BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 62, No. 1, 2014
DOI: 10.2478/bpasts-2014-0014

A mixed integer program for cyclic scheduling of flexible flow lines

T. SAWIK*

AGH University of Science & Technology, Department of Operations Research and Information Technology,
30 Mickiewicza Ave., 30-059 Kraków, Poland

Abstract. A new mixed integer programming formulation is presented for cyclic scheduling in flow lines with parallel machines and finite in-process buffers, where a Minimal Part Set (MPS) in the same proportion as the overall production target is repetitively scheduled. The cycle of parts in an MPS is not determined a priori, but is obtained along with the optimal schedule for all parts. In addition to the cyclic scheduling, a cyclic-batch scheduling mode is introduced, where within the MPS the parts of one type are processed consecutively. Numerical examples are included and some results of computational experiments are reported.

Key words: flexible flow line, cyclic scheduling, cyclic-batch scheduling, mixed integer programming.

1. Introduction

A flexible flow line (FFL) consists of several machines in series and/or in parallel, separated by finite in-process buffers and connected with a conveyor system that transfers the parts between the machines. The flow line is capable of processing several different parts types and each part must be processed by at most one machine in each stage. When a machine finishes to process a part and the downstream buffer is full the machine is blocked, while if the upstream buffer is empty the machine is starved. A common assumption is that there are always raw parts available at the input of the system, and hence the first machine is never starved and that there is always a space available at the output of the system and hence the last machine is never blocked. Typical performance measures of a FFL include the throughput or makespan for a mix of part types.

Blocking scheduling has received considerable attention from the study in [1], where a regular flow line with finite capacity buffers between machines is considered. Mixed integer programs for the blocking scheduling in a FFL are presented in [2–4]. A review of exact methods for flexible flow line scheduling problem with parallel identical machines to minimize makespan or total flow time is presented in [5], and the most recent comprehensive review can be found in [6].

The cyclic scheduling is often used when set up times are negligible and the demand for each part type remains constant over the scheduling horizon. Otherwise, the batch scheduling is applied, where it is more efficient to have long runs of identical parts to minimize sequence dependent set up times. Cyclic schedules reduce in-process inventory, are easy to keep track of and impose a certain discipline, and allow a simple shop floor control to be implemented. Therefore, the cyclic scheduling is widely used in practice, e.g., [7, 9].

The purpose of this paper is to provide the reader with a new mixed integer programming monolithic model for cyclic scheduling in a FFL, where a Minimal Part Set (MPS) is repetitively scheduled. MPS is the smallest possible set of parts in the same proportion as the overall production target. 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 1 part A, 3 parts B, and 5 parts C, in total 9 parts, which is to be repeated 100 times. In the cyclic scheduling the parts within an MPS can be ordered as in the general scheduling, e.g., C,C,A,B,C,B,C,B,C or in batches as in the batch scheduling, e.g., C,C,C,C,C,A,B,B,B. The latter scheduling mode can be called cyclic-batch scheduling.

The literature on cyclic scheduling of MPS in flow lines is concentrated on the study of unpaced (with no exogenous limit on the cycle time) asynchronous (e.g., [1, 10]) or synchronous (e.g., [11]) configurations. In contrast to a common two-level approach proposed in the literature (e.g., [7]), based on simple repetition of the optimal schedule for a single MPS, which is determined independently on the overall production target to be to fulfilled, in this paper the optimal cycle of parts in an MPS is not determined a priori, but is obtained along with the optimal schedule for all parts. In addition, the cyclic-batch scheduling mode is modeled, that is, the constrained cyclic mode combined with batch processing of part types in each run of an MPS.

The paper is organized as follows. Mixed integer program for cyclic scheduling in FFL is presented in the next section. Numerical examples are provided in Sec. 3, and conclusions are given in the last section.

2. Problem formulation

In this section mixed integer programming model **CFFL** is presented for cyclic or cyclic-batch scheduling in a FFL. The flow line consists of m processing (machine or buffer) stages $i \in I = \{1, \ldots, m\}$ in series. Buffers and machines are referred to as processors and each processing stage $i \in I$ is made up of $m_i \geq 1$ parallel identical processors. Let J_i be the circular set of parallel processors in stage i, where 'circu-

*e-mail: ghsawik@cyf-kr.edu.pl

lar" indicates that the set is ordered and wraps around, i.e., its first member is the successor of its last, and its last member is the predecessor of its first.

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.

The FFL produces various part types with negligible set up times, and each part must be processed without preemption on exactly one processor in each of the stages sequentially.

Denote by G, $K = \{1, \ldots, n\}$, and $K_g = \{\sum_{f \in G: f<=g-1} n_f + 1, \ldots, \sum_{f \in G: f<=g-1} n_f + n_g\}$ the ordered sets of indices, respectively of all part types, all individual parts, and all parts of type $g \in G$, where n_g and $n = \sum_{g \in G} n_g$ are, respectively the number of parts type g and the total number of parts in the schedule. Denote by $r_{ig} \geq 0$ the processing time in stage i of part type $g \in G$ and by $p_{ik} \geq 0$ the processing time in stage i of part $k \in K$, ($p_{ik} = r_{ig} \forall k \in K_g$), where all processing times are equal to zero for a buffer stage.

The MPS is defined as $\{\underline{n}_g : \ g \in G\}$, where $\underline{n}_g = n_g/S \ \forall \ g \in G$, and S is the greatest common divisor of integers $n_1, n_2, \ldots n_{|G|}$, 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.

The problem objective is to assign parts to processors in each stage and to schedule the parts cyclically so as to minimize the makespan. The decision variables are:

c_{ik} – completion time of part k in stage i;

d_{ik} – departure time of part k from stage i;

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

$y_{kl} = 1$, if part k precedes part l in the processing sequence; otherwise $y_{kl} = 0$.

Model CFFL: Cyclic scheduling of a flexible flow line

Minimize makespan

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

subject to

Part assignment constraints:

– in every stage each part is assigned to exactly one processor,

– in each multi-processor stage i, every $\underline{n}_g$-th part of each type g is assigned to every $\underline{n}$-th processor in the circular set J_i of parallel processors,

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

$$\begin{gathered} x_{i,next(j,J_i,\underline{n}),next(k,K_g,\underline{n}_g)} = x_{ijk}; \\ i \in I, j \in J_i, g \in G, k \in K_g : \\ k \leq last(K_g) - \underline{n}_g, m_i > 1, \end{gathered} \tag{3}$$

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

Part completion constraints:

– each part must be processed in the first stage and successively in all downstream stages,

$$c_{1k} \geq p_{1k}; \ k \in K, \tag{4}$$

$$c_{ik} - c_{i-1k} \geq p_{ik}; \ i \in I, \ k \in K : i > 1. \tag{5}$$

Part departure constraints:

– each part cannot be departed from a stage until it is completed in this stage,

– each part leaves the line as soon as it is completed in the last stage,

$$c_{ik} \leq d_{ik}; \ i \in I, k \in K : i < m, \tag{6}$$

$$c_{mk} = d_{mk}; \ k \in K. \tag{7}$$

No-buffering constraints:

– in every stage processing of each part starts immediately after its departure from the previous stage,

$$c_{ik} - p_{ik} = d_{i-1k}; \ i \in I, k \in K : i > 1. \tag{8}$$

Part non-interference constraints:

– no two parts assigned to the same processor can be processed simultaneously,

$$\begin{gathered} c_{ik} + Q(2 + y_{kl} - x_{ijk} - x_{ijl}) \geq d_{il} + p_{ik}; \\ i \in I, j \in J_i, k, l \in K : k < l, \end{gathered} \tag{9}$$

$$\begin{gathered} c_{il} + Q(3 - y_{kl} - x_{ijk} - x_{ijl}) \geq d_{ik} + p_{il}; \\ i \in I, j \in J_i, k, l \in K : k < l, \end{gathered} \tag{10}$$

where Q is a large positive constant not less than the schedule length.

Maximum completion time constraints:

– the schedule length is determined by the latest completion time of some part in the last stage,

$$c_{mk} \leq C_{max}; \ k \in K. \tag{11}$$

Cyclic processing constraints:

– parts of one type are processed in the order of their numbering,

– in each single-processor stage i (with $m_i = 1$), every $\underline{n}_g$-th part of each type g is processed periodically,

– in each multi-processor stage i (with $m_i > 1$), every $\underline{n}_g$-th part of each type g is processed periodically,

– in a single-processor stage i (with $m_i = 1$), no part of a new run of MPS can be started until all parts from the previous run are departed,

$$y_{kl} = 1; \ g \in G, k \in K_g, l \in K_g : k < l, \tag{12}$$

$$\begin{gathered} c_{i,next(k,K_g,\underline{n}_g)} \geq d_{ik} + \sum_{f \in G} \underline{n}_f r_{if}; \\ i \in I, g \in G, k \in K_g : \\ k \leq last(K_g) - \underline{n}_g, m_i = 1, \end{gathered} \tag{13}$$

$$c_{i,next(k,K_g,\underline{n}_g)} \geq c_{ik} - r_{ig} + \sum_{f \in G} \underline{n}_f r_{if}/m_i;$$
$$i \in I, g \in G, k \in K_g :$$
$$k \leq last(K_g) - \underline{n}_g, m_i > 1, \quad (14)$$

$$c_{ik} \geq d_{il} + p_{ik};$$
$$i \in I, f \in G, g \in G, k \in K_f, l \in K_g, 1 \leq s \leq S-1 :$$
$$s\underline{n}_f < ord(k, K_f) \leq (s+1)\underline{n}_f,$$
$$(s-1)\underline{n}_g < ord(l, K_g) \leq s\underline{n}_g, m_i = 1, \quad (15)$$

where $ord(k, K_f)$ denotes ordinal position of k in K_f.

Scheduling mode constraints:

– if cyclic mode is selected, then (16) ensures the same sequence of processing parts in each run of MPS,

$$y_{kl} = y_{next(k,K_f,s\underline{n}_f),next(l,K_g,s\underline{n}_g)};$$
$$f \in G, g \in G, k \in K_f, l \in K_g, 1 \leq s \leq S-1 : \quad (16)$$
$$f < g, 1 \leq ord(k, K_f) \leq \underline{n}_f, 1 \leq ord(l, K_g) \leq \underline{n}_g,$$

– if cyclic-batch mode is selected, then (17) ensures the same sequence of processing minimal batches of different part types in each run of MPS,

$$y_{kl} = y_{last(K_f),last(K_g)};$$
$$f \in G, g \in G, k \in K_f, l \in K_g, 1 \leq s \leq S :$$
$$f < g, (s-1)\underline{n}_f < ord(k, K_f) \leq s\underline{n}_f, \quad (17)$$
$$(s-1)\underline{n}_g < ord(l, K_g) \leq s\underline{n}_g.$$

Variable nonnegativity and integrality conditions:

$$c_{ik} \geq 0;\ i \in I, k \in K, \quad (18)$$

$$d_{ik} \geq 0;\ i \in I, k \in K, \quad (19)$$

$$x_{ijk} \in \{0,1\};\ i \in I, j \in J_i, k \in K \quad (20)$$

$$y_{kl} \in \{0,1\};\ k, l \in K : k < l. \quad (21)$$

Note that for a given sequence of parts at most one non-interference constraint (9) or (10) is active, and only if in stage i both parts k and l are assigned to the same processor j, that is, only if $x_{ijk} = x_{ijl} = 1$. Otherwise, constraints (9) and (10) are inactive and the two parts can be processed in parallel.

The assignment constraint (3) functions as a shuttle routing parts to successive processors in the circular set of parallel processors. In every multi-processor stage, the corresponding parts of one type in the consecutive MPSs (every $\underline{n}_g$-th part of each type g) are assigned to every $\underline{n}$-th parallel processor, i.e., successive parts in the input sequence are assigned to successive parallel machines. As a result the same order of processing the parts is maintained in the successive multi-processor stages, which is equivalent to the classical (single-processor) permutation flowshop scheduling, [5, 6]. Thus, in addition to the sequencing variables y_{kl} that maintain the same processing order in the single-processor stages, the assignment constraint (3) ensures the same processing order in the multi-processor stages. As a result, a permutation type schedule for a flexible flow line is obtained.

The permutation type schedule eliminates overpassing among the parts and by this implicitly reduces part flow time. For example, in surface mount technology lines in the electronics manufacturing, the total flow time of each part (printed wiring board) is limited by solder paste 'pop' life time, e.g., [3, 8].

While the assignment constraint (3) cuts off many unpromising permutation type schedules, model **CFFL** without constraint (3) may produce a non-permutation type schedule, with different orders of processing parts in the successive multi-processor stages. In general, the schedule length of a non-permutation type schedule can be shorter than that for the corresponding permutation type schedule (cf. Fig. 2 vs. Fig. 4 in Sec. 3). The **CFFL** model can be further strengthened by the addition of some valid inequalities, e.g., [9].

3. Computational examples

In this section numerical examples are presented to illustrate application of the proposed model for the cyclic scheduling of a surface mount technology line for printed wiring board assembly in the electronics manufacturing (Fig. 1).

Table 1

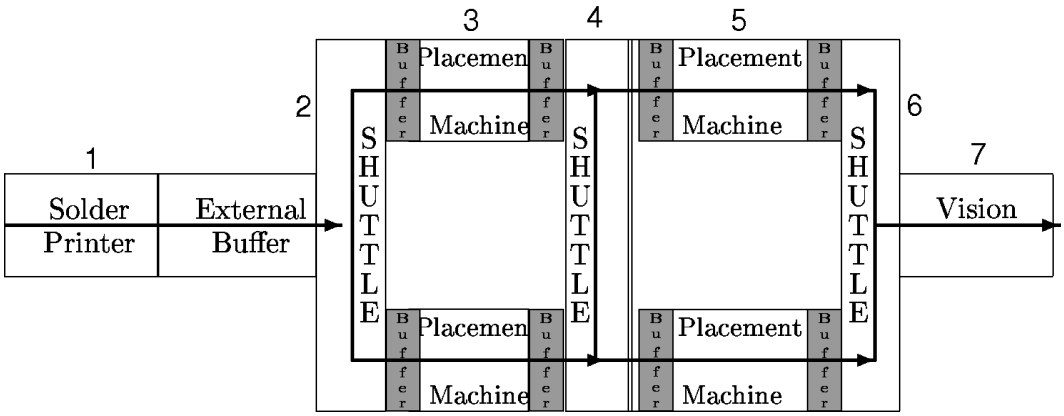

Fig. 1. A surface mount technology line with parallel machines and finite in-process buffers

Characteristics of CFFL model with constraint (3) and solution results

n, $(S \times MPS)$	Var.	Bin.	Cons.	Nonz.	$C_{\max}$	CPU$^{(a)}$
Cyclic scheduling						
5, (MPS)	159	88	524	2037	280	<1
25, $(5 \times MPS)$	951	600	12581	52707	900	4
50, $(10 \times MPS)$	2301	1600	48031	206912	1660	90
75, $(15 \times MPS)$	4051	3000	105921	461697	2450	867
100, $(20 \times MPS)$	6201	4800	186311	817182	3210	1343
200, $(40 \times MPS)$	18801	16000	733071	3271922	6310	11932
Cyclic-batch scheduling						
5, (MPS)	159	88	539	2094	280	<1
25, $(5 \times MPS)$	951	600	12671	53262	900	4
50, $(10 \times MPS)$	2301	1600	48086	208572	1660	55
75, $(15 \times MPS)$	4051	3000	106001	465382	2450	178
100, $(20 \times MPS)$	6201	4800	186416	823692	3210	514
200, $(40 \times MPS)$	18801	16000	733076	3271932	6310	11106

$^{(a)}$ CPU seconds for proving optimality on a MacBookPro, 2.8GHz Intel Core i7, RAM 16GB/Gurobi 5.5.

Table 2
Characteristics of CFFL model without constraint (3) and solution results

n, $(S \times MPS)$	Var.	Bin.	Cons.	Nonz.	$C_{\max}$	CPU or (GAP)$^{(a)}$
Cyclic scheduling						
5, (MPS)	159	88	534	2084	280	<1
25, $(5 \times MPS)$	951	600	12026	51781	880	(1.14%)
50, $(10 \times MPS)$	2301	1600	46641	205178	1660	(1.20%)
75, $(15 \times MPS)$	4051	3000	103756	460108	2440	(0.82%)
100, $(20 \times MPS)$	6201	4800	183371	816538	3220	(0.93%)
200, $(40 \times MPS)$	18801	16000	726831	3257258	$^{(b)}$	–
Cyclic-batch scheduling						
5, (MPS)	159	88	539	2094	280	<1
25, $(5 \times MPS)$	951	600	12031	51758	880	(1.14%)
50, $(10 \times MPS)$	2301	1600	46646	205188	1650	(0.61%)
75, $(15 \times MPS)$	4051	3000	103761	460118	2430	(0.41%)
100, $(20 \times MPS)$	6201	4800	183376	816548	3220	(0.93%)
200, $(40 \times MPS)$	18801	16000	726836	3257268	$^{(b)}$	–

$^{(a)}$ CPU seconds for proving optimality (or GAP% after 3600 CPU seconds) on a MacBookPro, 2.8GHz Intel Core i7, RAM 16GB/Gurobi 5.5.
$^{(b)}$ no feasible solution found within 3600 CPU seconds.

The line consists of $m = 7$ stages, where stages $i = 1, 3, 5, 7$ are machine stages and $i = 2, 4, 6$ are buffer stages. Stage $i = 1$ is a single machine for screen printing, each stage $i = 3, 5$ consists of 2 parallel machines for automatic placement of components, and stage $i = 7$ is a single machine for vision inspection. The stages $i = 2, 4, 6$ are represented by four, five, three buffers, respectively, including three shuttles routing the parts to the next placement machine, cf., part assignment constraint (3). If a shuttle is used as a buffer for some part waiting for its next machine and another part is waiting on a machine for transfer by the shuttle, then the machine is blocked by the completed part until the shuttle is available.

The production order consists of $n = 25,\ 50,\ 75,\ 100$ or 200 parts of three types, with MPS = $\{1, 2, 2\}$, and the number of MPS runs required to meet the production target is $S = 5,\ 10,\ 15,\ 20$ or 40, respectively. The processing times r_{ig} of each part type g at each stage i are shown below (for the buffer stages $i = 2, 4, 6$ all processing times are equal to zero)

$$\begin{bmatrix} 30, & 20, & 10 \\ 0, & 0, & 0 \\ 50, & 60, & 70 \\ 0, & 0, & 0 \\ 70, & 60, & 50 \\ 0, & 0, & 0 \\ 10, & 20, & 30 \end{bmatrix}.$$

The computational experiments were performed using the AMPL programming language and the Gurobi 5.5 solver (with

the default settings) on a laptop MacBookPro with Intel Core i7 processor running at 2.8GHz and with 16GB RAM. The solution results for model **CFFL** with constraint (3) are summarized in Table 1 and for model **CFFL** without constraint (3), in Table 2. The size of **CFFL** model is represented by the total number of variables, *Var.*, number of binary variables, *Bin.*, number of constraints, *Cons.*, and number of nonzero coefficients, *Nonz.* The last two columns of Table 1 give the makespan $C_{\max}$ and CPU time in seconds required to find proven optimal solutions, and in Table 2 the makespan $C_{\max}$ and GAP% after 3600 seconds of CPU time, respectively.

For the permutation type model **CFFL** with constraint (3), the Gurobi solver was capable of finding proven optimal solutions for all examples, with CPU time ranging from fraction of a second to a few hours. However, the CPU time required to find proven optimal solution using the non-permutation type model **CFFL** without constraint (3) was much longer, from a few hours to several hours. Comparison of the solution results presented in Tables 1 and 2 demonstrate that the assignment constraint (3) is a very efficient cut off constraint, that eliminates a large portion of unpromising permutation type schedules for the flexible flow line.

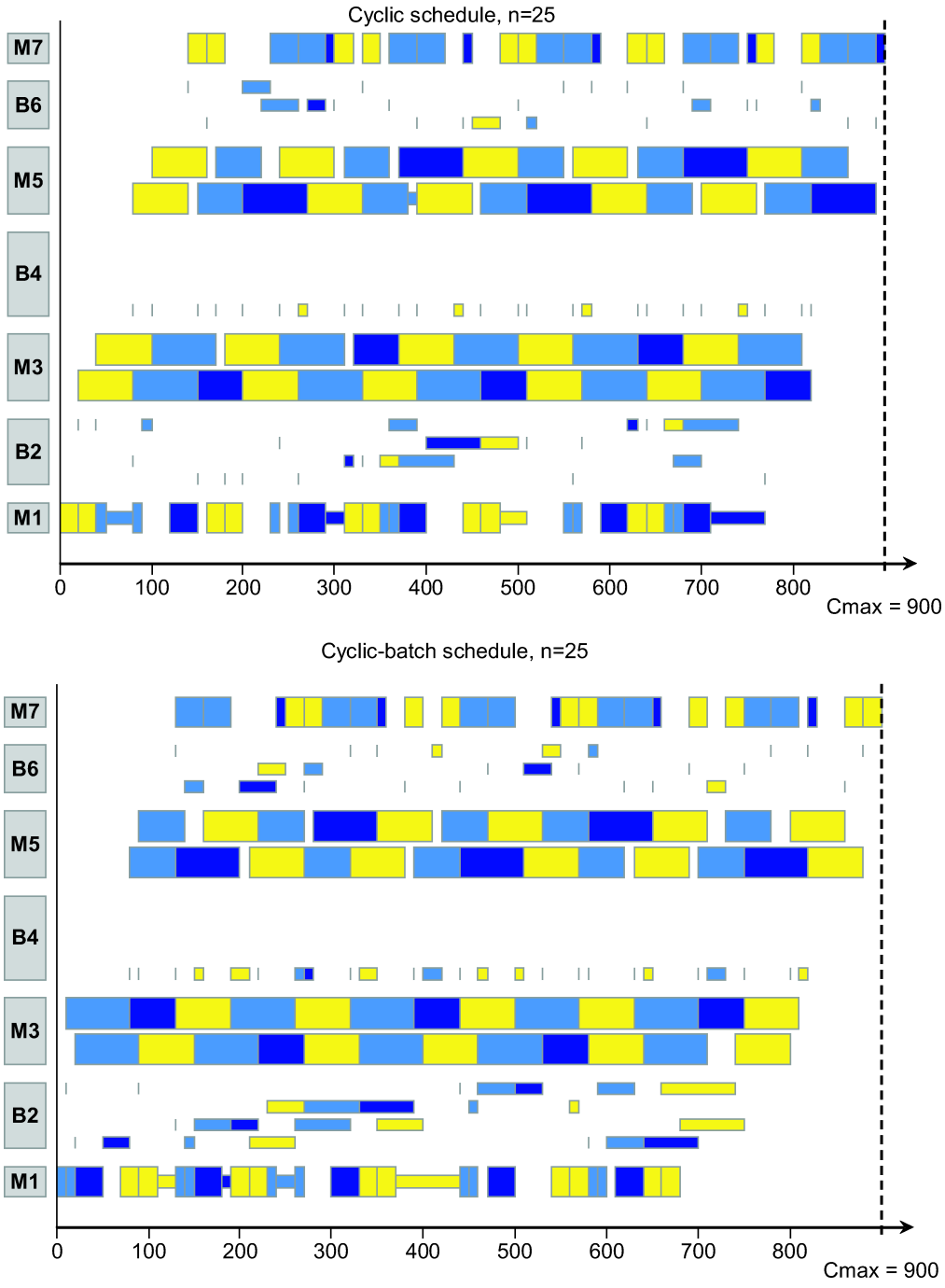

Fig. 2. Examples of cyclic and cyclic-batch, permutation type schedule for $n = 25$ parts

Table 1 shows that for both the cyclic and the cyclic-batch scheduling modes, the optimal makespan C_{max} is the same, however the CPU time required to find proven optimal solution was shorter for cyclic-batch scheduling. The difference in CPU times can be explained by a simpler structure of the cyclic-batch mode constraints (17), than that of the cyclic mode constraints (16).

Note that, because consecutive cycles overlap, for each production order the schedule length is shorter than the makespan of a cyclic schedule obtained by simple repetition of S optimal schedules for a single MPS, e.g., for $n = 200$, $(40 \times MPS)$, $C_{max} = 6310 < 11200 = 40 \times 280$. In addition, in the optimal cyclic schedules for different volumes of production orders, different cycles of parts can be observed within an MPS.

The optimal cyclic and cyclic-batch schedules for $n = 25$ and $n = 100$ parts are shown in Figs. 2 and 3, with different shading of different part types and with narrow bars indicating processor blocking. For $n = 25$ parts, the optimal input sequence of part types for a single MPS is (2,2,3,3,1) and

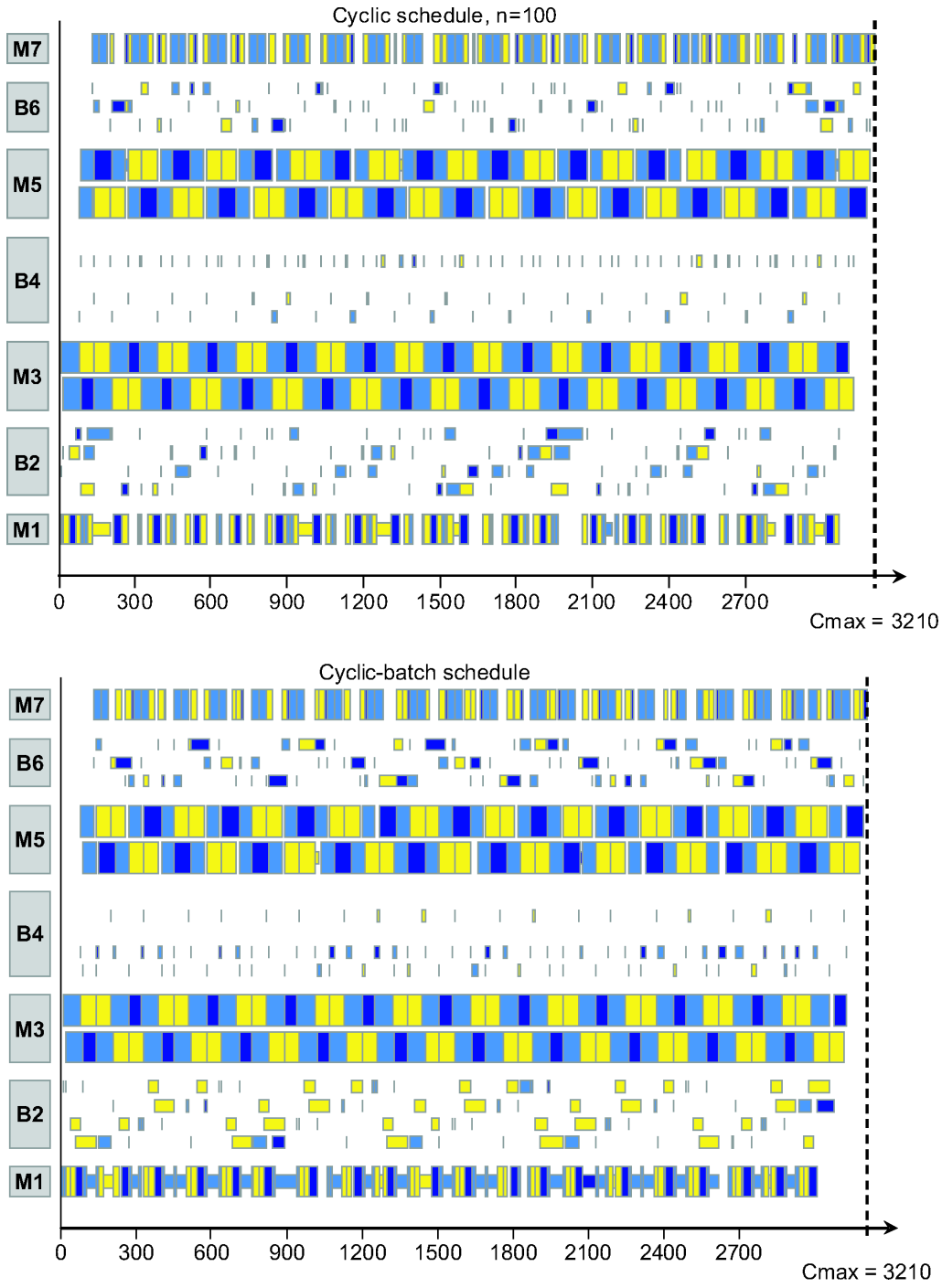

Fig. 3. Examples of cyclic and cyclic-batch, permutation type schedule for $n = 100$ parts

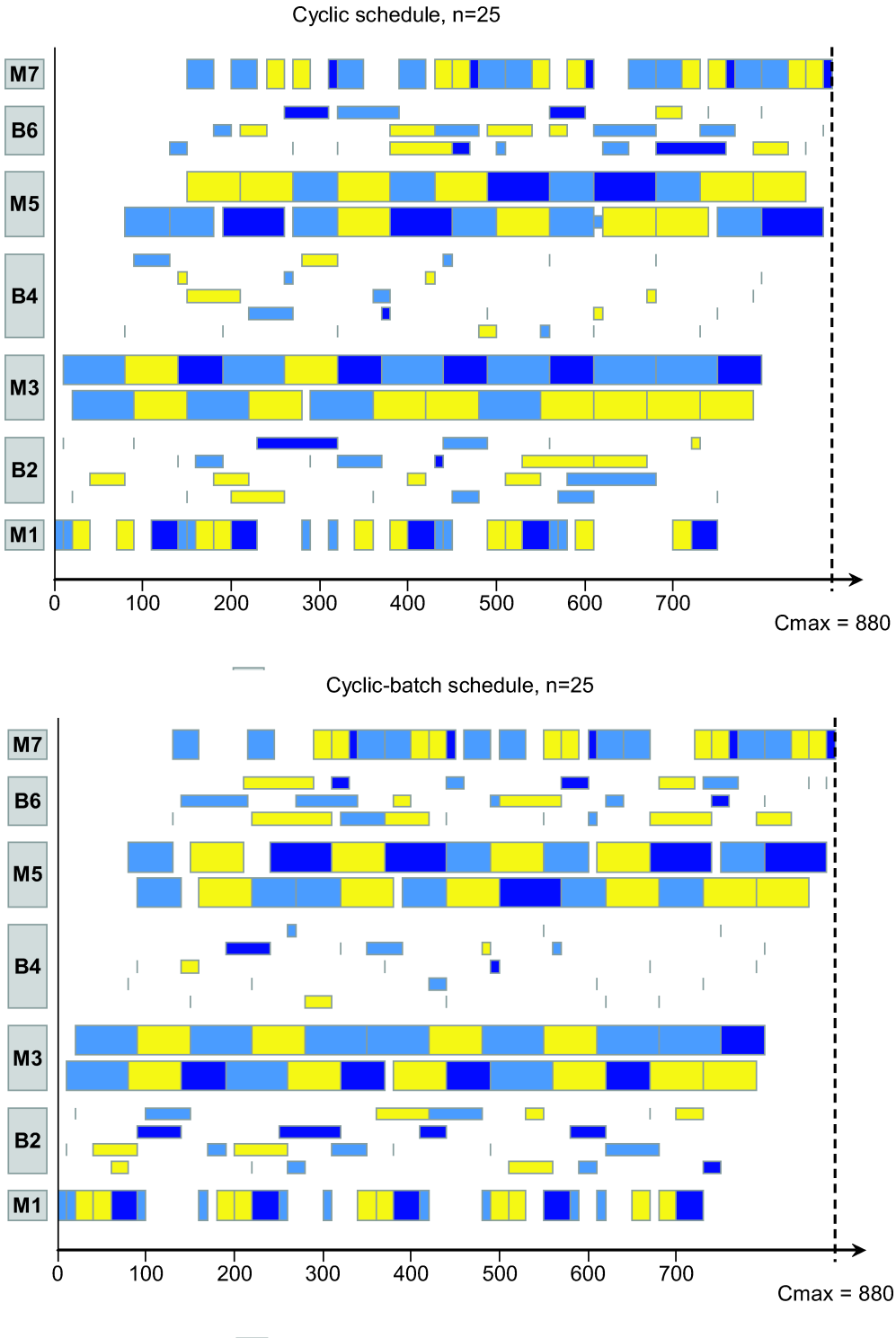

Fig. 4. Examples of cyclic and cyclic-batch, non-permutation type schedule for $n = 25$ parts

(3,3,1,2,2), respectively for cyclic and cyclic-batch scheduling mode, whereas for $n = 100$ parts, the corresponding optimal input sequences of part types for a single MPS are (3,3,2,1,2) and (3,3,2,2,1), respectively. The input sequence can be observed on the first machine, M1, in Figures 2 and 3.

For comparison, Fig. 4 shows optimal cyclic and cyclic-batch, non-permutation type schedules for $n = 25$ parts. The optimal input sequence for a single MPS is (3,3,2,2,1) for both cyclic and cyclic-batch scheduling modes. The non-permutation type schedules, however, are shorter ($C_{\max} = 880$) than the corresponding permutation schedules ($C_{\max} = 900$) in Fig. 2. Unlike in the permutation type solution presented in Fig. 2, in which the order of processing parts in the successive multi-processor stages is identical, in the non-permutation type schedules shown in Fig. 4, parts are processed in different orders.

4. Conclusions

The proposed model for cyclic scheduling is capable of optimizing throughput of a FFL. The computational experiments indicate that for the proposed monolithic approach to cyclic scheduling, the schedule length is shorter than the makespan of a cyclic schedule obtained by simple repetition of optimal schedules for a single MPS, to fulfill the overall production target. Furthermore, in the optimal cyclic schedule for the entire production target, the cycle of parts within each MPS does not necessarily coincides with that obtained for a single MPS, with no account on the total production volume.

In general, the size of the mixed integer program rapidly increases with the size of production order, that is, with the number S of cycles of an MPS, and so does the computation time required to find proven optimal schedules. The computational experiments prove a high efficiency of the assignment constraint (3) that is capable of cutting off a large portion of unpromising permutation type schedules for a flexible flow line. The future research should focus on a further strengthening the **CFFL** model to reduce the computational time required to find proven optimal solutions for large size problems, e.g., [9].

Acknowledgements. The author is grateful to two anonymous reviewers for reading the manuscript very carefully and providing constructive comments. This work has been partially supported by the NCN research grant and by AGH.

REFERENCES

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

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

[3] T. Sawik, "Mixed integer programming for scheduling surface mount technology lines", *Int. J. Production Research* 39 (14), 3219–3235 (2001).

[4] M. Magiera, "A relaxation heuristic for scheduling flowshops with intermediate buffers", *Bull. Pol. Ac.: Tech.* 61 (4), 929–942 (2013).

[5] T. Kis and E. Pesch, "A review of exact solution methods for the non-preemptive multiprocessor flowshop problem", *Eur. J. Operational Research* 164 (3), 592–608 (2005).

[6] I. Ribas, R. Leisten, and J.M. Framinan, "Review and classification of hybrid flowshop scheduling problems from a production system and a solutions procedure perspective", *Computers and Operations Research* 37, 1439–1454 (2010).

[7] E. Levner, V. Kats, D. Alcaide, and T.C.E. Cheng, "Complexity of cyclic scheduling problems: a state-of-the-art survey", *Computers and Industrial Engineering* 59, 352–361 (2010).

[8] T. Sawik, "Balancing and scheduling of surface mount technology lines", *Int. J. Production Research* 40 (9), 1993–1991 (2002).

[9] T. Sawik, "Batch vs. cyclic scheduling in flexible flow shops by mixed integer programming", *Int. J. Production Research* 50 (18), 5017–5034 (2012).

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

[11] P. Kouvelis and S. Karabati, "Cyclic scheduling in synchronous production lines", *IIE Transactions* 31 (8), 709–719 (1999).