We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

4

Lot Processing in Hybrid Flow Shop Scheduling Problem

Larysa Burtseva[1], Rainier Romero[2], Salvador Ramirez[1],
Victor Yaurima[3], Félix F. González-Navarro[1] and Pedro Flores Perez[4]
[1]*Autonomous University of Baja California, Mexicali,*
[2]*Polytechnic University of Baja California, Mexicali,*
[3]*CESUES Superior Studies Center, San Luis Rio Colorado, Sonora,*
[4]*Univesity of Sonora, Hermosillo, Sonora,*
Mexico

1. Introduction

The scheduling problem in manufacturing systems, where the product components are processed by means of lot units (pallets, containers, boxes) of many identical items, has been recently subject of intense research in modern industry. Most of the common examples in production per lots can be found in serial production like digital devices manufacturing, assembly lines, and information service facilities. The lot processing and the use of batch machines for parallel execution are typical for modern manufacturing systems.

One of the most popular machine environments in scheduling is Hybrid Flow Shop (namely in the following as HFS), in which n jobs must be processed in series of m stages optimizing a given objective function, and at least one stage has parallel machines. In a traditional scheduling problem, a job is indivisible, and it cannot be transferred to the next machine before its processing is finished (Marimuthu et al. 2009; Potts & Baker 1989). On the contrary, the division of jobs into sublots in many situations is permissible and desirable in order to accelerate manufacturing or respect just-in-time (JIT) system through parallel processing. It is known that the HFS problem with m = 2 (HFS2) is NP-hard even if one of two stages contains a single machine (Gupta & Tunc, 1998). A HFS with lot processing has additional difficulties such as differences in sublot finish times on machine, unknown lot (sublot) sizes, setup times through lot sequencing, uneven lot sizes under customer request that lead to the demand splitting and allocation problem, etc. These kinds of problems possess a high computational complexity; therefore most of published literature is dedicated to simple flow shop or two-stage HFS machine environment.

Lot processing programming has differences in comparison with traditional job scheduling. On the one hand the job notation depends on the problem assumptions and requires an explicit interpretation; for the other hand one lot as well as one customer's demand can be considered as a job. The theoretical and practical interest on this issue represent models where job splitting is permitted -i.e. one job is allowed to split into sublots to process in parallel on the same or on various machines-. In many models, the sublot size is not clearly

evident and must be optimized. When a sublot requires a considerable setup time on machine, the complexity of the problem become increased. The sublots grouping is another important aspect of the lot processing.

The scheduling theory offers two concepts to describe and to solve problems that involve treatment of lots: lot streaming and job batching. These concepts commonly appear together when jobs are allowed to be split. In this chapter, both topics and other relevant problems are discussed in context of HFS machine environment. A solution on a real manufacturing problem is described, where they are involved demand partitioning, lot splitting and batching sublots applied to a HFS2 problem with identical parallel machines at the first stage, and dedicated machines at the second stage.

The following contribution is organized as follows:

- Section 2 introduces the HFS scheduling problem. Some properties and assumptions with certain particularities are discussed and referenced to representative and original scientific contributions in this field of research.
- In Section 3, four major concepts are explained in an in-depth fashion: The Group technology, which embodies jobs or demands according to similar attributes, is aimed to achieve high-volume production efficiency; the batch modeling, where its main definitions are explained and discussed, as long as known scientific papers; and Burn-in operations, a batch processing submodel commonly solicited in semiconductor manufacturing, representing one of the most complex problems in industry; and the Cell architectures that addressed the optimization problem –i.e. minimize production costs and maximize productivity– by the proper arrangement of resources. Additionally, the setup time treatment issue is discussed.
- Lot streaming concept is discussed in Section 4. It is a technic that describes how a job must be split to process and to improve the scheduling result.
- A real manufacturing problem is analyzed in Section 5. A real example in electronic components manufacturing is presented; where the concepts described in this chapter are applied in order to show their usability, in theoretical and practical modeling context.
- Finally, some conclusions in Section 6 are written.

2. Particularities of HFS with lot processing

The HFS scheduling problem is a generalization of the classical flow shop problem by permitting multiple parallel processors in a stage toward to increase the overall capacities or to balance the capacities of the stages, or either to eliminate or to reduce the impact of bottleneck stages on the shop floor capacities (Morita & Shio, 2005). However, the number of variations of this problem is enormous. The HFS differs from the flexible flow line (Kochhar & Morris, 1987) and the flexible flow shop (Santos et al., 1995) problems. In a flexible flow line as well as in a flexible flow shop, available machines in each stage are identical. The HFS does not have this restriction. Some stages may have only one machine, but at least one stage must have a group of machines in parallel. These machines can be identical or generally different. The flow of products is unidirectional. In a classical HFS, each job is processed by at most one machine at each stage and at only one machine at a time. One machine processes no more than one job at each time. The time processing at the stages is known for each job.

The HFS where there are parallel machines at certain stages is very common in industries, which have the same technological route for all products as a sequence of stages. HFSs have important applications in flexible manufacturing systems (FMS), such as electronics and furniture manufacturing, process industries such as chemical, textile, metallurgical, semiconductors, printed circuit board assembly lines, pharmaceutical, oil, food, automobile manufactures, steel making systems, etc. –see, e.g., Tang et al. (2002), Jin et al. (2002), Mathirajan & Sivakumar (2006) and Yaurima et al. (2009)–. Therefore, HFS scheduling has attracted interest by scientific community, being reflected in literature contributions dealing with this particular problem; however the issue of lot processing in this type of shops has not gained sufficient attention.

Let's establish some typical assumptions for a HFS scheduling problem with lot processing:

- There are k stages of processing which occur in a linear order: 1…k. Each stage has a predetermined number of parallel machines. However, the number of machines varies from stage to stage.
- A job represents the processing of entire lot of identical pieces, or a demand to produce a several quantity of identical pieces, which are processed in lots.
- Each job visits the stages in unidirectional order. Any stages may be skipped for a particular job but the process flow for all jobs is the same.
- The lot size and the unit processing time on every machine are known in advance and are constant. The sublots belonging to the same lot may have different processing times, a consequence of different sizes.
- Sublots which do not have in-between setup time, form a production batch and are processed together without requiring any machine adjustment. Sublots belonging to different batches may need an essential setup time. A batch machine can process several sublots simultaneously as long as machine capacity is not exceed.
- All sublots contained in the same batch, start and complete at the same time. Interruptions in a sublot and batch processing generally are not allowed.
- Buffers are located between stages to store intermediate products and are mentioned when they are limited.

The three-field notation $\alpha|\beta|\gamma$, named Graham's triplet and recently actualized in (Ruiz & Vazquez-Rodriguez, 2010), is used to describe different HFS variants. The α field denotes the shop configuration, including the shop type and machine environment per stage. The α field is decomposed into four parameter α_1, α_2, α_3, and α_4, and positioned as $\alpha_1\alpha_2$ ($\alpha_3\alpha_4^{(1)}$, $\alpha_3\alpha_4^{(2)}$, …,$\alpha_3\alpha_4^{(\alpha_2)}$). Here, parameter α_1 indicates the considered shop. The parameter α_2 represents the number of stages in the shop. In this case, a HFS is denoted as FH in the α_1 position and the parameter value of α_2 is major that one. In the notation $\alpha_3\alpha_4^{(k)}$, the parameters α_3 and α_4 describe the type and the number of the machines respectively, of the machine set environments for the stage k. $\alpha_3 \in \{\varnothing, P, Q, R, D\}$, where P indicates identical parallel machines, Q uniform parallel machines, R unrelated parallel machines, and D dedicated parallel machines –see Błażewich et al. (2007) and Pinedo (2008) for in-depth definitions–. The notation D refers to an environment where the jobs to be processed are known in advanced for each machine. A single machine is denoted as $\alpha_3 = \varnothing$. Several authors as Potts and Kovaliov (2000), denotes a flow shop with bath processing as $\tilde{F}$; there are no other specific symbols reported in literature for another shop types.

The parameter β –when used– lists the shop properties, which enumerate specific constraints and assumptions of the problem. The most common model properties associated with a HFS problem with lot processing are listed in Table 1.

Property	**Description**
Batch (Batch processing)	A machine is able to process up to b jobs continuously without any setup.
$b_k = m_k$	The mk batch processing machines at the stage k.
Fmls (Job families)	The jobs belong to different job families. Jobs from the same family are processed on a machine one after another without any setup in between.
lot	Lot processing.
prmp	The preemptions of jobs are allowed.
R (Removal time)	Machines become free only after the setup of the job has been removed.
split	A job (lot) splits into several parts (sublots) so that their operations may be overlapped. In case of shop with lot processing this parameter refers lot streaming.
s_{si} (Sequence-independent setup times)	The setup time of machine depend only on the job to process.
s_{fg} (Sequence-dependent family setup times)	The setup time of machine to process job (batch) belonged to the family g depends on the previous job (batch) family *f*.
w_j (Weight or importance of job *j*)	That is the priority factor of the jobs in the system.

Table. 1 Model properties associated with a HFS problem.

Finally, the γ field provides the criterion to be minimized. The most solicited Criterion to be minimized in a HFS scheduling problem, is the completion time. This particular objetive function takes place when the last job leaves the system, and is labeled as *makespan* or C_{max}. Another common objetive functions are, among others:

1. $F_{\max}$ as maximum flow time.
2. $L_{\max}$ as maximum lateness.
3. $T_{\max}$ as maximum tardiness.
4. $E_{\max}$ maximum earliness.

Continuing with the context to the task at hand, the triplet $FH2(RM^{(2)})\,|\,lot^{(1)}, p_j^{(1)} = p, batch^{(1)}, s_{fg}^{(1)}, split^{(2)}, s_{fi}^{(2)}\,|\,C_{\max}$ denotes the problem of scheduling jobs in a two-stage HFS with one batch machine on the first stage and unrelated parallel machines on the second. There exist, on the first stage, a few conditions that must present: lot processing, equal processing times of lots, batching of lots and sequence-dependent family setup time. Lot streaming into sublots and sequence independent setup times are presented on the second stage. The goal is to achieve a *makespan* minimum.

An example of another classification scheme with explicit number of batch processing machines is problem $\tilde{F}\,2\,|\,b_1 = 1;\ b_2 = 2\,|\,C_{\max}$, which denotes the *makespan* minimization in a two-machine flow shop, where the first machine is a classical machine, and the second is a batch machine that process up to two jobs simultaneously. Another example is the problem $1\,|\,s_{fg}\,|\,\Sigma C_j$, which denotes the total completion time minimization on a single (classical) machine, where there are job families and sequence-dependent family setup times (Potts & Kovaliov, 2000). It must be noted that due to the novelty of this class of models and scarcity of related work, there exist a lack of conventional notations for the scheduling problems with lot processing.

3. Batch processing

3.1 Group technology

The manufactured products on a plant frequently have technical similarities; therefore can be sorted out into groups according to their design or manufacturing attributes, such as part shape, size, surface texture, material type, raw material estate. The technical similarities of the products within a group permit reduce essentially the number of setups on a machine. Consequently, manufacturing time is decreased and machine usage time is improved.

This idea was adapted as Group Technology (GT). The GT is an approach solicited in manufacturing and engineering management, that in general aims to achieve the efficiency of high-volume production by exploiting similarities of different products and activities in their production/execution (Cheng et al., 2008). The concept of GT is based on the simplification and standardization process, and according to Burbidge (1975) appeared at the beginning of 20th century. Numerous manufacturing companies have taken advantage of GT to improve the productivity and competitiveness –see Wemmerlov & Hyer (1989), Tatikonda & Wemmerlov (1992), Hadjinicola & Kumar (1993) and Gunasekaran et al. (2001)–. The first publications on scheduling in GT environments are trace back to Petrov (1966).

The GT originally emerged as a single machine concept that was created to reduce setup times (Mitrofanov, 1966). Then it was extended to the HFS problem with setup times dependent on the job sequence (Li, 1997). Andrés et al. (2005) introduced the concept of *coefficient of similarity* between each of the products, whose original rol was as parameter, allowing products to be grouped through a heuristic method; and contrary to the basic concept of *exploiting similarities* (taken from the GT philosophy). The first approach allows design engineers to retrieve existing drawings to support the design standardization of new parts and make an accurate cost estimation. The second one produces improvements to the control process, reduction of the setup time and standardized process plans (Kusiak, 1987).

From the GT paradigm, two important concepts arise, product family and batch. The jobs are supposed to be partitioned into F families, $F \geq 1$. A *batch* is a set of jobs of the same family that can be processed jointly (Brucker, 2004). Batching occurs only if setup costs or times are not negligible and several jobs of the same family have to be produced. The processing time depends only on the family of the batch. When the processing is performed in batches of identical items (lots), the processed operations are executed simultaneously. Thus, the completion time of all the jobs in a batch is the finishing time of the last job in the batch. Once the processing a batch is started, it cannot be interrupted, nor can other jobs be

added into the batch. The motivation for batching jobs is to gain in efficiency: the processing jobs in a batch maybe cheaper or faster than to individual processing (Potts & Kovaliov, 2000). The term of *family* denotes initial job partitioning, while the term of batch is used to denote a part of the solution. The task to calculate the *batch size* is to decide how many units must be processed consecutively. In Liu & Chang (2000) is indicated that batch sizes must be optimized, because the processing in large batches may increase the machine utilization and reduce the total setup time. However, large batch processing increases the flow time. Therefore, a tradeoff between flow time and machine utilization by selecting batch size and scheduling comes into discussion. According to the GT, no family can be split, only a single batch can be formed for each family.

Many publications use the term batch to denote the initial job partitioning and they use different names like sub-batch, lot, sublot, etc., to denote a set of jobs of the same family processed consecutively on the same machine.

3.2 Batch models

Batch setup models are partitioned into *batch availability* and *job availability* models (Potts & Kovaliov, 2000). According to the batch availability model, all the jobs of the same batch become available for processing and leave the machine together. For example, this situation occurs if the jobs in a batch are placed on a pallet, and the pallet is only moved from the machine when all of these jobs are processed. An alternative assumption is job availability (usually known in the literature as item availability), in which a job becomes available immediately after its processing is completed and completion times are independent of other jobs in the batch.

The processing time of a batch is calculated according to Lushchakova & Strusevich (2010) as follows:

- In Serial batching, also known as *s-batch* or "sum-batch", the processing time of a batch is equal to the total processing times of its jobs.
- In Parallel batching, also known as *p-batch* or "max-batch", the processing time of a batch is equal to the largest processing time of its jobs.

When jobs sizes are considered, the case is usually called *a problem with bounded batches* if the total sizes of the jobs, contained in a batch, must not exceed the capacity of the batch, i.e. $b > n$. As far as each job may have a different size, the number of jobs in each batch may be different. On the other hand, if any number of jobs is allowed to be inserted in a batch, it is an called *unbounded batch*, respectively, $b \leq n$. In this case batches are not restricted in processing any number of jobs (Yazdani & Jolai, 2010).

When one batch is completed, the recourse has to be adjusted for the next batch. Time needed for the setup activities depends on the families of both adjacent batches. A batch is called *feasible* if it can be processed without any tool switches.

3.3 Batch processing machines

In literature, parallel batching scheduling is known as batch processor scheduling or scheduling of *batch processing machine* (BPM). A BPM processes several jobs simultaneously.

The different jobs can be batched together but the processing time of the batch is given by the longest processing time among all jobs in the batch. The BPMs are encountered in many different environments such as chemical processes performed in tanks or kilns and burn-in operations in semiconductor industry. Problems related with scheduling BPMs has been received much attention in scheduling literature in recent years –see Lee et al. (1992), Uzsoy (1994), Li (1997), Brucker et al. (1998), Lee & Uzsoy (1999), Damodaran & Srihari (2004), Mathirajan & Sivakumar (2006), Damodaran et al. (2007) and Manjeshwar et al. (2009)–. Results of these researches are relevant to the HFS, although refer to the single BPM, parallel BPMs, or flow shop BPMs. Uzsoy (1994) described an application for burn-in operations in semiconductor manufacturing. In papers of Damodaran et al. (2007), Liao & Huang (2008), Manjeshwar et al. (2009), it have been provided applications of BPMs in the chemical treatment stage in rim (for bike) manufacturing facilities and in chambers for the environmental stress screening in the printed circuit board assembly environment, respectivelly.

The two important decisions made on BPMs are:

- Grouping part families into batches, and
- Scheduling the batches to improve a performance measure.

The main classification of BPMs is related with *incompatible job families* vs. *compatible job families* (Perez et al., 2005). In the first model, only products belonging to the same family may be processed simultaneously. Uzsoy (1995), Kempf et al. (1998), Dobson and Nambimadon (2001) developed deterministic algorithms to schedule BPMs with incompatible job families. In the second model, it is assumed that products belonging to alternative families may be processed simultaneously –see Lee et al. (1992) where it is modeled burn-in oven as BPM–. Due to the complexity of BPM problems, scheduling research almost focus on single and parallel BPMs (Perez et al., 2005).

Quard and Kuhn (2007) investigate a s-batch scheduling problem for a HFS. Each job belongs to a specific product type. Setup costs are incurred when changing a machine of one product. On each stage, all jobs have the same process time. The objective is to minimize setup costs and the mean flow time. A target number of setups –i.e. parallel machines– for each product type and product type sequence are calculated for scheduling all production stages. The main focus of this paper is to derive the analogy of the scheduling problem to a two-dimensional packing problem and the development of a solution procedure that uses this analogy to solve the original HFS scheduling problem. Genetic algorithms are used as a framework to incorporate these ideas.

Xuan and Tang (2007) addressed the *s*-stage HFS problem of scheduling *n* jobs with s-batch processing at the last stage, and reduced to a two-stage HFS. The objective is to minimize a given criterion with respect to the completion time. When the jobs are grouped at the stage *s*, each batch *l* has a given size b_l –i.e. consists of b_l jobs–. The batch size can be different for all batches. All the jobs from the same batch must be processed on a machine at stage *s* consecutively while satisfying given precedence constraints among the jobs within this batch. Each job *j* has a weight and the waiting of job processing between two adjacent stages causes a penalty cost. A sequence-independent setup time is considered separate from the processing time before the first job of batch *l* starts processing. It could be anticipatory, meaning that the setup of the next batch can start as soon as a machine becomes free to

process the batch. Transportation times are also considered separate from the processing time. This paper establishes an integer programming model and proposes a batch decoupling based Lagrangian relaxation algorithm for this problem solution.

A two-stage HFS scheduling problem in a metal-working company is studied by Luo et al. (2009). The first stage consists of multiple parallel bounded-batch machines with job availability model, and the second stage has only one machine. The setup time is separated from job processing time and depends upon preceding job. A blocking environment exists between two stages with no intermediate buffer storage. Preventive maintenance and machine breakdown are presented. Two types of machine unavailability namely deterministic and stochastic case are identified in this problem. The former occurs on a stage two machine with the start time and the end time known in advance. The latter occurs on one of the parallel BPM in stage one and a real-time rescheduling will be triggered. Minimizing the makespan is considered as the objective to develop the optimal scheduling algorithm. A genetic algorithm is used to obtain a near- optimal solution. The computational results with actual data are favorable and superior over the results from existing manual schedules.

A two-stage HFS with several identical bounded p-batch processing machines is considered in the paper of Bellanger & Oulamara (2009). The problem is motivated by the scheduling of tire in the manufacturing industry. A compatibility relation is defined between each pair of tasks, so that an undirected compatibility graph is obtained which turns out to be an interval graph. The goal is to make batching and sequencing decisions in order to minimize the makespan. Since the problem is NP-hard, several heuristics are developed along with their worst cases analysis. The case in which tasks have the same processing time on the first stage also is considered, and a polynomial time approximation scheme (PTAS) algorithm is presented.

The review described in the above lines, shows that scientific literature in this addressed problem has small number of contributions. One of the main factors that could lead to this condition is that the BPM entails a high computational complexity. Then only one or two stages shops are investigated.

3.4 Setup time treatment

The structure of the breakdown time when a job belongs to a machine includes three phase as follows (Cheng et al., 2000):

1. Sequence independent/dependent setup time that is independent/dependent on the job to be processed.
2. Processing time of the job.
3. Removal time that is independent/dependent on the job that had just been processed.

The setup time is defined as the time required to shifting from one job to another on a given machine. There are separable and non-separable from the process operation. The *non-separable* setup times are either included in the processing times or are negligible, and hence are ignored. There exist some situations in which the non-separable setup and removal operations must be modeled and closely coordinated. Such situations are common in automatic production systems which involve intermediate material handling devices, like automatic guided vehicles and robots, loading and unloading (Crama, 1997; Kim et al., 1997). The *separable* setup times are not part of processing operation.

When separable setup/removal times are not negligible in the scheduling problem, they should be explicitly treated. In many real-life industrial problems such as surface mount technology or printed circuit board manufacturing, job setup is not part of processing time and the required time is sequence-dependent. Cheng et al. (2000) presents an interesting review of flow shop scheduling research with setup times.

The separable setup times could be *anticipatory* (*detached*) or *non-anticipatory* (*attached*). A setup is anticipatory if it can be started before the corresponding job or batch becomes available on the machine. In such a situation, the idle time of a machine can be used to complete the setup of a job on a specific machine. Otherwise, a setup is non-anticipatory, and the setup operations start only when the job arrives at a machine as long as the setup is attached to the job. Furthermore, setup time of a job at a specific machine could be dependent on the job immediately preceding that job or be independent of it.

The setup may reflect the need to change a tool or to clean the machine. As in a family scheduling model, the jobs are partitioned into families according to their similarity, so that no setup is required for a job if it belongs to the same family of the previously processed job. However, a setup time is required at the start of the schedule and on each occasion when the machine switches from processing jobs in one family to jobs in another family. In such model, a batch is a maximal set of jobs that are scheduled contiguously on a machine and share a setup.

The next setup analysis is proposed in the paper of Potts & Kovalyov (2000). Let $\{1, \ldots, n\}$ denote the set of jobs to be processed and p_j is the processing time of job j, $j = 1, \ldots, n$. Other parameters include a release date r_j, a deadline d_j , a due date d_j, and a weight w_j. The jobs are partitioned into F families. Let n_j denote the number of jobs in family f, $f = 1, ..., F$. No setup is required between jobs of the same family. However, the *family setup time* on machine i when a job of family g is immediately preceded by a job of a different family f is s_{ifg}, or s_{i0g} if there is no preceding job. If, for each g, occurs that $s_{ifg} = s_{i0g} = s_{ig}$ for all $f \neq g$, then the setup times on machine i are *sequence independent*; otherwise, they are *sequence dependent*. If, for each machine i, $s_{ifg} = s_{fg}$ for all families f and g including the case $f = 0$, then the setup times are *machine independent*; otherwise, they are *machine dependent*. For the case of a single machine, setup times are, by definition, machine independent. Further, the reasonable assumption is that the *triangle inequality* holds for each machine i, which means that $s_{ifh} \leq s_{ifg} + s_{igh}$, for all distinct families f, g and h, including the case $f = 0$. Unless stated otherwise, the setups are assumed to be *anticipatory*, which means that a setup on a machine does not require the presence of any job. When there are release dates and for shop problems, sometimes the setups allow to be *non-anticipatory*, which means that the setup preceding the processing of some batch cannot start on the current machine before all jobs of this batch are released and have completed their processing on any previous machine.

The minor and mayor setups implementation in a two-stage HFS problem with part family and batch production is proposed in Li (1997). Sequence independent bath setups are considered in Quard & Kuhn (2007), Xuan & Tang (2007). Sequence dependent bath setups are included in the problem described by Luo et al. (2009).

3.5 Burn-in operation

The concept of batch processing is arisen from burn-in operation in semiconductor manufacturing industries which represent today one of the most complex industrial

environments. In semiconductor manufacturing, there are parallel machines, different types of processes like batch processes and single wafer processes, sequence-dependent setup times, prescribed customer due dates for the lots, very expensive equipment, reentrant process flows, etc. In such a changeable scenario, maintaining a competitive advantage and remaining profitable in operational terms requires minimization of cycle time, work-in-process (WIP) inventory and the maximization of throughput.

As a part of the complex production line that exists in a semiconductor manufacturing facility, operations involved in BPM are considered to be a bottleneck. This is because the processing times of the lots on the BPM are usually very long compared to other processes, and batching decisions may affect the performance of the entire semiconductor manufacturing process. Semiconductor manufacturing involves numerous batch-processing operations like oxidation, diffusion, deposition, etching, e-beam writing and heat treatment of wafer fabrication, baking of wafer probing, and burn-in operation of device testing.

The *semiconductor burn-in scheduling problem* was first introduced by Lee et al. (2006) and then studied by Mathirajan et al. (2010). The purpose of burn-in operation is to test the integrated circuit (IC) chips. Due to various processes employed in the manufacturing process, some chips may be "fragile" and may fail only after a short period of time. It is essential that these devices are identified and scrapped as "infant mortality". The process of identifying and scrapping these "fragile" devices is known as the *burn-in operation*. It involves subjecting the chips, placed in an oven, to electrical and thermal stress to force the failure of weak or fragile devices.

IC chips arrive at the burn-in area in lots consisting of a number of IC chips of the same product type. Each lot is referred as a job. In a burn-in operation, IC chips of each job are loaded onto *boards*; each job has different lot sizes so that those job sizes (number of demanded boards) are not identical. The boards are often product-specific, and a job cannot be processed without the necessary boards. Once IC chips have been loaded onto the boards, the boards are placed into an oven. Typically, the oven capacity is larger than the job size, so the number of boards in an oven can hold defines the oven capacity, and the size of a job is defined by the number of boards it requires. Each IC chip has a pre-specified minimum burn-in time, which may depend on its type and/or the customer's requirements. Since IC chips may stay in the oven for a period longer than their minimum required burn-in time, it is possible to place different products (jobs) in the oven simultaneously.

The processing time of each batch equals the longest minimum-exposure time among all the products (jobs) in the batch. Effective burn-in operation scheduling is a key issue because it causes frequently a bottleneck due to long processing times relative to other testing operations –e.g. days as opposed to hours– and because it occurs at the end of the manufacturing process and thus has a strong influence on ship dates (Azizoglu & Webster, 2000).

3.6 Cell architectures

The arisen of FMS has triggered many researches dedicated to distinct aspects of their design and functionality. New cell architectures are proposed and new types of management problems are analyzed. Most of the recourse models are dedicated to minimize production costs, that is, the productivity maximization, through optimally allocation and

synchronization machinery and parts. Some aspects of those architectures affect the performance of FMS. One of such aspects is the presence of setups.

In several flexible cell architectures each part is mounted on a fixed position in the cell and does not move until the processing machine –i.e. a robot– has completed all the required operations. A FMS recourse (machining center, robot, numerical control center, etc.) needs a certain time to switch the operating mode before the recourse can start (Co et al., 1990; Agnetis et al., 1993; Samaddar et al., 1999; Agnetis et al., 2003). There are generally two different types of setups. One setup occurs when a finished part is removed and replaced with a new part (*part replacement*). Another setup occurs when the machine switches from one operation type to another (*tool switch*). The part replacement setup usually makes more time than the tool switch (Agnetis et al., 2003).

Part replacement may be executed in two different ways named FMS *management policies* (Stecke & Kim, 1988):

- *Batching replacement*. All parts (up to a total k) currently being processed must be completed before new parts are loaded. A setup occurs whenever a new set of parts is loaded (*batch setup*), and another setup occurs at each tool switch (*tool setup*). Parts cannot be removed without stopping the cell operation.
- *Flexible replacement*. Part replacement takes places when it is completed. These setups only occur at tool switches.

Two subproblems must be simultaneously solved (Crama, 1997; Agnetis et al., 2003):

- Forming batches of at most k parts each one.
- Scheduling of the tools (and therefore the robot moves) required by program execution in each batch.

The *tool schedule* is the sequence of tools (possibly repeated) loaded by the machine when processing a given batch. The total time required to process a batch is given by the total duration of the operations, plus the *total tool switching time*, which must be minimized. The length of a tool schedule is the length of the shortest tool schedule for that batch. Hence, for each batch, the scheduling subproblem consists of finding the batch length and the corresponding tool schedule.

Agnetis et al. (2003) provide two illustrative examples of a tool schedule problem for a robot, where any tool schedule is feasible, and provide that it is NP-hard.

Example 1. There are three tools a, b, c, and a batch B consisting of three parts of program P_1 = *abac*, P_2 = *bacb*, and P_3 = *abcb*. For instance, the tool schedule *abcabc* allows to complete P_1 and P_3 but not P_2. Scheduling the tools in the order *abcbacb*, all the part programs can be completed, but this requires six tool switches. On the other hand, the tool schedule *abacb* allows to complete the three part programs using only four tool switches. The length of B is therefore five.

Since the robot performs only one operation at a time, and the total duration of all operations is given, the objective only depends on the amount of time spent in setup activities (both batch and tool setups). The total batch setup time is equal to T_s (b - 1), where b is the number of batches. The total setup time is obtained summing the tool setup times of all batches.

Example 2. Besides batch B of the previous example, suppose there is also another batch B' formed by parts $P_4 = bcba$, $P_5 = caba$, and $P_6 = bcab$. The length of B' is also five, obtained for the tool schedule *bcaba*. When batches B and B' are performed, the overall time spent in setup activities is given as $T_s + 8\ t_s$, where T_s is loading new batch setup time, and t_s is any tool switch setup time.

In some situations, the more relevant performance criterion is the number of batches (*switching instants*). This is the case, when the setup time of operations is proportional to the number of tool interchanges, or when the tool transportation system is congested. The distinction between the number of tool switches and the number of switching instants is considered in Tang & Denardo (1988a), Tang & Denardo (1988b) and Sodhi et al. (1994). The more general formulation of the *tool switching problem* is given by Crama (1997) as follows: Determine a part input sequence and an associated sequence of tool loadings such that all the tools required by the j-th part are present in the j-th tool loading and the total number of tool switches is minimized.

Crama et al. (1994) proved that the tool switching problem is NP-hard for any fixed $C \geq 2$, where the number C is the *capacity* of the tool magazine. They also observed that deciding whether there exists a job sequence requiring exactly M tool setups is NP-hard, where M is the total number of tools needed to process all the parts. This latter result conduced to the *gate matrix permutation problem* discussed in the paper of Mohring (1990). The gate matrix permutation problem is to assess the minimum value of the tool magazine capacity such that no tool needs to be set up twice. Since this problem is NP-hard, the tool switching problem is NP-hard too. When all setup times are equal –i.e. when the objective is only to minimize the total number of switches– then the integer program can be solved by a greedy algorithm which turns out to be equivalent to the so-called *keep tool needed soonest policy* (KTNS) –see Tang & Denardo (1988a)–.

4. Lot streaming

4.1 Job splitting

In most multi-stage scheduling studies associated with processing products, a *production batch* (lot) is treated as a single entity called *job* which consists of only one part. As a result, the schedule cannot be improved any more, even if there may be plenty of idle times at machines; that is, partial transfer of completed items in a job between machines is assumed to be impossible. Since the production lots are often large, items already processed on a machine need to wait a long time in the output buffer of this machine whereas the downstream machine may be idle. It can lead to large WIP inventories between the machines and make longer the makespan. When the above assumption is relaxed in the corresponding scheduling problem –i.e. when it is supposed that a job can be split– it may be possible to improve the quality of the resulting schedule. Hence, the production leading times, WIP inventory, interim storage and space requirements, and material handling system capacity requirements can be decreased (Truscott, 1986). The solutions can be implemented in a lesser time and at a lower cost than reorganization. Moreover, when the processing requirement of a job is considered as a total demand of a product in production planning, jobs can be split arbitrarily into continuous sublots and processed independently on m machines to finish the processing all demands as soon as possible (Xing & Zhang, 2000).

As is pointed out by Potts and Van Wassenhove (1992), there are two main advantages of splitting jobs into sublots. Firs, splitting jobs may improve customer service; each sublot can be delivered to the customer immediately upon completion, without waiting for the remaining sublots of the same job. The other motivation for splitting jobs in multi-stage production systems is to enable various operations of the same job to be overlapped by allowing processing of downstream operations to begin immediately for any sublot which has been processed at the current stage.

Baker and Pyke (1990) consider two cases of job splitting:

- Preemption, i.e., Interruption of the production run for a more urgent job;
- Lot streaming, where overlapping operations are permitted.

The concept and practice of lot streaming are not new. The term of lot streaming was first introduced by Reiter (1966). Graves & Kostreva (1986) gave the notion of *overlapping operations* in material requirements planning (MRP) systems. The use of transfer batches (or sublots) is a key element of synchronous manufacturing (Umble & Srikanth, 1990).

So, *lot streaming* is the process of splitting an entire production job (*process batch*) into sublots (*transfer batches*) and scheduling those sublots in an overlapping fashion, in order to accelerate the progress of an order in production. A job is defined here as a production order (*lot*) composed of many identical items (Potts & Baker, 1989), (Baker & Jia, 1993), (Çetinkaya & Duman, 2010). In many practical situations, splitting a lot is both possible and desirable. When a job is split into a number of sublots, a sublot can be processed on a machine even if the other sublots still have not been processed on the upstream machines. The *lot streaming problem* is to decide the optimal number of sublots for each job, the optimal size of each sublot and the optimal sequence for processing the sublots so that the production lead time is minimized. It combines lot sizing and scheduling decisions that were traditionally treated separately (Baker & Jia, 1993; Zhang et al., 2005).

Different sublots of the same job should be processed simultaneously at different stages. As a result of operation overlapping, the production is remarkably accelerated and the idle time on successive machines is reduced. In general, the makespan will be minimized if there is just one item in each sublot (Vickson & Alfredsson, 1992). Nevertheless, there may be practical considerations that make it undesirable to have a large number of unit-sized sublots. It may be possible to attain the minimum makespan with fewer sublots, or there may be difficult in tracking a large number or small sublots. A lot splitting problem finds a compromise between sizes of batch process and sublots when setups are long and difficult. An example of lot streaming benefits for three-machine flow shop, a single job, with the job processing times of 6, 3 and 6 time units is shown on Fig. 1 borrowed from Pan et al. (2010). If the job is not split into sublots, the job completion time is 15 time units (Fig. 1a). When the job is split into three sublots and no-idling production interruption time is allowed between any two adjacent sublots, the job completion time is reduced to 11 time units (Fig. 1b), whereas for the idling case, the job completion time is further reduced to nine time units (Fig. 1c). Obviously, the completion time of the job under the idling case is shorter than the one under the no-idling case with the same sublot type. However, there are also many practical applications for the lot-streaming flow shop scheduling under no-idling case.

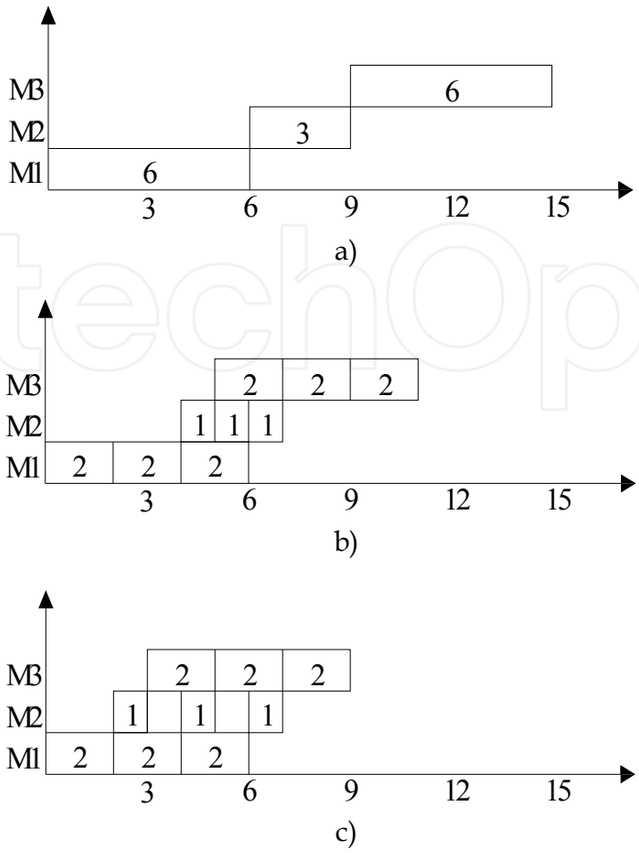

Fig. 1. An example of the lot-streaming flow shop scheduling: a) schedule without sublots; b) schedule with sublots under no-idling case; c) schedule with sublots under idling case.

In the past two decades, with the increasing interest in just-in-time (JIT) and optimized production technology (OPT) philosophies in manufacturing systems, the application of lot streaming idea in scheduling problems has received considerable attention. The JIT approach views each item in the lot as a single unit or job since these results look for a minimum makespan. Through the extensive use of JIT system in manufacturing, the performance measure related to both earliness and tardiness penalties has raised significant attention in lot streaming literature (Kulonda, 1984; Glass et al., 1994; Pan et al., 2010).

One more positive aspect of lot streaming is mentioned in Jeong et al. (1997):

"By assuming that a batch can be split, we can handle the problems that occur due to dynamic nature of shop floor more elegantly. Most past solutions to scheduling problems assume that a manufacturing shop is operating steady and peacefully. But a real factory is full of unexpectedness and dynamics. The causes of such dynamism are very diverse, e.g. human errors, rush orders, requests from customers to shorten the delivery dates, and hardware related events - tool breakage, machine failure, etc. These events cannot be expected beforehand, and therefore, it is very hard to strictly observe the original schedule. As suggested earlier, we can get much better solution considering alternative schedules which allow the splitting of current batches in smaller batches."

Sublot sizes can be (Trietsch & Baker, 1993):

- *Equal* - all sublots of a given lot are of equal size,
- *Consistent* - sublots sizes vary within a lot but are the same for all machines,
- *Variable* - sublot sizes can change from machine to machine.

There are *discrete* and *continuous* versions of lot splitting. In *discrete version* the sublot sizes are integers that correspond to discrete numbers of units in each sublot. Typically, these problems can be formulated as an integer linear program. Anticipating that this may be a difficult problem to solve, lot splitting becomes a *continuous version* of the problem in which the integer restrictions are relaxed. Such a routing may be acceptable if lot size is large and the number of sublots is small. The optimal makespan in the continuous version also serves as a lower bound on the optimal makespan in the discrete version, and the makespan produced by rounding the continuous solution serves as an upper bound (Trietsch & Backer, 1993).

Lot streaming is very common among modern manufacturing systems, furthermore, also introduces an additional complexity in the problems. Each sublot can be viewed as an individual job and the problem size drastically increases with the total number of sublots. If it is assumed that a job consists of a batch as in many real manufacturing environments, then we can obtain an improved schedule. However then, the size of the scheduling problem would become too large to be solved in practical time limit (Jeong et al., 1997). The review on lot streaming problems can be found in several papers –see Potts & Baker (1989), Potts & Wassenhove (1992), Trietsch & Baker (1993) and Chang & Chiu (2005)–.

The publications on lot streaming problems can easily be categorized into two main streams:

- dealing with the determination of the optimal sublot sizes for a single job (also called sublot sizing);
- addressing the sublot sizing and job sequencing decisions simultaneously for the multiple-job case, given various job and shop characteristics.

4.2 HFS lot streaming problem state of art

Lot streaming is very useful as many practical production systems can be considered as HFS, however has received very limited research attention. Some of them will be reviewed in the following.

Tsubone et al. (1996) studied the lot streaming problem in a two-stage HFS with one machine in the first stage and several process lines in the second stage. They examined the impact of lot size, sequencing rules and scheduling scenarios on production makespan, capacity utilization and WIP inventory, using simulation. Zhang et al. (2003) analyzed the integer version of the m-1 HFS lot streaming problem. They solved special cases of the equal-sublot version of the problem, in which one of the stages was obviously a bottleneck. The general problem is formulated as a mixed integer linear programming (MILP) model and solved using two heuristics. Both heuristics enumerated the number of sublots, and for each given number of sublots, allocated the sublots as evenly as possible to stage-1 machines. The sublot sizes were then determined by making them as equal as possible in one heuristic, and by using a smaller MILP model in the other. In this contribution, the

continuous version of the problem is studied and efficient optimal solutions to both the problem with given number of sublots and all the cases of the equal-sublot problem is provided. The continuous version does not restrict the sublot sizes to be integers. This is practical in situations where the product is of continuous type –e.g. those in process industries– and where an order consists of a large quantity of small items such as DVDs.

Manufacturing systems considered in studies by Oğuzc et al. (2004), Oğuzc & Frinket (2005) and Ying & Lin (2006) are referred as flow shop with multiprocessors (FSMP). It is a special case of HFS where the parallel machines are assumed to be identical. The overlapping of operations in parallel machines is not considered across stages. Thus, such studies may not be classified as lot streaming research as the very definition of lot streaming is to allow the overlapping of operations across stages.

Zhang et al. (2005) studied the multi-job lot streaming problem in two-stage HFS with m identical machines at the first stage and a single machine at the second stage and the objective is to minimize the mean completion time of the jobs. All the n jobs are available at time zero. The job sizes are different. A job can be split into sublots that will be treated as separate entities in production. Each sublot requires processing on any one of the machines at the first stage and then on the machine at the second stage. It is assumed that:

There is a given smallest allowable size for the sublots of each job, from which the maximum number of sublots allowed for the job that can be derived.

1. Each machine can process at most one sublot at a time.
2. Each sublot can be processed on at most one machine at a time.
3. The size of each sublot is kept consistent at the two stages.
4. A job can be continuously divisible. As the number of units in ajob is very large, the error between the objective value of the rounded solution and that of the continuous solution may be negligible.
5. The jobs are processed one after another, i.e. the sublots of different jobs are not mixed up in the sequence of processing on any machine.
6. A setup is needed before the processing of each sublot of a job on a machine. The setup times for sublots of the same job at the same stage are equal.
7. The unit processing times and the setup times are known constants.

The problem is then to decide the number and the sizes of sublots for each job and to schedule these sublots to minimize the mean completion time of all jobs. The completion time of a job is defined as the completion time of the last sublot of the job on the second-stage machine.

To solve this NP-hard problem, two heuristics were developed, both using the strategy of first sequencing the jobs and then lot streaming each job. The two heuristics differ in the way of sequencing the jobs. The first heuristic treats each job as a whole lot. The second heuristic considers the system as a pure flow shop with the stage-1 machines aggregated. It uses a profile of each job from the single-job lot streaming result as the time requirements in the artificial pure flow shop. When solving the lot streaming problem of each job in the sequence, both heuristics assign balanced numbers of sublots to the machines at the first stage and decide the sublot sizes using a LP model. A MILP model for the problem was also formulated and used to obtain a lower bound through relaxation. The lower bound was

used jointly with two other lower bounds obtained from the direct analysis of the problem structure. Extensive experiments showed that the aggregated machine heuristic performs much better.

Liu (2008) studied the single-job lot streaming problem in a two-stage HFS that has m identical machines at the first stage and one machine at the second stage, called m-1 HFS, the same machine environment that in Zhang et al. (2005). The job is splitted into sublots. It is not necessary for the sizes of the sublots to be equal or to be integers, but they must not be smaller than a lower bound, $x_0 \geq 0$. A sublot will be treated as an independent entity during the production in the system. Each sublot requires processing on anyone of the machines at the first stage and then on the machine at the second stage. The processing times of a sublot at the two stages are proportional to its size. The unit processing times on the stage-1 and stage-2 machines are $p^{(1)}$ and $p^{(2)}$, respectively. Each machine can process at most one sublot at a time. Each sublot can be processed on at most one machine at any time.

Before the processing of each sublot on a machine, a setup time is required for loading the sublot onto the machine. The setup times on the machines of stages 1 and 2 are $s^{(1)}$ and $s^{(2)}$, respectively, which are independent of the processing sequence and the sizes of the sublots. The problem is to determine the number and the sizes of the sublots and the schedule of processing them on the machines to minimize the makespan (the completion time of the entire job).

For the problem with a fixed number of sublots, Liu (2008) proved that it is optimal to use a rotation method for allocating and sequencing the sublots on the machines previously used by Zhang et al. (2003): First, number the sublots is 1, 2, . . ., l. Then, allocate and sequence these sublots on the stage-1 machines in ''rotation'', i.e. assign sublot 1 to machine 1, sublot 2 to machine 2, ..., sublot m to machine m, sublot m + 1 to machine 1, and so on. Finally, sequence the sublots on the stage-2 machine in the order of their numbers.

Hereafter it is referred to the first sublots on the first-stage machines as the first batch of sublots, to the second sublots on these machines as the second batch, and so on. The schedule given by the rotation method is characterized by the following features:

1. The numbers of sublots processed on the stage-1 machines are *balanced* –i.e. each stage-1 machine processes at least $\lfloor l/m \rfloor$ sublots and at most $\lceil l/m \rceil$ sublots–;
2. The sublots are processed on the stage-2 machine in the order of their batch numbers;
3. For the sublots in the same batch, the processing on the stage-2 machine is in the order of their first-stage machine numbers.

When the sublots are allocated using the rotation method, the only remaining decision is to determine the sizes of the sublots. As all the discrete decisions are fixed, the problem of determining the sublot sizes can be formulated as a LP model. Then the problem with equal sublot sizes is considered and an efficient solution to determining the optimal number of sublots is developed. Finally, optimal and heuristic solution methods for the general problem are proposed and the worst-case performance of the equal-sublot solution is analyzed. Computational experiments on a wide range of problem settings, Liu (2008) and Zhang et al. (2003) showed that the heuristic solutions are very close to optimal.

The work of Defersha (2011) was motivated by the gap perceived in research efforts in pure flow shop lot streaming and HFS scheduling. However, the issue of lot streaming in this

type of shops has not gained as much attention as for pure flow shop scheduling. Defersha (2011) refers to the work of Zhang et al. (2005) as the only paper that addressed lot streaming in HFS. However, this work is for a very special case where there are parallel machines only in the first stage and the number of stages is limited to two. Lot streaming in a more general HFS with parallel machines on any stage and the number of stages is not limited to two has been studied implicitly and partially.

This chapter is aimed to research in bridging the gap between the efforts in pure flow shop lot streaming and HFS scheduling by presenting a comprehensive mathematical model for lot streaming addressing to the work of Ruiz et al. (2008) which considers the overlapping operations in successive stage through the concept of negative time-lag. This mathematical model incorporates several other practical issues such as unrelated parallel machines, the possibility of certain jobs to skip certain stages, sequence-dependent setup times, anticipatory or non-anticipatory nature of setups, release dates for machines, and machine eligibility. The sublots are to be processed in the order of the stages and sublots of certain products may skip some stages. At a given stage, a sublot of a job can be assigned to one of the parallel machines eligible to process that particular job. For each job there is a sequence dependent setup time on each eligible machine and this setup may be anticipatory or non-anticipatory on different stages. Each machine can process at most one sublot at a time. Sublots of different products can be interleaved. The problem is to determine the size of each sublot of each job, the assignment and processing sequence of these sublots on each machine in each stage. The objective is to minimize the completion time of the last sublot to be processed in the system. The proposed model was solved to optimality for small problem size. The numerical example shown in this referenced paper demonstrated that lot streaming can result in larger makespan reduction in HFS where there is a limited research than in pure flow shop where research is abundant.

5. A problem of demand splitting in a two-stage HFS with lot processing

In this section, an exercise of application of the concepts described trough the section is shown. It consists about the analysis of a real problem settled in electronic components manufacturing; where theoretical formulations as long as modeling results are displayed.

5.1 Production model

The production model is set in reed switch manufacturing. Reed switches are used in electric sensors and relays. The components of a reed switch are two metallic contact blades (reeds) positioned in a hermetically sealed glass tube with a gap between them. The production planning on the plant is realized per lots for a planning horizon, according to the consumer's demands. A demand includes an accepted range of switch operate value named part number, a form of blades, an amount of pieces, a delivery date and needs several lots to be completed.

The technical route of the switch manufacturing is composed of several successive operations divided on a natural manner in two parts with an external operation of one day duration. The investigation is focused on the second part where the presented problem occurs: the classification of pieces and the blade form. The classification operation is realized per lots on one of the group of parallel identical machines, so as one lot is assigned a one of machines. The

classification machine measures the value of each piece and deposits it into one of 25 machine repositories according to the value range. Four tube glass types are used in the plant. One lot has a fixed amount of the pieces with the same tube glass type. When a machine starts processing, does not interrupt up to finishing lot. One lot does not divided between machines. There is a list of part numbers. A part number indicates several numbers of repositories so as the same repository can be included in different part numbers. After the classification, a lot is distributed between repositories and then the product is treated in pieces. The content of repositories is assigned on one of four blade form lines taking in account part number and the demand amounts. The rest of pieces not reclaimed for demands forms WIP inventory. The investigated resource model is presented on Figure 2.

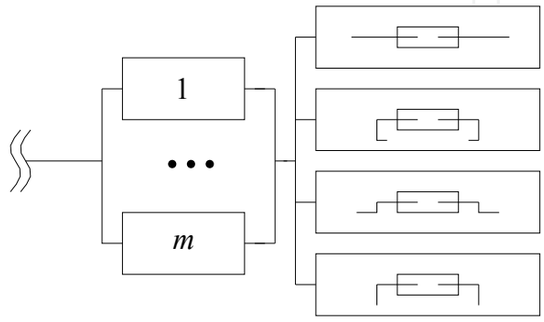

Fig. 2. The m identical classification machines and four blade form lines.

The information complexity causes numerous problems in the production planning and scheduling of the plant because they are realized empirically, based on previous experience, and therefore suffer from stochastic results of the lot processing.

5.2 Problem statement

According to the previous experience on the plant, the tube glass type has influence on the distribution of pieces between machine repositories. In (Romero & Burtseva, 2010) is shown that the distribution of pieces belonged to one lot has central tendencies specific for each glass type, and those empirical distributions trend to be near-normal (Fig. 3). A deterministic approach is used here to analyze and resolve the problem, i.e., all distributions are supposed to be known and fixed for each glass type, so as the content of a repository can be anticipatory assigned to a demand.

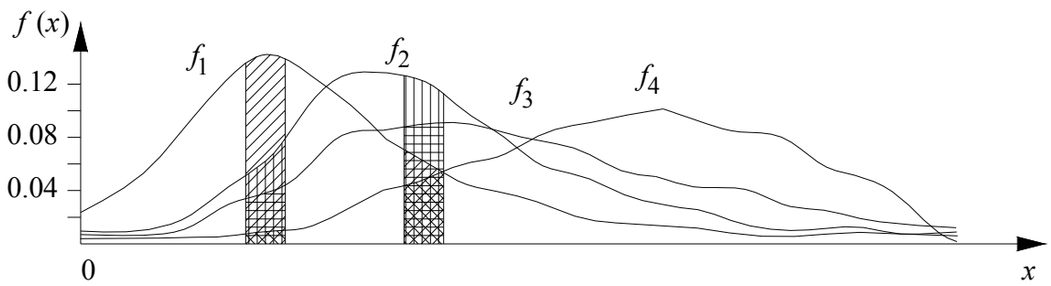

Fig. 3. The probability density functions $f(x)$ of the distributions of the pieces between repositories for different lot glass types.

The follow reasons cause the problem complexity. The quantity of lots for the completion of demands depends essentially on the selection of the glass type of the pieces in the lot in consequence of differences of distributions. Moreover, as the different part numbers often include the same repositories, the allocation of demands on repositories is not evident, e.g., demands 1 and 2 need repositories 1 and 2, and demand 3 needs repositories 2 and 3 (Fig. 4). As a result, the same quantity of pieces for the required part numbers can be obtained from different quantity of lots, and consecutively, the completion time will be different.

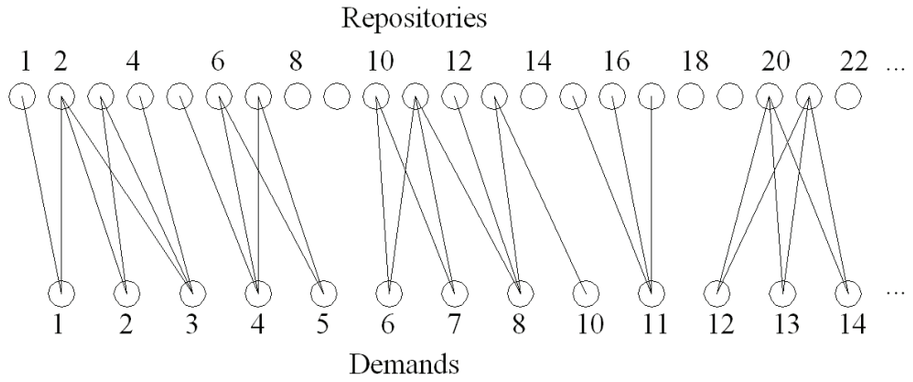

Fig. 4. Bipartited graph of the relationship between repositories and demands (an example).

So, the considered problem is as follows. The n demands must be carried out in two-stage HFS with m_1 parallel identical machines on the first stage and m_2 dedicated machines on the second. The first stage is realized per lots of size U, and the second per batches of sublots. There are G lot types. After first stage, a lot l is distributed into R sublots of size k_{gr} according to the row g of the matrix $K[g,r]$, $g = 1,\ldots,G$, $r = 1,\ldots, R$, i.e., the lot splitting is realized on a natural manner. The demands must be allocated per parts on lot repositories employing the binary chain j of R bits, whose r-element is equal 1 if the repository r is allowed to using for fulfillment of demand j, and is 0 otherwise. In result of the allocation, the sublots s_{lrj} are formed and assigned without mixing to one of m_2 dedicated machines of the second stage. There are sequence independent setup times before the first stage, needed for the allocation of the lot on machine, and between the first and the second stages for the sublots rebatching. The criterion is the makespan minimum.

Using the three-field notation $\alpha|\beta|\gamma$, the considered problem can be denoted as:

$$FH2, PM^{(1)}, DM^{(2)} \,|\, \text{lot}^{(1)}, \text{split}^{(1)}, p_l^{(1)} = p, \text{job constraints}^{(1)}, \text{s}_{\text{si}}^{(2)}, \text{batch}^{(2)} \,|\, C_{\max}. \qquad (1)$$

Given this situation, the optimization problem addressed in (1) is summarized as follows: The shop model represents a two-stage HFS with a set of parallel identical machines on the first stage and a set of dedicated machines on the second stage. On the first stage, there are lot processing and splitting the lots in form of distribution of pieces into the machine repositories. The lot processing time is constant for any lot. The job constraints on the first stage are mentioned as the restrictions of a job allocation on certain repositories. There is batching of sublots on the second stage. The criterion is the makespan minimum, i.e., the time when the last job is finished.

5.3 Notations

j Demand index, $j = 1,\ldots, N$.
J Demand list, $J = \{j_1, j_2,\ldots, j_N\}$
r Machine repository index, $r = 1,\ldots, R$.
g Lot type, $g = 1,\ldots,G$.
l Lot index, $l = 1,\ldots, L$.
M_1 Set of machines at the first stage, $M_1 = \{1,\ldots, m_1\}$
M_2 Set of machines at the second stage, $M_2 = \{1,\ldots, m_2\}$
V_j Set of repositories associated with job j.
D_j Size of demand j.
U Lot size.
$O = [o_{jr}]$ Utilization of repository r in the job j(part number, N chains of R bits), $O_j = \{o_{j1},\ldots, o_{jR}\}$.
$K = [k_{gr}]$ Matrix of distributions of pieces for a lot of type g into machine repositories r.
L Quantity of lots.
A Resulted lot list, $A = \{g_1, g_2,\ldots, g_l,\ldots, g_L\}$, $g_l \in \{1,\ldots,G\}$.
d_{lrj} Size of sublot s_{lrj}.

5.4 Problem assumptions

- The lots of all types are available at time zero. The size of any lot is U pieces and its processing time is fixed to be $p^{(1)}$. A setup time needed for lot allocating on a machine is included in the lot processing time. The lot selected to processing is identified completely by its glass type g. One lot does not divide between machines. When a machine begins lot processing does not stop until finish.
- The lot l can be assigned to any of m_1 parallel identical machines on the first stage, and next lot is assigned to a machine immediately after finishing of the previous lot. The occupied machines start and finish together. Each machine of the first stage can process at most one lot at time.
- After first stage, the lot of type g is distributed between R machine repositories according to the row g of the matrix $K = [k_{gr}]$; that is, one lot splits into R machine repositories.
- The sublots of size d_{lrj}, $l = 1, 2,\ldots, L$, $r = 1,\ldots, R$, $j = 1,\ldots, N$, are formed using repository contents. A sublot can be formed from only one repository. The content of one repository can be used for one o more sublots.
- The pieces that were not assigned to any job are not used and form the WIP. The WIP content is considered to be used on future demands.
- The sublots belonged to the same job j are joined in batches. The sublots belonged to the different jobs are not mixed. The batches are formed from sublots whose processing is assigned to the same dedicated machine of the second stage.
- There is a non-anticipatory sequence independent setup time s_{in} to form the sublots from the repository contents and the batches of sublots belonged to the same job to be processed on the second stage.
- Each machine of the second stage processes at most one batch at a time. The idle time between batches are permitted on a machine. When a dedicate machine begins processing of a batch does not stop until finish.

5.5 Model of lot batching

To schedule the demand processing, their allocation onto lots must be realized, then, the lots have to be scheduled on the first stage. A demand j can be completed only from certain repositories; formally, it is expressed as follows: The row j of binary matrix $O = [o_{jr}]$ corresponds to the set V_j of repositories associated with demand j. A matrix element $o_{jr} = 1$ if the repository r can be used to complete the demand j. In other cases, $o_{jr} = 0$. Matrixes O and K are employed to find the quantity of pieces of the next demand from the list which can be allocated on the next lot. In result of the allocation, the demand splitting into sublots d_{lrj} and the primary lot batching are realized, then the lot list $A = \{g_1, g_2, \ldots, g_l, \ldots, g_L\}$ of L lots is formed whose elements are the lot types g_l. So, the problem of lots number minimizing is:

$$L = \sum_{g=1}^{G} n_g \rightarrow \min \tag{2}$$

Subject to

$$\sum_{g=1}^{G} \sum_{l_g=1}^{n_g} \sum_{j=1}^{N} d_{l_g rj} \cdot o_{rj} \le \sum_{g=1}^{G} n_g k_{gr} \ , \ r = 1, \ldots, R, \tag{3}$$

$$\sum_{r=1}^{R} \sum_{j=1}^{N} d_{l_g rj} \le U \ , \ \forall g, l_g, \tag{4}$$

$$o_{rj} = \begin{cases} 1, \text{ if } r \in V_j \\ 0, \text{ otherwise} \end{cases} \quad \forall \ r, j, \tag{5}$$

$$\sum_{g=1}^{G} \sum_{l_g=1}^{n_g} \sum_{r=1}^{R} d_{l_g rj} \cdot o_{rj} = D_j \ , \quad j = 1, \ldots, N, \tag{6}$$

$$k_{gr}, d_{l_g rj} \in Z^0 \ , \ \forall g, l_g, r, j. \tag{7}$$

The constraint (3) describes the relationship between assigned job part sizes and total capacity of repository r, where the total of pieces assigned to a lot l_g must not exceeded its capacity U (4). The binary value o_{rj} in (5) is used to restrict the job part allocation only on the permitted repositories. The equality (6) means that all jobs must be allocated completely. The sizes k_{gr} and $d_{l_g, r, j}$ are the no negative integers (7).

The problem (2) is a generalization of bin packing. In the classical NP-hard bin packing problem one is required to pack a given list of items into the smallest possible number of unit-sized bins. Bin packing has been applied in various areas, e.g.: stock cutting, television programming, transportation, computer storage allocation, bandwidth allocation, scheduling, etc. (Coffman & Csirik, 2007). A difference from classical problem, in the

considered case, one lot is associated with a container formed by R sub-containers (bins of machine) (Fig. 5). The bin capacities in a container vary depending on the parameter g (glass type of lot), and are defined according to the pieces distribution given by row g of the matrix K, so that $k_{g1}+k_{g2}+...+k_{gR}$ represents the capacity of the container g (lot size). A demand is associated with an item that corresponds to a set of identical pieces. For each set is indicated the item quantity and bin numbers that are allowed for packing. Detailed survey of the research on the bin packing problem is given by Coffman & Csirik (2007).

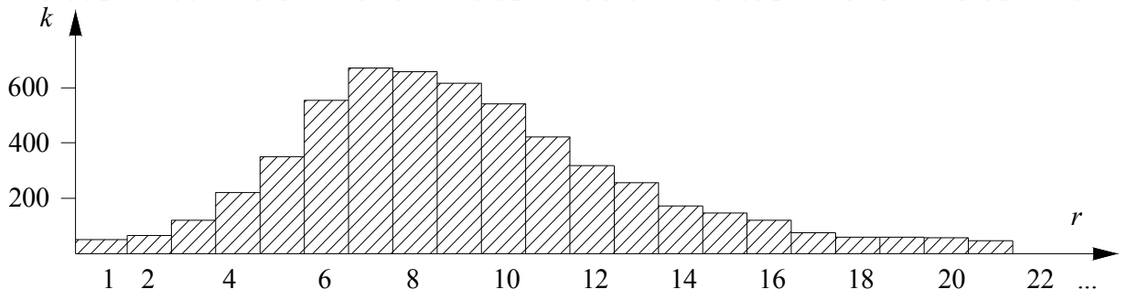

Fig. 5. A container of capacities $k_{g,r}$ formed by R bins,r = 1,…, R,associated with the lot type g.

Shachnai and Tamir (2004) define the problem called the Class-Constrained Bin Packing where the bins have a capacity v and c compartments. In their problem every item has the same size and a color. The items must be deposited on bin, subject to capacity constrains such that items of different colors are placed in different compartments. The goal is to minimize the number of used bins. Fresen and Langston (1986) define the variable sized bin packing problem, where the supply of containers is not only of a single bin type, but some fixed (finite) number of given sizes is available. The bin using cost is simply its size. The goal of the problem is to pack the items into bins witch sum of sizes is minimal. Eptsein and Levin (2008) propose a problem called Generalized Cost Variable Sized Bin Packing. There are given an infinite supply of bins of r types whose sizes are denoted by br < ... < b1 = 1. Items of sizes in (0, 1] are to be partitioned into subsets. A bin type i is associated with a cost c_i , it is assumed c1 = 1. The goal is to find a feasible solution whose total cost is minimized. Langston (1984) investigates the problem of maximizing the number of items packed into m available bins, where the bin sizes can be different. Menakerman and Rom (2001) investigate a bin packing problem variant in which items may be fragmented into smaller size pieces called fragments. Their model is derived from a scheduling problem presented in data over CATV network. Xing (2002) introduces the problem called Bin packing with Oversized Items, where items have a size large than the largest bin size. The bins cannot be overpacked; the oversized item is free to be divided up such that the part is no larger than the largest bin size. Mandal et al. (1998) show that the decision problem for N fragmentable object bin packing when N ≥ 2, is NP-hard. Since this, the problem (2) is NP-hard, too.

5.6 Algorithm

As follows, a heuristic offline algorithm based on the North West Corner rule provides a solution to the problem (2). It finds 1) the quantity of lots indicating for each lot-batch its type and the values of demand parts d_{lrj} allocated on it; 2) the lot sequence.

The algorithm is as follows:

1. Create a table T with N+1 rows and R+1 columns, where each of N rows is used for assignation of pieces of set j, j = 1,…, N , to R sub-container. Initially all cells are zero; n_g = 0, g = 1,…, G. In column R+1 is written the quantity of pieces in each set. The cell on intersection of row j and column r is available if o_{jr}. The unavailable cells are blocked for each row.
2. Sort the rows in the table T using a weight rule (more pieces, fewer pieces, larger number of available sub-container, etc).
3. Create G copies of the table.

   3.1 On the copy T_g, g = 1,…, G, add k_{gr} to the value of the cell r on row N+1, r = 1,..., R.

   3.2 Process the unblocked cells of the table per rows, starting in the upper left corner while the corresponding value on column R+1 is different from zero. The cell values of row N+1 are assigned to the corresponding cells of row r so that the assigned values do not overflow the value on cell (j, R+1). The assigned value is subtracted from the cells (N+1, r) and (j, R+1).

   It continues until the values of all cells in row N+1 are assigned to the available cells or the value of cell (j, R+1) is zero.
4. Calculate the totals in each table T_g, g = 1,… G, as the sum of values on the column R+1.
5. Select the table T_{g^*}, $i \in \{1,\ldots, G\}$, which total is minimum.
   - If totals values in two or more tables are minimal, select the table with minimal total of row N + 1.
   - If there is a tie in the two previous rules, select the first table.

   $n_{g^*} = n_{g^*} + 1$.

Keep the packing history (log): the table state T_{g^*} and the selected index g^*.

Add the table T_{g^*} state to the list Q of tables and the index g* to the list W where each element of the list is the number of type g related with the element in sequence of the list Q. The data position for both lists corresponds to the l index.

The quantity $d_{l_g,r,j}$ of pieces of the job j assigned to the repository r is obtained from the table Tg^* located in the l position of the list Q; the g value is located in the l position of the list W.

Clear the available cells for allocation on selected table.

$T_{g^*} \Rightarrow T$.

6. If exist values different from zero on column R+1, go to 2.
7. End

After first stage, the sublots $d_{l_g,r,j}$, j = 1,…, N, which processing is assigned on the same machine of the second stage, are coupled without mixing. The lot sequence defines the schedule of these batches on the dedicated machines. For considered real problem is characteristic that processing time of the lot is essentially larger than the second operation duration, independently of the machine number at the first stage –see Fig. 6–, therefore, to optimize the obtained schedule, the lots in the sequence must be arranged in decreasing order of the $\sum_{V_j} \sum_{l_g, r \in V_j} d_{l_g,r,j}$.

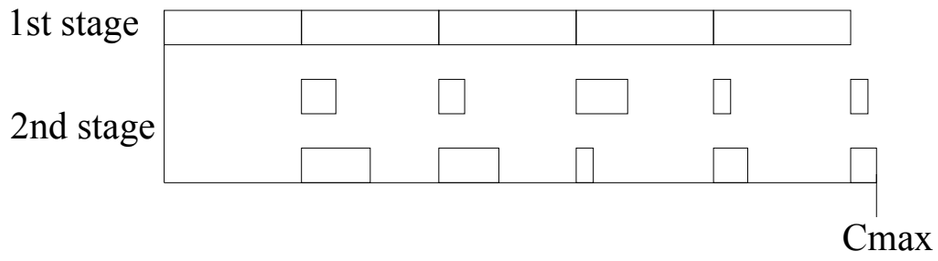

Fig. 6. An example of the problem (2) solution with m_1 =1, m_2 =2.

The next example is presented to illustrate the model working and the algorithm execution: There are 4 demands to carry out (Fig. 7a). For every demand, the next data are indicated: the demanded quantity of pieces, the permitted repositories (in parenthesis), and the assigned dedicated machine (1 or 2). Three lots are necessaries, of types g_1, g_1 and g_2 (Fig. 7b). They arrive on classification machines (Fig. 7c), distributed among machine repositories of the known capacities, and demand are splitted into parts and then are allocated on lots forming the sublots of the sizes $d_{l_{g,r,j}}$ (batching) (Fig. 7d). These sublots associated with the same demand are joined (Fig. 7e), then sublots which processing is assigned on the same machine of the second stage are coupled (Fig. 7f) (rebatching).

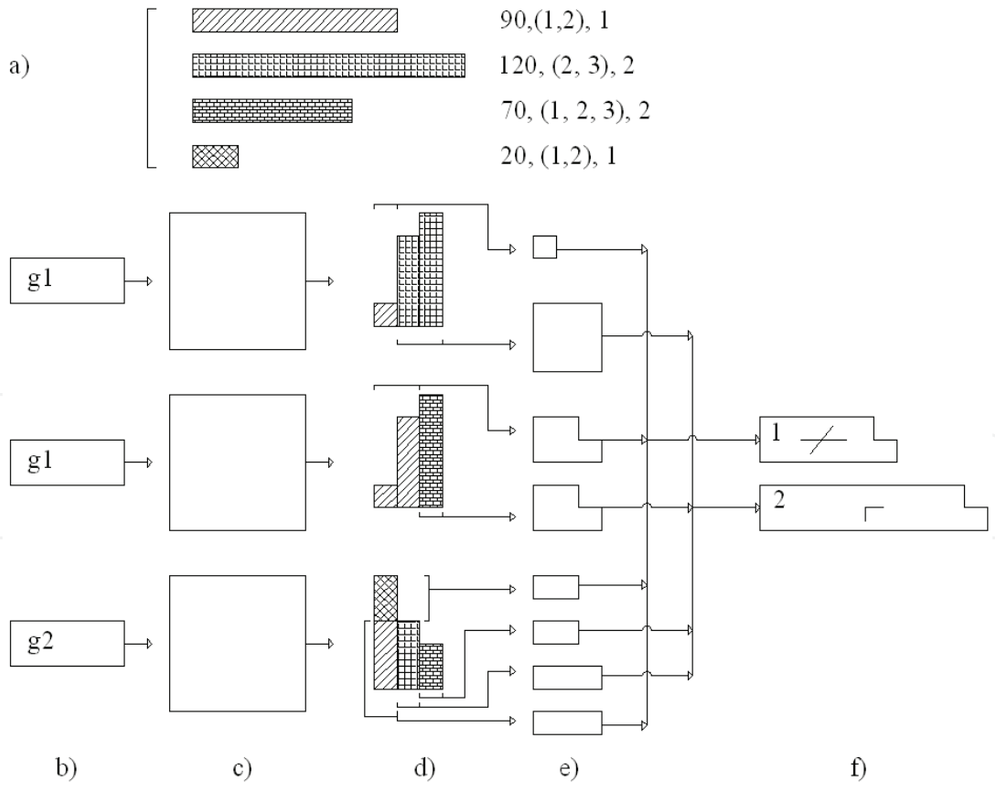

Fig. 7. Model working: a) demands; b) necessary lots indicating lot type; c) first stage machines; d) repository capacities and mode the demands splitting into sublots; e) rebatching sublots.

6. Conclusions

In the last decade, there has been significant interest in scheduling problems that involve lot processing, in consequence of the demand of the modern manufacturing systems. The grouping of jobs into families to process continuously a batch of jobs without any significant setup took to the machines efficiency increasing. But to large batches implicate the large waiting of the operation finish on downstream machines. The splitting of jobs and streaming of lot batches permit to optimize the machine loading to obtain the best schedule. The presented analysis shows that the models in HFS resources environment are insufficiently studied. The concepts described in this work were shown in an example for a real problem of manufacturing electronic components.

7. References

Andrés, C.; Albarracín, J.M.; Tormo, G.; Vicens, E. & García-Sabater, J.P. (2005). Group technology in a hybrid flowshop environment: a case study. *European Journal of Operational Research*, Vol.167, No.1, (November 2005), pp.272-281, ISSN 0377-2217.

Agnetis, A.; Lucertini, M. & Nicolo, F. (1993). Flow management in flexible manufacturing cells with pipeline operations. *Management Science*,Vol.39, No.3, (March 1993), pp. 294-306, ISSN 0025-1909.

Agnetis, A.; Alfieri, A. & Nicosia, G. (2003). Part batching and scheduling in a flexible manufacturing cell to minimize setup costs. *Journal of Scheduling*, Vol.6, No.1, (January-February 2003), pp. 87-108, ISSN 1094-6136.

Azizoglu, M. & Webster, S. (2000). Scheduling a batch processing machine with non-identical job-sizes. *International Journal of Production Research*. Vol.38, No.10, (July 2000), pp. 2173-2184, ISSN 0020-7543.

Baker, K.R. & Pyke, D.F. (1990). Solution procedures for the lot-streaming problem. *Decision Sciences*, Vol.21, No.3, (September 1990), pp. 475–491, ISSN 1540-5915.

Baker, K.R. & Jia, D. (1993). A comparative study of lot streaming procedures. *OMEGA International Journal of Management Sciences*, Vol.21, No.5, (September 1993), pp. 561–566, ISSN 0305-0483.

Bellanger, A. & Oulamara, A. (2009). Scheduling hybrid flowshop with parallel batching machines and compatibilities. *Computers & Operations Research*, Vol. 36, No. 6, (June 2009), pp. 1982-1992, ISSN 0305-0548.

Błażewich, J.; Ecker, K.; Pesch, E.; Schmidt, G. & Węglarz, J. (2007). *Handbook on scheduling: From Theory to Applications*. Springer, ISBN 9783540280460, Berlin, Heidelberg.

Brucker, P.; Gladky, A.; Hoogeveen, H.; Kovalyov, M. Y.; Potts, C. N. & Tautenhahn, T.(1998). Scheduling a batching machine. *Journal of Scheduling*, Vol.1, No.1, (June 1998), pp. 31–54, ISSN 1099-1425.

Brucker, P. (2004). Scheduling Algorithms, Springer, ISBN 3642089070, Osnabru□ck, Germany.

Burbidge, J.L. (1975). The Introduction of Group Technology, *Heinemann Press*, ISBN 0434901938, London.

Chang, J.H. & Chiu, H.N. (2005). A comprehensive review of lot streaming. *International Journal of Production Research,* Vol.43, No.8, (April 2005), pp. 1515–1536, ISSN 0020-7543.

Cheng, T.C.E.; Gupta, J.N.D. & Wang, G. (2000). A review of flowshop scheduling research with setup times. *Production and Operations Management*, Vol.9, No.3, (January 2009), pp. 262–82, ISSN 1937-5956.

Cheng, T.C. E.; Kovalyov, M.Y.; Ng, C.T. & Lam, S.S. (2008). Group sequencing around a common due date. *Discrete Optimization*. Vol.5, No.3, (August 2008), pp. 594-604, ISSN 1572-5286.

Co, H.C.; Biermann, J.S. & Chen, S.K. (1990). A methodical approach to the flexible manufacturing system batching, loading and tool configuration problems. *International Journal of Production Research*, Vol.28, No.12, pp. 2171-2186, ISSN 0020-7543.

Coffman, E. G. Jr. & Csirik, J. (2007). Performance Guarantees for One-Dimensional Bin Packing, In: *Handbook of Approximation Algorithms and Metaheuristics*. Taylor & Francis Group, Chapman and Hall, ISBN 9781584885504, Santa Barbara, USA.

Crama, Y.; Kolen, A.W.J.; Oerlemans, A.G. & Spieksma, F.C.R. (1994). Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, Vol.6, No.1, (January 1994), pp. 33-54, ISSN 0920-6299.

Crama, Y. (1997). Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research,* Vol.99, No.1, (May 1997), pp. 136–153, ISSN 0377-2217.

Çetinkaya & Duman (2010). Lot streaming in a two-machine mixed shop. *International Journal of Advanced Manufacturing Technology,* Vol.49, No. 9-12, *August* 2010, (*August* 2010), pp. 1161–1173, ISSN 0268-3768.

Damodaran, P. & Srihari, K. (2004). Mixed integer formulation to minimize makespan in a flow shop with batch processing machines. *Mathematical and Computer Modelling*,Vol.40, No.13, (December 2004), pp. 1465–1472, ISSN 0895-7177.

Damodaran, P.; Srihari, K.& Lam, S. (2007). Scheduling a capacitated batch processing machine to minimize makespan. *Robotics and Computer-Integrated Manufacturing*, Vol.23, No.2, (April 2007), pp. 208–216, ISSN 0736-5845.

Defersha, F.M. (2011). A comprehensive mathematical model for hybrid flexible flowshop lot streaming problem. *International Journal of Industrial Engineering Computations*, Vol.2, No.2, (April 2011), pp. 283-294, ISSN 1923-2926.

Dobson, G. & Nambimadon, R. S. (2001). The batch loading and scheduling problem. *Operations Research*, Vol.49, No.1, (January 2001), pp. 52–65, ISSN 1526-5463.

Epstein, L. & Levin, A. (2008). An APTAS for generalized cost variable sized bin packing. *SIAM Journal on Computing*, Vol.38, No.1, (March 2008), pp. 411-428, ISSN 1095-7111.

Friesen,D.K. & Langston, M.A. (1986). Variable size bin packing. *SIAM Journal on Computing*, Vol.15, No.1, (February 1986), pp. 222-230, ISSN 1095-7111.

Glass, C.A.; Gupta, J.N.D. & Potts C.N. (1994). Lot streaming in three-stage production processes. *European Journal of Operational Research*, Vol.75, No.2, (June 1994), pp. 378–394, ISSN 0377-2217.

Graves, S.C. & Kostreva, M.M. (1986). Overlapping operations in material requirements planning. *Journal of Operations Management*, Vol.6, No.3, (May-August 1986), pp. 283–294, ISSN 0272-6963.

Gunasekaran, A.; McNeil, R.; McGaughey, R. & Ajasa, T. (2001). Experiences of a small to medium size enterprise in the design and implementation of manufacturing cells.

International Journal of Computer Integrated Manufacturing, Vol.14, No.2, (March 2001), pp. 212–223, ISSN 1362-3052.

Gupta, J. N. D. & Tunc, E. A. (1998). Minimizing tardy jobs in a two-stage hybrid flowshop. *International Journal of Production Research*, Vol.36, No.9, (September 1998), pp. 2397–417, ISSN 0020-7543.

Hadjinicola, G.C. & Kumar, K. R. (1993). Cellular manufacturing at champion irrigation products. *International Journal of Operations and Production Management*, Vol.13, No.9, pp. 53-61, ISSN 0144-3577.

Jeong, S.; Sangbok, W.; Kang, S.& Park, J. (1997). A batch splitting heuristic for dynamic job shop scheduling problem. *Computers & Industrial Engineering*,Vol.33, No.3-4, (December 1997), pp. 781-784, ISSN 0360-8352.

Jin, Z.H.; Ohno, K.; Ito, T. & Elmaghraby, S.E. (2002). Scheduling hybrid flowshops in printed circuit board assembly lines. *Production and Operations Management*, Vol.11, No.2, (January 2009), pp. 216–230, ISSN 1059-1478.

Kempf, K. G.; Uzsoy, R. & Wang, C. S. (1998). Scheduling a single batch processing machine with secondary resource constraints. *Journal of Manufacturing Systems*, Vol. 17, No. 1, pp. 37–51, ISSN 0278-6125

Kim, J. S.; Kang, S. H. & Lee, S. M. (1997). Transfer batch scheduling for a two-stage flowshop with identical parallel machines at each stage. *OMEGA International Journal of Management Sciences*, Vol.25, No.5, (October 1997), pp. 547–555, ISSN 0305-0483.

Kochhar, S. & Morris, R.J.T. (1987). Heuristic methods for flexible flow line scheduling. *Journal of Manufacturing Systems*, Vol.6, No.4, pp. 299–314, ISSN 0278-6125.

Kulonda, D. J. (1984). Overlapping operations—a step toward just-in-time production. *Readings in Zero Inventory*, *Proceedings of APICS 27th Annual International Conference,* pp. 78–80, ISBN 0935406514, Las Vegas, NV, USA, October 9-12, 1984.

Kusiak, A. (1987). The generalized group technology concept, *International Journal of Production Research*, Vol.25, No.4, pp. 561–569, ISSN 0020-7543.

Langston, M. A. (1984). Performance of heuristics for a computer resource allocation problem, *SIAM Journal on Algebraic and Discrete Methods*, Vol.5, No.2, (June 1984), pp. 154-161, ISSN 0196-5212.

Lee, C. Y.; Uzsoy, R. & Martin-Vega, L. A. (1992). Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, Vol.40, No.4, (July-Aug 1992), pp. 764–775, ISSN 1526-5463.

Lee, C. Y. & Uzsoy, R. (1999). Minimizing makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research*, Vol.37, No.1, (January 1999), pp. 219–236, ISSN 0020-7543.

Lee, C.-Y.; Leung, J. Y-T. & Yu, G. (2006). Two machine scheduling under disruptions with transportation considerations. *Journal of Scheduling*, Vol.9, No.1, (February 2006), pp. 35-48, ISSN 1099-1425.

Li, S. (1997). A hybrid two-stage flowshop with part family, batch production, major and minor set-ups.*European Journal of Operational Research*, Vol.102, No.1, (October 1997), pp. 142-156, ISSN 0377-2217.

Liao, L. M., & Huang, C. J. (2008). An effective heuristic for two-machine flowshop with batch processing machines. *Proceedings of The 38th conference on computers and*

industrial engineering (ICCIE2008). ISBN 978-7-121-07437-0, Beijing, China, October 31 – November 2, 2008.

Liu, C.Y. & Chang, S.C. (2000). Scheduling flexible flow shops with sequence-dependent setup effects. *IEEE Trans Robotics Automation*, Vol.16, No.4, (August 2000), pp. 408-419, ISSN 1042-296X.

Liu, J. (2008). Single-job lot streaming in m-1 two-stage hybrid flowshops. *European Journal of Operational Research,* Vol.187, No.3, (June 2008), pp. 1171-1183, ISSN 0377-2217.

Luo, H.; Huang, G. Q.; Zhang, Y.; Dai, Q. & Chen, X. (2009).Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm. *Robotics and Computer Integrated Manufacturing,* Vol.25, No.6, (December 2009), pp. 962-971, ISSN 0736-5845.

Lushchakova,I. N.& Strusevich, V. A. (2010). Scheduling incompatible tasks on two machines. European Journal of Operational Research, Vol.200, No.2, (January 2010), pp. 334-346, ISSN 0377-2217.

Manjeshwar, K.; Damodaran, P. & Srihari, K. (2009). Minimizing makespan in a flow shop with two batch-processing machines using simulated annealing. *Robotics and Computer-Integrated Manufacturing,* Vol.25, No.3, (June 2009), pp. 667-679, ISSN 0736-5845.

Mandal, C. A.; Chakrabarti, P. P. & Ghose, S. (1998).Complexity of fragmentable object bin packing and an application. *Computers & Mathematics with Applications*, Vol.35, No.1, (June 1998), pp. 91-97, ISSN 0898-1221.

Marimuthu, S.; Ponnambalam, S.G. & Jawahar, N. (2009). Threshold accepting and ant-colony optimization algorithm for scheduling m-machine flow shop with lot streaming, *Journal of Material Processing Technology*, Vol.209, No.2, (January 2009), pp. 1026-1041, ISSN 0898-1221.

Mathirajan, M. & Sivakumar, A. L. (2006). A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *International Journal of Advanced Manufacturing Technology*, Vol. 29, No.9-10, (July 2006), pp. 990-1001, ISSN 1433-3015.

Mathirajan, M.; Bhargav, V. & Ramachandran, V. (2010). Minimizing total weighted tardiness on a batch-processing machine with non-agreeable release times and due dates. *The International Journal of Advanced Manufacturing Technology*, Vol.48, No.9-12, (June 2010), pp. 1133-1148, ISSN 1433-3015.

Menakerman, N. & Rom, R. (2001). Bin Packing with Item Fragmentation. *Proceedings of the 7th International Workshop on Algorithms and Data Structures WADS*, LNCS Vol. 2125, ISBN: 9783540424239, pp. 313-324, Providence, RI, USA, August, 8-10, 2001.

Morita, H. & Shio, N. (2005). Hybrid branch and bound method with genetic algorithm for flexible flowshop scheduling problem. *JSME International Journal Series C*, Vol.48, No.1, pp. 46–52, ISSN 1347-538X.

Mohring, R.H. (1990). Graph problems related to gate matrix layout and PLA folding, in: *Computational Graph Theory*. G. Tinhofer et al. (eds.), pp. 17-51, *Springer-Verlag*, ISBN 3211821775, Vienna, Austria.

Mitrofanov, S.P. (1966). *Scientific Principles of Group Technology*. National Lending Library, Yorkshire, UK.

Oğuzc, C.; Zinder, Y.; Do, V. H., Janiak, A. & Lichtenstein, M. (2004). Hybrid flow-shop scheduling problems with multiprocessor task systems. *European Journal of Operational Research*, Vol.152, No.1, (June 2010), pp. 115–131, ISSN 0377-2217.

Oğuzc, C. & Frinket, M. E. (2005). A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *Journal of Scheduling*, Vol.8, No.4, (July 2005), pp. 323–351, ISSN 1099-1425.

Pan, Q.-K.; Tasgetiren M. F.; Suganthan, P.N. & Chua, T.J. (2010). A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. *Information sciences,* Vol.18, No.12, (June 2011), pp. 1-14, ISSN 0020-0255.

Perez, I. C., Fowler, J. W., & Carlyle, W. M. (2005). Minimizing total weighted tardiness on a single batch process machine with incompatible job families. *Computers and Operations Research*, Vol. 32, No.2, (February 2005), pp. 327–341, ISSN 0305-0548.

Petrov, V.A. (1966). Flowline Group Production Planning, *Business Publications*, ISBN 0220794715, London.

Pinedo, M. L. (2008). Scheduling: Theory Algorithms, and Systems. *Springer Science+Business Media*, ISBN: 978-0-387-78935-4, NY.

Potts, C.N. & Baker, K.K. (1989). Flow shop scheduling with lot streaming. *Operation Research Letters*, Vol.8, No.6, (December 1989), pp. 297–303, ISSN 0167-6377.

Potts, C.N. & Van Wassenhove, L.N. (1992). Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. *Journal of the Operational Research Society*, Vol.43, No.5, (May 1992), pp. 395-406, ISSN 0160-5682.

Potts, C. N. & Kovalyov, M. Y. (2000). Scheduling with batching: a review. *European journal of operational research,* Vol.120, No. 2, (January 2000), pp. 228-249, ISSN 0377-2217.

Quadt, D. & Kuhn H. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research,* Vol.178, No.3, (May 2007), pp. 686–98, ISSN 0377-2217.

Reiter, S. (1966). A system for managing job shop production. *Journal of Business*, Vol.34, pp. 371-393, ISSN 00219398.

Romero Parra, R. & Burtseva, L. (2010). Implementation of Bin Packing Model for Reed Switch Production Planning, In: *IAENG Transactions on Engineering Technologies: Special Edition of the World Congress on Engineering and Computer Science-2009, San Francisco, CA, USA, October 20–22 2009*, Sio-Iong Ao (editor), Vol. 1247, pp. 403-412, AIP Conference Proceedings: ISBN: 978-0-7354-0794-7, Melville, NY.

Ruiz, R.; Serifoglu, F.S. & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research,* Vol.35, No.4, (April 2008), pp. 1151-1175, ISSN 0305-0548.

Ruiz, R. & Vazquez-Rodriguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research,* Vol. 205, No.1, (August 2010), pp. 1-18, ISSN 0377-2217.

Sammaddar, S., Rabinowitz, G., & Mehrez, A. (1999). Resource sharing and scheduling for cyclic production in a computer integrated manufacturing cell. *Computers & Industrial Engineering*, Vol.36, No.3, (July 1999), pp. 525-547, ISSN 0360-8352.

Santos, D.L., Hunsucker, J.L. & Deal, D.E. (1995). Global lower bounds for flow shops with multiple processors. *European Journal of Operational Research*, Vol.80, No.1, (January 1995), pp. 112-120, ISSN 0305-0548.

Shachnai, H. & Tamir,T. (2004). Tight Bound for online class-constrained packing. *Proceedings of the 6th Latin American Symposium LATIN 2004: Theoretical Informatics*, ISBN 3540212582, pp. 103-123, Buenos Aires, Argentina, April 5-8, 2004.

Sodhi, M.S.; Agnetis, A. & Askin, R.G. (1994). Tool addition strategies for flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, Vol.6, No.4, (October 1994), pp. 287-310, ISSN 1572-9370.

Stecke, K.E. & Kim, I. (1988). Study of FMS part type selection approaches for short term production planning. *International Journal of Flexible Manufacturing Systems*, Vol.1, No.1, (September 1988), pp. 7-29, ISSN 1572-9370.

Tang, C.S. & Denardo, E.V. (1988a). Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches. *Operations Research*, Vol.36, No.5, (September – October 1988), pp. 767-777, ISSN 1526-5463.

Tang, C.S. & Denardo, E.V. (1988b). Models arising from a flexible manufacturing machine, Part II: Minimization of the number of switching instants. *Operations Research*, Vol.36, No.5, (September – October 1988), pp. 778-784, ISSN 1526-5463.

Tang, L.; Luh, P.B.; Liu, J. & Fang, L. (2002).Steel-making process scheduling using Lagrangian relaxation. *International Journal of Production Research*, Vol.40, No. 1, (January, 2002), pp. 55–70, ISSN 0020-7543.

Tatikonda, M.V. & Wemmerlov, U. (1992). Adoption and implementation of group technology classification and coding systems: Insights from seven case studies. *International Journal of Production Research*, Vol.30, No.9, pp. 2087–2110, ISSN 0020-7543.

Trietsch, D. & Baker, K.R. (1993). Basic techniques for lot streaming. *Operations Research,* Vol.41, No. 6, (November-December 1993), pp. 1065–1076, ISSN 0030-364X.

Truscott, W. G. (1986). Production scheduling with capacity-constrained transportation activities. *Journal of Operation Management*, Vol.6, No.3-4, (November-December 1993), pp. 333–348, ISSN 0272-6963.

Tsubone, H.; Ohba, H. & Uetake, T. (1996). The impact of lot sizing and sequencing in manufacturing performance in a two-stage hybrid flow shop. *International Journal of Production Research*, Vol.34, No.11, pp. 3037–3053, ISSN 0020-7543.

Umble, M.M. & Srikanth, L. (1990). Synchronous Manufacturing: Principles for World Class Excellence. The Spectrum Publishing Company, Wallingford, CT, USA, ISBN 0-943953-05-7

Uzsoy, R. (1994). Scheduling a single batch processing machine with non-identical job sizes. *International Journal of Production Research*, Vol.32, No.7, pp. 1615–1635, ISSN 0020-7543.

Uzsoy, R. (1995). Scheduling batch processing machines with incompatible job families. *International Journal of Production Research*, Vol.33, No.10, pp. 2685–2708, ISSN 0020-7543.

Vickson, R.G. & Alfredsson, B.E. (1992). Two and three machines flow shop scheduling problems with equal sized transfer batches. *International Journal of Production Research*, Vol.30, No.7, pp. 1551–1574, ISSN 0020-7543.

Wemmerlov, U. & Hyer, N.L. (1989). Cellular manufacturing in the US industry: a survey of current practices. *International Journal of Production Research*, Vol.27, No.9, pp. 1511–1530, ISSN 0020-7543.

Xuan, H. & Tang, L.X. (2007). Scheduling a hybrid flowshop with batch production at the last stage. *Computers & Operations Research*, Vol.34, No.9, (September 2007), pp. 2718–33, ISSN 0305-0548.

Xing W. & Zhang, J. (2000). Parallel machine scheduling with splitting jobs. *Discrete Applied Mathematics*, Vol.103, No. 1-3, (July 2000), pp. 259–69, ISSN 0166-218X

Xing, W. (2002). A bin packing problem with over-sized items. *Operations Research Letters*, Vol.30, No.2, pp. 83-88, ISSN 0167-6377.

Yaurima, V.; Burtseva, L. & Tchernykh, A. (2009). Hybrid Flowshop with Unrelated Machines, Sequence Dependent Setup Time, Availability Constraints and Limited Buffers. *Computers & Industrial Engineering*, Vol.56, No.4, (May 2009), pp. 1452-1463, ISSN 0360-8352.

Yazdani, S.M.T. & Jolai, F. (2010). Optimal methods for batch processing problem with makespan and maximum lateness objectives. *Applied Mathematical Modelling,* Vol.34, No.2, (February 2010), pp. 314-324, ISSN 0307-904X.

Ying, K.-C. & Lin, S.-W. (2006). Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach. *International Journal of Production Research*, Vol. 44, No.16, (August 2006), pp. 3161–3177, ISSN 0020-7543.

Zhang, W.; Liu, J. & Linn, R. (2003). Model and heuristics for lot streaming of one job in M-1 hybrid flowshops. *International Journal of Operations and Quantitative Management*, Vol.9, (March 2002), pp. 49–64.

Zhang, W.; Yin, C.; Liu,J. & Linn, R. J. (2005). Multi-job lot streaming to minimize the mean completion time in m-1 hybrid flowshops. *International Journal of Production Economics,* Vol.96, No.2, (May 2005), pp. 189-200, ISSN 0925-5273.

Production Scheduling

Edited by Prof. Rodrigo Righi

ISBN 978-953-307-935-6
Hard cover, 242 pages
Publisher InTech
Published online 11, January, 2012
Published in print edition January, 2012

Generally speaking, scheduling is the procedure of mapping a set of tasks or jobs (studied objects) to a set of target resources efficiently. More specifically, as a part of a larger planning and scheduling process, production scheduling is essential for the proper functioning of a manufacturing enterprise. This book presents ten chapters divided into five sections. Section 1 discusses rescheduling strategies, policies, and methods for production scheduling. Section 2 presents two chapters about flow shop scheduling. Section 3 describes heuristic and metaheuristic methods for treating the scheduling problem in an efficient manner. In addition, two test cases are presented in Section 4. The first uses simulation, while the second shows a real implementation of a production scheduling system. Finally, Section 5 presents some modeling strategies for building production scheduling systems. This book will be of interest to those working in the decision-making branches of production, in various operational research areas, as well as computational methods design. People from a diverse background ranging from academia and research to those working in industry, can take advantage of this volume.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen