SCHEDULING OF PRINTED WIRING BOARD ASSEMBLY IN SURFACE MOUNT TECHNOLOGY LINES BY MIXED INTEGER PROGRAMMING

Tadeusz Sawik

University of Mining & Metallurgy
Department of Computer Integrated Manufacturing
Al.Mickiewicza 30, 30-059 Kraków, POLAND
tel: +48 12 617 39 92
ghsawik@cyf-kr.edu.pl

Abstract: *The paper presents a new mixed integer programming approach for scheduling of printed wiring board assembly in Surface Mount Technology (SMT) lines. A typical SMT line consists of several assembly stations in series and/or in parallel, separated by finite intermediate buffers. The objective of the scheduling problem is to determine the detailed sequencing and timing of all assembly tasks for each individual board, so as to maximize the line's productivity, which is defined in terms of makespan for a mix of board types. The limited intermediate buffers between stations result in a scheduling problem with machine blocking, where a completed board may remain on a machine and block it until a downstream machine becomes available. The proposed model can be used for optimization of assembly schedules with continuous or with limited machine availability by using commercially available software for integer programming. Numerical examples illustrate the approach.*

Keywords: *Electronics manufacturing, flexible assembly lines, scheduling, mixed integer programming.*

1 Introduction

Printed wiring board (PWB) assembly is typically performed on an automated Surface Mount Technology (SMT) line which includes three different processes in the following sequence: solder printing, component placement and solder reflow. A typical SMT line consists of several assembly stations in series and/or in parallel, separated by finite intermediate buffers and connected with a conveyor system that transfers the boards between the stations.

An example of SMT line with parallel stations is shown in Figure 1. The line consists of a PWB loader, a solder printer, two parallel placement machines for small components and two additional shuttles routing the PWB to the next available placement machine, one placement machine for fine pitch components and a reflow oven. The assembly process is as follows: A tote of bare (pre-assembly) PWBs is brought to the beginning of the line, and a material loader loads each PWB separately on the conveyor. Each PWB is transported by the conveyor system through each processing stage in the line and then is stored again in a tote box. The loader and the tote box are used as the input and output buffers of the line. There are external buffers in front of and behind each placement machine, except the last one. In addition, every placement machine has its own internal input and output buffers of a fixed capacity.

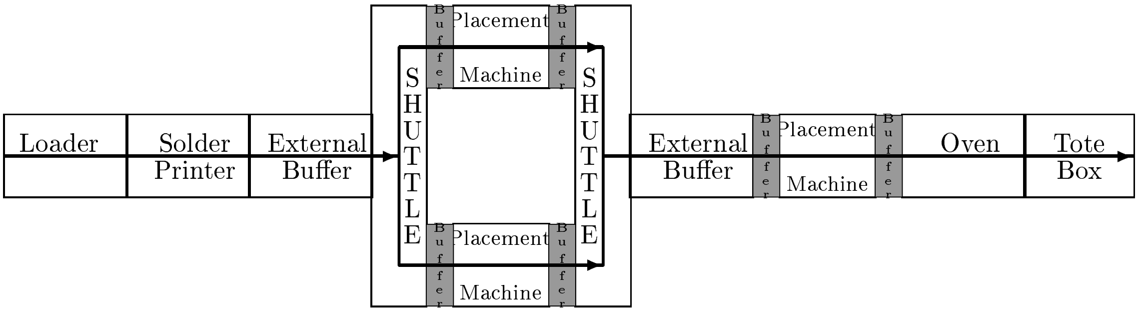

Figure 1: An SMT line with parallel stations

Various configurations of SMT lines can be encountered in the electronics assembly, e.g. [9]. For example, the single-pass lines, where one-pass through the line is required to complete a board or the double-pass reentrant lines, where the double-sided boards run twice through the same line, first to assemble the bottom side and then to assemble the top side.

An SMT line is a practical example of a flexible flow line with limited intermediate buffers and parallel machines, e.g. [7]. The line produces several different board types. Each board must be processed by at most one machine in each stage. A board which has completed processing on a machine in some stage is transferred either directly to an available machine in the next stage or to a buffer ahead of that stage. The limited intermediate buffers between stations result in a *blocking scheduling* problem, e.g. [1,2,5], where a completed board may remain on a machine and block it until a downstream machine becomes available. This prevents another board from being processed on the blocked machine.

In the scheduling of SMT lines 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 [3]. Examples of such events include preventive maintenance, production tests, change of component feeders, or prescheduled production runs to be completed during the upcoming periods. The objective is to determine an assembly schedule within the remaining free processing intervals for a mix of board types so as to complete the boards in minimum time.

Introduction of limited machine availability constraints may further complicate an SMT line scheduling problem. For example, the classical flowshop scheduling problem for two machines that can be solved in polynomial time by Johnson's rule becomes already NP-complete if there is a single interval of non-availability on one machine only [10].

This paper provides the reader with mixed integer programming formulations for scheduling flexible flow lines with finite capacity buffers and with continuous or limited machine availability. While integer programming formulations have been widely used to express the assembly line design and balancing problems, e.g. [6,7], their application in a much more complex problem of scheduling flexible flow lines such as SMT lines is very limited, e.g. [1,4,8].

The formulation presented in this paper can be applied for constructing assembly schedules by using commercially available software for mixed integer programming. This has been illustrated in the paper with numerical examples. The example problems have been modelled using an advanced algebraic modelling language AMPL with CPLEX solver.

The paper is organized as follows. In the next section mixed integer programming formulation is presented for blocking scheduling of a flexible flow line with limited machine availability. Numerical examples and some computational results are provided in Section 3, and conclusions are given in the last section.

2 Mixed integer program for scheduling flexible flow lines with machine blocking and downtimes

In this section a mixed integer programming model is presented for scheduling of a flexible flow line with limited intermediate buffers and limited machine availability. The interval of machine non-availability (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.

A unified modelling approach is adopted with the buffers viewed as machines with zero processing times. As a result the scheduling problem with buffers can be converted into one with no buffers but with blocking, e.g. [5,8]. The blocking time of a machine with zero processing time denotes board waiting time in the buffer represented by that machine. We assume that each board must be processed in all stages, including the buffer stages. However, zero blocking time in a buffer stage indicates that the corresponding board does not need to wait in the buffer. Let us note that for each buffer stage board completion time is equal to its departure time from the previous stage since the processing time is zero.

Notation used to formulate the problems is shown in Table 1, where buffers and machines are referred to as processors.

The flexible flow line under study consists of m processing stages in series. Each stage i, $(i = 1, \ldots, m)$ is made up of $n_i \geq 1$ identical parallel processors. The system produces various types of boards. Each board must be processed without preemption on exactly one processor in each of the stages sequentially. That is, each board must be processed in stage 1 through stage m in that order. The order of processing the boards in every stage is identical and determined by an input sequence in which the boards enter the line, i.e., a so-called permutation flowshop is considered.

Let $p_{ik} \geq 0$ be the processing time in stage i of board $k \in K$. For every board k denote by c_{ik} its completion time in each stage i, and by d_{ik} its departure time from stage i.

Processing without preemption indicates that board k completed in stage i at time c_{ik} had started its processing in that stage at time $c_{ik} - p_{ik}$. Board 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 n_{i+1} processors in stage $i+1$ are occupied, then the processor in stage i is blocked by board k until time $d_{ik} = c_{i+1k} - p_{i+1k}$ when board k starts processing on an available processor in stage $i+1$.

Table 1: Notation

		Indices
i	=	processing stage, $i \in I = \{1, \ldots, m\}$
j	=	processor in stage i, $j \in J_i = \{1, \ldots, n_i\}$
k	=	board, $k \in K = \{1, \ldots, v\}$
		Input parameters
b_{ijl}	=	the end of lth downtime $[b_{ijl} - p_{il}, b_{ij})$ of processor j in stage i
m	=	number of processing stages
n_i	=	number of parallel processors in stage i
p_{ik}	=	processing time for board k in stage i
v	=	number of boards to be completed
L	=	subset of preassigned dummy boards representing downtimes, $L \subset K$
Q	=	a large number not less than schedule length
		Decision variables
C_{max}	=	schedule length
c_{ik}	=	completion time of board k in stage i
d_{ik}	=	departure time of board k from stage i
x_{ijk}	=	1, if board k is assigned to processor $j \in J_i$ in stage $i \in I$; otherwise $x_{ijk} = 0$
y_{kl}	=	1, if board k precedes board l; otherwise $y_{kl} = 0$

Machine downtimes are viewed as dummy boards preassigned to time intervals and machines. Each dummy board requires processing on one machine only. The real boards can be assembled within the remaining free processing intervals.

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

The objective is to determine an assignment of the real boards to processors and the remaining free processing intervals in each stage in such a way as to complete all real boards in minimum time, that is, to minimize the makespan $C_{max} = \max_{k \in K: k \notin L}(c_{mk})$, where c_{mk} denotes the completion time of board k in the last stage m.

The mixed integer program **FBD** for scheduling flexible flow line with processor blocking and machine downtimes is presented below.

Model FBD: *Scheduling flexible flow line with finite in-process buffers and machine downtimes*

Minimize

$$C_{max} \tag{1}$$

subject to

Assignment constraints for stages with parallel processors

$$\sum_{j \in J_i} x_{ijk} = 1;\ i \in I, k \in K : n_i > 1, k \notin L \tag{2}$$

$$x_{ijk} = 1;\ i \in I, j \in J_i, k \in L : n_i > 1, b_{ijk} > 0 \tag{3}$$

$$x_{ijk} = 0;\ i \in I, j \in J_i, k \in L : n_i > 1, b_{ijk} = 0 \tag{4}$$

Processing stage bypassing constraints for dummy boards

$$c_{ik} = c_{i-1k};\ i \in I, k \in L : i > 1, p_{ik} = 0 \tag{5}$$

$$d_{ik} = d_{i-1k};\ i \in I, k \in L : i > 1, p_{ik} = 0 \tag{6}$$

Board completion constraints

$$c_{1l} \geq p_{1l} + \sum_{k \in K : k < l} p_{1k} y_{kl} + \sum_{k \in K : k > l} p_{1k}(1 - y_{lk});\ l \in K : n_1 = 1 \tag{7}$$

$$c_{1k} \geq p_{1k};\ k \in K : n_1 > 1 \tag{8}$$

$$c_{ik} - c_{i-1k} \geq p_{ik};\ i \in I,\ k \in K : i > 1 \tag{9}$$

Dummy boards completion constraints

$$c_{ik} = b_{ijk};\ i \in I, j \in J_i, k \in L : b_{ijk} > 0 \tag{10}$$

$$d_{ik} = b_{ijk};\ i \in I, j \in J_i, k \in L : b_{ijk} > 0 \tag{11}$$

Board non-interference constraints for stages with single processors

$$c_{ik} + Q y_{kl} \geq d_{il} + p_{ik};\ i \in I, k, l \in K : k < l, n_i = 1 \tag{12}$$

$$c_{il} + Q(1 - y_{kl}) \geq d_{ik} + p_{il};\ i \in I, k, l \in K : k < l, n_i = 1 \tag{13}$$

Board non-interference constraints for stages with parallel processors

$$c_{ik} + Q(2 + y_{kl} - x_{ijk} - x_{ijl}) \geq d_{il} + p_{ik};\ i \in I, j \in J_i, k, l \in K : k < l, n_i > 1 \tag{14}$$

$$c_{il} + Q(3 - y_{kl} - x_{ijk} - x_{ijl}) \geq d_{ik} + p_{il};\ i \in I, j \in J_i, k, l \in K : k < l, n_i > 1 \tag{15}$$

No-store constraints

$$c_{ik} = d_{i-1k} + p_{ik};\ i \in I, k \in K : i > 1 \tag{16}$$

Maximum completion time constraints

$$c_{mk} = d_{mk};\ k \in K \tag{17}$$

$$c_{mk} \leq C_{max};\ k \in K : k \notin L \tag{18}$$

$$C_{max} \geq \sum_{k \in K : k \notin L} p_{ik}/n_i + \sum_{h \in I : h \neq i} min_{k \in K : k \notin L}(p_{hk});\ i \in I \tag{19}$$

Variable elimination constraints

$$y_{kl} = 0;\ k, l \in K : k \geq l \tag{20}$$

Variable nonnegativity and integrality constraints

$$c_{ik} \geq 0;\ i \in I, k \in K \tag{21}$$

$$d_{ik} \geq 0;\ i \in I, k \in K \tag{22}$$

$$x_{ijk} \in \{0,1\};\ i \in I, j \in J_i, k \in K \tag{23}$$

$$y_{kl} \in \{0,1\};\ k, l \in K \tag{24}$$

The objective function (1) represents the schedule length to be minimized. Assignment constraint (2) ensures that in every stage with parallel processors each real board is assigned to exactly one processor, (3) preassigns each dummy board to an appropriate machine, while (4) along with (5) and (6) allow each dummy board to bypass all stages that are not in its processing route. Constraint (7) or (8) ensures that each board is processed in the first stage, and (9) guarantees that it is also processed in all downstream stages. Constraints (10) and (11) prefix completion and departure times of dummy boards. Constraints (12) and (13) are board non-interference constraints for single processors, and (14) and (15) for parallel processors. No two boards can be performed on the same processor simultaneously. For a given sequence of boards only one constraint of each pair (12) and (13) or (14) and (15) is active, and only if both boards k and l are assigned to the same processor. Equation (16) indicates that processing of each board in every stage starts immediately after its departure from the previous stage. Equation (17) ensures that each board leaves the line as soon as it is completed in the last stage. Finally, (18) defines the maximum completion time of all real boards, and (19) imposes a lower bound on it.

Mixed integer program **FBD** is a general formulation and allows for scheduling with arbitrary pattern of machine availability. In particular, when $L = \emptyset$ model **FBD** can be used for scheduling with all machines continuously available [7,8].

2.1 Scheduling modes

In order to reduce the complexity of the **general scheduling** problem, where any sequence of boards is allowed, the following scheduling modes can also be considered:

- **Batch scheduling**, where boards of a given type are scheduled consecutively, where in addition:
  - **(a)** the sequence of board types is fixed and equal to the optimal sequence determined for a minimum set of boards (e.g., one board of each type) or
  - **(b)** the sequence of board types is not determined a priori, but is obtained with the optimal schedule for all boards.
- **Cyclic scheduling**, where different board types are scheduled alternately in a cyclic order of board types, where in addition:
  - **(a)** the cycle of board types is fixed and equal to the optimal sequence determined for a minimum set of boards (e.g., one board of each type) or

(b) the cycle of board types is not determined a priori, but is obtained with the optimal schedule for all boards.

Scheduling batches of boards or a cyclic scheduling are more often met in practice than general scheduling. In the example problems presented in the next section, only batch scheduling mode is considered with input sequence of board types not fixed a priori.

3 Numerical examples

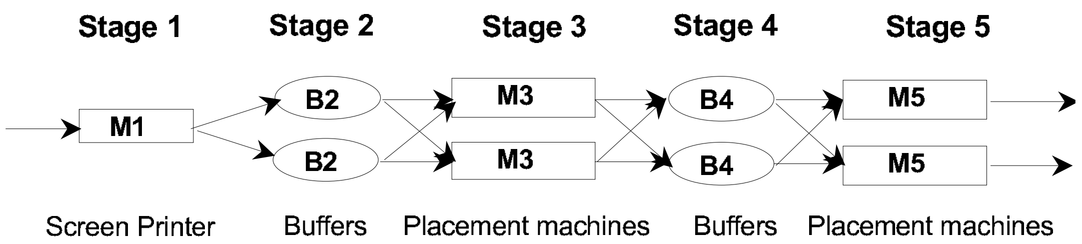

Figure 2: Example of an SMT line

In this section numerical examples are presented to illustrate application of the proposed model for scheduling SMT lines with continuous or with limited machine availability.

The SMT line configuration for the examples is shown in Figure 2. The line consists of $m = 5$ stages, where stage $i = 1$ is a screen printer, each stage $i = 3, 5$ represents 2 parallel machines for automatic placement of components and each stage $i = 2, 4$ represents 2 intermediate buffers. A non-available 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 may occur in one of the following time intervals: [0,400), [400,800), [800,1200).

The production order consists of $v = 30$ boards of 3 types, 10 boards of each type. The processing times for each board type are shown below (for the buffer stages $i = 2, 4$ all processing times are equal to zero)

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

For the example, batch schedules were determined, where the boards of a given type are scheduled consecutively and the sequence of board types was not fixed a priori. The relative increase of makespan $(C^D_{max} - C^*_{max})/C^*_{max}$ due to machine downtime is shown in Table 2, where C^*_{max} and C^D_{max} is the makespan with machines continuously available and with machine downtime, respectively. The relative increase of makespan depends on the position in the line of non-available machine and position in the schedule of the corresponding interval of non-availability. The more upstream is the position of non-available machine and the earlier occurs its downtime state, the greater is relative increase of makespan. For a downstream machine, however, the later downtime state results in the greater relative increase of makespan.

Table 2: Relative Increase of Makespan

	$(C^D_{max} - C^*_{max})/C^*_{max}[\%]$		
Downtime	[0,400)	[400,800)	[800,1200)
in Stage 1	39.33	27.92	30.68
in Stage 3	20.06	21.93	17.60
in Stage 5	6.69	10.91	13.37

Figure 3 shows assembly schedules with machines continuously available and with a single downtime [400,800) in stage 1, stage 3 or stage 5, with makespan $C^*_{max} = 1017$ and $C^D_{max} = 1301$, 1240 or 1128, respectively. In the Gantt charts boards of types 1, 2 and 3 are indicated with gray shading, black shading and cross hatching, respectively. Processor blocking is indicated with a narrow bar.

The assembly schedules for the examples were calculated within 3600 CPU seconds time limit on a laptop Compaq Presario 1830 with Pentium III, 450 MHz using AMPL with CPLEX v.6.5.2 solver. CPLEX was not always able to prove optimality within the allowed 3600 seconds of CPU time. However, in most cases the best solution was found much earlier than the time limit.

The experiments with various features of CPLEX solver to speed up the solution process have indicated that the best results are obtained with a nearly depth-first branch and bound strategy for node selection where a limited backtracking is allowed.

4 Conclusion

The paper shows that mixed integer programming can be used for modelling a hard optimization problem of scheduling flexible assembly lines with limited intermediate buffers and with continuous or with limited machine availability. Computational experiments have indicated (e.g. [8,9]) that the proposed model can be applied for scheduling of printed wiring board assembly in electronics industry, where the most important scheduling decisions include the input sequence of boards entering an SMT line and the assignment of boards to parallel machines or conveyors. The computational effort required to find optimal schedules for realistic problems can be reduced by introducing specific scheduling modes, such as batching or cyclic.

Acknowledgment

This work has been partially supported by AGH and KBN (Poland), and by the Motorola Advanced Technology Center (USA).

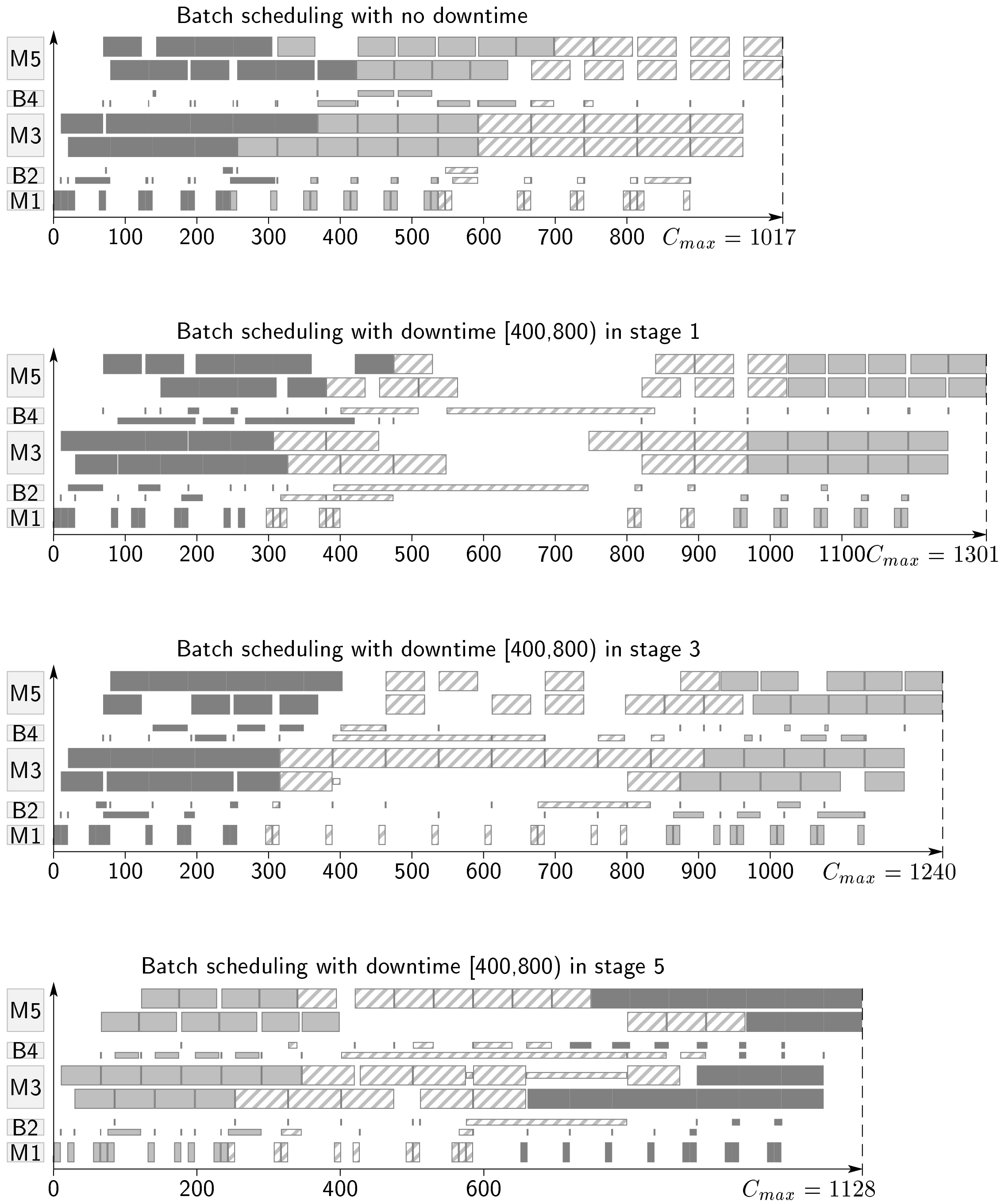

Figure 3: Schedules with machines continuously available and with downtimes

References

[1] Abadi, I.N.K., Hall, N.G., and Sriskandarajah C., Minimizing cycle time in a blocking flowshop. *Operations Research*, **48**, 2000, 177–180.

[2] Hall, N.G. and Sriskandarajah, C., A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, **44**, 1996, 510–525.

[3] Hopp, W. and Spearman, M., *Factory Physics: Foundations of Manufacturing Management*, McGraw-Hill, 1996.

[4] Karabati, S. and Kouvelis, P., Cycle scheduling in flow lines: modeling observations, effective heuristics and a cycle time minimization procedure. *Naval Research Logistics*, **43**, 1996, 211–231.

[5] McCormick, S.T., Pinedo, M.L., Shenker, S., and Wolf, B., Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research*, **37**, 1989, 925–936.

[6] Nemhauser, G.L. and Wolsey, L.A., *Integer and Combinatorial Optimization.* John Wiley, New York 1988.

[7] Sawik, T., *Production Planning and Scheduling in Flexible Assembly Systems.* Springer-Verlag, Berlin 1999.

[8] Sawik, T., Mixed integer programming for scheduling flexible flow lines with limited intermediate buffers. *Mathematical and Computer Modelling,* **31**, 2000, 39–52.

[9] Sawik, T., Schaller, A., and Tirpak, T.M., Issues in loading and scheduling of SMT lines. *Zeszyty Naukowe Politechniki Śląskiej, Automatyka*, no.129, 2000, 331–341.

[10] Schmidt, G., Scheduling with limited machine availability. *European Journal of Operational Research,* **121**, 2000, 1–15.