RAIRO Operations Research
RAIRO Oper. Res. **35** (2001) 71-83

SINGLE MACHINE PREEMPTIVE SCHEDULING TO MINIMIZE THE WEIGHTED NUMBER OF LATE JOBS WITH DEADLINES AND NESTED RELEASE/DUE DATE INTERVALS

Valery S. Gordon[1], F. Werner[2] and O.A. Yanushkevich[1]

Communicated by J. Carlier

Abstract. This paper is devoted to the following version of the single machine preemptive scheduling problem of minimizing the weighted number of late jobs. A processing time, a release date, a due date and a weight of each job are given. Certain jobs are specified to be completed in time, *i.e.*, their due dates are assigned to be deadlines, while the other jobs are allowed to be completed after their due dates. The release/due date intervals are nested, *i.e.*, no two of them overlap (either they have at most one common point or one covers the other). Necessary and sufficient conditions for the completion of all jobs in time are considered, and an $O(n \log n)$ algorithm (where n is the number of jobs) is proposed for solving the problem of minimizing the weighted number of late jobs in case of oppositely ordered processing times and weights.

Keywords: Single machine scheduling, release and due dates, deadlines, number of late jobs.

Mathematics Subject Classification. 90B36.

Received October, 1999. Accepted November, 2000.

[1] Institute of Engineering Cybernetics, National Academy of Sciences of Belarus, Minsk, Belarus; e-mail: gordon@newman.bas-net.by and yanush@newman.bas-net.by

[2] Otto-von-Guericke University of Magdeburg, Germany;
e-mail: frank.werner@mathematik.uni-magdeburg.de

1. Introduction

In this paper we consider the n job and single machine preemptive scheduling problem of minimizing the weighted number of late jobs with nested release/due date intervals assuming that certain specified jobs have to be completed in time.

In project management and in service industries, it is a widespread situation, when failure to meet due dates may result in a significant loss. Therefore, it is expedient to consider a problem of minimizing the weighted number of late jobs when at least some of the due dates are in fact deadlines. Sidney [16] was the first who considered the problem of minimizing the number of late jobs in the situation when certain specified jobs have to be in time. Unlike [16], we consider this situation for the case of given release dates and the weighted number of late jobs subject to nested release/due date intervals. The problem of minimizing the weighted number of late jobs with given deadlines for some jobs and similarly or oppositely ordered release and due dates was considered in [1]. Nested release/due date intervals for the problem of minimizing the weighted number of late jobs without deadlines were first considered by Lawler [9].

The paper is organized as follows. In Section 2, we introduce our notation and definitions and briefly review the known results related to the problem under consideration. In Section 3, we consider necessary and sufficient conditions for the existence of a single machine preemptive schedule in which all jobs are completed by their due dates. These conditions may be verified in $O(n)$ time for nested release/due date intervals provided that jobs are indexed in non-decreasing order of their due dates. In Section 4, we propose an $O(n \log n)$ algorithm for the problem of minimizing the weighted number of late jobs with deadlines and oppositely ordered processing times and job weights.

2. Preliminaries

The single machine problem of minimizing the weighted number of late jobs is one of the classical scheduling problems. We consider this problem under the assumption that release dates are given, preemption is permitted, release/due date intervals are nested and certain specified jobs have to be completed in time.

The problem under consideration can be posed as follows. There is a set $N = \{1, 2, \ldots, n\}$ of jobs which are to be processed on a single machine. The machine can execute at most one job at a time. Each job $i \in N$ is available for processing at time (*release date*) $r_i \geq 0$ and requires a *processing time* $p_i > 0$. Preemption ("job splitting") is allowed: the processing of any job may be interrupted and resumed at a later time. If C_i is the *completion time* of job i in a schedule, then job i is *late* (or *tardy*) when $C_i > d_i$, where d_i is the given *due date*, otherwise it is *in time*. Each job i of a given subset $Q \subseteq N$ must be completed not later than its due date, *i.e.*, the due dates for these jobs are considered as *deadlines*. If $C_i > d_i$ for $i \in N \backslash Q$, then there is a unit penalty $U_i = 1$, otherwise $U_i = 0$. There is also a nonnegative penalty (*weight*) w_i for the job being tardy. The objective is

to find a schedule which minimizes $\sum_{i\in N\setminus Q} w_iU_i$ provided that the deadlines are not violated for all $i \in Q$ (it is assumed that there exists a schedule with $C_i \le d_i$ for all $i \in Q$).

Release/due date intervals $[r_i, d_i]$, $i = 1, \dots, n$, are *nested*, that is each pair $[r_i, d_i]$, $[r_j, d_j]$ is such that either the two intervals are disjoint (except possibly at end points) or one interval is contained within the other (see Fig. 1).

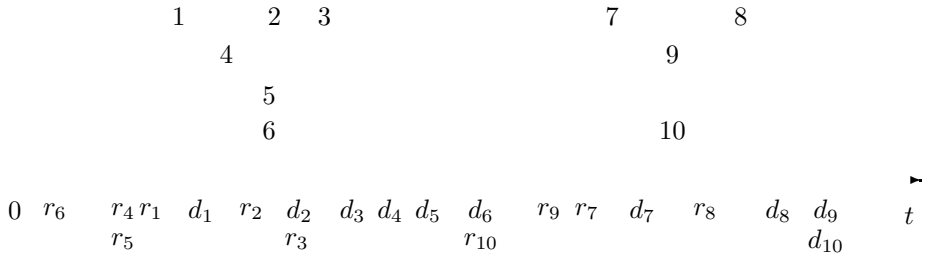

FIGURE 1.

If ties between equal $[r_i, d_i]$ intervals are broken arbitrarily, the nested release/due date intervals are partially ordered by the inclusion relation. The transitive reduction of this partial order is represented by a forest of in-trees, where each vertex of the tree represents a release/due date interval. See Figure 2 for the forest of in-trees which represents the release/due date intervals of Figure 1. It is obvious that the problem can be decomposed into problems for in-trees, each of which can be considered separately.

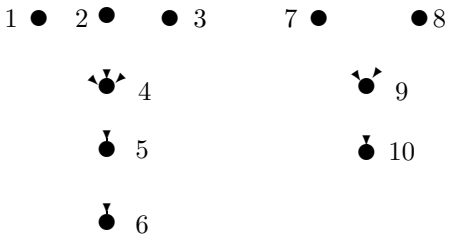

FIGURE 2.

The problem may be denoted as $1|pmtn; nst; \overline{d_i}, i \in Q|\sum w_jU_j$ in the notation of [3, 10], where *nst* denotes nested release/due date intervals and $\overline{d_i}, i \in Q$, indicates a given subset of jobs that have to meet their deadlines.

The first result for the single machine scheduling problem under consideration (subject to $Q = \emptyset$) is a well-known $O(n \log n)$ algorithm of Moore–Hodgson [15] which solves the problem $1||\sum U_j$, the special case in which all release dates are zero and all job weights are equal. The problem of minimizing the weighted number of late jobs (for the jobs simultaneously available for processing) is known to be

NP-hard in the ordinary sense (Karp [4]) and it is solvable in pseudopolynomial time (Lawler and Moore [11]). The problem of minimizing the number of late jobs with given release dates is strongly NP-hard in consequence of strong NP-hardness of the problem of minimizing the maximum lateness with given release dates (Lenstra *et al.* [13]). The preemptive case of the problem with given release dates is solvable in polynomial time for equal weights (Lawler [7]) and in pseudopolynomial time for unequal weights (Lawler [8]). Monma [14] gave an $O(n)$ algorithm for the $1|p_j = 1|\sum U_j$ problem; Lenstra and Rinnooy Kan [12] showed that the $1|chain, p_j = 1|\sum U_j$ problem is strongly NP-hard, where *chain* denotes chain-like precedence constraints.

Some polynomially solvable cases for the problem of minimizing the number of late jobs with given release dates or minimizing the weighted number of late jobs were considered by Lawler [6,9], Kise *et al.* [5], Gordon and Tanaev [2,17], Tanaev *et al.* [18].

The problem with $Q \neq \emptyset$ was considered by Sidney [16] for the case $r_i = 0$, $w_i = 1, i = 1, \ldots, n$, and by Gordon *et al.* [1] for the case of similarly or oppositely ordered release and due dates.

In Section 3, we will consider the following special cases of nested release/due date intervals. Let jobs be numbered in non-decreasing order of their due dates.

The release/due date intervals are *embedded* if $r_i \geq r_{i+1}, i = 1, \ldots, n-1$. The inclusion-relation graph of these intervals is a chain.

The release/due date intervals are *gathered* if $N = N_1 \cup N_2 \cup \ldots \cup N_{m+1}$, $N_i \cap N_j = \emptyset, 1 \leq i \neq j \leq m$, and the subsets N_i are such that

(a) for each of the sets $N_i, i = 1, \ldots, m+1$, the release/due dates intervals are embedded,
(b) each pair of the intervals for the jobs from N_i and N_j, where $1 \leq i \neq j \leq m$, is disjoint except possibly at end points, and
(c) all the intervals for the jobs from $N_i, i = 1, \ldots, m$, are contained in each of the intervals for the jobs from N_{m+1} (see Fig. 3 for the case of $m = 2$).

Note that for any $i \in N_v, j \in N_{v'}, k \in N_{m+1}, 1 \leq v < v' \leq m$, we have $i < j < k$. Let $\underline{n}_i$ and $\overline{n}_i$ be the jobs with the least and greatest numbers in $N_i, i = 1, \ldots, m+1$.

As it has been mentioned before, the nested release/due date intervals are partially ordered by the inclusion relation and can be represented by a forest of in-trees. For each tree, starting from the leaves we combine into one vertex a maximal set of in-degree 1 vertices of a chain (except the starting vertex which may have in-degree 0 or greater than 1). So, we combine each of such chains of maximal length into one vertex and transform each in-tree of the forest into an in-tree in which each vertex corresponds to embedded intervals (see Fig. 4, where vertices 4, 5 and 6 of Fig. 2 are combined into vertex A, and vertices 9, 10 are combined into vertex B). We shall use this representation of the forest later on in Theorem 4.

In this representation, the inclusion-relation graph for gathered intervals is a tree of height one.

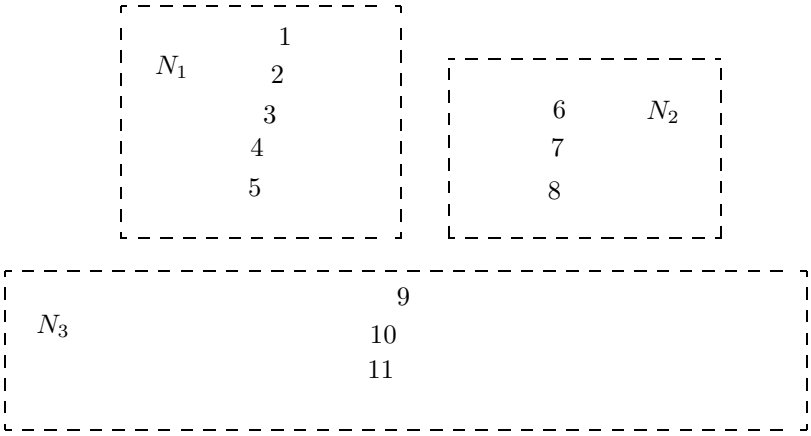

FIGURE 3.

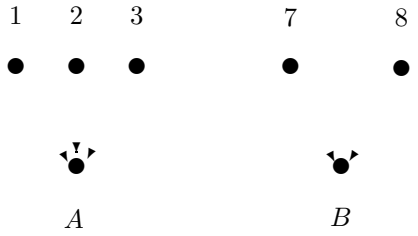

FIGURE 4.

On the other hand, each in-tree of the decomposition forest may be also represented as a tree of height one (as in Fig. 5), where $N_1, N_2, \ldots, N_m$ are the sets of jobs with disjoint nested intervals and N_{m+1} is a set of jobs with embedded intervals. As before, let $\underline{n}_i$ and $\overline{n}_i$ be the jobs with the least and greatest numbers in $N_i, i = 1, \ldots, m+1$. For example, the first in-tree of Figure 2 can be represented as in Figure 5 with $N_1 = \{1, 2, 3, 4\}$ and $N_2 = \{5, 6\}$ (here $m = 1$), or $N_1 = \{1\}, N_2 = \{2, 3\}, N_3 = \{4, 5, 6\}$ (here $m = 2$), or $N_1 = \{1\}, N_2 = \{2\}$, $N_3 = \{3\}, N_4 = \{4, 5, 6\}$ (here $m = 3$).

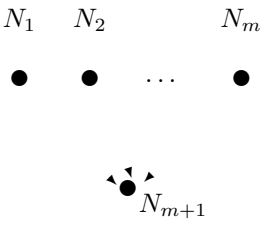

FIGURE 5.

This representation will be used later on in Theorem 3.

3. Due date feasible schedules

In this section we consider necessary and sufficient conditions for the existence of a schedule where all due dates are maintained (that is, $C_i \leq d_i$ for each job $i \in N$) and we show that these conditions may be verified in $O(n)$ time for nested release/due date intervals.

We call a schedule *due date feasible* if all jobs in it observe their due dates (*i.e.*, $C_i \leq d_i$ for each job $i \in N$).

In what follows, let the jobs be numbered in non-decreasing order of their due dates. The following theorem establishes necessary and sufficient conditions for the existence of a due date feasible single machine preemptive schedule of jobs with possibly different release dates.

Theorem 1. **[18]** *Let N_k^l be the set of all jobs $i \in N$ with $r_i \geq r_k$ and $d_i \leq d_k, 1 \leq k \leq l \leq n$. Then a due date feasible schedule exists if and only if $k \leq l$ implies*

$$\sum_{i \in N_k^l} p_i \leq d_l - r_k. \tag{1}$$

From this theorem we derive the following corollary.

Corollary 1. *For embedded release/due date intervals, a due date feasible schedule exists if and only if*

$$\sum_{j=1}^{i} p_j \leq d_i - r_i, \quad i = 1, \ldots, n. \tag{2}$$

Proof. For embedded release/due date intervals we have $r_i \geq r_{i+1}$, $i = 1, \ldots, n-1$. In this case, we obtain $N_k^l = \{1, \ldots, k\}$ for any k, l such that $1 \leq k \leq l \leq n$, and (1) may be rewritten as:

$$\sum_{j=1}^{k} p_j \leq d_l - r_k \tag{3}$$

for all $1 \leq k \leq l \leq n$.

The latter inequality for $k = l$ dominates this inequality for $k < l$: if (3) holds for $l = k$, it will be valid for all $l > k$. Deleting all inequalities except the dominating, we obtain (2). Therefore, if (2) is valid, a due date feasible schedule exists.

On the other hand, if a due date feasible schedule exists, then (3) is valid and for $k = l$ we obtain (2) from (3). □

We shall prove now the following statement:

Theorem 2. *Let release/due date intervals of the set N be gathered. A due date feasible schedule for N exists if and only if*

$$\sum_{j=\underline{n}_v}^{i} p_j \le d_i - r_i \tag{4}$$

holds for all i, $\underline{n}_v \le i \le \overline{n}_v, v = 1, 2, \dots, m$, and

$$\sum_{j \in N_1 \cup \dots \cup N_m} p_j + \sum_{j=\underline{n}_{m+1}}^{i} p_j \le d_i - r_i \tag{5}$$

holds for all i, $\underline{n}_{m+1} \le i \le \overline{n}_{m+1}$.

Proof. From Theorem 1, a due date feasible schedule exists if and only if (1) is valid for all $1 \le k \le l \le n$.

Let us show that (4) and (5) are valid if a due date feasible schedule for N exists.

Since (1) holds for all $1 \le k \le l \le n$, and $N_k^l = \{\underline{n}_v, \dots, k\}$ for $\underline{n}_v \le k \le l \le \overline{n}_v, v = 1, \dots, m+1$, we have from (1):

$$\sum_{j=\underline{n}_v}^{k} p_j \le d_l - r_k.$$

The latter inequality for $l = k$ leads to (4).

Since $N_k^l = N_1 \cup N_2 \cup \dots \cup N_m \cup \{\underline{n}_{m+1}, \dots, i\}$ if $k = l = i \in N_{m+1}$, we have from (1):

$$\sum_{j \in N_1 \cup \dots \cup N_m} p_j + \sum_{j=\underline{n}_{m+1}}^{i} p_j \le d_i - r_i,$$

for $\underline{n}_{m+1} \le i \le \overline{n}_{m+1}$, that is, (5) is valid.

Let us show that a due date feasible schedule for N exists if (4) and (5) are valid.

A due date feasible schedule exists if inequalities (1) are valid for all the jobs. It is sufficient to show that inequalities (1) are valid for

a) $k, l \in N_v, k \le l, 1 \le v \le m$;
b) $k \in N_v, l \in N_{v'}, 1 \le v < v' \le m$;
c) $k \in N_v, l \in N_{m+1}$, $1 \le v \le m$, and
d) $k, l \in N_{m+1}, k \le l$.

For case (a), we have embedded intervals for the jobs of the set $N_v, 1 \le v \le m$, and from Corollary 1, a due date feasible schedule for N_v exists if (2) is valid,

which may be rewritten as (4):

$$\sum_{j=\underline{n}_v}^{i} p_j \le d_i - r_i, \quad i = \underline{n}_v, \dots, \overline{n}_v.$$

For case (b), the set N_k^l is equal to $\{\underline{n}_v, \dots, k\} \cup N_{v+1} \cup \dots \cup N_{v'-1} \cup \{\underline{n}_{v'}, \dots, l\}$, and the inequality (1) may be rewritten as

$$\sum_{j \in \{\underline{n}_v, \dots, k\} \cup N_{v+1} \cup \dots \cup N_{v'-1} \cup \{\underline{n}_{v'}, \dots, l\}} p_j \le d_l - r_k, \tag{6}$$

where $k \in N_v, l \in N_{v'}$. To prove (6), consider the inequalities which follow from (4):

$$\sum_{j=\underline{n}_v}^{k} p_j \le d_k - r_k, \quad k \in N_v,$$
$$\sum_{j=\underline{n}_w}^{\overline{n}_w} p_j \le d_{\overline{n}_w} - r_{\overline{n}_w}, \quad v+1 \le w \le v'-1,$$
$$\sum_{j=\underline{n}_{v'}}^{l} p_j \le d_l - r_l, \quad l \in N_{v'}.$$

We obtain (6) by adding the above inequalities and taking into account that $d_k \le r_{\overline{n}_w}, d_{\overline{n}_w} \le r_l$ for $k \in N_v, v+1 \le w \le v'-1, l \in N_{v'}$.

For case (c), the set N_k^l is equal to $\{\underline{n}_v, \dots, k\} \cup N_{v+1} \cup \dots \cup N_m$, and inequality (1) may be rewritten as

$$\sum_{j \in \{\underline{n}_v, \dots, k\} \cup N_{v+1} \cup \dots \cup N_m} p_j \le d_l - r_k,$$

where $k \in N_v, l \in N_{m+1}$. Since (6) with $l = \overline{n}_m$ dominates the last inequality, we arrive to case (b).

For case (d), the set N_k^l is equal to $N_1 \cup \dots \cup N_m \cup \{\underline{n}_{m+1}, \dots, k\}$, and inequality (1) may be rewritten as

$$\sum_{j \in N_1 \cup \dots \cup N_m} p_j + \sum_{j=\underline{n}_{m+1}}^{k} p_j \le d_l - r_k, \tag{7}$$

where $k, l \in N_{m+1}$. To prove (7), it is sufficient to notice that inequality (5) dominates (7) since $d_k \le d_l$. □

From this theorem we obtain the following corollary:

Corollary 2. *The existence of a due date feasible schedule of jobs with gathered release/due date intervals can be verified in $O(n)$ time provided that jobs are numbered in non-decreasing order of their due dates.*

Proof. To check whether there exists a due date feasible schedule for the set $N = N_1 \cup N_2 \cup \ldots \cup N_{m+1}$ with gathered release/due date intervals, it is sufficient to verify (2) (or, which is the same (4)) separately for $N_1, N_2, \ldots, N_m$ and (5) for $\underline{n}_{m+1} \le i \le \overline{n}_{m+1}$. It is obvious that this may be done in $O(n)$ time. □

Theorem 3. *For the nested release/due date intervals which are represented by Figure 5, a due date feasible schedule exists if and only if due date feasible schedules exist for the jobs of the sets $N_i, i = 1, \ldots, m$, and inequality (5) holds for $\underline{n}_{m+1} \le i \le \overline{n}_{m+1}$.*

Proof. If a due date feasible schedule exists for all jobs, it exists for the jobs of the sets $N_i, i = 1, \ldots, m$, and (1) holds for all $1 \le k \le l \le n$.

For $k, l \in N_{m+1}, k \le l$, we have $N_k^l = N_1 \cup N_2 \cup \ldots \cup N_m \cup \{\underline{n}_{m+1}, \ldots, k\}$ and from (1) we obtain:

$$\sum_{j \in N_1 \cup \ldots \cup N_m} p_j + \sum_{j=\underline{n}_{m+1}}^{k} p_j \le d_l - r_k$$

for all $k, l \in N_{m+1}$. From the latter inequality we obtain (5) for $k = l = i \in N_{m+1}$.

Let us show that a due date feasible schedule for all jobs exists if it exists for each of the sets $N_i, i = 1, \ldots, m$, and (5) is valid. A due date feasible schedule exists if inequalities (1) are valid for all jobs. It is sufficient to show that these inequalities are valid for

(a) $k \in N_v, l \in N_{m+1}$, $1 \le v \le m$, and
(b) $k, l \in N_{m+1}, k \le l$.

For case (a), the set N_k^l is equal to $N' \cup N_{v+1} \cup \ldots \cup N_m$ where N' is a subset of N_v, and (1) may be rewritten as

$$\sum_{j \in N' \cup N_{v+1} \cup \ldots \cup N_m} p_j \le d_l - r_k, \tag{8}$$

for $k \in N_v, l \in N_{m+1}$.

Since a due date feasible schedule exists for the jobs of the sets $N', N_{v+1}, \ldots, N_m$, inequality (1) holds for $N_k^{l'} = N' \cup N_{v+1} \cup \ldots \cup N_m$, where $k \in N_v$ and $l' = \overline{n}_m$:

$$\sum_{j \in N' \cup N_{v+1} \cup \ldots \cup N_m} p_j \le d_{\overline{n}_m} - r_k.$$

It is obvious that (8) follows from the latter inequality.

For case (b), the proof is the same as for case (d) of Theorem 2. □

Now we may formulate the following statement:

Theorem 4. *A due date feasible schedule for the set of jobs with nested release/due date intervals exists if and only if inequalities (4) and (5) hold for the sets of jobs in the following procedure:*

(a) *represent the set of intervals as a forest of in-trees with the vertices corresponding to embedded intervals (as described in Sect. 2, see Fig. 4);*
(b) *consider each of the in-trees separately, and*
(c) *moving from the leaves of the tree to the root, verify (4) and (5) for each vertex with all its immediate predecessors (considering them as gathered intervals or as the intervals of Fig. 5) and combining these vertices into one nested interval vertex.*

Proof. It is obvious that the problem of the existence of a due date feasible schedule can be decomposed into problems for in-trees, each of which can be considered separately. To prove the theorem, it is sufficient to apply first Theorem 2 and then Theorem 3 for each subtree of height one, moving from the leaves of each tree to the root. □

Corollary 3. *The existence of a due date feasible schedule of jobs with nested release/due date intervals can be verified in $O(n)$ time provided that the jobs are numbered in non-decreasing order of their due dates.*

Proof. Step (c) of Theorem 4 represents the following recursion algorithm. Moving from the leaves to the root (of each tree separately), we apply first Theorem 2 for the gathered intervals of the subtrees of height one, and then (considering the intervals of these subtrees as one nested interval) we apply Theorem 3 to the next subtrees of height one until we come to the root. Since the operations corresponding to the verification of all inequalities (4) and (5) can be done in $O(n)$ time, the existence of a due date feasible schedule can be verified in $O(n)$ time. (Note that we do not need to recalculate the sums of processing times in (5) when we pass to the next subtree. We can use the results of the previous steps.) □

4. Nested release/due date intervals with deadlines

Throughout this section, we consider the problem $1|pmtn; nst; \overline{d_i}, i \in Q| \sum w_j U_j$ with the assumption that processing times and job weights are oppositely ordered which means that there exists an indexing of the jobs such that

$$p_{i_1} \le p_{i_2} \le \ldots \le p_{i_n} \text{ and } w_{i_1} \ge w_{i_2} \ge \ldots \ge w_{i_n}.$$

Note that this order is independent on the numbering of the jobs. As before, the jobs are numbered in non-decreasing order of their due dates. And at last, we assume that there exists a schedule with $C_i \le d_i$ for all $i \in Q$.

A schedule which minimizes $\sum_{i \in N \setminus Q} w_i U_i$ provided that no deadline is violated (*i.e.*, $C_i \le d_i$ for all $i \in Q$) will be called *optimal*.

We shall use the following fundamental observation [9] that applies to the general problem in either its nonpreemptive or preemptive versions, *i.e.*, to $1|r_j|\sum w_jU_j$ or $1|pmtn, r_j|\sum w_jU_j$: if a job is late, it might as well be arbitrary late. Hence, there exists an optimal schedule for $1|pmtn; nst; \overline{d_i}, i \in Q|\sum w_jU_j$ in which all jobs that are in time precede all the jobs that are late. So, the problem is

(a) to find a set R^* of late jobs (where $R^* \cap Q = \emptyset$) with the smallest value $f(R) = \sum_{i\in R} w_i$ among all sets $R, R \cap Q = \emptyset$, for which there exists a due date feasible schedule for the jobs of the set $N\backslash R$;
(b) to construct a due date feasible schedule for the set $N\backslash R^*$ (denote this schedule by $s^d_{N\backslash R^*}$), and
(c) to join to $s^d_{N\backslash R^*}$ an arbitrary permutation of the jobs of the set R^* (denote this permutation by π_{R^*}). Then, an optimal schedule s^* for the set N will be: $s^* = (s^d_{N\backslash R^*}, \pi_{R^*})$.

The following algorithm A provides the construction of an optimal schedule.

Algorithm A:

1. Let $R^* = \emptyset$. Starting from $i = 1$ and increasing i by 1 up to $i = n$, represent the set of release/due date intervals as in (a, b) of the procedure of Theorem 4 and verify inequalities (4) and (5) of this procedure. Note that in this case we move, as in step (c) of the procedure, from the leaves to the root of each tree. If (4) or (5) is not valid for some i, find a job with the largest p_k among the jobs k, $k \le i$, $k \notin Q$. Delete this job from the set N and add it to the set R^*.
2. Apply Lawler's algorithm [9] (which was proposed for the problem $1|pmtn, nst|\sum w_jU_j$) to the jobs of the set $N \setminus R^*$. As a result, we obtain a schedule s' for the jobs of the set $N \setminus R^*$.
3. Construct a schedule $s^* = (s', \pi_{R^*})$, where π_{R^*} is an arbitrary permutation of the jobs of the set R^*.

Theorem 5. *If schedule s^* is constructed by algorithm A, it is optimal.*

Proof. At the first step, if (4) or (5) is not valid for some i, $i \in N \setminus R^*$, a due date feasible schedule for the set of jobs $N \setminus R^*$ does not exist (Th. 4). At least one of the jobs of the set $N \setminus R^*$ must be deleted to obtain a due date feasible schedule. Since we delete each time the job with the largest p_k (and therefore with the smallest weight), we shall obtain as a result the smallest possible sum of weights for the set R^* when all other jobs (including all jobs of the set Q) are scheduled without violating their due dates.

When we apply Lawler's algorithm to the set $N \setminus R^*$ at the second step, we obtain a schedule with the minimal total weight of late jobs for the set $N \setminus R^*$. Since the jobs of the set $N \setminus R^*$ can be scheduled without violating the due dates, the total weight of late jobs in this schedule will be zero, and s' will be a due date feasible schedule for the set $N \setminus R^*$, *i.e.* $s' = s^d_{N\setminus R^*}$.

Therefore, a schedule $s^* = (s', \pi_{R^*})$, where jobs of the sequence π_{R^*} are late, is an optimal one. □

Corollary 4. *The complexity of Algorithm A is* $O(n \log n)$.

Proof. This statement is obvious, since finding a job with the largest p_k requires $O(\log n)$ time, deleting this job does not violate any valid inequality (4) or (5) and does not require to recalculate the sums of processing times (it is sufficient to subtract the processing time of this job). So, due to Corollary 3, the first step of the algorithm can be done in $O(n \log n)$ time. The second step requires the same time since the complexity of Lawler's algorithm [9] is $O(n \log n)$. □

The following statement shows that algorithm A may be used for the situation when the dependence of the weights on the processing times is the same for all jobs.

Remark 1. Algorithm A constructs an optimal schedule when there exists a non-increasing function f such that $w_i = f(p_i)$ for all jobs of the set $N \setminus Q$.

Remark 2. The assumption of the existence of a deadline feasible schedule (a schedule with $C_i \leq d_i$ for all $i \in Q$) can be verified by applying Theorem 4 for the set Q. If a deadline feasible schedule does not exist, problem $1|pmtn; nst; \overline{d_i}, i \in Q| \sum w_j U_j$ does not have a solution.

Remark 3. The Moore–Hodgson algorithm [9] for the problem with nested release/due date intervals and with opposite ordering of processing times and weights can be modified for the case of a given subset of jobs with deadlines. In the modified algorithm, to insert job $i \in Q$ into the schedule we have (instead of deleting from in-time jobs a job with the largest processing time as in [9]) to delete such a subset of jobs with largest processing times that the sum of their processing times is equal to or greater than p_i. But in this case the algorithm will run in $O(n^2)$ time.

This research was partly supported by the International Association for the Promotion of Cooperation with Scientists from the Independent States of the Former Soviet Union, INTAS 96-0820, and the International Science and Technology Centre, project B-104-98. We are grateful to the constructive suggestions of the anonymous referees which substantially improved the paper.

References

[1] V. Gordon, E. Potapneva and F. Werner, Single machine scheduling with deadlines, release and due dates. *Optimization* **42** (1997) 219-244.

[2] V.S. Gordon and V.S. Tanaev, Single machine deterministic scheduling with step functions of penalties, in: *Computers in Engineering.* Minsk (1971) 3-8 (in Russian).

[3] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey. *Ann. Discrete Math.* **5** (1979) 287-326.

[4] R.M. Karp, Reducibility among combinatorial problems, edited by R.E. Miller and J.W. Thatcher, *Complexity of Computer Computations*. Plenum Press, New York (1972) 85-103.
[5] H. Kise, T. Ibaraki and H. Mine, A solvable case of the one-machine scheduling problem with ready and due times. *Oper. Res.* **26** (1978) 121-126.
[6] E.L. Lawler, Sequencing to minimize the weighted number of tardy jobs. *RAIRO Oper. Res.* **10** (1976) 27-33.
[7] E.L. Lawler, *Scheduling a single machine to minimize the number of late jobs*. Preprint. Computer Science Division, University of California, Berkeley (1982).
[8] E.L. Lawler, A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. *Ann. Oper. Res.* **26** (1990) 125-133.
[9] E.L. Lawler, Knapsack-like scheduling problems, the Moore–Hodgson algorithm and the "tower of sets" property. *Math. Comput. Modelling* **20** (1994) 91-106.
[10] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, *Sequencing and scheduling: Algorithms and complexity*, edited by S.C. Graves, A.H.G. Rinnooy Kan and P.H. Zipkin, Logistics of Production and Inventory. North-Holland, *Handbooks Oper. Res. Management Sci.* **4** (1993) 445-522.
[11] E.L. Lawler and J.M. Moore, A functional equation and its application to resource allocation and sequencing problems. *Management Sci.* **16** (1969) 77-84.
[12] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity results for scheduling chains on a single machine. *European J. Oper. Res.* **4** (1982) 270-275.
[13] J.K. Lenstra, A.H.G. Rinnooy Kan and P. Brucker, Complexity of machine scheduling problems. *Ann. Discrete Math.* **1** (1977) 343-362.
[14] C.L. Monma, Linear-time algorithms for scheduling on parallel processors. *Oper. Res.* **30** (1980) 116-124.
[15] J.M. Moore, An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Sci.* **15** (1968) 102-109.
[16] J.B. Sidney, *An extension of Moore's due date algorithm*, edited by S.E. Elmaghraby, Symposium on the Theory of Scheduling and its Applications. Springer, Berlin, *Lecture Notes in Econom. and Math. Systems* **86** (1973) 393-398.
[17] V.S. Tanaev and V.S. Gordon, On scheduling to minimize the weighted number of late jobs. *Vestsi Akad. Navuk Belarus Ser. Fizi.-Mat. Navuk* **6** (1983) 3-9 (in Russian).
[18] V.S. Tanaev, V.S. Gordon and Y.M. Shafransky, *Scheduling Theory. Single-Stage Systems*. Kluwer Academic, Dordrecht (1994).