ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2002
Seria: AUTOMATYKA z. 134 Nr kol. 1554

Tadeusz SAWIK
Akademia Górniczo-Hutnicza

SIMULTANEOUS LOADING AND SCHEDULING WITH NO REVISITING OF STATIONS IN A FLEXIBLE ASSEMBLY SYSTEM[1]

Summary. The paper presents mixed integer programming approach to simultaneous loading and scheduling of a flexible assembly system (FAS). The FAS is made up of a network of assembly stages interconnected by transportation links, where each stage consists of one or more identical parallel stations. Each station has its own internal input and output buffer of a finite capacity and a limited work space for part feeders. The problem objective is to determine an allocation of assembly tasks and part feeders among the stations and to find an assembly schedule for a mix of products with no revisting of stations so as to complete the products in minimum time. Numerical example and some computational results are presented to illustrate applications of the proposed approach.

ROZDZIAŁ ZASOBÓW I HARMONOGRAMOWANIE BEZ POWROTÓW WYROBÓW DO MASZYN W ELASTYCZNYM SYSTEMIE MONTAŻOWYM

Streszczenie. W pracy przedstawiono model programowania całkowitoliczbowego mieszanego do jednoczesnego obciążenia maszyn i szeregowania zadań w elastycznym systemie montażowym. System składa się z sieci stacji montażowych. Każda stacja obejmuje jedną lub kilka jednakowych maszyn pracujących równolegle, z własnymi buforami wejściowymi i wyjściowymi o skończonych pojemnościach oraz ograniczoną przestrzenią roboczą na podajniki części. Montowany wyrób przechodzi przez różne stacje, odwiedzając każdą co najwyżej raz. Należy wyznaczyć rozdział zadań montażowych i podajników części pomiędzy stacje oraz harmonogram montażu bez powrotów wyrobów do raz odwiedzanych stacji, tak aby zminimalizować czas wykonywania zadanego zbioru różnych wyrobów. Wyniki eksperymentów obliczeniowych ilustrują zastosowanie proponowanego podejścia.

[1]This work was partially supported by AGH and KBN

1. Introduction

A flexible assembly system (FAS) is a network of assembly stages interconnected by transportation links, where each stage consists of one or more identical parallel stations. Each station has a finite work space for part feeders and finite capacity input and output buffers for temporary storage of products waiting for processing or for transfer between the stations. In the system different types of assembly tasks can be performed to assemble various types of products. Each product visits a subset of assembly stations, where the required part feeders have been assigned, however revisiting of stations is not allowed.

The two major short-term planning issues in flexible assembly systems are loading and scheduling. Given a mix of products to be assembled, the objective of the loading problem is to allocate assembly tasks and part feeders among the assembly stations with limited work space and by this to select assembly routes for a mix of products, so as to balance the station workloads and to eliminate revisiting of stations by products. In contrast, the objective of the scheduling problem is to determine the detailed sequencing and timing of all assembly tasks for each individual product, so as to maximize the system productivity, which may be defined in terms of the assembly schedule length (makespan) for a mix of products. The limited in-process buffers result in scheduling problem with machine blocking (e.g. [1, 4]), where a completed product may remain on a machine and block it until output buffer of the machine becomes empty. This prevents another product from being processed on the blocked machine.

In this paper simultaneous loading and scheduling of a FAS is considered with no revisiting of stations and a mixed integer programming formulation is proposed to solve the problem.

The integer programming approach has been widely used to solve the loading problems (e.g. [3, 5]), some scheduling problems (e.g. [2]), and recently also to schedule surface mount technology lines (e.g. [7]).

Mixed integer programming models for simultaneous or sequential loading and scheduling of various FAS configurations with unlimited in-process buffers and revisiting of stations were presented in [6].

The paper is organized as follows. Mixed integer programming model for simultaneous FAS loading and scheduling is presented in the next section. A numerical example and some results of computational experiments with AMPL/CPLEX solver are presented in section 3 and conclusions are made in the last section.

2. Mixed integer program for simultaneous loading and scheduling

Let us consider a FAS made up of m processing stages $i \in I = I_A \cup I_B = \{1, \ldots, m\}$, for assembly ($I_A$) and for buffering ($I_B$). The processing stages are interconnected by transportation paths that link any pair of assembly stages. Transportation times between the stages, however, are assumed to be negligible. Each assembly stage $i \in I_A$ consists of $m_i \geq 1$ identical parallel assembly stations. In addition, each assembly stage has its own internal input and output buffer stages

Table 1

Notation

		Indices
h	=	processor in stage i, $h \in H_i = \{1, \ldots, m_i\}$
i	=	processing stage, $i \in I = I_A \cup I_B = \{1, \ldots, m\}$
j	=	processing task, $j \in J = \{1, \ldots, n\}$
k	=	product, $k \in K = \{1, \ldots, v\}$
		Input parameters
a_{ij}	=	working space required for assignment of task j to assembly station in stage i
b_i	=	total working space of each assembly station in stage $i \in I_A$ (number of tasks that may be assigned to each station in stage i, if all $a_{ij} = 1$)
p_{ijk}	=	processing time in stage i of task j of product k
H_i	=	the set of parallel processors at stage i
I_A	=	the set of assembly stages
I_B	=	the set of buffer stages
$I_B(i)$	=	$\{i-1, i+1\}$ – the set of input and output buffer stages of assembly stage $i \in I_A$
I_j	=	the set of assembly stages capable of performing task j
J_k	=	the ordered set of tasks required for product k
Q	=	a large number not less than the schedule length
		Decision variables
C_{max}	=	schedule length
c_{ik}	=	completion time in stage i of product k
d_{ik}	=	departure time from stage i of product k
x_{ij}	=	1, if task j is assigned to processing stage $i \in I_j$; otherwise $x_{ij} = 0$ (task assignment variable)
y_{ks}	=	1, if product k precedes product s; otherwise $y_{ks} = 0$ (product sequencing variable)
z_{ihk}	=	1, if product k is assigned to processor $h \in H_i$ in stage $i \in I$; otherwise $z_{ihk} = 0$ (product assignment variable)

$i-1$ and $i+1$ of a fixed capacity m_{i-1} and m_{i+1} buffers, respectively. Denote by $I_B(i) = \{i-1, i+1\}$ the set of input and output buffer stages of assembly stage $i \in I_A$.

In the system n different types of assembly tasks $j \in J = \{1, \ldots, n\}$ can be performed to simultaneously assemble v products $k \in K = \{1, \ldots, v\}$ of various types. Let J_k be the sequence of assembly tasks required to complete product k.

Each assembly station in stage $i \in I_A$ has a finite work space b_i where a limited number of component feeders and gripper magazines can be placed. As a result only a limited number of assembly tasks can be assigned to one assembly station. Let $I_j \subset I_A$ be the subset of assembly stages capable of performing task j, and let a_{ij} be the amount of working space of assembly station in stage $i \in I_j$, required for assignment of task j. Finally, denote by $p_{ijk} > 0$ the assembly time required to perform in stage $i \in I_j$ task $j \in J_k$ of product k.

The problem objective is to determine an allocation of assembly tasks and part feeders among the stations with limited working space and to find an assembly schedule for a mix of products so as to complete the products in minimum time with no backtracking, i.e., with no revisiting of stages by products.

An assignment of assembly tasks to stations determines a processing route for each product, i.e., a sequence of stations to be visited in order to complete the required sequence of tasks. The "no backtracking" requirement implies that for each product a subset of successive tasks is assigned to one assembly station, and hence each assembly station can be visited at most once by every product.

A unified modeling 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. The blocking time of a machine with zero processing time denotes product waiting time in the buffer represented by that machine. We assume that each product assigned to some assembly station must also visit the input and output buffers of that station. However, zero blocking time in a buffer stage indicates that the corresponding product does not need to wait in the buffer. Let us note that for each buffer stage, a product's completion time is equal to its departure time from the previous stage, since the buffer processing time is zero.

Waiting of product in the input or output buffer connected with an assembly stage where the assembly task j is to be performed is referred to as buffering task j. Both assembly stations and buffers are referred to as processors, and both assembly and buffering tasks are referred to as processing tasks.

For each type of product the total assembly time in each stage depends on the assignment of assembly tasks and the corresponding part feeders. The "no backtracking" requirement enables a subset of successive tasks of each product assigned to an assembly station to be performed contiguously, with no breaks between the tasks. Therefore, for each product k and each assembly stage $i \in I_A$, the total assembly time is a variable determined by the summation of the assembly times for all tasks $j \in J_k$ that have been assigned to this stage, i.e. $\sum_{j \in J_k} p_{ijk} x_{ij}$, where x_{ij} is task assignment binary variable (see, Table 1).

For every product k let c_{ik} denote its completion time in each stage i, and d_{ik} its departure time from stage i. Processing without preemption indicates that product k completed in stage i at time c_{ik} starts its processing in that stage at time $c_{ik} - \sum_{j \in J_k} p_{ijk} x_{ij}$. Product 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 of its processing route. If at time c_{ik} all processors in the next stage are occupied, then the processor in stage i is blocked by product k until a downstream processor becomes available.

The mathematical formulation of the mixed integer program for simultaneous loading and scheduling with no backtracking of a FAS is presented below.

Minimize

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

subject to

Task assignment with no backtracking constraints

$$\sum_{i \in I_j} x_{ij} = 1;\ j \in J \tag{2}$$

$$\sum_{j \in J} a_{ij} x_{ij} \leq b_i;\ i \in I_A \tag{3}$$

$$x_{ij} = x_{lj};\ j \in J, i \in I_j, l \in I_B(i) \tag{4}$$

$$x_{iq} \geq x_{ij} + x_{ir} - 1;\ k \in K,\ j, q, r \in J_k, i \in I_j \bigcap I_r :\ j \prec q \prec r \tag{5}$$

Product assignment constraints

$$\sum_{h \in H_i} z_{ihk} \geq x_{ij};\ k \in K, j \in J_k, i \in I_j \tag{6}$$

$$\sum_{h \in H_i} z_{ihk} \leq 1;\ i \in I, k \in K \tag{7}$$

$$z_{ihk} = z_{lfk};\ i \in I_A, l \in I_B(i), h \in H_i, f \in H_l, k \in K :\ m_i = m_l, h = f \tag{8}$$

Product completion constraints

$$c_{ik} \geq \sum_{j \in J_k} p_{ijk} x_{ij};\ i \in I_A, k \in K \tag{9}$$

$$c_{lk} + Q(2 - x_{ij} - x_{lr}) \geq c_{ik} + \sum_{g \in J_k} p_{lgk} x_{lg};\ k \in K, j, r \in J_k, i \in I_j, l \in I_r :$$
$$i \neq l, j \prec last(J_k), r = next(j, J_k) \tag{10}$$

Product non-interference constraints

$$c_{ik} + Q(2 + y_{ks} - z_{ihk} - z_{ihs}) \geq d_{is} + \sum_{j \in J_k} p_{ijk} x_{ij}; i \in I, h \in H_i, k, s \in K : k < s \tag{11}$$

$$c_{is} + Q(3 - y_{ks} - z_{ihk} - z_{ihs}) \geq d_{ik} + \sum_{r \in J_s} p_{irs} x_{ir}; i \in I, h \in H_i, k, s \in K : k < s \tag{12}$$

Buffering constraints

$$c_{l-1k} + Q(2 - x_{ij} - x_{lr}) \geq d_{i+1k};\ k \in K, j, r \in J_k, i \in I_j, l \in I_r :$$
$$i \neq l, i < m, l > 1, j \prec last(J_k), r = next(j, J_k) \tag{13}$$

$$c_{l-1k} - Q(2 - x_{ij} - x_{lr}) \leq d_{i+1k};\ k \in K, j, r \in J_k, i \in I_j, l \in I_r :$$
$$i \neq l, i < m, l > 1, j \prec last(J_k), r = next(j, J_k) \tag{14}$$

$$c_{ik} = d_{i-1k} + \sum_{j \in J_k} p_{ijk} x_{ij};\ i \in I_A, k \in K : i > 1 \tag{15}$$

$$c_{i+1k} = d_{ik};\ i \in I_A, k \in K : i < m \tag{16}$$

Completion and departure time constraints

$$d_{ik} \leq Q \sum_{j \in J_k} x_{ij};\ i \in I, k \in K \tag{17}$$

$$c_{ik} \leq d_{ik};\ i \in I, k \in K \tag{18}$$

$$d_{ik} \leq C_{max};\ i \in I, k \in K \tag{19}$$

$$\sum_{k \in K, j \in J_k} p_{ijk} x_{ij} / m_i \leq C_{max};\ i \in I_A \tag{20}$$

$$\sum_{i \in I_A, j \in J_k} p_{ijk} x_{ij} \leq C_{max};\ k \in K \tag{21}$$

Variable elimination constraints

$$x_{ij} = 0;\ i \in I_A, j \notin I_j \tag{22}$$

$$y_{ks} = 0;\ k, s \in K : k \geq s \tag{23}$$

Variable nonnegativity and integrality constraints

$$c_{ik} \geq 0;\ i \in I, k \in K \tag{24}$$

$$d_{ik} \geq 0;\ i \in I, k \in K \tag{25}$$

$$x_{ij} \in \{0,1\};\ i \in I, j \in J \tag{26}$$

$$y_{ks} \in \{0,1\};\ k, s \in K \tag{27}$$

$$z_{ihk} \in \{0,1\};\ i \in I, h \in H_i, k \in K \tag{28}$$

The objective function (1) represents the schedule length to be minimized. Constraint (2) ensures that each task type is assigned to exactly one stage, and (3) that total space required for the tasks assigned to each assembly stage does not exceed the stage finite work space available. Equation (4) ensures that the buffering tasks are assigned to the input and output buffer of the assembly stage where the corresponding assembly task is assigned. Constraint (5) ensures that consecutive tasks of each product are assigned to the same assembly stage, so that backtracking (revisiting of stages) is not required, ($\prec$ denotes precedence relations among assembly tasks).

Constraints (6) and (7) ensure that in every assembly stage each product is assigned to exactly one processor, if at least one of its required tasks is assigned to this stage, and equation (8) ensures that the product is assigned to the input and output buffers of the assembly station selected by (6) and (7).

Constraint (9) ensures that each product is processed in all stages, where the tasks required for its completion are assigned, and (10) maintains for each product the precedence relations among its tasks.

Constraints (11) and (12) are product non-interference constraints. No two products can be performed on the same processor simultaneously. For a given sequence of products either constraint (11) or (12) is active, and only if both products k and s are assigned to the same processor.

A pair of constraints (13) and (14) indicate that each product arrives in an input buffer $l-1$ of an assembly stage $l \in I_A$ immediately after its departure from the output buffer $i+1$ of the preceding assembly stage $i \in I_A$ of its processing route. Equation (15) ensures that in every stage $i \in I_A$ assembly of each product starts immediately after its departure from the input buffer $i-1$, and (16) that each product arrives in the output buffer $i+1$ immediately after its departure from the assembly stage $i \in I_A$.

Constraint (17) ensures that the product does not visit stages where its required tasks are not assigned, and (18) indicates that in every stage product departure time is not later than its completion time. Finally (19) defines the maximum completion time, and (20), (21) impose lower bounds, that account on maximum workload and maximum total processing time, respectively.

3. Numerical examples

In this section a numerical example and some computational results are presented to illustrate application of the proposed mathematical programming formulation.

The FAS configuration for the example is shown in Fig. 1. The system is made up $m = 9$ processing stages. The set of assembly stages is $I_A = \{2, 5, 8\}$ and the set of buffering stages is $I_B = \{1, 3, 4, 6, 7, 9\}$. Each assembly station has its internal input and output buffer of a unit capacity. The system consists of $m_i = 2$ parallel processors in stages $i = 1, 2, 3$, $m_i = 3$ parallel processors in stages $i = 4, 5, 6$ and $m_i = 2$ parallel processors in stages $i = 7, 8, 9$.

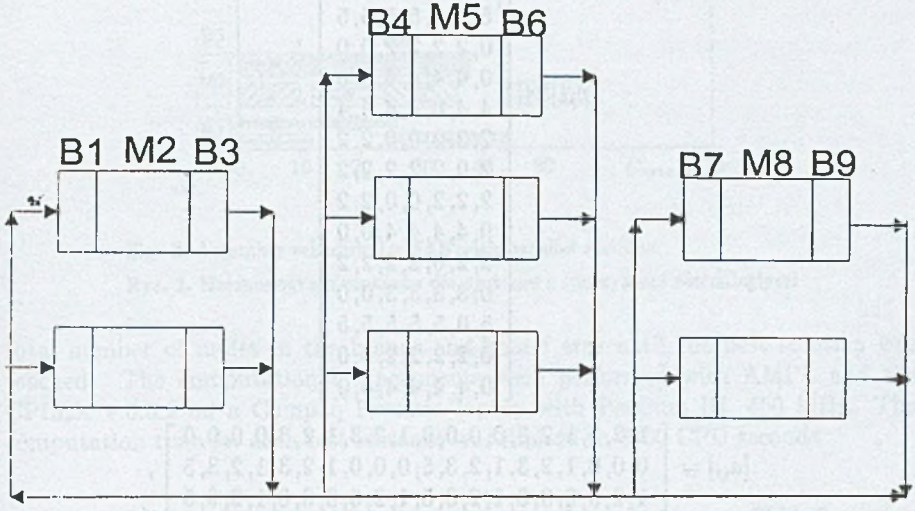

Fig. 1. FAS with parallel stations

Rys. 1. System z maszynami równoległymi

The production batch consists of $v = 7$ products to be assembled of $n = 20$ types of components. The ordered sets $J_k, k \in K$ of tasks required for each product k are shown below.

$$J_1 = (1, 2, 3, 4, 6, 8, 11, 12, 13, 14, 16, 18)$$

$$J_2 = (1, 2, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 19, 20)$$

$$J_3 = (2, 3, 4, 5, 7, 8, 9, 10, 12, 13, 14, 15, 17, 18, 19, 20)$$

$$J_4 = (1, 3, 5, 6, 7, 8, 9, 10, 11, 13, 15, 16, 17, 18, 19, 20)$$

$$J_5 = (1, 3, 5, 6, 7, 8, 9, 10, 11, 13, 15, 16, 17, 18, 19, 20)$$

$$J_6 = (1, 2, 3, 4, 6, 8, 11, 12, 13, 14, 16, 18)$$

$$J_7 = (1, 2, 3, 4, 6, 8, 11, 12, 13, 14, 16, 18).$$

The assembly times ($p_{ijk} = p_{jk}, \forall i \in I_j,\ j = 1,\dots 20,\ k = 1,\dots 7$), work space required for feeder assignments ($a_{ij},\ i = 2,5,8,\ j = 1,\dots 20$), and the total work space ($b_i,\ i = 2,5,8$) available at each station are given below

$$[p_{jk}] = \begin{bmatrix} 4,4,0,4,4,4,4 \\ 2,2,2,0,0,2,2 \\ 2,0,2,2,2,2,2 \\ 2,2,2,0,0,2,2 \\ 0,4,4,4,4,0,0 \\ 2,2,0,2,2,2,2 \\ 0,3,3,3,3,0,0 \\ 5,0,5,5,5,5,5 \\ 0,2,2,2,2,0,0 \\ 0,4,4,4,4,0,0 \\ 4,4,0,4,4,4,4 \\ 2,2,2,0,0,2,2 \\ 2,0,2,2,2,2,2 \\ 2,2,2,0,0,2,2 \\ 0,4,4,4,4,0,0 \\ 2,2,0,2,2,2,2 \\ 0,3,3,3,3,0,0 \\ 5,0,5,5,5,5,5 \\ 0,2,2,2,2,0,0 \\ 0,4,4,4,4,0,0 \end{bmatrix},$$

$$[a_{ij}] = \begin{bmatrix} 1,2,3,1,2,3,0,0,0,0,1,2,3,1,2,3,0,0,0,0 \\ 0,0,0,1,2,3,1,2,3,5,0,0,0,1,2,3,1,2,3,5 \\ 1,2,3,0,0,0,1,2,3,5,1,2,3,0,0,0,1,2,3,5 \end{bmatrix},$$

$$b_2 = 16,\ b_5 = 20,\ b_8 = 18.$$

For the example problem the mixed integer programming approach has constructed an assembly schedule with minimum makespan $C_{max} = 78$. The schedule is shown in Gantt chart presented in Fig. 2. In the figure M indicates an assembly stage and B stands for a buffering stage. Products are numbered and indicated with different patterns.

In order to evaluate performance of the mixed integer programming approach and the CPLEX solver, additional test instances of the example problem were solved. The problem characteristics and computational results are shown in Table 2. For the test instances the number of assembly stages $|I_A|$ was equal to 3,5,6 or 10, the total number of processing stages m was 9,15,18 or 30, the number of assembly task types n was 10 or 20, and the total number of assembly tasks $\sum_{k=1}^{n} |J_k|$ was 50 or 100. Each stage $i \in I$ consists of $m_i = 2$ parallel processors.

The size of the mixed integer programming model for the test instances is represented by the total number of variables, *Var.*, number of binary variables, *Bin.*, number of constraints, *Constr.*, and number of nonzero coefficients, *Nonz.*, in the constraint matrix. The last two columns of the table give best solution value and

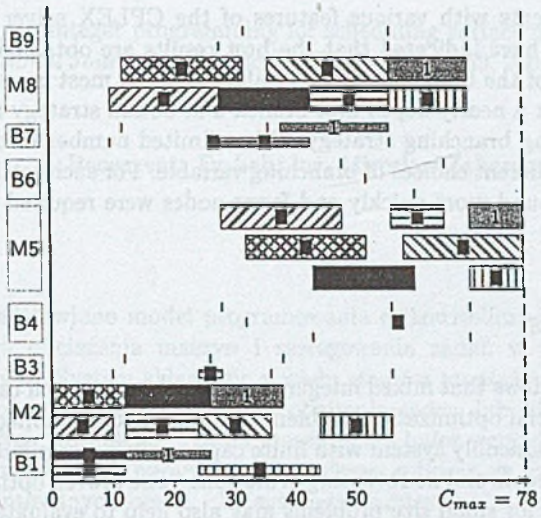

Fig. 2. Assembly schedule for FAS with parallel stations

Rys. 2. Harmonogram montażu dla systemu z maszynami równoległymi

total number of nodes in the branch-and-bound tree until the best solution was reached. The computational experiments were performed with AMPL and the CPLEX v.6.5.2 on a Compaq Presario laptop with Pentium III, 450 MHz. The computation time for each test instance was limited to 3600 CPU seconds.

Table 2

Problem Characteristics and Solution Results

$m \times m_i, n, \sum_{k=1}^{v} \lvert J_k \rvert$	Var.	Bin.	Constr.	Nonz.	C^*_{max}	Nodes
9x2,10,50	354	227	1863	9059	41	511
9x2,20,100	434	307	5877	23652	83	858
15x2,20,100	708	497	10774	47335	71	1390
18x2,20,100	846	593	13980	62980	67	1316
30x2,20,100	1394	973	27937	138435	64	11163

* Best makespan found within time limit of 3600 CPU seconds

For the test instances CPLEX solver was not able to prove optimality within the allowed 3600 seconds of CPU time, however the best solutions were found much earlier than the time limit.

It should be noted that the number of "no backtracking" constraints (5) is $O(|I_A|n^3v)$, where $|I_A|$, n, and v denote respectively, the number of assembly stages, number of task types and number of products. In some of the test instances the number of constraints (5) was as large as half of the total number of all the constraints, which indicates that no backtracking requirement significantly contributes to the computation time of the FAS loading and scheduling problem.

The experiments with various features of the CPLEX solver to speed up the solution process have indicated that the best results are obtained for various non-default settings of the branch-and-bound algorithm. In most cases, the best results were obtained for a nearly depth-first branch-and-bound strategy for node selection and for the strong branching strategy with a limited number of different branches considered for different choices of branching variable. For such settings good feasible solutions were found more quickly and fewer nodes were required to reach the best solutions.

4. Conclusions

This paper shows that mixed integer programming approach can be used to solve hard combinatorial optimization problem of simultaneous loading and scheduling a general flexible assembly system with finite capacity in-process buffers, limited work space for part feeders and no revisiting of stations. The proven optimal solutions that can be obtained for small size problems may also help to evaluate the performance of various heuristic algorithms constructed for the loading and scheduling problems. However, the computational effort required to find proven optimal schedules for realistic problems can be very high. In such cases a hierarchical, two-level approach (e.g. [6]) may help to find best assembly schedules at a much lower computational cost. In the two-level approach the solution of the loading problem at the top-level creates a job shop problem with finite in-process buffers to be solved at the base-level, where both the problems are simpler than the original mixed integer program for the simultaneous loading and scheduling.

REFERENCES

1. Hall N.G. and Sriskandarajah C.: A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, vol.44, 1996, pp.510–525.
2. Jiang J and Hsiao W.: Mathematical programming for the scheduling problem with alternate process plans in FMS, *Computers and Industrial Engineering*, vol.27, no.10, 1994, pp.15–18.
3. Kirkavak N, and Dincer C.: Analytical loading models in flexible manufacturing systems. *European Journal of Operational Research*, vol.71, 1993, pp.17–31.
4. McCormick, S.T., Pinedo M.L., Shenker S. and Wolf B.: Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research*, vol.37, 1989, pp.925–936.
5. Sawik T.: *Production Planning and Scheduling in Flexible Assembly Systems*. Springer-Verlag, Berlin 1999.
6. Sawik T.: Simultaneous versus sequential loading and scheduling of flexible assembly systems. *International Journal of Production Research*, vol. 38, 2000, pp.3267–3282.

7. Sawik T.: Mixed integer programming for scheduling surface mount technology lines. *International Journal of Production Research*, vol. 39, 2001, pp. 3219–3235.

Recenzent: Dr hab. inż. Mirosław Zaborowski, Prof. Poł. Śl.

Streszczenie

W pracy przedstawiono model programowania całkowitoliczbowego mieszanego do jednoczesnego obciążenia maszyn i szeregowania zadań w elastycznym systemie montażowym. System składa się z wielu stadiów montażowych połączonych siecią transportową, zaś każde stadium obejmuje jedną lub kilka jednakowych maszyn pracujących równolegle. Każda maszyna ma bufor wejściowy i wyjściowy o skończonej pojemności oraz ograniczoną przestrzeń roboczą, w której umieszczane są podajniki montowanych części. W systemie montowane są jednocześnie różne typy wyrobów. Każdy wyrób przechodzi w różnej kolejności przez wiele stadiów montażowych, odwiedzając co najwyżej raz każde stadium. Należy wyznaczyć rozdział zadań montażowych i podajników części pomiędzy stadia oraz harmonogram montażu bez powrotów wyrobów do raz odwiedzanych stadiów, tak aby zminimalizować czas wykonywania zadanego zbioru wyrobów. Wyniki eksperymentów obliczeniowych ilustrują zastosowanie proponowanego podejścia.