7

Scheduling methods for hybrid flow shops with setup times

Larysa Burtseva[a], Victor Yaurima[b] and Rainier Romero Parra[c]
[a]*Autonomous University of Baja California, Mexicali,*
[b]*CESUES Superior Studies Center, San Luis Rio Colorado, Sonora,*
[c]*Polytechnic University of Baja California, Mexicali,*
Mexico

1. Introduction

Many real manufacturing systems process a large number of product variants in the same flow. These products may differ in some optional components; consequently, the processing time on a machine differs from one product to the next, and the need to prepare one or more machines before beginning or after the finishing of jobs is frequently presented. The preparation activities are: machine adjustment and feeders preparation to process a next job, dismantling after a previous job, machine calibrating, inspection of accessories or tools, cleaning of the machines and adjacent areas, etc. In the scheduling theory, the time required to shift from one job to another on a given machine is defined as additional production cost or setup time. The scheduling problems, which consider the setup times, have a high computational complexity. Pinedo (2008) presents a proof of the NP-hardness of the single machine case with setup consideration. They are more complex when the resource model has the parallel machine environment.

The time that a job spends on a machine includes three phases: setup, processing, and removal. In the majority of investigations dedicated to production planning and scheduling it is assumed that the setup/removal times are negligible or nonseparable, therefore they are included in the job processing time, and hence are ignored. The nonseparable setup time assumption simplifies the analysis, and these problems can be formulated and solved as standard scheduling problem. However, an explicit treatment of the setup times in most applications is required and represents a special interest, because machine setup time is a significant factor for production scheduling in many cases. It may easily consume more than 20% of available machine capacity if it is not well handled (Pinedo, 2008).

Numerous examples of scheduling problems which consider separable setup times are given in the literature, including electronics manufacturing, automobile assembly plant, the packaging industry, textile industry, steel manufacturing, airplane engine plant, label sticker manufacturing company, semiconductor industry, maritime container terminal, ceramic tile manufacturing sector, as well as in electronics industry in sections for inserting components on printed circuit boards (PCB), where this kind of problems is frequent.

The purpose of this chapter is to present a class of deterministic scheduling problems in a multi-stage parallel machine environment called hybrid flow shop with setup times and appropriate methods for its resolution. The chapter includes a description of model with necessary definitions and notations; concepts of product family and batch, which are important elements of setup time analysis as well as a classification of setup times and problems that each category produces. The last section is focused on problems with sequence-depended setup times in hybrid flow shops. A review of investigated cases is explained, including the application of genetic algorithms for this kind of scheduling problems: structure of a genetic algorithm and description of several crossover operators appropriated to use based on previous investigations of authors. This section includes an algorithm and an example of a complex problem solution. A conclusion is presented at the chapter end.

2. Hybrid flow shop with setup times

In the scheduling theory, a multi-stage production process with the property that all products have to pass through a number of stages in the same order is classified as a flow shop. In a simple flow shop, each stage consists of a single machine, which handles at most one operation at a time. It is more realistic to assume that, at every stage, a number of machines that operate in parallel are available. This model is known as a hybrid flow shop (HFS). Some stages may have only one machine, but for the model to be qualified as a HFS, at least one stage must have multiple machines in parallel. These machines can be identical, or have different capacities. Each job is processed by at most one machine at each stage. The flow of products in the plant is unidirectional; each product is processed at only one machine in each stage.

The HFS models are common in the industry, which have the same technological route for all products as a sequence of stages, and any stages have a group of machines to realize the same operation. Various process industries, such as chemical, textile, metallurgical, semiconductors, printed circuit board, pharmaceutical, oil, food, and automobile manufacture, can be modeled as a HFS. In such industries, at some stages the facilities are duplicated in parallel 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.

Among scheduling problems which consider separable setup times in parallel machine environment, there is a class of problems of a high computational complexity, where setup from one product to another occurs on a machine; and machine parameters, which have to be changed during a setup, differ according to the production sequence. It leads to sequence-dependent setup times and consequently to sequence-dependent setup costs.

A HFS with setup times has the following characteristics:

- There are k stages of processing in a linear order: 1, 2, …, k.
- Each of the n jobs visits the stages in this order, though all jobs do not need to visit all stages. Stages may be skipped for a particular job, but the process flow for each job is the same.

- Each stage has a predetermined number of parallel machines. However, the number of machines varies from stage to stage.
- The processing time for every job on every machine that it visits is known in advance and is constant.
- A job represents the processing of an item or a set of identical items (a container, a pallet, a box, a lot or a part) called batch.
- The jobs can belong to different job families. Jobs from the same family may have different processing times, but they can be processed on a machine after another without requiring any adjustment of machine in between.
- Every job is to be processed on one machine at a time without preemption and a machine processes no more than the job at a time. When an operation is started on a machine, it must be finished without interruption.
- Typically, buffers are located between stages to store intermediate products.
- The problem consists of assigning the jobs to machines at each stage and sequencing the jobs assigned to the same machine so that some optimality criteria are minimized.

The following index are used to describing the problems: j for job, j = 1,…, n, i for stage, i = 1, 2, …,k; m_i for number of machines at the stage i; l for machine index, l = 1, 2, …, m_i.

The three-field notation $\alpha|\beta|\gamma$ is used to describe all details of considered HFS problem variant. The α field denotes the shop configuration, including the shop type and machine environment per stage. The α field discomposes into four parameter, i.e. α_1, α_2, α_3, and α_4, 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, and parameter α_2 indicates the number of stages. For the HFS notation, FH is in the α_1 position, and the value of parameter α_2 has to be major that one. For each stage, parameters α_3 and α_4 indicate the machine set environments. More specifically, α_3 indicates information about the type of the machines while α_4 indicates the number of machines in the stage.

The possible machine set environments on the stage i of a HFS are:

1. *Single machine* (1): a special case; any stages (not all) in a HFS can have only one machine.
2. *Identical machines in parallel* (Pm_i): job j may be processed on any of m_i machines;
3. *Uniformed machines in parallel* (Qm_i): the m_i machines in the set have different speeds; a job j may be processed on anyone machine of set, however its processing time is proportional of the machine speed.
4. *Unrelated machines in parallel* (Rm_i): a set of m_i different machines in parallel. The time that a job spends on a machine depends on the job and the machine.

When there are several consecutive stages with the same machine set environments, the parameters α_3 and α_4 can be grouped as $((\alpha_3\alpha_4^{(i)})_{i=s}^{k})$, where s and k are the index of the first and the last consecutive stage, respectively. For example, the notation $FH4$, $(1,(P2^{(i)})_{i=2}^{3},R3^{(4)})$ refers to a HFS configuration with four stages where there are one machine at the first stage, two identical machines in parallel at second and third stages and three unrelated parallel machines in the fourth stage.

The β field provides the shop properties; also other conditions and details of the processing characteristics, which may enumerate multiple entries, also may be empty if they are not.

The following model properties are frequently associated with a setup time HFS scheduling problem:

batch(*b*) Batch processing. A machine is able to process up to *b* jobs continuously without any setup.

brkdown Machine breakdown implies that a machine may not be continuously available.

fmls Job families. The *n* jobs belong to *F* different job families. Jobs from the same family may have different processing times, but they can be processed on a machine after another without requiring any setup in between.

M_{jk} Machine eligibility restrictions. Processing of job *j* is restricted to the set *Mj* of machines at stage *k*.

r_j Release dates. The job *j* cannot start processing before release data r_j.

R Removal time. Machines become free only after the setup of the job has been removed.

S_{si} Sequence-independent setup times. The setup time of machine depends only on the job to process and does not depend on the previous job.

S_{sd} Sequence-dependent setup times. The setup time of machine required to process next job depends on the previous job.

w_j The priority factor denoting the weight or importance of job *j* relative to the other jobs of system.

The γ field establishes the objective to be minimized. The more common objective functions to minimize in a HFS scheduling problem are: $C_{\max}$ as maximum completion time; $F_{\max}$ as maximum flow time; $L_{\max}$ as maximum lateness; $T_{\max}$ as maximum tardiness; $E_{\max}$ maximum earliness, among others. The most used objective function to be minimized in a HFS scheduling problem is the completion time when the last job to leave the system, referred to a makespan or $C_{\max}$.

A HFS standard scheduling problem with *k* stages and a number of the identical parallel machines in each stage is denoted as *FHk*, $((PM^{(i)})_{i=1}{}^{k})\,|\,|\,C_{max}$. In this case, the formula defines a HFS with *k* stages, $|M^{(i)}|$ identical machines in parallel on stage *i*, $i = 1, \ldots, k$; there are not any special parameter β, and the objective is the makespan minimizing.

Figure 1 illustrates the physical relationship between machines and stages, which corresponds to the notation $FH3, (1, P3^{(2)}, R2^{(3)})\,|\,M_{j3}, S_{sd}\,|\,T_{\max}$, referring to tri-stage HFS. The stage 1 has one machine, stage 2 has three identical machines in parallel, and stage 3 has two parallel unrelated machines; M_{j3} and S_{sd} indicate that there are machine eligibility restrictions at stage 3 and setup times depended on the sequence of jobs. The objective is the maximum tardiness minimizing. Moreover, the figure shows that there are unlimited buffers between stages to storage unfinished products, so called Work In Process (WIP).

A production system, to be classified as a HFS has to be flexible. It is important to know the differences between a flexible production system and a traditional one; what exactly means the concept of flexibility and what justifies the use of specific production planning models for flexible production systems. Automated manufacturing systems display flexibility in multiple and intertwined ways, pertaining to the equipment, processes, products,

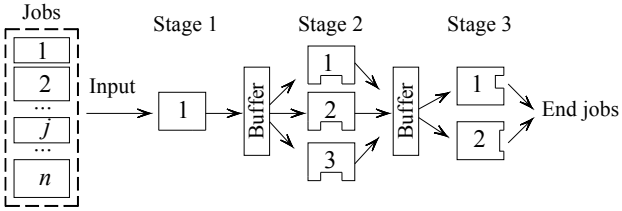

Fig. 1. Resource model for a tri-stage HFS.

production volumes, etc. Among the more important concepts, are the following (Crama, 1997), (Vairaktarakis, 2004):

1. *machine flexibility,* the ability of the machines to perform various types of operations without requiring a prohibitive effort in switching from one operation to another;
2. *material handling flexibility,* the ability of the material handling system to move different parts efficiently for proper positioning and processing through the manufacturing facility;
3. *operation flexibility*, the ability to realize it in different ways;
4. *processing flexibility,* that means that jobs may skip stages or there is a set of part types that the system can produce without major setups;
5. *routing flexibility,* the ability of a manufacturing system to produce a part by alternate routes through the system.

A planning production model with sets machines in parallel has to comply with one of these concepts to be classified as a flexible flow shop tacking in account that the flow of products in the plant is unidirectional. The hybridizing occurs when any products require special manufacture conditions, e.g., different qualities and capacities of machines at the same stage, assignment any jobs on certain machines, and another special conditions.

Meanwhile, the HFS has been studied since the 70th, the researcher put much attention to this model and some new designs were discovered on the recent years. This fact probably implicates confusions in the terminology and notations. Actually, there is not in the literature a conventional classification of this kind of flow shops. A variety of known models should be interpreted as a HFS or its special case.

There are:

Flexible flow shop (FFS); a HFS in the parallel identical machine environment when the machines in each set are identical (*processing flexibility* within a production stage which is derived from the ability to process a job on any parallel machine at stage). Some authors, as e.g., Pinedo (2008), Jungwattanakit et al., (2009) do not use the notion HFS, and describe the more complex configurations as a FFS with not identical parallel machines at least on one stage. Moreover, a variety of authors do not differ between terms of FFS and HFS referring this model as a flexible (hybrid) flow shop (Allahverdi et al., 2008); or use the HFS term in parallel identical machine environment (Naderi et al., 2009).

Flexible flow line (FFL) and *Flow shop with multiple processors* (FSMP or MPFS) are equivalent to a FFS (Lin & Liao, 2003). Zandieh at al. (2006) considering that the HFS is known commonly as a flexible flow line, because the flow of jobs in that system is unidirectional.

Hybrid flexible flow shop or Flexible hybrid flow line (HFFL); this model is equivalent to a HFS where jobs might skip stages (*processing flexibility* across production stages) (Ruiz & Vazquez-Rodriguez, 2010), (Allahverdi et al., 2008)

Parallel HFS (PHFS) system represents a HFS decomposed into smaller HFS sub-designs operated in parallel. More specific, a PHFS is composed of a number of independent sub-designs each of which is a HFS of the unidirectional routing (*routing flexibility*) (Vairaktarakis, 2004).

The HFS scheduling problems which consider setup times are among the most difficult classes of scheduling problems. It is known, that a one-machine sequence-dependent setup scheduling problem is equivalent to a traveling-salesman problem which is NP-hard, even for a small system, the complexity of this problem is beyond the reach of existing theories (Pinedo, 2008). A HFS restricted to two processing stages, even in the simplest case when one stage contains two identical machines and the second only a single machine, is already NP-hard, according to Gupta (1988). Moreover, the special case where there is a single machine per stage, known as the flow shop, and the simplest case where there is a single stage with several machines, known as the parallel machine environment, are also NP-hard (Glover & Laguna., 1997). The total number of possible solutions for a HFS to be $n!(\Pi_{i=1}^{k} m_i)^n$ while the number of possible solutions in a regular flow shop scheduling problem is $n!$ The complicity of a HFS scheduling problem with setup time condition depends essentially on setup time nature.

3. Batch processing

A technical similarity between products of a plant often reflects an obvious grouping of them into product groups. Products can be sorted out into groups according to their design attributes, which include part shape, size, surface texture, material type, raw material estate, or according to their manufacturing attributes. The technical similarities of the products within a group permit reduce essentially the setups number on a machine, when a setup from one product to another occurs and hence manufacturing time would be decreased and consequently machine usage time would be improved.

This idea is adapted by the Group Technology (GT) (Andrés et al., 2005). The GT concept is based on the simplification and standardization process. It was dedicated originally to single machine environment to reduce setup times. This concept was further extended to the production planning in productive systems which have some available resources in each of the stages of production and not negligible setups known as the HFS problem with setup times dependent on the sequence (Li, 1997).

From the GT surge the concepts of 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. Batching occurs only if setup costs or times are not negligible and several jobs of the same product type have to be produced. When the processing is realized in batches (lots, pallets, containers, boxes), the operations processed simultaneously start together and complete together, with just a single setup in the beginning. Their processing time depends only on the family of the batch.

When one batch is completed, the resources have to be adjusted for the next batch. The time needed for the setup depends on the families of both adjacent batches. A batch is called

feasible if it can be processed without any tool switches. While families are supposed to be given in advance, batch formation is a part of the decision making process. To *batch-sizes* calculating has to decide how many units must be produced consecutively. In (Liu & Chang, 2000) is indicated that the processing in large batches may increase machine utilization and reduce the total setup time. However, large batch processing increases the flow time. There is a tradeoff between flow time and machine utilization by selecting batch size and scheduling. According to the GT, no family can be split, only a single batch can be formed for each family.

Batch setup models are further partitioned into *batch availability* and *job availability* models. According to the batch availability model, all the jobs of the same batch become available for processing and leave the machine together. Two rules that define the processing time of a batch are distinguished (Lushchakova & Strusevich, 2010):

- In the case of sequential batch processing, also known as ''sum-batch", the processing time of a batch on machine is equal to the total processing times of its jobs;
- In the case of simultaneous batch processing, also known as ''max-batch", the processing time of a batch on machine is equal to the largest processing time of its jobs.

In the job availability model, each job's start and completion times are independent on other jobs in its batch.

The term of *family* denotes initial job partitioning, while the term of *batch* is used to denote a part of the solution. Many publications use the term batch to denote the initial job partitioning and they use different names like sub-batch, lot, sub-lot, etc., to denote a set of jobs of the same family processed consecutively on the same machine. In the literature, a job availability model is considered, if not stated otherwise.

Li (1997) gives an example of scheduling problem from an airplane engine plant, Pratt and Whitney Inc. (PWI). The blade line, one of the production lines at PWI, characterized as a two-stages HFS, produces various types of blades used in airplane engines. Each stage of the blade line at PWI has a different number of machines. The types of blades that have similar processing requirements are grouped into families. A major setup is required if a machine at any stage switches from one family of blades to the other. A minor setup is required if a machine switches from one type of blade to another type in the same family. Since setup times are not insignificant and unit processing times for all types of blades are very short, the plant processes each type of blade in batches (lots).

The *batch setup time* (cost) can be machine dependent or sequence (of families) dependent. It is sequence-dependent if its duration (cost) depends on the families of both the current and the immediately preceding batches, and is sequence-independent if its duration (cost) depends solely on the family of the current batch to be processed.

A HFS scheduling problems with setup times which consider job processing in batches can be *sequence-dependent* as well as *sequence-independent*. Most studies assume that either no setup has to be performed or that setup times are sequence-independent and there is only a single unit of each product type. In this case, a job's setup time may be added to its process time. However, if setup times are sequence-dependent or if several jobs of the same product type have to be produced, setups have to be considered explicitly.

In a non-batch processing environment, a setup time (cost) is incurred prior to the processing of each job. The corresponding model can also be viewed as a batch setup time (cost) model in which each family consists of a single job.

4. Classification of HFS with setup times

The setup times, defined as the time required to shift from one job to another on a given machine, are considered as separable and non-separable from the processing 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 nonseparable 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 an automatic guided vehicles and robots, loading and unloading (Crama, 1997), (Kim et al., 1997), (Pinedo, 2008).

When these operations are *separable*, i.e. they are not a part of processing operation, the structure of the breakdown time when a job belongs to a machine is as follows (Cheng et al., 2000):

1) Setup time that is independent on the job sequence. This operation consists of activities such as fetching the required details, and fixtures, and setting them up on the machine.
2) Setup time that is dependent on the job to be processed. The carrying out of this operation includes the time required to put the job in the jigs and fixtures and to adjust the tools.
3) Processing time of the job being processed.
4) Removal time that is independent on the job that has been processed. This operation includes activities such as dismounting the jigs, the fixtures and/or tools, inspecting/sharpening of the tools, and cleaning the machine and the adjacent area.
5) Removal time that is dependent on the job that just has been processed. This operation includes activities such as disengaging the tools from the job, and releasing the job from the jigs and fixtures.

Three phases of job processing can be grouped as following: the separable setup, the processing, and the separable removal times represented by items (1, 2), (3), and (4, 5), respectively. When separable setup/removal times are not negligible in the scheduling problem, they should be explicitly considered.

The setup times which are separable from the processing 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 can start only when the job arrives at a machine as the setup is attached to the job. Further, setup times of a job at a specific machine could be dependent on the job immediately preceding that job or be independent on it. If it is not stated explicitly that setups are non-anticipatory.

As follows, a classification of HFS scheduling problems with setup times, derivate from the classification presented in (Cheng et al., 2000), is described. These problems generally fall into the following four board categories depending on practical situations (Figure 2):

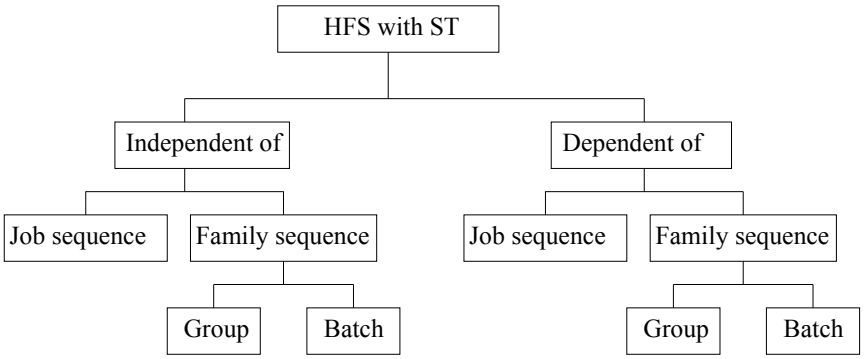

Fig. 2. Classification of HFS with setup times.

- HFS with sequence-independent job setup times.
- HFS with sequence-dependent job setup times.
- HFS with sequence-independent family setup times:
  - HFS with sequence-independent group setup times;
  - HFS with sequence-independent batch setup times.
- HFS with sequence-dependent family setup times:
  - HFS with sequence-dependent group setup times;
  - HFS with sequence-dependent batch setup times.

HFS with sequence-independent job setup times. The setup times are separable from the processing times and sequence-independent, i.e., depend only on the job to process and do not depend on the job sequence on this machine. Such setup times could be either detached or attached to the processing times. However, if this setup time is attached to a job, the idle time of a machine cannot be used, and hence the setup time have to be considered as a part of the processing time and the problem can be formulated and solved as a standard HFS problem. For this reason, only *detached* sequence-independent job setup times represent a special interest. Further, removal times could be either zero or positive. The removal times, if they are present, have to be included in makespan definition.

Three papers with different setup/removal restrictions are mentioned as references. In (Kim et al., 1997) the C_{max} minimizing problem for FFS with two stages, independent setup times and negligible removal times is considered. A scheduling rule similar to the Johnson's rule is suggested to minimize makespan. Another work, (Low, 2005), addresses to a HFS with J stages and unrelated parallel machines at each stage, independent setup and dependent removal times. The objective is to minimize total flow time in the system. A simulated annealing-based heuristic is proposed to solve the addressed problem in a reasonable running time. In (Harjunkoski & Grossmann, 2002) is considered a HFS model where setup

times are included, but they are only dependent on the machine and not on the job. The objective is to minimize job assignment costs and one-time machine-initialization costs.

HFS with sequence-dependent job setup times (Li, 1997), (Naderi et al., 2009).This situation occurs when the part of the setup of job *i* can be used for processing the next job *j*. It implicates that the removal time of the job *i* will depend on the job *j* to be processed next. On the other hand, the setup time of job *j* also depends on the job *i* being processed currently, because the setup part of the previous job *i* can be used for the next job *j*. The net effect of these two factors is that the setup time of job *j* depends on the immediately preceding job *i*. Sequence-dependent setup times are of the anticipatory (detached) type because their nature. Therefore, the setup information cannot move with the job; and sequence-dependent setup times cannot be of the attached type as information about the currently processed job *i*; and the next job *j* requires to determine the needed setup time to be processed.

HFS with sequence-independent/dependent family setup times. In the above two categories the setups are associated with individual jobs. However, in many real-world situations, the processing of jobs is realized taking in account the job family. When jobs belonging to the same family scheduled contiguously, they only need a common setup operation, and so called *family setup times* (FST) are involved. The job partitioning into families implicates two next situations:

- the setup operation arises only when a machine shifts from processing a job in one family to processing a job in another family;
- a job containing several identical items may be split into multiple sublots and the setup operation arises only when a machine shifts from processing the sublot of one job to processing a sublot of another job.

  In general, these FST scheduling problems require two interrelated decisions:
- the size and number of the sublots of each family where the items of a single sublot are processed together;
- the scheduling of each sublot through the HFS where each sublot requires setup on each machine. According to the GT assumption, a family does not split into sublots, and the jobs of the same family are processed together. It refers to so called HFS with *group setup time* (GST) problem. However, if the families are split, it requires a solution of an interrelated *batching* problem to find the optimal size of each sublot. It refers to a so called *HFS with batch setup times* (BST) problems.

Since the BST problems require the solution of two interrelated problems (that of batching and scheduling), these problems are relatively harder to solve than their corresponding GST problems requiring only the solution of a scheduling problem (Monma & Potts, 1989).

The setup times in the Sequence-Independent Family Setup Times problem could be either attached or detached. Since each sublot in case of BST (or family in case of GST problem) consists of multiple jobs, the sequence-independent setup time cannot be added to the processing time of any one of these jobs, as the first job in the sequence of the sublot or batch is not known until the scheduling problem is solved. However, for the sequence-dependent BST and GST problems, the sequence-dependent sublot or batch setup times are only of the detached type.

Batch scheduling problem with setup times arises frequently in process industries, parts manufacturing environments and cellular assembly systems (such as chemical, pharmaceutical, food processing, metal processing, printing industries and semiconductor testing facilities). Detailed surveys of the recent publications about HFS with setup times might be consulted in (Ribas et al., 2010), (Ruiz, 2010), (Allajverdi, et al., 2008), (Zandieh et al., 2006).

5. HFS with sequence-dependent setup times

5.1 Investigated problems

In recent years, many researchers put attention to the HFS problem with sequence-dependent setup times in consequence of its complexity, the variety of models as realistic as theoretical, and used tools to the algorithm creation. On Table 1 are summarized publications dedicated to the investigation of the HFS with sequence-dependent setup times. The first column indicates the year of publication, the second is the bibliographical reference, the third describes the problem; and the final column shows the resolution method, type of approach as well as other case details.

Year	Author	Problem	Comments
1991	Guinet	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd} \mid \{C_{\max}, \overline{T}\}$	ad-hoc heuristics, textile industry
1993	Adler et al.	$FMm,((RM^{(k)})_{k=1}^{m}) \mid S_{ad} \mid \overline{T}^{w}$	DR, packging industry
	Voss	$FHm,((PM^{(1)},1^{(2)}) \mid S_{sd} \mid C_{\max}$	TS and heuristics
1995	Aghezzaf et al.	$FHm,((RM^{(k)})_{k=1}^{m}) \mid S_{sd} \mid \{C_{\max}, F_{\max}, \overline{F}\}$	MPF, heuristics, carpet manufacturing
1997	Li	$FH2,((1^{(1)},PM^{(2)})) \mid batch, S_{sd}, split \mid C_{\max}$	heuristics, major and minor setups, airplane engine plant
2000	Liu & Chang	$FHm,((PM^{(k)})_{k=1}^{m})) \mid S_{sd}, block \mid \overline{E}^{w} + \overline{T}^{w}$	MPF based heuristics
2003	Kurz & Askin	$FHm,((PM^{(k)})_{k=1}^{m})) \mid S_{sd} \mid C_{\max}$	heuristics
	Lin & Liao	$FHm,((PM^{(k)})_{k=1}^{2})) \mid S_{sd}^{(1)}, M_{j}^{(2)} \mid wT_{\max}$	heuristic, label sticker manufacturing
2004	Kurz & Askin	$FHm,((PM^{(k)})_{k=1}^{m})) \mid S_{sd} \mid C_{\max}$	MPF, MPR-GA

2005	Andres et al.	$FH3,((PM^{(k)})_{k=1}^{3}) \mid S_{sd} \mid other$	MPF, GT
	Pearn et al.	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd} \mid \overline{T}$	MPF, heuristics, packaging industry
	Tang et al.	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd} \mid C_{\max}$	NN
2006	Ruiz & Maroto	$FHm,((RM^{(k)})_{k=1}^{m}) \mid S_{sd}, M_j \mid C_{\max}$	MPR-GA
	Zandieh	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd} \mid C_{\max}$	Artificial Immune System
2007	Chen et al.	$FH3,((RM^{(k)})_{k=1}^{3}) \mid S_{sd}, block, prec \mid C_{\max}$	MPF, lower bounds, TS, container terminal
	Voss & Witt	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd} \mid \overline{T}^{w}$	MPF, DR, heuristics. Multi-project RCPSP, steel manufacturing
2008	Jungwattanakit et al.	$FHm,((RM^{(k)})_{k=1}^{m}) \mid S_{sd}, r_j \mid \alpha C_{\max} + (1-\alpha)\overline{U}$	MPF, heuristics, GA, SA, TS
	Ruiz et al.	$FHm,((RM^{(k)})_{k=1}^{m}) \mid skip, rm, lag, S_{sd}, M_j, prec \mid C_{\max}$	MPF, heuristics
2009	Jungwattanakit et al.	$FHm,((RM^{(k)})_{k=1}^{m}) \mid S_{sd}, r_j \mid \alpha C_{\max} + (1-\alpha)\overline{U}$	MPF, heuristics, DR, GA
	Naderi et al., a	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd}, transport \mid \{\overline{F}, \overline{T}\}$	SA
	Yaurima et al.	$FHm,((RM^{(k)})_{k=1}^{m}) \mid S_{sd}, M_j, buffer \mid C_{\max}$	MPR-GA, electronic indusry
	Naderi et al., b	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd} \mid \{\overline{F}, \overline{T}\}$	SA
	Alfieri	$FHm,((PM^{(k)})_{k=1}^{m}) \mid S_{sd}, reentry, batch \mid several$	simulation, heuristics, TS

Table 1. Investigated problems of HFS with sequence-dependent setup times.

The table shows that in recent years the publications are dedicated to more complex models of the problem, with unrelated parallel machine environment, release times, limited buffers, lags, and machine eligibility among others. A general framework to solve the problem includes: dispatching rules (DR), neural networks (NN), tabu search (TS), multiple permutation representation (MPR), local search (LS), simulated annealing (SA), genetic algorithms (GA). Mathematical programming formulation (MPF) is developed for many models.

The procedures to seek a solution of a HFS problem can be classified into two principal categories (Quadt & Kuhn, 2007): *optimal procedures* and *heuristics*. The literature does not

report application of any optimal procedure, like Dynamic Programming or Branch & Bound methods, to solution of a HFS problem with sequence-dependent setup times. Heuristics do not necessarily find an optimal solution. However, they are usually faster than optimal procedures and some of them are used for realistic problem sizes. Heuristics may be split into *holistic* and *decomposition* approaches. Holistic approaches consider the complete scheduling problem in an integrated way. A simple holistic approach is to use dispatching rules to select the next job that has to be produced whenever a machine becomes idle. The use of such heuristics is very common in HFS, see, e.g., (Adler et al., 1993), (Voss & Witt, 2007), (Jungwattanakit et al., 2009).
Most holistic procedures are local search methods or metaheuristics. In HFS, a job consists of several operations, one for each production stage. Thus, a HFS schedule assigns machine for each operation as well as a production sequence for each machine. A move to a neighboring schedule may change the machine assignment of an operation or its position in the sequence. This neighborhood is very large. Hence, local search procedures and metaheuristics must find ways to limit the size of the neighborhood. This can be done by only allowing certain moves that appear promising. Examples of metaheuristic techniques application are given in (Tang et al., 2005), (Chen et al., 2007), (Naderi et al., 2009), (Yaurima et al., 2009), among others.

In contrast to holistic approaches, decomposition approaches divide the problem into segments that are considered consecutively, with respect to the production stages, the individual jobs, or the sub-problems to be solved (batching, loading, and sequencing), e.g., (Li, 1997), (Alfieri, 2009).

The HFS scheduling problems with sequence-dependent setup times are among the most difficult classes of scheduling problems. When a practical problem of large instance sizes does not require of a fast result obtain, a good approximate solutions are achieved through a genetic algorithm (GA).

5.2 GA approach

A GA is a well known search technique used to find solutions to optimization problems. It was proposed by Holland (1975). All GA act according to the scheme represented on Figure 3. Candidate solutions are encoded by chromosomes (also called genomes or individuals). The set of initial individuals forms the population. Fitness values are defined over the individuals and measure the quality of the represented solution. The genomes are evolved through the genetic operators generation by generation to find optimal or near-optimal solutions. Three genetic operators are repeatedly applied: selection, crossover, and mutation. The selection picks chromosomes to mate and produce offspring. The crossover combines two selected chromosomes to create next generation of chromosomes. The mutation randomly reorganizes the structure of genes in a chromosome, so that a new combination of genes may appear in the next generation. The individuals evolve until some stopping criterion is met.

The first paper was by Ruiz and Maroto (2006) where application of GA techniques to HFS problem with sequence-dependent setup times had been realized. There were considered the makespan minimization criterion on a m-stage problem with unrelated parallel machines, sequence-dependent setup times and machine eligibility. The proposed GA was

superior to

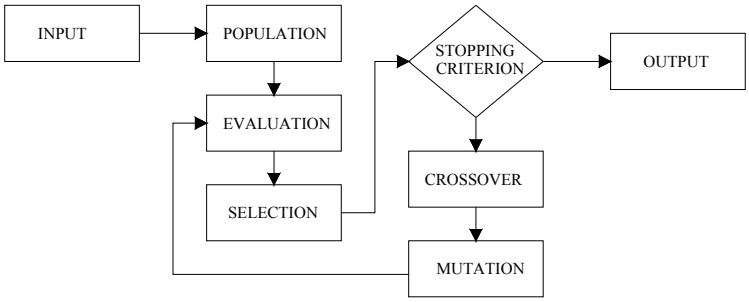

Fig. 3. GA scheme

a wide range of heuristics and other metaheuristics, among them, ACO based heuristics, TS procedures, other GAs, SA and deterministic procedures. A similar problem, with unrelated parallel machines at each stage, and setup times, was approached in (Jungwattanakit, et al. 2008) using GAs and later in (Jungwattanakit, et al., 2009) applying several heuristics including DR, GA, tailored heuristics, TS and SA. In these two papers, the authors study a linear combination of the makespan and a number of tardy jobs as an objective. Recently, in (Yaurima, et al., 2009) is proposed a GA for a complex HFS problem considering makespan minimization on an *m*-stage problem with sequence-dependent setup times, unrelated parallel machines, machine eligibility and limited buffers.

5.3 Crossover operators

The crossover operator is an important factor for a good performance of the GA. The following crossover operators should be considered to solve the examined problem according to previous investigations of authors.

1. OBX - *Order Based Crossover* (Gen, 1997), (Figure 4).

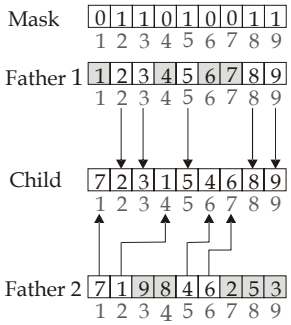

Fig. 4. OBX Crossover

This operator is based on a binary mask. The values of the mask equal to one indicate that the corresponding sequence of elements from parent 1 to child, is copied. The remaining

elements from parent 2 are copied. The mask values are generated randomly and uniformly in all crossover operations.

2. PPX - *Precedence Preservative Crossover* (Bierwirth, et al., 1996), (Figure 5).

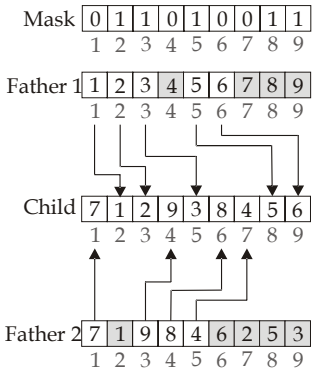

Fig. 5. PPX Crossover

This operator is based on a binary mask. The values of the mask equal to 1 indicate that corresponding elements from parent 1 are copied to child and values equal to 0 indicates that the elements are copied from parent 2, according to each value of the mask one at a time alternately.

3. OSX - *One Segment Crossover* (Guinet & Salomon, 1996), (Figure 6).

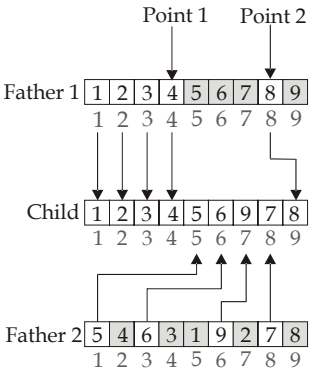

Fig. 6. OSX Crossover

Two points are randomly chosen. The elements from parent 1 since position 1 to the first place are copied. Elements from parent 2 since first point to the second point are copied. Finally, the items from parent 1 since the second point to last position are copied, considering not copied elements.

4. TP - *Two Point* (Michalewicz, 1996), (Figure 7).

Two points are randomly chosen. The elements from parent 1 since first position to the first point and since second point to the last position are copied. The elements from parent 2 since first point to the second point are copied.

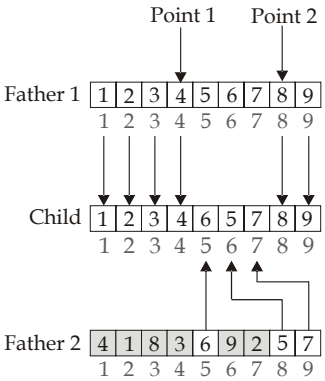

Fig. 7. TP Crossover

5. SB2OX - *Similar Block 2-Point Order Crossover* (Ruiz & Maroto, 2006), (Figure 8).

The common blocks in both parents (at least two consecutive identical jobs) are copied to the children, then two random cut points are defined and the section between these two points directly copied to children. The missing elements of each offspring are copied in the relative order of the other parent.

6. ST2PX - *Setup Time Two Point Crossover* (Yaurima, et al., 2009), (Figure 9).

In this crossover operator the sequence-dependent setup time is considered. Two points randomly in the sequence are chosen. The elements since first position to the first point and since second point to the last position, are copied from parent 1. The elements since first point to the second point are copied from parent 2 according to the minimum setup time of one machine randomly chosen from the first stage.

5.4 A problem of makespan minimizing in a HFS with multiple constrains

A complex problem of makespan minimizing in a HFS with sequence-dependent setup times, unrelated machines, availability constraints and limited buffers is presented. The real case of the television production environment is considered (Yaurima, et al., 2009).

Different television models are distinguished by their set of PCBs. The monthly production plan is developed based on current requirements, machines availability and resource constrains. It is updated daily depending on the final section requirements. It is examined the auto-Insertion section, where various PCB types are manufactured with automated machines for 70 television models, 45 machines and production units of different brands are dealt with.

The auto-insertion section is represented by a HFS with six stages (operations) common for all PCB types. However, some PCBs do not require all six operations. Each stage consists of several insertion machines in parallel, and they are dedicated to the certain types of component processing. At each instant of time, each machine works on at most one PCB,

and each PCB is processed by at most one machine. The PCBs are moving along the assembly line, from one machine to another until it became a complete unit.

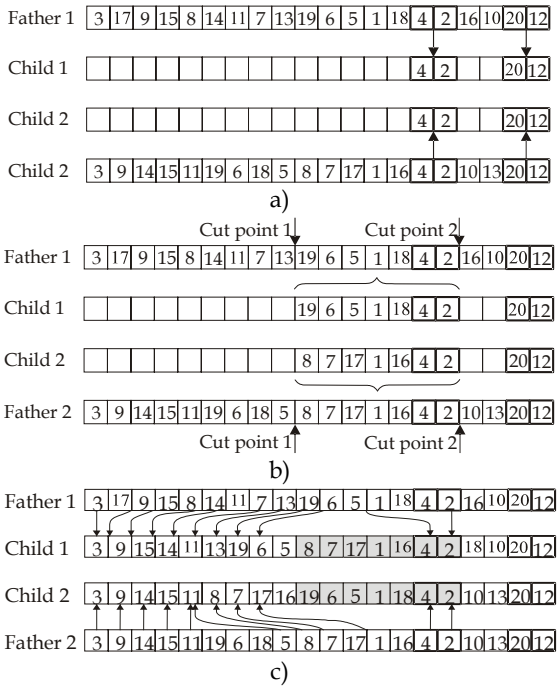

Fig. 8. SB2OX Crossover:
a) the common jobs in both parents are copied over to the offspring;
b) jobs before a randomly chosen cut point are inherited from the direct parent;
c) the missing elements in the offspring are copied in the relative order of the other parent.

The flow is determined by technological constraints. Machines of different brands with identical functionality but with different speeds or capabilities are included in the stage. The processing time depends on the machine brand. It is considered scheduling in the presence of machine eligibility restrictions when not all machines can process all PCBs, and machine availability restrictions when the use of machines depends on their current state: active or in maintenance service. Adjustment of the machine and the preparation of its feeder are required when the board type is changed. The feeders have different capacities (number of slots). For example, machines could have 60 slots or 80 slots. The time needed for adjustment essentially depends on the board type previously processed in the machine. It cannot be neglected in the television PCB production environment. Hence, a sequence-dependent setup time is needed. Each machine has a limited capacity buffer for storing WIP. If the storage is filled to full capacity, the production on this machine is blocked.

The problem is modeled as a HFS with the following constraints: (1) From two to six successive stages with the common flow pattern for all PCB types; (2) Stages with unrelated machines; (3) Machine eligibility/availability; (4) Sequence-dependent setup time; (5) Limited buffers. The goal is to find a schedule that minimizes the total production time.

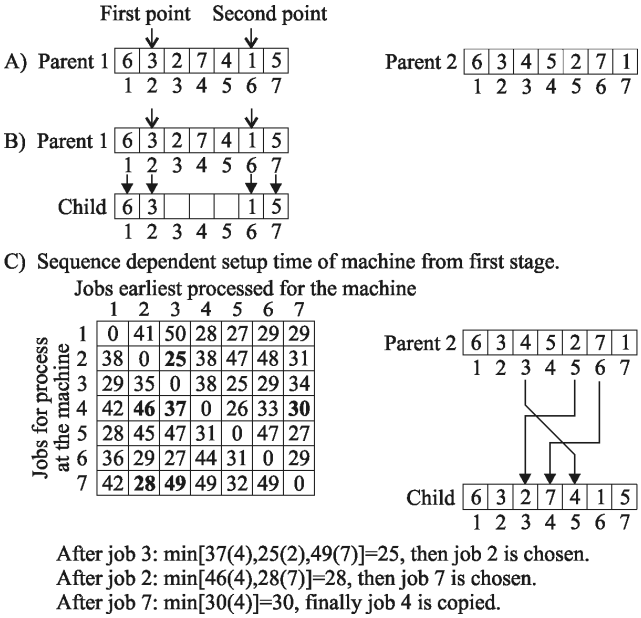

Fig. 9. ST2PX Crossover

The problem is denoted as $FHm,((RM^{(i)})_{i=1}^{(m)} \mid S_{sd}, M_j, Block \mid C_{\max}$. The next is the problem statement: Let a set N of n jobs, $N = \{1,2,...,n\}$ given at time 0 has to be processed in a set M of m consecutive production stages, $M = \{1,2,...,m\}$, without preemption. The objective to minimizing is total completion time known as makespan. On stage $i \in M$, a set $M_i = \{1,2,...,m_i\}$ of unrelated parallel machines is given, where $|M_i| \geq 1$.

Each job has to be processed by exactly one machine at each stage. Let $p_{i,l,j}$ be the processing time of job $j \in N$, on machine $l \in M_i$, at stage i. A machine based sequence-dependent setup time is considered. Let $S_{i,l,j,k}$ be the setup time on machine l, at stage i, when processing job $k \in N$, after processing job j. A set of eligible machines that can process job j at stage i, is denoted as $E_{i,j}$, $1 \leq |E_{ij}| \leq m_i$. For each machine $l \in M_i$ a limited buffer for jobs is given. A maximal storage capacity in front of each machine l is $b_{i,l}$, $1 \leq | b_{i,l} | \leq n$.

Many authors separate sequencing and assignment decisions in the HFS problems. To solve this problem, a way proposed by Ruiz and Maroto (2006) is used, where the assignment of jobs to machines in each stage is done by a evaluation function. In the HFS with no setup times and no availability constraint assignment of the job to the first available machine would result in the earliest completion time of the job. In the HFS with unrelated parallel machines it is demonstrated that if the first available machine is very slow for a given job, assigning the job to this machine can result in a later completion time compared with assignment to other machines. With the consideration of the setup times this problem

becomes worse. To solve it, in our algorithm, a job is assigned to the machine that can finish the job at the earliest time at a given stage, taking into consideration different processing speeds, setup times, machine availability, and buffer size.

The calculation of the total completion time $C_{\max}$ is as follow: Let π be a job permutation or sequence; $\pi_{(j)}$ be the job at the *j*th position in the sequence, $j \in N$. Each job has to be processed at each stage, so *m* tasks per job are considered. Let L_{i_l} be the last job assigned to machine l at stage i, $l \in M_i$. Let $S_{i_l,L_{i_l},\pi_{(j)}}$ be the setup time of machine l at stage i when processing of job $\pi_{(j)}$ after having processed the previous work assigned to this machine $l(L_{i_l})$. Let $C_{i,\pi_{(j)}}$ be the completion time of job $\pi_{(j)}$ at stage i, $i \in M$, then

$$C_{i,\pi_{(j)}} = \min_{l=1}^{m_j}\{\max\{C_{i,L_{i_l}} + S_{i_l,L_{i_l},\pi_{(j)}}; C_{i-1,\pi_{(j)}}\} + p_{i_l,\pi_{(j)}}\}$$

The makespan is calculated as follows:

$$C_{\max} = \max_{j=1}^{n}\{C_{m,\pi_{(j)}}\}$$

The GA, was tuned up by the following parameters elected in the parameter calibration step: crossover ST2PX; mutation Swap; crossover probability 0.8; mutation probability 0.1; population size 200. The execution steps of this algorithm (GA_{SBC}) are presented below.

Algorithm. GA_{SBC}.

Input: The population of P_{size} individuals.
Output: An individual of length *n*.

```
01. generate_population
02. regeneration = 1
03. while not stopping_criterion do
04.         for i=0 to P_size
05.                 evaluate_objective_function(i)
06.         keep_the_best_individual_found()
07.         if actual_best_makespan >= previous_best_makespan
08.                 iterations_without_improvement = iterations_without_improvement +1
09.                 if iterations_without_improvement = 25
10.                         if regeneration = 10
11.                                 stopping_criterion = true
12.                         else
13.                                 sort_the_population_in_ascending_order_of_Cmax()
14.                                 regenerate_population()
15.                                 regeneration = regeneration+1
16.                                 iterations_without_improvement = 0
17.         select_individuals_by_the_binary_tournament_selection
18.         crossover ST2PX with probability 0.8
19.         mutation SWAP with probability 0.1
```

5.5 Example

The following example illustrates this algorithm execution. Let is considered an instance with parameters n = 7, m = 3, $m_1 = m_2 = 2$, and m_3=1. Let Table 2 sets up eligibility, and Table 3 processing times. The number -1 means that the machine l is not eligible or not available for the job j. Table 4 shows sequence-dependent setup times of job k if job j precedes to job k**.** Table 5 shows the limited buffers sizes.

Job j	Stage i		
	1	2	3
1	{1}	{1}	{1}
2	{2}	{1,2}	{1}
3	{1,2}	{1,2}	{1}
4	{1,2}	{2}	{1}
5	{1,2}	{1,2}	{1}
6	{1}	{2}	{1}
7	{2}	{2}	{1}

Table 2. A set of eligible machines at stage i that can process job j

Stage i		1	1	2	2	3
Machine l		1	2	1	2	1
Job j	1	54	-1	69	-1	60
	2	-1	76	75	67	55
	3	58	93	51	82	75
	4	59	95	-1	52	88
	5	75	62	58	73	93
	6	50	-1	-1	52	61
	7	-1	57	-1	66	93

Table 3.The processing time $p_{i,l,j}$ of job j, on machine l, at stage i.

Job k		1	2	3	4	5	6	7
Job j	1	0	41	50	28	27	29	29
	2	38	0	25	38	47	48	31
	3	29	35	0	38	25	29	34
	4	42	26	37	0	26	33	30
	5	28	45	47	31	0	47	27
	6	36	29	27	44	31	0	29
	7	42	28	49	49	32	49	0

Table 4. Sequence-dependent setup times for the first machine

Stage i	1	1	2	2	3
Machine l	1	2	1	2	1
Buffer $b_{i,l}$	2	2	3	2	3

Table 5. Limited buffers

Let a population with 10 individuals is generated (Figure 10). Figure 11 presents the fitness value of each individual. The best solution is represented by the individual 2 with makespan 817. The population is ordered and regenerated: 20% best individuals are kept, 40% are replaced by simple Insert mutation of the best individual, and reminding worst 40% are replaced by randomly generated individuals. Figure 12 shows the regeneration result.

Figure 13 shows result of the binary selection. The ST2PX crossover is applied with probability 0.8 (Figure 14). Let is assumed that the first point is at position 2, and the second point is at position 6 (Fig. 14A). Elements from position 1 to position 2 of parent 1 are copied

to the child. Elements from position 6 to position 7 (last position) are copied from parent 1 (Fig. 14B). The remaining positions of the child are filled with best elements from parent 2, taking into account the sequence-dependent setup times (Fig. 14C). Three jobs (4, 2 and 7) can be processed at position 3 after processing job 3 at position 2. Hence, three setup times (37, 25, 49) are compared, and job 2 with minimal setup time 25 is chosen. Two setup times (46 and 28) are compared for position 4, and job 7 is chosen. The last job (4) is copied to position 5. Finally, the SWAP mutation is applied with probability 0.1 (Fig. 15). Fig. 16 shows the Gantt chart of the final result.

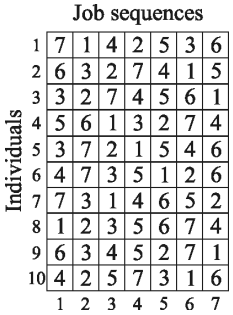

Fig. 10. Initial population

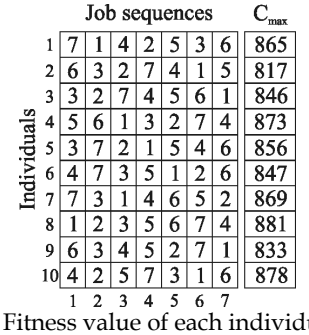

Fig. 11. Fitness value of each individual

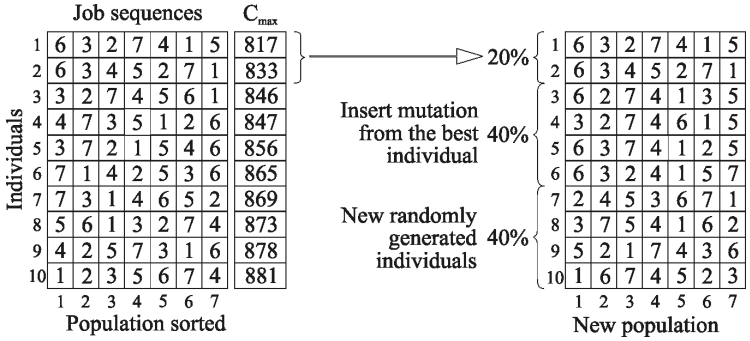

Fig. 12. Regeneration procedure

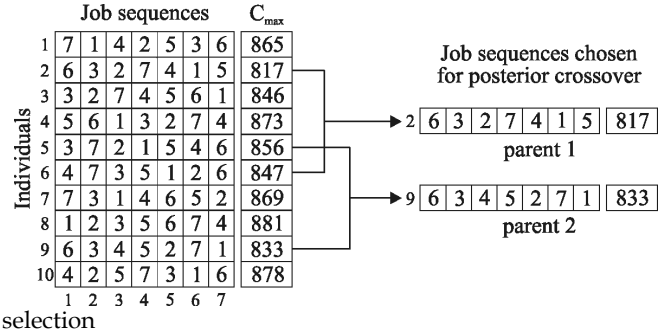

Fig. 13. Binary selection

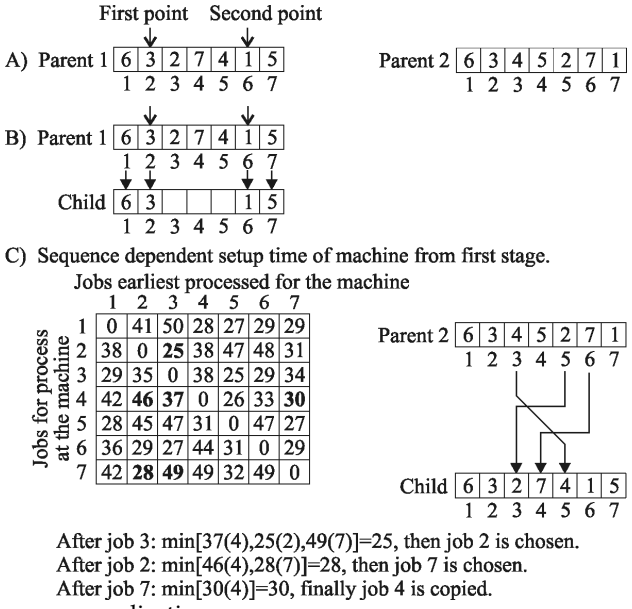

Fig. 14. ST2PX crossover application

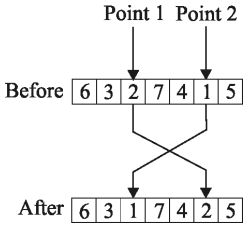

Fig. 15. SWAP mutation

6. Conclusion

There are several applications of the HFS scheduling problems which consider setup times in industry, and the variety of models as realistic as theoretical is practically innumerable; then this field of study will attract always the researcher attention. The hardest situation involving setup times is HFS problem with sequence-dependent setup times. It is among the most difficult classes of scheduling problems. Due the complexity, artificial intelligence and metaheuristic techniques should be used for practical problems with multistage parallel machine environment and large instance sizes, in particularity, evolutionary algorithms.
Actually, the authors are exploring a mixed model which consist of a HFS combined with a number of assemble lines. There are considered setup times of machines. The problem involves splitting of lots. Its solution consumes all topics exposed in this chapter.

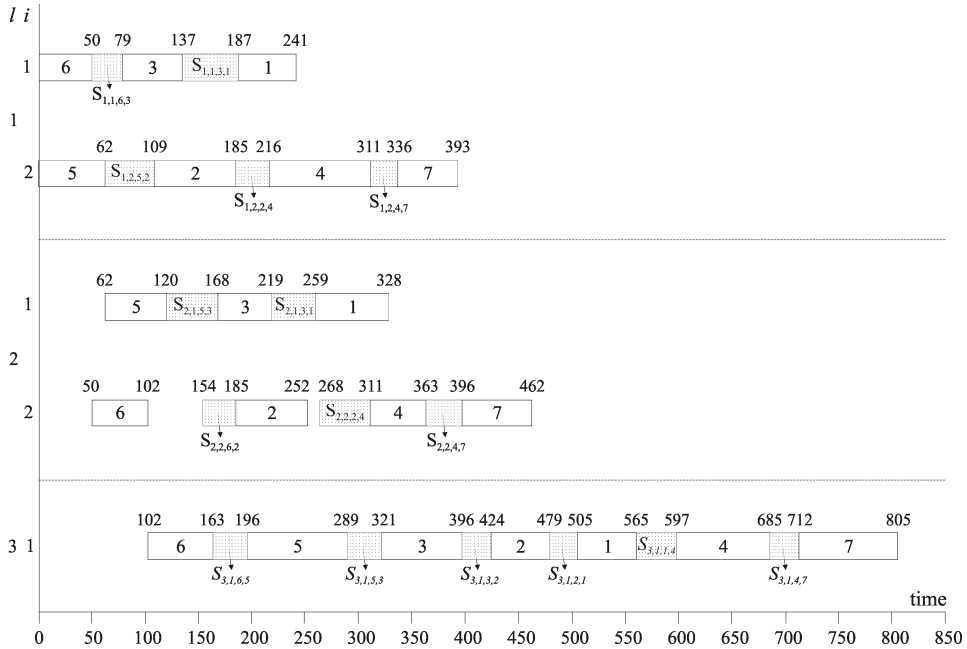

Fig. 16. Gantt chart for the problem solution (C_{max} = 805)

7. References

Adler, L.; Fraiman, N.; Kobacker, E.; Pinedo, M.; Plotnicoff, J.C. & Wu, T.P. (1993). Bpss: a scheduling support system for the packaging industry. *Operations Research,* Vol. 41, No. 4, (July-August 1993) 641–648, ISSN 0030-364X

Aghezzaf, E.-H.; Artiba, A.; Moursli, O. & Tahon, C. (1995). Hybrid flowshop problems, a decomposition based heuristic approach, *Proceedings of the International Conference on Industrial Engineering and Production Management*, IEPM'95, FUCAM-INRIA. pp. 43–56.

Agnetis, A.; Pacifici, A.; Rossi, F.; Lucertini, M.; Nicoletti, S.; Nicolo, F.; Oriolo, G.; Pacciarelli, D. & Pesaro, E. (1997). Scheduling of flexible flow lines in an automobile assembly plant. *European Journal of Operational Research.* Vol. 97, No. 2, (March 1997) 348–362, ISSN 0377-2217

Alfieri, A. (2009). Workload simulation and optimisation in multi-criteria hybrid flowshop scheduling: a case study. *International Journal of Production Research.* Vol. 47, No. 18, (January 2009) 5129– 5145, ISSN 0020-7543.

Allahverdi, A.; Ng, C. T.; Cheng, T. C. E.& Kovalyov, M. Y. (2008) A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187, No. 3, (June 2008) 985–1032, ISSN 0377-2217

Andres, C.; Albarracin, JM.; Tormo, G.; Vicens, E. & Garcia-Sabater, JP. (2005). Group technology in a hybrid flowshop environment: a case study. *European Journal of Operational Research.* vol. 167. No. 1, (November 2005) 272–81, ISSN 0377-2217

Arthanary, L. & Ramaswamy, K. (1971). An extension of two machine sequencing problem. *Journal of the Operational Research Society of India*,Vol. 8. No. 4. 10–22.

Bierwirth, C.; Mattfeld, D. & Kopfer, H. (1996). On permutation representations for scheduling problems. Parallel Problem Solving from Nature - PPSN IV, Vol. 1141. pp. 310-318. LNCS, Springer. ISBN 978-3-540-61723-5, Berlin

Botta-Genoulaz, V. (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. *International Journal of Production Economics*. Vol. 64. No. 1-3, (March 2000) 101–111, ISSN 0925-5273

Brah, S. A. & Hunsucker, J. L. (1991). Branch and bound algorithm for the flow shop with multiple processors. *European Journal of Operational Research*. Vol. 51. No. 1, (March 1991) 88-99, ISSN 0377-2217

Chen, B. (1995). Analysis of classes of heuristics for scheduling a two-stage flow-shop with parallel machines at one stage. *Journal of the Operational Research Society*. Vol. 46. No. 2. (February 1995) 234–244, ISSN 0160-5682

Chen, L.; Bostel, N.; Dejax, P.; Cai, J.C.; Xi, L.F. (2007). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal, *European Journal of Operational Research.* Vol. 181, No. 1, (August 2007) 40–58, ISSN 0377-2217

Cheng, T. C. E.; Gupta, J. N. D. & Wang, G. (2000). A review of flowshop scheduling research with setup times. *Production and Operations Management Society*. Vol. 9, No. 3, (September 2000) 262-282, ISSN 1059-1478

Crama, Y. (1997). Combinatorial Optimization Models for Production Scheduling in Automated Manufacturing Systems. *European Journal of Operational Research*. Vol. 99, No. 1 (May 1997) 136–153, ISSN 0377-2217

Gen, M. & Cheng, R. (1997). Genetic algorithms & engineering optimization. *John Wiley & Sons*, ISBN 0-471-31531-1, NY

Glover, F. & Laguna, M. (1997). Tabu search. *Kluwer Academic Publishers*, ISBN:079239965X Boston

Guinet, A. (1991). Textile production systems: a succession of non-identical parallel processor shops. *Journal of the Operational Research Society.* Vol. 42, No. 8, (August 1991) 655–671, ISSN 0160-5682

Guinet, A., & Solomon, M. M. (1996). Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. *International Journal of Production Research*. Vol. 34, No. 6, (June 1996) 1643-1654, ISSN 0020-7543

Gupta, J.N.D. (1988), Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society.* Vol. 39, No. 4, (April 1988) 359-364, ISSN 0160-5682

Gupta, J. N. D. & Tunc, E. A. (1994). Scheduling a two-stage hybrid flowshop with separable setup and removal times. *European Journal of Operational Research*. Vol. 77, No.3, (September 1994) 415–428, ISSN 0377-2217

Gupta, J.N.D.; Strusevich, V.A. & Zwaneveld, C. (1997). Two-stage no-wait scheduling models with set-up and removal times. *Computers & Operations Research.* Vol. 24. No. 11, (November 1997) 1025–1031, ISSN 0305-0548

Harjunkoski, I. & Grossmann, I.E. (2002). Decomposition techniques for multistage scheduling problems using mixed integer and constraint programming methods. *Computers & Chemical Engineering*. Vol. 26, No. 11, (November 2002) 1533–1552, ISSN 0098-1354

Holland, J. H. (1992). Adaptation in Natural and Artificial Systems. *Ann MIT Press*, 0-262-08213-6, Boston

Jungwattanakit, J.; Reodecha, M.; Chaovalitwongse, P.; & Werner, F. (2008). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *International Journal of Advanced Manufacturing Technology*. Vol. 37, No. 3–4 (May 2008) 354–370, ISSN 0268-3768

Jungwattanakit, J.; Reodecha, M.; Chaovalitwongse, P.; & Werner, F. (2009). A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research*. Vol. 36, No. 2, (Febrary 2009) 358–378, ISSN 0305-0548

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 Science*. Vol. 25, No. 5, (October 1997) 547–555, ISSN 0305-0483

Kurz, M.E.; Askin, R.G. (2003). Comparing scheduling rules for flexible flow lines. *International Journal of Production Economics.* Vol. 85, No. 3, (November 2003) 371–388, ISSN 0925-5273

Kurz, M.E.; Askin, R.G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research.* Vol. 159, No. 1, (November 2004) 66–82, ISSN 0377-2217

Leon, V.J. & Ramamoorthy, B. (1997). An adaptable problem-space-based search method for flexible flow line scheduling. *IIE Transactions*. Vol. 29, No. 2, (February 1997) 115–125, ISSN 0740-817X

Li, S. (1997). A hybrid two-stage flowshop with part family, batch production, major and minor setups. *European Journal of Operations Research.* Vol. 102. No. 1, (Octubre 1997) 142–156, ISSN 0377-2217

Lin, H.-T. & Liao, C.-J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*. Vol. 86, No. 2, (November 2003) 133–143, ISSN 0925-5273

Liu, Ch.-Y. and Chang, Sh.-Ch. (2000). Scheduling Flexible Flow Shops with Sequence-Dependent Setup Effects. *IEEE Transactions on Robotics and Automation.* Vol. 16, No. 4, (August 2000) 408-419, ISSN 1042-296X

Low, Ch. (2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. *Computers & Operations Research*. Vol. 32, No 8, (August 2005) 2013–2025, ISSN 0305-0548

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

Michalewicz, Z. (1996). Genetic *Algorithms* + *Data Structures* = *Evolution Programs*. Springer-Verlag, ISBN 3-540-606776-9, Berlin

Monma, C. L. & Potts, C. N. (1989). On the Complexity of Scheduling with Batch Setups. *Operations Research*. Vol. 37, No. 5. (September 1989) 798–804, ISSN 0030-364X

Moursli, O. & Pochet, Y. (2000). A branch-and-bound algorithm for the hybrid flowshop. *International Journal of Production Economics*. Vol. 64. (Mach 2000)113–125, ISSN 0925-5273

Naderi, B.; Zandieh, M.; Khaleghi, A.; Ghoshe Balagh & Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Systems with Applications.* Vol. 36. No. 6, (August 2009) 9625–9633, ISSN 0957-4174

Naderi, B.; Zandieh, M.; Roshanaei, V. (2009). Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness, *International Journal of Advanced Manufacturing Technology*. Vol. 41, No. 11–12, (April 2009). 1186–1198, ISSN 0268-3768

Pearn, W. L.; Chung, S. H.; Yang, M. H. & Chen, C. Y.(2005). The integrated circuit packaging scheduling problem (icpsp): A case study. *International Journal of Industrial Engineering-Theory Applications and Practice*. Vol. 12. No. 3. 296–307, ISSN 1943-670X

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

Quadt, D. & Kuhn, D. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research.* Vol. 178, No. 3 (May 2007) 686–698, ISSN 0377-2217

Ribas, I.; Leisten, R. J. & Framiñan, M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*. Vol. 37, No. 8, (August 2010) 1439–1454, ISSN 0305-0548

Ruiz, R. & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*. Vol. 169, No. 3 (March 2006) 781–800, ISSN 0377-2217

Ruiz, R.; Sivrikaya, F.; Şerifoğlu, T. U. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, Vol. 35, No. 4, (April 2008) 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) 1–18, ISSN 0377-2217

Tang, L.X.; Liu, W.X. & Liu, J.Y. (2005). A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment. *Journal of Intelligent Manufacturing*. Vol. 16, No. 3, (June 2005) 361–370, ISSN: 0956-5515

Vairaktarakis, G. (2004). Flexible Hybrid Flowshop, *In: Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. J. Y.-T. Leung, 5-1 – 5-33, ISBN: 1-58488-397-9, Boca Raton, Florida.

Voss, S. & Witt, A. (2007). Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. *International Journal of Production Economics*. Vol. 105, No. 2, (Febrary 2007) 445–458, ISSN 0925-5273

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) 1452-1463, ISSN 0360-8352

Zandieh, M.; Ghomi, S.M.T. Fatemi & Husseini, S.M. Moattar. (2006) An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times, *Applied Mathematics and Computation.* Vol. 180, No. 1, (September 2006) 111–127, ISSN 0096-3003

Future Manufacturing Systems

Edited by Tauseef Aized

ISBN 978-953-307-128-2
Hard cover, 268 pages
Publisher Sciyo
Published online 17, August, 2010
Published in print edition August, 2010

This book is a collection of articles aimed at finding new ways of manufacturing systems developments. The articles included in this volume comprise of current and new directions of manufacturing systems which I believe can lead to the development of more comprehensive and efficient future manufacturing systems. People from diverse background like academia, industry, research and others can take advantage of this volume and can shape future directions of manufacturing systems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Larysa Burtseva, Victor Yaurima and Rainier Romero Parra (2010). Scheduling Methods for Hybrid Flow Shops with Setup Times, Future Manufacturing Systems, Tauseef Aized (Ed.), ISBN: 978-953-307-128-2, InTech, Available from: http://www.intechopen.com/books/future-manufacturing-systems/scheduling-methods-for-hybrid-flow-shops-with-setup-times

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