded by $O(n^2)$.

Chapter 3

Case study 1: A Tabu Search algorithm for production scheduling in packaging department

In this chapter, we address the problem of scheduling packaging operations in Packaging department of a real pharmaceutical plant. A detailed graph has been needed to model the problem for its resolution with a tabu search algorithm. This case study shows that scheduling technology is nowadays mature to solve production and, in general, practical scheduling problems.

3.1 Introduction

In this chapter we take into account a real case study of production scheduling in a pharmaceutical industry, in the specific, in its packaging department. Previous attempts to design a computerized tool for production scheduling at this department have had limited success, and similar performance has been observed in practice for many computerized tools for planning and scheduling [31, 32]. A common reason for this behavior is that the models adopted by computerized systems suffer from excessive simplification and do not incorporate all the relevant aspects of the shop floor [54]. On the other hands, example The problem can be formulated as a multi-purpose machine scheduling problem [51] with additional constraints related to the availability of tools and opera-

tors, removal and setup times, release times, due dates and deadlines. The objectives are the joint minimization of makespan and maximum tardiness, which are addressed in lexicographic order. To the best of our knowledge, no published work addresses exactly the problem studied in this chapter, even though many algorithms have been proposed to solve relaxations of this problem, addressing different subsets of constraints. Among the others, we cite the parallel machine scheduling problems with setup and removal times, release and due dates [48], the vehicle routing problem with fixed number of vehicles and time windows [3, 4, 9] and the resource constrained scheduling problem [6].

The remainder of this chapter is organized as follows. In Section 3.2 a detailed description of the problem is given. A graph representation of the problem is then presented in Section 3.3 and a Tabu Search algorithm proposed in 3.4. Finally, in Section 3.5 we report on our computational experiments, carried on real industrial instances and on randomly generated instances, giving some final comments in Section 3.6

3.2 Description of the problem

The packaging department is described in Section 1.2 and in [53]. It contains three packaging lines working in parallel, each of which performs, on a single lot, all operations from the production of blisters to the final individual specific packages. Therefore, from a scheduling point of view, a line acts as a single machine and a single lot can be considered as a single job. a descriptive model of the problem is given in the following.
Machine unavailability can be viewed as a special job characterized by a release date, corresponding to its starting time, a processing time, corresponding to its duration, and a deadline corresponding to its finish time. Each maintenance operation must be executed on the prescribed machine while the lack of operators can be moved from a machine to another.

Given the description in 1.2 and the considerations above, we report below the three-fields classification scheme of Graham for this scheduling problem:

$$OMPM|r_i, d_i, D_i, R_{sd}, S_{sd}, unavail_j|Lex(C_{max}, T_{max})$$

Concerning the shop environment, $OMPM$ is the *Open shop Multi-Purpose Machines*. In fact, the department contains parallel machines but each job has its own set of machines and tools on which it can be processed and has to be assigned to one machine and one tool in its sets. Then the jobs assigned

to the same tool or machine must be sequenced. Viewing the jobs assigned to the same tool as operations of an extended job, the sequencing part of the problem is similar to an open shop problem. Instead, for what concerns the constraints of the problem, we have the presence of release times, due dates and deadlines (r_i, d_i, D_i, respectively). R_{sd} and S_{sd} indicate sequence-dependent removal times and sequence-dependent setup times. Moreover, resources (machines and operators) are not available all the time but only during well defined periods ($unavail_j$). Specifically, in our case job processing is resumable, i.e., can be interrupted when the machine or the operators become unavailable and resumed later (but no other job can be processed in between). Setups and removals are not resumable, i.e., they cannot be started if unavailability arise before completion. The optimality criteria of the problem is $Lex(C_{max}, T_{max})$, i.e. the minimization of makespan and maximum tardiness in lexicographic order.

We notice that, in the scheduling literature, even relaxed versions of this problem, such as the vehicle routing problem with release and due dates, are considered particularly difficult NP-hard problems [40]. Moreover, some constraints, such as the resumable unavailability of resources, are not frequently addressed in scheduling literature, at least in combination with others.

3.3 Graph representation of a solution

In this section, we introduce the notation used throughout the chapter and then we formally describe the constraints satisfied by feasible solutions.

Problem data consist of the following. A set of n jobs $\mathcal{J} = \{1, 2, ..., n\}$, each consisting of a single operation, must be scheduled on a set of m machines $\mathcal{M} = \{1, 2, ..., m\}$. Each job must be processed entirely on the same machine, which can process at most one job at a time. We denote with $\mathcal{M}_j \subseteq \mathcal{M}$ the set of machines *compatible* with job j, i.e. able to process j. To process job j, a machine must be equipped with a fixed number b of operators and with a tool T_j, chosen in the set of tools $\mathcal{T} = \{1, 2, ..., t\}$. We denote with $\mathcal{T}_j \subseteq \mathcal{T}$ the set of tools compatible with job j. Also, let p_j, r_j, d_j and $\tilde{d}_j$ be the processing time, release date, due date and deadline of job j, respectively.

A given *removal time* g_{ij} is required, between two jobs i and j processed consecutively on the same machine, to clean and calibrate the machine and to remove tool T_i if i and j use different tools. If $T_i \neq T_j$ an additional *setup time* f_{ij} is required after g_{ij} to set up the new tool T_j. A similar situation occurs if job i is processed on machine h', job j is processed on machine h and both use

consecutively the same tool. Letting $\beta(i)$ the job sequenced after i on machine h' and $\alpha(j)$ the job preceding j on machine h, a removal time $g_{i\beta(i)}$ is required to remove the tool from h' and a setup $f_{\alpha(j)j}$ is required to set up the tool on h. All setup/removal times satisfy the triangular inequality, i.e., $f_{ij} + f_{jk} \geq f_{ik}$ and $g_{ij} + g_{jk} \geq g_{ik}$.

The number o_i of operators available during shift $i = 1, \ldots, s$ is known in advance, and operators must be assigned to the machines for the entire duration of a shift. Therefore, at most $\left\lfloor \frac{o_i}{b} \right\rfloor$ machines can be active during shift i. We model this situation by introducing $m - \left\lfloor \frac{o_i}{b} \right\rfloor$ dummy jobs called *unavailabilities*, with release date, deadline and processing time equal to the shift start, completion and duration, respectively. These jobs will be scheduled on the machines with all the other jobs. In some cases, the subset of inactive machines for a shift is partially specified in advance, for example when preventive maintenance operations are planned for some machines during that shift.

We denote with $\mathcal{U} = \{(n+1), \ldots, (n+q)\}$ the set of all unavailabilities, with $q = \sum_{i=1,\ldots,s} \left(m - \left\lfloor \frac{o_i}{b} \right\rfloor\right)$, and with $\mathcal{M}_u \subseteq \mathcal{M}$ the set of the machines compatible with unavailability $u \in \mathcal{U}$.

Finding a schedule consists of solving five subproblems. We postpone the feasibility issue to the end of this section.

(*i*) Assign each job $j \in \mathcal{J}$ to a machine $h \in \mathcal{M}_j$. Let $\mathcal{J}^M(h)$ be the set of jobs assigned to machine $h = 1, \ldots, m$.

(*ii*) Assign each job $j \in \mathcal{J}$ to a tool $k \in \mathcal{T}_j$. Let $\mathcal{J}^T(k)$ be the set of jobs assigned to tool $k = 1, \ldots, t$.

(*iii*) Assign each unavailability $u \in \mathcal{U}$ to a machine $h \in \mathcal{M}_u$ such that unavailabilities assigned to the same machine are non-overlapping. Let $\mathcal{U}(h)$ be the set of unavailabilities assigned to machine $h = 1, \ldots, m$, and μ be the m-tuple $\mathcal{U}(1)$,$\mathcal{U}(2), \ldots, \mathcal{U}(m)$.

(*iv*) Sequence the jobs in $\mathcal{J}^M(h)$, $h = 1, \ldots, m$ and the jobs in $\mathcal{J}^T(k)$, $k = 1, \ldots, t$. Let σ_h be the resulting sequence on machine h and π_k be the resulting sequence on tool k. Let also σ be the m-tuple $\sigma_1, \sigma_2, \ldots, \sigma_m$ and π be the t-tuple $\pi_1, \pi_2, \ldots, \pi_t$.

(*v*) Define a schedule, i.e., a *starting time* S_j and a *completion time* $C_j \leq \tilde{d}_j$ for each job $j \in \mathcal{J}$ such that each machine processes at most one job/unavailability at a time. Let S and C be the vectors of starting/completion time of all jobs, $S = (S_1, S_2, \ldots, S_n)$, $C = (C_1, C_2, \ldots, C_n)$.

Let a *solution* H denote the triple $H = (\mu, \sigma, \pi)$, i.e., the output of subproblems $(i) - (iv)$, and let a *schedule* denote the pair (S, C). For each job $j \in \mathcal{J}^M(h)$ we denote with $\alpha(j)$ the job preceding j in σ_h and with $\beta(j)$ the job succeeding j in σ_h. Similarly, for each job $j \in \mathcal{J}^T(k)$ we denote with $\gamma(j)$ the job preceding j in π_k and with $\delta(j)$ the job succeeding j in π_k.

Let us now focus on the computation of a minimum makespan schedule of a solution H. In order to compute S_j and C_j we assume that the preceding jobs $\alpha(j)$ and $\gamma(j)$ (if exist) have already set $C_{\alpha(j)}$ and $C_{\gamma(j)}$. We schedule non-preemptive activities $g_{\gamma(j)\beta(\gamma(j))}$, $g_{\alpha(j)j}$ and $f_{\alpha(j)j}$ in the earliest available time periods (see Figure 3.1). Then we schedule preemptive job j, possibly splitting it in successive availability periods. Denoting with $(\cdot)$ and $[\cdot]$ open and closed intervals respectively, the earliest available starting time for the removal activity $g_{\gamma(j)\beta(\gamma(j))}$ on machine h' is

$$X_j = \min\{\tau : \tau \geq C_{\gamma(j)}, (\tau, \tau + g_{\gamma(j)\beta(\gamma(j))}) \cap \bigcup_{i \in \mathcal{U}(h')} [r_i, \tilde{d}_i] = \emptyset\}.$$

Similarly, the earliest available starting times for the removal and setup activities $f_{\alpha(j)j}$ and $g_{\alpha(j)j}$ on machine h are

$$Y_j = \min\{\tau : \tau \geq C_{\alpha(j)}, (\tau, \tau + g_{\alpha(j)j}) \cap \bigcup_{i \in \mathcal{U}(h)} [r_i, \tilde{d}_i] = \emptyset\},$$

$$Z_j = \min\{\tau : \tau \geq X_j + g_{\gamma(j)\beta(\gamma(j))}, \tau \geq Y_j + g_{\alpha(j)j}, (\tau, \tau + f_{\alpha(j)j}) \cap \bigcup_{i \in \mathcal{U}(h)} [r_i, \tilde{d}_i] = \emptyset\}.$$

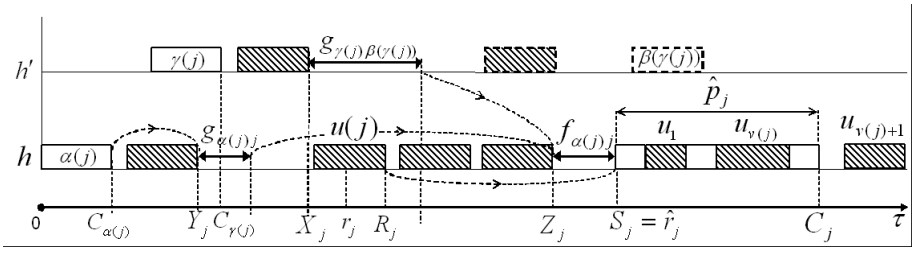

Figure 3.1: Gantt chart for machines h and h' and computation of S_j

If machine h is unavailable at the release time r_j of job j, let $u(j)$ be the last unavailability on machine h starting before r_j, i.e., such that $r_{u(j)} = \max\{r_l \leq$

$r_j, l \in \mathcal{U}(h)\}$. Then, job j cannot start before $R_j = \max\{\tilde{d}_{u(j)}; r_j\}$. Therefore, the earliest available starting time of j is:

$$S_j = \max\{R_j; Z_j + f_{\alpha(j)j}\}. \tag{3.1}$$

The earliest completion time of preemptive job j equals $S_j + p_j$ plus the total time machine h is unavailable for processing between the start and the completion of job j. Let us denote with $u_1, \ldots u_v$ the sequence of unavailabilities in $\mathcal{U}(h)$ starting after S_j, ordered for increasing values of their release times $S_j \leq r_{u_1} \leq \ldots \leq r_{u_{v-1}} \leq r_{u_v}$, and let $p_{u_1}, \ldots p_{u_v}$ be their respective processing time. Also, let $u_{v(j)}$ be the last unavailability on machine h before C_j, i.e., $v(j) = \min\{l : S_j + p_j + \sum_{i=1}^{l} p_{u_i} \leq r_{u_{l+1}}\}$. Then, the completion time of job j is:

$$C_j = S_j + p_j + \sum_{i=1}^{v(j)} p_{u_i}. \tag{3.2}$$

We denote the pair (S, C) computed according to equation (3.1) and (3.2) as the schedule associated to solution H. Let us define for job j a modified processing time $\hat{p}_j = C_j - S_j$ and a modified release time $\hat{r}_j$

$$\hat{r}_j = \begin{cases} r_j & \text{if} \quad S_j = \max\{C_{\alpha(j)} + g_{\alpha(j)j}; C_{\gamma(j)} + g_{\gamma(j)\beta(\gamma(j))}\} + f_{\alpha(j)j} \\ S_j & \text{if} \quad S_j > \max\{C_{\alpha(j)} + g_{\alpha(j)j}; C_{\gamma(j)} + g_{\gamma(j)\beta(\gamma(j))}\} + f_{\alpha(j)j} \end{cases} \tag{3.3}$$

A solution H and the associated schedule can be represented with a graph $\mathcal{G}(H) = (V, E(\sigma) \cup F(\pi) \cup A)$. V is the set of nodes, one for each job $j \in \mathcal{J}$, weighted with $\hat{p}_j$, plus four auxiliary nodes *start*, *end*, *viol_dead*, *viol_due*. $E(\sigma)$ is a set of arcs, one for each pair of consecutive nodes j and $\beta(j)$ processed on the same machine in H and weighted with $g_{j\beta(j)} + f_{j\beta(j)}$. $F(\pi)$ is a set of arcs, one for each pair of consecutive nodes j and $\delta(j)$ processed on the same tool in H and weighted with $g_{j\beta(j)} + f_{\alpha(\delta(j))\delta(j)}$. Finally, A is a set of additional arcs. For each node $j \in \mathcal{J}$ there is an arc in A from *start* to j, with weight $\hat{r}_j$, an arc $(j, end) \in A$, with weight zero, an arc $(j, viol\_dead) \in A$ if a deadline is defined for j, with weight $-\tilde{d}_j$, and an arc $(j, viol\_due) \in A$ if a due date is defined for j, with weight $-d_j$.

Figure 3.2 shows the pictorial representation of a node j and associated arcs in $\mathcal{G}(H)$, where arcs in $E(\sigma) \cup A$ are depicted with solid lines while arcs in $F(\pi)$ are depicted with dashed lines. Note that the graph structure depends on the assignment and sequencing of the jobs on tools and machines, while the arc weights also depend on the assignment of the unavailabilities to the machines.

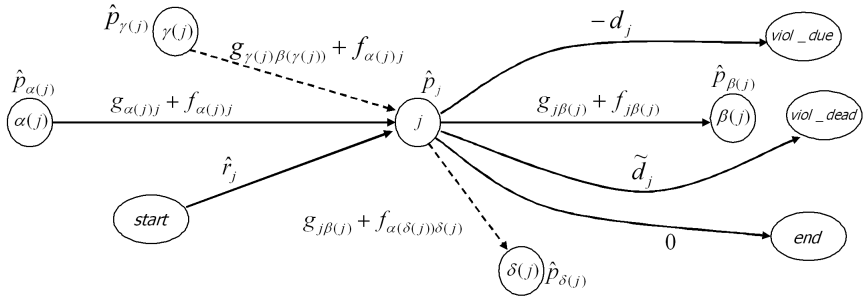

Figure 3.2: Node j and weighted sequencing arcs in $\mathcal{G}(H)$

For a given graph $\mathcal{G}$, we define the *head* $h(j)$ of node j as the length of the longest path in $\mathcal{G}$ from *start* to j (excluding the weight of node j) and the *tail* $q_a(j)$ as the length of the longest path from j to the auxiliary node $a \in \{end, viol\_dead, viol\_due\}$ (including the weight of node j). With this notation, the starting time S_j of job j in a solution is the head of the node associated to job j. Then, $h(end)$ and $\max\{0, h(viol\_due)\}$ are equal to the makespan C_{max} and the maximum tardiness T_{max} of the solution, respectively.

A solution H is feasible if each job is assigned to a compatible machine and to a compatible tool, each unavailability is assigned to a compatible machine, unavailabilities assigned to the same machine are non-overlapping, and $\mathcal{G}(H) = (V, E(\sigma) \cup F(\pi) \cup A)$ is acyclic. A schedule (S, C) is feasible if the associated solution H is feasible and $h(viol\_dead) \leq 0$, which implies $h(i) + \hat{p}_j \leq \tilde{d}_j$ for each $j \in \mathcal{J}$.

3.4 Tabu search algorithm

In this section, we describe our tabu search algorithm. Section 3.4 describes the algorithm used to find an initial solution, not necessarily feasible. Section 3.4 shows the basic moves used by the tabu search algorithm and structural properties of the neighborhood based on these moves. In Section 3.4 we define a restricted version of the neighborhood, which is used by our tabu search algorithm. Finally, in Section 3.4, we describe two procedures to evaluate the quality of a neighbor.

Greedy algorithm

A fast and simple greedy algorithm is used to obtain an initial solution. The algorithm is designed to reproduce the behavior of human scheduler when building a feasible schedule. In fact, the human schedulers in the plant do not follow any formal procedure to schedule production orders, and the schedules are simply the result of intuition and past experience. However, the schedules produced by hand are quite similar to those produced with the algorithm summarized in the following steps:

1. compute an estimate workload for each machine as follows:
   a) compute the minimum workload W_h for machine $h \in \mathcal{M}$ summing up processing time + minimum setup time + minimum removal time of all jobs that can be executed on machine h only: $W_h = \sum_{j:\mathcal{M}_j=\{h\}} (p_j + \min_i\{f_{ij}\} + \min_i\{g_{ij}\})$.
   b) compute the quantity $Q(i)$, equal to the processing time + minimum setup time + minimum removal time of all jobs that can be executed on machine h and other i machines in $\mathcal{M}$, and add to W_h the quantity $\frac{Q(i)}{i+1}$, for $i = 1, \ldots, m-1$.
2. assign human operators to machines, proportionally to the values W_h;
3. partition the time horizon into L time intervals of given length λ, and schedule the jobs in each time interval $[(l-1)\lambda, l\lambda]$, for $l = 1, \ldots, L$, according to the following algorithm:
   a) let $\mathcal{J}_l$ be the set of jobs with deadline or due-date smaller than $l\lambda$;
   b) group the jobs in $\mathcal{J}_l$ requiring the same tool and sequence the jobs in each group according to the ERD rule (Earliest Release Date first rule);
   c) sequence the groups one at a time by assigning a block to the next available machine according the SST rule (Shortest Setup Time first rule) until all groups are scheduled.

Neighborhood structure

Our tabu search algorithm makes use of two basic moves. The first one is based on the interchange of a pair of adjacent jobs sequenced on the same machine/tool, which is commonly adopted in the tabu search literature on open

shop and job shop scheduling problems [36]. The second move is based on removing a job from the sequence of jobs processed on a machine/tool and inserting it in the sequence of another machine/tool. Similar moves are commonly used when dealing with parallel machine scheduling or vehicle routing problems [9, 13].

The *interchange* move $\varphi^M(i,j)$ is defined as follows: given a solution H and a pair of consecutive jobs i and j in σ_h of machine h, a new solution H' is obtained from H reversing the precedence order (i,j). A similar move $\varphi^T(i,j)$ deals with rescheduling of consecutive jobs i and j in the sequence π_k of tool k. When two jobs are processed consecutively both on the same machine h and tool k, then a move $\varphi^{MT}(i,j)$ is applied, which exchanges their relative positions both in σ_h and π_k, in order to avoid a cycle of precedence constraints between i and j.

With the *rerouting* move $\theta^M(i,j)$, i can be either a job or an unavailability while j is a job. Moreover, i and j are processed on different machines h and h', respectively, with $h' \in \mathcal{M}_i$. If i is a job, this move consists of removing i from σ_h and inserting it before j in $\sigma_{h'}$. If i is an unavailability, the move consists of removing i from machine h and inserting it in the same machine of j in its given time window. Similarly, move $\theta^T(i,j)$ is applied to jobs i and j processed on different tools k and k', respectively, with $k' \in \mathcal{T}_i$.

Acyclicity property

A solution H is feasible only if the corresponding graph $\mathcal{G}(H)$ is acyclic. Our tabu search algorithm does not perform moves leading to cyclic graphs. To this aim, a cyclicity test is preliminarily checked before each move and the move is removed from the neighborhood if it leads to a cycle. The test can be performed in time $O(n)$ using the Bellman's algorithm, but we next show that this check can be performed in constant time if the conditions of Theorems 3.4.1 or 3.4.2 hold.

In the following we call critical path of $\mathcal{G}(H)$ one of the longest paths from *start* to one of the auxiliary nodes and denote H the incumbent solution, H' the solution obtained after a move, $h(i)$ the head of node i in H and $h'(i)$ the head of node i in H'.

Theorem 3.4.1 *Given an arc (i,j) on the critical path of $\mathcal{G}(H)$, the interchange moves $\varphi^M(i,j)$, $\varphi^T(i,j)$, and $\varphi(i,j)^{MT}$ do not create cycles.*

Proof. Let us suppose that there is a cycle in $\mathcal{G}(H')$ after a move $\varphi(i,j)$.

Thus, there must be a path in $\mathcal{G}(H)$ from i to j, disjoint from arc (i, j). Let us consider the three moves separately.

- $\varphi^{MT}(i, j)$ move: in this case $j = \beta(i) = \delta(i)$ and there are no other arcs outgoing from i, other than those ingoing in j or in the auxiliary nodes. Therefore, there is no path from i to j, disjoint from arc (i, j) in $\mathcal{G}(H')$, a contradiction.

- $\varphi^{M}(i, j)$ move: in this case $j = \beta(i)$. Let us suppose that a path p from i to j exists in $\mathcal{G}(H)$, besides arc (i, j). This path must include at least arcs $(i, \delta(i))$ and $(\gamma(j), j)$, which are the only arcs outgoing from i and ingoing in j other than (i, j) and the arcs of set A. There are only two possibilities: either $T_i \neq T_j$ or $T_i = T_j$. In both cases, path p includes at least the removal of tool T_i, the setup of tool T_j and the processing time $p_{\delta(i)} > 0$. If $T_i = T_j$ the length of p is strictly larger than the length of arc (i, j). If $T_i \neq T_j$ it follows from the triangular inequality that the weight of all setups occurring in p is larger or equal to $g_{ij} + f_{ij}$. Therefore, the weight of p is strictly larger than the weight of the critical arc (i, j), a contradiction.

- $\varphi^{T}(i, j)$ move: in this case $j = \delta(i)$. Let us suppose that a path p from i to j exists in $\mathcal{G}(H)$, besides arc (i, j). This path must include at least arcs $(i, \beta(i))$ and $(\alpha(j), j)$, which are the only arcs outgoing from i and ingoing in j other than (i, j) and the arcs of set A. Also in this case, if follows from the triangular inequality that the length of path p is greater than the weight of the critical arc (i, j), a contradiction.

In conclusion, path p cannot exist if the arc (i, j) is critical, and the thesis follows. □

As far as the rerouting moves are concerned, let us first observe that there can be a cycle in $\mathcal{G}(H')$ after move $\theta^{M}(i, j)$ if and only if there is a path in $\mathcal{G}(H)$ from j to $\gamma(i)$ or from $\delta(i)$ to $\alpha(j)$. We notice that, if $j = \beta(i)$ the acyclicity of $\mathcal{G}(H)$ implies that of $\mathcal{G}(H')$. Otherwise, a sufficient condition for the acyclicity of $\mathcal{G}(H')$ is given by the following theorem. The proof directly follows from results proved in [11].

Theorem 3.4.2 *After a move $\theta^{M}(i, j)$, if $h(j) + \hat{p}_j > h(\gamma(i))$ and $h(\delta(i)) + \hat{p}_{\delta(i)} > h(\alpha(j))$ then $\mathcal{G}(H')$ is acyclic.*

Similar conditions hold for $\theta^T(i,j)$. When sufficient conditions do not hold the algorithm explicitly checks the existence of a path from i to j on $\mathcal{G}(H)$, besides (i,j). If such a path exists, the move is removed from the neighborhood.

Connectivity

In this section, we show that the neighborhood defined by moves $\varphi(i,j)$ and $\theta(i,j)$ is connected, i.e., we prove that it is possible to reach a global minimum starting from any feasible solution. Our proof is similar to that reported in [11]. Let A be a resource assignment, i.e., the definition of sets $\mathcal{J}^M(h)$, $\mathcal{U}(h)$ and $\mathcal{J}^T(k)$ for each machine $h \in \mathcal{M}$ and each tool $k \in \mathcal{T}$. Let H_A [respectively, H_A^*] be a generic [an optimal] solution associated to A. H_A [respectively, H_A^*] defines the sequences [the optimal sequences] σ_h and π_k of elements in $\mathcal{J}^M(h)$ and $\mathcal{J}^T(k)$ for each $h \in \mathcal{M}$ and $k \in \mathcal{T}$. We also call $\mathcal{G}(H)$ the graph representation of H and $\mathcal{G}(\overline{H})$ its transitive closure, i.e., the graph obtained from $\mathcal{G}(H)$ including all the redundant arcs. A first result is the following:

Lemma 3.4.1 *Given a resource assignment A, an optimal sequencing H_A^* and a solution H_A, if H_A is not optimal there is an arc (i,j) such that (i,j) is critical in $\mathcal{G}(H_A)$ and such that $(j,i) \in \mathcal{G}(\overline{H}_A^*)$.*

Proof. Let us suppose that all the critical arcs of $\mathcal{G}(H_A)$ also belong to $\mathcal{G}(\overline{H}_A^*)$ as well. Therefore all the critical paths of $\mathcal{G}(H_A)$ also belong to $\mathcal{G}(\overline{H}_A^*)$, thus implying that H_A^* is not optimal, a contradiction. Hence, there must be at least a critical arc $(i,j) \in \mathcal{G}(H_A)$ that does not belong to $\mathcal{G}(\overline{H}_A^*)$, i.e., such that $(j,i) \in \mathcal{G}(\overline{H}_A^*)$. □

From this lemma the following theorem can be proved.

Theorem 3.4.3 *Given a resource assignment A and a solution H_A, if H_A is not optimal there is a sequence of interchange moves leading from H_A to an optimal solution H_A^*.*

Proof. The proof directly follows from Lemma 3.4.1 and Theorem 3.4.1. Starting from H_A and given an optimal solution H_A^*, Lemma 3.4.1 guarantees that there exists a pair of consecutive jobs that can be reversed, and Theorem 3.4.1 guarantees that the resulting solution, say $\tilde{H}_A$, is feasible. If $\tilde{H}_A$ is optimal the thesis follows, otherwise the same procedure can be repeated starting from $\tilde{H}_A$, finally leading to an optimal solution. □

We have shown how an optimal sequencing for a given assignment A can be reached starting from any solution. We next show that an optimal assignment can be reached starting form any assignment. To this aim, we prove the following preliminary result.

Theorem 3.4.4 *Given a solution H, a machine h, a tool k and a job i, compatible with machine h [with tool k], if $i \notin \mathcal{J}^M(h)$ [if $i \notin \mathcal{J}^T(k)$] there always exists a rerouting move that moves i to $\mathcal{J}^M(h)$ [respectively, to $\mathcal{J}^T(k)$] leading to a feasible solution.*

Proof. Theorem 3.4.2 provides sufficient conditions for acyclicity after a rerouting move. Therefore, to demonstrate the thesis, it is sufficient to show that, if job i is compatible with machine h [with tool k], and given σ_h [given π_k], there are always in σ_h [in π_k] two consecutive jobs l and j such that i can be inserted between l and j.

We limit ourselves to prove this for move $\theta^M(i,j)$ only, the proof for $\theta^T(i,j)$ being very similar. First observe that every arc in σ_h always satisfies at least one of the two conditions in theorem 3.4.2. In fact, if the first condition is not true, then $h(j)+\hat{p}_j \leq h(\gamma(i))$. It follows that $h(l) < h(j)+\hat{p}_j \leq h(\gamma(i)) < h(\delta(i))+\hat{p}_{\delta(i)}$. Similarly, if the second condition is not true, then $h(\delta(i))+\hat{p}_{\delta(i)} \leq h(l)$, which implies $h(j)+\hat{p}_j \geq h(l)+\hat{p}_l+\hat{p}_j > h(l) \geq h(\delta(i))+p_{\delta(i)} > h(\gamma(i))$. In particular the second condition is always satisfied for node *start* and the first for at least an auxiliary node $i \in \{end, viol\_dead, viol\_due\}$. Note also that i cannot satisfy the second condition. Then, in σ_h there must be one last node l such that $h(\delta(i))+\hat{p}_{\delta(i)} > h(l)$ and $h(\delta(i))+\hat{p}_{\delta(i)} \leq h(\beta(l))$. Therefore, $j=\beta(l)$ must satisfy the condition $h(j)+\hat{p}_j > h(\gamma(i))$, which concludes the proof. □

Theorem 3.4.5 *The neighborhood defined by moves φ and θ is connected.*

Proof. Consider an optimal solution H^*. Theorem 3.4.3 shows that starting from any solution H_A it can be reached an optimal sequencing H_A^* for the given assignment A. If H_A^* is not globally optimal, there must be at least a job i which is assigned to a different resource in H_A^* and H^*. Theorem 3.4.4 guarantees that there exists a rerouting move that moves i to the same machine as in H^*, thus leading to a new feasible solution H'. If H' is optimal the thesis follows, otherwise the same procedure can be repeated starting from H', finally leading to an optimal solution. □

Neighborhood exploration

At each step of the algorithm we restrict the interchange move to consecutive jobs (i and $\alpha(i)$ or $\gamma(i)$) on a critical path, i.e., a longest path from *start* to one of the other auxiliary nodes. This is a common strategy, e.g., when solving job shop scheduling problems [36]. Specifically, we analyze a critical path from *start* to *viol_dead* when $h(viol\_dead) > 0$. Otherwise, we explore the paths from *start* to *end* and *viol_due*.

When dealing with the *rerouting* move $\theta^M(i,j)$, we can move either jobs or unavailabilities. In the former case, we restrict the choice of i to the jobs positioned on a critical path and j to those nodes such that one of the two following conditions is satisfied:

- there is a strictly positive slack time $\Delta_j > 0$ before j, on the machine processing it, during which the machine is available and not busy;
- $j = \delta(i)$, i.e., j is the job using the same tool of i, immediately after it.

When the number of operators in a shift is smaller than b times the number of machines, we also allow to move unavailabilities. We say that an unavailability u is *critical* if one of the two following conditions holds: (1) there is a node j on a critical path of $\mathcal{G}(H)$ and u interrupts the processing of job j; (2) $\hat{r}_j > r_j$ and $\tilde{d}_u$ equals one of the quantities X_j, Y_j, Z_j, R_j defined in Figure 3.1. We allow moving an unavailability i with move $\theta^M(i,j)$ only if i is a critical unavailability.

Critical paths and extended critical paths

We define two different critical paths. One is the classical longest path on $\mathcal{G}(H)$ from *start* to a specific auxiliary node, i.e., the path for which the sum of the arc weights is maximum. This path might contain a small number of arcs when its first arc is a modified release time $\hat{r}_j > r_j$. In this case $\hat{r}_j$ is computed according to (3.3) and can take into account the sequencing of a large number of jobs. Therefore, the starting time of job j might be reduced as well by anticipating the starting time of job $\alpha(j)$, $\gamma(j)$ or by rerouting one of these two jobs since their modification can affect the value of $\hat{r}_j$. In other words, incorporating in the critical path a longest path from *start* to node j different from arc $\hat{r}_j$, would allow a larger possibility to improve upon the incumbent solution. More formally, let us consider an ordinary longest path from *start* to any auxiliary node $a \in \{end, viol\_dead, viol\_due\}$, and let j be the first node

after *start* on this path. The extended critical path is recursively defined as the path including all the nodes on the longest path plus the extended critical path from *start* to j. The latter quantity is the empty set if $S_j = R_j$ holds in equation (3.1). If $S_j > R_j$ then, if $Y_j + g_{\alpha(j)j} \geq X_j + g_{\gamma(j)\beta(\gamma(j))}$ the extended critical path includes the longest path from *start* to $\alpha(j)$, otherwise it includes the longest path from *start* to $\gamma(j)$. These paths are evaluated iteratively with the same procedure.

Move evaluation

In the neighborhood of the incumbent solution H we look for a solution H' minimizing the following penalty function.

$$f(H') = b*max\{0, h(viol\_dead)\} + c*h(end) + d*max\{0, h(viol\_due)\}. \quad (3.4)$$

We developed two alternative algorithms for evaluating $f(H')$ after a move. The first one gives an estimate of $f(H')$ in constant time, while the second provides the exact value of $f(H')$ in linear time. At each step of the tabu search, the neighbor with the smallest value of $f(H')$ is selected as the new incumbent, and the schedule is updated accordingly. Then, according to the equations (3.1) and (3.2) of Section 3.4, job starting and completion times are updated in $\mathcal{G}(H)$. We next describe the two evaluation algorithms.

Approximate evaluation

The approximate evaluation of $f(H')$ consists of using heads and tails of $\mathcal{G}(H)$ to estimate the length of a longest paths from *start* to an auxiliary node *end*, *viol_dead*, or *viol_due*. We recall the notation $h(i)$ and $q_a(i)$ [respectively, $h'(i)$ and $q'_a(i)$] to define the length of the longest path in $\mathcal{G}(H)$ [respectively, in $\mathcal{G}(H')$] from node start to i and from i to node $a \in \{end, viol\_dead, viol\_due\}$. With respect to move $\varphi(i,j)$, the estimate L'_a of the length of a longest path in $\mathcal{G}(H')$ from *start* to auxiliary node a is computed as an estimate of the longest path passing through i or j:

$$L'_a = max\{h'(i) + q'_a(i), h'(j) + q'_a(j)\}. \quad (3.5)$$

As for the solutions obtained with the $\varphi^M(i,j)$ move, we have (see Figure 3.3):

$$\begin{aligned}
h'(j) &= \max\{Y_i + g_{\alpha(i)j} + f_{\alpha(i)j}; X_j + g_{\gamma(j)\beta(\gamma(j))} + f_{\alpha(i)j}; R_j\} \\
h'(i) &= \max\{h'(j) + p_j + g_{ji} + f_{ji}; X_i + g_{\gamma(i)\beta(\gamma(i))} + f_{ji}; R_i\} \\
q'_a(i) &= \max\{p_i + g_{i\beta(j)} + f_{i\beta(j)} + q_a(\beta(j)); p_i + g_{i\beta(j)} + f_{\alpha(\delta(i))\delta(i)} + q_a(\delta(i))\} \\
q'_a(j) &= \max\{p_j + g_{ji} + f_{ji} + q'_a(i); p_j + g_{ji} + f_{\alpha(\delta(j))\delta(j)} + q_a(\delta(j))\}.
\end{aligned}$$

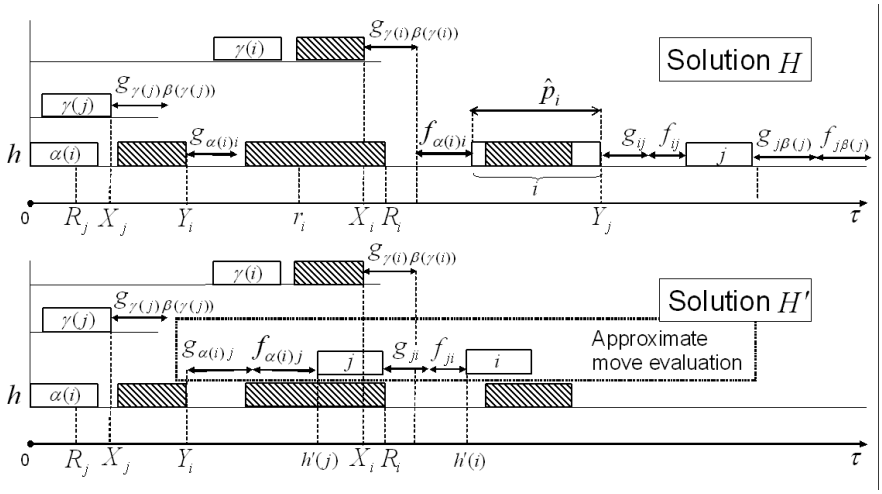

Figure 3.3: Approximate evaluation of move $\varphi^M(i,j)$.

These four quantities can be computed in constant time using the information available from $\mathcal{G}(H)$. For moves $\varphi^T(i,j)$, $\varphi^{MT}(i,j)$ there are similar equations. As for moves $\theta^M(i,j)$ [respectively, $\theta^T(i,j)$] we estimate the length of two longest paths: the one passing through the former nodes $\alpha(i)$ and $\beta(i)$ [respectively, $\gamma(i)$ and $\delta(i)$] and the one passing through i in $\mathcal{G}(H')$. Also these quantities can be computed in constant time using the information available from $\mathcal{G}(H)$.

Exact evaluation

We first notice that when changing the sequences σ and π, also the arc weights in $\mathcal{G}(H)$ may change. For this reason, it is necessary re-computing the exact values for the starting/completion times of each job after each move. With our exact evaluation strategy we compute these exact values not only after the application of a move, but also for evaluating the neighborhood. We next show that this computation requires linear time if the nodes of $\mathcal{G}(H')$ are numbered according to a topological order.

Theorem 3.4.6 *The exact evaluation of function $f(H')$ can be computed in time $O(n+q)$.*

Proof. Let $TO(j)$ be the position of node j in a topological order of the nodes of $\mathcal{G}(H')$. We notice that, for both kinds of moves $\varphi(i,j)$ and $\theta(i,j)$, the starting times of the nodes preceding $\min\{TO(i), TO(j)\}$ remain unchanged when passing from H to H'. If we are moving an unavailability, updating the values R_j for all $j \in \mathcal{J}$ requires linear time if the unavailabilities and the jobs are ordered for increasing values of r_j. Then, we consider the jobs in topological order. For each job j to be processed on machine h, we compute its starting time S_j by first scheduling the removal time $g_{\alpha(j)j}$ on machine h. If the removal time can be accommodated between the completion of job $\alpha(j)$ and the start of the next unavailability in $\mathcal{U}(h)$ this position is accepted, otherwise the value Y_j is computed by scanning the list of unavailabilities in $\mathcal{U}(h)$. Similar computation is necessary to compute X_j and to position the removal time $g_{\gamma(j)\beta(\gamma(j))}$ on the machine processing job $\gamma(j)$ and then the setup time $f_{\alpha(j)j}$. Finally, S_j can be computed as in equation (3.1). Given S_j, the completion time C_j can be computed in linear time according to equation (3.2) by computing the value $v(j)$ and inserting the unavailabilities in $\mathcal{U}(h)$ one at a time for increasing release time, starting from S_j. Note that the whole procedure requires a sequence of elementary steps, each requiring constant time. At each elementary step either: (i) a removal or a setup is positioned in the schedule, or (ii) the starting/completion time of a job is defined, or (iii) it is decided that some unavailability precedes one of the previous values. This unavailability is not considered again in subsequent computations. Therefore, the overall evaluation requires a total of $O(n+q)$ elementary steps, and the thesis follows. □

3.5 Computational Experiments

In this section we describe the performance of four different versions of our tabu search algorithm, developed by varying the size of the neighborhood and the evaluation strategy. We evaluate either the classical critical path (option A) or the extended one (option B) described in Section 3.4. We estimate the penalty function $f(H)$, for each H in the neighborhood of the incumbent, by choosing either the approximate evaluation (option C) or the exact one (option D), as described in Section 3.4. We have, therefore, four versions of the algorithm for the pairs of options A-C, A-D, B-C and B-D.

Each version depends on two parameters, namely the tabu list length λ and the number ν of non-improving moves examined before applying a restart strategy. Our restart strategy consists of moving one unavailability from a

machine to another, randomly chosen. This kind of move strongly disrupt the schedule since it generates an overload equal to the duration of one shift in one machine and a big slack in the other. For each version of the algorithm, λ and ν are calibrated with the procedure CALIBRA, described in [1], for varying λ in the interval $[5, 20]$, and ν in the interval $[100, 1000]$. CALIBRA uses the Taguchi methodology [44] for fractional factorial experimental designs coupled with a local search procedure to estimate the best values of all parameters. The resulting pairs (λ, ν) for each version of our tabu search algorithm are: (9,876) for A-C, (16,876) for A-D, (7,876) for B-C and (18,876) for B-D version.

We fixed the values $b = 10^{10}$, $c = 10^5$ and $d = 1$ in the objective function $f(H)$ for all experiments. This corresponds to giving maximum priority to the respect of deadlines and then considering makespan and tardiness in lexicographical order, since a tardiness of two weeks is penalized less than increasing the makespan by one minute. In Section 3.5 we compare the four versions of our tabu search algorithm and draw several conclusions. All versions of the algorithm were coded in C language and were run on a Pentium® 4, 3.0 GHz with 1024 MB Ram. Two different sets of problem instances, described in the following section, were generated: the first set comes from the industrial practice and consists of 2 real instances and 24 realistic instances divided into three groups of easy, medium and hard instances. Realistic instances represent a wide range of possibilities that can arise in practice. For the real instances we compare the actual performance attained at the plant when scheduling by hand and by computer. For the realistic instances we compare the performance of our algorithms with that of the greedy algorithm described in Section 3.4, which was designed to perform similarly to the human schedulers of the plant. The second set of instances is randomly generated, in order to evaluate the performance of our algorithm in a more general context.

Data set description

The two real instances, correspond to the real production plan for several weeks of production during September and October 2006. The first instance concerns with 18 days of production during which 26 production orders are scheduled. The second instance concerns with 16 days of production during which 19 production orders are scheduled. In both instances, there are no urgent orders (i.e., no deadlines). All the jobs are available from the first day and the due dates are fixed equal to the end of the last working day. Operators availability in each week allows to activate three blister lines from Monday to Friday for two consecutive shifts of 7 hours in each day, plus a short shift of three hours

in which two blisters can be activated from Tuesday to Friday only.

The second set of 24 realistic instances is divided into three groups with 8 instances in each group. The first group contains easy instances. There are no urgent orders, the release dates [the due dates] coincide with the start of the first week [the end of the second week] for all jobs and the total processing times of all jobs (with the exclusion of removal and setup times) is approximately 50% of the department capacity. The second group contains instances of medium level of difficulty. There are no urgent orders but the release dates and the due dates may vary over the two weeks and the workload is slightly larger. The third group contains hard instances, with urgent orders, variable release/due dates and larger workload. Instances of this kind may arise when a disruption occurs in a different plant and its production is redirected to other plants, thus causing a larger workload and a number of urgent orders.

The second data set consists of 120 random instances, obtained by generating 10 instances for each pair (n, m), with the number of jobs n varying in the set $\{20, 40, 60, 100\}$, and the number m of machines varying in $\{2, 3, 4\}$. Processing times, release dates, due dates, deadlines, setups and removal times are also randomly generated, as well as the tool assignment for each job. The total workload for each instance is approximately equal to 90% of the department capacity.

Computational results

In this section we report on our computational experience on the real, realistic and random instances described in Section 3.5.

Real and realistic instances

In Table 3.1 we report on the results achieved by the four version of our tabu search. In columns 2 and 3 the makespan (in hours) and the maximum tardiness (in hours) for the manual solutions are shown and in the subsequent columns the same values are reported for the four configurations A-C, A-D, B-C and B-D. The first two lines refer to the two real instances while the following three lines of the table to the realistic instances (in each line, the average over the 8 instances is reported). For the two real instances, the greedy algorithm of Section 3.4 achieves in both instances $T_{max} = 0$, While $C_{max} = 393.00$ and $C_{max} = 443.00$, respectively for the first and the second instance. These values are quite similar to those obtained manually, so the greedy algorithm can be used as a surrogate of the human schedulers. Therefore, column "Manual" of

Table 3.1 reports results obtained by the human schedulers of the plant for the real instances (rows 1 and 2), and the computation time is the time actually taken by the operators to compute a solution by hand. For the realistic instances (rows E, M and D, corresponding to easy, medium and difficult instances, respectively) we don't have results from human schedulers so we use the greedy algorithm as a surrogate of their performance, and in column $Time$ we report the computation time of the greedy algorithm. In the following five lines we report the average computation time required and the standard deviation of the makespan C_{max} for each group of instances and each algorithm.

Table 3.1: Comparison between manual and computerized schedules

	Manual		A-C		A-D		B-C		B-D	
Inst.	C_{max}	T_{max}	C_{max}	T_{max}	C_{max}	T_{max}	C_{max}	T_{max}	C_{max}	T_{max}
1	408.75	0	380	0	377	0	377	0	377	0
2	440.25	0	424.25	0	424	0	424	0	424	0
E	255.38	0	249.38	0	240.14	0	249.63	0	238.50	0
M	368.31	77.31	256.13	13.56	248.63	0	254.56	0	248.88	0
D	381.13	90.13	267.69	0	266.94	0	270.19	0	264.63	0
	σ	$Time$	σ	$Time$	σ	$Time$	σ	$Time$	σ	$Time$
1	–	> 3600	–	4	–	3	–	13	–	1
2	–	> 3600	–	< 1	–	1	–	< 1	–	1
E	5.12	< 1	6.66	11.63	8.56	10.29	3.70	13.38	8.02	11.75
M	21.43	< 1	7.85	11.25	7.62	14.38	10.15	16.63	11.21	16.00
D	17.97	< 1	9.04	15.00	8.95	21.13	6.37	15.13	7.65	18.88

The four versions of the tabu search algorithm clearly outperform the manual schedules, with respect to both makespan and tardiness. Specifically, for what concern the makespan C_{max}, the improvement achieved by the tabu search with respect to the manual/greedy algorithm is always quite larger than the standard deviation. For what concern T_{max}, the greedy algorithm violates the due dates in the 16 medium and hard instances, the tabu search version A-C violates the due dates in only one medium instance, the other three versions do never violate the due dates. For what concern the violation of the deadlines (not reported in table), the greedy algorithm violates the deadlines in the eight hard instances while the four versions of the tabu search do never violate the deadlines.

Random Instances

The second set of instances consists of 120 random instances described in Section 3.5. Table in Figure 3.4 shows the performance of the four configurations of our tabu search algorithm. For each version of the algorithm, columns C_{max} and T_{max} report the average values of makespan and maximum tardiness, respectively, on the 10 instances associated to each pair (n, m). Column σ shows the standard deviation of C_{max} over the 10 instances.

		A-C			A-D			B-C			B-D		
n	m	C_{max}	T_{max}	σ	C_{max}	T_{max}	σ	C_{max}	T_{max}	σ	C_{max}	T_{max}	σ
20	2	252.32	0	17.96	243.43	0	18.87	251.98	0	16.53	243.51	0	19.61
40	2	251.48	0.19	10.23	243.45	0	10.49	251.66	0	10.42	243.26	0	10.46
60	2	247.69	0	11.83	238.38	0	11.81	247.36	0	11.00	238.87	0	10.90
100	2	272.85	28.01	8.53	268.49	0	9.10	273.27	26.94	8.21	268.17	0	9.22
20	3	272.84	0	45.70	256.80	0	39.08	272.11	0	45.56	256.35	0	40.06
40	3	258.94	0	11.46	250.38	0	11.63	261.19	0	11.09	249.97	0.76	10.93
60	3	253.13	0	14.11	246.16	0	14.36	255.19	0	14.30	245.69	0	14.82
100	3	260.08	8.21	9.52	253.38	0	10.01	260.23	1.49	9.82	253.54	0	9.87
20	4	256.77	0	16.39	251.57	0	20.64	260.82	0	29.62	250.75	0	19.20
40	4	255.32	3.32	10.80	245.66	0	11.91	254.69	0	11.13	245.01	0	11.73
60	4	241.70	0	14.89	233.61	1.14	13.97	244.04	0	13.31	233.36	0	14.78
100	4	253.17	0	11.66	247.73	0	9.98	253.31	0	10.10	246.68	0	10.92
Average		256.36	3.31	15.26	248.25	0.10	15.15	257.15	2.37	15.92	247.93	0.06	15.21

Figure 3.4: Performance of the tabu search algorithm for varying n and m

A few comments are in order. Computing the exact value of the objective function (option D) for all neighbors provide much better results than using an approximate evaluation (option C). In particular, combination B-D outperforms the others with respect to both makespan and maximum tardiness.

Using extended paths provides slightly better results than using the classical paths in combination with the exact evaluation. When using the approximate evaluation, better performance are provided by the classical path. In other words, exact evaluation performs better with a larger neighborhood, while approximate evaluation performs best with a smaller neighborhood, the improvements being reached after several restart phases.

It is interesting analyzing the time needed by the algorithm to reach the best solution found within one minute of computation time. With combinations A-C and A-D the best solutions are found after 7.53 and 9.97 seconds on average,

respectively. With combinations B-C and B-D the best solutions are found after 6.83 and 10.03 seconds on average, respectively. We notice that despite the larger number of solutions to explore in each iteration using the extended path requires approximately the same computation time to find the best solution.

3.6 Conclusions

In this chapter, we studied a practical scheduling problem arising in the packaging department of a pharmaceutical production plant. The problem is formulated as a multi-purpose machine scheduling problem with additional constraints. A tabu search algorithm is able to find good solutions within short computation time, compared to the solutions found by hand and by a greedy algorithm. The makespan reduction is between 3.6% and 7.5% for easy instances and increases to more than 30% for hard instances, which is a remarkable improvement in the productivity of the plant. More important, when analyzed by the plant managers the solutions were considered feasible in practice. The results achieved confirm that scheduling technology is mature to solve complex real problems. To this aim, however, it is important to face the complexity of practical scheduling problems by using detailed scheduling models.

Chapter 4

Coordination of production scheduling and distribution in a pharmaceutical plant

In many industrial settings production scheduling decisions are weakly coordinated. The flow of information is often unidirectional and limited to the output of one department which determines input data and constraints to be satisfied by the subsequent department. In this chapter we report on a practical case study showing that closer coordination provided by fast automated scheduling tools can generate significant cost savings. We focus on coordination between the packaging of final products and their distribution to wholesalers, i.e the coordination between Packaging and Distribution department of a pharmaceutical plant. The scheduling problem is analyzed in the previous chapter while the distribution problem can be formulated as a vehicle routing problem with soft time windows. The combined scheduling and delivery problem, with the objective of minimization of total production and distribution costs, is considered and two resolution approaches are proposed, i.e. a decentralized and a centralized approach. As well as for the scheduling problem, we adapt a classic tabu search algorithms for the resolution of both the distribution problem in the decentralized approach and the combined problem in the centralized approach. Computational experiments show that good solutions to difficult instances of the combined problem can be obtained within less than a minute of computation with both approaches. The overall performance of the centralized approach significantly improves that of the decentralized approach.

4.1 Introduction

In the last decades there is an increasing research activity on coordination of multiple decisions in supply chain management as well as among different stages of a production system. The need for coordination arise from the observation that decisions made in a stage of the supply chain influence decisions made in the other stages. Moreover, decisions that are locally the most convenient for a particular stage can force the other stages to choose from a set of decisions not very convenient for them. Coordinated decisions avoid the generation of inconvenient constraints from a stage versus the following one, and allow the following stage to choose from a better set of decisions than in an uncoordinated scenario. In other words, coordination enables to take decisions that are globally more convenient for the whole system.
Coordination issues are particularly important in the pharmaceutical supply chain, in which the legal and economical implications of product stockout requires the adoption of standards of product quality and availability close to 100%. Availability of final products requires not only to achieve excellence at each stage of the planning process, from strategic planning to real time scheduling and delivery, but also in the coordination between different stages.
The lack of coordination in a supply chain is mainly due to the variety of actors with conflicting objectives that compete and cooperate within the supply chain. Nevertheless, even within the same company decisions can be scarcely coordinated for several reasons, e.g., because they are taken by different decision makers at different time points of the process and by considering only a subset of all the constraints and factors affecting the overall company costs. This is the case, for example, of production planning and scheduling. Scheduling in a single department strongly depends on the production plan as well as on scheduling in other departments. Unfortunately, production planning and scheduling, as well as single departments, are usually loosely integrated. Usually, the presence of large stocks between two adjacent departments compensate for the lack of coordination between them; anyway, some benefits, as the reduction of stocking costs, could be obtained by introducing a stronger coordination. The need of coordination is more evident in departments that operates as pull systems, i.e. systems where the operations in a stage are guided by a request of subsequent stages. This is the case of the packaging and distribution levels. This stages are guided by wholesalers orders that can dynamically vary. The two stages may have different response to these variation, then, the decisions taken in the packaging stage result in constraints for the distribution phase that may cause bad solutions. In such cases, coordination

may produce significant savings because the overall objective of both stages is to reduce the overall company costs.

In this chapter we address a combined production scheduling and distribution problem arising in a pharmaceutical production plant. We refer in particular to the coordination between the packaging and the distribution of finished products. The packaging stage is the last manufacturing stage of the production process. Scheduling decisions are taken on a weekly basis independently from the subsequent distribution stage. However, the decisions taken result in constraints for the distribution department that might cause delays or extra costs in the delivery of some products. Therefore, closer coordination between the two departments might significantly improve the overall company performance.

The production scheduling problem is described in details in Chapter 3, while the distribution problem is a version of the vehicle routing problem with soft time windows that takes into account multiple deliveries to the same customer in the same or in different shipments. The overall objective function is the minimization of total production and distribution costs. The purpose of this study is to quantify the benefits of coordination for this difficult practical case study.

Two different resolution approaches are presented: a decentralized approach, in which the production scheduling problem is solved independently from the vehicle routing problem, and a centralized approach, in which the two subproblems are solved simultaneously. The performance of the two approaches are then compared in order to evaluate the benefits of replacing the currently used decentralized approach with centralized optimization.

The remainder of this chapter is structured as follows. After a brief introduction in Section 4.1, in Section 4.2 we present a review of literature on supply chain coordination and scheduling. In Section 4.3 we define the distribution problem and a Tabu Search algorithm to solve it. Section 4.4 introduces the combined problem and the two different resolution approaches: decentralized and centralized. The computational experiments follow in Section 4.5.

4.2 Scheduling and delivery in literature

The coordination between different stages at the operational level is still an under-researched topic in the academic literature, but is critical to the success-

ful implementation of planning activities, in particular for the pharmaceutical industry. An integrated approach to planning and scheduling in the pharmaceutical industry can be found in [50], where a hierarchically structured moving horizon framework is proposed. The problem of integrating the production schedules of two departments in the production process of a furniture manufacturer has been studied by Agnetis et al [2]. The two departments rank the jobs on the basis of their color and size respectively, but a common job sequence that trades off the two departments' objectives is needed.

The literature on integrated production scheduling and delivery is analyzed by several authors. In the survey of Sarmiento and Nagi [47] various cases are analyzed in which two or more different functions (supply process, distribution, inventory management, production scheduling ...), are integrated into a single optimization model. In particular, papers on scheduling-inventory-distribution integration are classified into different categories (single/multiple supply location, deterministic and stochastic, routing and no-routing models). The benefit of an integrated approach is demonstrated in many cases. Supply chain scheduling is discussed in [21]. An arborescent supply chain is considered with a supplier making deliveries to several manufacturers that, in turn, serve several customers. Each actor is considered as a single machine and jobs are delivered in batches. Also in this case, cooperation results in significant total cost reduction. Routing decisions are considered in [8] where different cases of scheduling and distributions are examined, with the aim to both maximize a function of customer service level and to minimize a function of total delivery costs. Two different machine configurations, single and parallel, and two situations for customers, single or multiple, are considered. Problems are divided into two classes corresponding to two different measures of customer service level: average delivery time in the first class and maximum delivery time in the second one. Exact algorithms are provided for the easiest problems, while intractability proofs and heuristic algorithms are proposed for more difficult problems. Mak and Wong [27] presents a genetic algorithm to solve a complex integrated inventory-production-distribution problem.

Tardiness or lateness issues are the main goals in the literature on newspapers distribution [24, 46]. Hurter and Van Buer [24] study a problem of coordination between the printing (i.e., production) and distribution departments in a newspaper company. The authors study the tradeoff between the conflicting preferences of the two departments. Russell et al. [46] presents a tabu search metaheuristic for the coordination of the production and delivery of newspapers. The integrated approach significantly improves the performance with respect to existing operations.

4.3 The distribution problem

The distribution problem VRP can be modeled as a Vehicle Routing Problem with soft Time Windows and Multiple Deliveries. A set of n jobs $\mathcal{J} = \{1, 2, ..., n\}$, each corresponding to a single production order, must be delivered to a set of c customers $\mathcal{C} = \{1, 2, ..., c\}$, from a single depot referred to as the customer 0. A release time e_j and a due date d_j are associated to job $j \in \mathcal{J}$. Each job must be delivered by a single vehicle to a single customer. We denote with $C_j \in \mathcal{C}$ the customer of job j. Differently from the most common definitions of time windows for the vehicle routing problem, in our case the release time of a job if related to the completion of its packaging operation while the due date is related to its delivery date. Therefore, the vehicle carrying job j cannot depart from the depot before e_j. The tardiness of job j delivered at time t to C_j is $\max\{0, t - d_j\}$. For some jobs a strict deadline can be specified when a late delivery would cause a stockout at the final customer. We denote as $\tilde{d}_j$ the deadline of job j when specified. To each pair of customers (h, k) is associated a travel time t_{hk}. A set of r vehicles $\mathcal{V} = \{1, 2, ..., r\}$, with $r \geq n$, each with maximum capacity Q is available to deliver jobs to customers. In our model, all jobs have the same size and therefore Q is the maximum number of jobs that can be carried by a vehicle.

Finding a solution to VRP consists of solving the following two subproblems:

(i) Assign each job $j \in \mathcal{J}$ to a vehicle $v \in \mathcal{V}$. Let $\mathcal{J}^V(v)$ be the set of jobs assigned to vehicle v and $\mathcal{C}^V(v) = \{C_j, j \in \mathcal{J}^V(v)\}$ be the set of customers served by vehicle v.

(ii) Sequence the customers in $\mathcal{C}^V(v) \cup \{0\}$,$v = 1, \ldots, r$. Let us denote with $\rho_v = (\rho_v(1), \ldots, \rho_v(|\mathcal{C}^V(v)|))$ the sequence of customers assigned to vehicle v, and with ρ the r-tuple $(\rho_1, \rho_2, \ldots, \rho_r)$, i.e., the sequencing for all vehicles. The route of vehicle v always starts from the depot 0, visits the customers in ρ_v in the given sequencing and returns to the depot after visiting the last customer in ρ_v. Let $\mathcal{J}^V(v, h) = \{j \in \mathcal{J}^V(v) | C_j = h\}$ be the set of jobs ordered by customer h that are delivered by vehicle v.

A solution ρ is feasible if: (i) each job is assigned to a vehicle, (ii) each vehicle delivers at most Q jobs, and (iii) each vehicle does not depart from the depot before the maximum release date of the jobs to deliver, i.e., there is a release time $E_v = \max\{e_j : j \in \mathcal{J}^V(v)\}$ for vehicle v.

Given a solution ρ it is possible to calculate:

- the arrival time A_{vh} of vehicle v at customer h. Letting i_h be the position of customer h in sequence ρ_v, i.e., $h = \rho_v(i_h)$, we have: $A_{vh} = A_{v\rho_v(i_h-1)} + t_{\rho_v(i_h-1)h}$, with $A_{v\rho_v(1)} = E_v + t_{0\rho_v(1)}$;
- the arrival time A_j of job j: $A_j = A_{\rho_v(C_j)C_j}$;
- the total delivery time $t(\rho_v)$ of route ρ_v of vehicle v as the sum of all travel times on the route.

A solution ρ of the distribution problem can be represented with a graph (see Figure 4.1) similar to the graph used to represent a solution H of the scheduling problem. It can be defined by $\mathcal{G}(\rho) = (V, E(\rho) \cup F)$, where V is a set of nodes, one for each job $j \in \mathcal{J}$, for each customer $h \in \mathcal{C}^V(v)$, and for each vehicle $v \in \mathcal{V}$, plus three auxiliary nodes *start_d*, *viol_dead*, *viol_due*. Hereinafter, unless specified otherwise, we will identify a node with the corresponding vehicle, customer or job and vice versa. $E(\rho)$ is a set of arcs depending on the chosen routing ρ. There is an arc in $E(\rho)$ connecting each node $v \in V$ to the first node in the route ρ_v, plus an arc between each pair of consecutive nodes h and k in route ρ_v for all v, weighted with t_{hk}, and an arc from each node $h \in \mathcal{C}^V(v)$ to each node $j \in \mathcal{J}^V(v,h)$. In the set of fixed arcs F, there is an arc from *start_d* to each node $v \in \mathcal{V}$, with weight E_v, and from each node $j \in \mathcal{J}$ to *viol_dead* if a deadline is defined for j, with weight $-\tilde{d}_j$, or from j to *viol_due* if a due date is defined for j, with weight $-d_j$.

In $\mathcal{G}(\rho)$ we define the *head* $h(j)$ of a node j as the length of longest path from *start_d* to j and its *tail* $q_a(j)$ as the length of critical path from j to an auxiliary node $a \in \{viol\_due, viol\_dead\}$.

A Tabu search algorithm for the vehicle routing problem

Neighborhood definition

Our tabu search procedure for problem VRP implements the main features of the TABUROUTE algorithm introduced by Gendrau, Hertz and Laporte [14], adapted to our problem.

Two basic moves are used for generating the neighborhood. With the first move (*reroutingC*), all the jobs to be delivered by a vehicle to the same customer are moved to the route of another vehicle. With the second move (*reroutingJ*), a single job is moved from a vehicle to another.

The *reroutingC* move $\psi^C(h,w)$ is defined as follows: given a solution ρ, a customers h in ρ_v and a vehicle $w \neq v$, a new solution ρ' is obtained from

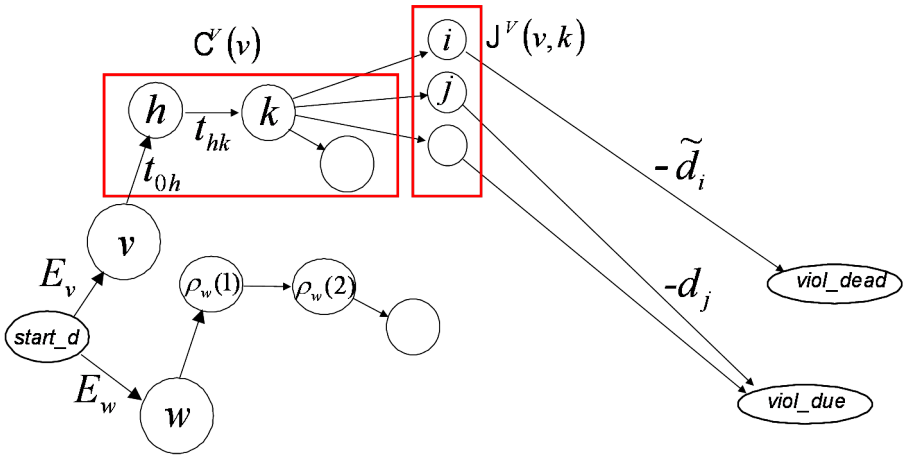

Figure 4.1: Graph model for the distribution problem

ρ by moving customer h and all the associated jobs to route ρ_w, positioning customer h just before the closest customer $k = argmin_{l \in \mathcal{C}^V(w)}\{t_{hl}\}$ in ρ_w. If $h = k$ (i.e., $t_{hk} = 0$), only the jobs are moved to ρ_w and associated to k, while customer h is deleted from ρ_v and not added to ρ_w.

The *reroutingJ* move $\psi^J(j, w)$ consists in moving a single job j from its current vehicle, say v, to vehicle $w \neq v$. If ρ_w already contains a customer $k = C_j$, then j is added to $\mathcal{J}^V(v, k)$, otherwise customer C_j is added to ρ_w and inserted in the route immediately before its closest customer k in ρ_w, i.e., the one with smallest $t_{C_j k}$. If $|\mathcal{J}^V(v, C_j)| = 1$, move $\psi^J(j, w)$ coincides with $\psi^C(C_j, w)$ and customer C_j is removed from ρ_v.

Other two moves are defined to be used as perturbation strategy. The *unify* move joins two routes ρ_v and ρ_w in a single route, thus reducing the number of vehicle used by the solution. The *divide* move generates two routes from a route ρ_v and adds a vehicle to the solution for the new route. The former move is applied after a given number of iterations during which the algorithm did not find feasible solutions, i.e., all the solutions explored violate the vehicle capacity constraints. The latter perturbation move is applied after a given number of iterations during which the algorithm was not able to improve the current optimum.

Neighborhood exploration

Two strategies are used for neighborhood exploration. The first is based on the critical path criteria, while the second randomly explores a subset of neighbors.

With the first strategy, the exploration is restricted to moves involving nodes on a critical path, i.e., a longest path from *start_d* to node *viol_due* or *viol_dead*. Specifically, we consider a critical path from *start_d* to *viol_dead* if $h(viol\_dead) > 0$, otherwise a critical path from *start_d* to *viol_due* if $h(viol\_due) > 0$. Let us denote with $\mathcal{P}_h$ the set of the p customers closest to h in terms of time t_{hk}. Move $\psi^C(h, w)$ generates a neighbor for each customer h served by vehicle v in the current critical path and for each vehicle $w \neq v$ such that:

- w already serves customer h;
- w serves one customer k belonging to $\mathcal{P}_h$ but not customer h (in this case h is inserted before customer k);
- w is empty.

Similarly, other neighbors are generated by moving only the last job j on the current critical path. Let v be the vehicle delivering job j and $h = C_j$, then the neighbor is generated by move $\psi^J(j, w)$, with w chosen according to the three previous conditions.

The random exploration is derived from the TABUROUTE algorithm but with the following modifications to take into account the different objective function of the problem. Let us partition the customer-set of $\mathcal{G}(\rho)$ into three sub-sets W^1,W^2 and W^3 defined as follows. For each vehicle v, let us consider its route ρ_v and the associated delivery time $h(k)$ for customer $k \in \mathcal{C}^V(v)$. Let $\overline{h}_1$ be the position of the last customer on ρ_v for which the delivery time violates the customer deadline, the value $\overline{h}_1 = 0$ corresponding to the case in which no deadline is violated:

$$\begin{aligned} \overline{h}_1 = \max\{i : \quad & (i = 0) \\ \vee \quad & (\quad (1 \leq i \leq |\mathcal{C}^V(v)|) \\ & \wedge \quad (\max_{j \in \mathcal{J}^V(v,\rho(i))}\{h(j) + q_{viol\_dead}(j)\} > 0) \quad) \quad \} \end{aligned}$$

Let $\overline{h}_2$ be the position of the last customer on ρ_v after $\overline{h}_1$ for which the delivery time violates the customer due date, with $\overline{h}_2 = \overline{h}_1$ if no customer after $\overline{h}_1$ violates the due date:

$$\begin{aligned}\overline{h}_2 = \max\{i : (i = \overline{h}_1) \\ \vee \quad (\quad (\overline{h}_1 + 1 \leq i \leq |\mathcal{C}^V(v)|) \\ \wedge \quad (\max_{j \in \mathcal{J}^V(v,\rho(i))}\{h(j) + q_{viol\_due}(j)\} > 0) \quad) \quad \}\end{aligned}$$

The set $W^1 = \{\rho_v(1), \ldots, \rho_v(\overline{h}_1)\}$ contains all the customers from the beginning of ρ_v until $\overline{h}_1$, the set $W^2 = \{\rho_v(\overline{h}_1 + 1), \ldots, \rho_v(\overline{h}_2)\}$ contains all the customers after $\overline{h}_1$ until $\overline{h}_2$, and $W^3 = \{\rho_v(\overline{h}_2 + 1), \ldots, \rho_v(|\mathcal{C}^V(v)|)\}$ contains the remaining customers.

With the random exploration strategy, a set $\mathcal{Q}$ of q customers is generated by randomly choosing ξ^1 customers in W^1, ξ^2 customers in W^2 and ξ^3 customers in W^3, where $\xi^1 > \xi^2 > \xi^3$ are three parameters of the algorithm such that $q = \xi^1 + \xi^2 + \xi^3$. Then, for each customer $h \in \mathcal{Q}$, neighbors are generated by move $\psi^C(h, w)$, with w chosen as in conditions above.

Move evaluation

Let $\tilde{T}_{max} = \max\{0, h(viol\_dead)\}$ denote the maximum deadline violation, $T_{max} = \max\{0, h(viol\_due)\}$ the maximum tardiness, n_V the number of vehicles serving at least one customer and t_{tot} the total delivery time of all routes of a solution ρ. The objective function of VRP is a weighed sum of $\tilde{T}_{max}$, T_{max}, n_V and t_{tot}:

$$f_{VR}(\rho) = \omega\tilde{T}_{max} + \omega_1 T_{max} + \omega_3 n_V + \omega_4 t_{tot} \tag{4.1}$$

In our tabu search, the exploration of infeasible solutions is allowed in order to search for more effective solutions, however infeasible solutions are penalized. Letting $s_{tot} = \sum_{v \in \mathcal{V}} \max\{0, |\mathcal{J}^V(v)| - Q\}$ be the total surplus of all vehicles, the function used to evaluate the neighborhood of the incumbent solution ρ is the following:

$$f_2(\rho) = f_{VR}(\rho) + \omega_5 s_{tot} \tag{4.2}$$

In the following we denote with ρ' a generic solution in the neighborhood of ρ and with similar notation we denote the variables related to ρ'. In order to evaluate ρ', values n'_V, s'_{tot} and t'_{tot} are calculated exactly for each neighbor. In particular value n'_V decreases when the only customer in a vehicle is moved, and it increases when a customer is moved to an empty vehicle. The new surplus s'_{tot} when a customer h is moved from vehicle v to vehicle w is calculated as $s'_{tot} = s_{tot} + \max\{0, |\mathcal{J}^V(v)| - Q - |\mathcal{J}^V(v,h)|\} - \max\{0, |\mathcal{J}^V(v)| - Q\} + \max\{0, |\mathcal{J}^V(w)| - Q + |\mathcal{J}^V(v,h)|\} - \max\{0, |\mathcal{J}^V(w)| - Q\}$,

in which $\max\{0, |\mathcal{J}^V(v)| - Q - |\mathcal{J}^V(v,h)|\} - \max\{0, |\mathcal{J}^V(v)| - Q\}$ is the variation of surplus in the source vehicle v and $\max\{0, |\mathcal{J}^V(w)| - Q + |\mathcal{J}^V(v,h)|\} - \max\{0, |\mathcal{J}^V(w)| - Q\}$ is the variation in the destination vehicle w. Similarly, the new total delivery time t'_{tot}, when a customer h is moved from vehicle v and inserted before customer k in vehicle w, is calculated as $t'_{tot} = t_{tot} + t_{\rho_v(i_h-1)\rho_v(i_h+1)} - t_{\rho_v(i_h-1)h} - t_{h\rho_v(i_h+1)} + t_{hk} + t_{\rho_w(i_k-1)h} - t_{\rho_w(i_k-1)k}$, where $t_{\rho_v(i_h-1)\rho_v(i_h+1)} - t_{\rho_v(i_h-1)h} - t_{h\rho_v(i_h+1)}$ is the delivery time variation for the source route and $t_{hk} + t_{\rho_w(i_k-1)h} - t_{\rho_w(i_k-1)k}$ that of the destination route.

In order to estimate the heads of nodes $h'(viol\_due)$ and $h'(viol\_dead)$ we use the following criteria. Let us first observe that moving jobs from v_1 to v_2 may cause a variation of the release time of the two vehicles. We denote with $\Delta_{E_1} = E'_{v_1} - E_{v_1}$ and $\Delta_{E_2} = E'_{v_2} - E_{v_2}$ the variation of release time of vehicles v_1 and v_2, respectively, when passing from $\mathcal{G}(\rho)$ to $\mathcal{G}(\rho')$.

For a *reroutingC* move $\psi^C(h, v_2)$, moving customer h from vehicle v_1 to vehicle v_2 before customer $k \neq h$, we have $E'_{v_2} = \max\{e_j : j \in \mathcal{J}^V(v_1) - \mathcal{J}^V(v_1, h)\}$ and $E'_{v_1} = \max\{e_j : j \in \mathcal{J}^V(v_2) \cup \mathcal{J}^V(v_1, h)\}$. Then, we estimate the new head $h'(viol\_due)$ as the maximum of two quantities $h'_1(viol\_due)$ and $h'_2(viol\_due)$ that take into account the contributions to $h'(viol\_due)$ of vehicles v_1 and v_2. Let us denote with $T_k = \max\{0, h(k) + \max_{j \in \mathcal{J}^V(v,k)}\{q_{viol\_due}(j)\}\}$ the maximum tardiness of the jobs belonging to customer k and delivered by vehicle v. Then, the quantity $h'_1(viol\_due)$ is estimated as the maximum between the tardiness associated to the customer preceding h in ρ_{v_1} and the tardiness of all customers following h in ρ_{v_1}. Specifically, the new head of the node preceding h is $h'(\rho_{v_1}(i_h - 1)) = h(\rho_{v_1}(i_h - 1)) + \Delta_{E_1}$ and its tardiness is $T'_{\rho_{v_1}(i_h-1)} = \max\{0, h'(\rho_{v_1}(i_h - 1)) + \max_{j \in \mathcal{J}^V(v_1, \rho_{v_1}(i_h-1))}\{q_{viol\_due}(j)\}\}$. The new head of the node following h is $h'(\rho_{v_1}(i_h + 1)) = h'(\rho_{v_1}(i_h - 1)) + t_{\rho_{v_1}(i_h-1)\rho_{v_1}(i_h+1)}$ while the tail of $\rho_{v_1}(i_h + 1)$ is unchanged and takes into account also all the customers in ρ_{v_1} visited after $\rho_{v_1}(i_h + 1)$. The estimate of $h'_1(viol\_due)$ is therefore:

$$h'_1(viol\_due) = \max\{T'_{\rho_{v_1}(i_h-1)}, h'(\rho_{v_1}(i_h + 1)) + q_{viol\_due}(\rho_{v_1}(i_h + 1))\}. \quad (4.3)$$

Similarly to $h'_1(viol\_due)$, the quantity $h'_2(viol\_due)$ is estimated as the maximum tardiness of the jobs belonging to one of the following customers: (i) the customer preceding k in ρ_{v_2}, (ii) customer h, (iii) customer k and (iv) all the customers following k in ρ_{v_2}. The new head of customer $\rho_{v_2}(i_k - 1)$ is $h'(\rho_{v_2}(i_k - 1)) = h(\rho_{v_2}(i_k - 1)) + \Delta_{E_2}$ and its tardiness becomes $T'_{\rho_{v_2}(i_k-1)} = \max\{0, h'(\rho_{v_2}(i_k-1)) + \max_{j \in \mathcal{J}^V(v_2, \rho_{v_2}(i_k-1))}\{q_{viol\_due}(j)\}\}$. The tardiness of h becomes $T'_h = \max\{0, h'(\rho_{v_2}(i_k-1)) + t_{\rho_{v_2}(i_k-1)h} + \max_{j \in \mathcal{J}^V(v_1,h)}\{q_{viol\_due}(j)\}\}$.

The tardiness of k and all its successors in ρ_{v_2} is $\max\{0, h'(\rho_{v_2}(i_k - 1)) + t_{\rho_{v_2}(i_k-1)h} + t_{hk} + q_{viol\_due}(k)\}$.

$$h'_2(viol\_due) = \max\{T'_{\rho_{v_2}(i_k-1)}, T'_h, h(\rho_{v_2}(i_k-1))+t_{\rho_{v_2}(i_k-1)h}+t_{hk}+q_{viol\_due}(k)\}. \quad (4.4)$$

Similar discussions hold for move $\psi^J(j, w)$ and for $h'(viol\_dead)$. At each iteration of the tabu search, the neighbor with the smallest value of $f_2(\rho')$ is selected as the new incumbent, and the starting and arrival times are updated accordingly.

4.4 Combined problem

The combined problem consists of finding a global solution $W = (H, \rho)$ where H is a solution to the scheduling problem SP and ρ is a solution to the vehicle routing problem VRP. The objective function $F(W)$ is the minimization of the overall production and distribution costs. As components of $F(W)$ we consider a linear combination of the makespan $C_{max}(H)$ of schedule H, which is a common surrogate of machine utilization and internal production costs, and the cost function of the vehicle routing problem, i.e., $F(W) = \omega_2 C_{max} + f_{VR}(\rho)$. The parameter ω_2 reflect the relative importance of the makespan component with respect to $f_{VR}(\rho)$.

In this section we present two approaches studied for the resolution of this problem. The first, decentralized, approach follows the commonly adopted decomposition approach in which SP is solved independently from VRP. Due dates at the customer are converted into due dates for each job in the production scheduling problem by letting a fixed value Δ_d for delivery, which is subtracted to the due date for delivery. The completion time of each job in the scheduling problem becomes a release time for the same job in the vehicle routing problem.

The second, centralized, approach solves the two subproblems simultaneously as a single production scheduling and delivery problem $SP + VRP$.

Decentralized approach

In the decentralized approach the problem is decomposed into two subproblems:

- the scheduling problem SP,
- the vehicle routing problem VRP.

SP is formulated with the model described in Section 3.3 with objective function $f_{SP}(H) = \omega\tilde{T}_{max} + \omega_1 T_{max} + \omega_2 C_{max}$, and solved with the version B-D of algorithm TS described in 3.4. In order to define a due date for each job $j \in \mathcal{J}$, the due dates for delivering jobs to final customers are converted into due dates for production scheduling by letting a fixed time Δ_d for delivery, i.e., $d_j^S = d_j^V - \Delta_d$ where d_j^S and d_j^V are the due dates for SP and for VRP, respectively. In the second phase the vehicle routing problem VRP is solved with the algorithm introduced in Section 4.3. The solutions of the two sub-problems are then joined to form the solution of the combined problem.

Given the best solution $\overline{H}$ to problem SP found by TS and the corresponding vector of completion times $\overline{C}$, problem VRP is formulated by defining a release time $e_j = \overline{C}_j$ for each job $j \in \mathcal{J}$. An initial solution for VRP is found by applying a simple constructive algorithm that assigns the first available job in each machine in $\overline{H}$ to an empty vehicle until the maximum capacity Q is reached; this procedure is repeated by considering other vehicles until all jobs in $\overline{H}$ are assigned to a vehicle. Then the tabu search algorithm of Section 4.3 is applied to minimize the objective function of VRP, $f_{VR}(\rho) = \omega\tilde{T}_{max} + \omega_1 T_{max} + \omega_3 n_V + \omega_4 t_{tot}$.

Letting $\overline{\rho}$ be the best solution found to VRP, a solution $\overline{W} = (\overline{H}, \overline{\rho})$ to the combined problem is obtained by coupling the solutions of the two sub-problems. The cost of $\overline{W}$ is then:

$$F(\overline{W}) = \omega_2 C_{max}(\overline{H}) + f_{VR}(\overline{\rho}) \tag{4.5}$$

Centralized approach

With the centralized approach, the scheduling problem and the vehicle routing problem are solved simultaneously as a single combined problem. To this aim, an integrated model has been defined and a global algorithm developed.

Model

The model adopted to represent a solution is an extension of the graph representations used for the single sub-problems. In particular the two graph representations are joined to form a global representation of the whole solution (see Figure 4.2). This is done by the following simple modifications:

- node *start_d* and corresponding arcs are removed from $\mathcal{G}(\rho)$
- an arc (j, v) is added from the node associated to job $j \in \mathcal{G}(H)$ to the node associated to the vehicle v for each $j \in \mathcal{J}^V(v)$.

The resulting graph is denoted with $\mathcal{G}(W)$.

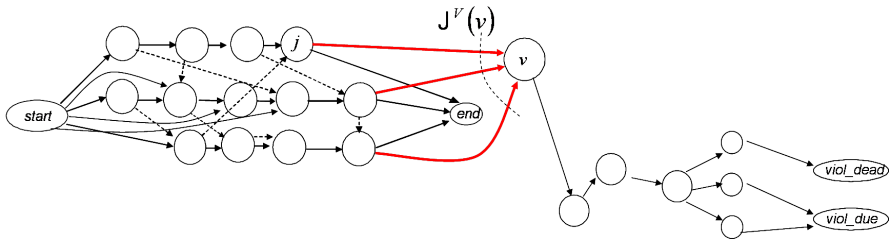

Figure 4.2: Graph model for the centralized resolution approach for the combined problem

Neighborhood definition and exploration

In this section we briefly describe the tabu search algorithm we use to solve the combined problem. The neighborhood used is the union of neighborhoods defined for the SP and VRP. We next describe two strategies used for neighborhood exploration, similar to those adopted for solving the two subproblems. One is based on the critical path criteria, and the other randomly explores a subset of neighbors.

The first strategy take into account critical paths in $\mathcal{G}(W)$ from *start* to node *end*, *viol_due* or *viol_dead*. Specifically, we consider a critical path from *start* to *viol_dead* if $h(viol\_dead) > 0$, otherwise a critical path from *start* to *viol_due* if $h(viol\_due) > 0$, otherwise a critical path from *start* to *end*. Paths from *start* to *viol_dead* or *viol_due* can be divided into two paths: a path from *start* to a node representing a vehicle, and a path from a node representing a vehicle to *viol_dead* or *viol_due*. The first path refers to the production scheduling component of the solution. Neighbors are therefore generated by considering *interchange* and *rerouting* moves involving nodes in this path (see section 3.4). The second path refers to the routing component of W. Other neighbors are therefore generated by considering *reroutingC* and *reroutingJ* moves involving nodes in this path (see section 4.3). Finally, nodes in the critical path from *start* to *end* are used to generate other neighbors with the *interchange* and *rerouting* moves.

The random exploration is the same described in Section 4.3.

Move evaluation

The objective function of the combined problem is defined as:

$$F(W) = \omega\tilde{T}_{max} + \omega_1 T_{max} + \omega_2 C_{max} + \omega_3 n_V + \omega_4 t_{tot} \tag{4.6}$$

with $\tilde{T}_{max} = \max\{0, h(viol\_dead)\}$ and $T_{max} = \max\{0, h(viol\_due)\}$. Similarly to the tabu search algorithm for the VRP problem, the exploration of infeasible solutions is allowed in order to search for more effective solutions, and infeasible solutions are penalized. The function used to evaluate the neighborhood of the incumbent solution ρ, is the following:

$$F_2(W) = F(W) + \omega_5 s_{tot} \tag{4.7}$$

Depending on the chosen move, only some quantities in Equation (4.7) can vary after move implementation. In particular, C_{max} can only vary after an *interchange* move or a *rerouting* move, n_V,s_{tot} and t_{tot} can only vary after a *reroutingC* or a *reroutingJ* move, while $\tilde{T}_{max}$ and T_{max} can vary after any move. When a *reroutingC* or a *reroutingJ* move is applied, quantities n_V,s_{tot},t_{tot} are calculated exactly, while for $\tilde{T}_{max}$ and T_{max} an estimate is calculated as in Section 4.3.

Let us now consider the *interchange* and *rerouting* moves. C_{max} is calculated exactly as in Section 3.4. For the estimate of $\tilde{T}_{max}$ and T_{max} there is one significant difference with the decentralized case. In fact, in the centralized case any change to the production schedule directly affects the solution of the vehicle routing problem. In order to take into account the global effects on $\tilde{T}_{max}$ and T_{max} of the *interchange* moves $\varphi^M(i,j)$ and $\varphi^T(i,j)$, and of the *rerouting* move $\theta^M(i,j)$, an exact computation of the new heads is performed for the scheduling part of the graph in order to evaluate exactly the release time E'_v of each vehicle $v \in \mathcal{V}$. This computation is performed at no extra cost since the new heads of these nodes are evaluated exactly for the estimation of C_{max}. Since the tails of vehicle nodes are unchanged when passing from $\mathcal{G}(W)$ to $\mathcal{G}(W')$, $\tilde{T}_{max}$ becomes $\tilde{T}_{max} = \max_{v\in\mathcal{V}}\{E'_v + q_{viol\_dead}(v)\}$, while $T_{max} = \max_{v\in\mathcal{V}}\{E'_v + q_{viol\_due}(v)\}$.

4.5 Computational experiments

This section reports on the performance of the two resolution approaches for the combined problem and compares the corresponding results. For all experiments, the parameters in the objective functions $F(W)$ and $F(\overline{W})$ are set as

follows: $\omega = 10^{10}$, $\omega_1 = 10$, $\omega_2 = 1$, $\omega_3 = 10^2$ and $\omega_4 = 1$. As for ω_5, this parameter varies dynamically during the execution of the algorithm, as in [14]. The algorithms are coded in C language and run on a Pentium® 4, 3.0 GHz with 1024 MB Ram. The testbed for the experiments contains the 24 realistic instances described in Section 3.5 for the production scheduling problem, divided in three groups of easy, medium and hard instances. The instances have been extended by adding a vehicle routing part. A customer is randomly associated to each job and the number c of customers is set $c = n/2$, so that an average of 2 jobs is delivered to each customer. Then, each customer is assigned to one out of 15 Italian cities. Table in Figure 4.3 reports the distances (in km) between the cities and the depot. For all vehicles an average speed of $60km/h$ is considered. The difference between the due dates/deadlines for the vehicle routing problem and for the scheduling problem are fixed equal to 48 hours for each job. The capacity of each vehicle is $Q = 10$.

Location	DEP.	AN	BA	BO	BZ	FI	GE	LI	MI	NA	PE	RC	RM	TO	TS	VE
DEP.	0	381	417	467	729	361	595	396	662	188	286	665	80	780	757	621
Ancona	381	0	490	209	479	255	495	349	427	403	151	970	283	544	489	341
Bari	417	490	0	697	967	743	983	797	915	279	339	530	481	1032	977	829
Bologna	467	209	697	0	280	107	286	175	218	613	358	1177	399	335	313	165
Bolzano	729	479	967	280	0	387	391	449	268	882	628	1447	679	412	328	214
Firenze	361	255	743	107	387	0	267	94	325	525	404	1089	292	433	420	272
Genova	595	495	983	286	391	267	0	214	149	749	644	1313	516	166	538	390
Livorno	396	349	797	175	449	94	214	0	311	549	491	1113	316	380	488	340
Milano	662	427	915	218	268	325	149	311	0	831	576	1395	613	144	415	283
Napoli	188	405	279	613	882	525	749	549	831	0	254	571	233	915	892	744
Pescara	286	151	339	358	628	404	644	491	576	254	0	819	233	693	638	490
Reggio C.	665	970	530	1177	1447	1081	1313	1113	1395	571	819	0	797	1479	1457	1309
Roma	80	283	481	399	679	292	516	316	613	233	233	797	0	682	712	564
Torino	780	644	1032	335	412	433	166	380	144	915	693	1479	682	0	560	425
Trieste	757	489	977	313	328	420	538	488	415	892	638	1457	712	560	0	164
Venezia	621	341	829	165	214	272	390	340	283	744	490	1309	564	425	164	0

Figure 4.3: Distances between customer locations (km)

Tables 4.1 and 4.2 show the results obtained with the decentralized and centralized approach, respectively. Each row in the tables reports the average results over eight instances. Column 2 reports the average computation time (in seconds) required to find the best solution to a problem instance.

It can be observed from tables 4.1 and 4.2 that the centralized approach provides much better results than the decentralized one. The improvement is

particularly important for the hard instances, for which there are two instances over eight for which the decentralized approach is not able to find a solution without violating some deadlines (see the very high values in columns $F(\overline{W})$ and $F(W)$ of tables 4.1 and 4.3, respectively, due to the high value of the penalization constant ω in the corresponding objective functions). This result confirms the need for good coordination among different stages of the production process when high standards of quality and reliability must be provided to the final customers. The algorithm proposed is very promising in this respect since effective solutions to practical size instances of the combined problem are provided within less than a minute of computation.

Table 4.1: Test for decentralized approach

	Decentralized						
Instance	$time$	$F(\overline{W})$	C_{max}	$\tilde{T}_{max}$	T_{max}	n_v	t_{tot}
Difficult	57.0	973751461.5	254.8	0.97	56.0	4.6	177.7
Medium	55.6	1173.9	256.4	0	40.3	3.6	151.8
Easy	63.6	637.9	234.0	0	0	3	103.9

Table 4.2: Test for centralized approach

	Centralized						
Instance	$time$	$F(W)$	C_{max}	$\tilde{T}_{max}$	T_{max}	n_v	t_{tot}
Difficult	38.9	777.7	242.5	0	0	3.1	222.7
Medium	27.5	761.8	246.7	0	0	3.0	215.1
Easy	30.9	630.8	226.4	0	0	3.0	104.4

Tables 4.3, 4.4 and 4.5 show the results obtained with the centralized approach when using different neighborhood structures and refer to difficult, medium and easy instances, respectively. Each row in the tables reports the average results over eight instances. This second set of experiments aims to demonstrate the effectiveness of the different neighborhood structures. In columns 2 and 3 we show the number of iterations and the computation time (in seconds) of the tabu search to find the best solution.

It can be observed that the Critical Path neighborhood (CP) is particularly effective for the minimization of the minmax components in the objective function (C_{max}, $\tilde{T}_{max}$ and T_{max}) while the random neighborhood is particularly effective with the minsum components (n_v and t_{tot}). However, the joint

Table 4.3: Centralized approach with different neighborhoods

	Difficult instances							
Neighborhood	*iter*	*time*	$F(W)$	C_{max}	$\tilde{T}_{max}$	T_{max}	n_v	t_{tot}
CP+Random VR	63.4	38.9	777.7	242.5	0	0	3.1	222.7
Random VR	40.7	5.1	9528750821.0	314.9	9.5	0	3.1	196.1
CP	68.5	28.4	920.9	240.1	0	0	4.5	230.8

use of both neighborhoods is more effective than any of the two stand alone neighborhoods.

Table 4.4: Centralized approach with different neighborhoods

	Medium instances							
Neighborhood	*iter*	*time*	$F(W)$	C_{max}	$\tilde{T}_{max}$	T_{max}	n_v	t_{tot}
CP+Random VR	18.9	27.5	761.8	246.7	0	0	3.0	215.1
Random VR	23.7	4.9	855.2	349.2	0	0	3.0	205.9
CP	18.9	29.6	761.8	246.7	0	0	3.0	215.1

Table 4.5: Centralized approach with different neighborhoods

	Easy instances							
Neighborhood	*iter*	*time*	$F(W)$	C_{max}	$\tilde{T}_{max}$	T_{max}	n_v	t_{tot}
CP+Random VR	584.6	30.9	630.8	226.4	0	0	3.0	104.4
Random VR	128.0	5.2	653.1	245.2	0	0	3.0	107.9
CP	134.5	20.9	740.3	223.6	0	0	3.0	216.7

Conclusion

This thesis deals with production scheduling in pharmaceutical industry. Manufacturing process in this industry is characterized by an high complexity of processes. This is due to various reasons as the complexity of chemical and biochemical procedures at the basis of the whole production process, cross-contamination that has to be avoided in order to guarantee the quality of products and finally the need of specialized man-power for the different operations. Moreover, compared to other manufacturing industry, 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. Hence, given the complexity of manufacturing processes and the issue of on-time delivery, it can happen that production plans released in the planning stage are not sustainable by the production capacity. In this situation, it is evident that scheduling stage is a critical operation and so, that an automated scheduling system is important, both to obtain good scheduling solutions and to have a better control of the production process. A case study of production scheduling, concerning the packaging department in a real pharmaceutical industry, has been analyzed in this thesis. A complex model and a tabu search algorithm have been developed to solve the related multi-purpose machine scheduling problem. Our tabu search algorithm is able to find good solutions within short computation time, compared to the solutions found by human schedulers. These results confirm that scheduling technology is mature to solve complex real problems; to this aim, however, it is important to face the complexity of practical scheduling problems by using detailed scheduling models.

Possible future developments of this research line could involve:

- the inclusion of further details in the scheduling problem as the transport time of tools from a machine to another one;

- the development of a more sophisticated tabu search algorithm, featuring advanced instruments as a dynamic tabu list size or diversification and intensification techniques;

The second topic studied in thesis has been the coordination between scheduling and distribution stages in the pharmaceutical production. The need for coordination arises from the observation that decisions made in a stage of the supply chain can result in inconvenient decisions for the following stages: coordination might enable to take decisions that are globally more convenient for the whole system. Coordination issues are particularly important in the pharmaceutical supply chain, in which the legal and economical implications of product stock-out requires the adoption of standards of product quality and availability close to 100%. Availability of final products requires not only to achieve excellence at each stage of the planning process, from strategic planning to real time scheduling and delivery, but also in the coordination between different stages. The need for coordination is more evident in departments that operates as pull systems as the packaging and distribution departments. The object of the second case study of this thesis has been the coordination between these two departments of the real pharmaceutical industry taken into account in the first case study. The aim of the study has been to investigate on the benefits that the introduction of a centralized decision support system can bring with respect to a decentralized one. Results have demonstrated that the centralized approach provides much better results than the decentralized one with even a less computational effort. These results confirm the need for excellence in the coordination among different stages of the production process, other than within each stage, when high standards of quality and reliability must be provided to the final customers.

Future research should be dedicated to the coordination of scheduling decisions among a larger number of departments. Following this research line, the manufacturing department could be included in the centralized approach, as well as the planning department or the marketing department. In fact the two latter departments determine the amount of work and the due dates to be met by the scheduling function. More detailed models for the distribution problem could also take into account the addition of other practical constraints as work-shifts and limited availability of human operators. Also the problem of managing the inventory between scheduling and distribution might be considered in these models, from the viewpoint of both the cost of inventory and the space available for storing finished products. From the perspective of solution techniques, there is a lot of space for the development of more sophisticated

centralized techniques. Also the development of more effective decentralized techniques is a promising direction that needs further development. Following this research line, the coordination between the departments should not be limited to the simple communication of products completion times from packaging to distribution, but the two departments might negotiate completion times for the packaging department that enable the distribution department to improve the overall performance.

Bibliography

[1] B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54:99–114, 2006. [cited at p. 42]

[2] A. Agnetis, P. Detti, C. Meloni, and D. Pacciarelli. Set-up coordination between two stages of a supply chain. *Annals of Operations Research*, 107:15:32, 2001. [cited at p. 50]

[3] O. Braysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003. [cited at p. 27]

[4] O. Braysy and M. Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005. [cited at p. 27]

[5] P. Brucker. *Scheduling Algorithms, 5th edition.* Springer, 2007. [cited at p. 10]

[6] P. Brucker, A. Drexl, R. Mohring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999. [cited at p. 27]

[7] P. Brucker and S. Knust. *Complex scheduling.* Springer, 2006. [cited at p. 10]

[8] Z. Chen and G. L. Vairaktarakis. Integrated Scheduling of Production and Distribution Operations. *Management Science*, 51:614–628, 2005. [cited at p. 50]

[9] W.C. Chiang and R.A. Russell. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on Computing*, 9(4):417–430, 1997. [cited at p. 27, 34]

[10] G. C. Cole. *Pharmaceutical production facilities. Design and applications, 2nd edition*. CRC Press, 1998. [cited at p. 2, 3, 5]

[11] S. Dauzère-Pérès, W. Roux, and J.B. Lasserre. Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research*, 107(2):289–305, 1998. [cited at p. 35, 36]

[12] M. Dell'Amico and M. Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41:231–252, 1993. [cited at p. 16]

[13] P. M. França, M. Gendreau, G. Laporte, and F. M. Muller. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 43:78–89, 1996. [cited at p. 13, 34]

[14] M. Gendreau, A. Hertz, and G. Laporte. A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, 40:1276–1290, 1994. [cited at p. 52, 61]

[15] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and operations research*, 13:533–549, 1986. [cited at p. 11]

[16] F. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997. [cited at p. 11]

[17] J. Grabowski, E. Nowicki, and S. Zdrzalka. A block approach for single-machine scheduling with release dates and due dates. *European Journal of Operational Research*, 26(2):278–285, 1986. [cited at p. 16]

[18] J. Grabowski, E. Skubalska, and C. Smutnicki. On Flow Shop Scheduling with Release and Due Dates to Minimize Maximum Lateness. *The Journal of the Operational Research Society*, 34:615–620, 1983. [cited at p. 16]

[19] J. Grabowsky and M.Wodecki. A very fast tabu search algorithm for job shop problem. *Metaheuristics Optimization via Memory an Evolution, Tabu Search and Scatter Search,Kluwer*, 2004. [cited at p. 16]

[20] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic machine scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979. [cited at p. 11]

[21] N.G. Hall and C.N. Potts. Supply chain scheduling: batching and delivery. *Operations Research*, pages 566–584, 2002. [cited at p. 50]

[22] S. Hartmann, R. Kolisch, S. Hartmann, Dr. R. Kolisch, and Lehrstuhl Fur Produktion Und Logistik. Experimental evaluation of state-of-the-art heuristics for the resourc-constrained project scheduling problem. *European Journal of Operational Research*, 127:394–407, 1998. [cited at p. 24]

[23] J. Hurink, B. Jurisch, and M. Thole. Tabu search for the job shop scheduling problem with multi-purpose machines. *OR Spektrum*, 15:205–215, 1994. [cited at p. 21]

[24] A.P Hurter and M.G. Van Buer. The newspaper production/distribution problem. *Journal of Business Logistics*, 17(1):85–107, 1996. [cited at p. 50]

[25] R. Kolisch and S Hartmann. Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. *Handbook of recent advances in project scheduling, Boston: Kluwer Academic Publishers.*, 141:147–178, 1998. [cited at p. 24]

[26] R. Kolisch and S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006. [cited at p. 24]

[27] K.L. Mak and Y.S. Wong. Design of integrated production-inventory-distribution systems using genetic algorithm. *Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 414:454–460, 1995. [cited at p. 50]

[28] A. Mascis and D. Pacciarelli. Machine scheduling via alternative graphs. *Report DIA-46-2000, University of Rome, Italy*, 2000. [cited at p. 20]

[29] A. Mascis and D. Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research 143*, 103:498–517., 2002. [cited at p. 20]

[30] M. Mastrolilli and L.M. Gambardella. Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*, 3:3–20, 2000. [cited at p. 22]

[31] K.N. McKay, M. Pinedo, and S. Webster. Practice-focused research issues for scheduling systems. *Production and Operations Management*, 11:249–258, 2002. [cited at p. 26]

[32] K.N. McKay and V. C. S. Wiers. Integrated decision support for planning, scheduling, and dispatching tasks in a focused factory. *Computers In Industry*, 50:5–14, 2003. [cited at p. 26]

[33] T. E. Morton and D. W. Pentico. *Heuristic Scheduling Systems*. John Wiley & Sons, 1993. [cited at p. 10]

[34] E. Nowicki and C. Smutnicki. A fast tabu search algorith for the job shop problem. *Management Science*, 42(6), 1996. [cited at p. 17]

[35] E. Nowicki and C. Smutnicki. A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, 91:160–175, 1996. [cited at p. 16, 17, 18]

[36] E. Nowicki and C. Smutnicki. An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling*, 8:145–159, 2005. [cited at p. 13, 17, 34, 38]

[37] D. Pacciarelli. The alternative graph formulation for solving complex factory scheduling problems. *International Journal of Production Research*, 40:3641–3653, 2002. [cited at p. 9]

[38] D. Pacciarelli, C. Meloni, and M. Pranzo. Models and methods for production scheduling in pharmaceutical industry. In J. Weglarz, editor, *K. Kempf, P. Keskinocak, and R. Uzsoy (Eds.) Handbook of Production Planning,Kluwer International Series in Operation Research and Management Science*. Kluwer, 2009. [cited at p. 7]

[39] B. Piachaud. *Outsourcing of R&D in the Pharmaceutical Industry : From Conceptualization to Implementation of the Strategic Sourcing Process*. Palgrave Macmillan, 2005. [cited at p. 5]

[40] M. Pinedo. *Scheduling. Theory, algorithms, and systems*. Prentice-Hall, 1995. [cited at p. 9, 28]

[41] M. Pinedo. *Planning and Scheduling in Manufacturing and Services.* Springer, 2005. [cited at p. 9]

[42] A. Reisman, A. Kumar, and J. Motwani. Flowshop scheduling/sequencing research: a statistical review of the literature, 1952–1994. *IEEE Transactions on Engineering Management*, 44:316–329, 1997. [cited at p. 9]

[43] B. Roy and B. Sussmann. Les problems d'ordonnancements avec constraintes disjonctif. *Node DS 9 bis, SEMA, Montrouge*, 1964. [cited at p. 14]

[44] R.K. Roy. *A Primer on the Taguchi Method.* Van Nostrand Reinhold, New York, 1990. [cited at p. 42]

[45] R. Ruiz, F.S. Şerifoğlu, and T. Urlings. Modeling realistic hybrid flexible flowshop scheduling problems. *Computers and Operations Research*, 35:1151–1175, 2008. [cited at p. 9]

[46] R. Russella, W.-C. Chianga, and D.Zepedab. Integrating multi-product production and distribution in newspaper logistics. *Computers & Operations Research*, 35:1576–1588, 2008. [cited at p. 50]

[47] A.M. Sarmiento and R. Nagi. A review of integrated analysis of production-distribution systems. *IIE Transactions*, 31(11):1061–1074, 1999. [cited at p. 50]

[48] J. M. J. Schutten and R. A. M. Leussink. Parallel machine scheduling with release dates, due dates and family setup times. *International Journal of Production Economics*, 46:119–125, 1996. [cited at p. 27]

[49] N. Shah. Pharmaceutical supply chains: key issues and strategies for optimisation. *Computers and Chemical Engineering*, 28:929–941, 2004. [cited at p. 2, 5]

[50] H. Stefansson and N. Shah. Multi-scale planning and scheduling in the pharmaceutical industry. *L. Puigjaner and A. Espuna (eds.),European Symposium on Computer Aided Process Engineering, Elsevier*, 2, 2005. [cited at p. 50]

[51] V. T'kindt and J.C. Billaut. *Multicriteria Scheduling.* Springer, 2006. [cited at p. 26]

[52] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra. Job Shop Scheduling by Simulated Annealing. *OPERATIONS RESEARCH*, 40(1):113–125, 1992. [cited at p. 15]

[53] L. Venditti, D. Pacciarelli, and C. Meloni. A tabu search algorithm for scheduling pharmaceutical packaging operations. *European Journal of Operational Research*, 202. [cited at p. 27]

[54] G.H. Vieira, J.W. Herrmann, and E. Lin. Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of Scheduling*, 6:39–62, 2003. [cited at p. 26]