UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Overview on sequencing in mixed model flowshop production line with static and dynamic context

Gerrit Färber, Anna Coves

IOC-DT-P-2005-7
Febrer 2005

Institut d'Organització i Control
de Sistemes Industrials

Overview on:

Sequencing in mixed model flowshop production lines with static and dynamic context

Universitat Politècnica de Catalunya (UPC)

Institut d'Organització i Control de Sistemes Industrials (IOC)

Gerrit Färber

Dra. Anna M. Coves Moreno

Date: 14.02.2005

Contents

PREFACE **1**

1 INTRODUCTION **1**

2 SEQUENCING IN FLOWSHOPS **3**

2.1 Definition of classical flowshop 4

2.2 Nomenclature of parameters for flowshops 4

2.3 Objective Functions 8

2.3.1 Time orientated objectives 8

2.3.2 Cost orientated objectives 11

2.3.3 Combined objectives 11

2.4 Diversity of flowshops 11

2.5 Characteristics of flowshops 13

2.6 Setup cost/time in flowshops 16

2.6.1 Appearance of setup 16

2.6.2 Sequencing problems considering setup 17

3 RESEQUENCING IN FLOWSHOPS **19**

3.1 Objectives of resequencing 19

3.2 Methods for resequencing 20

3.2.1 Buffers 20

3.2.1.1 Infinite buffers 21

3.2.1.2 Large ASRS buffers 21

3.2.1.3 Small buffers 22

3.3 Hybrid or flexible flowshop 23

3.4 Merging and splitting 24

3.5 Re-entrant flowshops 24
3.6 Change of job attributes (no physical change) 24
3.7 Undesired resequencing 24
3.8 Related work on resequencing 25

4 STATIC AND DYNAMIC DEMAND 29

4.1 Static demand 29
4.2 Dynamic demand 30
4.3 Transition from pure static to dynamic demand 31
4.4 Solution techniques 32

5 OPTIMIZATION METHODS 33

5.1 Elements of optimization 34
5.2 Complexity of problems 36
5.3 Exact methods 36
5.4 Approximate methods 38
5.4.1 Heuristics 39
5.4.2 Metaheuristics 40
5.5 Evaluation of the performance of a heuristic 41
5.6 Literature providing input data 42

6 SUMMARY 42

REFERENCES 44

PREFACE

In the classical production line only products with the same options were processed at once. Products of different models, providing distinct options, were either processed on a different line or major equipment modifications were necessary. For today's production lines, this is no longer desirable and more and more rise the necessity of manufacturing a variety of different models in the same line. Mixed model production lines consider more than one model being processed in the same production line in an arbitrary sequence. Setup time/cost then becomes relevant, resulting in an additional time/cost, each time a model change occurs.
Procedures, which sequence products in optimal order, can minimize considerably the sum of setup time/cost. Arrangements, taking into account the possibility of resequencing the products within the line, are even more convincing. Buffers can be used to let jobs bypass the buffered jobs, resequencing takes effect.

1 INTRODUCTION

A production line structured as a **flowshop** requires all jobs (products) to visit the workstations in the same sequence. A conveyor belt transports the jobs past the fixed stations having designated operators. The reverse arrangement would also be possible in which the operators move from one job to the next, this however is more common in production lines with products having large dimensions and extensive weight. The current market demands greater flexibility and variety of products together with the reduction of life cycles. This leads to the use of multi model or **mixed model** production lines. In the multi model production the products form lots of the same model, whereas in the mixed model production the job sequence may be arbitrary. The model mix may be as simple as providing various options for a basic product.

Each station of the production line performs different tasks. The assignation of these tasks to the stations, subject to technological precedence relations, is called the **line balancing problem**. The process of balancing the load results in the design of the production line, usually implying the minimiza-

tion of the station number and the determination of a cycle time, obtained by calculating, e.g., an average of the task-times, necessary to assemble the various models. The balancing procedure in many cases results in the prevention of the occurrence of bottlenecks so that the final production line will not experience stoppage and unnecessary inventory will not accumulate. Studies on the production line balancing problem are numerous and may be object to various criteria like cost-oriented or profit-oriented approaches, as described in the survey of Becker and Scholl, 2003, and in Scholl, 1995. The survey of Erel and Sarin, 1998, explains different measures for balancing problem with respect to its complexity; furthermore, it gives an extensive classification with related solution procedures, such as heuristics as well as optimum seeking algorithms. Further comprehensive studies are found in Ghosh and Gagnon, 1989.

Once the line is balanced and the design of the line is obtained, it is necessary to achieve a reasonable, if not optimum, order for the jobs to be processed consecutively. Most of the existing literature mentions the optimization of line balancing and job ordering in a consecutive order and therefore focus on one of the two. Kim and Kim, 2000, present a genetic algorithm, optimizing the two at the same time. It has to be mentioned that in a productive industry it actually makes sense having the two problems separated. It would be very inefficient if a minor change in demand results in a new order and also in the re-assignation of the tasks to the stations.

Within the problem of determining the order of the jobs two confusing terms are used by various authors; **sequencing** and **scheduling**. We assume that the sequencing problem determines an appropriate order for the jobs to be processed within, e.g., the shortest possible time called makespan, used e.g. by Bard et al., 1992, Bolat, 1994 and Lahmar et al., 2003. Whereas we assume that solving the scheduling problem results in prioritizing the order of the jobs due to resource usage and due-date-limits of the jobs, see for example Sun et al., 2003, for a survey of static scheduling problems. Due to the fact that the sequencing problem also results in a schedule for the jobs on the stations, many authors use the term scheduling instead of sequencing. The work of Beaty, 1992, highlights that the two problems are either intimately tied together or irrelevant to each other and many times are used interchangeably. In order to clarify the topic, the author also discusses various combinations of the two.

The present document discusses solution techniques for the sequencing problem of mixed-model flowshop productions. Such type of production line is found in an increasing number in real production industry, resulting from the increased necessity of customer orientated product spectrums. This implies that orders are no longer accumulated to lots of the same models and then stored until a customer orders the product of this exact model, but are rather produced on short notice demand in production lines allowing the possibility of producing various models at the same time. The great majority of published research done in this field limits the solutions to permutations sequences where the order of jobs is determined before the jobs enter the production line and maintain unchanged until the end of the line.

Following the introductory chapter, the general structure and characteristics of the sequencing problem in mixed model flowshops is presented in chapter 2, also discussing problems that consider setup time and setup cost whenever a model change occurs at a station. Chapter 3 is devoted to discussing the challenges that arise in flowshop problems that consider resequencing of products rather than permutation sequences. Chapter 4 deals with the difference between static and dynamic demands. In chapter 5 a short summary of optimization methods is given and finally in chapter 6 a summary of the flowshop with resequencing considerations is given.

2 SEQUENCING IN FLOWSHOPS

The sequencing problem in flowshops appear when variations of the same basic product are produced in the same production line. These variations imply that the processing times on the individual stations differ, dependent on the model to be processed. This type of problem is called the mixed model flowshop and is defined by various parameters which reflect the complexity of the possible layouts and the different operation modes of the production line. This chapter of the document gives an overview of the nomenclature of common parameters, the diversity of flowshops presented in the literature, common objective functions and finally characteristics of sequencing problems of flowshops.

2.1 Definition of classical flowshop

The sequencing problem in the classical mixed model flowshop consists in a set of n jobs $(J_1, J_2, ..., J_j, ..., J_n)$ which have to be sequenced on m stations $(I_1, I_2, ..., I_i, ..., I_m)$, arranged in series. Each job has m sets of operations and requires its first set of operations on station 1, its second on station 2, and so on. The **set of sequences** Π $(\Pi_1, \Pi_2, ..., \Pi_i, ...\Pi_m)$ describes the order in which the jobs are processed on the m stations; in a **permutation flowshop** the sequence Π of jobs in station i is the same for all stations, i.e., $\Pi_1 = \Pi_2 = ... = \Pi_i = ... = \Pi_m$.

Furthermore, the **processing time** p_{ij} of job j on station i is known and constant. The time job j enters the system is called **start time** S_j, similarly, the time job j exits the system is called **completion time** C_j. If **setup time** is concerned, an additional time st_{fgi} may occur, necessary to change the setup of station i, in order to be able to process job $j+1$, which is of model g, after job j, which is of model f. The **setup cost** sc_{fgi} is defined respectively. An extensive survey on setup considerations is presented by Allahverdi et al., 1999. Due to additional complexity most algorithms do not consider setup-time nor -cost, assuming that setup-time/cost is sequence independent and can simply be added to the processing time/cost. Another simplification is used by algorithms for batch processing which pool jobs of the same model and process them in lots.

2.2 Nomenclature of parameters for flowshops

The nomenclature presented here is according to the nomenclature found in the literature, however, the terms used in the literature not always are coherent or unique. In order to prevent misunderstanding and improper use, the nomenclature used here is given.

Pinedo, 1995, arranges these parameters into a triplet $\alpha|\beta|\gamma$ that helps classifying sequencing and scheduling problems. The triplet determines the specific problem with α describing the station environment, β providing details of the processing characteristics and constraints and γ containing the objective to be minimized. The first and the second field usually have one entry and the third various or none. The most relevant parameters for flowshop sequencing are presented at next:

Task t: Non-divisible activity which is performed in either station. The balancing problem solves the problem of assigning tasks to stations and therefore often is called the assignment problem. In the Sequencing problem this task assignation is already performed and only stations are considered.

Job j: A part, subassembly or assembly, processed by a station is called job. In a mixed model production line the jobs belong to different models which include different processing times at the stations, depending on the model type. The number of jobs to be processed is n.

Station i: One or more tasks may be assigned to station i. In the classical flowshop problem m stations are aligned in series and all jobs j have to pass the stations in the same order. The length of station i is L_i. A station may be open or closed, depending on whether or not the operator working in it is allowed to cross its boundary.

Operation: Processing of a job in a station is called operation. This operation can include various performed tasks at one and the same station and is determined by the processing time p_{ij}.

Processing time p_{ij}: Also called assembly-time, is the time that job j maintains at station i while being processed. Due to the nature of a flowshop, job j that is not processed at station i has to pass this station with a processing time equal to zero.

Preemptive/Nonpreemptive: Preemptive operation means that processing times may be interrupted and resumed at a later time, even on another station. Furthermore an operation may be interrupted several times. If preemption is not allowed, the operation is called nonpreemptive.

Setup time st_{fgi}: Setup time is concerned if an additional time appears to change the setup of station i, in order to be able to process job $j+1$ which is of model g after job j which is of model f. If the setup time is independent of the model, it can be simply added to the processing time.

Setup cost sc_{fgi}: In a similar way, setup cost is concerned if an additional cost appears to change the setup of station i, in order to be able to process job $j+1$ which is of model g after job j which is of model f. If the setup cost is independent of the model, it can be simply added to the processing cost.

Start-time S_j: The time job j enters the system is called start-time.

Completion-time C_j: The time job j exits the system is called completion-time and is the completion time on the last station on which it requires processing.

Demand D: The demand describes the total volume of n jobs to be processed. In the scheduling problem the individual jobs can furthermore be specified by start-date and due-date. These values describe the earliest possible point of time to start working on a particular job and when the finished products has to be delivered to the customer. In order to fulfill the due-dates a penalty may be applied for delivering too early or too late. In the sequencing problem it is frequent to release customer orders to production once a certain number of jobs has been accumulated, and then sequence and produce these orders together as a lot, see for example Burns and Daganzo, 1987. The demand in this case is static and only depends on the volume of jobs and the objective usually is to minimize the processing time to complete the entire order, called makespan.

Model M: In the mixed model flowshop several variations, called models, of the same basic product are manufactured. The difference from one model to another may be due to an option that is not applied to all models or likewise in a variation of an option. Therefore the mixed-model sequencing problem consists of the determination of the consecutive order of the models. M_i determines the model of job i. **Minimal-Part-Set** (MPS): The MPS is denoted by the vector $d(d_1, d_2, ..., d_k)$ which represents a product mix, such that $d_M = D_M/h$. D_M being the number of units of model type M which needs to be assembled during an entire planning horizon and h being the greatest common divisor of $D_1, D_2, ..., D_M$. Obviously, h times repetition of the MPS sequence meets the total demand. With the MPS the number of possible sequences is reduced to $D!/(d_1! \cdot d_2! \cdot ... \cdot d_k!)$, Korkmazel and Meral, 2001. The MPS is considered to be a good choice, however, Klundert and Grigoriev, 2001, show that many times reducing the sequence to the one of the MPS does not result in the optimal sequence.

Launch-interval λ: The time between two consecutive jobs entering the production line is called launch-interval. Usually it is a constant value, also called cycle time. A constant launch-interval results in a fixed production rate (production quantity per unit of time).

Job sequence Π_i: The job sequence defines the order of jobs at station i. A job sequence that is the same for all stations is called a **permutation** sequence. In flowshops the station sequence, the order in which the individual jobs visit the stations, is the same for all jobs.

Machine breakdown/maintenance: Machine breakdown/maintenance describes the state of a station which does not permit processing of any job due to failure or failure prevention. In real production systems the breakdowns occur in a stochastic way and can be simulated using the values Mean-Time-Between-Failure (MTBF) and Mean-Time-To-Repair (MTTR).

Precedence: The precedence gives a dependency of jobs in respect to the processing. A job j is said to be predecessor of job k if job j has to be processed before job k. An immediate predecessor then is a job that has to be processed immediately before another job.

Rework operations: The detection of defective jobs may cause either rework operations or removal of the job from the line. The occurrence of defective jobs in the production is of probabilistic nature.

Paced/unpaced production line: In a paced production line the mechanical material handling equipment like conveyor belts couple the stations in an inflexible manner. The jobs are either steadily moved from station to station at constant speed or they are intermittently transferred after processing. The available amount of time for the operation is the same in both cases. In the unpaced line, in contrast, the stations are decoupled by buffers. In a specific case this buffer stores jobs that can not be passed to the downstream station which is still occupied with processing the previous job.

Deterministic-stochastic models: The deterministic model is characterized by the fact that the elements, e.g. processing-time, do not involve variation and that the consequences of any given decision can be predicted in a precise manner. The stochastic model is characterized by its explicit recognition of variation and uncertainty, which could exist in one or more of the elements with known probabilistic behavior. This may result, for example, in a variation of the operators performance.

Buffer: Buffers were originally introduced between two consecutive stations to decouple them in order to avoid blocking and starving. Buffers are often located before and after bottleneck stations. The reason is that this

already critical part of the production usually is the limiting section. In automobile productions buffers of enormous dimensions can be found, which in principle decouple the main successive production sections. This buffer is, furthermore, used to reorder the jobs, available in the buffer, on a large scale.

Static/dynamic demand: A static demand refers to the fact that the entire demand necessary to produce in a time window is produced in an accumulated lot, known beforehand. Whereas a dynamic demand implies that the customer orders arrive continuously or at least are not completely determinable beforehand.

2.3 Objective Functions

Within the sequencing problem of mixed-model production lines a variety of objective functions are to be found, the most common being time and cost orientated objectives. As a basic principle of optimization, the considered solutions are part of a set of feasible solutions and with the use of additional objectives lead to the optimal solution. For example, the method proposed by Dar-El and Cucuy, 1977, minimizes the overall line length and first determines the feasible solutions that result in a non-idle-time schedule.

2.3.1 Time orientated objectives

Makespan $C_{\max}$: One of the most common objective functions in sequencing is to minimize the maximum completion time necessary to process the entire demand, called makespan or total production time. The makespan optimization generally ensures high utilization of the production resources, early satisfaction of the customer demand and the reduction of in-process inventory by minimizing the total production run.

Makespan:

$$\max\{C_j | j = 1, ..., n\}$$

Maximum flow time $F_{\max}$: The minimization of the flow time leads to stable and even utilization of resources, rapid turn-around of jobs and the minimization of in-process inventory. French, 1982, mentions that in the

case where all release dates are zero, $C_{\max}$ and F_{max} are identical. The weighted flowtime includes a weight related to the station.

Maximum flow time:

$$\max\{C_j - S_j | j = 1, ..., n\}$$

Weighted flow time:

$$\sum_{J=1}^{n} \omega_j (C_j - S_j)$$

Mean flow time $\overline{F}$: Allahverdi, 2003, highlights and proves with a simple example that that the maximum flow time and the mean flow time are of different kind.

Mean flow time:

$$\sum_{j=1}^{n} (C_j - S_j)/n$$

Weighted mean flow time:

$$\sum_{j=1}^{n} \omega_j (C_j - S_j)/n$$

Setup time: In a mixed model production, setup time st_{fgi} may occur when at station i a job $j+1$ of model type g follows job j of model type f. Minimizing total setup time, furthermore, tends to decrease the total flowtime.

Setup time:

$$\sum_{j=1}^{n} st_{fgi}$$

Idle time: Idle time I_{ij} at station i occurs when an operator is kept waiting for job j. This may be caused by a job that has not yet arrived, or because an auxiliary operator is still occupied with the job. When setup time occurs, that is separable from the processing time, the operator can benefit from this idle time in order to perform the necessary changes for the next job to

be processed. French, 1982, highlights that the mean and the maximum for idle time are taken over the stations rather than over the jobs.

Idle time:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}$$

Mean idle time:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}/m$$

Utility time: Utility time U_{ij} at station i occurs when an operator has to continue with job $j+1$ before finishing with job j. In this case an auxiliary operator finishes the job; the time the auxiliary operator requires is called utility time. As well as the idle time, here the mean is taken over the stations. The minimization of idle and utility time is, for example, applied by Sarker and Pan, 2001, varying the station length and using individual weights for the calculation of idle and utility time.

Utility time:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} U_{ij}$$

Mean utility time:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} U_{ij}/m$$

Deviation: In general for all of the above mentioned time oriented objectives it is possible to use the deviation, or the squared deviation, over stations or over jobs, in order to equalize the deviation and to avoid solutions that provide extreme values for single stations or jobs, see e.g. Bukchin, 1998.

Regular/non-regular objective: Baker, 1974, explains that an objective function is called regular objective if the function increases only if at least one of the completion times of the jobs increases. Surveys on non-regular objective functions are presented by Baker and Scudder, 1990, and Raghavachari, 1988.

2.3.2 Cost orientated objectives

Line length: Kim et al., 1996, study the problem of minimizing the overall length of the production line. The production line in study contains hybrid stations, being a mixture of open and closed stations.

Line length:

$$\sum_{i=1}^{m} L_i$$

Setup cost: The occurrence of setup cost may lead to the objective of minimizing the total setup cost to keep the production costs small. Setup cost sc_{fgi} may occur when at station i a job $j+1$ of model type g follows job j of model type f.

Setup cost:

$$\sum_{i=1}^{m} sc_{fgi}$$

2.3.3 Combined objectives

In the literature, the use of individual objective functions, as mentioned above, as well as combinations can be found. As an example for combined objectives in sequencing problems Allahverdi, 2003, uses the bicriteria of makespan and mean flowtime, whereas Bard et al., 1992, uses makespan and line length as bicriteria for their algorithm.

2.4 Diversity of flowshops

Once the flowshop production line is designed (balanced), the problem of ordering the jobs surfaces. If release-dates and due-dates exist, the problem is called the scheduling problem, otherwise it is referred to as the sequencing problem. Next to the classical flowshop exist a variety of the same. This results from the manifold problems, that can be found in real life production systems and their specific products. In chemical processing, for example, it is common practice that once a job is started it can not be interrupted and

therefore leads to a no-wait flowshop. The more common variations are highlighted as follows:

Non-permutation flowshop (classical): One of the pioneers who mention the flowshop problem is Johnson, 1954. In the classical flowshop m stations are arranged in series, according to the technological sequence of the operations. A set of n jobs has to be processed on these stations. Each of the n jobs has the same ordering of stations for its processing sequence. Each job can be processed on one, and only one station at a time and each job is processed only once on each station. Furthermore each station can process only one job at a time. Jobs may bypass another job between two stations. The problem consists in finding a job sequence for each station i. The case with $m = 1$ is called the **single station** case.

Buffers, installed between the stations are generally used for queuing reason in order to decouple the individual stations from each other and prevent blocking of a station; depending on the type of buffer, they offer the possibility of changing the sequence downstream of the buffer.

Permutation flowshop: Here the solutions are restricted to job sequences $\Pi_1, ..., \Pi_n$ with $\Pi_1 = \Pi_2 = ... = \Pi_n$, that is, the sequence on the first station is maintained for all stations in the flowshop. A set of permutation sequences is denoted dominant if no better sequence can be found than the best permutation sequence. This for example occurs in the no-wait flowshop.

Zero-buffer and no-wait flowshop: These two variations of the classical flowshop do not allow the jobs to form queues between the stations. The first case is with buffers of zero capacity. In this case a job j finishing on station i cannot advance to station $i+1$ if there is still a job being processed. Station blocking of station i is the result. The second case, described by Aldowaisan and Allahverdi, 1998, is more restrictive. Once a job begins its processing on station 1, that job must continue without delay to be processed on each of the m stations. Here only sequences are feasible which do not result in blocking of any station.

No-idle flowshop: This constraint implies that each station, once started with processing, has to process all operations assigned to it without interruption. As mentioned by Cepek et al., 2002, a real life situation can be found, for example, if machines represent expensive pieces of equipment which have

to be rented only for the duration between the start of its first operation and the completion of its last operation.

Flexible/hybrid/compound flowshop: Another variation of flowshops mentioned in the literature is the one in which parallel stations exist, see e.g. Li, 1997, Gendreau et al., 2001, Azizoglu et al., 2001, Riane et al., 1998 or Sun et al., 2003. This type of flowshop is named, by the majority of authors, flexible, hybrid or compound flowshop. The parallel station reduces cycle times needed for an operation at a station. Since in the mixed-model case the processing time of a station is dependent on the model, it gives the possibility of one job overtaking its predecessor. For this type of flowshop basically two setups exist: identical parallel stations and non-identical parallel stations.

2.5 Characteristics of flowshops

The flowshop is a widely studied problem. Within these studies not only the basic arrangement is taken into account but furthermore simplified setups, the most common being the permutation flowshop or the reduction to single station. Also variations were studied, amongst these the hybrid or flexible flowshop, providing stations with parallel stations. In this section some properties of flowshops, derived from certain setups, are presented.

The classical flowshop permits the sequence of the jobs to change after each station which then leads to a total number of sequences being $(n!)^m$. For the simplified case of the permutation flowshop, in which the jobs have a unique sequence for all stations, the number of sequences is reduced to $n!$.

In the single station flowshop only permutation sequences are possible and the same makespan is achieved for all sequences. As shown by Baker, 1974, the mean flow time is minimized by the shortest-processing-time rule (SPT) and the weighted flow time is minimized by shortest-weighted-processing-time rule (SWPT). Furthermore, in the flowshop a common practice is to first determine the bottleneck station and consider it as being the only station.

The single station flowshop considering sequence dependent setup time corresponds to what is usually called the travelling salesperson problem (TSP). Allahverdi et al., 1999, explains that each city correspond to a job and the

distance between cities corresponds to the time required to change from one job to another.

Johnson, 1954, considers a two and three stations flowshop problem ($F_2||C_{max}$ and $F_3||C_{max}$) with makespan objective. His studies lead to the conclusion that in the two and three stations case the optimum permutation sequence is dominant. Also an exact solution for the two stations case is presented, namely SPT(1)-LPT(2). For the three stations case this method only finds the optimal solution if station two is dominant. That is, all processing times on the second station are larger then the largest on the two other stations. Gupta and Reddi, 1978, proposes improved dominance conditions that are not limited to the dominance of the second station. Moreover, exist various heuristics that take advantage of the Johnsons rule in order to solve larger problems, see for example the CDS heuristic of Campbell et al., 1970 or Riane et al., 1998. Péridy et al., 1999, points out that the dominance of the permutation sequence also holds if on the first station exist precedence relations of jobs.

Baker, 1974, explains certain properties of the classical flowshop. With respect to any regular objective function, it is sufficient to consider only schedules in which the same job sequence occurs on the first two stations. Furthermore, with respect to the makespan objective, it is sufficient to consider only schedules in which the same job sequence occurs on stations $(m-1)$ and m. Hence, for the makespan problem, it is sufficient for the same job order to occur on stations $(m-1)$ and m, so that $(n!)^{m-2}$ schedules constitute a dominant set for $m > 2$. Thus, for a three-station flowshop with makespan minimization, the optimal sequence is a permutation sequence. For the case $m > 3$, Potts et al., 1991, determined the ratio, for $m = 2 \cdot n$, between the best permutation sequence and the optimum sequence being greater than $1/2 \cdot \lceil \sqrt{m} + 1/2 \rceil$. For a four station flowshop the difference is already 50%.

Pinedo, 1995, discusses and proofs that inverting the station sequence and the job sequence $j_1, ...j_m$ of a permutation flowshop with unlimited intermediate storage to $j_m, ...j_1$ results in the same makespan. He presents a mixed integer program (MIP) for the classical flowshop of m stations with makespan objective and permutation sequences, known as $F_m|prmu|C_{max}$, and proves that the classical flowshop with makespan objective and more

than two stations is strongly NP-hard, see also Garey et al., 1976. Additionally, two special cases are discussed: station-dominance with increasing processing times for successive stations, and the proportional-flowshop where all processing times on the individual stations are the same for all jobs.

Buffers were initially introduced in permutation flowshops to decouple stations in order to prevent station blocking and starvation, see Pinedo, 1995. Blocking appears in the case in which the downstream station experiences problems, and therefore may not be able to process the assigned job. On the other hand, locating a buffer right after the station being down due to problems prevents its downstream station from starvation. The reasons for a station to be down are various, for example preventive maintenance, unavailable subassembled parts or repair of machine, etc. Moreover, bottleneck stations are generally provided with a buffer to avoid serious problems that may result in a line stoppage. The decoupling buffer is operated in first-in-first-out (FIFO) strategy. The flowshop with limited intermediate storage buffers is the more complex case in which a job may not be discarded from a station and therefore results in blocking which is the station is kept idle. In his studies of permutation sequences Pinedo, 1995, discusses a flowshop with limited buffers by reducing the storage buffers to zero. Reason for this is that a buffer can be presented by a station with zero processing time.

A directed graph is used for the calculation of the makespan of a permutation sequence. In the case of infinite buffers the nodes represent the processing time, see for example Ríos-Mercado and Bard, 1999, they also include setup time being the arcs between the nodes. When buffers are finite or not present, Pinedo, 1995, uses a different directed graph, the processing time here is presented by the arcs between the nodes. In both cases the makespan is calculated by the maximum weighted path. Lomnicki, 1965, presents a branch and bound solution for the three station case. Even though permutation sequences are dominant here, the graph-theoretical interpretation of Roy, 1962, is presented that allows the calculation of the makespan of a sequence which is not a permutation sequence.

Most solution techniques for sequencing in flowshop focuses on permutation sequences. Exact approaches for makespan minimization are presented by Ignall and Schrage, 1965, Lomnicki, 1965, Szwarc, 1971, Lageweg et al., 1978, Potts, 1980, Companys, 1992, Carlier and Rebai, 1996.

In a recent review of Framinan et al., 2002, heuristic methods for sequencing problems with focus on makespan objective are presented. Framinan and Leisten, 2003, furthermore present a comparison of heuristics for flowtime minimization in permutation flowshops. The most promising heuristics for permutation flowshops seem to be the Nawaz-Enscore-Ham (NEH) heuristic by Nawaz et al., 1983. Allahverdi, 2003, studies makespan and mean flowtime as multicriteria objective, using a linear combination. He proposes three new heuristics, AAH1, AAH2 and AAH3 and compares them with a series of existing heuristics, both for the 2- and for the m-station case.

In what follows we focus on two special types of flowshops, considering setup cost/time and afterwards non-permutation flowshops that permit to change the sequence between stations in order to allow further optimization.

2.6 Setup cost/time in flowshops

The mixed model production line produces a variety of products in the same line. These products may differ only in some optional components that are applied or not applied at a station. In this case the processing time at this station differs from one product to the next but does not require a special setup. In the case in which the operator needs to change the setup of the station in order to process the next job, the change of the setup may result in an additional production cost or an additional time, necessary to realize the change in setup. Pinedo, 1995, presents a proof of the NP-hardness of the single station case with setup consideration.

2.6.1 Appearance of setup

Burns and Daganzo, 1987, discuss a flow shop with setup costs and distinguish between three different types of setup cost/time:

- **Wasted material** resulting in an additional cost due to, for example, discard of the paint in the paint shop of an automobile production. This setup cost has only impact on the objective function that minimizes the production costs.

- **Station downtime** and **labor** required to change the setup. This occurs for example when the mounting or a tool needs to be changed. In this case the schedule of the jobs is influenced directly, this means a sequence without setup time is not possible because some job change requires additional time for preparation.
- **Product quality implications** which affect the performance of the station. For example the paint quality may temporarily decline when a change of color occurs.

Apart from the differentiation of the setup cost/time, a distinction is done in respect to the stations. Bolat, 1994, classifies the stations into two types: Some stations are assigned to do exactly the same operation on every job, and some allow operations that are not performed on every job or performed on every job but with some options. In a mixed model production line which consist of various stations usually both types of stations are found.

2.6.2 Sequencing problems considering setup

A partly efficient but widespread technique to avoid setup is to form batches, groups of jobs belonging to the same model. In this way the number of instances in which setup occurs is the number of batches that are to be processed. This simple method neither permits an advanced optimization of the sequence, neither additional constraints such as an option can only be applied on every second job. Nevertheless until now it is a widely applied method in the industry.

Another approach considers that setup time exists, but the average of the setup time is added to the processing time. The adapted processing times are then used to solve the sequencing problem; however, the determined sequence needs to be revised for feasibility. The method is sensitive to inhomogeneous distribution of the setup time.

In order to further improve the solution, the setup cost/time is considered completely separated. Allahverdi et al., 1999, highlight that when setup cost is directly proportional to setup time, a sequence that is optimal with respect to setup cost is also optimal with respect to setup time. Their survey on setup considerations furthermore considers sequencing problems

in flowshops regarding the following characteristics:

- **Batch setup**: Here jobs are grouped into batches and a major setup is incurred when switching between jobs belonging to different batches, whereas a minor setup is incurred for switching between jobs within the batches.
- **Sequence dependent setup**: Including setup cost/time that depends on the succeeding job gives the possibility to further improve the sequence. In the symmetric case the appearance of setup cost/time is the same for a change from model f to g and for a change from model g to f.
- **Setup time separable from processing time**: The case in which setup time is separable from processing time leads to the possibility of further reducing the total processing time. This results from the fact that once on a station i a job j is finished, the setup can already be changed way before job $j+1$ arrives.

Apart from the single use of these characteristics exist various authors that incorporate other line considerations. Burns and Daganzo, 1987, discuss the paced flowshop line with setup costs. In addition, constraints are used to determine the minimum distance between two jobs of the same model, if not fulfilled, the station capacity is exceeded. The proposed method uses grouping and spacing to sequence the jobs. The work does not tempt to achieve the optimum solution but establishes analytical principles that aid in developing effective production line job sequencing methods.

The work of Kim and Kim, 2000, proposes two methods for solving the sequencing problem including setup times necessary for a model-change and uses closed stations. The first method proposed is a branch and bound using a lower bound that calculates the lowest possible unfinished work of the remaining sequence. The authors remark that the method finds the optimum solution for mid-scale problems. The second method is a heuristic method called MST (Minimum-Setup-Time) and favors jobs that prevent idle and unfinished work.

With a genetic algorithm Kim et al., 1996, find near optimal solutions solving the problem of minimizing the overall length of the production line. The

production line in study contains hybrid stations which is a mixture of open and closed stations. They highlight the importance to achieve a proper balance between exploitation and exploration of the genetic algorithm. These characteristics describe the good choice of the size of populations and the technique of forming new populations. The results are then compared with a conventional, slowly proceeding branch and bound algorithm provided by standard software packages.

Bolat, 1994, presents next to a branch and bound algorithm a heuristic technique, applied to an automobile production line which takes into account setup costs, caused by discarded paint and solvent when a change in color occurs. Bolat highlight the importance of considering both setup and utility costs, mainly to evaluate the economic benefit of investments in setup cost reduction. The used heuristic, TRIM, basically selects jobs that result in less utility-work without considering setup costs.

3 RESEQUENCING IN FLOWSHOPS

As mentioned in chapter 2, in the classical flowshop with three stations and makespan objective, the permutation sequence is dominant. Pinedo, 1995, demonstrates that the problem of three or more stations with makespan objective is strongly NP-hard and that for a production line consisting of more than three stations furthermore a permutation sequence is not dominant and clearly leads to additional complexity.

3.1 Objectives of resequencing

In order to permit a job to overtake another job within a production line, arranged as a flowshop, normally the line has to undergo essential changes, many times combined with investment. These changes may result in hardware to be installed, like buffers, but also in additional efforts in terms of logistics implementations. Clearly, these additional efforts are only reasonable if the resequencing pays off the necessary investment. This cost calculation obviously depends on the particular production line and therefore has to be taken into account for the individual cases.

The major objective of resequencing a pre-arranged set of jobs in flowshops is further minimization of production costs, for example resulting in a higher utilization of the production resources. This is desirable even more when setup cost/time is involved or the processing times of the individual jobs diverge among one another.

Apart from the deliberate resequencing of jobs in order to improve the productivity, an undesired resequencing, of a permutation sequence, may occur. Here the objective is to regenerate the original sequence.

3.2 Methods for resequencing

Various possibilities of resequencing jobs in a flowshop exist, some of which may already be included in the lines setup. Others may require additional installations:

- Offline or intermittent **buffers**; the latter not operated in FIFO mode,
- Hybrid/flexible flowshops containing **parallel stations**,
- **Splitting** and **merging** of parallel lines,
- **Re-entrant** of jobs in the production line and
- Change of **job attributes** (no physical change).

References on methods for resequencing are summarized in table 1 to table 3 on page 26 to 28.

3.2.1 Buffers

The buffer types, used to temporarily store jobs within a flowshop, basically have three objectives: decoupling, resequencing and end-product storage. Here only buffers are concerned that permit resequencing which results in the use of non-permutation sequences. The usefulness of this type of buffers is obvious already in the simple case, in which a line is divided into two parts; a permutation sequence that is optimum for the first part usually is not optimum for the second part.

3.2.1.1 Infinite buffers The case of infinite buffers is basically a theoretical case in which no limitation exists with respect to the number of jobs that may be buffered between two stations. Roy, 1962, presents a graph-theoretical interpretation which allows the calculation of the makespan of a sequence in the flowshop that is not a permutation sequence.

3.2.1.2 Large ASRS buffers Large buffers are used, for example, in the automobile industry. Production lines in automobile industry are divided into three parts: body-, paint- and assembly-shop. Two buffers are located between each part of the production line. The reason for using large buffers in this case is to resequence the jobs in a large scale. As a result, batches are formed and each shop is optimized separately. Only in the case in which the optimal sequence for one shop is the same as for the following, no resequencing is performed.

Solution techniques using large buffers, called **Automatic Storage and Retrieval System** (ASRS), due to Lee and Schaefer, 1997, were introduced in the automobile industry in the 1950s. Choi and Shin, 1997, call their ASRS-buffer a painted-body-storage (PBS) which uses a dynamic sequencing method for the resequencing. Their buffer has various rows and the jobs arriving at the buffer then get loaded to the row that already has jobs loaded being similar.

Lee and Schaefer, 1997, present an approach to statically and dynamically load an ASRS buffer. Two modes are used, in the single cycle a job is stored or retrieved within one cycle and in the double cycle job is stored and another retrieved in the same cycle. The objective is to minimize the total travel time required by processing the entire storage and retrieval.

Inman, 2003, study how to undo undesired scramble of the original sequence for automotive production. The scrambling is caused by parallel inspection stations with random inspection times and repair loops. The ASRS size here depends on the most negative sequence displacement, the goal is to obtain a 90% service level.

Ding and Sun, 2004, utilize a buffer, placed between two successive parts of the automobile production, to resequence a fixed number of vehicles. The algorithm determines a loading sequence for k incoming vehicles to a buffer

with k storing places and then releases them downstream. The resequencing can also be used to overcome unintentional sequence alternation caused by rework or equipment breakdowns. An exact and a heuristic solution are presented. The method can only be used with a large buffer because the number of vehicles to be resequenced at once is depending on the buffer size. Furthermore, the method overlooks the fact that it might be beneficial to incorporate the possibility of determining an improved sequence already at the beginning of the line that would result in a better overall solution.

3.2.1.3 Small buffers The buffer, initially used for decoupling, basically forms a serial chain of the incoming jobs. Two operation modes exist: The **free-access operation** permits to remove any stored job from the buffer, in contrast to the **FIFO operation** where only jobs can be removed from the buffer in first-in-first-out sequence. The buffer location can be **intermittent**, located between two stations, or **offline**, removing the job from the line. An intermittent buffer of size larger than one can be used for resequencing, if not operated in FIFO mode. The **offline** buffer may be operated in either mode; jobs remaining in the line bypass the buffered jobs and resequencing takes effect. As compared to the large ASRS buffers, the use of essentially smaller sized buffers conduces to a completely different implementation. It has to be clarified how removing or adding a job affects the line. As explained by Pinedo, 1995, mathematically a buffer is a station with zero processing time.

Lahmar et al., 2003, and Lahmar and Benjaafar, 2003, study the problem of resequencing a pre-arranged set of jobs with the objective of minimizing changeover costs. The example under study is the paint shop of an automobile production, a setup cost appears every time a color change occurs and paint is flushed. For example the cost for a metallic paint is higher than for an ordinary paint. The study considers pull-off tables, originally designed for rework, which allow pull-off one car at a time. A color assignment matrix is used to define the possible colors with which a car may be painted. The problem with 1 buffer is solved optimally and with N buffers the problem is decomposed to N problems with 1 buffer, not guaranteeing optimality. The conclusion shows that the biggest amount of cost saving is achieved by only implementing few pull-off tables in the line.

The introduction of buffers to a production line also has negative effects. Amongst these are the increase of work-in-process, the additional cost for installation and maintenance and the increase of necessary area. All of these factors are cost or time relevant issues and should not be overlooked in the design of the production line.

3.3 Hybrid or flexible flowshop

Hybrid flowshop problems overcome one of the limitations of the classical model of flowshops by allowing parallel stations. As described earlier, the use of parallel stations in a mixed-model production line gives the possibility of one job overtaking its predecessor. The parallel station may also serve as a buffer with no processing time, and let various jobs pass by the buffered job. This however is not a desirable case because the actual parallel station may block a job from being processed and therefore provides only limited resequencing possibilities in terms of cost savings.

Sawik, 2000, presents a mixed integer programming formulation for a flexible flowshop with one or more identical parallel stations. Also blocking due to infinite intermediate buffers, alternative routing and reentrant of products is included. Additional constraints are introduced by Riane et al., 1998, which include jobs that are assigned to a certain parallel station and therefore the flow can not freely be determined. Next to the mixed integer programming formulation two heuristic procedures are proposed, based on dynamic programming and on branch and bound.

Pinedo, 1995, explains the heuristic method used at IBM for a flexible flowshop with limited buffers and bypass, called flexible-flow-line-loading (FFLL). The loading of job j is determined by minimizing the overload that is cumulated until job j. Further heuristics are described by e.g. the forward-and-backward heuristic of Li, 1997, and the divide-and-merge heuristic of Gendreau et al., 2001, and Sun et al., 2003.

3.4 Merging and splitting

Engstrőm et al., 1996, report the introduction of parallel segments of stations to the automobile manufacturer Volvo that permit to resequence jobs where the line split and merge. The splitting of a production line is somewhat more challenging due to the fact that two parallel lines may not perform the same options and constraints exist that may additionally influence the sorting.

3.5 Re-entrant flowshops

Within the studies of Sawik, 2000, also the re-entrant flowshop is considered that manufacture double sided printed-circuit-boards (PCBs). The PCBs run twice through the same line, first to assemble the bottom side and then to assemble the top side. Instead of assembling all PCBs on one side first, the optimum sequence may consist in interleaving the first PCBs already having finished the first run on the bottom side.

3.6 Change of job attributes (no physical change)

Instead of physically changing the job order Rachakonda and Nagane, 2000, mention the solution implemented in the automobile production of the Ford assembly plant in Wixcon, USA. Here a dynamic resequencing system is used, involving the swapping of cars by changing their attributes. Consequently it is not necessary to physically change the job position within the sequence.

3.7 Undesired resequencing

Undesired resequencing occurs in many real life arrangements in which jobs require repair, exist parallel inspection stations with unequal processing time or a problem occurs in the part delivery, necessary to process the sequence in the correct order.

Inman, 2003, and Korkmaz et al., study how to undo undesired scramble of the original sequence for automotive production. The scrambling is

caused by parallel inspection stations with random inspection times and repair loops. Inman uses a large buffer, ASRS-automated storage and retrieval system. Choi and Shin, 1997, call their ASRS-buffer a painted-body-storage (PBS) which uses a dynamic sequencing method for the resequencing. The considered buffer has various rows and arriving jobs then get loaded to the row that already has jobs loaded being similar.

3.8 Related work on resequencing

Péridy et al., 1999, uses a selection of elimination rules to reduce the search tree of the branch and bound algorithm proposed by Potts, 1980, for the permutation flowshop and by Carlier and Rebai, 1996, for the classical flowshop. The idea is to calculate improved bounds by not including certain sequences that result infeasible due to the elimination rules. The simplest elimination rule is for example the rule by Johnson. Pugazhendhi et al., 2002 and Pugazhendhi and Thiagarajan, 2004, study non-permutation sequences in flowline-based manufacturing system. The line is similar to the classical flowshop with some or all jobs having missing operations in some stations. This leads to a property which is not proper of the flowshop, i.e. the station precedence are not the same for all job. Hence, an optimum sequence can be obtained that is not feasible if the processing time would be infinitesimal small instead of zero.

Table 1: Resequencing Methods I

Type	Reference	Shop	Static Dynamic	Method	Objective	Sequence-Type	Observation
Infinite buffers	Roy, 1962	FS	S	-	-	Non-perm	Graph-theoretical interpretation for makespan calculation in a FS.
	Péridy et al., 1999	FS	S	E	Makespan	Perm/ Non-perm	Elimination rules for lower bounds.
	Pugazhendhi et al., 2002	FBMS	S	H	Makespan	Non-perm	Station precedence not same as for flowshop.
	Pugazhendhi and Thiagarajan, 2004	FBMS	S	H	Makespan Flowtime	Non-perm	Station precedence not same as for flowshop. Also considering sequence-dependent setup times.
AS/RS buffer	Lee and Schaefer, 1997	FS	S/D	E/H	Traveltime of jobs in buffer	Non-perm	AS/RS buffer operates in single- or double-cycle mode for load and unload.
	Choi and Shin, 1997	FS	D	H	Deviation from ideal sequence	Non-perm	Measures deviation of leaving jobs from the desired and sequences due to spacing constraints between jobs.
	Ding and Sun, 2004	FS	S	E/H	Sequence restauration	Non-perm	Buffer with k storing places to resequence k jobs.
	Inman, 2003	FFS	S/D	E/H	Sequence restauration	Non-perm	AS/RS buffer size depends on most negative sequence dspacement. Unwanted sequence-change occurs due to repair loops and parallel stations
	Korkmaz et al.,	FS	D	H	Sequence restauration	Non-perm	Unwanted sequence-change occurs due to part shortage or inspection stations.

Shop:

FS	Flowshop
FBMS	Flowline-based Manufacturing system
FFS	Flexible Flowshop
FS-PL	Flowshop with parallel lines

Method:

M	Mixed integer programming formulation
I	Integer programming formulation
E	Exact method
H	Heuristic method

Table 2: Resequencing Methods II

Type	Reference	Shop	Static Dynamic	Method	Objective	Sequence-Type	Observation
Small buffers	Lahmar et al., 2003	FS	S	I/E/H	Setup cost	Non-perm	Resequencing with feature assignment. Decomposition of problem with N buffers to N problems with one buffer.
	Lahmar and Benjaafar, 2003	FS	S	I/H	Setup cost	Non-perm	Study of resequencing limited by the number of jobs a certain job can move forward or backward.
Hybrid or flexible flowshop	Sawik, 2000	FFS	S	M	Makespan	Non-perm	Limited intermediate buffers. Station blocking and alternative processing routes are possible.
	Riane et al., 1998	FFS	S	M/H	Makespan	Perm/Non-perm	Exist jobs assigned to a certain parallel station.
	Pinedo, 1995	FFS	S	H	Makespan	Non-perm	Exist unlimited intermittent buffers.
	Li, 1997	FFS	S	H	Makespan	Non-perm	Forward-and-backward heuristic. Considers major and minor setups, part families and batch splitting. Only single station case.
	Gendreau et al., 2001	FFS	S	H	Makespan	Non-perm	Divide-and-merge heuristic. Considers setup-time. Only single station case.
	Sun et al., 2003	FFS	S	H	Makespan	Non-perm	Exist jobs assigned to a certain parallel station.

Shop:

FS	Flowshop
FBMS	Flowline-based Manufacturing system
FFS	Flexible Flowshop
FS-PL	Flowshop with parallel lines

Method:

M	Mixed integer programming formulation
I	Integer programming formulation
E	Exact method
H	Heuristic method

Table 3: Resequencing Methods III

Type	Reference	Shop	Static Dynamic	Method	Objective	Sequence-Type	Observation
Mergin and splitting	Engstrőm et al., 1996	FS-PL				Non-perm	Various parallel stations lead to the possibility of resequencing.
Re-entrant flowshop	Sawik, 2000	FFS	S	M	Makespan	Non-perm	Limited intermediate buffer. Also re-entrant of jobs is considered.
Change of job attributes	Rachakonda and Nagane, 2000	FS	S	H	Setup cost	Non-perm	Swapping of cars in the sequence, without changing their physical location, but changing their other attributes.

Shop:

FS	Flowshop
FBMS	Flowline-based Manufacturing system
FFS	Flexible Flowshop
FS-PL	Flowshop with parallel lines

Method:

M	Mixed integer programming formulation
I	Integer programming formulation
E	Exact method
H	Heuristic method

4 STATIC AND DYNAMIC DEMAND

The production planning for mixed model flowshops can be various and usually depends on the planning horizon and in some cases on the possibility of decoupling the customer orders from the production planning. In the latter case the incoming customer orders may only give a guideline for what needs to be produced in the plant. For example, it might be advisable to produce a second part of a single-part order. This results from the fact that a negligible additional cost occurs, and in case of a quality problem with one part, a second one is available as reserve. On the other hand it might be advisable for a production not to start the production of this single-part order until a reasonable number of the same parts have accumulated.

Even though many sequencing procedures take into account a fixed, static, demand, it would also be desirable to include urgent customer orders when the production is already up and running. This then leads to the need of dynamic planning.

Engel et al., 1997, distinguish between the two cases of build-to-plan and build-to-order production. The two terms describe the dynamics of the input sequence. In the first case the demand is well known in advance and only a few model types are produced repeatedly. In the latter case each product is determined by an individual selection of options corresponding to a customer order.

In what follows, more detailed explanations of the characteristics of static and dynamic production planning are given. These individual cases lead to intermediate cases that may be found in the transition from the static to the dynamic case.

4.1 Static demand

In a sequencing problem a demand is called static if the information required forming a feasible but not necessarily optimal sequence is known before the first job is processed. In this case the sequencing and the execution of the sequence are considered consecutively. Díaz et al., 2003, mentions that this type of production planning is often referred to as off-line sequencing (scheduling). The static production planning accumulates customer orders

to lots; the lot then is sequenced and released to the production. Burns and Daganzo, 1987, highlight that an increased lotsize allows more efficient sequences to be identified, whereas longer lead times for customer orders are necessary.

Minimal-part-set (MPS): One of the most common representations is the use of the MPS which is the least common multiple of the individual models for the entire demand. Repetition of the MPS leads to the required demand. See for example Sarker and Pan, 1998, Bard et al., 1992, or Hyun et al., 1998.

The MPS is considered to be a good choice, however, Klundert and Grigoriev, 2001, show that many times it is not the best choice. They use a travellings salesperson problem (TSP) where the cities are to be visited various times in order to resolve the problem and conclude that better sequences are possible which obviously result in extended computational effort. Seasonal production is a classical example of a static production planning which refers to a production demand that is known for a predetermined period of time, i.e. the season.

4.2 Dynamic demand

A sequencing problem is called dynamic if the sequence is constructed while jobs are already entering the flowshop, the sequence is determined online. This is typically the case if the total demand or production relevant parameters are not known in advance, but becomes available during the execution. The sequencing is then only possible for jobs we have already knowledge of, i.e. it can only be done on the basis of a limited planning horizon; while new jobs arrive, the current sequence has to be updated appropriately. This uncertainty of timing in the production leads to the difficulty of controlling efficiently the material supply of the line and respond to the customer orders in a predictable manner. In order to apply an adequate measure of performance, the objective functions, presented in part 2.3, may be modified. Rather than the absolute value, the objective function should determine the mean value over the jobs.

Smed et al., 1999, mention various reasons that lead to dynamic demand planning: machine breakdown, component shortage and maintenance delay, urgent prototype series surpassing normal production, and the production plan itself which can be subject to sudden alterations during the production period.

Vieira et al., 2003, in their framework of resequencing (rescheduling), mainly for the dynamic case, furthermore present common performance measures of resequencing procedures. A separation is done into measures of schedule efficiency, schedule stability, and cost.

4.3 Transition from pure static to dynamic demand

The above mentioned separation into static and dynamic demand is not always realistic and leads to the use of models that share properties of both of the two individual cases. In fig.1 three different cases demonstrate the transition from the static to the dynamic demand. In the first case (1) the demand is known beforehand and permits an off-line optimization, (2) is the intermediate case in which the demand is known with a limited time horizon. (3) then is the dynamic case in which the orders arrive without advise.

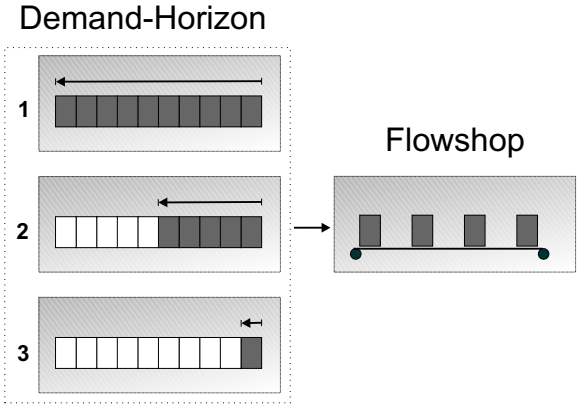

Figure 1: Illustration of the demand horizon. 1) entire demand is known beforehand and permits an off-line optimization 2) demand is known with a limited horizon and 3) demand is not known. The last case is the most dynamic one and does not permit sequencing before the jobs enter the first station of the flowshop and therefore has to be calculated on-line.

These variations lead to different solution techniques that focus on arranging the job order before the jobs enter the line (sequencing) or involve the option of reordering the jobs in the line (resequencing). In what follows, the distinct configurations of possible demands is listed which also are shown in fig.2.

- **Static case (permutation sequence)**: Determination of the optimal permutation sequence. The demand may be defined by the Minimal-Part-Set.
- **Static case (non-permutation sequence)**: Determination of the overall optimal sequence, also including resequencing within the line. The demand may be defined by the Minimal-Part-Set.
- **Semi-Dynamic case**: The focus here is the determination of a better sequence for a given incoming sequence by using e.g. buffers, except in the first station. In the case in which the demand is completely known, the resequencing is calculated off-line.
- **Nearly-Dynamic case**: Determination of introduction of jobs to the plant with dynamically changing demand. A predetermined number of jobs ready to enter into the plant is given which can be ordered before entering into the line. In this case not only sequencing before the line (permutation) but also resequencing in the line (non-permutation) might be considered.
- **Dynamic case**: Here the jobs enter the first station without the possibility of sequencing beforehand. This predetermined sequence then is resequenced within the line in order to improve the production.

4.4 Solution techniques

Solution techniques are various and basically are separated depending if a static or dynamic case is studied. Furthermore, it depends on the size of the problem. In the static case with moderate size optimum seeking algorithms like branch and bound or dynamic programming is applied; see for example Dar-El and Cucuy, 1977, Sarker and Pan, 2001, or Stafford and Tseng, 2002. Larger sized static problems are generally solved with heuristic procedures

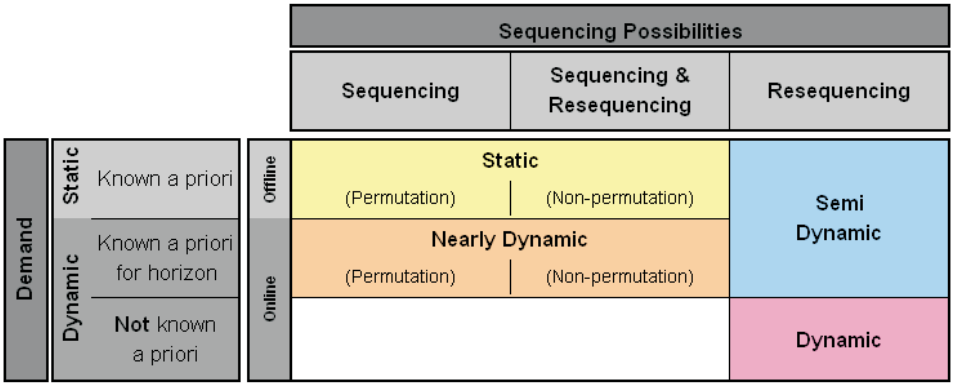

Figure 2: Sectioning of Static and Dynamic demand as a function of sequencing possibilities and the demand horizon.

that for the sake of computation do not ensure an optimal sequence, see for example Bard et al., 1992, some of which are derivations of exact solutions, see Ríos-Mercado and Bard, 1999. In the more complex case of dynamic process planning procedures are used called dispatching rules that give priorities in the selection of jobs in a queue formed in the buffers before the stations. These dispatching rules also find application in scheduling problems and in jobshop applications where the routing of the individual jobs is different for each job.

5 OPTIMIZATION METHODS

Various attempts with distinct approaches exist to solve the problems occurring in flowshops. The simplest attempt would be an enumeration of all the possible solutions, obviously leading to total number of $(n!)^m$. This would result in over 24 billion possibilities for a problem as simple as 5 jobs and 5 stations. All attempts have in common to start with taking apart the entire problem in order to isolate the static part of the problem, namely the line itself. The optimization routine then applies the input data for the line, together with the data for demand, resulting in a measure of efficiency, obtained by some objective function.

The separation of the numerous optimization methods with all its modifications usually concerns two major divisions. Primarily exist exact, optimum

seeking, methods which by nature are limited in problem size. On the other hand the non-exact methods, namely heuristics and metaheuristics. In the latter case the literature presents detailed algorithms, as well as general rules on how to proceed with the search of good solutions. Obviously, gaining a reduction in computational time results in a reduction of quality. This reduction, however, can many times lead to near optimum solutions which for real life problems are indispensable.

5.1 Elements of optimization

The assembly line optimization consists of several parts which can firstly be treated separately in order to establish a consistent and structured model. Fig.3 shows the main parts: **demand**, **optimization variables**, **line parameters**, **line efficiency**, and the **optimization procedure**.

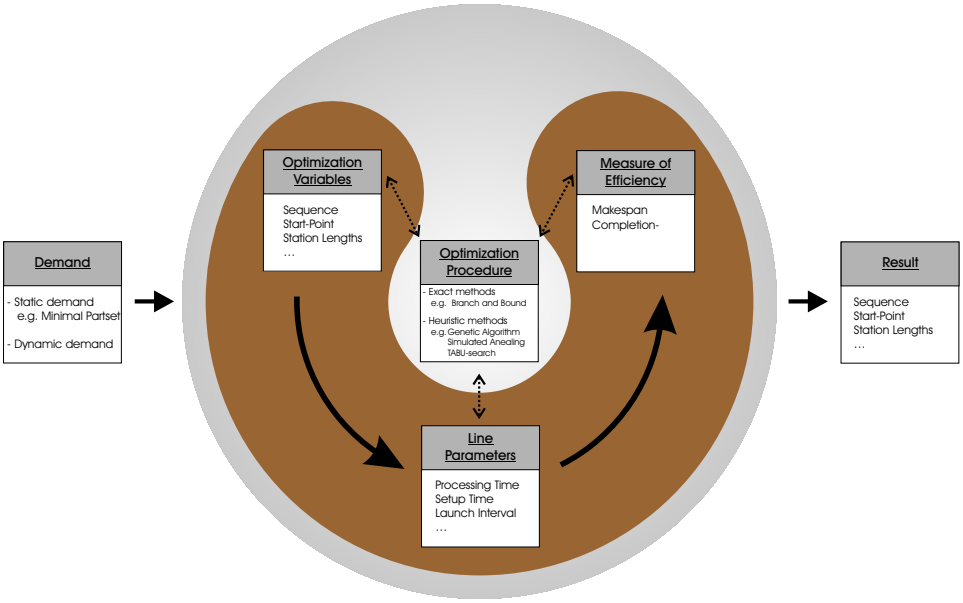

Figure 3: Structure of the sequencing problem in assembly line.

As described in the previous parts of this document, two lines may have very distinct aspects and objectives to optimize. Nevertheless, before starting with the simulation and optimization, it is necessary to model the line and define all the surrounding parameters that concretize and limit the problem in question.

- **Demand**: The demand is synonym to customer order. No matter if a static or a dynamic case is studied, the demand describes the amount of products that need to be processed. This is true for sequencing problems; in scheduling problems the demand may furthermore be accompanied by start- and due-date.

  The customer orders may be accumulated to lots and then be processed all at once, which is the most static case and many times the Minimal-Part-Set (MPS) is used to work only with the least-common-multiple of the different models to be produced. More dynamic cases imply that customer orders arrive while the production has already started.

- **Optimization variables**: The optimization variables in principle are nothing else than line parameters, with the difference that they define the pool of variable parameters that are to be optimized by the optimization procedure. It is desirable to keep the number and ranges of optimization parameters relatively small in order to limit the considered problem to a tractable one. Many times an initial set of parameters that describe a feasible solution is found by a fast heuristic and serves as an initial bound. Depending on the objective of optimization and the possibilities that provide a line-arrangement, the optimization parameters can be various, such as launch interval, station length, buffer size, etc.

- **Line parameters**: The line parameters describe the pool of static parameters that are defined beforehand and are fixed for the entire optimization procedure. For example, the sequencing problem in flow-shop optimization in general uses a constant launch interval that is obtained by the design and balancing procedure, performed beforehand. Other line parameters may be assembly-time, setup cost/time, buffer location, precedence relations, etc.

- **Line efficiency**: The line efficiency is a measure of performance and is calculated with the objective function to be optimized. This measure of efficiency in the general case is based on a single objective, but also combined objectives are possible that, e.g., minimize makespan as well as the idle time of operators.

- **Optimization procedure**: Once the line is defined with all its input and output parameters, an optimization procedure is needed to optimize the line efficiency by, e.g., varying the line parameters, always taking into account that the demand has to be satisfied. Exist basically two groups of methods: the exact methods that obtain the optimal solution, if existent, and approximation methods which for computational merit not necessarily obtain the optimal solution.

5.2 Complexity of problems

For the classification of combinatorial problems, such as sequencing in flowshop, a concept is used that represents the computational effort related with the problem in question. A problem is assigned to the P-problem (polynomial-time) class if an algorithm exist which solves the problem in polynomial time. Thereafter, a problem is assigned to the NP-problem (nondeterministic polynomial-time) class if it is possible to solve it within polynomial time on a hypothetical non-deterministic Turing machine, allowing parallel executions. The class of P-problems is a subset of the class of NP-problems, but there also exist problems which are not NP.

In the problem classification of Pinedo, 1995, several sequencing problems in flowshop production lines are specified. Polynomial time solvable problems $F_2|block|C_{\max}$ and $F_2|nwt|C_{\max}$ are two station cases with blocking and no-wait respectively. Also a special case for m stations, $F_m|p_{ij} = p_j|\sum C_j$ with equal processing times for a product j at all stations is part of these problems. Within the strongly NP-hard problems $F_2||\sum C_j$ is the two station case with total flowtime optimization and $F_3||C_{\max}$ and $F_3|nwt|C_{\max}$ are three station cases with and without no-wait sequences and the makespan objective. Extended explanations on computational complexity can be found in Garey and Johnson, 1979.

5.3 Exact methods

Exact methods always lead to the optimal solution. Clearly exists a limit on problem size for exact methods that is way inferior compared to heuristic methods. In order to reduce the complexity for the necessary computation

it is advisable to isolate the problem as tight as possible, i.e. when dominant rules can be used such as permutation sequences are sufficient for the two station case, it is not necessary to consider all possible sequences at all stations.

In order to find the minimum or maximum of an objective function the most obvious approach would be of analytical nature, using its derivation (**Calculus** and **Lagrange multipliers**). It is important that the function and its first derivation is continuous. As highlighted by Nicholson, 1971, these methods can be used only on small problems containing few continuous variables with simple objective functions and no constraints, therefore clearly is not applicable to flowshop sequencing problems.

- **Complete enumeration**: The simplest way to solve the sequencing problem in flowshops is the complete enumeration. The solution for all sequences on all stations is calculated. The method is not very efficient and as described before has a complexity of $(n!)^m$.

- **Branch and bound**: A widely used and very promising strategy in many problems seem to be the branch and bound approach, being an implicit enumeration or tree search method which can find an optimal solution by systematically examining the subsets of feasible solutions (Bellman et al., 1982).

  The branch and bound method was developed by Little et al., 1963, for the use on the travelling salesperson problem and modified by Ignall and Schrage, 1965, and Lomnicki, 1965, for application to flowshop sequencing. It is also called back-track programming and uses generally one of the three seeking strategies: depth first which first explores a feasible sequence, breadth-first which completely explores the branches of the same level before descending in the tree, and frontier or branch-from-lowest-bound which explores the solution space in terms of the most promising branch. French, 1982, highlights that the first method results in more computational expense but at the same time requires less storage, compared to the latter.

  Ignall and Schrage, 1965, as well as Lomnicki, 1965, independently developed a branch and bound method for the three station case which is presented by French, 1982. A considerable analysis of tree search methods, related with branch and bound is presented by Pastor, 1999,

also including a resolution method called branch and win (Pastor and Corominas, 2004).

- **Linear, integer and mixed integer programming**: With the help of linear programming large problems with continuous variables can be solved, provided that the objective function is linear and constraints are in the form of linear equalities and inequalities. The simplex method, first published by Dantzig in 1948 (Dantzig, 1948), solves problems of this type. Nicholson, 1971 and Luenberger, 1984 give detailed introductions on linear programming.

  Integer and mixed integer programming finds its application on problems which consider that all or some of the variables are constrained to integer values. In contrast to linear programming the solution of these problems is NP-hard.

- **Dynamic programming**: This method originates from Bellman, 1957, and can be used in multi-stage decision processes, in particular types of problem structures. Just as the branch and bound method it is limited to comparatively small problems. The method tries to reduce the dimensionality of a problem and one of the challenges is to convert the actual problem to a multi-stage decision process in order to solve it recursively. Bautista et al., 1996, furthermore present an algorithm based on bounded dynamic programming (BDP) for the sequencing problem in JIT, known as the Monden problem (Monden, 1983). Dynamic programming is used here with the additional use of bounds, obtained solving the problem with a heuristic algorithm.

5.4 Approximate methods

These methods do not guarantee an optimal solution. The idea is to reach solutions that, for the sake of computation or storage, are near optimal solutions. In order to exhaustively study their performance it is indispensable to compare the obtained solution with the optimal solution obtained by an exact method. Challenging problems are found when non-linear objective functions are given. Nicholson, 1971, highlights that here the difficulty lies in the fact that a non-linear objective function is used that may have local minima and therefore usually finds solutions that describe local optima and do not result in the optimal solution.

5.4.1 Heuristics

Exist a wide range of heuristic methods and some of them are based on exact methods. The simplest being the **randomly generated solutions**. Lee and Shaw, 2000, mention that heuristic methods either belong to **improvement heuristics**, also called **neighborhood search**, or **constructive heuristics**. Improvement heuristics attempt to continuously improve the solution by modifying the sequence, whereas constructive heuristics build a sequence of jobs and once a decision is taken, it cannot be reversed. The first usually generates better sequences at the expense of larger computation. The most simple neighborhood search starts with a feasible initial solution that is used as initial seed. In the next step solutions are evaluated that are part of the neighborhood of the seed. If none of the solutions is better than the seed with respect to the performance measure it is the final solution. Otherwise the solution that was proved to be better is used as the new seed. This procedure obviously is not capable of avoiding falling in a local minimum.

Silver, 2004, additionally annotates the separation to **inductive methods**, which first solve a mathematically related problem and extends it to the actual problem, **reduction of solution space**, here the possible solutions are drastically cut back, and **approximation methods**, here a solution for the approximated mathematical model is to be calculated.

A special case of heuristic technique is the priority dispatching rules which are used for a production with buffers between the stations that provide the possibility of selecting any of the stored jobs. Priority dispatching stands for the procedure of selecting a job from a queue with the highest priority. The priority may be calculated depending on various measures. The rules can be as simple as "first come first serve" (FCFS), "shortest processing time" (SPT), "longest total remaining processing on other machines" (LTROM) but also may be a combination of simple rules. Dispatching rules are also called greedy-procedure since it makes the selection in the most favorable way, without regard to the possibilities that might arise later. Collections of existing rules can be found in Pinedo, 1995, or Blackstone et al., 1982. These dispatching rules also find application in jobshop arrangements where the routing of the individual jobs is different for each job.

5.4.2 Metaheuristics

The survey of Silver, 2004, describes metaheuristics as high level heuristic procedure which are designed to guide towards achieving reasonable solutions. Metaheuristics are particularly concerned with not getting trapped at a local optimum. Metaheuristics have one or more adjustable parameters which permits flexibility, but for any application requires careful calibration.

- **Simulated annealing**: This method goes through number of iterations, comparing the current best sequence with a neighbor, see e.g. Kirkpatrick et al., 1983. Further a probability indicates if a worse sequence is chosen next. The major difference with the general neighboring search is that worse solutions are allowed in order to find an optimum that is not the next local optimum.

- **Tabu search**: Exist a Tabu-list that prevents returning to a local minimum. The procedure stops when a predetermined number of moves are performed without having improved the seed, see e.g. Glover, 1990. An essential parameter of the technique is the list size; if chosen wrong the search may result in cycling or may be constrained unduly.

- **Genetic algorithm**: Rather than using a single initial solution this technique uses an increased number of initial solutions, see e.g. Kim et al., 1996 or Hyun et al., 1998. These solutions, called individuals, form a generation. The succeeding generation is generated through reproduction and mutation of individuals that were part of the previous generation. Individuals are also referred to as chromosomes and are characterized by their fitness, which is measured by the associated objective function. Mutation then is achieved by cross over of chromosomes. Hyun et al., 1998, highlights the two characteristics of the genetic algorithm which are exploitation and exploration. Exploitation is the ability to find good solutions quickly, whereas exploration describes the behavior of maintaining a set of diverse individuals.

- **Beam search**: Beam search is a partial branch and bound with the basic idea to eliminate the branches of the tree that are likely to not include the optimal solution, see Morton and Pentico, 1993. The used parameter is the beam width that is the number of nodes that are retained at each level.

- **Ant colony**: The basic concept is to mimic the pheromone trail used by real ants as a medium for communication and feedback among ants and was first proposed by Dorigo et al., 1996. The algorithm is a population based, cooperative search procedure which iteratively construct solutions, guided by (artificial) pheromone trails. Solution components are defined which the ants use to iteratively construct solutions, the ants mark the utilized components with pheromone. The pheromone then indicate the intensity of ant-trails with respect to solution components,and such intensities are determined on the basis of the influence or contribution of each solution component with respect to the objective function. Rajendran and Ziegler, 2004 apply the concept on permutation flowshop.

- **GRASP**: The GRASP (greedy randomized adaptive search procedure) is an iterative process, in which each iteration consists of two phases. A construction phase, in which a feasible solution is produced, and a local search phase, in which a local optimum in the neighborhood of the constructed solution is sought. The best overall solution is kept as the result. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the restricted candidate list (RCL). A recent survey on GRASP is presented by Festa and Resede, 2004.

5.5 Evaluation of the performance of a heuristic

Dannenbring, 1977, uses a set of six different evaluation measures for testing of heuristic procedures: (1) Relative error, (2) Consistency, (3) Error Potential Ratio (4) Percentage of solution equaling the optimum, the estimate of the optimum or the best heuristic solution, (5) Improvement Potential and (6) Sampling quality.

Silver, 2004, furthermore explains that two main measures of performance exist to evaluate the performance of a heuristic method. First, compar-

ing the obtained value of the objective function to the achievable by the optimal solution or some other benchmark procedure, already mentioned in Dannenbring, 1977. Then, second, the computational requirements in terms of computational effort and memory consumption for realistic sized problems. In particular, for what percentage of the problem instances does the heuristic obtain the optimal solution. If the results are found to be sensitive, then it is helpful to specify under which conditions the heuristic should or should not be used.

In case no optimal solution is available the heuristic solution can, for a minimization, be compared with the lower bound. For a small gap the difference to the optimal solution is small. For a large gap, however, a poor lower bound was used, or heuristic solution is far away from the optimal solution.

5.6 Literature providing input data

For the case of larger problem sizes it is more difficult to find adequate input data. This is contingent on the great variety of existent problems and the fact that many times only results are presented rather than input data. Rajendran and Chaudhuri, 1992, use a test-bed of 670 problems considering 5, 10, 15, 20 and 25 stations as well as 5, 6, 7, 8, 9 and 10 jobs, also used by Framinan et al., 2002.

Taillard, 1993, presents a widely used test-bed. A detailed explanation on the generation of input data for various production lines, including the flowshop problem is given. A random number generator is used, based on Bratley et al., 1983, that produces evenly spread values and problems with 5, 10 and 20 stations and from 20 to 500 jobs are proposed. The advantage of this generator is the repeatability in the generation of the test-bed.

6 SUMMARY

The problem of sequencing in **mixed model flowshops** is known to be NP-hard already for the simple case of three stations. Johnson, 1954, one of the pioneers, proposed a solution that solves the case of two stations with optimality, known as the Johnson-rule. Here permutation sequences are

dominant, i.e. the job sequence is the same for all stations. This simplification indeed is only valid until three stations case with makespan objective. If more than three stations exist Potts et al., 1991, show that considerable improvements are possible, using resequencing capacities. This is even more evident when setup cost/time, necessary when changing from one product model to the next, appears.

In the present work a literature overview was given on solutions techniques considering basic as well as more advanced and consequently more complex arrangements of mixed model flowshops. We first analyzed the occurrence of **setup time/cost**; existing solution techniques are mainly focused on permutation sequences. Thereafter we discussed objectives resulting in the introduction of a variety of methods allowing **resequencing** of jobs within the line. The possibility of resequencing within the line ranges from 1) offline or intermittent buffers, 2) parallel stations, namely flexible, hybrid or compound flowshops, 3) merging and splitting of parallel lines, 4) re-entrant flowshops, to 5) change of job attributes without physically interchanging the position. In continuation the differences in the consideration of **static and dynamic demand** was studied. Also intermittent setups are possible, depending on the horizon and including the possibility of resequencing, four problem cases were highlighted: static, semi dynamic, nearly dynamic and dynamic case. Finally a general overview was given on existent **solution methods**, including exact and approximation methods. The approximation methods are furthermore divided in two cases, known as heuristics and metaheuristics.

Reviewing the literature regarding sequencing considerations in mixed model flowshops, it seems that extended work is done on the classical flowshop, mainly focusing on permutation sequences. Furthermore exist a remarkable amount of publications solving flowshops with setup time/cost; in contrast to arrangements taking into account resequencing capabilities within the line, resulting in non-permutation sequences. Besides the fact that solution methods in the literature are not numerous, it seems that production lines in the industry are not yet considering the advantages that come with the possibilities of resequencing jobs in a mixed model production. This may either be caused by additional hardware to be installed, like buffers, but also due to extra efforts in terms of logistics complexity. Clearly, these additional efforts are reasonable if the resequencing pays off the necessary investment.

References

Aldowaisan, T. and Allahverdi, A. (1998). Total flowtime in no-wait flowshops with separated setup times. *Computers & Operations Research*, 25(9):757–765.

Allahverdi, A. (2003). The two- and m-machine flowshop scheduling problems with bicriteria of makespan and mean flowtime. *European Journal of Operational Research*, 147(2):373–396.

Allahverdi, A., Gupta, J., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, 27(2):219–239.

Azizoglu, M., Cakmak, E., and Kondakci, S. (2001). A flexible flowshop problem with total flow time minimization. *European Journal of Operational Research*, 132(3):528–538.

Baker, K. (1974). *Introduction to Sequencing and Scheduling*. John Wiley & Sons, Inc., New York.

Baker, K. and Scudder, G. (1990). Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 38:22–36.

Bard, J., Dar-El, E., and Shtub, A. (1992). An analytic framework for sequencing mixed model assembly lines. *International Journal of Production Research*, 30(1):35–48.

Bautista, J., Companys, R., and Corominas, A. (1996). Heuristics and exact algorithms for solving the monden problem. *European Journal of Operational Research*, 88(1):101–113.

Beaty, S. (1992). Genetic algorithms for instruction sequencing and scheduling. In *Workshop on Computer Architecture Technology and Formalism for Computer Science Research and Applications*. Naples, Italy.

Becker, C. and Scholl, A. (2003). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*. working paper.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Bellman, R., Esogbue, A., and Nabeshima, I. (1982). Mathematical aspects of scheduling and applications. *International Series in Modern Applied*

Mathematics and Computer Science, 33(5):769–772. Pergamon Press, Volume 4.

Blackstone, J., Phillips, D., and Hogg, G. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, 20(1):27–45.

Bolat, A. (1994). Sequencing jobs on an automobile assembly line: objectives and procedures. *International Journal of Production Research*, 32(5):1219–1236.

Bratley, P., Fox, B., and Schrage, L. (1983). *A guide to simulation*. Springer-Verlag, New York.

Bukchin, J. (1998). A comparative study of performance measures for throuput of a mixed model assembly line in a jit environment. *International Journal of Production Research*, 36(10):2669–2685.

Burns, L. and Daganzo, C. (1987). Assembly line job sequencing principles. *International Journal Production Research*, 25(1):71–99.

Campbell, H., Dudek, R., and Smith, M. (1970). A heuristic algorithm for the n-job,m-machine sequencing problem. *Management Science*, 20(10):630–637.

Carlier, J. and Rebai, I. (1996). Two branch and bound algorithms for the permutation flowshop problem. *European Journal of Operational Research*, 90(2):238–251.

Cepek, O., Okada, M., and Vlach, M. (2002). Nonpreemptive flowshop scheduling with machine dominance. *European Journal of Operational Research*, 139(2):245–261.

Choi, W. and Shin, H. (1997). A real-time sequence control system for the level production of the automobile assembly line. *Computers and Industrial Engineering*, 33(3-4):769–772.

Companys, R. (1992). Algoritmo lomnicki pendular. *Technical Report, Universitat Politècnica de Catalunya, Spain*.

Dannenbring, D. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, 23(11):1174–1182.

Dantzig, G. (1948). *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ.

Dar-El, E. and Cucuy, S. (1977). Optimal mixed-model sequencing for balanced assembly lines. *Omega*, 5(3):333–342.

Díaz, A., Tchernykh, A., and Ecker, K. (2003). Algorithms for dynamic scheduling of unit execution time tasks. *European Journal of Operational Research*, 146(2):403–416.

Ding, F. and Sun, H. (2004). Sequence alteration and restoration related to sequenced parts delivery on an automobile mixed-model assembly line with multiple departments. *International Journal of Production Research*, 42(8):1525–1543.

Dorigo, M., Maniezzo, V., and A., C. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and CyberneticsPart B*, 26(1):29–41.

Engel, C., Zimmermann, J., and Steinhoff, A. (1997). *Objectives for Order-Sequencing in Automobile Production: Drexl, A. and Kimms, A. 1998, Beyond Manufacturing Resource Planning (MRP II) - Advanced Models and Methods for Production Planning*. Springer, Berlin. 307-331.

Engstrőm, T., Jonsson, D., and Johansson, B. (1996). Alternatives to line assembly: Some swedish examples. *International Journal of Industrial Ergonomics*, 17(3):235–245.

Erel, E. and Sarin, S. (1998). A survey of the assembly line balancing procedures. *Production Planning and Control*, 9(5):414–434.

Festa, P. and Resede, M. (2004). An annotated bibliography of grasp. Technical Report TD-5WYSEW., AT and T Labs Research.

Framinan, J., Gupta, J., and Leisten, R. (2002). A review and classification of heuristics for permuation flowshop scheduling with makespan objective. *Technical Report OI/PPC-2001/02*. Version 1.2.

Framinan, J. and Leisten, R. (2003). Comparison of heuristics for flowtime minimisation in permuation flowshops. *Technical Report IO-2003/01*. Version 0.5.

French, S. (1982). *Sequencing and Scheduling*. John Wiley, New York.

Garey, M. and Johnson, D. (1979). *The complexity and Intractability-A guide to the theory of NP-Completeness*. W.H.Freeman and Company, NY.

Garey, M., Johnson, D., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:117–29.

Gendreau, M., Laporte, G., and Guimaraes, E. (2001). A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup-times. *European Journal of Operational Research*, 133(1):183–189.

Ghosh, S. and Gagnon, R. (1989). A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 27(4):637–670.

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, 20(4):74–94.

Gupta, J. and Reddi, S. (1978). Improved dominance conditions for the three-machine flowshop scheduling problem. *Operations Research*, 26:200–203.

Hyun, C., Kim, Y., and Kim, Y. (1998). A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. *Computers Operations Research*, 25(7/8):675–690.

Ignall, E. and Schrage, L. (1965). Application of the branch and bound technique to some flow-shop problems. *Operations Research*, 13(3):400–412.

Inman, R. (2003). Asrs sizing for recreating automotive assembly sequences. *International Journal of Production Research*, 41(5):847–863.

Johnson, S. (1954). Optimal two and three stage production schedules with setup times included. *Naval Research Logistics Quaterly*, 1(1):61–68.

Kim, Y., Hyun, C., and Kim, Y. (1996). Sequencing in mixed model assembly lines: A genetic algorithm approach. *Computers Operations Research*, 23(12):1131–1145.

Kim, Y. and Kim, J. (2000). A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. *Applied Intelligence*, 13(3):247–258.

Kirkpatrick, S., Gelatt Jr., C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Klundert, J. and Grigoriev, A. (2001). Throuput rate optimization in high

multiplicity sequencing problems. In *Conference proceedings of the 17 International Symposium on Mathematical Programming*.

Korkmaz, O., Wood, D., and Gunal, A. Throughput analysis with respect to buffer and batch sizes using simulation. *http://www.pmcorp.com/pub_simulation.shtm*. consulted on 26.08.2004.

Korkmazel, T. and Meral, S. (2001). Bicriteria sequencing methods for the mixed-model assembly line in just-in-time production systems. *European Journal of Operational Research*, 131(1):188–207.

Lageweg, B., Lenstra, J., and Rinnooy, K. (1978). A general bounding scheme for the permutation flowshop problem. *Operations Research*, 26:53–67.

Lahmar, M. and Benjaafar, S. (2003). Sequencing with limited flexibility. *Operations Research*. working paper.

Lahmar, M., Ergan, H., and Benjaafar, S. (2003). Resequencing and feature assignment on an automated assembly line. *IEEE Transactions on Robotics and Automation*, 19(1):89–102.

Lee, H. and Schaefer, S. (1997). Sequencing methods for automated storage and retrieval systems with dedicated storage. *Computers and Industrial Engineering*, 32(2):351–362.

Lee, I. and Shaw, M. (2000). A neural-net approach to real time flow-shop sequencing. *Computers & Industrial Engineering*, 38(1):125–147.

Li, S. (1997). A hybrid two-stage flowshop with part family, batch production, major and minor set-ups. *European Journal of Operational Research*, 102(1):142–156.

Little, J., Murty, K., Sweeney, D., and Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, 11:972–989.

Lomnicki, Z. (1965). A branch and bound algorithm for the exact solution of the three machine scheduling problem. *Operations Research Quaterly*, 16:89–100.

Luenberger, D. (1984). *Linear and Nonlinear Programming*. Addison-Wesley.

Monden, Y. (1983). *Toyota Production System*. Institute of Industrial Engineering Press, Norcross, GA.

Morton, T. and Pentico, D. (1993). *Heuristic scheduling systems: with applications to production systems and project management*. Wiley, New York.

Nawaz, M., Enscore, E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. *Omega*, 11(1):91–95.

Nicholson, T. (1971). *Optimization in industry, optimization techniques*, volume 1. Longman Group Limited.

Pastor, R. (1999). *Metalgoritmo de optimización combinatoria mediante la exploración de grafos*. PhD thesis, Universitat Politécnica de Catalunya.

Pastor, R. and Corominas, A. (2004). Branch and win: Or tree search algorithms for solving combinatorial optimisation problems. *Sociedad de Estadística e Investigación Operativa*, 12(1):169–191.

Péridy, L., Pinson, E., and Rivreau, D. (1999). Elimination rules for the flow-shop and permuatation flow-shop problems. *http://www.math-appli-uco.fr*.

Pinedo, M. (1995). *Scheduling. Theory, Algorithms and Systems*. Prentice Hall Inc., Englewood Cliffs, New Jersey, 1 edition.

Potts, C. (1980). An adaptive branching rule for the permutation flowshop problem. *European Journal of Operational Research*, 5(2):19–25.

Potts, C., Shmoys, D., and Williamson, D. (1991). Permutation vs. non-permutation flow shop schedules. *Operations Research Letters*, 10(5):281–284.

Pugazhendhi, S. and Thiagarajan, S. (2004). Generating non-permutation schedules in flowline-based manufacturing systems with sequence-dependent setup times of jobs: a heuristic approach. *The International Journal ofAdvanced Manufacturing Technology*, 23(1/2):64–78.

Pugazhendhi, S., Thiagarajan, S., Rajendran, C., and Anantharaman, N. (2002). Performance enhancement by using non-permutation schedules in flowline-based manufacturing systems. *Computers and Industrial Engineering*, 44(1):133–157.

Rachakonda, P. and Nagane, S. (2000). Simulation study of paint batching problem in automobile industry. *http://sweb.uky.edy/˜pkrach0/Projects/MFS605Project.pdf*. consulted 14.07.2004.

Raghavachari, M. (1988). Scheduling problems with non-regular penalty functions: A review. *Opsearch*, 25:144–164.

Rajendran, C. and Chaudhuri, D. (1992). An efficient heuristic approach to the scheduling of jobs in a flowshop. *European Journal of Operational Research*, 61(3):318–325.

Rajendran, C. and Ziegler, H. (2004). Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, 155:426–438.

Riane, F., Artiba, A., and Elmaghraby, S. (1998). A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan. *European Journal of Operational Research*, 109(2):321–329.

Ríos-Mercado, R. and Bard, J. (1999). A branch-and-bound algorithm for permutation flow shops with sequence-dependent setup times. *IIE Transaction*, 31:721–731.

Roy, B. (1962). Cheminement et connexité dans les graphes-applications aux problémes d'ordonnancement. *Revue METRA, série spéciale*, (1). 130 pages.

Sarker, B. and Pan, H. (1998). Designing a mixed-model assembly line to minimize the costs of idle and utility times. *Computers and Industrial Engineering*, 34(3):609–628.

Sarker, B. and Pan, H. (2001). Design configuration for a closed-station, mixed-model assembly line: a filing cabinet manufacturing system. *International Journal of Production Research*, 39(10):2251–2270.

Sawik, T. (2000). Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling*, 31(13):39–52.

Scholl, A. (1995). *Balancing and Sequencing of Assembly lines*. Heidelberg: Physica-Verlag.

Silver, E. (2004). An overview of heuristic solution methods. *Journal of the operational research society*, 55(9):936–956.

Smed, J., Johnsson, M., Johtela, T., and Nevalainen, O. (1999). Techniques and applications of production planning in electronics manufacturing systems. *Technical Report No. 320*. Turku Centre for Computer Science.

Stafford, E. and Tseng, F. (2002). Two models for a family of flowshop sequencing problems. *European Journal of Operational Research*, 142(2):282–293.

Sun, X., Morizawa, K., and Nagasawa, H. (2003). Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. *European Journal of Operational Research*, 146(3):498–516.

Szwarc, W. (1971). Elimination methods in the *mxn* sequencing problem. *Naval Research Logistics Quaterly*, 18:295–305.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285.

Vieira, G., Herrmann, J., and Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies and methods. *Journal of Scheduling*, 6(1):39–62.