Part One

Short-Term Scheduling in Supply Chains

Chapter 1

Scheduling of Flexible Flow Shops

1.1 INTRODUCTION

This chapter deals with mixed integer programming (MIP) models for scheduling of flow shops, the design in which dedicated machines are arranged in series or in series and parallel, and in which a transportation system imposes a unidirectional flow of parts, with revisiting of machines not allowed. The serial configuration of machines is widely used in many industries, in particular, the serial/parallel configuration with several processing stages in series and one or more parallel machines in each stage, is common in a high-tech industry.

The proposed MIP models cover a wide range of flow shop configurations that can be encountered in modern supply chains. They include flow shops with single or with parallel machines, with infinite, finite, or no in-process buffers, with machines continuously available or machines with one or more intervals of unavailability. The following models are presented:

F1 for scheduling flow shops with single machines and infinite in-process buffers

FP for scheduling flow shops with parallel machines and infinite in-process buffers

F1B for scheduling flow shops with single machines and no in-process buffers

FPB for scheduling flow shops with parallel machines and finite in-process buffers

FPBD for scheduling flow shops with parallel machines, finite in-process buffers, and machine down times

All the above models consider makespan minimization as a main scheduling criterion that aims at reaching a high throughput of the flow shop. The models, however, can easily be enhanced for the other common criteria such as total completion time or maximum or total tardiness, if the due dates for some parts are given. The model enhancements are described in Section 1.2.3. The MIP models can also be easily

Scheduling in Supply Chains Using Mixed Integer Programming. By Tadeusz Sawik

enhanced for scheduling flow shops with nonnegligible transportation times between processing stages (Section 1.2.4) or for scheduling reentrant flow shops (Section 1.2.5), in which a part visits a set of stages more than once.

Finally, for a comparison with the proposed MIP models, two simple, constructive heuristics for scheduling flexible flow shops with finite or with no in-process buffers and with nonzero transportation times, are described in Section 1.3.

1.2 MIXED INTEGER PROGRAMS FOR SCHEDULING FLOW SHOPS

In this section basic MIP formulations are developed for scheduling flexible flow shops of different configuration.

1.2.1 Scheduling Flow Shops with Infinite In-Process Buffers

A regular flow shop consists of m machines in series with unlimited capacity buffers between the machines. In the line n parts of various types are processed (for notation used, see Table 1.1). Each part must be processed without preemption on each machine sequentially. That is, each part must be processed in stage 1 through stage m in that order. The order of processing the parts in every stage is identical and determined by an input sequence in which the parts enter the line, that is, a so-called permutation flow shop is considered (e.g., Baker and Trietsch, 2009).

Table 1.1 Notation: MIP Models for Scheduling Flexible Flow Shops

Indices

i = processing stage, $i \in I = \{1, \ldots, m\}$
j = processor in stage i , $j \in J_i = \{1, \ldots, m_i\}$
k = part, $k \in K = \{1, \ldots, n\}$

Input parameters

m = number of processing stages
m_i = $|J_i|$—number of parallel processors in stage i
n = number of parts
p_{ik} = processing time for part k in stage i
Q = a large positive constant not less than the schedule length

Decision variables

c_{ik} = completion time of part k in stage i (timing variable)
d_{ik} = departure time of part k from stage i (timing variable)
x_{ijk} = 1, if part k is assigned to processor $j \in J_i$ in stage $i \in I$; otherwise $x_{ijk} = 0$ (assignment variable)
y_{kl} = 1, if part k precedes part l in the processing sequence; otherwise $y_{kl} = 0$ (sequencing variable)

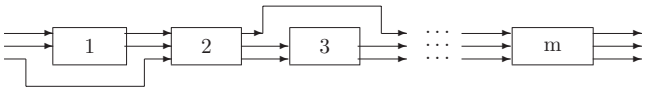

Figure 1.1 A general flow shop with single machines.

Let p_{ik} be the processing time on machine $i \in I$ of part $k \in K$. For every part k denote by c_{ik} its completion time in each stage i as well as its departure time from stage i.

Processing without preemption indicates that part k completed in stage i at time c_{ik} had started its processing in that stage at time $c_{ik} - p_{ik}$. Each part k completed in stage i at time c_{ik} immediately departs that stage for the next stage $i + 1$, if it is not occupied with another part; otherwise the part is transferred to the buffer with unlimited capacity between stages i and $i + 1$.

In contrast to the regular flow shop, in which each part requires m operations, each on a different machine, in a general flow shop (Fig. 1.1) parts may require fewer than m operations, not necessarily on adjacent machines. Then, the general case can be represented as a regular flow shop in which the processing times on some machines are zero.

The production schedule is specified by an input sequence in which the parts enter the line and are processed on each machine as well as by all completion times required for detailed scheduling of each individual part. The scheduling objective is to determine a permutation of parts such that all the parts are completed in a minimum time, that is, to minimize the makespan $C_{\max} = \max_{k \in K}(c_{mk})$, where c_{mk} denotes the completion time of part k in the last stage m.

The problem of scheduling a regular flow shop with single machines and infinite in-process buffers is formulated below as a mixed integer program *F1*.

Model F1: *Scheduling Flow Shops with Single Machines and Infinite In-Process Buffers*

Minimize

$$C_{\max} \tag{1.1}$$

subject to

1. *Part Completion Constraints*:

– each part must be processed on the first machine and successively on all downstream machines,

$$c_{1k} \geq p_{1k};\ k \in K \tag{1.2}$$

$$c_{ik} - c_{i-1k} \geq p_{ik};\ i \in I,\ k \in K: i > 1 \tag{1.3}$$

2. *Part Noninterference Constraints*:

– no two parts can be processed simultaneously on the same machine,

$$c_{ik} + Qy_{kl} \geq c_{il} + p_{ik};\ i \in I, k, l \in K: k < l \tag{1.4}$$

$$c_{il} + Q(1 - y_{kl}) \geq c_{ik} + p_{il};\ i \in I, k, l \in K: k < l \tag{1.5}$$

3. *Maximum Completion Time Constraints*:

– the schedule length is determined by the latest completion time of some part on the last machine,

$$c_{mk} \leq C_{\max};\ k \in K \tag{1.6}$$

4. *Variable Nonnegativity and Integrality Conditions*:

$$C_{\max} \geq 0 \tag{1.7}$$

$$c_{ik} \geq 0;\ i \in I, k \in K \tag{1.8}$$

$$y_{kl} \in \{0, 1\};\ k, l \in K: k < l. \tag{1.9}$$

The noninterference constraints (1.4) and (1.5) were constructed as follows. For any two parts k and l processed by the same machine i either part k precedes part l, and then processing of l cannot be started until processing of k is completed, or l precedes k, and then k cannot be started until l is completed. As a result, for all $i \in I$ and $k, l \in K: k \neq l$ a pair of the following disjunctive constraints must hold

$$c_{il} - p_{il} \geq c_{ik} \quad \text{or} \quad c_{ik} - p_{ik} \geq c_{il}.$$

These disjunctive constraints can be replaced with an equivalent pair of conjunctive constraints:

$$c_{il} + Q(1 - y_{kl}) \geq c_{ik} + p_{il};\ i \in I, k, l \in K: k < l$$

$$c_{ik} + Qy_{kl} \geq c_{il} + p_{ik};\ i \in I, k, l \in K: k < l$$

where y_{kl} is the binary sequencing variable defined below

$$y_{kl} = \begin{cases} 1, & \text{if part } k \text{ precedes part } l \text{ in the processing sequence} \\ 0, & \text{if part } l \text{ precedes part } k \text{ in the processing sequence,} \end{cases}$$

and Q is a large positive constant, not less than the schedule length $C_{\max}$.

Note that for all $k \neq l$, $y_{kl} + y_{lk} = 1$ or equivalently $y_{lk} = 1 - y_{kl}$, and hence in the above constraints it is sufficient to define y_{kl} for $k < l$.

In the next model, a flow shop with parallel machines is considered which consists of m processing stages in series with unlimited capacity buffers between the successive stages and each stage i, $(i = 1, \ldots, m)$ is made up of $m_i \geq 1$ parallel identical machines (Fig. 1.2).

The flow shop with parallel machines is also known as a hybrid flow shop, flow shop with multiple machines, flexible flow shop, flexible flow line, or multiprocessor flow shop. The problem of scheduling a flow shop with parallel machines may be seen

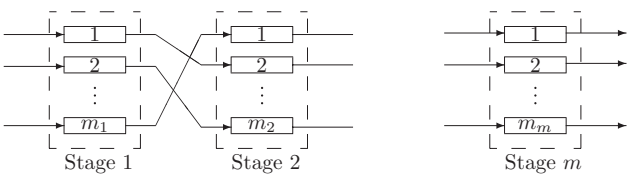

Figure 1.2 A flow shop with parallel processors.

as a combination of two particular types of scheduling problems: the parallel machine scheduling problem and the regular flow shop with single machines scheduling problem. The key decision of the problem of scheduling parallel machines is the assignment of parts to machines, whereas the key decision of scheduling a regular flow shop with single machines is the sequence of parts through the shop. Hence, the main decisions in the operation of the flow shop with parallel machines are to assign and schedule the parts to the machines in each stage, that is, to determine the order in which the parts are to be processed on the different machines of each stage.

In order to determine the assignment of parts to machines in each stage the following decision variables need to be added to the mixed integer programming formulations:

$$x_{ijk} = 1, \quad \text{if part } k \text{ is assigned to machine } j \in J_i \text{ in stage } i \in I;$$
$$\text{otherwise } x_{ijk} = 0.$$

The problem of scheduling a flow shop with parallel machines and infinite in-process buffers is formulated below as a mixed integer program *FP*.

Model FP: *Scheduling Flow Shops with Parallel Machines and Infinite In-Process Buffers*

Minimize (1.1) subject to

1. *Part Completion Constraints*: (1.2), (1.3)
2. *Maximum Completion Time Constraints*: (1.6)
3. *Machine Assignment Constraints*:
   - in every stage each part is assigned to exactly one machine,

$$\sum_{j \in J_i} x_{ijk} = 1; \ i \in I, k \in K \tag{1.10}$$

4. *Part Noninterference Constraints*:
   - no two parts assigned to the same machine can be processed simultaneously,

$$c_{ik} + Q(2 + y_{kl} - x_{ijk} - x_{ijl}) \geq c_{il} + p_{ik}; \ i \in I, j \in J_i, k, l \in K: k < l \tag{1.11}$$

$$c_{il} + Q(3 - y_{kl} - x_{ijk} - x_{ijl}) \geq c_{ik} + p_{il}; \ i \in I, j \in J_i, k, l \in K: k < l \tag{1.12}$$

5. *Variable Nonnegativity and Integrality Conditions*: (1.7) to (1.9) and

$$x_{ijk} \in \{0, 1\}; \ i \in I, j \in J_i, k \in K. \tag{1.13}$$

Part noninterference constraints (1.11) and (1.12) incorporate additional assignment variables x_{ijk} and x_{ijl}. For a given sequence of parts at most one constraint of (1.11) and (1.12) is active, and only if in stage i both parts k and l are assigned to the same machine j, i.e., only if $x_{ijk} = x_{ijl} = 1$ (which is always true for stages with

a single machine). Otherwise, both (1.11) and (1.12) are inactive. In other words, non-interference constraints (1.11) and (1.12) prevent simultaneous processing of any two parts assigned in some stage to the same machine. However, two parts assigned to different parallel machines in the same stage can be processed in parallel, and then both constraints (1.11) and (1.12) are inactive.

In the above problem setting, all machines within each stage are considered to be identical. Therefore, the processing time of a part on each stage does not depend on the specific machine to which it is assigned. The proposed MIP models are also capable of scheduling flow shops with unrelated parallel machines, in which the processing times of a job in a stage depend on each specific machine within this stage, that is, with the input parameters p_{ik} replaced with p_{ijk} for each machine $j \in J_i$ and some or all stages $i \in I$. This may be due to the differences between the machines themselves, to the fact that a certain type of machine is better suited for processing a particular part, whereas others are not, or because the parts have some special characteristics and can only be assigned to machines that better physically meet them. Also, flexible flow shops with parallel uniform machines in some or all stages (e.g., Kyparisis and Koulamas, 2006) can be considered, where each machine $j \in J_i$ has an associated speed v_j, that is, when part k is processed in stage i by machine j, it requires p_{ik} / v_j time units to be completed.

1.2.2 Scheduling Flow Shops with Finite In-Process Buffers

In this section mixed integer programming models for scheduling are presented first for a special case of the flow shop with single machines and no buffers and then for a general case of the flow shop with parallel machines and finite in-process buffers.

A unified modeling approach is adopted with the buffers and machines jointly called processors. The buffers are viewed as special processors with zero processing times but with blocking. As a result the scheduling problem with finite in-process buffers can be converted into one with no buffers but with blocking, see Figure 1.2.

The blocking time of a processor with zero processing time represents part waiting time in the buffer represented by that processor. We assume that each part must be processed in all stages, including the buffer stages. However, zero blocking time in a buffer stage indicates that the corresponding part does not need to wait in the buffer. Let us note that for each buffer stage part completion time is equal to its departure time from the previous stage since the processing time is zero.

A flow shop with single machines and no in-process buffers represents a special type of regular flow shop with no store constraints in which there is only one machine in each stage, and every part visits every machine.

Assume that the flow shop consists of m machines in series with no intermediate buffers between the machines. The system produces n parts of various types and each part must be processed without preemption in each stage 1 through stage m in that order. Let p_{ik} be the processing time in stage i of part k. For every part k denote by c_{ik} its completion time in each stage i. A part completed on a machine blocks this

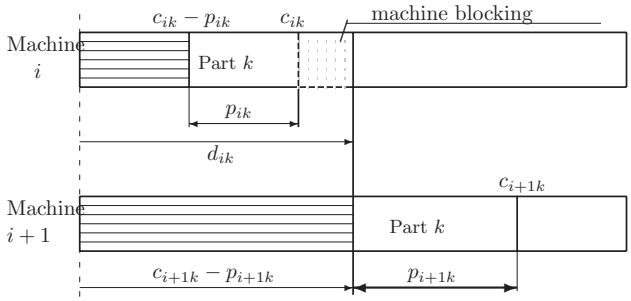

Figure 1.3 A partial schedule for part k in a flow shop with no in-process buffers.

machine until the next machine is available and denote by d_{ik} its departure time from stage i. This is a new variable that needs to be added to the previous models to account for the blocking scheduling that may occur when the intermediate buffers have limited capacity or there are no buffers at all.

A Gantt chart in Figure 1.3 shows a partial schedule for some part k in stages i and $i+1$. Processing without preemption indicates that part k completed in stage i at time c_{ik} had started its processing in that stage at time $c_{ik} - p_{ik}$. Part k completed in stage i at time c_{ik} departs at time $d_{ik} \geq c_{ik}$ to the next stage $i+1$. If at time c_{ik} machine $i+1$ is occupied, then machine i is blocked by part k until time $d_{ik} = c_{i+1k} - p_{i+1k}$ when part k can start processing on machine $i+1$.

The problem of scheduling the flow shops with no in-process buffers is formulated below as a mixed integer program *F1B*.

Model F1B: *Scheduling Flow Shops with Single Machines and Blocking (No In-Process Buffers)*

Minimize (1.1) subject to

1. *Part Completion Constraints*: (1.2), (1.3)
2. *Maximum Completion Time Constraints*: (1.6)
3. *Part Departure Constraints*:
   - each part cannot be departed from a machine until it is completed on this machine,
   - each part leaves the line as soon as it is completed on the last machine,

$$c_{ik} \leq d_{ik};\ i \in I, k \in K: i < m \tag{1.14}$$

$$c_{mk} = d_{mk};\ k \in K \tag{1.15}$$

4. *No Buffering Constraints*:
   - on every machine processing of each part starts immediately after its departure from the previous machine,

$$c_{ik} - p_{ik} = d_{i-1k};\ i \in I, k \in K: i > 1 \tag{1.16}$$

5. *Part Noninterference Constraints*:

– no two parts can be processed simultaneously on the same machine,

$$c_{ik} + Qy_{kl} \geq d_{il} + p_{ik};\ i \in I, k, l \in K: k < l \qquad (1.17)$$

$$c_{il} + Q(1 - y_{kl}) \geq d_{ik} + p_{il};\ i \in I, k, l \in K: k < l \qquad (1.18)$$

6. *Variable Nonnegativity and Integrality Conditions*: (1.7)–(1.9), and

$$d_{ik} \geq 0;\ i \in I, k \in K. \qquad (1.19)$$

Inequalities (1.17) and (1.18) represent disjunctive constraints for scheduling with machine blocking, that is, they express interrelations between completion and departure times of any two different parts on the same machine i. No two parts can be performed on the same machine simultaneously. For any two different parts k and l either part k precedes part l, and then l cannot be started until k is departed from machine i (i.e., $c_{il} - p_{il} \geq d_{ik}$), or part l precedes part k, and then k cannot be started until l is departed (i.e., $c_{ik} - p_{ik} \geq d_{il}$). For a given sequence of parts only one constraint of (1.17) and (1.18) is active.

Finally, the most general problem of scheduling flow shops with parallel machines and finite in-process buffers is considered. Notation used to formulate the problem is shown in Table 1.1, where buffers and machines are jointly called processors. In the flow shop with parallel processors and no in-process buffers shown in Figure 1.2, processors represent either machines or buffers. If stage i is a buffer stage, then m_i is the number of buffers in that stage, otherwise m_i is the number of machines. Note that for the buffer stages all processing times are zero.

A Gantt chart in Figure 1.4 shows a partial schedule for some part k in stages i and $i + 1$. Processing without preemption indicates that part k completed in stage i at time c_{ik} had started its processing in that stage at time $c_{ik} - p_{ik}$. Part k completed in stage i at time c_{ik} departs at time $d_{ik} \geq c_{ik}$ to an available processor in the next stage $i + 1$. If at time c_{ik} all m_{i+1} processors in stage $i + 1$ are occupied, then the processor in stage i is blocked by part k until time $d_{ik} = c_{i+1k} - p_{i+1k}$ when part k starts processing on an available processor in stage $i + 1$. If $i + 1$ is a buffer stage, then $p_{i+1k} = 0$, $d_{ik} = c_{i+1k}$, and $d_{ik+1} - d_{ik}$ is the waiting time of part k in the buffer stage $i + 1$.

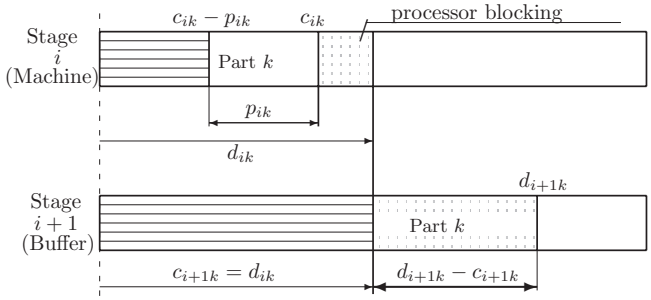

Figure 1.4 A partial schedule for part k in a flow shop with finite in-process buffers.

The problem of scheduling flow shops with parallel machines and finite in-process buffers is formulated below as a mixed integer program *FPB*.

Model FPB: *Scheduling Flow Shops with Parallel Machines and Blocking (Finite In-Process Buffers)*

Minimize (1.1) subject to

1. *Part Completion Constraints*: (1.2), (1.3)
2. *Maximum Completion Time Constraints*: (1.6)
3. *Processor Assignment Constraints*: (1.10)
4. *Part Departure Constraints*: (1.14), (1.15)
5. *No Buffering Constraints*: (1.16)
6. *Part Noninterference Constraints*:
   - no two parts assigned to the same processor can be processed simultaneously,

$$c_{ik} + Q(2 + y_{kl} - x_{ijk} - x_{ijl}) \geq d_{il} + p_{ik};\ \ i \in I, j \in J_i, k, l \in K: k < l \tag{1.20}$$

$$c_{il} + Q(3 - y_{kl} - x_{ijk} - x_{ijl}) \geq d_{ik} + p_{il};\ \ i \in I, j \in J_i, k, l \in K: k < l \tag{1.21}$$

7. *Variable Nonnegativity and Integrality Conditions*: (1.7)–(1.9), (1.13), (1.19).

Part noninterference constraints (1.20) and (1.21) incorporate assignment variables x_{ijk} and x_{ijl}. For a given sequence of parts at most one constraint of (1.20) and (1.21) is active, and only if in stage i both parts k and l are assigned to the same processor j (machine, if $p_{ijk} > 0$ or buffer, if $p_{ijk} = 0$), i.e., only if $x_{ijk} = x_{ijl} = 1$. Otherwise, both (1.20) and (1.21) are inactive. For any two parts assigned in some stage to the same processor (machine or buffer), the noninterference constraints (1.20) and (1.21) prevent simultaneous processing by the same machine or simultaneous occupying of the same buffer space. However, two parts assigned to different parallel processors (machines or buffers) in the same stage can be processed in parallel, and then both constraints (1.20) and (1.21) are inactive.

Model *FPB* for scheduling flexible flow shops with parallel processors and blocking (finite in-process buffers) is a general formulation and includes all the previous models as well as various special cases. For example, if $|J_i| = 1,\ \forall\, i \in I$ and if $p_{ik} > 0,\ \forall\, i \in I,\ k \in K$ model *FPB* reduces to model *F1B* for scheduling flow shops with single machines and no in-process buffers.

1.2.3 Alternative Objective Functions

In addition to the makespan, minimization of the total completion time of all parts, $C_{\text{sum}} = \sum_{k \in K} c_{mk}$ is sometimes considered in practice. The objectives $C_{\max}$ and C_{sum} aim at completing the production order as fast as possible and completing them fast in average, respectively. The total completion time represents average flow time and is also used as an implicit measure of work in process. Furthermore,

where due dates D_k for completing of some parts $k \in K$ are given, then minimization of the maximum tardiness among all the parts, $T_{\max} = \max_{k \in K}\{0, c_{mk} - D_k\}$ or minimization of the total tardiness of all the parts, $T_{\text{sum}} = \sum_{k \in K} \max\{0, c_{mk} - D_k\}$ can be used as an optimality criterion of the processing schedule. The aim of $T_{\max}$ and T_{sum} is to minimize the maximum and the average delay, respectively, of part completion times with respect to given due dates.

If minimization of C_{sum} or T_{sum} is selected as an alternative objective function, then variable $C_{\max}$ and the *maximum completion time constraints* (1.6) should be removed from the MIP model. In addition, if T_{sum} is selected, new variables T_k that represent tardiness of each part k are introduced. The new objective function is to minimize $T_{\text{sum}} = \sum_{k \in K} T_k$ subject to the additional constraints that define the tardiness variables:

Part tardiness constraints

$$c_{mk} \le D_k + T_k; \;\; k \in K$$
$$T_k \ge 0; \;\; k \in K.$$

If in the objective function (1.1), $C_{\max}$ is replaced with $T_{\max}$, then the *maximum completion time constraints* (1.6) should be replaced with the *maximum tardiness constraints* to ensure that for each tardy part k, its tardiness $(c_{mk} - D_k)$ cannot exceed the maximum tardiness $T_{\max}$ to be minimized:

Maximum tardiness constraints

$$c_{mk} \le D_k + T_{\max}; \;\; k \in K$$
$$T_{\max} \ge 0.$$

1.2.4 Transportation Times

When in addition to processing times also transportation times between successive stages should be considered, then the *no buffering constraints* (1.16) should be replaced with the following constraints (1.22).

No buffering constraints with transportation times considered: in every stage processing of each part starts immediately after its arrival at this stage,

$$c_{ik} - p_{ik} = d_{i-1k} + q_{i-1}; \;\; i \in I, k \in K: i > 1 \tag{1.22}$$

where q_i is the transportation time required to transfer a part from stage i to stage $i + 1$.

1.2.5 Reentrant Flow Shops

The proposed MIP models can also be applied for scheduling reentrant flow shops, where a part visits a set of stages more than once. For example, in a double-pass surface mount technology line (see Chapter 2) the double-sided printed wiring boards run twice through the same line, first to assemble the bottom side and then to assemble the top side. In order to extend an MIP for scheduling a double-pass reentrant flow shop, the number of parts is doubled to $2n$. A pair of parts $(k, k + n)$, $k = 1, \ldots, n$ represents

the bottom and the top sides of board k. The release time for part $k+n$ cannot be less than the completion time of part k, that is, additional part completion constraints should be added for each part $k+n$, $k=1,\ldots,n$

Part completion constraints: each part $k+n$ can be started in the first stage only after completion of part k in the last stage,

$$c_{1,k+n} - p_{1,k+n} \geq c_{m,k};\ k=1,\ldots,n. \tag{1.23}$$

1.2.6 Scheduling Modes

The MIP models *F1*, *FP*, *F1B*, and *FPB* represent *general scheduling* problems, where any input sequence of parts is allowed. In order to reduce the complexity of the general scheduling problem, the following scheduling modes can also be considered.

- *Batch scheduling*, where parts of a given type are scheduled consecutively, and in addition:
  - the sequence of part types is fixed and equal to the optimal sequence determined for a Minimal Part Set (MPS) in the same proportion as the overall production target or
  - the sequence of part types is not determined *a priori*, but is obtained along with the optimal schedule for all parts.
- *Cyclic scheduling*, where a Minimal Part Set (MPS), in the same proportion as the overall production target, is repetitively scheduled and in addition:
  - the cycle of parts in an MPS is fixed and equal to the optimal sequence determined for the MPS or
  - the cycle of parts in an MPS is not determined *a priori*, but is obtained along with the optimal schedule for all parts.

The cyclic scheduling mode is often used when set up times are negligible and the demand for each part type remains constant over the scheduling horizon. As a result, inventory holding costs are reduced. Otherwise, the batch scheduling mode is applied, where it is more efficient to have long runs of identical parts to minimize sequence dependent set up times. While acyclic schedules (e.g., general or batch scheduling mode) often lead to the highest throughput or the smallest makespan, a cyclic schedule does not guarantee the maximum throughput or the minimum makespan.

MIP models for the general scheduling can also be used for the batch or cyclic scheduling mode after a simple addition of the constraints presented below.

Denote by

$$G,$$
$$K=\{1,\ldots,n\},$$

and

$$K_g = \left\{ \sum_{f\in G: f<=g-1} n_f + 1,\ldots, \sum_{f\in G: f<=g-1} n_f + n_g \right\}$$

the ordered sets of indices, respectively of all batches of parts, all individual parts, and all parts of type $g \in G$. (n_g and $n = \sum_{g \in G} n_g$ denote, respectively the number of parts type g and the total number of parts in the schedule.) Let $r_{ig} \geq 0$ be the processing time in stage i of part type $g \in G$.

Batch scheduling mode constraints:

$$y_{kl} = 1; g \in G, k \in K_g, l \in K_g : k < l \tag{1.24}$$

$$y_{kl} = y_{last(K_f),last(K_g)}; f \in G, g \in G, k \in K_f, l \in K_g : f < g \tag{1.25}$$

$$c_{ik+1} \geq d_{ik} + r_{ig}; i \in I, g \in G, k \in K_g : k < last(K_g), m_i = 1, \tag{1.26}$$

where $last(K_f)$ is last part in the ordered set K_f.

Equation (1.25) selects a sequence of processing different part types and constraints (1.24) and (1.26) ensure that boards of one type are processed consecutively.

In the cyclic scheduling mode, first the Minimal Part Set (MPS) should be determined, i.e., the smallest possible set of parts in the same proportion as the overall production target, which is to be repetitively processed in a cyclic mode. For example, if the production target is 100 units of part A, 300 units of part B and 500 units of part C, then MPS is one unit of A, three units of B, and five units of C, in total nine parts, which is to be repeated 100 times to meet the production target, using the same order of processing parts in each run of MPS. In the cyclic scheduling, all parts in an MPS can be ordered either arbitrarily, as in the general mode (e.g., C,C,A,B,C,B,C,B,C) or in batches of identical parts, as in the batch mode (e.g., C,C,C,C,C,A,B,B,B). The latter cyclic mode combined with the batch processing of identical parts in each run of MPS can be called a cyclic-batch scheduling mode.

Formally, the Minimal Part Set (MPS) is defined as

$$\{\underline{n}_g : g \in G\},$$

where

$$n_g = S\,\underline{n}_g \;\; \forall\, g \in G$$

and S is the greatest common divisor of integers $n_1, n_2, \ldots n_{|G|}$ ($|\cdot|$denotes the power of a set $\cdot$), i.e., a total of S runs of an MPS is required to meet the overall production target. Denote by $\underline{n} = \sum_{g \in G} \underline{n}_g$, the total number of parts in an MPS.

Cyclic scheduling mode constraints: (1.24) and

$$y_{kl} = y_{next(k,K_f,s\underline{n}_f),next(l,K_g,s\underline{n}_g)}; f \in G, g \in G, k \in K_f, l \in K_g,$$

$$1 \leq s \leq S-1 : f < g, 1 \leq ord(k, K_f) \leq \underline{n}_f, 1 \leq ord(l, K_g) \leq \underline{n}_g \tag{1.27}$$

$$c_{i,next(k,K_g,\underline{n}_g)} \geq d_{ik} + \sum_{f \in G} \underline{n}_f\, r_{if}; i \in I, g \in G, k \in K_g :$$

$$k \leq last(K_g) - \underline{n}_g, m_i = 1 \tag{1.28}$$

$$c_{ik} \geq d_{il} + p_{ik};\ i \in I, f \in G, g \in G, k \in K_f, l \in K_g, 1 \leq s \leq S-1:$$
$$s\underline{n}_f < ord(k, K_f) \leq (s+1)\,\underline{n}_f\,,\ (s-1)\,\underline{n}_g < ord(l, K_g) \leq s\,\underline{n}_g\,,\ m_i = 1, \quad (1.29)$$

where $ord(k, K_f)$ denotes ordinal position of k in K_f, and $next(k, K_f, s\underline{n}_f)$ is the part type f, $s\underline{n}_f$ positions after part k in the ordered set K_f.

Equation (1.27) imposes the same sequence on processing parts in each run of an MPS, constraint (1.28) ensures periodic processing of every $\underline{n}_g$th part of each type g, and constraint (1.29) ensures that no part of a new run of MPS can be started on a machine until all parts from the previous run are departed.

Cyclic-batch scheduling mode constraints: (1.24), (1.28), (1.29) and

$$y_{kl} = y_{last(K_f),last(K_g)};\ f \in G, g \in G, k \in K_f, l \in K_g, 1 \leq s \leq S:$$
$$f < g,\ (s-1)\,\underline{n}_f < ord(k, K_f) \leq s\underline{n}_f\,,\ (s-1)\,\underline{n}_g < ord(l, K_g) \leq s\underline{n}_g\,. \quad (1.30)$$

Equation (1.30) selects the same cyclic sequence of processing minimal batches of different part types over all S runs of MPS.

If the production target consists of equal batch sizes for all part types, that is, $n_g = n/|G|\ \forall g \in G$, then $\underline{n}_g = 1\ \forall\ g \in G$. As a consequence, MPS is made of one unit of each part type, in total of $\underline{n} = |G|$ different parts, and the unit MPS set should be repeatedly processed $S = n/|G|$ times. In this case, the cyclic scheduling mode constraints reduce to the following set of constraints.

Cyclic scheduling mode constraints for a unit MPS: (1.24) and

$$y_{kl} = y_{last(K_f),last(K_g)};\ f \in G, g \in G, k \in K_f, l \in K_g:$$
$$f < g,\ ord(k, K_f) = ord(l, K_g) \quad (1.31)$$
$$y_{kl} = 1 - y_{next(k,K_f),l};\ f \in G, g \in G, k \in K_f, l \in K_g:$$
$$f < g,\ k < last(K_f),\ ord(k, K_f) = ord(l, K_g) \quad (1.32)$$
$$c_{i,next(k,K_g)} \geq d_{ik} + \sum_{f \in G} r_{if};\ i \in I, g \in G, k \in K_g:$$
$$k < last(K_g), m_i = 1, \quad (1.33)$$

where $next(k, K_f)$ is next part after k in the ordered set K_f.

Equations (1.31), (1.32) select the same sequence of processing different part types in successive runs of an MPS, and constraint (1.33) ensures that successive parts of one type are processed periodically.

In addition to the sequencing and timing constraints, the following assignment constraints should be added to the MIP formulations to model alternate assignment of parts to parallel machines, for example, the assignment by a shuttle device in a surface mount technology line (see, Chapter 2).

Assignment constraints for batch scheduling mode:

- in each stage with parallel machines, successive parts of one type are assigned to successive parallel machines,

$$x_{i,next(j,J_i),k+1} = x_{ijk};\ i \in I, j \in J_i, g \in G, k \in K_g: \\ k \le last(K_g) - 1, m_i > 1, \tag{1.34}$$

where $next(j, J_i)$ is the next machine after j in the circular set J_i of parallel machines in stage i.

Assignment constraints for cyclic and cyclic-batch scheduling mode:

- in each stage i with parallel machines, corresponding parts in successive runs of MPS are assigned to every $\underline{n}$th machine in the circular set J_i of machines

$$x_{i,next(j,J_i,\underline{n}),next(k,K_g,\underline{n}_g)} = x_{ijk};\ i \in I, j \in J_i, g \in G, k \in K_g: \\ k \le last(K_g) - \underline{n}_g,\ m_i > 1, \tag{1.35}$$

where $next(j, J_i, \underline{n})$ is the parallel machine in stage i, $\underline{n}$ $(= \sum_{g \in G} \underline{n}_g)$ positions after machine j in the circular set J_i of parallel machines, and $next(k, K_g, \underline{n}_g)$ is the part type g, $\underline{n}_g$ positions after part k in the ordered set K_g.

1.2.7 Computational Examples

In this subsection two numerical examples are presented to illustrate possible applications of the proposed MIP models.

The flexible flow shop configurations for the example problems are provided in Figures 1.5 and 1.6. The flow shop with no buffers in Figure 1.5 is made up of $m = 3$ processing stages, each representing a single machine or a set of parallel machines.

The following two variants of the three-stage flow shop with no buffers will be considered in the examples:

F3_1—the flow shop with single machines: $m_1 = m_2 = m_3 = 1$

F3_P—the flow shop with parallel machines: $m_1 = 2, m_2 = 3, m_3 = 2$

The flow shop with finite in-process buffers in Figure 1.6 consists of $m = 5$ stages, where stages $i = 1, 3, 5$ represent single or parallel machines and stages $i = 2, 4$ represent one or more buffers.

The following two variants of the five-stage flow shop with finite in-process buffers is considered in the examples:

F5_1—the flow shop with single processors: $m_1 = m_2 = m_3 = m_4 = m_5 = 1$

F5_P—the flow shop with parallel processors: $m_1 = 2, m_2 = 3, m_3 = 3, m_4 = 3, m_5 = 2$

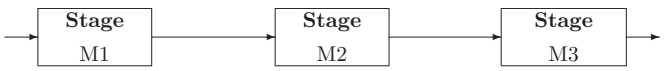

Figure 1.5 A three-stage flow shop with single machines and no in-process buffers.

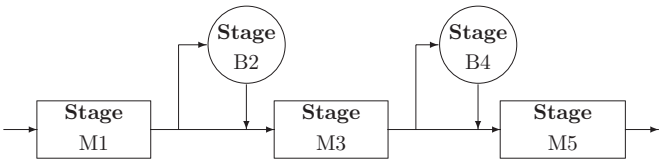

Figure 1.6 A five-stage flow shop with parallel machines and finite in-process buffers.

The production order consists of $n = 10$ parts, and the processing times p_{ik} are shown below for the three-stage and the five-stage flow shop, respectively:

$$[p_{ik}] = \begin{bmatrix} 4, 2, 1, 5, 4, 3, 5, 2, 1, 8 \\ 2, 5, 8, 6, 7, 4, 7, 3, 6, 2 \\ 2, 8, 6, 2, 2, 4, 2, 7, 1, 8 \end{bmatrix},$$

$$[p_{ik}] = \begin{bmatrix} 4, 2, 1, 5, 4, 3, 5, 2, 1, 8 \\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 \\ 2, 5, 8, 6, 7, 4, 7, 3, 6, 2 \\ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 \\ 2, 8, 6, 2, 2, 4, 2, 7, 1, 8 \end{bmatrix}.$$

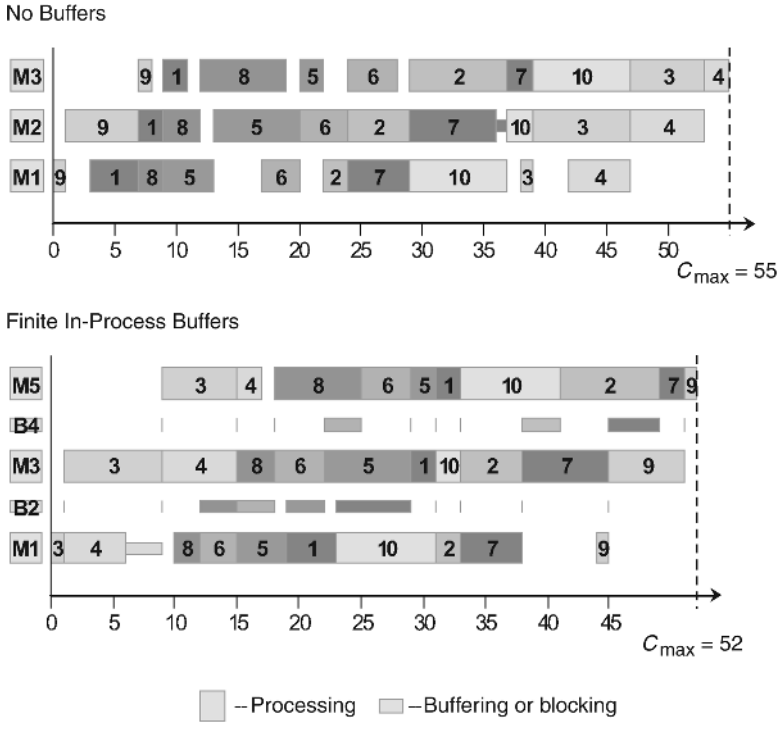

Figure 1.7 Optimal schedules for flow shops with single processors.

Let us note that for the buffer stages $i = 2, 4$ in the five-stage flow shop all processing times are equal to zero.

For the example problems the following simple lower bounds on the schedule length $C_{\max}$ can be calculated

$$LBC_{\max} = \max_{i \in I} \left\{ \sum_{k \in K} p_{ik} + \min_{k \in K} \left(\sum_{h \in I : h < i} p_{hk} \right) + \min_{k \in K} \left(\sum_{h \in I : h > i} p_{hk} \right) \right\} = 52,$$

$$LBC_{\max} = \max_{i \in I} \left\{ \left\lceil \sum_{k \in K} p_{ik}/m_i \right\rceil + \min_{k \in K} \left(\sum_{h \in I : h < i} p_{hk} \right) + \min_{k \in K} \left(\sum_{h \in I : h > i} p_{hk} \right) \right\} = 26,$$

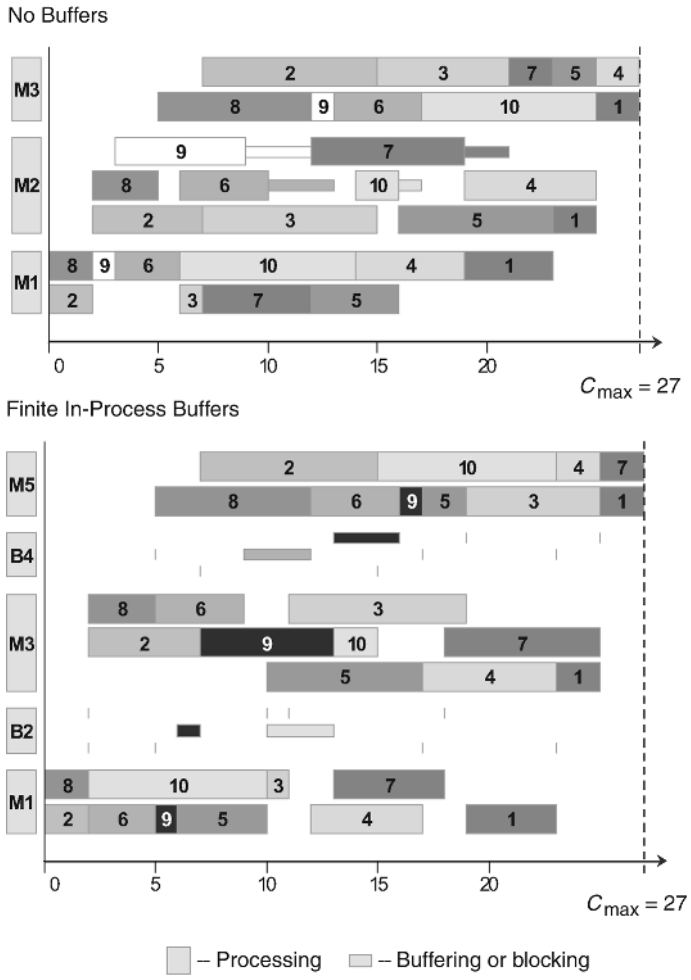

Figure 1.8 Optimal schedules for flow shops with parallel processors.

for the flow shop with single processors and with parallel processors, respectively ($\lceil \cdot \rceil$ is the smallest integer not less than $\cdot$). The above lower bounds are calculated as maximum over all stages of a stage workload and minimum flow time of a part through the remaining stages.

The optimal schedules for the examples were determined by solving the mixed integer programs *F1B* and *FPB* for the flow shop configurations *F3_1*, *F5_1*, and *F3_P*, *F5_P*, respectively. The solution values are $C_{\max} = 55$, $C_{\max} = 52$, $C_{\max} = 27$, and $C_{\max} = 27$, respectively, for configurations *F3_1*, *F5_1*, *F3_P*, and *F5_P*. The optimal schedules are shown on Gantt charts in Figures 1.7 and 1.8.

1.3 CONSTRUCTIVE HEURISTICS FOR SCHEDULING FLEXIBLE FLOW SHOPS

This section presents two fast constructive heuristics for scheduling flexible flow shops with finite in-process buffers or with no in-process buffers. The heuristics are not based on MIP; however, they use similar variables to determine the processing schedule and their performance can be compared to MIP approaches presented in the previous sections.

Unlike the exact MIP procedures capable of finding the proven optimal solution, heuristics are not able to find a proven optimal solution. In general, the heuristics can be divided into constructive procedures and improvement procedures. A constructive procedure is capable of constructing the solution from scratch step by step, which can be done either job-by-job or stage-by-stage, whereas an improvement procedure improves a given initial or constructed solution. The constructive heuristics proposed in this section are single-pass, part-by-part procedures, in which during every iteration a complete processing schedule is constructed for one part. The selection of the part and its complete schedule are based on the cumulative partial schedule obtained for all parts selected so far. The decisions in every iteration are made using a local optimization mechanism aimed at minimizing total idle time along the route of the selected part. For this reason, the heuristics will be called *Route Idle Time Minimization* (RITM) (Sawik, 1993) or *Route Idle Time Minimization–No Store* (RITM-NS) (Sawik, 1994, 1995b), respectively, for the flow shop with finite in-process buffers or the flow shop with no in-process buffers. Notation used to formulate the scheduling problems and the heuristic algorithms is introduced in Table 1.2.

1.3.1 A Fast Heuristic for Scheduling Flow Shops with Finite In-Process Buffers

The flexible flow shop under study consists of $m \geq 2$ processing stages in series with limited in-process buffers between the successive stages. Each stage i ($i = 1, \ldots, m$) is made up of $m_i \geq 1$ parallel identical machines. Ahead of each stage i($i = 2, \ldots, m$) there are b_i buffers, where each buffer can hold one part at a time (see Fig. 1.9).

Table 1.2 Notation: Heuristic Algorithms

Indices

g = part type, $g \in G$
i = processing stage, $i \in I = \{1, \ldots, m\}$
j = parallel machine in stage i, $j \in J_i = \{1, \ldots, m_i\}$
k = part, $k \in K = \bigcup_{g \in G} K_g = \{1, \ldots, n\}$
l = position in the loading sequence, $l = 1, \ldots, n$

Input parameters

b_i = number of buffers ahead of stage i ($i = 2, \ldots, m$)
K_g = subset of parts type g
n_g = demand for part type g, ($n_g = |K_g|$)
n = total number of parts in the production order, $n = \sum_{g \in G} n_g$
P_{ig} = processing time in stage i of part type g
p_{ik} = P_{ig}, $k \in K_g$—processing time in stage i of part k
q_i = transportation time from stage i to stage $i + 1$

Decision variables

k_l = part in position l ($l = 1, \ldots, n$) in the loading sequence $[k_1, k_2, \ldots, k_n]$
f_{il} = finish time of processing part k_l on machine in stage i
r_{il} = release time of part k_l from stage i
s_{il} = start time of part k_l on machine in stage i

Auxiliary variables

t_{il} = idle time on machine in stage i incurred by part k_l (waiting time for start of processing part k_l and blocking time by finished part k_l)
Ξ_{bil} = total elapsed time, when buffer b ($b = 1, \ldots, b_i$) ahead of stage i is available for assignment after the first l parts have been scheduled
ξ_{il} = the earliest time at which a buffer ahead of stage i is available for assignment after the first l parts have been scheduled, $\xi_{il} = \min_{1 \le b \le b_i} \{\Xi_{bil}\}$
$\beta(i, l)$ = buffer ahead of stage i with the earliest available time after the first $l - 1$ parts have been scheduled, $\beta(i, l) = \arg\min_{1 \le b \le b_i} \{\Xi_{bi,l-1}\}$
Γ_{jil} = total elapsed time, when machine j ($j = 1, \ldots, m_i$) at stage i is available for assignment after the first l parts have been scheduled
γ_{il} = the earliest time at which a machine in stage i is available for assignment after the first l parts have been scheduled, $\gamma_{il} = \min_{1 \le j \le m_i} \{\Gamma_{jil}\}$
$\mu(i, l)$ = machine in stage i with the earliest available time after the first $l - 1$ parts have been scheduled, $\mu(i, l) = \arg\min_{1 \le j \le m_i} \{\Gamma_{ji,l-1}\}$

The system produces different part types $g \in G$. Let $P_{ig} \ge 0$ be the processing time in stage i of part type g and let q_i be the transportation time required to transfer a part from stage i to stage $i + 1$.

A part completed in stage i is transferred either directly to an available machine in the next stage $i + 1$ (or another downstream stage depending on the part processing

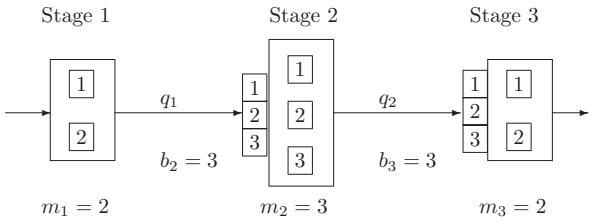

Figure 1.9 A three-stage flexible flow shop with finite in-process buffers.

route) or to a buffer ahead of that stage. If all b_{i+1} buffers ahead of stage $i+1$ are occupied, then the machines in stage i are blocked by the completed parts until a buffer is available.

Given are demands n_g, $g \in G$ for all part types. The problem objective is to determine an assignment of all $n = \sum_{g \in G} n_g$ parts to machines in each stage over a scheduling horizon to meet the demand in minimum time, that is, to minimize the makespan $C_{\max}$.

The production schedule for all n parts is specified by the loading sequence (permutation of parts) $[k_1, k_2, \ldots, k_n]$ in which the parts enter the line, as well as the times required for detailed scheduling of each individual part. In particular, for each part k_l entering the line at position l ($l = 1, \ldots, n$) its start time s_{il}, finish time f_{il}, and release time r_{il} in each stage i should be determined. These variables satisfy the following formulas:

$$s_{1l} = \gamma_{1,l-1} \tag{1.36}$$

$$s_{il} = \max\{r_{i-1,l} + q_{i-1};\ \gamma_{i,l-1}\};\ i = 2, \ldots, m \tag{1.37}$$

$$f_{il} = s_{il} + p_{ik_l};\ i = 1, \ldots, m \tag{1.38}$$

$$r_{il} = \max\{f_{il},\ \xi_{i+1,l-1} - q_i\};\ i = 1, \ldots, m-1 \tag{1.39}$$

$$r_{ml} = f_{ml} \tag{1.40}$$

$$\xi_{il} = \min_{1 \le b \le b_i}\{\Xi_{bil}\};\ i = 2, \ldots, m \tag{1.41}$$

$$\gamma_{il} = \min_{1 \le j \le m_i}\{\Gamma_{jil}\};\ i = 1, \ldots, m \tag{1.42}$$

where the earliest available times Ξ_{bil} for buffers and Γ_{jil} for machines are calculated iteratively as follows

$$\Xi_{bil} = \begin{cases} \Xi_{bil-1} & \text{if } b \neq \beta(i,l) \\ s_{il} & \text{if } b = \beta(i,l) \end{cases} \tag{1.43}$$

$$\Gamma_{jik} = \begin{cases} \Gamma_{jil-1} & \text{if } j \neq \mu(i,l) \\ r_{il} & \text{if } j = \mu(i,l) \end{cases} \tag{1.44}$$

The formulae (1.43) and (1.44) were derived under the assumption that in every stage i part k_l is assigned to the buffer $\beta(i,l)$ and the machine $\mu(i,l)$ with the earliest available times (for additional interpretation of the variables introduced, see Fig. 1.10).

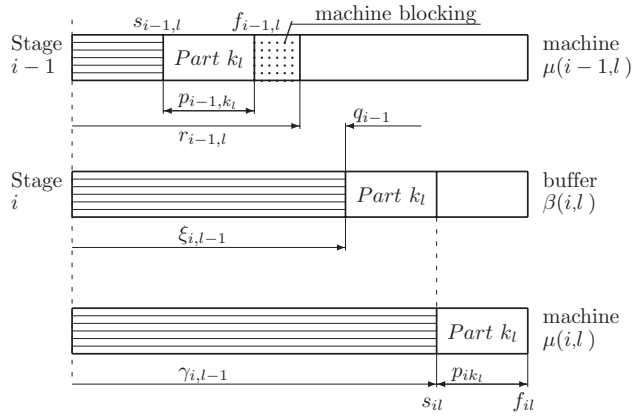

Figure 1.10 A partial schedule for part k_l.

Note that finish time f_{il} and release time r_{il} of the lth part in the loading sequence of the heuristic correspond to completion time c_{ik} and departure time d_{ik} of part k in the MIP models presented in Section 1.2.2.

Finally, notice that machine idle time t_{il} incurred in stage i by part k_l consists of two components:

- Time $(s_{il} - \gamma_{i,l-1})$ of machine waiting for start of processing part k_l
- Time $(r_{il} - f_{il})$ of machine blocking by finished part k_l

Hence, t_{il} can be expressed as follows

$$t_{il} = (s_{il} - \gamma_{i,l-1}) + (r_{il} - f_{il}) = r_{il} - \gamma_{i,l-1} - p_{ik_l};\ \ i = 1, \ldots, m \tag{1.45}$$

In order to minimize the makespan $C_{\max}$, the RITM heuristic presented in the next section assigns parts to the earliest available machines and aims at minimizing the total idle time $\sum_{i=1}^{m} \sum_{l=1}^{n} t_{il}$ along the routes of all n parts.

The RITM algorithm is a single-pass, part-by-part heuristic in which the loading sequence and the corresponding complete schedule are determined once. During every iteration a part for loading into the system is chosen as well as its complete processing schedule is determined. The decisions in every iteration are made based on the complete processing schedule determined for each part type waiting for loading. Given a cumulative partial schedule for the first $(l-1)$ parts selected so far, first the best route along the line is found as a sequence

$$[\mu(1, l), \beta(2, l), \mu(2, l), \ldots, \beta(m, l), \mu(m, l)] \tag{1.46}$$

of m machines and $m-1$ buffers (one in every stage) with the earliest available times. For each part type g waiting to enter the line the complete processing schedule is determined along the best route obtained. To evaluate the processing schedule for each part type considered for loading, the total duration of idle time t_g along the best route is

determined:

$$t_g = \sum_{i=1}^{m} t_{ig} \tag{1.47}$$

Finally, the part type with the smallest total idle time t_g is selected for loading and its complete processing schedule is added to the cumulative partial schedule obtained so far.

Description of the algorithm RITM is given below ($\overline{G}$ denotes the set of part types loaded in the required number of parts).

Algorithm RITM

Step 0. Starting

1. Order all part types according to nonincreasing total processing times $p_g = \sum_{i=1}^{m} P_{ig}$, that is $p_1 \geq p_2 \geq \cdots \geq p_{|G|}$.

2. Set:

$$\Xi_{bi0} = 0,\ b = 1, \ldots, b_i,\ i = 2, \ldots, m.$$

$$\Gamma_{ji0} = 0,\ j = 1, \ldots, m_i,\ i = 1, \ldots, m,$$

$$\beta(i, 1) = 1,\ i = 2, \ldots, m,\ \mu(i, 1) = 1,\ i = 1, \ldots, m,$$

$$\overline{G} = \emptyset,\ l = 1.$$

Step 1. Selection of a part type for loading

1. For each part type $g \notin \overline{G}$ waiting for loading at position l determine start time s_{il}, finish time f_{il}, release time r_{il}, and duration t_{ig} of machine idle time, in every stage i on machine $\mu(i, l)$ with the earliest available time γ_{il-1}. Next, determine total idle time t_g.

2. Select for loading such a part type g_l that minimizes the total idle time, that is

$$g_l = \arg \min_{g \notin \overline{G}} \{t_g\}$$

To break ties, select the part type with the lowest number (i.e., with the largest total processing time).

Step 2. Determining complete processing schedule for the selected part type

For the selected part type g_l determine complete processing schedule (times s_{il}, f_{il}, r_{il}) by assigning it in every stage i to the buffer $\beta(i, l)$ and the machine $\mu(i, l)$ with the earliest available times, respectively ξ_{il-1} and γ_{il-1}, after the first $l - 1$ parts have been scheduled.

The processing schedule for a part $k_l \in K_{g_l}$ add to the cumulative partial schedule for the first $(l - 1)$ parts.

Step 3. Checking the state of completion of the production order

For $g = g_l$ set $n_g = n_g - 1$. If $n_g = 0$, then set $\overline{G} = \overline{G} \cup \{g\}$.
If $\overline{G} = G$, then terminate.

Otherwise determine for each buffer b ($b = 1, \ldots, b_i,\ \ i = 2, \ldots, m$) and for each machine j ($j = 1, \ldots, m_i,\ \ i = 1, \ldots, m$) the earliest available time, respectively Ξ_{bil} and Γ_{jil}, after the first l parts have been scheduled.

For each stage i find buffer $\beta(i, l+1) = \arg\min_{1 \le b \le b_i}(\Xi_{bil})$ and machine $\mu(i, l+1) = \arg\min_{1 \le j \le m_i}(\Gamma_{jil})$ with the earliest available time ξ_{il} and γ_{il}, respectively.

Set $l = l + 1$ and go to *STEP 1*.

To determine the computational complexity of the algorithm RITM notice that the algorithm requires n iterations and in every iteration a complete processing schedule is computed for one part according to the following procedure.

First, the best route is found as a sequence of at most m machines and $m-1$ buffers in the successive stages with the earliest available times. This step requires $O(M+B)$ computations since at most all M machines and B buffers are considered ($M = \sum_{i=1}^{m} m_i$ and $B = \sum_{i=2}^{m} b_i$ is the total number of machines and buffers, respectively). Then, for each of the remaining at most $|G|$ part types ($|\cdot|$ denotes the power of a set $\cdot$), the complete processing schedule is computed based on the best route found. This step requires $O(m|G|)$ computations to determine the times for the $|G|$ flow shop schedules on m machines. The best processing schedule (and by this a part for loading) with the smallest total idle time is next chosen and added to the cumulative partial schedule for all parts loaded so far.

Since the above procedure which requires $O(M + B + m|G|)$ computations is repeated in each of the n iterations, and the number of part types $|G| \le n$, the computational complexity of the algorithm RITM is $O(mn^2)$.

In order to evaluate the effectiveness of the proposed heuristic algorithm, the following lower bound on makespan can be used as a surrogate for the minimum makespan value

$$LBC_{\max} = \max_{1 \le i \le m} \left\{ \left\lceil \sum_{g \in G} n_g P_{ig} / m_i \right\rceil + \min_{g \in G} \left(\sum_{h \in I: h < i} p_{hg} \right) + \min_{g \in G} \left(\sum_{h \in I: h > i} p_{hg} \right) \right\} + \sum_{i \in I} q_i, \tag{1.48}$$

where $\lceil \cdot \rceil$ is the smallest integer not less than $\cdot$.

The above lower bound is the sum of total transportation time and maximum over all stages of a stage average workload plus minimum flow time of a part type in all upstream and downstream stages.

1.3.2 A Fast Heuristic for Scheduling Flow Shops with No In-Process Buffers

This subsection presents a single-pass heuristic algorithm for the scheduling of parts through a flexible flow shop with no in-process buffers (Sawik, 1995b). Since there are no buffers between the stages, intermediate queues of parts waiting in the system for their next operations are not allowed. The algorithm, called *Route Idle Time*

Minimization–No Store (RITM-NS), is a special variant of the RITM heuristic designed for scheduling flexible flow shops with finite in-process buffers and presented in the previous subsection. During every iteration a part for loading into the system is chosen as well as its complete processing schedule is determined. The decisions in every iteration are made based on the complete processing schedule determined for each part type waiting for loading. Given a cumulative partial schedule for the first $(l-1)$ parts selected so far, first the best route along the line is found as a sequence

$$[\mu(1, l), \mu(2, l), \ldots, \mu(m, l)]$$

of m machines (one in every stage) with the earliest available times. For each part type waiting for entering the line the complete processing schedule is determined along the best route. To evaluate the processing schedule for each part type k considered for loading, the total duration of machine idle time along the best route is determined. Finally, the part with the smallest total machine idle time is selected for loading and its complete processing schedule is added to the cumulative partial schedule obtained so far. The RITM-NS algorithm can be obtained from the RITM algorithm by removing from the latter all the buffer variables Ξ_{bil}, ξ_{il-1} and $\beta(i, l)$. A brief description of the RITM-NS algorithm is given below:

Algorithm RITM-NS

Step 0. Starting (as for RITM with buffer variables Ξ_{bil} and $\beta(i, l)$ deleted)

Step 1. Selection of a part type for loading (as for RITM with (1.39) replaced by $r_{il} = \max\{f_{il};\ \gamma_{i+1,l-1} - q_i\},\ i = 1, \ldots, m-1$)

Step 2. Determining complete processing schedule for the selected part (as for RITM with buffer variables ξ_{il-1} and $\beta(i, l)$ deleted)

Step 3. Checking the state of completion the production order (as for RITM with buffer variables Ξ_{bil}, ξ_{il-1} and $\beta(i, l)$ deleted).

The computational complexity of the RITM-NS algorithm is $O(mn^2)$, the same as that of the RITM heuristic.

1.3.3 Computational Examples

1.3.3.1 Flexible Flow Shop with Finite In-Process Buffers

First, an illustrative example for a three-stage flow shop with finite in-process buffers is presented. The flow shop (see Fig. 1.9) is made up of $m_1 = 2$ machines in stage $i = 1$, $m_2 = 3$ machines in stage $i = 2$, and $m_3 = 2$ machines in stage $i = 3$.

The number of buffers ahead of stage 2 and stage 3 are $b_2 = 3$ and $b_3 = 3$, respectively. Transportation times between stages are $q_1 = q_2 = 1$.

The production order consists of four part types and their corresponding production requirements (in number of parts) are $n_1 = 8$, $n_2 = 4$, $n_3 = 2$, and $n_4 = 3$. Therefore, the total number of parts to be scheduled is $n = 17$.

Processing times P_{ig} ($i = 1, 2, 3;\ g = 1, 2, 3, 4$) for each stage i and each part type g are shown below:

$$P_{11} = 5,\ P_{21} = 3,\ P_{31} = 7,$$
$$P_{12} = 2,\ P_{22} = 4,\ P_{32} = 6,$$
$$P_{13} = 3,\ P_{23} = 6,\ P_{33} = 1,$$
$$P_{14} = 1,\ P_{24} = 4,\ P_{34} = 2.$$

For the above data the lower bound on makespan is $LBC_{\max} = 51$ (1.48). The RITM heuristic constructs a schedule with the makespan $C_{\max} = 55$ and the loading sequence

$$[2, 2, 2, 2, 4, 3, 1, 3, 4, 4, 1, 1, 1, 1, 1, 1, 1].$$

Instead of using a Gantt chart, the constructed processing schedule is presented in the form of an assignment table (Table 1.3), where the assignment of parts to machines and buffers in each stage is indicated for every period of unit time duration. Numbers of the first and the last period of each assignment are given in the first column of the table.

The minimum values of total idle time t_{k_l} for every iteration $l = 1, \ldots, 17$ are the following:

$$8, 8, 4, 0, 0, 0, 0, 2, 1, 1, 0, 0, 1, 1, 3, 3, 4.$$

In order to better illustrate the solution procedure, some of the computations performed in iteration number 8 of the RITM algorithm are shown below as an example.

The partial loading sequence obtained during the first seven iterations is

$$[j_1, j_2, \ldots, j_7] = [2, 2, 2, 2, 4, 3, 1].$$

The partial schedule for $l = 7$ parts selected so far leads to the following values of times Ξ_{bil} ($b = 1, \ldots, b_i;\ i = 2, 3$) and Γ_{jil} ($j = 1, \ldots, m_i;\ i = 1, 2, 3$) at which, respectively, buffers and machines are available for assignment,

In stage 1: $\Gamma_{117} = 10$, $\Gamma_{217} = 7$

In stage 2: $\Xi_{127} = 11$, $\Xi_{227} = 7$, $\Xi_{327} = 9$, $\Gamma_{127} = 14$, $\Gamma_{227} = 11$, $\Gamma_{327} = 15$

In stage 3: $\Xi_{137} = 20$, $\Xi_{237} = 20$, $\Xi_{337} = 21$, $\Gamma_{137} = 22$, $\Gamma_{237} = 28$

The resulting earliest availability times: ξ_{ik} for buffer $\beta(i, 8)$ ahead of stage i ($i = 2, 3$), and γ_{ik} for machine $\mu(i, 8)$ in stage i ($i = 1, 2, 3$) are shown below:

$$\xi_{27} = 7,\ \xi_{37} = 20,\ \gamma_{17} = 7,\ \gamma_{27} = 11,\ \gamma_{37} = 22.$$

Therefore, the best route along the line is the following sequence of machines and buffers available at the earliest times:

$$[\mu(1, 8), \beta(2, 8), \mu(2, 8), \beta(3, 8), \mu(3, 8)] = [2, 2, 2, 1, 1].$$

In iteration 8, part k_8 for loading at position $l = 8$ has to be selected from among 10 remaining parts of three types $g = 1, 3, 4$. A complete processing schedule

Table 1.3 A Heuristic Schedule

Periods from–to	Assignment of parts to machines and buffers												
	Stage 1		Stage 2						Stage 3				
	Machines		Buffers			Machines			Buffers			Machines	
	1	2	1	2	3	1	2	3	1	2	3	1	2
1–3	2	2											
4	2	2				2	2						
5	4	3				2	2						
6	1	3	2			2	2	2					
7	1	3	2	4		2	2	2					
8	1	3				2	4	2					
9	1	3			3	2	4	2				2	2
10	1	3				2	4	3				2	2
11	4	4				2	4	3			2	2	2
12	1	1				1	3	3			2	2	2
13–14	1	1	4	4		1	3	3	2	4	2	2	2
15	1	1	4			4	3	3		4		2	2
16	1	1				4	3	4		4		2	2
17	1	1				4	3	4	3	4	1	2	2
18–19	1	1		1	1	4	3	4	3	4	1	2	2
20	1	1				1	1	4	3	4	1	2	2
21	1	1				1	1		3	4	1	4	3
22	1	1				1	1		3	4	4	4	1
23	1	1				1		1		4	4	3	1
24–25	1	1				1		1	1	1	4	4	1
26	1	1				1			1	1		4	1
27	1								1	1	1	4	1
28	1						1	1	1	1	1	1	1
29–30	1						1	1	1		1	1	1
31	1							1	1		1	1	1
32								1	1	1	1	1	1
33						1		1	1	1	1	1	1
34–35						1			1	1	1	1	1
36										1	1	1	1
37–41									1	1	1	1	1
42									1		1	1	1
43–48									1			1	1
49												1	1
50–55												1	

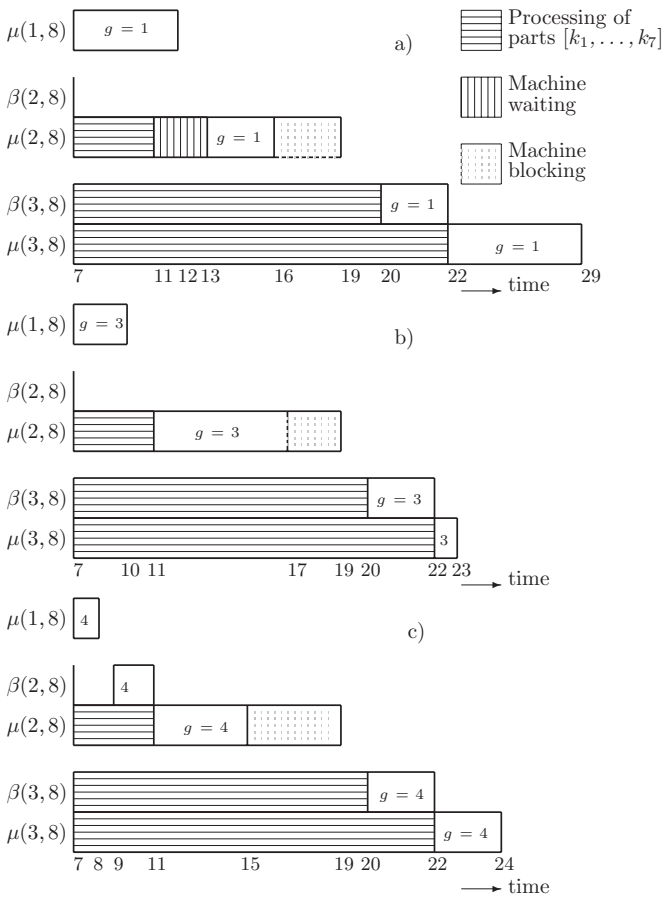

Figure 1.11 Processing schedules for part types $g = 1, 3, 4$ available for assignment to position 8 of the input sequence.

obtained along the best route for each of those three part types is shown in Figure 1.11, for part type 1 (Fig. 1.11 (a)), part type 3 (Fig. 1.11 (b)), and part type 4 (Fig. 1.11 (c)), respectively. The corresponding total idle times are $t_1 = 5$, $t_3 = 2$, $t_4 = 4$, and hence the minimum is $t_3 = 2$. Therefore, part type $g_8 = 3$ is selected for loading at position $l = 8$, and in iteration 8 its complete processing schedule shown in Figure 1.11 (b) is added to the cumulative partial schedule of the first seven parts.

1.3.3.2 Flexible Flow Shop with No In-Process Buffers

Below an illustrative example is presented for the same production order and the three-stage flexible flow shop with no in-process buffers, shown in Figure 1.12. The flow shop is made up of $m_1 = 2$ machines in stage $i = 1$, $m_2 = 3$ machines in stage $i = 2$, and $m_3 = 2$ machines in stage $i = 3$. Transportation times between stages are $q_1 = q_2 = 1$.

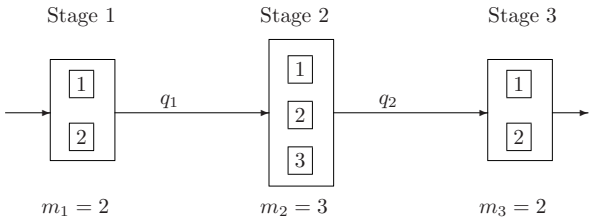

Figure 1.12 A three-stage flexible flow shop with no in-process buffers.

The lower bound on makespan is $LBC_{\max} = 51$ (1.48). The RITM-NS heuristic constructs a schedule with the makespan $C_{\max} = 52$ and the loading sequence

$$[4, 4, 2, 1, 3, 2, 3, 1, 1, 1, 1, 2, 1, 1, 1, 2, 4].$$

The optimal schedules for the two examples were obtained using the MIP model FPB with constraints (1.16) replaced by (1.22) to allow for inclusion of the transportation times between successive stages. The Gantt charts for the two optimal schedules are shown in Figures 1.13 and 1.14, where the three machine stages and the two buffers stages in Figure 1.13 are denoted by M1, M3, M5 and B2, B4, whereas the three machine stages in Figure 1.14 are marked with M1, M2, M3. The figures indicate that the optimal schedules have the same length, $C_{\max} = 52$.

It is interesting to compare the two heuristic schedules constructed for the two different flexible flow shops (with finite in-process buffers or with no in-process buffers) and the same production order. Unlike for the optimal schedules shown in

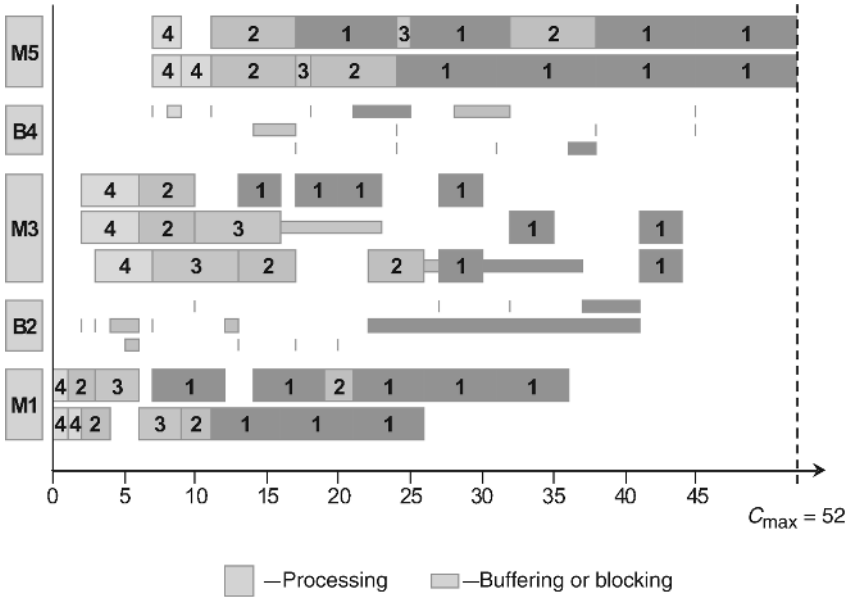

Figure 1.13 An optimal schedule for the flexible flow shop with finite in-process buffers.

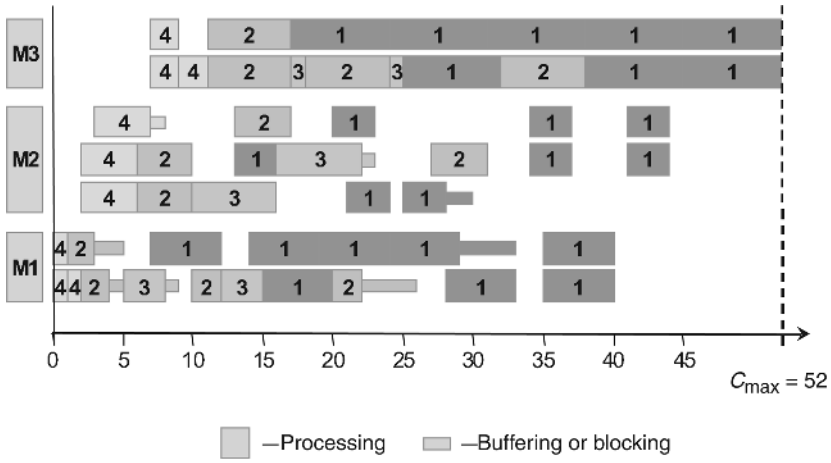

Figure 1.14 An optimal schedule for the flexible flow shop with no in-process buffers.

Figures 1.13 and 1.14, the heuristic schedule length for the more severe case, that is, with no in-process buffers, is shorter. The result of the RITM-NS heuristic outperforms that of the RITM heuristic in terms of the shorter makespans obtained. Such anomalies are sometimes encountered when heuristic algorithms are applied for scheduling, e.g., Blażewicz et al. (1994). As the machine blocking phenomenon may occur more frequently, the RITM heuristic mechanism for the selection of successive parts for loading may perform better when flexible flow shops with no in-process buffers are scheduled. For the latter case the machine blocking times and the resulting total idle times along the best processing routes are more often positive and may significantly differ for different part types. This allows for a better selection of successive part types for loading and makes the solution space smaller (Sawik, 1993, 1994, 1995b). In contrast, the worst performance of the RITM heuristic can be expected when it is applied for scheduling flexible flow shops with unlimited in-process buffers, where no machine blocking occurs, so that successive part types for loading cannot be clearly determined using the proposed mechanism.

1.4 SCHEDULING FLOW SHOPS WITH LIMITED MACHINE AVAILABILITY

In the scheduling of flexible flow shops the common assumption that all machines are continuously available for processing throughout the scheduling horizon does not apply if certain prescheduled downtime events have to be considered. In this section the assumption that all machines are continuously available for processing throughout the scheduling horizon has been restricted and limited machine availability is allowed.

The interval of machine nonavailability (scheduled downtime) is defined to be a period of time when the machine is not available to perform its intended function due to planned downtime events. The scheduled downtime state of the machine may include various planned events such as preventive maintenance, production tests,

change of consumables/chemicals, setups, or prescheduled production runs to be completed during the upcoming periods.

Model *FPB* can be easily enhanced for scheduling with limited machine availability where machine downtimes are viewed as dummy parts preassigned to time intervals and machines. Each dummy part requires processing on one machine only. Then the real parts that must be completed in minimum time can be processed within the remaining free processing intervals.

Let $L \subset K$ be the set of dummy parts representing machine downtimes and let p_{il}, $a_{ijl} - p_{il}$, and a_{ijl} denote, respectively, the duration, the beginning, and the end of downtime $l \in L$ on machine $j \in J_i$ in stage i.

Model FPBD: *Scheduling Flow Shops with Parallel Machines, Finite In-Process Buffers, and Machine Downtimes*

Minimize (1.1) subject to

1. *Part Completion Constraints*: (1.2), (1.3)
2. *Part Departure Constraints*: (1.14), (1.15)
3. *No Buffering Constraints*: (1.16)
4. *Part Noninterference Constraints*: (1.20), (1.21)
5. *Processor Assignment Constraints*:
   - in every stage with parallel processors each real part is assigned to exactly one processor,
   - each dummy part is preassigned to an appropriate machine,

$$\sum_{j \in J_i} x_{ijk} = 1; \ i \in I, k \in K \backslash L \tag{1.49}$$

$$x_{ijk} = 1; \ i \in I, j \in J_i, k \in L: a_{ijk} > 0 \tag{1.50}$$

6. *Dummy Parts Completion and Departure Constraints*:
   - the completion and departure times of each dummy part are prefixed,

$$c_{ik} = a_{ijk}; \ i \in I, j \in J_i, k \in L: a_{ijk} > 0 \tag{1.51}$$

$$d_{ik} = a_{ijk}; \ i \in I, j \in J_i, k \in L: a_{ijk} > 0 \tag{1.52}$$

7. *Stages Bypassing Constraints by Dummy Parts*:
   - each dummy part bypasses all the stages that are not in its processing route,

$$c_{ik} = c_{i-1k}; \ i \in I, k \in L: i > 1, p_{ik} = 0 \tag{1.53}$$

$$d_{ik} = d_{i-1k}; \ i \in I, k \in L: i > 1, p_{ik} = 0 \tag{1.54}$$

$$x_{ijk} = 0; \ i \in I, j \in J_i, k \in L: a_{ijk} = 0 \tag{1.55}$$

8. *Maximum Completion Time Constraints*:
   - the schedule length is determined by the latest completion time of some real part in the last stage,
   - the maximum completion time is bounded from below by the average workload of each processing stage and the minimum processing time of

a part in all remaining stages,

$$c_{mk} \leq C_{\max};\ k \in K \backslash L \tag{1.56}$$

$$C_{\max} \geq \sum_{k \in K \backslash L} p_{ik}/m_i + \min_{k \in K \backslash L} \left(\sum_{h<i} p_{hk} \right) + \min_{k \in K \backslash L} \left(\sum_{h>i} p_{hk} \right);\ i \in I. \tag{1.57}$$

9. *Variable Nonnegativity and Integrality Conditions*: (1.7) to (1.9), (1.13), (1.19).

The mixed integer program *FPBD* is a general formulation and allows for scheduling with an arbitrary pattern of machine availability.

1.5 COMPUTATIONAL EXAMPLES

In this section numerical examples and some computational results are presented to illustrate possible applications of the MIP models proposed for scheduling flexible flow shop with continuous or with limited machine availability.

The flexible flow shop configuration for the examples is provided in Figure 1.15 and it represents the front of a surface mount technology line for printed wiring board assembly in electronics manufacturing (for more details, see Chapter 2). The flow shop consists of $m = 5$ stages, where stage $i = 1$ is a single machine for screen printing, each stage $i = 3, 5$ represents two parallel machines for automatic placement of components, and each stage $i = 2, 4$ represents two intermediate buffers.

The production order consists of $n = 30$ parts of three types, 10 parts of each type. The processing times for each part type are shown below (for the buffer stages $i = 2, 4$ all processing times are equal to zero)

$$\begin{bmatrix} 10, & 10, & 10 \\ 0, & 0, & 0 \\ 56, & 59, & 74 \\ 0, & 0, & 0 \\ 53, & 54, & 55 \end{bmatrix}$$

The processing schedules were determined for the following two cases:

- *Batch scheduling*, where 10 parts of each type are assembled and the parts of a given type are scheduled consecutively. The optimal sequence of part types is obtained along with the optimal schedule for all parts.

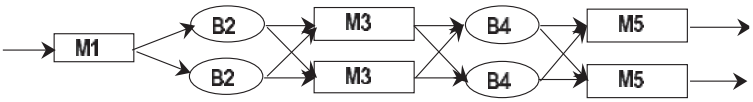

Figure 1.15 A flexible flow shop with parallel machines and finite in-process buffers.

- *Cyclic scheduling*, where 10 parts of each type are assembled and the parts of different types are scheduled alternately in a cyclic order. The optimal cycle of part types is obtained along with the optimal schedule for all parts.

1.5.1 Scheduling with Continuous Machine Availability

In this subsection the optimal schedules are presented for the example problem with continuous machine availability.

The processing schedules obtained are shown on Gantt charts in Figure 1.16, where parts of types 1, 2, and 3 are indicated with different shading. Processor blocking is indicated with a narrow bar. The optimal sequence of part types for batch scheduling is 3, 2, 1, and for cyclic scheduling 1, 2, 3. The solution values obtained are as

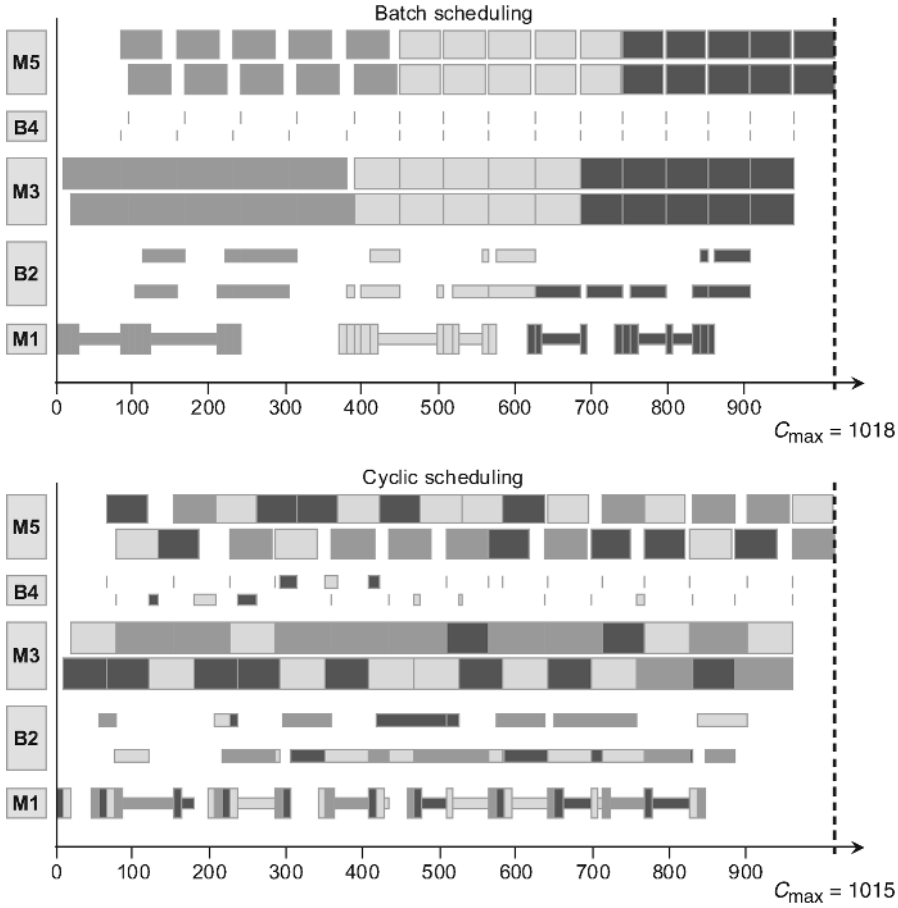

Figure 1.16 Processing schedules with no downtime.

follows: $C^*_{\max} = 1018$ for batch scheduling, $C^*_{\max} = 1015$ for cyclic scheduling. Model *FPB* was used to find the schedules.

1.5.2 Scheduling with Limited Machine Availability

In this subsection processing schedules with limited machine availability are presented for the example problem with a single interval of nonavailability on one machine only. In the example a nonavailable machine was either a single machine in stage 1 or parallel machine 1 in stage 3 or parallel machine 1 in stage 5, and the downtime state occurs in one of the following time intervals: [0,400), [400,800), [800,1200).

Three examples of processing schedules with downtime [400,800) obtained for three different locations in the line of nonavailable machines are shown on Gantt charts in Figures 1.17 to 1.19. Both batch and cyclic scheduling modes are illustrated. Model *FPBD* was used to find the schedules.

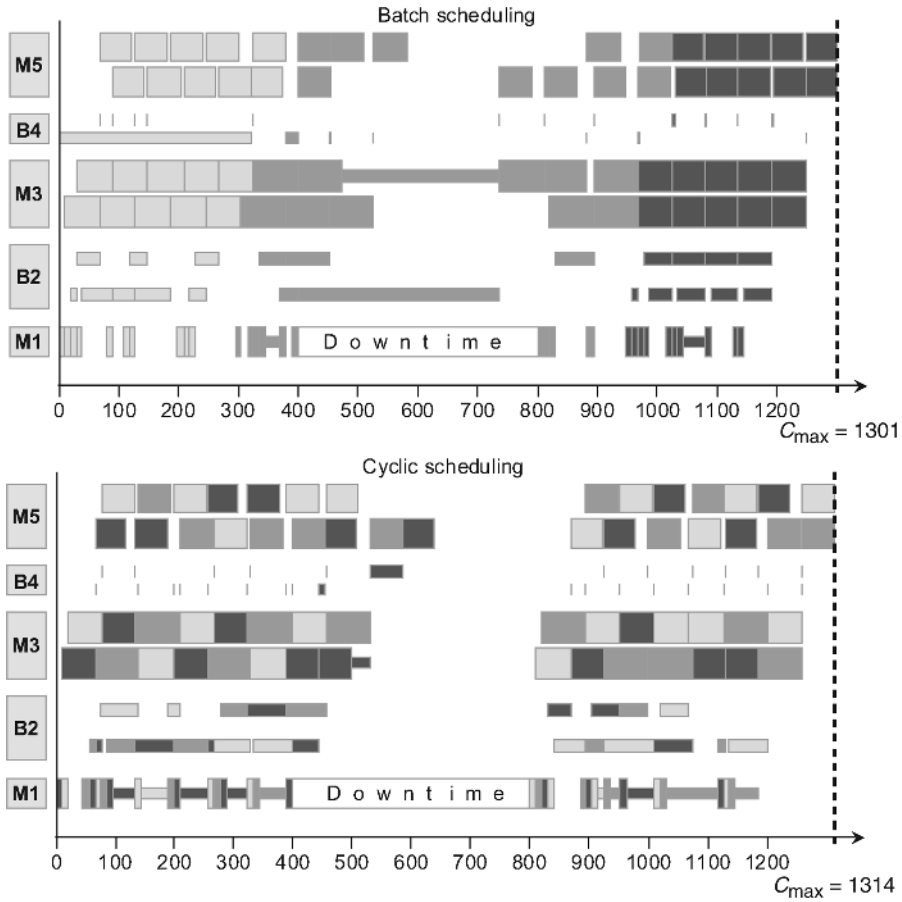

Figure 1.17 Processing schedules with downtime [400,800) in stage 1.

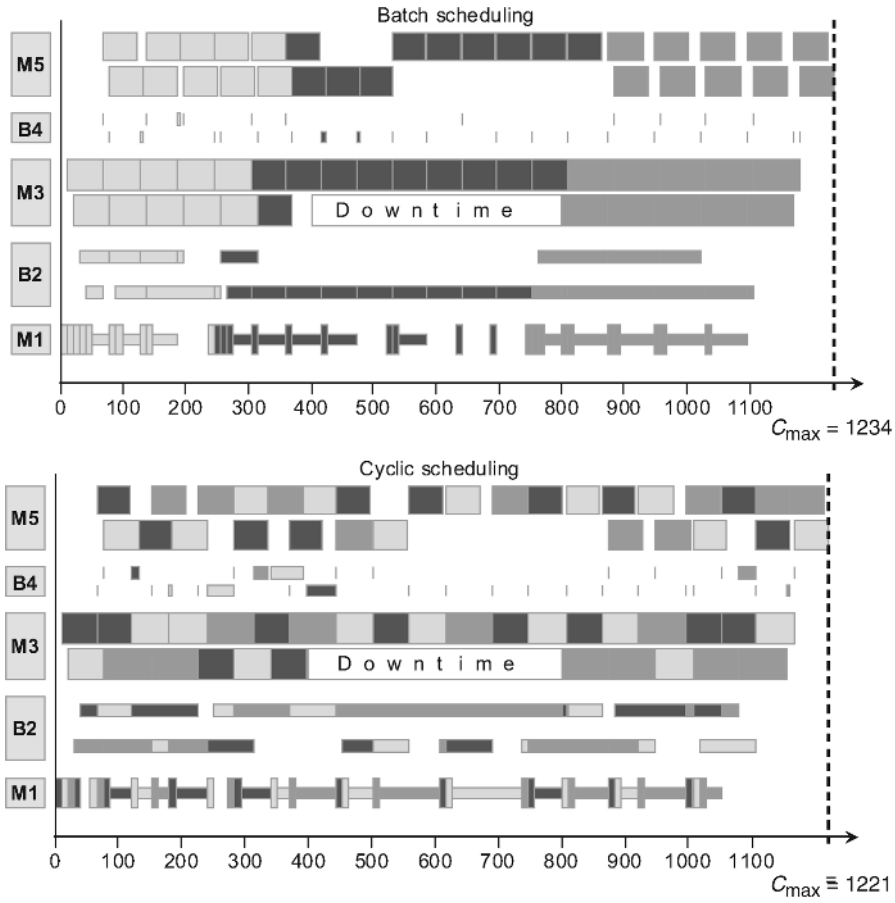

Figure 1.18 Processing schedules with downtime [400,800) in stage 3.

The characteristics of mixed integer programs *FPB* and *FPBD* for the example problems corresponding to Gantt charts in Figures 1.16 to 1.19, and the solution results are summarized in Table 1.4. The size of MIP models for the example problems is represented by the total number of variables, *Var.*, number of binary variables, *Bin.*, number of constraints, *Cons.*, and number of nonzero coefficients, *Nonz.*, in the constraint matrix. The last two columns of Table 1.4 give the lower bound *LB* and the makespan $C^*_{\max}$ or $C^d_{\max}$, respectively, for continuous or limited machine availability.

Table 1.5 shows the relative increase of makespan $(C^d_{\max} - C^*_{\max})/C^*_{\max}$ due to limited machine availability for various locations in the line of the downtime state. The computational results presented in Table 1.5 indicate that the relative increase of makespan depends on the position in the line of the nonavailable machine and position in the schedule of the corresponding interval of nonavailability. The more upstream the position of the nonavailable machine and the earlier the downtime state occurs, the greater is the relative increase of makespan. For a downstream machine, however, a later downtime state results in a greater relative increase of makespan.

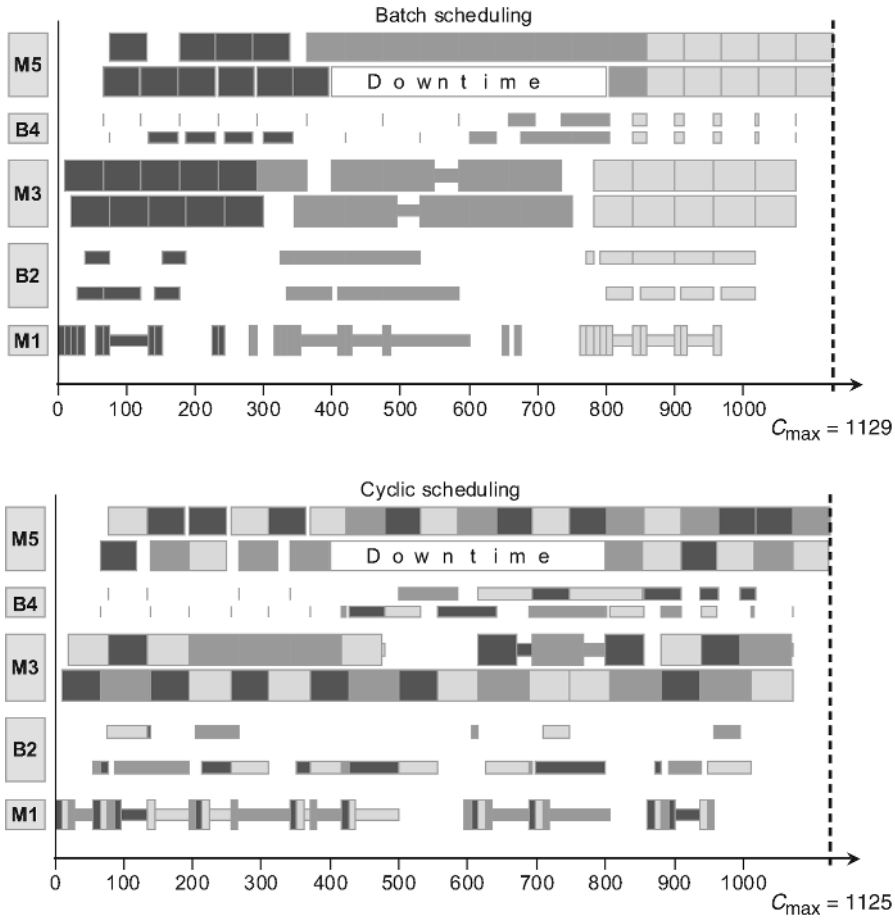

Figure 1.19 Processing schedules with downtime [400,800) in stage 5.

Table 1.4 Characteristics of *FPB*/*FPBD* Models and Solution Results

Problem	Var.	Bin.	Cons.	Nonz.	*LB*	C_{max}
Figure 1.16/Batch	841	540	8581	37,164	1008	1018
Figure 1.16/Cyclic	787	486	8311	35,544	1008	1015
Figure 1.17/Batch	871	570	8502	36,894	1232	1301
Figure 1.17/Cyclic	817	516	8178	35,116	1232	1314
Figure 1.18/Batch	871	570	8637	37,224	1208	1234
Figure 1.18/Cyclic	817	516	8367	35,604	1208	1221
Figure 1.19/Batch	871	570	8761	37,615	1076	1129
Figure 1.19/Cyclic	817	516	8491	35,965	1076	1125

Table 1.5 Relative Increase of Makespan

Downtime Scheduling Mode	$(C^d_{\max} - C^*_{\max})/C^*_{\max}[\%]$					
	Beginning: 0–400		Middle: 400–800		Final: 800–1200	
	Batch	Cyclic	Batch	Cyclic	Batch	Cyclic
Downtime in Stage 1	39.33	39.21	27.92	28.82	30.68	32.25
Downtime in Stage 3	20.06	19.02	21.93	20.49	17.60	15.29
Downtime in Stage 5	6.69	7.45	10.91	10.29	13.37	16.86

The computational experiments with various flow shop configurations have indicated that relative increase of makespan due to the downtime state depends on the location and capacity of the intermediate buffers as well as the relative position in the line of the stage with a nonavailable machine and the bottleneck stage with the largest workload. Sufficient capacity of in-process buffers immediately preceding and succeeding the downtime machine may significantly reduce the relative makespan increase. For instance, if in the example problem, capacity m_2 of the buffer stage 2 (see Fig. 1.15) increases from 2 to 12, then the downtime [400,800) of machine in stage 1 is fully compensated and the makespan decreases from 1301 to 1018 and from 1314 to 1015, respectively for batch and cyclic scheduling (see, Fig. 1.20). The additional buffer capacity results in the decrease of schedule length to the corresponding optimal value attained for continuous machine availability.

1.6 COMMENTS

MIP or IP models for scheduling, in particular flow shop scheduling, were originally devised around 1960. The models for the regular flow shop scheduling problem with single machines, unlimited buffers between successive machines, and permutation schedules can be divided into two broad classes depending on the type of variables and constraints used to assign jobs to various sequence positions. Wagner (1959) proposed IP formulation with sequence-position binary variables and machine idle times before and after processing jobs in each sequence position, whereas Manne (1960) introduced an alternative approach based on sequencing (precedence) variables and job noninterference disjunctive constraints, or their algebraic equivalents, to control the assignment of jobs to various sequence positions. The Manne model assures that only one of each pair of constraints can hold, so that job k either precedes job l somewhere in the processing sequence, or it does not, thus implying that job l precedes job k. For the assignment of jobs to sequence positions, Wilson (1989) uses the classic assignment problem and inequality constraints similar to those in the Manne model, to insure that the start of each job on each machine is no earlier than its finish on the previous machine and that the job in each position in the sequence does not start on a machine until the job in the preceding position in the sequence has completed its

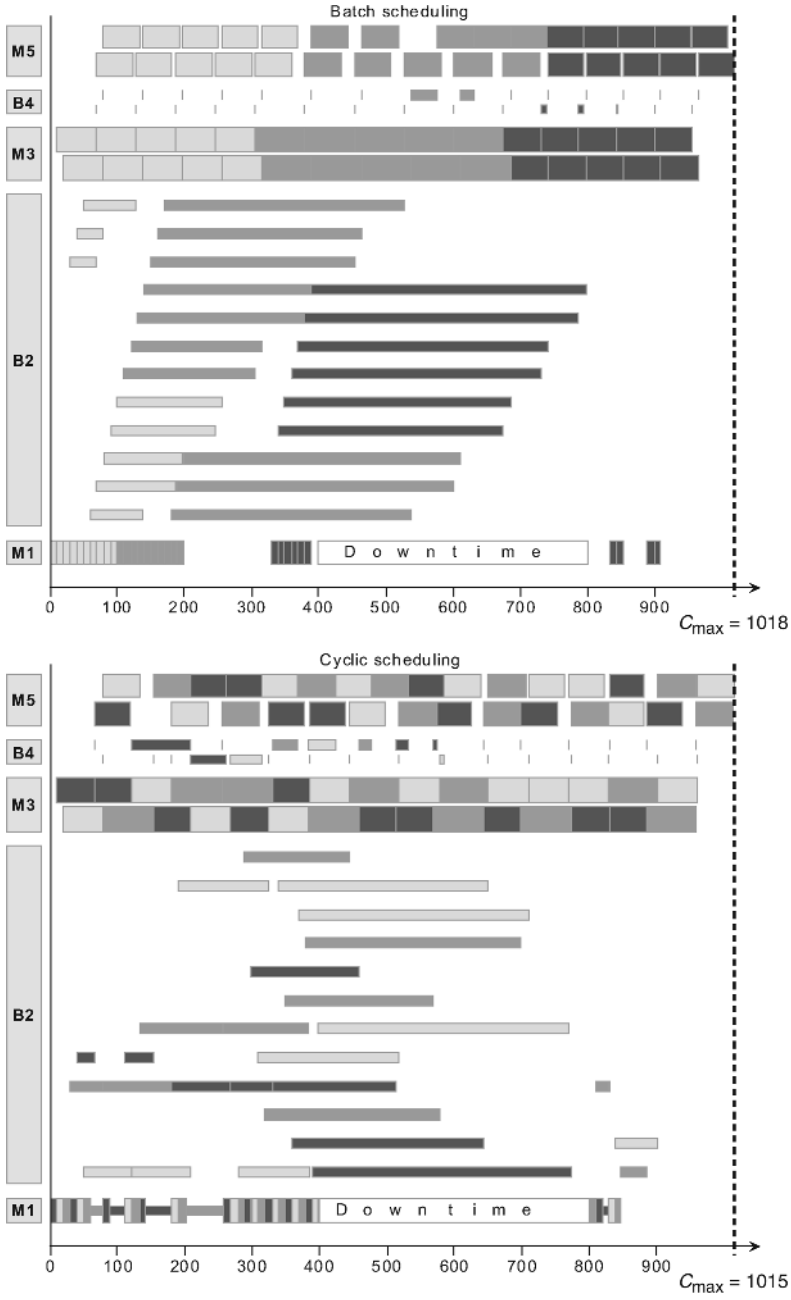

Figure 1.20 Processing schedules with downtime [400,800) in stage 1 and 12 buffers in stage 2.

processing on that machine. The sequence-position models are considered to be more efficient than the precedence models, for example, Tseng et al. (2004).

The first research papers about flexible flow shop scheduling appeared around 1970. Salvador (1973) published one of the pioneer papers on flexible flow shop scheduling by modeling the production system in the synthetic fibers industry as a no-wait scheduling of a flexible flow shop. Garey and Johnson (1979) showed that the flexible flow shop scheduling problem with makespan objective is NP-complete, and over the years the NP nature of most flexible flow shop scheduling problems has been shown. Therefore, a large number of heuristics and approximation algorithms have been proposed for scheduling different flexible flow shop configurations.

Blocking scheduling (e.g. Abadi et al., 2000) has received considerable attention from the study of McCormick et al. (1989), who consider a regular flow shop with finite capacity buffers between machines. A unified modeling approach has been adopted with the buffers viewed as machines with zero processing times to convert the flow shop scheduling problem with buffers into one with no buffers but with blocking, for example, McCormick et al. (1989) and Sawik (2000a).

The literature on cyclic (periodic) scheduling of Minimal Part Set (MPS) in flow shops is concentrated on the study of unpaced (with no exogenous limit on the cycle time) asynchronous (e.g., McCormick et al., 1989; Deane and Moon, 1992, Karabati and Kouvelis, 1996) or synchronous (e.g., Kouvelis and Karabati, 1999) configurations. MPS scheduling in paced assembly lines in which the cycle time is set exogenously, has been discussed in the book by Scholl (1998).

Double-pass reentrant flow shops, where a part visits a set of stages more than once were considered by various researcher, for example, Graves et al. (1983) and Tirpak (2000).

During the last decade, research in this area has focused on more realistic problems, including sequence-dependent set ups on machines (e.g., Jain et al., 1996; Liu and Chang, 2000; Kurz and Askin, 2004), limited availability of machines (e.g., Schmidt, 2000), etc. Reviews on flexible flow shop scheduling can be found in Vignier et al. (1999) or Linn and Zhang (1999). Kis and Pesch (2005) review exact methods for flexible flow shop scheduling problem with parallel identical machines to minimize makespan or total flow time, while H. Wang (2005) classifies the papers according to the solution procedure adopted (optimal, heuristics, and artificial intelligence). Finally, Quadt and Kuhn (2007) propose a taxonomy for flexible flow shop scheduling procedures focusing on heuristic procedures. The most recent comprehensive reviews can be found in Ribas et al. (2010) and Ruiz and Vazquez-Rodriguez (2010).

Introduction of limited machine availability constraints further complicates a flexible flow shop blocking scheduling problem. For example, the regular flow shop scheduling problem for two machines that can be solved in polynomial time by Johnson's rule (Johnson, 1954) becomes already NP-complete if there is a single interval of nonavailability on one machine only (see, Schmidt, 2000).

It can be observed that a lot of research work on scheduling flexible flow shops considers makespan minimization, while other common criteria, such as flow time (e.g., Rajendran and Chaudhuri, 1992; Guinet and Solomon 1996) or total tardiness

(e.g., G. C. Lee and Kim, 2004; Sawik, 2005a) are less studied. Also it is observed that little research has been done concerning multicriteria problems, for example, Alfieri (2009).

With respect to solution approaches, branch and bound and MIP are the most frequently used exact procedures. At the time of Brah and Hunsucker (1991), instances of a very limited size with up to eight jobs and two stages with three parallel machines each could be solved within several hours of CPU time. More recently, the best branch and bound algorithms proposed are efficient to solve instances with 15 or 20 jobs and at most five stages. It can also be observed that the scheduling problems become more complex when the number of machines increases and when there is no single bottleneck stage. When a single bottleneck exists, then tight lower and upper bounds restrict the search effort required to find an optimal solution.

The constructive heuristics RITM and RITM-NS presented in Section 1.3 for scheduling flexible flow shops with machine blocking were developed by Sawik (1993, 1994, 1995b, 1999). The two algorithms belong to the best available constructive heuristics for scheduling flexible flow shops with machine blocking, for example, Nowicki and Smutnicki (1996) and Ribas et al. (2010).

EXERCISES

1.1 Denote by $m \leq \overline{m}$, the variable number of processing stages, not greater than $\overline{m}$, of a flow shop with single machines and infinite in-process buffers and by $\overline{C}$, an upper bound on the schedule length. Formulate a mixed integer program for finding the minimum-length flow shop such that a given set of part types is completed by the common due date $\overline{C}$.

1.2 Denote by $m_i \leq \overline{m}_i$, the variable number of parallel machines in stage $i = 1, \ldots, m$, not greater than $\overline{m}_i$, of an m-stage flow shop with parallel machines and infinite in-process buffers and by $\overline{C}$, an upper bound on the schedule length. Formulate a mixed integer program for finding the flow shop with minimum total number of parallel machines such that a given set of part types is completed by the common due date $\overline{C}$.

1.3 Consider batch scheduling in a flexible flow shop with parallel machines and finite in-process buffers, in which a different due date is given for each part type. Formulate the mixed integer program for scheduling the batches of part types to minimize the number of tardy part types.

1.4 In the computational examples in Section 1.3.3, Figures 1.13 and 1.14 indicate that the makespan of optimal schedule is identical for the flexible flow shops with finite in-process buffers and with no in-process buffers. Explain why the optimal makespans are identical.

1.5 Consider problem *FPBD* of scheduling a flow shop with parallel machines, finite in-process buffers, and machine downtimes. Suppose that for a particular problem instance no feasible solution exists with the schedule length not longer than a given bound $\overline{C}$, however, one may obtain a feasible solution by increasing the capacity of in-process buffering. Let $m_i \leq \overline{m}_i$ be the variable number of in-process buffers in stage $i \in I$ (such that $\sum_{k \in K} p_{ik} = 0$), not greater than $\overline{m}_i$. Formulate the mixed integer program for scheduling a flow shop with parallel machines, variable capacity of in-process buffers, and machine downtimes, such that the schedule length is not longer than $\overline{C}$ and the total capacity of in-process buffers is minimized.