ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: AUTOMATYKA z.118 Nr kol. 1338

Czesław SMUTNICKI
Politechnika Wrocławska

SCHEDULING OF FLEXIBLE FLOW LINES AND WORK CENTRES

Summary. The paper deals with a flow line production/service system. It consists of the sequence of processing centres, each center has a number of identical parallel machines, and there are intermediate buffers between centers. An improvement approximation algorithm for the problem of finding the schedule with minimum makespan is presented.

SZEREGOWANIE ELASTYCZNYCH LINII PRZEPŁYWOWYCH I GNIAZD PRODUKCYJNYCH

Streszczenie. W pracy rozważany jest system produkcyjny/obsługi z linią przepływową. Linia ta składa się z ciągu centrów obróbczych, przy czym każde centrum posiada pewną liczbę identycznych maszyn, a także bufory o ograniczonej pojemności pośredniczące w przekazywaniu zleceń pomiędzy centrami. Dla tego problemu, z kryterium minimalizacji maksymalnego terminu zakończenia zadań, proponowany jest pewien algorytm popraw.

1. Introduction

The paper deals chiefly with a problem (FP) derived from a hybrid combination of two classic scheduling problems, namely the flow shop and parallel shop, and described briefly as follows. There is a set of parts and a set of processing centers each of which has a set of parallel identical machines. A part is associated with a sequence of operations processed at successive centers, and all parts flow through centers in the same order. At a center, a part can be processed on any machine. Moreover there are buffers with limited capacity that mediate in transferring parts between machines/centers. We want to find a schedule that minimizes the makespan, one of the most frequently used criterion.

Architecture of automated manufacturing systems usually does not allow to form internal queues of parts (transported on pallets or in containers), or limits the length of these queues due to *buffer* size. With respect to FP, the following problems (with an increasing description complexity) can be considered: (U) system has buffers with infinite capacity or finite however large enough, (W) system works without buffers, (B) system works with buffers of finite capacity. Buffers can be located in system B either before machines (M) or before centers (C). If the capacity of a buffer is greater than one, we should also consider the buffer service rule, e.g. arbitrary order (A), FIFO (F), LIFO (L). Hence, each problem FP can be characterised by additional triple $\alpha|\beta|\gamma$ where α refers to the system type, β to the buffer type, γ to the buffer service rule.

FP creates the basic model for a broad class of problems called in the literature the *flexible flow line scheduling*. These are pure problem FP or problems obtained directly from FP by introducing a few specific additional assumptions, e.g. some parts can skip one or more machines during the route, buffers have infinite capacity, there is a transport time between centers, etc. A reach review of industrial applications, among others in chemical branches, polymer and petroleum industry, computer systems, telecommunication networks, FMS, spaceship processing, etc., one can find in [4].

Although FP have quite simple formulation, it is troublesome from the algorithmic point of view. Its NP-hardness essentially restricts the set of approaches which can be applied to solve the problem. From the literature one can find exact algorithms based on the branch-and-bound ($B\&B$) [8] and mixed-integer programming (MIP), as well as a variety of approximation methods, see the review in [4]. Some of these procedures have been designed for special cases, e.g. for the systems that have only two centers, the ones where only one of the centers has parallel machines, etc.; see the newest paper in this area [1]. Algorithms for FP have been also considered in [11, 15]. Research outcomes show that $B\&B$ algorithms become useless for more than 10 parts. Similarly, the size of MIP models is impractically large even for a small number of parts and centers. Therefore, more attention has been paid recently to the approximation methods. Currently, only a few *constructive* algorithms applicable to FP are known, [2, 4, 11, 15]. Surprisingly, up to now there is no *improving* algorithm, although many recent papers have recommended the local search approach as the most promising for very hard optimization problems [3, 14].

2. $U|o|o$ system

The system has m machines located in c machine centers, and let $\mathcal{M} = \{1, \dots, m\}$ be the set of machines, $\mathcal{C} = \{1, \dots, c\}$ be the set of centers. The center $l \in \mathcal{C}$ has $m_l \geq 1$ identical parallel machines, and let $\mathcal{M}_l = \{k_l+1, \dots, k_l+m_l\} \subseteq \mathcal{M}$ be the set of machines in this center, where $k_l = \sum_{i=1}^{l-1} m_i$. The set of e parts (elements, customer orders) $\mathcal{E} = \{1, 2, \dots, e\}$ have to be processed in this system. Each part $k \in \mathcal{E}$ corresponds to a set of o_k operations $\mathcal{O}_k = \{l_k+1, \dots, l_k+o_k\}$ processed in that order, where $l_k = \sum_{i=1}^{k-1} o_i$, and let $\mathcal{O} = \bigcup_{k=1}^{e} \mathcal{O}_k = \{1, \dots, o\}$ be the set of all operations that have to be processed. Operation j corresponds to the processing of part e_j[1] at center c_j during an uninterrupted processing time $p_j > 0$ and can be performed on any machine from $\mathcal{M}_{c_j}$. Each machine can execute at most one part at a time, each part can be processed on at most one machine at a time, and each part k flows through the system so that $c_{j-1} < c_j$ for $j-1, j \in \mathcal{O}_k$. A *feasible schedule* is defined by a couple of vectors (S, P), $S = (S_1, \dots, S_o)$, $P = (P_1, \dots, P_o)$, such that above constraints are satisfied, where operation j is started on machine $P_j \in \mathcal{M}_{c_j}$ at time $S_j \geq 0$.

To provide a formal mathematical model of the problem we introduce some notions. Let $\mathcal{L}_l = \{j \in \mathcal{O} : \ c_j = l\}$ be the set of operations that have to be processed at a center $l \in \mathcal{C}$. *Batch* is a subset of operations processed on a separate machine at a center, however, due to machine identity, it is not assigned to any particular machine. To determine machine workload, each set $\mathcal{L}_l$ has to be partitioned into m_l batches $\mathcal{N}_i \subset \mathcal{L}_l$, $i \in \mathcal{M}_l$, and let $n_i = |\mathcal{N}_i|$, $i \in \mathcal{M}_l$, $l \in \mathcal{C}$. The *batch processing order* is prescribed by a permutation of operations $\pi_i = (\pi_i(1), \dots, \pi_i(n_i)) \in \mathcal{P}(\mathcal{N}_i)$ from a batch $\mathcal{N}_i$, where $\pi_i(k)$ denotes the element of $\mathcal{N}_i$ which is in position k in π_i, and $\mathcal{P}(\mathcal{N}_i)$ is the set of all permutations on the set $\mathcal{N}_i$. The *overall processing order* is defined by m-tuple $\pi = (\pi_1, \dots, \pi_m)$. All such processing orders create the set $\Pi = \{\pi = (\pi_1, \dots, \pi_m) : \ (\pi_i \in \mathcal{P}(\mathcal{N}_i), \ i \in \mathcal{M}), \ ((\mathcal{N}_i, \ i \in \mathcal{M}_l)$ is a partition of the set $\mathcal{L}_l, \ \ l \in \mathcal{C})\}$.

Next, let us consider relations between π and a feasible schedule. Assume that π is given. For each $j \in \mathcal{O}$ we define *sequential* predecessor/successor $\underline{s}_j, \overline{s}_j$ and *technological* predecessor/successor $\underline{t}_j, \overline{t}_j$ as follows: $\underline{s}_{\pi_i(j)} = \pi_i(j-1)$ for $j = 2, \dots, n_i$ and $\underline{s}_{\pi_i(1)} = \circ$ (null); $\overline{s}_{\pi_i(j)} = \pi_i(j+1)$ for $j = 1, \dots, n_i - 1$ and $\overline{s}_{\pi_i(n_i)} = \circ$; $\underline{t}_j = j-1$ if $j > 1$ and

[1] $e_j = k$ for any $j \in \mathcal{O}_k$.

$e_{j-1} = e_j$, and $\underline{l}_j = o$ otherwise; $\bar{l}_j = j+1$ if $j < o$ and $e_{j+1} = e_j$, and $\bar{l}_j = o$ otherwise. For this π, starting times S_j and completion times C_j of operations have to satisfy the following clear constraints

$$C_{\underline{l}_j} \leq S_j,\ j \in \mathcal{O};\ \underline{l}_j \neq o, \tag{1}$$

$$C_{\underline{s}_j} \leq S_j,\ j \in \mathcal{O};\ \underline{s}_j \neq o, \tag{2}$$

$$C_j = S_j + p_j,\ j \in \mathcal{O}. \tag{3}$$

These constraints lead to an obvious recursive formulae on starting times

$$S_j = \max\{S_{\underline{l}_j} + p_{\underline{l}_j}, S_{\underline{s}_j} + p_{\underline{s}_j}\} \tag{4}$$

where $S_o = 0 = p_o$, which allow us to find them in a time $O(o)$. Appropriate completion times can be found by using (3). The makespan value associated with π will be denoted by $C_{\max}(\pi)$, and thus $C_{\max}(\pi) = \max_{j \in \mathcal{O}} C_j$.

To determine the feasible schedule from the given $\pi \in \Pi$, we have to assign batches $\mathcal{N}_i$, $i \in \mathcal{M}_l$, to machines $i \in \mathcal{M}_l$ at center l, $l \in \mathcal{C}$[2]. Since machines at a center are identical, the assignment can be done in any way. Therefore, a feasible schedule (S, P) one can obtain in the following manner: set $P_j = i$ for $j \in \mathcal{N}_i$, and find S_j by using (4). It is clear that any π represents $\prod_{l=1}^{c}(m_l)!$ feasible schedules with the same makespan value and various assignments of batches to machines. Finally, we can state our problem as that of finding $\pi \in \Pi$ which minimizes $C_{\max}(\pi)$.

The analysis is based on an auxiliary graph model associated with a fixed $\pi \in \Pi$. The graph $\mathcal{G}(\pi) = (\mathcal{O}, \mathcal{A}^* \cup \mathcal{A}(\pi))$ has a set of nodes $\mathcal{O}$ and a set of arcs $\mathcal{A}^* \cup \mathcal{A}(\pi)$, where $\mathcal{A}^* = \bigcup_{j \in \mathcal{O}: \underline{l}_j \neq o}\{(\underline{l}_j, j)\}$ and $\mathcal{A}(\pi) = \bigcup_{j \in \mathcal{O}: \underline{s}_j \neq o}\{(\underline{s}_j, j)\}$. Arcs from $\mathcal{A}^*$ proceed from constraints (1), (3) and represent the route of parts through the centers; an arc $(\underline{l}_j, j) \in \mathcal{A}^*$ has weight $d_{\underline{l}_j}$. Arcs from $\mathcal{A}(\pi)$ proceed from constraints (2), (3) and represent the processing order in batches; each such arc has weight zero. Each node $j \in \mathcal{O}$ represents the operation j, event "starting" S_j of this operation, and has weight p_j. Starting time S_j equal the length of the longest path to node j (without p_j) in $\mathcal{G}(\pi)$, whereas completion time C_j equal the length of the longest path to node j (including p_j) in this graph. The makespan $C_{\max}(\pi)$ equals the length of the longest path (critical path) in $\mathcal{G}(\pi)$.

[2]Batch allow us to reduce the number of considered feasible schedules.

Let us consider a critical path in $\mathcal{G}(\pi)$ interpreted as a sequence of nodes (operations) $U = (u_1, \dots, u_w)$, $u_j \in \mathcal{O}$. We define *block* B_{ab} as a maximal subsequence of successive operations $(u_a, \dots, u_b)$ from U such that $P_{u_a} = \dots = P_{u_b}$. We also denote $\mathcal{U} = \{u_1, \dots, u_w\}$, $\mathcal{B}_{ab} = \{u_a, \dots, u_b\}$, and for any $j \in \mathcal{O}$ we define x_j so that $\pi_{P_j}(x_j) = j$.

The proposed solution algorithm is based on tabu search approach, which is a modern, useful technique for constructing approximation algorithms of various hard optimization problems, [3]. The basic idea of this approach applied to our problem consists in starting from an initial solution (father) and searching through its descendants (the set of solutions called neighborhood generated in a specific way from father) for a solution with the lowest makespan. Then the search repeats from the the best one, as a new father solution, and the process is continued. The father's neighborhood is generated by moves; a move changes the positions of a number of elements in father solution. In order to avoid cycling, becoming trapped to a local optimum, and more general to guide the search to "good regions" of the solution space, a memory of the search history is introduced. Among many classes of this memory, we refer only to the tabu list which recorded, for a chosen span of time, selected attributes of subsequently visited solutions and/or performed moves, treated them as a form of prohibition for the future moves.

Based on conclusions from [5, 6], we consider only so called *insertion moves*. In our case this type of moves is associated with π and can be defined by a triple (j, i, y), such that operation j is removed from batch $\mathcal{N}_{P_j}$ and then inserted in $\mathcal{N}_i$ in position y in the permutation π_i (y becomes new position of j in π_i). Clearly, it has to be $i \in \mathcal{M}_{c_j}$, $1 \le y \le n_i + 1$ if $i \ne P_j$, $1 \le y \le n_i$ if $i = P_j$.

The neighborhood of π consists of orders π_v generated by moves v from a given set. By using natural, simple approach one can propose the *insertion neighborhood* $\{\pi_v : v \in \mathcal{V}(\pi)\}$ generated by the move set

$$\mathcal{V}(\pi) = \bigcup_{j \in \mathcal{O}} \bigcup_{i \in \mathcal{M}_{c_j}} \mathcal{V}_{ji}(\pi), \tag{5}$$

where $\mathcal{V}_{ji}(\pi) = \bigcup_{y=1}^{n_i+1} \{(j, i, y)\}$ if $i \ne P_j$ and $\mathcal{V}_{ji}(\pi) = \bigcup_{y=1:\, x_j - 1 \ne y \ne x_j}^{n_i} \{(j, i, y)\}$ if $i = P_j$. The condition $x_j - 1 \ne y \ne x_j$ is added to prevent redundancy of moves [3]. The set $\mathcal{V}_{ji}(\pi)$ contains moves such that operation j is deleted from the permutation π_{P_j} and inserted in all possible positions in the permutation π_i. One can verify that the number of moves in

[3]Moves are redundant if provide the same solutions.

$\mathcal{V}(\pi)$ equals approximately oc.

Final quality of the tabu search algorithm depends among others on the computational complexity of the single neighborhood search and on the neighborhood size. Our previous research show that such the algorithm behaves much better if we evaluate descendants directly by the criterion value, [5, 6, 7]. Therefore $\mathcal{V}(\pi)$ should be given up as expensively time-consuming, and we propose next a reduced neighborhood.

The main idea of the reduction consists in eliminating some moves from $V(\pi)$ for which it is known a priori (without computing the makespan) that they will not immediately improve $C_{\max}(\pi)$. By employing certain graph property "if there exists in $\mathcal{G}(\pi_v)$ a path containing all nodes from $\mathcal{U}$, where $\mathcal{U}$ is found for $\mathcal{G}(\pi)$, we have $C_{\max}(\pi_v) \geq C_{\max}(\pi)$" we propose the following reduced set of moves

$$\mathcal{W}(\pi) = \bigcup_{j \in \mathcal{U}} \bigcup_{i \in \mathcal{M}_{e_j}} \mathcal{W}_{ji}(\pi), \tag{6}$$

where $\mathcal{W}_{ji}(\pi) \subseteq \mathcal{V}_{ji}(\pi)$ are defined only for $j \in \mathcal{B}_{ab} \subseteq \mathcal{U}$ in the following manner. If $a = b$, we put $\mathcal{W}_{ji}(\pi) = \emptyset$. If $i \neq P_j$ and $a \neq b$ then $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi)$. In the remain cases, i.e. if $i = P_j$ and $a \neq b$, we set: $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(x_{u_a} + 1, x_{u_b} - 1)$ if $j \in \mathcal{B}_{ab} \setminus \{u_a, u_b\}$, where $\mathcal{X}_j(k,l) = \bigcup_{y=k}^{l} \{(j, P_j, y)\}$; $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(1, x_j)$ if $j = u_a$; $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(x_j, n_i)$ if $j = u_b$. A strongest reduction is possible for $i = P_j$ and $a \neq b$: if $a = 1$ and $j \neq u_b$, we can set $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(1, x_{u_b} - 1)$; by symmetry if $b = w$ and $j \neq u_a$, we can set $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(x_{u_a} + 1, n_i)$.

A rough analysis of neighborhood sizes shows clear advantage since $|\mathcal{V}(\pi)| \,/\, |\mathcal{W}(\pi)| \approx o/w$ and $w \ll v$. Assuming uniform distribution of machines over centers and uniform distribution of parts over batches, we have $o/w \approx m$. The proposed neighborhood possesses also a *connectivity property* which provides for tabu search possibility of finding optimal solution, however without assurance that it will be discovered, [7]. The skipped part of the neighborhood is "less interesting" due to Property 1.

Property 1. *For any processing order π_v, $v \in \mathcal{V}(\pi) \setminus \mathcal{W}(\pi)$, we have $C_{\max}(\pi_v) \geq C_{\max}(\pi)$.*

3. $W|o|o$ and $B|M|F$ systems

The production system has mathematical model from Section 2 with an additional constraints that each buffer Q_i to the machine i has capacity $q_i \geq 0$, e.g. can store at most q_i parts at a time. It means that each part completed on some machine and directed to a machine i can be stored in the buffer Q_i if it is not fulfilled, otherwise remains on the primal machine (blocks this machine) until it can be transferred to a buffer. Due to buffer service rule, jobs pass through the buffer Q_i in the order given by π_i.

First, we consider relations between a processing order $\pi \in \Pi$ and a feasible schedule (S, P). Let C'_j denote the time of releasing the machine P_j after completing operation j. For a given π appropriate feasible schedule have to satisfy constraints (1), (3) and additionally

$$C_j \leq C'_j, \ j \in \mathcal{O}, \tag{7}$$

$$C'_{\underline{s}_j} \leq S_j, \ j \in \mathcal{O}; \ \underline{s}_j \neq \circ. \tag{8}$$

To set buffering constraints let us consider events associated with operation $j = \pi_i(x)$ for some $1 \leq x \leq n_i$, $i = P_j$, such that $\underline{t}_j \neq \circ$. There are two disjoin cases.

(a) If the buffer Q_i contains at most $q_i - 1$ $(q_i \geq 1)$ parts associated with recently performed operations $\pi_i(x - q_i + 1), \ldots, \pi_i(x - 1)$, then part $e_{\underline{t}_j} = e_j$ can be passed to the buffer Q_i immediately after completion of operation $\underline{t}_j$, i.e. $C'_{\underline{t}_j} = C_{\underline{t}_j}$. It means that all parts associated with operations $\pi_i(1), \ldots, \pi_i(x - q_i)$ must be taken from buffer Q_i before time moment $C'_{\underline{t}_j}$, therefore, since $S_{\pi_i(y)} < S_{\pi_i(y+1)}$, we have

$$S_{\pi_i(x-q_i)} \leq C'_{\underline{t}_j}, \ x = q_i + 1, \ldots, n_i. \tag{9}$$

(b) If the buffer Q_i contains exactly q_i $(q_i \geq 0)$ parts associated with recently performed operations $\pi_i(x - q_i), \ldots, \pi_i(x - 1)$, then part e_j will be passed to the buffer Q_i after completion of operation $\underline{t}_j$ in a time moment when a successive part from Q_i will be taken for processing, i.e.

$$C'_{\underline{t}_j} = \min\{S_{\pi_i(x-q_i)}, \ldots, S_{\pi_i(x-1)}\} = S_{\pi_i(x-q_i)} \tag{10}$$

Clearly, (10) can be replaced by more general constraint (9). To eliminate variables C'_j from constraints (7)-(9), we introduce notion of *b-sequential* predecessor/successor $\underline{r}_j$, $\overline{r}_j$ of operation j defined as follows: $\underline{r}_{\pi_i(j)} = \pi_i(j - q_i)$ for $j = q_i + 1, \ldots, n_i$, and $\underline{r}_{\pi_i(j)} = \circ$ for $j = 1, \ldots, q_i$; $\overline{r}_{\pi_i(j)} = \pi_i(j + q_i)$ for $j = 1, \ldots, n_i - q_i$, and $\overline{r}_{\pi_i(j)} = \circ$ for $j = n_i + q_i + 1, \ldots, n_i$. Consequently, (9) can be written as $S_{\underline{r}_j} \leq C'_{\underline{t}_j}$. Next, from (8) we have $C'_j \leq S_{\overline{s}_j}$ if $\overline{s}_j \neq \circ$. This allow us to eliminate C'_j and to express constraint (9) as follows

$$S_{\underline{r}_j} \leq S_{\overline{s}_{\underline{t}_j}}, \tag{11}$$

which holds for $j \in \mathcal{O}$ such that $\underline{r}_j \neq o$, $\underline{t}_j \neq o$, $\overline{s}_{\underline{t}_j} \neq o$. Finally we obtain a recursive formulae on starting times

$$S_j = \max\{S_{\underline{t}_j} + p_{\underline{t}_j}, S_{\underline{s}_j} + p_{\underline{s}_j}, S_{\underline{r}_{\underline{t}_j}}\} \tag{12}$$

where $\underline{r}_o = o$, $\overline{s}_o = o$, $S_o = 0 = p_o$. Similarly as (4) all S_j can be calculated in $O(o)$ time.

An appropriate auxiliary graph model can be formulated as follows $\mathcal{G}(\pi) = (\mathcal{O}, \mathcal{A}^* \cup \mathcal{A}(\pi) \cup \mathcal{A}'(\pi))$, where $\mathcal{O}$, $\mathcal{A}^*$ and $\mathcal{A}(\pi)$ are defined in Section 2. The set of arcs $\mathcal{A}'(\pi) = \bigcup_{j \in \mathcal{O}:\, \underline{r}_j \neq o;\, \underline{t}_j \neq o;\, \overline{s}_{\underline{t}_j} \neq o} \{(\underline{r}_j, \overline{s}_{\underline{t}_j})\}$ represent buffering constraints (11); an arc $(\underline{r}_j, \overline{s}_{\underline{t}_j}) \in \mathcal{A}'(\pi)$ has weight minus $p_{\underline{r}_j}$. Notions S_j, C_j, and $C_{\max}(\pi)$ have the same interpretation as in Section 2.

By analysing $\mathcal{G}(\pi)$ one can detect possibility of cycle existence. Note, this has not be possible in system $U|o|o$. Clearly, in such case no feasible schedule (S, P) can be obtained. Hence, we will consider further on only *feasible overall processing orders* $\pi \in \Pi^F \subseteq \Pi$, i.e. orders without a cycle in appropriate graph $\mathcal{G}(\pi)$.

Thus, for a given $\pi \in \Pi^F$, the feasible schedule (S, P) can be found in the same manner as in Section 2, however using (12) instead of (4) for calculating S.

Let us define the critical path U and blocks B_{ab} in $\mathcal{G}(\pi)$ the same way as in Section 2. Unfortunately, Property 1, fundamental for the use of move set $\mathcal{W}(\pi)$, is not true in this case. Therefore, equally the block notion and reduced set of moves are useless. To restore all advantageous properties we have to introduce some new notions. The *extended critical sequence* $U^* = (u_1^*, \ldots, u_z^*)$ is a sequence obtained from U by inserting between any successive operations u_{k-1} and u_k such that $(u_{k-1}, u_k) \in \mathcal{A}'(\pi)$ an additional element $\underline{s}_{u_k}$. Note, U^* need not be any path in $\mathcal{G}(\pi)$. Next, we define an *extended block* B_{ab}^* as a maximal subsequence of successive operations $(u_a^*, \ldots, u_b^*)$ from U^* such that $P_{u_a^*} = \ldots = P_{u_b^*}$. For convenience of notations we set $\mathcal{U}^* = \{u_1^*, \ldots, u_z^*\}$, $\mathcal{B}_{ab}^* = \{u_a^*, \ldots, u_b^*\}$. For each extended block B_{ab}^*, an *anti-block* B_b° is defined as $B_b^\circ = ()$ (empty) if $b = z$ or $(u_b^*, u_{b+1}^*) \in \mathcal{A}^*$, and $B_b^\circ = (j_1, \ldots, j_{q_i+1})$ such that $j_1 = u_b^*$, $j_{k+1} = \overline{s}_{j_k}$ for $k = 1, \ldots, q_i$, $i = P_{j_1}$, otherwise. By the definition of $\mathcal{A}'(\pi)$ such sequence always exists and $j_{q_i+1} = \overline{r}_{j_1}$. Denote by $\mathcal{B}_b^\circ$ the set of elements from the sequence B_b^*, and let $\mathcal{U}^\circ = \bigcup_{\mathcal{B}_{ab}^* \subseteq \mathcal{U}^*} \mathcal{B}_b^\circ$. We propose the following set of moves for tabu search method

$$\dot{\mathcal{W}}(\pi) = \bigcup_{j \in \mathcal{U}^* \cup \mathcal{U}^\circ} \; \bigcup_{i \in \mathcal{M}_{\nu_j}} \dot{\mathcal{W}}_{ji}(\pi) \tag{13}$$

Sets $\hat{\mathcal{W}}_{ji}(\pi) \subseteq \mathcal{V}_{ji}(\pi)$ are defined depending on extended blocks and anti-blocks, only for $j \in \mathcal{B}^{*}_{ab} \cup \mathcal{B}^{\circ}_{b} \subseteq \mathcal{U}^{*} \cup \mathcal{U}^{\circ}$. If $j \in \mathcal{B}^{*}_{ab}$ such that $\mathcal{B}^{\circ}_{b} = \emptyset$ we define $\hat{\mathcal{W}}_{ji}(\pi)$ in exactly the same way as $\mathcal{W}_{ji}(\pi)$, see Section 2, however with respect to extended block $\mathcal{B}^{*}_{ab}$. If $j \in \mathcal{B}^{*}_{ab} \cup \mathcal{B}^{\circ}_{b}$, $\mathcal{B}^{\circ}_{b} \neq \emptyset$, and $i \neq P_j$, we set $\hat{\mathcal{W}}_{ji}(\pi) = \mathcal{V}_{ji}(\pi)$. In remain cases, i.e. if $i = P_j$, $a \neq b$, $\mathcal{B}^{\circ}_{b} \neq \emptyset$ we set: $\hat{\mathcal{W}}_{ji}(\pi) = \mathcal{V}_{ji}(\pi)$ if $j = u^{*}_{b}$; $\hat{\mathcal{W}}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(x_{u^{*}_{a}}+1, x_{u^{*}_{b}}-1)$ if $j \in \mathcal{B}^{*}_{ab} \setminus \{u^{*}_{a}, u^{*}_{b}\}$; $\hat{\mathcal{W}}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(x_{u^{*}_{b}}+1, x_{\bar{r}_{u^{*}_{b}}}-1)$ if $j \in \mathcal{B}^{\circ}_{b} \setminus \{u^{*}_{b}, \bar{r}_{u^{*}_{b}}\}$; $\hat{\mathcal{W}}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(1, x_j)$ if $j = u^{*}_{a}$; $\hat{\mathcal{W}}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \mathcal{X}_j(x_j, n_i)$ if $j = \bar{r}_{u^{*}_{b}}$.

Property 2. *For any processing order* π_v, $v \in \mathcal{V}(\pi) \setminus \hat{\mathcal{W}}(\pi)$, *we have either* $\pi_v \notin \Pi^F$ *or* $C_{\max}(\pi_v) \geq C_{\max}(\pi)$.

Thus, solution method from Section 2 is applicable to these systems as well.

4. Comments and conclusions

The proposed approach can be extended to more general cases which cover most of the practical applications in flexible flow lines scheduling problems and also in scheduling parts in a flexible cell production (work centers). One can note that the following constraints can be modelled without essential complication of the model: there is a job transport time between centers, jobs are arrived in various time moments, machines are available after its ready time, jobs must be finished before the given due date, jobs must be finished just in time, machines are non uniform, etc.

The remain components of the proposed algorithm (not discussed here) are presented in detail in [7] and have been selected among many alternative constructions that were tested and examined in a few recent papers, [5, 6, 7]. Excellent computational results obtained for all problems from cited papers can be obtained also in all considered cases.

Further research should be lead to examine the remain types of buffers mentioned in Section 1. Primal results of analysis of the system $B|C|o$ show useless of models based on batch processing orders. Instead of them the models based on *center processing order* should be used with quite another definition of moves.

BIBLIOGRAPHY

1. Chen B.: Analysis of Classes of Heuristics for Scheduling a Two-Stage Flow Shop with Parallel Machines at One Stage, Journal of Operational Research Society, 46, 1995, 234-244.

2. Ding F.Y., Kittichartphayak D.: Heuristics for scheduling flexible flow lines, Computers Industrial Engineering 26, 1994, 27-34.

3. Glover F., Laguna M.: Tabu Search. Modern Heuristic Techniques for Combinatorial Problems, C.Reeves, ed., Blackwell Scientific Publishing, 1993, 70-141.

4. Hunsucker J.L., Shah J.R.: Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. European J. Oper. Res. 72, 1994, 102-114.

5. Nowicki E., Smutnicki C.: A fast tabu search algorithm for the job shop. Report ICT PRE 8/93, Technical University of Wroclaw, 1993. Management Science. (in print), 1996.

6. Nowicki E., Smutnicki C.: A fast tabu search algorithm for the flow shop. Report ICT PRE 8/94, Technical University of Wroclaw, 1994. European J. Oper. Res. (in print), 1996.

7. Nowicki E., Smutnicki C.: Flow shop with parallel machines. A tabu search approach. Report ICT PRE 30/95, Technical University of Wroclaw, 1995.

8. Shaukat A. B., Hunsucker J.L.: Branch and bound algorithm for the flow shop with multiple processors. European J. Oper. Res. 51, 1991, 88-99.

9. Sriskandarajah C., Sethi S.P.: Scheduling algorithms for flexible flowshops: Worst and average case performance. European J. Oper. Res. 43, 1989, 43- 60.

10. Santos D.L., Hunsucker J.L., Deal D.E.: Global lower bounds for flow shops with multiple processors, European Journal of Operational Research, 80, (1995), 112-120.

11. Sawik T.: A scheduling algorithm for flexible flow lines with limited intermediated buffers. Applied Stochastic Models and Data Analysis 9, 1993, 127-138.

12. Sawik T.: Scheduling flexible flow lines with no in-process buffers. International Journal of Production Research, 1994, in print.

13. Smutnicki C.: Block approach in flow shop problems with limited storage space (Polish), Zeszyty Naukowe Politechniki Śląskiej, Automatyka 84, 1986, 223-233.

14. Vaessens R.J.M., Aarts E.H.L., Lenstra J.K.: Job Shop Scheduling by Local Search, Memorandum COSOR 94-05, Eidhoven University of Technology.

15. Wittrock R.J.: An adaptable scheduling algorithm for flexible flow lines. Operations Research 36, 1988, 445-453.

Recenzent: Dr hab. inż. Ewa Dyduch-Dudek

Wpłynęło do Redakcji do 30.06.1996 r.

Abstract

The paper deals with a fundamental problem considered in flexible flow line scheduling. A number of parts should be performed in a sequence of processing centers, where each center has a number of identical parallel machines, and there are intermediate buffers between centers. An improvement approximation algorithm for this problem of finding the schedule with minimum makespan is presented. This algorithm is based on tabu search approach with reduced neighborhood which employs the notions of a critical path and blocks of operations.