ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1998
Seria: AUTOMATYKA z. 125 Nr kol. 1391

Czesław SMUTNICKI
Instytut Cybernetyki Technicznej Politechniki Wrocławskiej

HARMONOGRAMOWANIE PRACY KOMÓRKI PRODUKCYJNEJ Z KRYTERIUM DOKŁADNOŚCI DOSTARCZANIA

Streszczenie. W pracy rozważono problem dostarczania dokładnie na czas różnych elementów produkowanych przez komórkę produkcyjną, wyposażoną w pewną liczbę niejednorodnych, równoległych maszyn. Problem jest modelowany i rozwiązywany przy wykorzystaniu pewnych szczególnych własności oraz standardowych pojęć z klasycznej teorii szeregowania. Podejście zaproponowane i rozwijane w pracy może być stosowane do rozwiązywania bardziej złożonych problemów tego typu. Podejście to odwołuje się do klasycznych problemów szeregowania własności grafowych, tzw. własności blokowych oraz algorytmów lokalnych poszukiwań. Szczególną uwagę zwrócono na te własności teoretyczne, które są najbardziej użyteczne dla metody Poszukiwań z Pamięcią Adaptacyjną.

SINGLE-CELL SCHEDULING WITH A DELIVERY PERFORMANCE

Summary. In this paper we discuss a problem of perfect supply of various items produced by a manufacturing cell having a number of unrelated parallel machines. The problem is modelled and solved applying some special properties and a lot of conventional notions from the classic scheduling theory. The proposed and developed approach can be applied to solve more complex problems of this type. The base of this approach refers to classic scheduling problems, graph properties (including generalisation of so called block properties) and local search algorithms. A significant attention has been paid to these theoretical properties which are the most useful for the Adaptive Memory Search (AMS) algorithm.

1. Introduction

The Just-in-time (JIT) strategy of manufacturing assumes that each items are produced exactly on time and only in the quantities needed. (The role of JIT systems in modern manufacturing systems are also discussed in [6].) Among many characteristic elements of JIT systems, the performance (regularity) of the supply of items by the system output plays a significant role.

In this paper we discuss a problem of delivery of items which can be modelled and solved using conventional notions of scheduling problems such as ready times, due dates,

delivery times, etc. Although the problem is inspired by JIT systems, it can be also applied in conventional manufacturing systems. The proposed approach can be extended to solve more complex problems of this type. The approach refers to graph properties and local search algorithms, especially to the Adaptive Memory Search (AMS). It has been verified that AMS provides *reasonably good* solution in a *reasonably short* time, [7, 4, 5].

2. The problem

Consider a production cell having m machines, and let $\mathcal{M} = \{1, \ldots, m\}$ be the set of these machines. The set of n parts $\mathcal{N} = \{1, 2, \ldots, n\}$ has to be processed by this cell. Each part $j \in \mathcal{N}$ requires a single machine for processing, needs the time $p_{ij} > 0$ while is processed on machine $i \in \mathcal{M}$ and has a delivery interval $[b_j, c_j]$ within which this part has to be completed in the ideal case. Because of co-operation with preceding stages, each part has a date a_j at which it becomes available for processing. Naturally, it is assumed that $a_j \leq b_j \leq c_j$, $j \in \mathcal{N}$ (this assumption is not restrictive and can be released without any problems). Each part can be performed on any machine from $\mathcal{M}$. Once started, a part cannot be interrupted. Each machine can execute at most one part at a time, each part can be processed on at most one machine at a time. A *feasible schedule* is defined by a couple of vectors (S, T), $S = (S_1, \ldots, S_n)$, $T = (t_1, \ldots, t_n)$, such that the above constraints are satisfied, where part j is started on machine $t_j \in \mathcal{M}$ at time S_j. The formulated model is a nontrivial generalisation of the single-cell work-loading and scheduling model from [6], obtained by incorporation part ready times a_j.

The deviation from on time part supply can be measured in several ways, see for example the survey [1]. Since generally problems with on-time supply belong to the quite troublesome class of scheduling problems with so called "non-regular" goal function, the chosen measure has a significant influence on the applied solution method. In this paper we propose a new measure which allow us to solve the problem by applying some conventional models from the classic scheduling theory. For this purpose we use the maximum deviation the from the delivery interval, namely

$$\max_{j \in \mathcal{N}} [b_j - C_j, C_j - c_j]^+ \tag{1}$$

where $[x_1, x_2, \ldots]^+ = \max\{0, x_1, x_2, \ldots\}$. We wish to find the schedule that minimises (1).

To determine *machines workload*, set $\mathcal{N}$ has to be partitioned into m subsets $\mathcal{N}_i \subset \mathcal{N}$,

each of which is associated with suitable machine $i \in \mathcal{M}$, and let $n_i = |\mathcal{N}_i|$, $i \in \mathcal{M}$. The *processing order* of parts from $\mathcal{N}_i$ is prescribed by a permutation $\pi_i = (\pi_i(1), \ldots, \pi_i(n_i)) \in \mathcal{P}(\mathcal{N}_i)$, where $\pi_i(j)$ denotes the element of $\mathcal{N}_i$ which is in position j 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 $\pi = (\pi_1, \ldots, \pi_m)$. All such processing orders create the set

$$\Pi = \{\pi : (\pi_i \in \mathcal{P}(\mathcal{N}_i),\ i \in \mathcal{M}),\ ((\mathcal{N}_i,\ i \in \mathcal{M}) \text{ is a partition of } \mathcal{N})\}. \tag{2}$$

Each π generates a feasible schedule (S, T) in the following manner: set $t_{\pi_i(j)} = i$, $j = 1, \ldots, n_i$, $i \in \mathcal{M}$, and find starting times $S = (S_1, \ldots, S_n)$ by solving the problem

$$L(\pi) = \min_{S \in \mathcal{S}(\pi)} \max_{j \in \mathcal{N}} [b_j - C_j, C_j - c_j]^+ \tag{3}$$

with the set $\mathcal{S}(\pi)$ introduced by constraints

$$C_{\pi_i(s)} \le S_{\pi_i(s+1)},\ \ s = 1, \ldots, n_i - 1,\ \ i \in \mathcal{M}. \tag{4}$$

$$a_j \le S_j,\ \ j \in \mathcal{N}, \tag{5}$$

$$C_{\pi_i(s)} = S_{\pi_i(s)} + p_{i,\pi_i(s)},\ \ s = 1, \ldots, n_i,\ \ i \in \mathcal{M}, \tag{6}$$

Now, we can rephrase originally stated problem as that of finding $\pi \in \Pi$ that minimises (3) under constraints (4)–(6). The problem (3) – (6) can be decomposed into m independent single-machine sub-problems

$$L(\pi) = \max_{i \in \mathcal{M}} L(\pi_i) \tag{7}$$

where

$$L(\pi_i) = \min_{S \in \mathcal{S}(\pi_i)} \max_{j \in \mathcal{N}_i} [b_j - C_j, C_j - c_j]^+ \tag{8}$$

and $\mathcal{S}(\pi_i)$ is given by (4)–(6) for fixed $i \in \mathcal{M}$.

Observe, the decomposition of the problem stated can be performed on several levels. On the upper level, we would like to find the optimal processing order $\pi \in \Pi$ which minimises $L(\pi)$. On the lower level, for each fixed π we need to find the value of $L(\pi)$ by solving the problem (3)–(6). Note, this solution generates the schedule (S, T). Due to (7) the latter problem (lower level) can be decomposed still further into subproblems associated with individual machines. In the sequel we provide some special properties suitable for constructing a solution algorithm.

Property 1. For any π_i, $i \in \mathcal{M}$ we have

$$L(\pi_i) \ge \frac{1}{2}[C_{\max}(\pi_i) - K]^+ \tag{9}$$

where $K = \max_{j\in\mathcal{N}} c_j$ and $C_{\max}(\pi_i)$ is the makespan value for the permutation π_i in the the problem of scheduling jobs from the set $\mathcal{N}_i$ on a single machine with job *heads* (ready times) $r_{ij} = b_j - p_{ij}$, *tails* (delivery times) $q_j = K - c_j$ to minimise the makespan $\max_{j\in\mathcal{N}_i}(C_j + q_j)$.

Proof. We refer to the well-known formulae providing the makespan value for the single-machine problem with heads and tails and the makespan criterion, see e.g. [3], which takes in our case the following form

$$C_{\max}(\pi_i) = \max_{1\le k\le l\le n_i} (r_{i\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} + q_{\pi_i(l)}). \tag{10}$$

Observe that for every pair $1 \le k \le l \le n_i$ and every $S \in \mathcal{S}(\pi)$, we have

$$\max_{j\in\mathcal{N}_i} [b_j - C_j, C_j - c_j]^+ = \max_{1\le s\le n_i} [b_{\pi_i(s)} - C_{\pi_i(s)}, C_{\pi_i(s)} - c_{\pi_i(s)}]^+$$

$$\ge \max\{[b_{\pi_i(k)} - C_{\pi_i(k)}]^+, [C_{\pi_i(l)} - c_{\pi_i(l)}]^+\} \ge \frac{1}{2}[b_{\pi_i(k)} - C_{\pi_i(k)} + C_{\pi_i(l)} - c_{\pi_i(l)}]^+$$

$$= \frac{1}{2}[b_{\pi_i(k)} - p_{i\pi_i(k)} + \sum_{s=k}^{l} p_{i\pi_i(s)} + K - c_{\pi_i(l)} - K]^+ \ge \frac{1}{2}[r_{i\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} + q_{\pi_i(l)} - K]^+. \tag{11}$$

Taking the maximum of right-hand sides of (11), applying (10) and some obvious dependencies we obtain

$$L(\pi_i) \ge \max_{1\le k\le l\le n_i} \frac{1}{2}[r_{i\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} + q_{\pi_i(l)} - K]^+$$

$$= \frac{1}{2}[\max_{1\le k\le l\le n_i} (r_{i\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} + q_{\pi_i(l)}) - K]^+ = \frac{1}{2}[C_{\max}(\pi_i) - K]^+ \tag{12}$$

which completes the proof of this property.

Property 2. For any π_i, $i \in \mathcal{M}$ we have

$$L(\pi_i) \ge [L_{\max}(\pi_i)]^+ \tag{13}$$

where $L_{\max}(\pi_i)$ is the lateness value for the permutation π_i in the the problem of scheduling jobs from the set $\mathcal{N}_i$ on a single machine with job *heads* (ready times) a_j, *due dates* c_j to minimise the maximum lateness $\max_{j\in\mathcal{N}_i}(C_j - c_j)$.

Proof. Extending the formulae from [3] to the considered case we obtain the expression on the maximal lateness in the following form

$$L_{\max}(\pi_i) = \max_{1\le k\le l\le n_i} (a_{\pi_i(k)} + \sum_{s=k}^{l} p_{i\pi_i(s)} - c_{\pi_i(l)}). \tag{14}$$

Next, from (4) and (6), for any $1 \le k \le l \le n_i$ we have

$$C_{\pi_i(l)} \ge S_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)}. \tag{15}$$

Applying this fact and the condition (5) we can write for any $S \in \mathcal{S}(\pi)$ the following sequence of inequalities

$$\max_{j\in\mathcal{N}_i} [b_j - C_j, C_j - c_j]^+ = \max_{1\le s\le n_i} [b_{\pi_i(s)} - C_{\pi_i(s)}, C_{\pi_i(s)} - c_{\pi_i(s)}]^+$$

$$\ge [C_{\pi_i(l)} - c_{\pi_i(l)}]^+ \ge [S_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} - c_{\pi_i(l)}]^+ \ge [a_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} - c_{\pi_i(l)}]^+ \tag{16}$$

Taking the maximum of right-hand sides of (16), applying (14) and some obvious dependencies we obtain

$$L(\pi_i) \ge \max_{1\le k\le l\le n_i} [a_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} - c_{\pi_i(l)}]^+ = [L_{\max}(\pi_i)]^+ \tag{17}$$

which completes the proof of the property.

These two properties imply the following one (the proof is clear).

Property 3. For any $\pi \in \Pi$ we have

$$L(\pi) \ge [\frac{1}{2}(C_{\max}(\pi) - K), L_{\max}(\pi)]^+ \stackrel{\text{def}}{=} LB(\pi) \ge 0 \tag{18}$$

where

$$C_{\max}(\pi) = \max_{i\in\mathcal{M}} C_{\max}(\pi_i), \quad L_{\max}(\pi) = \max_{i\in\mathcal{M}} L_{\max}(\pi_i). \tag{19}$$

The subsequent property shows that the obtained bound $LB(\pi)$ is tight.

Property 4. For any $\pi \in \Pi$ there exist $S \in \mathcal{S}(\pi)$ such that

$$L(\pi) \le LB(\pi) \tag{20}$$

Proof. We propose the method of constructing such schedule and next we show that it has the required property. The schedule is given by the recursive formulae

$$S_{\pi_i(s)} = \max\{a_{\pi_i(s)}, S_{\pi_i(s-1)} + p_{i\pi_i(s-1)}, b_{\pi_i(s)} - p_{i\pi_i(s)} - LB(\pi)\}, \ s = 1, \ldots, n_i, \ i \in \mathcal{M}, \tag{21}$$

where $\pi_i(0) = 0$, $i \in \mathcal{M}$, and $S_0 + p_0 = -\infty$.

It is easy to verify that values S_j found by (21) satisfy constraints (4)–(6). Next, by (21) we immediately have

$$b_{\pi_i(s)} - C_{\pi_i(s)} = b_{\pi_i(s)} - S_{\pi_i(s)} - p_{i\pi_i(s)}$$

$$= b_{\pi_i(s)} - \max\{a_{\pi_i(s)}, S_{\pi_i(s-1)} + p_{i\pi_i(s-1)}, b_{\pi_i(s)} - p_{i\pi_i(s)} - LB(\pi)\} - p_{i\pi_i(s)}$$

$$\leq b_{\pi_i(s)} - (b_{\pi_i(s)} - p_{i\pi_i(s)} - LB(\pi)) - p_{i\pi_i(s)} \leq LB(\pi), \tag{22}$$

and this holds for any s, $1 \leq s \leq n_i$, and any i, $i \in \mathcal{M}$. Thus we conclude that

$$b_j - C_j \leq LB(\pi),\ j \in \mathcal{N}. \tag{23}$$

Next, we will show that also $C_j - c_j \leq LB(\pi)$ for any $j \in \mathcal{N}$. For each fixed i, $i \in \mathcal{M}$, and fixed l, $1 \leq l \leq n_i$, let k be the index ($1 \leq k \leq l$) such that $C_{\pi_i(k-1)} < S_{\pi_i(k)}$, and $C_{\pi_i(s-1)} = S_{\pi_i(s)}$ for $s = k+1, \ldots, l$. Note that

$$C_{\pi_i(l)} = S_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)}. \tag{24}$$

¿From (21) it has to be $S_{\pi_i(k)} = \max\{a_{\pi_i(k)}, b_{\pi_i(k)} - p_{i\pi_i(k)} - LB(\pi)\}$. Consider separately two subcases:

(a) Case "$S_{\pi_i(k)} = b_{\pi_i(k)} - p_{i\pi_i(k)} - LB(\pi)$". Applying (24), (21) and (18) we have

$$C_{\pi_i(l)} - c_{\pi_i(l)} = S_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} - c_{\pi_i(l)} = b_{\pi_i(k)} - p_{i\pi_i(k)} - LB(\pi) + \sum_{u=k}^{l} p_{i\pi_i(u)} - c_{\pi_i(l)}$$

$$= r_{i\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} + q_{\pi_i(l)} - K - LB(\pi) \leq C_{max}(\pi_i) - K - LB(\pi)$$

$$\leq C_{max}(\pi) - K - \frac{1}{2}(C_{max}(\pi) - K) \leq \frac{1}{2}(C_{max}(\pi) - K) \leq LB(\pi) \tag{25}$$

(b) Case "$S_{\pi_i(k)} = a_{\pi_i(k)}$". Similarly we have

$$C_{\pi_i(l)} - c_{\pi_i(l)} = S_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} - c_{\pi_i(l)} = a_{\pi_i(k)} + \sum_{u=k}^{l} p_{i\pi_i(u)} - c_{\pi_i(l)}$$

$$\leq L_{max}(\pi_i) \leq L_{max}(\pi) \leq LB(\pi) \tag{26}$$

Both (25) and (26) provides

$$C_j - c_j \leq LB(\pi),\ j \in \mathcal{N}. \tag{27}$$

Finally, (23) and (27) show that

$$L(\pi) = \max_{j \in \mathcal{N}}[b_j - C_j, C_j - c_j]^+ \leq LB(\pi) \tag{28}$$

which yields the property required for the schedule found by (21) and completes the proof.

The problem stated can be transformed into two problems (solved simultaneously) of scheduling jobs from the set $\mathcal{N}$ on m parallel non-uniform, unrelated machines, with different assumptions and different types of criteria. Note that for the given π the value $L(\pi)$ and the appropriate schedule (S, T) can be found in a time $O(n)$.

3. Further properties

It is also convenient in the analysis to use a simple graph representation of a fixed job processing order π. Being honest, this graph is used for the calculation and interpretation of two different values $C_{\max}(\pi)$ and $L_{\max}(\pi)$. It has the same structure for both mentioned cases but differs with weights assigned to arcs. Precisely, the graph takes the form of $\mathcal{G}(\pi) = (\mathcal{N} \cup \{o, *\}, \mathcal{A} \cup \mathcal{A}(\pi))$, with the set of nodes $\mathcal{N}$ extended by two auxiliary nodes o (start) and $*$ (end), and the set of arcs $\mathcal{A} = \bigcup_{j \in \mathcal{N}} \{(o, j), (j, *)\}$ and $\mathcal{A}(\pi) = \bigcup_{i \in \mathcal{M}} \bigcup_{j=1}^{n_i - 1} \{(\pi_i(j), \pi_i(j+1))\}$. Each arc $(o, \pi_i(j)) \in \mathcal{A}$ has weight $r_{i\pi_i(j)}$ ($a_{\pi_i(j)}$, respectively), whereas each arc $((\pi_i(j), *) \in \mathcal{A}$ has weight $q_{\pi_i(j)}$ (minus $c_{\pi_i(j)}$, respectively). Each node $\pi_i(j) \in \mathcal{N}$ has weight $p_{i\pi_i(j)}$. Nodes o and $*$ have weight zero. The makespan $C_{\max}(\pi)$ (lateness $L_{\max}(\pi)$, respectively) equals the length of the longest path in this graph. To distinguish these two cases we use C-graph and L-graph names. We define the notion *block* B as follows. If $LB(\pi) = 0$ we say that block is empty, π provides optimal solution. Otherwise we consider two cases. If $L_{\max}(\pi) \geq \frac{1}{2}(C_{\max}(\pi) - K)$ then we define block as the sequence of nodes $B = (g, \ldots, h)$ such that $(o, g, \ldots, h, *)$ is a critical path in L-graph. If $L_{\max}(\pi) < \frac{1}{2}(C_{\max}(\pi) - K)$ we define block as the sequence of nodes $B = (g, \ldots, h)$ such that $(o, g, \ldots, h, *)$ is a critical path in C-graph. Block corresponds to the sequence of jobs (parts) processed on some machine i^* consecutively from the position k^* to the position l^* without inserted machine idle time, i.e.

$$\pi_{i^*}(k^*) = g, \quad \pi_{i^*}(l^*) = h, \quad k^* \leq l^*, \tag{29}$$

thus critical for the criterion $L(\pi)$ value.

4. A solution method

To solve the problem stated we refer to the Adaptive Memory Search (AMS). Applications of AMS to similar problems have been presented in previous papers, see for example [4, 5, 6], so we will show only some elements valid for this application. The basic idea of AMS follows from [2]. The approach is particularly recommended for solving some very hard discrete and combinatorial optimisation problems. Fundamentals of AMS derive from the natural solution process performed by man. AMS carried out the local search through the solution space with the use of semi-learned memory of the search history to avoid repetitions and to guide the search into promising regions of the space. It has been

verified experimentally in many applications that AMS provides solutions very close to optimal in a relatively short time.

Based on many conclusions obtained in previous studies, there has been considered only the neighbourhood created by so-called *insertion moves*. In our case, this type of moves is associated with π and can be defined by a triple $v = (j, i, y)$, such that job (part) j is removed from $\mathcal{N}_{t_j}$ and then inserted in $\mathcal{N}_i$ in position y in the permutation π_i (y becomes new position of j in π_i extended in such a way). Clearly, it has to be $1 \le y \le n_i + 1$ if $i \ne t_j$, $1 \le y \le n_i$ if $i = t_j$. The neighbourhood of π consists of orders $(\pi)_v$ generated by moves v from a given set. Using natural, simple approach one can propose the *insertion neighbourhood* $\{(\pi)_v : v \in \mathcal{V}(\pi)\}$ generated by the move set

$$\mathcal{V}(\pi) = \bigcup_{j \in \mathcal{N}} \bigcup_{i \in \mathcal{M}} \mathcal{V}_{ji}(\pi) \tag{30}$$

where $\mathcal{V}_{ji}(\pi) = \bigcup_{y=1}^{n_i+1} \{(j, i, y)\}$ if $i \ne t_j$ and $\mathcal{V}_{ji}(\pi) = \bigcup_{y=1}^{n_i} \{(j, i, y)\}$ if $i = t_j$. To avoid solution repetitions, set $\mathcal{V}(\pi)$ is on-line examined to remove redundant moves (moves are redundant if they provide the same solutions). The number of moves in $\mathcal{V}(\pi)$ is $(n-1)^2$. The neighbourhood based on $\mathcal{V}(\pi)$ can be searched completely in a time $O(n^3)$.

The philosophy of using so called reduced neighbourhood is associated with detection a priori and avoiding making so called useless moves (moves that do not improve the current goal function value). The detection of some kinds of useless moves is offered by the block notion. It is summarised in the following list of properties.

Property 5. If $j \notin \mathcal{B}$, then for any $i \in \mathcal{M}$ all moves from $\mathcal{V}_{ji}(\pi)$ are useless.

Property 6. If $j \in \mathcal{B} \setminus \{g, h\}$, then each move $v = (j, i^*, y)$, $k^* < y < l^*$, is useless.

Property 7. If $j = g$, then each move $v = (j, i^*, y)$, $1 \le y < k^*$, is useless.

Property 8. If $j = h$, then each move $v = (j, i^*, y)$, $l^* < y \le n_{i^*}$, is useless.

Property 5 provides the highest reduction of the move set. Proofs of properties can be done analysing paths in the graph $\mathcal{G}(\pi)$. Employing Properties 5 – 8 we propose the following *reduced set of moves*

$$\mathcal{W}(\pi) = \bigcup_{j \in \mathcal{B}} \bigcup_{i \in \mathcal{M}} \mathcal{W}_{ji}(\pi), \tag{31}$$

where sets $\mathcal{W}_{ji}(\pi)$ are defined only for $j \in \mathcal{B}$ in the following manner. If $i \ne i^*$ we set $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi)$. If $i = i^*$ and $j \in \mathcal{B} \setminus \{g, h\}$ we set $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \{v = (j, i, y) : k^* <$

$y < l^*\}$. If $i = i^*$ and $j = g$ than $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \{v = (j, i, y) : 1 \leq y < k^*\}$. If $i = i^*$ and $j = h$ than $\mathcal{W}_{ji}(\pi) = \mathcal{V}_{ji}(\pi) \setminus \{v = (j, i, y) \div l^* < y \leq n_{i^*}\}$. Clearly set $\mathcal{W}(\pi)$ is also scanned to eliminate redundant moves. The number of moves in $\mathcal{W}(\pi)$ is m times less than that of $\mathcal{V}(\pi)$. The neighbourhood based on $\mathcal{W}(\pi)$ possesses the *connectivity property* (the proof can be done by analogy to that from [6]).

The general scheme and remain details (e.g. memory attributes and structures) of the proposed AMS algorithm can be taken from [6]. Some additional theoretical research should be done to design the *search accelerator* which is the specific decomposition and aggregation of calculation to reduce the algorithm running time. All advantageous properties of the proposed there algorithm can be obtained also in this case.

BIBLIOGRAPHY

1. Baker K., Scudder G.: Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 30:22–36, 1990.
2. Glover F., Laguna M.: Tabu Search. Kluwer, 1997.
3. Grabowski J., Nowicki E., Zdrzałka S.: A Block Approach for Single Machine Scheduling with Release Dates and Due Dates. *European Journal of Operational Research*, 26:278–285, 1986.
4. Nowicki E., Smutnicki C.: A fast taboo search algorithm for the job shop. *Management Science*, 6:797–813, 1996.
5. Nowicki E., Smutnicki C.: A fast taboo search algorithm for the flow shop. *European Journal of Operational Research*, 91:160–175, 1996.
6. Smutnicki C.: Optimization and control in just-in-time manufacturing systems. Oficyna Politechniki Wrocławskiej, 1997.
7. Vaessens R., Aarts E., Lenstra J.K.: Job Shop Scheduling by Local Search. *INFORMS Journal on Computing*, 8:302-317, 1996.

Recenzent: Dr hab.inż. Eugeniusz Toczyłowski, prof. Politechniki Warszawskiej

Abstract

A problem of precise on time supply of various items produced by a manufacturing cell having a number of unrelated parallel machines has been considered. Some special properties of the problem has been proved. A transformation to some classic problems

from the scheduling theory has been shown. The proposed and developed approach can be extended to cover more complex problems of this type. The base of this approach refers to graph properties (including generalisation of so called block properties) and local search algorithms. A significant attention has been paid to these theoretical properties which are the most useful for the Adaptive Memory Search (AMS) algorithm.