Open Shop Scheduling Problems with Late Work Criteria

Jacek Błażewicz [1], Erwin Pesch [2], Małgorzata Sterna [3], Frank Werner [4]

[1] Institute of Computing Science, Poznań University of Technology
Piotrowo 3A, 60-965 Poznań, Poland
blazewic@sol.put.poznan.pl

[2] Institute of Information Systems, FB 5 - Faculty of Economics, University of Siegen
Hölderlinstr. 3, 57068 Siegen, Germany
pesch@fb5.uni-siegen.de

[3] Institute of Computing Science, Poznań University of Technology
Piotrowo 3A, 60-965 Poznań, Poland
Malgorzata.Sterna@cs.put.poznan.pl

[4] Faculty of Mathematics, Otto-von-Guericke-University
PSF 4120, 39016, Magdeburg, Germany
Frank.Werner@mathematik.uni-magdeburg.de

Abstract: The paper concerns the application of a non-classical performance measure, a late work criterion (Y, Y_w), to scheduling problems. It estimates the quality of the obtained solution with regard to the duration of the late parts of tasks not taking into account the quantity of this delay. The paper provides the formal definition of the late work parameter, especially in the shop environment, together with its practical motivation. It contains general complexity studies and the results of investigating open-shop scheduling cases, i.e. two polynomial time algorithms for problems $O \mid pmtn, r_i \mid Y_w$ and $O2 \mid d_i = d \mid Y$, as well as the binary *NP*-hardness proof for $O2 \mid d_i = d \mid Y_w$.

Keywords: scheduling problems, optimality criteria, late work criterion, open-shop problem.

1. Introduction

The scheduling theory concerns problems of allocating resources to perform a set of activities in order to achieve a certain goal. This purpose of the scheduling process could be considered as finding a feasible solution of the analyzed problem as well as determining the best solution with reference to a given optimality criterion (cf. [5, 8, 13, 21]). The performance measures define the quality of the obtained schedule based on input parameters of particular tasks and, usually, on their completion times. They take into account all tasks existing in the system in order to estimate its behavior from a global point of view. The selection of the objective function results from the peculiarities of the considered problem; it depends on objectives that are important for the scheduling process. The rapid development of industrial systems, which can be supported by the scheduling theory, results in the necessity of continuous research in this branch of science. Trying to cover realistic problems, besides proposing new approaches and models, new parameters and criteria are considered as well.

The paper concerns a performance measure based on the amount of late work in the system [4, 7, 22, 26]. This objective function was first proposed in the context of parallel machines [4, 6] and then applied to the one-machine scheduling problem [22, 23]. Based on this concept, a new branch of the research has also appeared which modifies the original formulation of the late work for the real-time applications by considering so called imprecise computations [cf. 3, 15].

In general [4], the late work Y_i for task T_i with the processing time p_i finished at time C_i can be defined as the amount of the work that is executed after the due date d_i. In the non-preemptive case, the late work for task T_i is defined as $Y_i = min\{max\{0, C_i - d_i\}, p_i\}$. In other words, it is determined as $Y_i = min\{D_i, p_i\}$, where $D_i = max\{0, C_i - d_i\}$ denotes the tardiness for task T_i (according to the notation provided in [5]). The preemptive case is defined in a similar way, but it requires summing all parts of a task, possibly preempted, executed after its due date. For task T_i executed in k_i parts, where the k-th part starts at time S_i^k and finishes at time C_i^k, the late work is defined as $Y_i = \sum_{k=1}^{k_i} max\{C_i^k - max\{d_i, S_i^k\}, 0\}$. Taking into account the late work for all n activities being analyzed in a system, two basic criteria can be defined, such as the total and total weighted late work, i.e. $Y = \sum_{i=1}^{n} Y_i$ and $Y_w = \sum_{i=1}^{n} w_i Y_i$, respectively.

The late work based criteria belong to the group of performance measures involving due dates. However, classical criteria, formulated for problems with deadlines (due dates), such as e.g. the maximum lateness or mean tardiness, calculate the penalty for solutions where some tasks exceed their due dates with respect to the time of their completion. In some applications, the penalty should be determined with reference to the amount of the late work independently of the time of the completion of a task. In the case of the late work criterion, only the amount of late work is important, if the whole task is delayed.

The late work criteria are not artificial performance measures. They find their motivation in real-time systems. For example, the late work based approach can be applied in control systems [4, 6], where the amount of data not collected from sensing devices before the due date corresponds to the late work. In such systems, sensing devices expose data, which are collected by the control process in predefined time windows, between release and due dates. If the data are exposed after the time required, they cannot be used by the control procedure, which must work out the decision based only on the measurements gathered in the feasible interval. Thus, the information not collected before the due dates is lost and influences the precision and the quality of the control process. The less information is lost the more adequate decisions can be taken by the control procedure. A similar situation appears in a computer controlled manufacturing system environment (CIM, FMS), where an adaptive control method can base its computations only on data collected

before their start. After exceeding due dates, samples become unavailable and the information represented by them is lost decreasing the quality of estimations [22, 23]. The late work criteria can be also analyzed in agriculture, especially in all cases concerning perishable goods, as for example harvesting [22]. In this case, tasks represent different stretches of land that have to be harvested. Because they differ in climate and soil conditions as well as in the corn culture, they have different times at which crops collecting should be started and finished. Processing times estimate quantities of crops. After a given due date, crops perish causing financial loss. Minimizing the total late work is equivalent to minimizing the amount of wasted crops. Summing up, the late work criteria apply to all those scheduling problems that concentrate on the amount of late work delayed after a given due date not on the duration of this delay.

The late work criterion was first proposed in the context of parallel machines by Błażewicz [4], who showed the strong *NP*-hardness of problem $P \mid r_i \mid Y_w$. The proof concept is based on the complexity analysis for the minimal mean tardiness problem [13]. The preemptive case $P \mid pmtn, r_i \mid Y_w$ is polynomially solvable by a transformation to a min-cost flow problem. It results in an algorithm of the overall complexity $O(n^7 \log n)$, where n denotes the number of tasks. This approach was further extended by Błażewicz and Finke [6] to the case of a fixed number of uniform machines $Qk \mid pmtn, r_i \mid Y_w$. They proposed an $O(k^3 n^7 \log kn)$ method, where k denotes the number of machines and n equals the number of tasks.

The concept of late work has also been considered by Potts and Wassenhove, who concentrated on one-machine scheduling problems. They showed the *NP*-hardness of problem $1 \mid \mid Y$ by a transformation from the knapsack problem [22]. The authors proposed for this case a pseudo-polynomial dynamic programming algorithm of $O(nUB)$ complexity, where n is the number of tasks and UB denotes an upper bound of the criterion value obtained by an application of the earliest due date list method [22, 23]. The formulation of a pseudo-polynomial algorithm allows one to classify $1 \mid \mid Y$ as binary *NP*-hard. There are also some special cases of the considered problem analyzed. The assumption of a common due date for all tasks ($1 \mid d_i = d \mid Y$) makes the case trivial [22], because any schedule is optimal with the criterion value equal to $max\{\sum_i p_i - d, 0\}$. Similarly, introducing identical processing times ($1 \mid p_i = p \mid Y$) allowed one to solve this scheduling problem in polynomial time by running a modified earliest due date list algorithm (EDD) of complexity $O(n \log n)$ [22]. The EDD approach applies also to the preemptive case of the considered problem, $1 \mid pmtn \mid Y$, which appears to be easier than its non-preemptive version.

As we have already mentioned, the late work has become also the inspiration of the research in the field of real-time systems [25]. Based on this concept, the idea of imprecise computations has been developed [3, 11, 15, 16, 24]. It assumes that a hard real-time task is logically divided into mandatory and optional parts. The first one must be completed before the task deadline in order to

obtain a feasible solution, while the latter may be late or not finished at all. The optional part refines the mandatory one and does not influence the feasibility of a schedule, but increases the precision of computations and reduces the error of a result. This specific heterogeneous character of the task definition causes that the imprecise computation model, although it has its origins in the concept of the late work, belongs to a different research stream.

PROBLEM	COMPLEXITY	REFERENCE
$P \mid r_i \mid Y_w$	unary *NP*-hard	[4]
$P \mid pmtn, r_i \mid Y_w$	$O(n^7 \log n)$	[6]
$Qk \mid pmtn, r_i \mid Y_w$	$O(k^3 n^7 \log kn)$	[6]
$1 \mid pmtn \mid Y$	$O(n \log n)$	[22]
$1 \mid \mid Y$	binary *NP*-hard	[22]
$1 \mid d_i = d \mid Y$	$O(n)$	[22]
$1 \mid p_i = p \mid Y$	$O(n \log n)$	[22]

Table 1. Results for the late work criteria

Summing up, the field of late work scheduling has not been widely explored (see Table 1) which causes some problems in estimating the complexity of other cases, not analyzed yet. However, based on the gathered results, the late work criterion seemed to be settled in the difficulty rank between the maximum lateness and mean tardiness criteria [4].

The paper is organized as follows. Section 2 provides the results of general complexity studies concerning the difficulty of the scheduling problems with the late work criteria in comparison with classical performance measures. Section 3 shows the results obtained for the open-shop environment. We present polynomial-time algorithms for problems $O \mid pmtn, r_i \mid Y_w$ and $O2 \mid d_i = d \mid Y$, as well as an *NP*-hardness proof and a dynamic programming approach for the weighted case $O2 \mid d_i = d \mid Y_w$. The paper finishes with some conclusions provided in Section 4.

2. General Complexity Studies

The classical performance measures form a graph of criteria interrelations [5], which is often very helpful in the analysis of open scheduling problems. Relations among optimality criteria can deliver some suggestions on an expected complexity of a newly considered case and, in this way, guide the research to the most promising direction. The late work criterion has not been included in the mentioned interrelation graph so far. In order to settle the relation among the new performance measure and the classical ones, strict reducibility rules will be formulated using the reducibility and equivalence relations [20].

We say that problem P' is reducible to problem P ($P' \propto P$) if for any instance of problem P' an instance of problem P can be constructed in polynomial time, such that solving the instance of P will solve the instance of P'. Problems P' and P are equivalent ($P' \approx P$) if $P' \propto P$ and $P \propto P'$. Moreover, we use the three-field notation $\alpha \mid \beta \mid \gamma$, where symbol α describes the machine environment, β describes the task and resource characteristics and γ denotes the optimality criterion.

Theorem 1. $\alpha \mid \beta \mid L_{max} \propto \alpha \mid \beta \mid Y$

Proof:

P' denotes problem $\alpha \mid \beta \mid L_{max}$ and P problem $\alpha \mid \beta \mid Y$. For an instance of problem P' with due dates d_i', we set L' as a threshold value of L_{max} and construct an instance of problem P by setting $d_i = d_i' + L'$ for each i. P' has a solution with a value smaller than or equal to L' if and only if P has a solution with a value smaller than or equal to 0. We will concentrate on the preemptive case, as on the most general one. Because the lateness is calculated with reference to the completion time of the last k_i-th part $C_i^{k_i}$ of a task we have

$$L_{max} \le L' \Leftrightarrow \max_i \{ C_i^{k_i} - d_i' \} \le L' \Leftrightarrow \forall_i (C_i^{k_i} - d_i' \le L') \Leftrightarrow \forall_i (C_i^{k_i} - (d_i' + L') \le 0) \Leftrightarrow \forall_i (C_i^{k_i} \le d_i).$$

The fact that $\forall_{1 \le k \le k_i} C_i^k \le C_i^{k_i}$ implies that $\forall_i \forall_{1 \le k \le k_i} (C_i^k \le d_i)$. Moreover, for any task, we have $\forall_{1 \le k \le k_i} S_i^k < C_i^k$, so $\forall_{1 \le k \le k_i} S_i^k < d_i$ and $\forall_{1 \le k \le k_i} \max\{d_i, S_i^k\} = d_i$. In consequence, we obtain:

$$\forall_i \forall_{1 \le k \le k_i} (C_i^k \le \max\{d_i, S_i^k\}) \Leftrightarrow \forall_i \forall_{1 \le k \le k_i} (C_i^k - \max\{d_i, S_i^k\} \le 0) \Leftrightarrow$$

$$\forall_i \forall_{1 \le k \le k_i} (\max\{ C_i^k - \max\{d_i, S_i^k\}, 0 \} = 0) \Leftrightarrow \forall_i (\sum_{k=1}^{k_i} \max\{ C_i^k - \max\{d_i, S_i^k\}, 0 \} = 0) \Leftrightarrow$$

$$\forall_i (Y_i = 0) \Leftrightarrow \sum_i Y_i \le 0.$$

+

Taking into account Theorem 1, the classical graph of interrelations among different optimality criteria [5] can be extended with the late work criteria as it is shown in Figure 1.

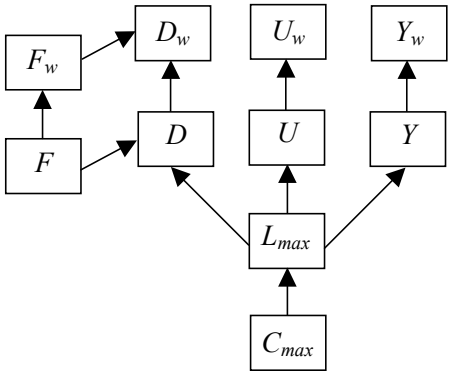

Figure 1. The extension of the graph of interrelations among optimality criteria

This extension made it possible to contrast the new criterion to classic ones and supports the analysis of open problems with the late work criteria, which must be at least as difficult as their versions with the maximum lateness one.

The similar analysis of the scheduling cases for tasks of unit-processing times ($p_i = 1$) and one machine or parallel identical as well as uniform machines ($\alpha = \{1, P, Q\}$) showed that problems with the number of tardy tasks criterion are equivalent to the ones with the late work performance measure provided that all problem parameters are integers (i.e. $\alpha \mid p_i = 1 \mid U_w \approx \alpha \mid p_i = 1 \mid Y_w$). It is enough to observe that, when tasks cannot be preempted and have unit-processing times, then each late task influences both criteria in the same way. Based on this relation some solutions formulated for the number of tardy tasks criteria can be immediately applied to the late work criteria.

The following two scheduling problems with the number of tardy tasks are polynomially solvable: $P \mid p_i = 1, r_i \mid U_w$ and $Q \mid p_i = 1 \mid U_w$. That means that similar approaches can be used for solving those problems with the late work criterion, i.e. $P \mid p_i = 1, r_i \mid Y_w$ (the network approach [1, 8, 27]) and $Q \mid p_i = 1 \mid Y_w$ (the approach by Dessouky et al. [12]). Moreover, the special case of the single machine problem can be solved as a relaxation of problem $1 \mid p_i = p, r_i \mid U_w$ using the algorithm by Baptiste [2].

The equivalencies mentioned above, allowed us also to determine $1 \mid p_i = 1, chains \mid Y$ as the minimal *NP*-hard case, i.e. the easiest problem which is already *NP*-hard. The proof concept is similar to the one proposed by Lenstra and Rinnooy Kan [19] for the number of tardy tasks criterion. Finally, the maximal open problem $Q \mid p_i = 1, r_i \mid Y_w$ can be specified for which the complexity status is unknown but all harder cases are *NP*-hard [27].

3. Late Work Criteria in Shop Environment

The shop environment requires adjusting the definition of the late work parameter. It must take into account the fact that particular tasks form superordinate activities - jobs. The late work is calculated for a job by summing all late parts of tasks constituting this job.

Introducing the following notations:

J_i - the i-th job,

n - the number of jobs J_i,

d_i - the due date for job J_i,

m - the number of machines M_j,

T_{ij} - the task representing processing job J_i on machine M_j,

p_{ij} - the processing time of job J_i on machine M_j,

C_{ij} - the completion time of job J_i on machine M_j,

the non-preemptive late work Y_i for job J_i is defined as: $Y_i = \sum_{T_{ij} \in J_i} min\{max\{0,\ C_{ij}\ -d_i\}, p_{ij}\}$ (cf. Figure 2). For the preemptive case all parts of a preempted task executed after the due date must be added. The definition takes into account all k_{ij} parts of task T_{ij}, where the k-th part starts at time S_{ij}^k and finishes at time C_{ij}^k: $Y_i = \sum_{T_{ij} \in J_i} \sum_{k=1}^{k_{ij}} max\{C_{ij}^k - max\{d_i, S_{ij}^k\}, 0\}$.

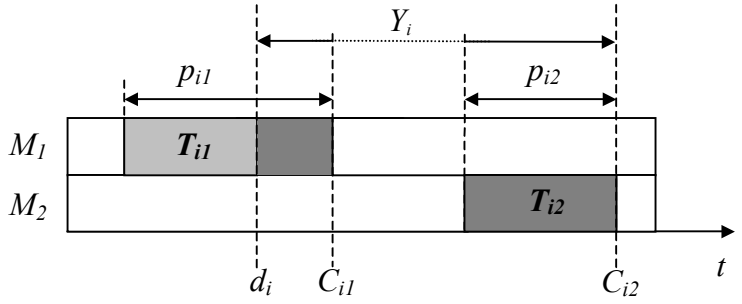

Figure 2. The late work definition for the 2-machine shop problem

Actually, there are no results concerning shop scheduling problems with late work criteria although they have their practical motivations arising especially from agriculture and flexible manufacturing systems [26].

3.1. Problem $O \mid pmtn, r_i \mid Y_w$

First we study an open-shop scheduling problem with release dates and weights where tasks can be preempted $O \mid pmtn, r_i \mid Y_w$. To solve this case, we have to schedule n jobs J_i with weights w_i on m machines M_j (i.e. tasks T_{ij}) with respect to their release dates r_i and due dates d_i in the open shop environment in order to minimize the weighted late work. This problem can be solved in two phases. First, we determine the early parts of particular tasks within predefined time intervals by solving a linear programming problem [9]. Then, we construct an optimal schedule by applying within those intervals the algorithm for problem $O \mid pmtn \mid C_{max}$ [14].

The intervals mentioned above are obtained by sequencing release dates r_i and due dates d_i of all jobs in non-decreasing order. In consequence, after omitting repetitive values, we have a non-decreasing sequence of k different time moments a_k, corresponding to r_i or d_i, where $k \le 2n$. The values a_k form k-1 different time intervals $[a_1, a_2]$, $[a_2, a_3]$, ..., $[a_{k-1}, a_k]$. Moreover, we introduce an additional k-th interval $[a_k, a_{k+1}]$ where $a_{k+1} = a_k + \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}$. The length of the r-th interval is equal to $t_r = a_{r+1} - a_r$.

Solving the linear programming problem formulated below, we determine optimal portions p_{ij}^r of task T_{ij} executed within particular intervals $[a_r, a_{r+1}]$.

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{\substack{r=1 \\ d_i \le a_r}}^{k} p_{ij}^r w_i \tag{1}$$

$$\text{Subject to:} \quad \sum_{\substack{r=1 \\ a_r < r_i}}^{k} p_{ij}^r = 0 \quad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m \tag{2}$$

$$\sum_{\substack{r=1 \\ r_i \le a_r}}^{k} p_{ij}^r = p_{ij} \quad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m \tag{3}$$

$$\sum_{j=1}^{m} p_{ij}^r \le t_r \quad \text{for } i = 1, \ldots, n \text{ and } r = 1, \ldots, k \tag{4}$$

$$\sum_{i=1}^{n} p_{ij}^r \le t_r \quad \text{for } j = 1, \ldots, m \text{ and } r = 1, \ldots, k \tag{5}$$

$$0 \le p_{ij}^r \le p_{ij} \quad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m \text{ and } r = 1, \ldots, k \tag{6}$$

In the linear programming problem, we minimize the weighted sum of those portions of tasks which have been assigned to intervals starting after the job deadline, i.e. weighted late work expressed in term (1). Because no job (J_i) can be executed before its release date (r_i), thus the sum of all portions of tasks assigned to intervals starting before the job release date ($a_r < r_i$) has to be equal to zero (constraints 2). Consequently, whole tasks (i.e. p_{ij} units of work for task T_{ij}) have to be done in the intervals starting after their job release dates ($r_i \le a_r$) that is enforced by constraints (3). Moreover, the sum of all portions of job J_i assigned to a particular interval $[a_r, a_{r+1}]$ cannot exceed the length of this interval t_r (constraints 4), otherwise a job would have to be performed on more than one machine at the same time. Then, the total amount of work assigned in the particular interval $[a_r, a_{r+1}]$ to a single machine M_j cannot exceed the length of this interval t_r (constraints 5), otherwise a machine would have to perform more than one task at the same time. Finally, the portions of tasks in particular intervals have to be non-negative numbers not exceeding the processing times of those tasks (constraints 6).

To find the optimal portions p_{ij}^r of tasks T_{ij} within particular possible time intervals $[a_r, a_{r+1}]$, we have to solve the linear programming problem with $O(knm) = O(n^2m)$ variables and $O(n^2+nm)$ constraints. Taking into account the fact that a solution of the LP problem can be found in polynomial time by Khachiyan's method [9], determining the optimal values p_{ij}^r is also done in polynomial time.

Based on the solution of the linear programming problem, an optimal schedule of problem $O \mid pmtn,\ r_i \mid Y_w$ is constructed as follows.

We consider each time interval $[a_r, a_{r+1}]$ for $r = 1, \ldots, k\text{-}1$ separately and apply the method by Gonzalez and Sahni [14] for problem $O \mid pmtn \mid C_{max}$ to task parts assigned to this interval by the linear programming method, i.e. the tasks T_{ij} with $p_{ij}^r > 0$. All those tasks T_{ij} can be feasibly processed in an analyzed interval $[a_r, a_{r+1}]$, with regard to constraints (3), which require $r_i \le a_r$. Moreover, the due dates d_i are not important at this stage of the analysis, because the optimal assignment of tasks to intervals (from the late work criterion point of view) has been already determined by solving the LP problem.

Summing up, to construct the final solution, for each interval $[a_r, a_{r+1}]$ we have to schedule task parts $p_{ij}^r > 0$ starting from time a_r and minimizing C_{max} for this subset of task parts in order to execute all task parts considered before the end of the interval a_{r+1}. Based on Gonzalez and Sahni's method, the tasks are scheduled within the time [5, 8, 14]:

$$C_{max}^* = a_r + max\ \{\ \max_{j=1,\ldots,m}\ \{\ \sum_{i=1}^{n} p_{ij}^r\ \},\quad \max_{i=1,\ldots,n}\ \{\ \sum_{j=1}^{m} p_{ij}^r\ \}\}.$$

Taking into account the constraints (4) and (5) the optimal schedule length within interval $[a_r, a_{r+1}]$ does not exceed the length of this interval t_r.

We apply Gonzalez and Sahni's [14] method k-1 times obtaining k-1 subschedules for particular intervals $[a_r, a_{r+1}]$, where $r = 1,\ldots, k\text{-}1$. A single run of this method requires $O(s^2(n+m)^{0.5})$ time, where s is the number of tasks assigned to the interval (i.e. the number of tasks T_{ij} with $p_{ij}^r > 0$), which can be reduced to $O(s^2)$ [8]. The task parts assigned to the last interval $[a_k, a_{k+1}]$ are late, because $a_k = \max_{i=1,\ldots,n} \{r_i,\ d_i\}$, and can be sequenced in an arbitrary order at the end of the schedule. Concatenating all subschedules for particular k intervals, we obtain an optimal solution of problem $O \mid pmtn,\ r_i \mid Y_w$.

The result obtained for problem $O \mid pmtn,\ r_i \mid Y_w$ confirms the previously presented relation between the late work criterion and the maximum lateness one. The problem $O \mid pmtn, r_i \mid L_{max}$, which should not be more difficult than its late work version according to the new graph of criteria interrelations, is also polynomially solvable by Cho and Sahni's algorithm [10]. Moreover, it is the maximal polynomially solvable case for the maximal lateness criterion [27] and, consequently, for the late work criteria as well.

3.2. Problem $O2 \mid d_i = d \mid Y$

The solution of problem $O2 \mid d_i = d \mid Y$ is based on the classical approach to problem $O2 \mid \mid C_{max}$ proposed by Gonzalez and Sahni [14]. First, we construct a schedule by Gonzalez and Sahni's method, then we modify it by shifting some jobs in order to minimize the idle time before the common due date and, consequently, to minimize the late work in the system.

We denote the set of jobs as $\mathcal{J} = \{J_1, J_2, ..., J_n\}$. We use symbol $\Pi(X)$ to indicate an arbitrary sequence of jobs from the subset $X \subseteq \mathcal{J}$ and $p_j(X)$ to express the total processing time of this subset X on a particular machine M_j for $j \in \{1, 2\}$, i.e.: $p_j(X) = \sum_{J_i \in X} p_{ij}$. Thus, the total processing time on both machines is determined as $p_1(\mathcal{J})$ and $p_2(\mathcal{J})$. Finally, the sequence of the jobs on the machine M_j is denoted by $\Pi_j(X)$.

As we have mentioned, the initial solution is constructed by the algorithm by Gonzalez and Sahni for problem $O2 \mid \mid C_{max}$. The set of jobs $\mathcal{J}$ is divided into two sets of the longer tasks on the first and the second machine, respectively, i.e. $A = \{J_i \in \mathcal{J} : p_{i1} \geq p_{i2}\}$ and $B = \{J_i \in \mathcal{J} : p_{i1} < p_{i2}\}$. Then, any two different jobs $J_r \in A$ and $J_s \in B$ such that $p_{r1} \geq \max_{J_i \in A} \{p_{i2}\}$ and $p_{s2} \geq \max_{J_i \in B} \{p_{i1}\}$ are chosen. Only if one of the sets A, B is empty, jobs J_r, J_s are taken from the same set, e.g. those with the longest tasks on machines M_1 and M_2, respectively. Furthermore, additional sets obtained by excluding the jobs J_r, J_s from the sets A, B are defined as follows: $A' = A \setminus \{J_r, J_s\}$, $B' = B \setminus \{J_r, J_s\}$.

Similarly as in Gonzalez and Sahni's method we have to consider two symmetric cases when $p_1(\mathcal{J}) - p_{s1} \geq p_2(\mathcal{J}) - p_{r2}$ and the opposite one. In the description of our method and the proof of its optimality we concentrate on the case mentioned, i.e. with $p_1(\mathcal{J}) - p_{s1} \geq p_2(\mathcal{J}) - p_{r2}$.

After determining sets A, B and jobs J_r, J_s, we construct two separate subschedules $(J_s, \Pi(B'))$ and $(\Pi(A'), J_r)$ with arbitrarily ordered jobs from sets A' and B'. Then, both subschedules are joined and the tasks of $(J_s, \Pi(B'))$ processed on machine M_2 are shifted to the right, such that no idle time between the task executions occurs. After placing the task of job J_r on machine M_2 as the first one, we obtain an initial schedule for solving problem $O2 \mid d_i = d \mid Y$ depicted in Figure 3. To find an optimal solution for the late work criterion, we have to minimize the idle time before the common due date.

Figure 3. The initial schedule in the process of solving $O2 \mid d_i = d \mid Y$

This optimization is performed by Algorithm 1 sketched below. Further details of the approach are provided in the proof of its optimality.

In the simplest case (Case 1), when both machines finish their work at the same time (as in Figure 3) the optimal solution for the late work criterion is obtained by scheduling all jobs in the reverse order. Because we have no idle time between tasks, the schedule is optimal. Case 2 describes the situation if the machines finish their work at different times, that means that the special job J_r consists of very long tasks. In this case, depending on the problem instance, Gonzalez and Sahni's sequence can be still optimal (Case 2.1) or it has to be changed by shifting some tasks in order to pack as many jobs as possible before the common due date (Case 2.2). To determine the optimal schedule it is enough to select the best sequence among a few possible ones.

Algorithm 1

if $(p_1(J) - p_{s1} \geq p_2(J) - p_{r2})$, then

construct a solution of problem $O2 \mid\mid C_{max}$ obtaining schedule $\Pi_{GS}(J)$ shifted to the right as in Figure 3;

Case 1. if in schedule $\Pi_{GS}(J)$ we have $p_{r1} \leq p_2(J) - p_{r2}$ i.e. both machines finish their work at the same time (see Fig. 3), then construct an optimal sequence $\Pi^*(J)$ by scheduling all tasks in the reverse order with regard to $\Pi_{GS}(J)$;

Case 2. if in schedule $\Pi_{GS}(J)$ machines differ in finishing times, i.e. $p_{r1} > p_2(J) - p_{r2}$, then

Case 2.1. if the duration of job J_r determines the schedule length, i.e. if $p_{r2} \geq p_{s1} + p_1(A' \cup B')$, then choose the better solution between $\Pi_{GS}(J)$ and the schedule obtained by sequencing tasks in the reverse order with regard to $\Pi_{GS}(J)$;

Case 2.2. otherwise, if the idle time appears only on machine M_2, i.e. $p_{r2} < p_{s1} + p_1(A' \cup B')$, then select the job $J_{s^*} \in J \setminus \{J_r\}$ with the shortest task on machine M_1, i.e. with $p_{s^*1} = \min_{J_i \in J \setminus \{J_r\}} \{p_{i1}\}$ and

Case 2.2.1. if $p_{r2} < p_{s^*1}$, then construct the schedule as follows: $\Pi_1(J) = (J_{s^*}, J_r, \Pi(J \setminus \{J_r, J_{s^*}\}))$, $\Pi_2(J) = (J_r, \Pi(J \setminus \{J_r, J_{s^*}\}), J_{s^*})$. If it includes an idle time before d, then select the better sequence between the one presented and $\Pi_1'(J) = (J_r, \Pi(J \setminus \{J_r\}))$, $\Pi_2'(J) = (\Pi(J \setminus \{J_r\}), J_r)$;

Case 2.2.2. if $p_{s^*1} \leq p_{r2} \leq p_{s1} + p_1(B')$, then split sequence $\Pi(B')$ into $\Pi^1(B')$ and $\Pi^2(B')$, such that the last job in $\Pi^1(B')$ is the first job in set B' which is completed not earlier than at p_{r2}, and schedule jobs as follows: $\Pi_1(J) = (J_s, \Pi^1(B'), J_r, \Pi^2(B'), \Pi(A'))$, $\Pi_2(J) = (J_r, J_s, \Pi(B'), \Pi(A'))$;

Case 2.2.3. if $p_{r2} > p_{s1} + p_1(B')$, then determine set $A'' \subseteq A'$ such that no idle time appears between the jobs in schedule $\Pi_1(J) = (J_s, \Pi(B'), \Pi(A''), J_r, \Pi(A' \setminus A''))$, $\Pi_2(J) = (J_r, J_s, \Pi(B'), \Pi(A'))$ otherwise compare this schedule with $\Pi_1(J)=(J_r, \Pi(A'),\Pi(B'), J_s)$, $\Pi_2(J)=(\Pi(A'),\Pi(B'), J_s, J_r)$ selecting the better one.

else apply the same procedure as for the previous case changing the roles of jobs J_r and J_s, sets A' and B' and the machines M_1 and M_2.

As we have announced, the remaining details of the schedule construction are provided in the optimality proof of Algorithm 1, presented below.

Theorem 2

The solution constructed by Algorithm 1 is an optimal solution of problem $O2 \mid d_i = d \mid Y$.

Proof:

We will present the case study for instances with $p_1(\mathcal{J}) - p_{s1} \geq p_2(\mathcal{J}) - p_{r2}$ (similar to the one presented in Figure 4). The case when $p_1(\mathcal{J}) - p_{s1} < p_2(\mathcal{J}) - p_{r2}$ can be treated symmetrically.

Case 1 $p_{r1} \leq p_2(\mathcal{J}) - p_{r2}$

The sequence (see Figure 3) analyzed in the reverse order (see Figure 4), from the last task to the first one is optimal with regard to the late work criteria independently of the value of the common due date d, because no idle time occurs between particular tasks of jobs executed on both machines (in the solution generated by Gonzalez and Sahni's method the idle time may occur only at the beginning or at the end of schedules on particular machines [14]). The presented case concerns the situation when machine M_1 has a bigger load than M_2 (as in Figures 3, 4) as well as the complementary one with the longer total processing time on machine M_2.

Figure 4.

Case 2 $p_{r1} > p_2(\mathcal{J}) - p_{r2}$

Case 2.1 $p_{r2} \geq p_{s1} + p_1(A' \cup B')$

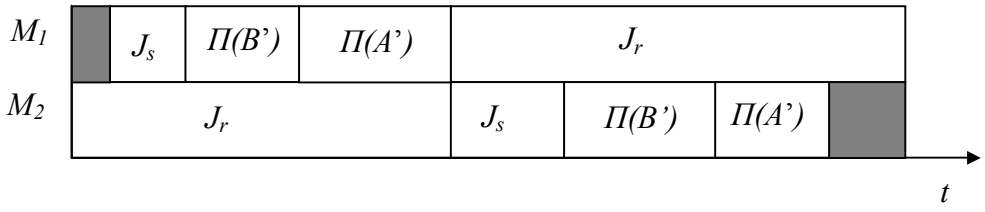

Figure 5.

In the analyzed case (see Figure 5), we determine values Δ_1, Δ_2 equal to the possible unavoidable idle times that may occur on machine M_1 and M_2 assuming that jobs on M_1 are shifted to the left or on M_2 to the right, respectively. The idle times are caused by the very long job J_r, which enforces the shape of the schedule and their values depend on the due date.

$$\Delta_1 = \begin{cases} 0 & \text{if } d \le p_{s1} + p_1(B' \cup A') \\ d - (p_{s1} + p_1(B' \cup A')) & \text{if } p_{s1} + p_1(B' \cup A') < d \le p_{r2} \\ p_{r2} - (p_{s1} + p_1(B' \cup A')) & \text{otherwise} \end{cases}$$

$$\Delta_2 = \begin{cases} 0 & \text{if } d \le p_{s2} + p_2(B' \cup A') \\ d - (p_{s2} + p_2(B' \cup A')) & \text{if } p_{s2} + p_2(B' \cup A') < d \le p_{r1} \\ p_{r1} - (p_{s2} + p_2(B' \cup A')) & \text{otherwise} \end{cases}$$

If $\Delta_1 \le \Delta_2$, then an optimal schedule is obtained by scheduling all jobs as presented in Figure 6, as early as possible from the left to the right. The unavoidable idle time can occur before processing job J_r on machine M_1. Depending on the due date value, this idle time appears after or before d.

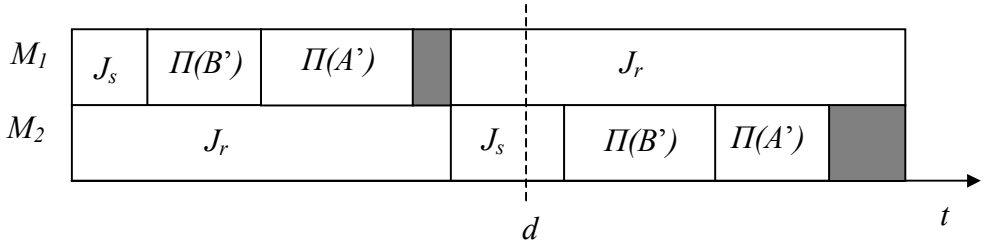

Figure 6.

Otherwise, if $\Delta_1 > \Delta_2$, an optimal schedule is obtained by scheduling all jobs conversely, as presented in Figure 7, i.e. starting from the right and processing each task as early as possible. In this case, the unavoidable idle time may occur only on machine M_2 before processing job J_r. Depending on the value of d, it appears before or after the common due date.

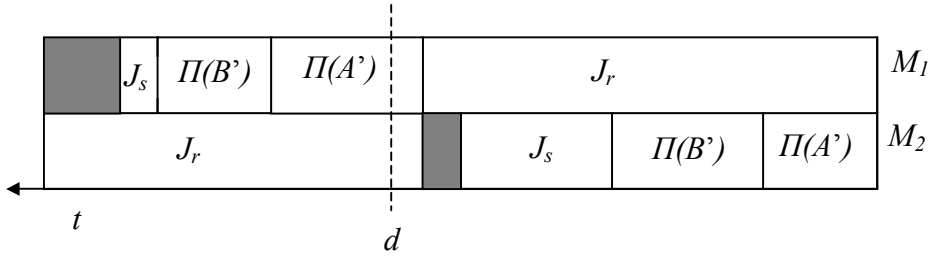

Figure 7.

It is obvious that the schedule constructed minimizes the total late work because the idle time before the task of job J_r processed as the second one is the only one in-between tasks and cannot be avoided. The appearance of idle time is the only reason of shifting the work after the due date and the only source of the late work influencing the criterion value.

Case 2.2 $p_{r2} < p_{s1} + p_1(A' \cup B')$

In the considered case (see Figure 8), job $J_{s*} \in \mathcal{J} \setminus \{J_r\}$ with the shortest task on machine M_1, i.e. with $p_{s*1} = \min\limits_{J_i \in \mathcal{J} \setminus \{J_r\}} \{p_{i1}\}$, is selected. Then, the following subcases must be taken into account. It is worth noting that job J_{s*} may be identical with job J_s.

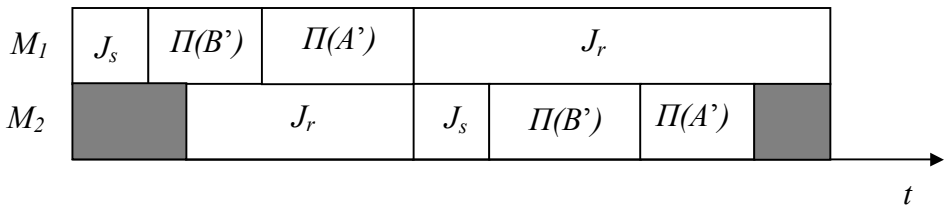

Figure 8.

Case 2.2.1 $p_{r2} < p_{s*1}$

To solve the problem, we choose the job orders on machines M_1 and M_2 as follows:

$$\Pi_1(\mathcal{J}) = (J_{s*}, J_r, \Pi(\mathcal{J} \setminus \{J_r, J_{s*}\})) \text{ and } \Pi_2(\mathcal{J}) = (J_r, \Pi(\mathcal{J} \setminus \{J_r, J_{s*}\}), J_{s*})$$

where all jobs except job J_{s*} are processed first on M_2 (see Figure 9).

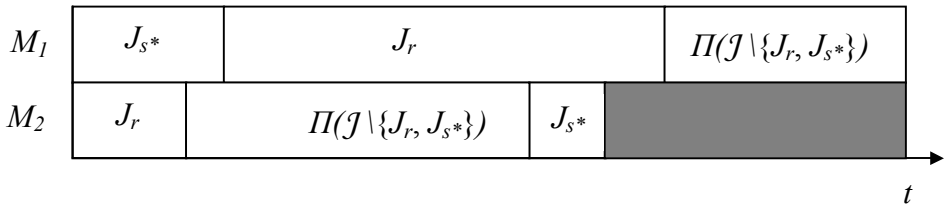

Figure 9.

If on machine M_2, the last job in $\Pi(\mathcal{J} \setminus \{J_r, J_{s*}\})$ is completed before p_{s*1}, then an idle time before processing job J_{s*} occurs on machine M_2. If $d > p_{s*1}$, then this idle time on M_2 occurs before the common due date and we have to compare the obtained schedule with the schedule having the job orders on machine M_1 and on M_2 as follows:

$$\Pi_1'(\mathcal{J}) = (J_r, \Pi(\mathcal{J} \setminus \{J_r\})) \text{ and } \Pi_2'(\mathcal{J}) = (\Pi(\mathcal{J} \setminus \{J_r\}), J_r).$$

Then, the better schedule among both with respect to the criterion value is chosen. If there is no idle time before J_{s*} on M_2, then the constructed schedule is optimal for any value of the common due date because no idle time appears on the machines.

Case 2.2.2 $p_{s*1} \le p_{r2} \le p_{s1} + p_1(B')$

In the analyzed case, the sequence $\Pi(B')$ should be divided into two subsequences $\Pi^1(B')$ and $\Pi^2(B')$, where the last job in $\Pi^1(B')$ is the first job in set B' which is completed not earlier than at p_{r2}. Jobs are processed as follows (see Figure 10):

$$\Pi_1(\mathcal{J}) = (J_s, \Pi^1(B'), J_r, \Pi^2(B'), \Pi(A')) \text{ and } \Pi_2(\mathcal{J}) = (J_r, J_s, \Pi(B'), \Pi(A')).$$

In the obtained schedule no idle time appears between the tasks, so it is optimal with respect to the late work criterion.

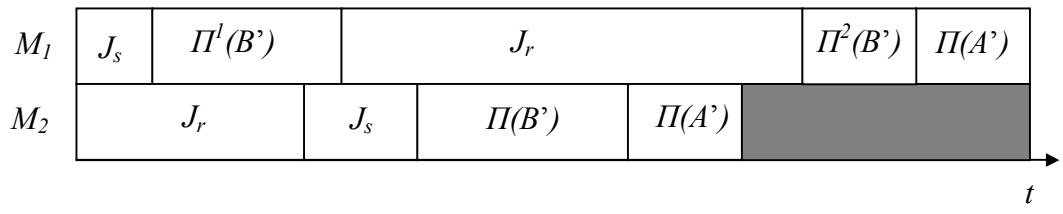

Figure 10.

Case 2.2.3 $p_{r2} > p_{s1} + p_1(B')$

In the analyzed case, we must construct a partial schedule, which does not contain jobs from set A', as it is depicted in Figure 11. We determine value δ as the size of the gap that must be filled on machine M_1 before job J_r and value Δ as the maximal duration of a job placed in gap δ without introducing idle time on M_2, where $\delta = p_{r2} - (p_{s1} + p_1(B'))$ and $\Delta = p_{r2} + p_{s2} + p_2(B') - (p_{s1} + p_1(B'))$.

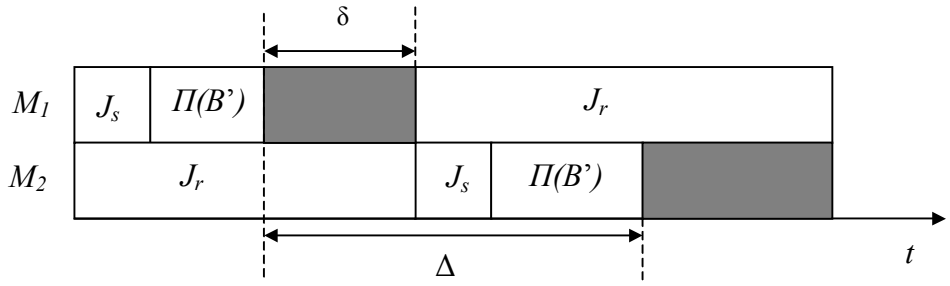

Figure 11.

If set A' contains a job J_k such that $\delta \leq p_{k1} \leq \Delta$, then the mentioned gap on M_1 can be filled without introducing idle time. The machine orders

$$\Pi_1(\mathcal{J}) = (J_s, \Pi(B'), J_k, J_r, \Pi(A' \setminus \{J_k\})) \text{ and } \Pi_2(\mathcal{J}) = (J_r, J_s, \Pi(B'), \Pi(A'))$$

yield an optimal schedule, where jobs J_s, J_k and all jobs of set B' are processed first on M_1 and the other jobs are first processed on M_2.

If such a job J_k fitting the gap does not exist, but there exists a job $J_{k'}$ with $p_{k'1} < \delta$, then we schedule job $J_{k'}$ on both machines after the last job in $\Pi(B')$, and repeat the above consideration after changing δ and Δ to $\delta := \delta - p_{k'1}$ and $\Delta := \Delta - (p_{k'1} - p_{k'2})$.

If, at some step, the set of unscheduled jobs $A'' \subseteq A'$ contains only jobs with processing time $p_{k1} > \Delta$ for $J_k \in A''$ and the current value of Δ, then we must consider every job $J_{k''} \in A''$ as a candidate for the next job on M_1, and all jobs of $A'' \setminus \{J_{k''}\}$ as the jobs scheduled next on M_2.

If for one of those possibilities the completion time of job $J_{k''}$ on M_1 is not greater than the completion time of the last job of set $A'' \setminus \{J_{k''}\}$ on M_2, then an optimal solution has been found. The partial schedule can be completed by scheduling job $J_{k''}$ as the last job on M_2, and all jobs of set $A'' \setminus \{J_{k''}\}$ after job J_r as the last jobs on M_1. In this case, no idle time between the processing of the tasks occurs.

If each of those possibilities leads to idle time on M_2 before the processing of the last job, we must choose the job $J_{k''} \in A''$ which creates the smallest idle time on M_2 before processing job $J_{k''}$ on this machine. $J_{k''}$ must be the last job on M_2 because all other jobs have been already executed and the machine waits for this last task being idle. Then, the jobs from set $\mathcal{J} \setminus \{J_{k''}\}$ on M_1 before $J_{k''}$ are successively removed and scheduled after job J_r on this machine in order to reduce the idle time

before $J_{k''}$ on M_2. If the idle time before the last job on M_2 disappears, then an optimal schedule has been found independently of the value of the common due date d.

If it is impossible to remove completely the idle time before processing the last job on M_2 and this idle time is located before the common due date, i.e. $d > p_2(\mathcal{J} \setminus \{J_{k''}\})$, then the obtained schedule must be compared with the schedule, where job J_r is scheduled first on machine M_1 and the remaining jobs are first processed on M_2, i.e.:

$$\Pi_1(\mathcal{J}) = (J_r, \Pi(A'), \Pi(B'), J_s) \quad \text{and} \quad \Pi_2(\mathcal{J}) = (\Pi(A'), \Pi(B'), J_s, J_r).$$

From both schedules, the one with the smaller criterion value is chosen.

Based on the case study presented above, we can construct an optimal schedule for problem $O2 \mid d_i = d \mid Y$. In the cases where no idle time between the processing of the tasks occurs, no more work can be executed before the common due date because the machines are occupied without any break. Thus, the late work must be minimal and the criterion value has to be optimal. The idle time may appear only in the schedules obtained in Cases 2.1, 2.2.1 and 2.2.3.

In Cases 2.1 and 2.2.1, the idle time that may appear in the solution is unavoidable. It is immediately clear that after comparing the two schedules mentioned and choosing the better one, the obtained schedule cannot be improved with respect to the late work criterion. Moreover, in Case 2.2.1, the final criterion value does not depend on the fact which job J_r has been chosen among the possible ones.

In the remaining Case 2.2.3, a problematic situation arises when an idle time occurs on machine M_2. It means that in some step, the processing time for all remaining jobs J_k from the set A' is bigger than gap Δ, i.e. $p_{k1} > \Delta$. If the idle time before the chosen last job $J_{k''} \in A'$ cannot be removed, it results in the situation presented in Figure 12.

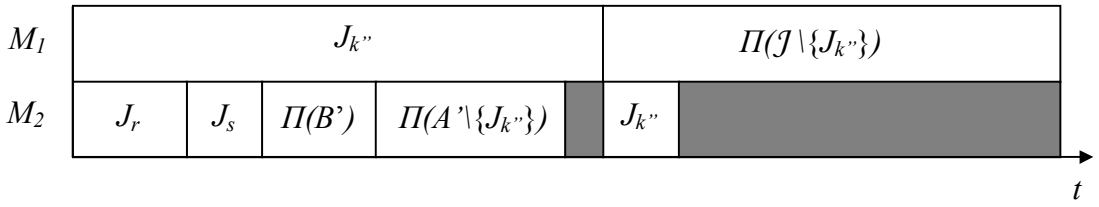

Figure 12.

The depicted schedule is the best one with respect to the late work criterion when job J_r is not processed as the first job on machine M_1 since, among all possible variants, job $J_{k''}$ causing the smallest idle time on M_2 starting at time $t \geq p_{s1} + p_1(B')$ was chosen. Then all jobs before $J_{k''}$ on M_1 have been moved after J_r, but it has not reduced the idle time to zero as it is shown in Figure 12. The described schedule is compared with the best schedule with job J_r processed first on M_1, and,

thus, the better out of them with respect to the criterion value must be an optimal one, depending on the due date value. These two schedules must be compared, because they are the best schedules, when we construct a solution from the left to the right and the other way round. Which one is optimal, depends on the due date value d.

+

Taking into account the fact, that in all cases considered within Algorithm 1 particular jobs have been analyzed at most once, the presented optimal method for problem $O2 \mid d_i = d \mid Y$ keeps the $O(n)$ time complexity of Gonzalez and Sahni's procedure [14].

Moreover, based on Theorem 1 and the fact that problem $O2 \mid\mid L_{max}$ is already *NP*-hard [18], the considered two-machine open-shop problem with different due dates $O2 \mid\mid Y$ is also *NP*-hard. This observation confirms the importance of the newly proven relation between the maximal lateness and late work criteria presented in Section 2.

3.3. Problem $O2 \mid d_i = d \mid Y_w$

The weighted case of the considered two-machine open-shop problem is binary *NP*-hard which is proven by a transformation from the partition problem [13, 17] defined below and the existence of a pseudopolynomial approach presented at the end of the section.

Definition 1

Let a finite set A be given and a positive integer size $s(a_i)$ for each element $a_i \in A$. The decision version of the partition problem is: Does there exist a subset $A' \subseteq A$ such that $\sum_{a_i \in A'} s(a_i) = \sum_{a_i \in A \setminus A'} s(a_i)$?

Theorem 3

The decision version of problem $O2 \mid d_i = d \mid Y_w$ is binary *NP*-complete.

Proof:

For a given instance of the partition problem, we construct an instance of problem $O2 \mid d_i = d \mid Y_w$ as follows:

$$n = |A| + 1,$$
$$p_{i1} = s(a_i), \quad p_{i2} = s(a_i), \quad w_i = 1 \qquad \text{for } i = 1, \ldots, n\text{-}1,$$
$$p_{n1} = B, \quad p_{n2} = B, \quad w_n = 2B+1,$$
$$d = 2B,$$

where $\sum_{a_i \in A} s(a_i) = 2B$.

The set of jobs $\mathcal{J}$ contains the jobs representing the elements of set A and an additional job J_n. We will show that the partition problem has a solution if and only if the corresponding instance of problem $O2 \mid d_i = d \mid Y_w$ has a solution with the criterion value $Y_w \le 2B$.

(if-part) If the partition problem has a solution, then set A can be divided into two sets A' and $A \setminus A'$ such that $\sum_{a_i \in A'} s(a_i) = \sum_{a_i \in A \setminus A'} s(a_i) = B$. The solution of the scheduling problem is constructed as follows (see Figure 13): $\Pi_1(\mathcal{J}) = (J_n, \Pi(A))$, $\Pi_2(\mathcal{J}) = (\Pi(A'), J_n, \Pi(A \setminus A'))$, where the jobs of A' are executed on M_1 after d.

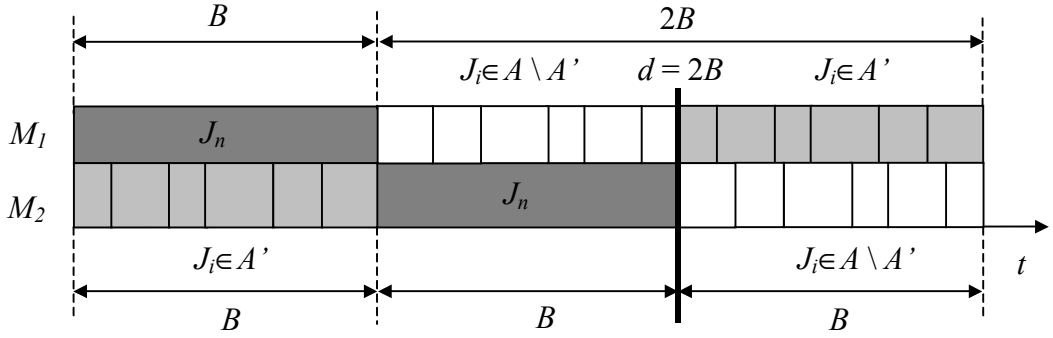

Figure 13.

Due to the construction of the schedule no tasks of the same job overlap on the machines. The amount of the late work is equal to $2B$ and all late tasks have a unary weight. Hence, the criterion value Y_w equals $2B$ and there exists a solution of the scheduling problem with criterion value not greater than $2B$.

(only-if part) Now assume that problem $O2 \mid d_i = d \mid Y_w$ has a solution with $Y_w \le 2B$, and we show that this is possible only if the partition problem has a solution too. Taking into account the fact that all parameters of the problem are integers, the smallest possible portion of a task which can be late is equal to one unit. Each late unit of job J_n would increase the criterion value of $w_n = 2B+1 > 2B$. Hence, job J_n must be processed early and occupies each machine for B time units. We assume that J_n is the first job executed on M_1 (the other case can be considered in a similar way). That means that the remaining jobs can be executed on M_1 one by one without idle times and there are B time units of unary-weighted late work on this machine. Consequently, the gap of length B before J_n on machine M_2 must be completely filled with tasks. Otherwise, the idle time occurs and more than B units of work have to be executed after J_n, i.e. after the due date d, that would make the criterion value bigger than $2B$. The mentioned partition of jobs before and after J_n on M_2 defines the solution of the partition problem.

+

The following dynamic programming approach shows that the considered problem $O2 \mid d_i = d \mid Y_w$ is binary NP-hard. The algorithm calculates the parts of the jobs which are processed before the due date d such that the total weighted late work Y_w is minimized. Actually, to simplify the approach we maximize the total weighted early work in the system, which is equivalent to the criterion under consideration.

We denote with $f_k(A, B)$ the maximal weighted early work for the jobs J_k, ..., J_n provided that the total processing time of the totally early tasks of the jobs of the set $\{J_k, ..., J_n\}$ is not larger than $(d - A)$ on M_1 and $(d - B)$ on M_2, respectively. First, we calculate initial conditions, $f_{\tilde{n}+1}(A, B)$, for a set J^d of jobs with partially early tasks, where $\tilde{n} = |J \backslash J^d|$. This set may contain two jobs, one or no job. For the remaining jobs $J_k \in J \backslash J^d$, the recurrence relations $f_k(A, B)$ are determined.

For any two-job set $J^d = \{J_r, J_s\}$, we calculate the initial conditions twice. Assuming that job J_a denotes a job partially early on M_1, while J_b denotes a job partially early on M_2, we determine the initial weighted early work for $J_a=J_r$ and $J_b=J_s$ and then for $J_a=J_s$ and $J_b=J_r$. For a given pair of jobs J_a, J_b the following four cases are possible:

- the second tasks of both jobs, J_a on M_2 and J_b on M_1, are early (Term T1, Figure 14.1),
- only job J_b has its second task on M_1 early (Term T2, Figure 14.2),
- only job J_a has its second task on M_2 early (Term T3, Figure 14.3),
- both jobs have the second tasks late (Term T4, Figure 14.4).

In the cases mentioned above, the initial weighted early work is determined by terms T1 - T4:

T1: $w_b\, p_{b1} + w_a \min\{p_{a1}\text{-}1,\ d\text{-}(A+p_{b1}),\ d\text{-}(B+p_{a2})\} + w_a\, p_{2a} + w_b \min\{p_{b2}\text{-}1,\ d\text{-}(B+p_{a2}),\ d\text{-}(A+p_{b1})\}$,

T2: $w_b\, p_{b1} + w_a \min\{p_{a1}\text{-}1,\ d\text{-}(A+p_{b1})\} + w_b \min\{p_{b2}\text{-}1,\ d\text{-}B,\ d\text{-}(A+p_{b1})\}$,

T3: $w_a \min\{p_{a1}\text{-}1,\ d\text{-}A,\ d\text{-}(B+p_{a2})\} + w_a\, p_{2a} + w_b \min\{p_{b2}\text{-}1,\ d\text{-}(B+p_{a2})\}$,

T4: $w_a \min\{p_{a1}\text{-}1,\ d\text{-}A\} + w_b \min\{p_{b2}\text{-}1,\ d\text{-}B\}\}$.

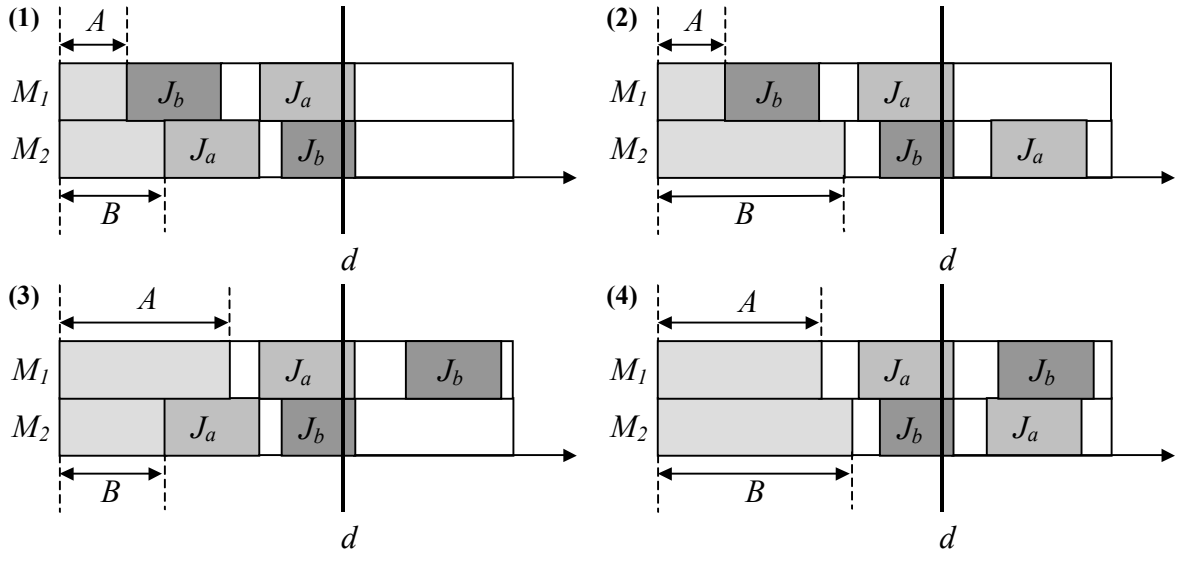

Figure 14. Initial conditions for set $J^d = \{J_a, J_b\}$

Determining the initial weighted early work for set J^d, we choose the best way of executing jobs J_a and J_b. Depending on the values of parameters A, B some schedules are not possible. If free gaps on both machines are sufficiently big, then we check the solutions T1, T2, T3 (Term 1). Solution T4 is not considered because it is dominated by the remaining ones. If the gap on M_1 is too small to schedule job J_b early, then only two solutions T3 and T4 are possible (Term 2). Similarly, if the gap on M_2 is too small to schedule job J_a early, then only two solutions are possible T2 and T4 (Term 3). In both cases, the choice depends on weights w_a, w_b. Finally, for big values A and B, we cannot schedule second tasks of J_a and J_b early and only solution T4 is possible (Term 4). If A and B exceed d, then the solution is infeasible because we cannot schedule J_a and J_b early on any machine (Term 5). Similarly, we reject those pairs of jobs J_a, J_b for which $p_{a1}=1$ or $p_{b2}=1$.

if $A < d$ and $B < d$ and $p_{a1} > 1$ and $p_{b2} > 1$, then

if $A + p_{b1} < d$ and $B + p_{a2} < d$, then $f_{\tilde{n}+1}(A, B) = \max\{\text{T1, T2, T3}\}$ (1)

if $A + p_{b1} \geq d$ and $B + p_{a2} < d$, then $f_{\tilde{n}+1}(A, B) = \max\{\text{T3, T4}\}$ (2)

if $A + p_{b1} < d$ and $B + p_{a2} \geq d$, then $f_{\tilde{n}+1}(A, B) = \max\{\text{T2, T4}\}$ (3)

if $A + p_{b1} \geq d$ and $B + p_{a2} \geq d$, then $f_{\tilde{n}+1}(A, B) =$ T4 (4)

else $f_{\tilde{n}+1}(A, B) = -\infty$ (5)

If we assume that there is only one job with a partially early task, i.e. $J^d=\{J_x\}$, then the initial conditions are formulated in a similar way. In this case, we have again to consider four subcases as shown in Figure 15. The corresponding weighted early work is given by the following formulas:

T5: $w_x p_{x2} + w_x \min\{p_{x1}-1, d-A, d-(B+p_{x2})\}$ (Figure 15.1),

T6: $w_x \min\{p_{x1}-1, d-A\}$ (Figure 15.2),

T7: $w_x p_{x1} + w_x \min\{p_{x2}-1, d-B, d-(A+p_{x1})\}$ (Figure 15.3),

T8: $w_x \min\{p_{x2}-1, d-B\}$ (Figure 15.4).

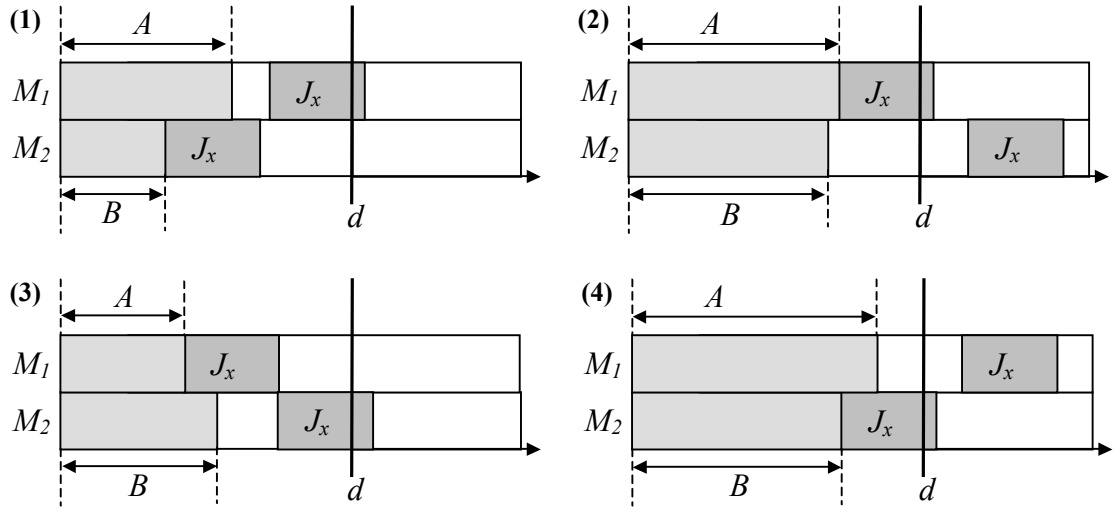

Figure 15. Initial conditions for set $J^d = \{J_x\}$

Determining the initial conditions for jobs J_x with unary tasks we have to exclude from the analysis the cases when those tasks would be partially early. To obtain this goal, it is enough to set T5 = T6 =- ∞ if $p_{x1} = 1$ and to fix T7 = T8 =- ∞ if $p_{x2} = 1$.

To determine the initial weighted early work for a particular job J_x, we have to choose for given values of A and B the best way of scheduling this job among possible solutions:

if $((A \le d \text{ and } B < d)$ or $(A < d \text{ and } B \le d))$ and $(p_{x1} > 1$ or $p_{x2} > 1)$, then

if $A + p_{x1} \le d$ and $B + p_{x2} \le d$, then $f_{\tilde{n}+1}(A, B) = \max\{T5, T7\}$ (6)

if $A + p_{x1} > d$ and $B + p_{x2} \le d$, then $f_{\tilde{n}+1}(A, B) = \max\{T5, T6\}$ (7)

if $A + p_{x1} \le d$ and $B + p_{x2} > d$, then $f_{\tilde{n}+1}(A, B) = \max\{T7, T8\}$ (8)

if $A + p_{x1} > d$ and $B + p_{x2} > d$, then $f_{\tilde{n}+1}(A, B) = \max\{T6, T8\}$ (9)

else $f_{\tilde{n}+1}(A, B) = - \infty$ (10)

If both tasks of J_x can be scheduled early (Term 6), then we choose the better schedule when J_x is totally early on M_2 (T5) or on M_1 (T7). If J_x cannot be scheduled totally early on M_1 (Term 7), then, depending on A, B values, either processing J_x early only on M_1 (T6) or on both machines (T5) is more profitable. Similarly, if J_x cannot be scheduled totally early on M_2 (Term 8), then we select the better solution between two possible ones: when J_x is processed early on both machines (T7) or only on M_2 (T8). For sufficiently big A, B values (Term 9), we can start processing of only one task of J_x before d – either on M_1 (T6) or on M_2 (T8).

If we assume that there is no partially late task in the system, i.e. $J^d = \varnothing$, then the initial conditions take a simple form:

for $A \le d$ and $B \le d$, set $f_{\tilde{n}+1}(A, B) = 0$ (11)

We calculate the initial conditions presented above in $O(d^2)$ time, for all $O(n^2)$ ordered pairs of jobs J_a, J_b, then for all $O(n)$ single jobs J_x and finally for empty set J^d.

For a particular set J^d, we renumber the remaining jobs as $J_1, \ldots, J_{\tilde{n}}$ ($\tilde{n} = |J \setminus J^d|$) and determine the maximal weighted early work subject to the set J^d by calculating the recurrence relations $f_k(A, B)$ for $k = \tilde{n}, \ldots, 1$. The recurrence function can take only four values depending on the way J_k is scheduled:

T9: $w_k (p_{k1} + p_{k2}) + f_{k+1}(A + p_{k1}, B + p_{k2})$, if J_k is totally early,

T10: $w_k p_{k1} + f_{k+1}(A + p_{k1}, B)$, if J_k is early only on M_1,

T11: $w_k p_{k2} + f_{k+1}(A, B + p_{k2})$, if J_k is early only on M_2,

T12: $f_{k+1}(A, B)$, if J_k is totally late.

For given values A and B we select the best solution among the possible ones, i.e.:

if $A + p_{k1} \le d$ and $B + p_{k2} \le d$ and $p_{k1} + p_{k2} \le d$, then $f_k(A, B) = \max\{T9, T10, T11, T12\}$ (12)

if $A + p_{k1} \le d$ and $B + p_{k2} \le d$ and $p_{k1} + p_{k2} > d$, then $f_k(A, B) = \max\{\text{T10, T11, T12}\}$ (13)

if $A + p_{k1} \le d$ and $B + p_{k2} > d$, then $f_k(A, B) = \max\{\text{T10, T12}\}$ (14)

if $A + p_{k1} > d$ and $B + p_{k2} \le d$, then $f_k(A, B) = \max\{\text{T11, T12}\}$ (15)

if $A + p_{k1} > d$ and $B + p_{k2} > d$, then $f_k(A, B) = \text{T12}$ (16)

If J_k can be executed early, then all cases T9-T12 are possible (Term 12). If job J_k is too long to be scheduled totally early, then it has to be late on at least one machine (Term 13). For big B values, J_k cannot be early on M_2 (Term 14), while for big A values, it cannot be early on M_1 (Term 15). If A and B are sufficiently big, then J_k has to be executed late (Term 16). The calculation of the recurrence relations presented above takes $O(nd^2)$ time for all jobs J_k.

The best function value for a fixed set J^d is given by $\max\{f_1(A,B): 0 \le A, B \le d\}$, where different A, B values model different idle times which may appear either on M_1 or on M_2. To find an optimal solution of the problem, we have to check all sets J^d and to calculate the recurrence relations for the remaining jobs $J \setminus J^d$. Thus, the overall complexity of the method is $O(n^3d^2)$. After determining the best set J^d and having the optimal objective function value calculated, the corresponding optimal schedule of the totally early tasks of the jobs can be constructed in $O(n)$ time by the algorithm by Ganzalez and Sahni for problem $O2 \mid\mid C_{max}$ [14]. It is worth mentioning that it is necessary to schedule the task of the job that is partially late on the machine with the bigger machine load on the other machine in such a way that it is completed until the smaller machine load with respect to the totally early tasks. The late tasks of the jobs are sequenced arbitrarily after the common due date. The schedule construction does not change the pseudo-polynomial time complexity of the whole approach.

It is worth noting that also an alternative dynamic programming formulation is possible requiring only one backward run through accordingly defined recurrence relations. But in this case, evaluating the recurrence relation for any values A, B, and job J_k takes $O(p_{max})$ time, where p_{max} is the maximum task processing time. This variant of a dynamic programming method would result in an $O(nd^2\min\{p_{max}, d\})$ approach.

The existence of a pseudo-polynomial method for problem $O2 \mid d_i = d \mid Y_w$ proves its binary *NP*-hardness and allows one to determine completely the complexity status of the case analyzed.

4. Conclusions

The presented paper returns to the interesting field of the scheduling theory concerning the late work performance measure and extends the state of the art with several new results.

We have introduced the late work criteria into the classical graph of objective functions interrelations comparing them with the maximum lateness. The relation showed may be helpful in

the complexity analysis of open problems with the late work performance measures, because research may be directed by the complexity status of the same cases with the maximum lateness objective function. Then, we have proven the equivalence between the late work criterion and the number of late tasks for scheduling problems with a single machine, identical or uniform machines and non-preemptive, unit-processing time activities and integer parameter values. This outcome allowed to transfer a couple of results reported in the literature to the late work field.

First of all, we have considered the late work criteria in the shop environment, especially in the open-shop one. We have proposed a polynomial time algorithm for problem $O \mid pmtn, r_i \mid Y_w$ based on the linear programming approach and Gonzalez and Sahni's method for problem $O \mid pmtn \mid C_{max}$. Moreover, the relation between the maximum lateness and late work criteria allowed us to state that problem $O \mid pmtn, r_i \mid Y_w$ is a maximal polynomially solvable case. Furthermore, the polynomial-time algorithm for problem $O2 \mid d_i = d \mid Y$ has been proposed based on a modification of Gonzalez and Sahni's approach to problem $O2 \mid \mid C_{max}$. Then, the *NP*-hardness proof for problem $O2 \mid d_i = d \mid Y_w$ has been provided together with a pseudo-polynomial dynamic programming approach.

Because the research on the scheduling problems with the late work criteria has not been intensively performed, there are many open cases in this field concerning different machine environments. In our further work, we are mostly concentrating on two-machine flow and job shop cases with a common due date with and without weights. Additionally, taking into account the proven relation between the maximum lateness and the late work criteria, the especially challenging problems are those which are polynomially solved for the L_{max} objective function. For such cases, the existence of the polynomial-time exact methods is still an open question.

Acknowledgement

We would like to thank the anonymous referees for their constructive comments which allowed us to improve the paper.

Małgorzata Sterna has been awarded the Annual Stipend for Young Scientists of the Foundation for Polish Science.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Networks Flows: Theory, Algorithms and Applications (Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1993).

[2] P. Baptiste, Polynomial time algorithms for minimizing the weighted number of late jobs on a single machine when processing times are equal, Journal of Scheduling 2 (1999) 245-252.

[3] R. Bettati, D. Gillies, C.C. Han, K.J. Lin, C.L. Liu, J.W.S. Liu, W.K. Shih, Recent results in real-time scheduling, in: A.M. van Tilborg, G.M. Koob, eds., Foundations of Real-time Computing: Scheduling and Resource Management (Kluwer Academic Publishers, Boston, 1991) 129-156.

[4] J. Błażewicz, Scheduling preemptible tasks on parallel processors with information loss, Recherche Technique et Science Informatiques 3 (1984) 415-420.

[5] J. Błażewicz, K.H. Ecker, E. Pesch, G. Schmidt, J. Węglarz, Scheduling Computer and Manufacturing Processes (Springer, Heidelberg, 2nd edition, 2001).

[6] J. Błażewicz, G. Finke, Minimizing mean weighted execution time loss on identical and uniform processors, Information Processing Letters 24 (1987) 259-263.

[7] J. Błażewicz, E. Pesch, M. Sterna, F. Werner, Total late work criteria for shop scheduling problems, in: K. Inderfurth, G. Schwödiauer, W. Domschke, F. Juhnke, P. Kleinschmidt, G. Wäscher, eds., Operations Research Proceedings 1999 (Springer, Heidelberg, 2000) 354-359.

[8] P. Brucker, Scheduling Algorithms (Springer, Heidelberg, 2nd edition, 1998).

[9] M.W. Carter, C.C. Price, Operations Research. A Practical Introduction (CRC Press, Boca Raton, 2001).

[10] Y. Cho, S. Sahni, Preemptive scheduling of independent jobs with release and due times on open, flow and job shops, Operations Research 29 (1981) 511-522.

[11] J.Y. Chung, W.K. Shih, J.W.S. Liu, D.W. Gillies, Scheduling imprecise computations to minimize total error, Microprocessing and Microprogramming 27 (1989) 767-774.

[12] M.I. Dessouky, B.J. Lageweg, J.K. Lenstra, S.L. Van De Velde, Scheduling identical jobs on uniform parallel machines, Statistica Neerlandica 44 (1990) 115-123.

[13] M.R. Garey, D.S. Johnson, Computers and Intractability (W.H. Freeman and Co., San Francisco, 1979).

[14] T. Gonzalez, S. Sahni, Open shop scheduling to minimize finish time, Journal of the ACM 23 (1976) 665-679.

[15] K. Ho, J.Y.T. Leung, W.D. Wei, Minimizing constrained maximum weighted error for doubly weighted tasks , Technical Report UNL-CSE-92-018, University of Nebraska, 1992.

[16] K. Ho, J.Y.T. Leung, W.D. Wei, Minimizing maximum weighted error for imprecise computation tasks, Journal of Algorithms 16 (1994) 431-452.

[17] J. Józefowska, B. Jurisch, W. Kubiak, Scheduling shops to minimize the weighted number of late jobs, Operation Research Letters 16 - 5 (1994) 277-283.

[18] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Minimizing maximum lateness in a two-machine open shop, Mathematics of Operations Research 6 (1981) 153-158; Erratum, Mathematics of Operations Research 7 (1982) 635.

[19] J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity results for scheduling chains on a single machine, European Journal of Operational Research 4 (1980) 270-275.

[20] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics 1 (1977) 343-362.

[21] M. Pinedo, X. Chao, Operation Scheduling with Applications in Manufacturing and Services, (Irwin/McGraw-Hill, Boston, 1999).

[22] C.N. Potts, L.N. Van Wassenhove, Single machine scheduling to minimize total late work, Operations Research 40-3 (1991) 586-595.

[23] C.N. Potts, L.N. Van Wassenhove, Approximation algorithms for scheduling a single machine to minimize total late work, Operations Research Letters 11 (1992) 261-266.

[24] W.K. Shih, J.W.S. Liu, J.Y. Chung, Algorithms for scheduling imprecise computations with timing constraints, SIAM Journal on Computing 20 (1991) 537-552.

[25] J.A. Stankievic, M. Spuri, K. Ramamritham, G.C. Buttazzo, Deadline Scheduling for Real-Time Systems (Kluwer Academic Publishers, Boston, 1998).

[26] M. Sterna, Problems and Algorithms in Non-Classical Shop Scheduling (Scientific Publishers OWN, Polish Academy of Sciences, Poznań, 2000).

[27] *http://www.mathematik.uni-osnabrueck.de/research/OR/class/*