A Polynomial Algorithm for Special Case of the One-Machine Scheduling Problem with Time-Lags

Helena Ramalhinho-Lourenço[1]

Abstract

The standard one-machine scheduling problem consists in scheduling a set of jobs in one machine which can handle only one job at a time, minimizing the maximum lateness. Each job is available for processing at its release date, requires a known processing time and after finishing the processing, it is delivery after a certain time. There also can exists precedence constraints between pairs of jobs, requiring that the first jobs must be completed before the second job can start. An extension of this problem consists in assigning a time interval between the processing of the jobs associated with the precedence constrains, known by finish-start time-lags. In presence of this constraints, the problem is NP-hard even if preemption is allowed. In this work, we consider a special case of the one-machine preemption scheduling problem with time-lags, where the time-lags have a chain form, and propose a polynomial algorithm to solve it. The algorithm consist in a polynomial number of calls of the preemption version of the Longest Tail Heuristic. One of the applicability of the method is to obtain lower bounds for NP-hard one-machine and job-shop scheduling problems. We present some computational results of this application, followed by some conclusions.

Keywords: one-machine scheduling, polynomial algorithms, lower bounds.
Journal of Economic Literature Classification: C61,C63

[1] Department of Economics and Management, Universitat Pompeu Fabra, R. Trias Fargas 25-27, 08005 Barcelona, Spain (e-mail: ramalhin@upf.es; web page: http://www.econ.upf.es/~ramalhin)

1. Introduction

The One-Machine Scheduling Problem with Time-Lags consists in scheduling a set of jobs in one machine, minimizing the maximum lateness. Each job has a release date, a processing time and a delivery time. Preemption is allowed. There exists a set of generalized precedence constraints between jobs which are defined as follows: for certain pairs of jobs it is required a positive time-lag between the completion time of the first job and the start of the second.

We will designate the one-machine scheduling problem with simple precedence constraints as the standard problem, since it is a well-known problem. This problem is NP-Hard, Lenstra, Rinnooy Kan and Brucker (1977), but if preemption is allowed the problem can be solved in polynomial time, Horn (1974). However, if time-lags are considered, the problem is NP-hard even if preemption is allowed.

The aim of this paper is to present a polynomial algorithm to solve a special case of the one-machine scheduling problem with time-lags, where all the time-lags are in a chain form, i.e. there exists a ordered subset of the jobs such that any pair of consecutive jobs is associated with a positive time-lags. None of the remaining jobs can be associated with positive time-lags.

Our interest in this special case arise from two main applications of the problem. The problem can be used to obtain lower bounds for the job-shop scheduling problem and for one-machine scheduling problem with general time-lags. Since, these problems can be found frequently in practice, solving to optimality the special one-machine scheduling problem with time-lags in polynomial time can lead to more efficient enumerative algorithms, for the previous problems.

The paper is organized as follows: first, we present the one-machine scheduling problem with time-lags, a review the solutions methods and an enumerative method to solve it. In the next section, we describe the polynomial algorithm for the special case, the early-late algorithm. In section 4, we present some examples and in the next section, we describe the computational experiments when using the early-late algorithm to obtain lower bounds for the job-shop scheduling problem. Section 6 concludes with general remarks on this work and further research.

2. The one-machine scheduling problem with time-lags

The one-machine scheduling problem with time-lags can be described as follows: a set of jobs $J_1, \ldots, J_n$ have to be scheduled on one machine; each job J_j, $j \in N = \{1, \ldots, n\}$, has a release date r_j, a processing time p_j, and a delivery time q_j. Each job cannot be processed before its release time. Whereas at most one job can be processed at a time, all jobs can be simultaneously delivered; if C_j denotes the time at which job J_j completes processing, then it is delivered at time $L_j = C_j + q_j$. There also can exists precedence constrains, <, between the jobs, i.e. if $J_j < J_k$ then J_k cannot start processing before J_j has been finished. These precedence constraints can be represented by a directed graph. Finally, between the completion time of J_j, C_j, and the starting time of J_k, S_k, there must exists a time interval $l_{jk} \geq 0$, i.e $C_j + l_{jk} \leq S_k$, know as finish-start time-lags. The objective is to minimize the maximum lateness, i.e. $L_{\max} = \max_{j \in N} L_j$. We denote this problem by $1 \left| r_j, q_j, prec\left(l_{ij}\right) \right| L_{\max}$ and if preemption is allowed, by $1 \left| pmtn, r_j, q_j, prec\left(l_{ij}\right) \right| L_{\max}$. Let $L_{\max}(\sigma, I)$ be the length of the feasible schedule σ for an instance I.

The $1 \left| r_j, q_j, prec\left(l_{ij}\right) \right| L_{\max}$ can be considered as a generalization of the usual one-machine scheduling problem $1 \left| r_j, q_j, prec \right| L_{\max}$, i.e. the time lags are all equal to zero. The problem is NP-Complete, even if preemption is allowed. The result was proved independently by Lourenço (1993) and Balas, Lentra and Vazacoupolos (1995).

The Horn's algorithm, Horn (1974), solves to optimality the standard one-machine problem if preemption is allowed. If no preemption is allowed the Carlier branch-and-bound algorithm, Carlier (1982), solves the standard one machine problem, and Balas, Lentra and Vazacoupolos (1995) proposed a branch-and-bound algorithm to solve the $1 \left| r_j, q_j, prec\left(l_{ij}\right) \right| L_{\max}$.

Brucker, Hilbig and Hurink (1997) presented a branch-and-bound algorithm for solving the single-machine scheduling problem with arbitrary time-lags and have shown that most of the classical scheduling problems can be polynomially reduced to this problem. Brucker and Knust

(1998) considered a single-machine scheduling problem with time-lags $l_{ij} = l \geq 0$ and derive new complexity results and reductions from other scheduling problems. They also presented a survey on the complexity results and algorithms for single-machine problems.

The following web page maintains a list of recent results and open-problems in the area of scheduling: http://www.mathematik.uni-osnabrueck.de/research/OR/.

The Longest Tail Rule can be generalized to obtain a feasible schedule to the one-machine scheduling problem with time-lags by dynamically updating the release dates of the jobs to conform with the time lags constraints. We will designate this method by **LTRTL**.

The LTRTL schedules the jobs sequentially choosing at each step the job with the longest delivery time among those available for scheduling. Let $C_j(\sigma)$ denote the completion time of job J_j in schedule σ, $j=1,...,n$. We shall say that, in the case of time-lags, a job J_j is said to be available at time t, if $\bar{r}_j \leq t$, where the release date is updated as follows $\bar{r}_j = \max\left[r_j, \max_{\forall k: J_k < J_j} \{C_k(\sigma) + l_{kj}\}\right]$, and all jobs J_k such that $J_k < J_j$ are already completed.

The Horn's algorithm can also be generalized to obtain feasible solutions for the $1\left|pmtn, r_j, q_j, prec\left(l_{ij}\right)\right|L_{\max}$ problem in a similar way as done above for the LTRTL. Due to this similarity, we will call this the preemptive LTRTL (**pLTRTL**). The main difference is as follows: if a job with higher delivery time becomes available, the processing of the current job can be interrupted and this job will be reintroduce in the list of available jobs but considering only the remaining processing time.

The pLTRTL creates preemptions only at the modified release dates $\bar{r}_j$, and not at the first one. Therefore, the rule creates at most *n-1* preemptions. The pLTRTL can be easily implemented to run in *O(nlog n)* time with the use of two priority queues.

Clearly, the optimal value of the preemptive standard one-machine scheduling problem is a lower bound for the $1\left|pmtn, r_j, q_j, prec\left(l_{ij}\right)\right|L_{\max}$. It is well known that this value is equal to

$L(K) = \min_{j\in K} r_j + \sum_{j\in K} p_j + \min_{j\in K} q_j$, for some $K \subseteq \{1,\ldots,n\}$, Carlier (1982). The jobs in K bellow to the critical path in the conjunctive graph of the correspondent optimal schedule σ^*. Let J_c be the critical job of the schedule σ^*, i.e. the job J_c such that $L_{\max}(\sigma^*, I) = C_c(\sigma^*) + q_c$.

Next, we present some basic results which will help us to design the proposed polynomial algorithm. We can assume that the input data already satisfy the following conditions:
if $J_j < J_k$ then $r_k \geq r_j + p_j + l_{jk}$, and $q_j \geq l_{jk} + p_k + q_k$.
If not, the data can be modified without changing the solution of the problem, Lageweg, Lenstra and Rinnooy Kan (1976).

Theorem 1: Let σ be a schedule obtained by the pLTRTL for some instance I of $1|pmtn, r_j, q_j, prec(l_{ij})|L_{\max}$. So, we have that $L_{\max}(\sigma, I) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + \min_{j\in K} q_j$ for some subset *K* of the set of the jobs.

Proof. Let *t* be the latest time such that each job J_j processed in the interval $[t, C_c(\sigma)]$ has $\bar{r}_j \geq t$. Denote by $\bar{r}_j$ the updated released times; note that, the only jobs that have their released date increased are the ones associated with the time-lags. In other words, if during the execution of the algorithm, the release date of job J_j is increased, then job J_j is only available for processing at time $\bar{r}_j$. If the job J_j is not associated with any time-lag constraint, then $\bar{r}_j = r_j$. Moreover, when J_k is scheduled, if $J_k < J_j$, then the release date of J_j can be increased, $\bar{r}_j = C_k(\sigma) + l_{kj}$, to satisfy the time-lags.

Let K be the index of the subset of the jobs processed in this interval. This interval contains no idle time, since if there is idle time, by pLTRTL, the end of this idle time satisfied the criteria used to select *t*, and is later than *t*.

Now, if $\min_{j\in K} q_j = q_c$, since there is no idle time between *t* and $C_c(\sigma)$, and by the choice of *t* we can conclude that: $L_{\max}(\sigma, I) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + \min_{j\in K} q_j$.

To prove that $\min_{j\in K} q_j = q_c$, assume this is not true, and consider the latest job J_l processed before J_c such that $q_l < q_c$. All jobs, $J_k, k \in K$ processed between the completion time of J_l, say *t'*, and the completion time of J_c, have delivery time $q_k \geq q_c > q_l$, therefore they have to be released after the completion of J_l, because otherwise, the modified preemptive LTRTL would have preempt J_l to start some J_k. So, all these jobs, J_k have $\bar{r}_k \geq t'$. But in this case we should have select *t'* instead of t, which is contradiction. σ

Lemma 2: Consider any feasible schedule σ obtained by the pLTRTL, if a job has some part processed in the critical path K then this job is totally processed between $\min_{j\in K} \bar{r}_j$ and $\min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j$.

Proof. Suppose that we replace each job J_j by new jobs with processing times equal to 1, and the same release dates, delivery times and time-lags.
Then, by Theorem 1, $L_{\max}(\sigma, I) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + \min_{j\in K} q_j$.
Let K_1 be the subset of the unit jobs corresponding to the units in *K*, then

$$L_{\max}(\sigma, I) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + \min_{j\in K} q_j = \min_{i\in K_1} \bar{r}_i + |K_1| + \min_{i\in K_1} q = L_{\max}\left(\sigma^{unit}, I^{unit}\right).$$

If a unit job is one unit of job J_j, and belongs to K_1, then every other unit of this job belongs to K_1, and therefore this job is totally processed between $\min_{j\in K} \bar{r}_j$ and $\min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j$.

Otherwise, suppose that exist a job J_j which has one unit processed outside the interval between $\min_{j\in K} \bar{r}_j$ and $\min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j$. Let $\overline{K} = K_1 \cup \left\{J_j^{unit}\right\}$, where J_j^{unit} is that unit job that does not belong to K_1. So let, $L_{\max}\left(\bar{\sigma}, I\right) = \min_{j\in \overline{K}} \bar{r}_j + \left|\overline{K}\right| + \min_{j\in \overline{K}} q_j$, and also all jobs in K have their units jobs in $\overline{K}$, then

$L_{\max}\left(\bar{\sigma},\bar{I}\right)=\min_{j\in K}\ \bar{r}_j+\sum_{j\in K}p_j+\min_{j\in K}\ q_j=L_{\max}(\sigma,I)$. Since $\min_{j\in\overline{K}}\ \bar{r}_j=\min_{i\in K_1}\ \bar{r}_i$ and $\min_{j\in\overline{K}}\ q_j=\min_{i\in K_1}\ q_i$, then $L_{\max}\left(\bar{\sigma},\bar{I}\right)=L_{\max}\left(\sigma^{unit},I^{unit}\right)+1=L_{\max}(\sigma,I)$, which is a contradiction because $L_{\max}\left(\sigma^{unit},I^{unit}\right)=L_{\max}(\sigma,I)$ by above. σ

Theorem 3: There exists a set of delivery times, such that if we apply the pLTRTL rule to a new instance with all data equal to the original instance *I*, but considering this new set of delivery times, we obtain an optimal schedule for *I*.

Proof. Given an optimal schedule σ^* for the instance I, with length $L_{\max}(\sigma^*,I)$, let the new set of delivery times be $\bar{q}_j=L_{\max}(\sigma^*,I)-C_j(\sigma^*)$ for all jobs. These delivery time satisfy the property that $\bar{q}_j>\bar{q}_k$ whenever $J_j<J_k$. Modify the release dates so that they also conform to the precedence constraints and the time-lags: $\bar{r}_j=\max\left\{r_j,\max_{\forall k:J_k<J_j}\left(C_k(\sigma^*)+l_{jk}\right)\right\}$. Designate this instance by $\bar{I}$.

Apply the Horn's rule to $\bar{I}$ to obtain the schedule $\bar{\sigma}$. By the optimality of this method, each job J_j has completion time in $\bar{\sigma}$ no later then $C_j(\sigma^*)$. Therefore, the new schedule is feasible and also optimal for *I*.

Now let $\hat{I}$ be the instance with the original release dates and the delivery times $\bar{q}_j$. Apply the pLTRTL to $\hat{I}$ to obtain the schedule $\hat{\sigma}$. Note that if $\hat{r}_j$ denotes the modified release dates formed by computing $\hat{\sigma}$, and we apply the Horn's rule to the instance $\hat{I}$ with release dates $\hat{r}_j,\ j=1,\ldots,n$ and delivery times $\bar{q}_j,\ j=1,\ldots,n$, we again obtain $\hat{\sigma}$.

We claim that $\hat{r}_j\le\bar{r}_j$ for *j=1,..., n*. If this holds, then $\bar{\sigma}$ is also a feasible schedule for instance $\hat{I}$. Since the Horn's rule minimizes $L_{\max}$, then $L_{\max}\left(\hat{\sigma},\hat{I}\right)\le L_{\max}\left(\bar{\sigma},\bar{I}\right)=L_{\max}(\sigma^*,I)$. Since $\hat{\sigma}$ also satisfies the time-lags constraints, it is optimal for *I*. This also implies that $C_j(\hat{\sigma})\le C_j(\sigma^*),\ j=1,\ldots,n$.

To prove that $\hat{r}_j \le \bar{r}_j, j = 1,\ldots,n$, we need only focus on the jobs with time-lags. Suppose that there exists a job such that $\hat{r}_j > \bar{r}_j$, and let J_j be the first job in the schedule $\hat{\sigma}$ that verifies this condition. But, since $\hat{r}_k \le \bar{r}_k$, for all jobs J_k scheduled before than J_j in $\hat{\sigma}$, and the delivery times are equal in both instances, then $C_k(\hat{\sigma}) \le C_k(\sigma^*)$, so

$\hat{r}_j = \max\left\{r_j, \max_{\forall k: J_k < J_j} C_k(\hat{\sigma}) + l_{jk}\right\} \le \max\left\{r_j, \max_{\forall k: J_k < J_j} C_k(\sigma^*) + l_{jk}\right\} = \bar{r}_j$, which is a contradiction.

Therefore, $\hat{\sigma}$ is an optimal schedule to the problem, and the theorem is proved. σ

Note that the delivery times are only consider in the pLTRTL as priorities to decide which is the next job to be scheduled. Therefore, an obvious enumerative method for the $1|pmtn, r_j, q_j, prec(l_{ij})|L_{\max}$ can be design considering all possible combinations of priorities:

- At level zero, there is a node called the root of the search tree.
- The root has *n* children, where each child is a node where the priority of job J_1 is fixed, i.e. $\tilde{q}_1 = k, k = n,\ldots,1$.
- For each node of level *i*, we have an associated vector *v* of dimension *i* which represents the fixed priorities of the first *i* jobs. We branch on each of this nodes, by considering the *n* possible priorities of job J_{i+1}. Children are arranged from left to right in order of decreasing priorities.
- At the leaves we have all possible priorities of the jobs. We can apply the pLTRTL to each one, using the priorities. The schedule with the smallest length is optimal.

We would like to point out that the only use of the priorities is to decide which job is scheduled next in the pLTRTL. The value of a schedule σ is always obtained by using the delivery times, i.e. $L_{\max}(\sigma, I) = \max_{j \in N} L_j$.

Next, we consider a special case of the $1|pmtn, r_j, q_j, prec(l_{ij})|L_{\max}$ problem, where all the positive time-lags have the special form of a chain. Let $H = \{1,2,\ldots,h\} \subseteq N$ be a subset of jobs such that $J_1 < J_2 < \ldots < J_h$ and there are no positive time-lags neither precedence constraints between jobs on this chain and the remaining ones. We will denote this problem as $1\left|pmtn, r_j, q_j, chain\left(l_{ij}\right)\right|L_{\max}$. Next we will proved that the above enumerative method can be simplified to solve to optimality this problem, and in the next chapter we will go further and prove that, by modifying in a correct way the release dates and the delivery times, we will solve this problem in polynomial time.

Lemma 4: To obtain the optimal solution of $1\left|pmtn, r_j, q_j, chain\left(l_{ij}\right)\right|L_{\max}$, we need only to change the delivery times of the chain jobs in the previous enumerative method.

Proof. Recall the proof of Theorem 3. Given an optimal schedule $\sigma*$ for an instance *I*, with length $L_{\max}(\sigma*, I)$, let the delivery times be $\bar{q}_j = L_{\max}(\sigma*, I) - C_j(\sigma*)$ for $j \in H$; and $\bar{q}_j = q_j$, for $j \notin H$. As before, obtain $\bar{\sigma}$ and by the optimality of the Horn's rule, each job $J_j, j \in H$ has completion time in $\bar{\sigma}$ no later than $C_j(\sigma*)$. Therefore, the new schedule is feasible and also optimal for *I*. Also, note that the only jobs that have their release dates changed are the chain jobs. Therefore, to complete the prove just follow the prove of Theorem 3.σ

This method makes *O(n^h)* calls to the pLTRTL method. Next, we will present an algorithm that makes a polynomial (in *n*) number of calls of the pLTRTL. This algorithm is based on the enumerative method just described, but using an efficient pruning of the search tree and good dominance relations.

3. Early-Late Algorithm

The early-late algorithm finds the optimal solution of the $1\left|pmtn, r_j, q_j, chain\left(l_{ij}\right)\right|L_{\max}$ by considering a polynomial number of tree nodes of the previous enumerative method, eliminating the remaining ones by using lower bounds and dominance relations.

At node N_i of the search tree at level *i*, we associated a vector *v* of dimension *i* which represents the fixed priorities of the first *i* jobs in the chain. Let this subset of jobs be denoted by H_1. Since the priorities of these jobs are fixed and they are the first *i* jobs in the chain, then we can compute the completion time of these jobs in any schedule obtained by applying the pLTRTL of any descendent of N_i. We also associate with N_i an instance I_i which uses the priorities *v* for the jobs in H_1 and the original priorities for the remaining jobs. The release dates of the jobs in the chain are modified during the running of the algorithm. All the other release dates are unchanged.

The basic idea of the algorithm is to obtain a lower bound and an upper bound for instance I_i at each iteration, which originally is equal to *I*. If the upper bound is equal to the lower bound the algorithm stops because that optimal solution was found. Otherwise, the instance I_i is modified using some dominance rules. The only modifications needed are the release dates and delivery times of the chain jobs. When a modification is made, it means that we are changing from one node to another in the search tree of the enumerative method. We will prove the optimality of the algorithm and that it runs in polynomial time.

A lower bound of I_i, $LB(I_i)$, is obtained by applying the Horn's rule to a modified I_i where we consider the original delivery times and ignore the time lags. The release dates are the same for both instances, but can be different from the original instance *I*. Note that the modified instance is relaxation of I_i, so the optimal value is a lower bound for $L_{\max}(\sigma^*, I_i)$.

By applying the pLTRTL to I_i, using the priorities, we obtain an upper bound for the I_i, designated by $L_{\max}(\sigma_i, I_i)$. Note that, the schedule obtained is also feasible for the original instance *I*, since only the priorities of the jobs are changed and the release dates are increased. Therefore $L_{\max}(\sigma_i, I_i)$ is a upper bound for *I*.

Theorem 5: Let σ_i be an schedule obtained by applying the pLTRTL with the priorities to instance I_i, then $L_{\max}(\sigma_i, I_i) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + q_c$ for some subset of the job, K.

Proof. Let *t* be the latest time such that each job J_j processed in the interval $[t, C_c(\sigma_i)]$ has $\bar{r}_j \geq t$. Let K be the index of the subset of the jobs processed in this interval. This interval contains no idle time, since if there is idle time, by the pLTRTL, the end of this idle time satisfied the criterion used to select *t*, and is later than *t*. Now, since there is no idle time between *t* and $C_c(\sigma)$ and by the choice of *t* we can conclude that:

$L_{\max}(\sigma, I_i) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + q_c$. σ

If $L_{\max}(\sigma_i, I_i) = LB(I_i)$ then the schedule σ_i is optimal for I_i, and consequently we do not need to branch from this node further.

If $L_{\max}(\sigma_i, I_i) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + q_c > LB(I_i) \geq \min_{j\in K} r_j + \sum_{j\in K} p_j + \min_{j\in K} q_j = L(K)$,

then σ_i has a job scheduled late, if $\min_{j\in K} \bar{r}_j > \min_{j\in K} r_j$ or early, $q_c > \min_{j\in K} q_j$. So, we will changed the priority of such a job. Therefore the algorithm will be designated by early-late algorithm, since at each step the priority of one job will be adjusted to force the scheduling of this job earlier or later than it is in the current solution.

Next, we will present two important definitions and some results, followed by the description of the algorithm. Later on, we will prove that the early-late algorithm obtains the optimal solution for the $1|pmtn, r_j, q_j, chain(l_{ij})|L_{\max}$ problem in polynomial time.

Definition 1: A schedule σ is late-active if there exists a job J_l in the chain, such that J_l is the first chain job in the schedule for which $l \in K$ and $r_l < \min_{j\in K} \bar{r}_j$.

Consider the following notation: define $C_{LB}(j)$ as a lower bound on the completion time of job J_j; initially, let $C_{LB}(j) = r_j + p_j$, $j = 1, \ldots, n$. This value will be updated during the algorithm. Moreover, whenever these values change, we update the release dates in the following way: $r_j = \max\{r_j, C_{LB}(j-1) + l_{j-1,j}\}$, $j = 2, \ldots, h$. Also, let σ_i be the schedule obtained at iteration *i* of the early-late algorithm by the pLTRTL considering the priorities of instance I_i.

Theorem 6: If schedule σ_i is late-active, then exists at least a job $J_k \in H$ such that $C_k(\sigma_i) > C_{LB}(k)$.

Proof. If σ_i is late-active, then $L_{\max}(\sigma_i, I_i) > L(K)$, since $l \in K$ and $r_l < \min_{j\in K} \bar{r}_j$. If $J_k, k \in H$ such that $C_k(\sigma_i) > C_{LB}(k)$ does not exist, then $C_j(\sigma_i) = C_{LB}(j),\ j = 1,\ldots,h$.
Consequently, $\bar{r}_j = r_j, j = 1,\ldots,h$ because clearly $r_j \le \bar{r}_j$ and
$r_j = \max\{r_j, C_{LB}(j-1) + l_{j-1,j}\} \ge C_{LB}(j-1) + l_{j-1,j} = \bar{r}_j,\ j = 2,\ldots,h$.
Therefore $L_{\max}(\sigma_0, I_i) = L(K)$, which proves that must exist a job $J_k, k \in H$ such that $C_k(\sigma_i) > C_{LB}(k)$.σ

The first job J_k in the chain verifying $C_k(\sigma_i) > C_{LB}(k)$ is called the late-active job. Note that $k < l$ and $k \notin K$

In this case, J_k is scheduled "late", and as a result, we have no guarantee that this schedule is optimal. Thus, to get a possible improvement in the schedule we should complete the late-active job earlier, which implies an increase in the priority of this job.

Next, we will define early-active schedule, which apply for the case where $q_c > \min_{j\in K} q_j$.

Definition 2: A schedule σ is early-active if there exists a job $J_k, k \in H \cap K$, which is different from J_c, the critical job, and J_k is the last job in the critical path for which $q_k < q_c$.This job is called the early-active job.

For an early-active schedule, since J_k is scheduled earlier than J_c, we cannot prove optimality. To obtain a possible improvement of such a schedule, we will see next that we should decrease the priority of the early-active job.

The next theorem proves that we can decrease the priority of the early-late job J_k, since we will not find a better schedule than s if $\tilde{q}_k \geq \tilde{q}_c$, where $\tilde{q}_j$ represents the priority of job J_j.

Theorem 7: Let s be an early-active schedule obtained by applying pLTRTL to instance I_i, where J_c is the critical job and J_k is the early-active job. Let I' be an instance equivalent to I_i, but where $\tilde{q}_k \geq \tilde{q}_c$. Apply the pLTRTL to I' to obtain the schedule s'. Then $L_{\max}(s, I_i) \leq L_{\max}(s', I')$.

Proof. When we apply the pLTRTL to this instance I', the job J_k will have at least as high a priority as the critical job J_c. Note that $c \notin H$, otherwise J_c would have been scheduled before J_k. Since $J_k \in H$, $J_c \notin H$ and $\tilde{q}_k \geq \tilde{q}_c$, J_k will be always completed before J_c. Hence J_c will be not completed earlier than $C_c(s)$, since no other data was changed. Note that there is no idle time between $\min_{j \in P(s)} \bar{r}_j$ and $C_c(s)$ in s, then J_c is completed at $C_c(s)$ in s'. Therefore $L_{\max}(s, I_i) = C_c(s) + q_c \leq L_{\max}(s', I')$.$\nabla$

The next lemma says that we can eliminate for further analysis all nodes where $\tilde{q}_k \geq \tilde{q}_c$ and at least one job in H_1 has smaller priority than in node N_i.

Lemma 8: Let s be an early-active schedule obtained by applying the pLTRTL to instance I_i, where J_c is the critical job and J_k is the active job. Let I' be an instance equivalent to I_i, but where $\tilde{q}_k \geq \tilde{q}_c$ and at least one job in H_1 has smaller priority than in I_i. Apply the pLTRTL to I' to obtain the schedule s'. Then either $L_{\max}(s, I_i) \leq L_{\max}(s', I')$ or if not, then by considering $\tilde{q}_k = \tilde{q}_c - 1$ and applying the pLTRTL, we still obtain s'.

Proof. The main idea behind this result is as follows. Observe that if we decrease the priority of some job J_j, $j \in H_1$, and apply the pLTRTL, this job is completed in s' no earlier than $C_j(s)$. If s' is different from s, then $\bar{r}_k$ is bigger in s' than in s. Since $\tilde{q}_k \geq \tilde{q}_c$ then either some job in K is completed after J_k, and as before $L_{\max}(s, I_i) \leq L_{\max}(s', I')$, or else by considering $\tilde{q}_k = \tilde{q}_c - 1$ and applying the pLTRTL, we still obtain σ', because all jobs that are completed after J_c in σ have smaller priority than $\tilde{q}_c$, otherwise J_c will not be the critical job.∇

The next theorem will prove that we can increment the lower bound of the completion time of the late-active job, and for all chain jobs before this one.

Theorem 9: If J_k is late-active at iteration *i* and we let $\tilde{q}_k = n$, then there is no schedule σ better than the σ_i where $C_k(\sigma) < C_{LB}(k)$, for $C_{LB}(k) = C_k(\sigma_{i+1})$ where $C_k(\sigma_{i+1})$ is the completion time of J_k in the schedule σ_{i+1} obtained by the pLTRTL at iteration $i+1$. Therefore, we can update $C_{LB}(j) = C_j(\sigma_{i+1}), \forall j \le k$.

Proof.
If σ_0 is late-active and J_k is the active job, then we do $\tilde{q}_k = n$. At σ_1, iteration 1, J_k has the higher priority at time $t = r_k$ and by the way we update $\tilde{q}_k$, J_k is fully scheduled between r_k and $r_k + p_k$. Also, if $J_j < J_k$, and $J_j \in H$, then J_j cannot be completed earlier, since $C_j(\sigma_0) = C_j(\sigma_1) = r_j + p_j = C_{LB}(j)$. Note that J_k is the first job in the chain such that $C_k(\sigma_0) > C_{LB}(k)$.

Suppose that the claim is true for all schedules $\sigma_0, \ldots, \sigma_{i-1}$. Let H_1 denote the index-subset of jobs such that $C_j(\sigma) = C_{LB}(j)$, and $H_2 = H - H_1$. If σ_i is late-active, and J_k is the late-active job, then we do $\tilde{q}_k = n$, and $\tilde{q}_j = n$ for $j \notin H_1$ and $J_j < J_k$, and obtain σ_{i+1}. At iteration *i*, if $j \in H_1$, then $j \in H_1$ in the next iteration and $C_j(\sigma_{i+1}) = C_j(\sigma_i) = C_{LB}(j)$. For $j \in H$, if $j \notin H_1$ and $J_j < J_k$, then at iteration $i+1$, $C_j(\sigma_{i+1}) = C_{LB}(j)$ since $\tilde{q}_j = n$, and so we make $j \in H_1$. For J_k, since $\tilde{q}_k = n$, then at time $t = r_k$, J_k is the available job with higher priority; so $C_k(\sigma_{i+1}) = r_k + p_k = C_{LB}(k)$. Therefore, if σ_i is late-active, by induction, the jobs just added to H_1 verify the claim.

If s_i is early-active and J_k is the active job, then $k \in H_1$. If $j \in H_1$ and $J_j < J_k$ then $C_j(s_{i+1}) = C_j(s_i) = C_{LB}(j)$. For J_k and $J_j, k, j \in H_1$ such that $J_j < J_k$, then by Theorems 7 and 8, if $C_j(s) < C_{LB}(j)$ then s is no better than the best schedule found. ∇

A straightforward argument shows that if $L_{\max}(s_i, I_i) > LB(I_i)$ must be late-active or early-active. The next theorem will show that if neither happens, then the optimal schedule for I_i was found.

Theorem 10: If the schedule s obtained by applying the pLTRTL to instance I_i, is neither late-active or early-active, then s is an optimal schedule for I_i.

Proof. We have seen already in Theorem 6, if $C_j(s) = C_{LB}(j),\ j = 1,\ldots,h$, then the schedule is optimal for I_i. And, the same happens if for all $j \in K \cap H$, $j \in H_1$ and $c \in H$ then $L_{\max}(s_i, I_i) = L(K)$, by previous theorems.

By Theorem 5, we have that $L_{\max}(s, I_i) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + q_c$. If K has no job such that $j \in H_1$, then $q_c = \min_{j\in K} q_j$. And, if for all jobs $j \in K \cap H$, $j \in H_1$, then $\bar{r}_j = \min_{j\in K} r_j$.

Lets see now, the case that exists at least one $l \in H_2$ in the critical path. Let $l \in H_2$ be the first job in the critical path and, let $k \in H_2$ be the first job in the schedule for which $C_k(s) > C_{LB}(k)$, so $r_k = \bar{r}_k$. If $r_l \geq \min_{j\in K} \bar{r}_j$ and $k \notin K$, then $\min_{j\in K} \bar{r}_j \leq r_l \leq r_g \leq \bar{r}_g$, for $g \in H$ and $J_l < J_g$; therefore $\min_{j\in K} \bar{r}_j = \min_{j\in K} r_j$. Since $g \in H_2$, then $q_c = \min_{j\in K} q_j$. If $k \in K$, then $\min_{j\in K} \bar{r}_j \leq r_k$. In any case, $L_{\max}(s, I_i) = L(K) = \min_{j\in K} \bar{r}_j + \sum_{j\in K} p_j + \min_{j\in K} q_j$, therefore s is optimal. Furthermore, if s is neither early-active or late-active then $L_{\max}(s, I_i) = L(K)$. ∇

The Early-Late Algorithm

Next, we describe the early-late algorithm. Assume that the input data already satisfies the conditions associated to the precedence constraints and time intervals, i.e.
if $J_j < J_k$ and $J_j, J_k \notin H,$ then $r_k \geq r_j + p_j$, and $q_j \geq p_k + q_k$;
if $J_j < J_k$ and $J_j, J_k \in H,$ then $r_k \geq r_j + p_j + l_{jk}$, and $q_j \geq l_{jk} + p_k + q_k$.

1. The algorithm maintains a lower bound on the completion time of the jobs in the chain; initially let $C_{LB}(j) = r_j + p_j, \forall j.$ To obtain the initial priorities, sort the delivery times in nondecreasing order and obtain a permutation π. Let $\tilde{q}_j = \pi(q_j), \forall j$. The algorithm also maintains a subset of jobs H_1; for $j \in H_1$, there is a dominance relation which implies that if $C_j(\sigma) < C_{LB}(j)$, then there is no schedule better than the best schedule found. Initially, H_2=H and $H_1 = \phi$.

2. Apply the Horn's rule to the original instance *I* at the root of the search tree to obtain σ_0^{LB}, and a lower bound to the problem, $LB(I)$. If the time-lag between J_j and J_{j+1}, for all $j = 1,...,h-1$, is satisfied by σ_0^{LB}, STOP. (This schedule is optimal.)

3. Obtain the feasible schedule σ_0 by applying the pLTRTL to *I*. If $L_{\max}(\sigma_0, I) = LB(I)$, STOP. (This schedule is optimal). Otherwise, σ_0 is late-active. Let J_{k_0} be the late-active job. Let $H_1 = H' = \{1, \ldots, k_0\} \subset H$, and let $H_2 = H - H_1$, and for all $j \in H'$, let $\tilde{q}_j = n$. Consider this instance I_0, as the current instance. Therefore we are in the search tree at the node at level k_0 on the leftmost path from the root. Let $i = 0$, $\sigma_{best} = \sigma_0$ and $L_{\max}(\sigma_{best}, I_{best}) = L_{\max}(\sigma_0, I)$.

4. Let $i = i + 1$. Obtain the schedule σ_i by applying the pLTRTL with priorities to current instance.

   4.1. For $j = 1$ to h do:

4.1.1. if $j \in H_1$ then $C_{LB}(j) = C_j(\sigma_i)$;

4.1.2. $r_j = \max\{r_j, C_{LB}(j-1) + l_{j-1,j}\}$;

4.2. If $L_{\max}(\sigma_{best}, I_{best}) > L_{\max}(\sigma_i, I_i)$, then update σ_{best} and the length $L_{\max}(\sigma_{best}, I_{best})$;

4.3. Obtain a lower bound to the current instance I_i, designated by $LB(I_i)$. If $LB(I_i) \geq L_{\max}(\sigma_i, I_i)$, or we obtain an optimal schedule for the current instance I_i, then STOP (σ_{best} is optimal);

4.4. Otherwise,

4.4.1. if σ_i is late-active and J_k is the late-active job, let J_{l-1} be the job in H_1 with biggest index, and so, $H' = \{J_l, \ldots, J_k\} \subset H$; and let $H_1 = H_1 \cup H'$, $H_2 = H_2 - H'$ and for all $j \in H'$ let $\tilde{q}_j = n$. In this case, we are going *down* in the search tree to level *k* by the left most path.

4.4.2. Else, σ_i is early-active, and J_k is the early-active job, J_c is the critical job, let $\tilde{q}_k = \tilde{q}_c - 1$ and $i = i + 1$. In this case, this means going *to the right* in the search tree, i.e. the node where $\tilde{q}_k = \tilde{q}_c - 1$ and all remaining priorities are maintained. Also, as we will see later, we can eliminate all the nodes where $\tilde{q}_k \geq \tilde{q}_c$, since these nodes do not lead to better schedules than can also be found by considering $\tilde{q}_k = \tilde{q}_c - 1$.

4.4.3. Go to 4.

5. End of the early-late algorithm. Output the best schedule σ_{best} and the length $L_{\max}(\sigma_{best}, I_{best})$.

Next, we will prove that the algorithm gives the optimal solution.

Theorem 11: The early-late algorithm obtains the optimal schedule for the one-machine scheduling problems with positive time-lags in a chain of jobs.

Proof. Consider that, at iteration *i* we are at node N_i and obtained the schedule σ_i. If σ_i is late-active and J_k is the active job, then for all job $j \in H'$, $J_j < J_k$ and for J_k, we let

$\tilde{q}_j = n$.This means that we are going *down* in the search tree to level k by the left most path. If σ_i is early-active, and J_k is the active job, then by Theorems 7 and 8, it is enough to consider $\tilde{q}_k = \tilde{q}_{c-1}$, i.e. we decrease the priority of job J_k. Therefore, this means that we are going *to the right* in the search tree, going to the node at level k where $\tilde{q}_k = \tilde{q}_{c-1}$ and all the remaining priorities are maintained. Furthermore, by Lemma 8, all the nodes where $\tilde{q}_k \geq \tilde{q}_c$ can be eliminated, since these nodes do not lead to a better schedule that cannot also be find by considering $\tilde{q}_k = \tilde{q}_{c-1}$.

The algorithm will maintain the best schedule found so far, designated by σ_{best}. The algorithm stops when either $LB(I_i) = L_{\max}(\sigma_{best}, I_{best})$ or when we obtain an optimal schedule to instance I_i, i.e., $L_{\max}(\sigma_i, I_i) = LB(I_i)$.

Suppose that the algorithm stopped at node N_i. Then, at any descendent of this node in which we consider different priorities to the jobs in the chain but not in H_1, will not lead to a schedule of better value than $LB(I_i)$. Consider any other node at the same level, to the right of N_i. At this node, at least one of the jobs in H_1 has smaller priority than at N_i. For these nodes, where none of the jobs in H_1 have higher priority than at N_i, implies that at least some job J_l will be completed no earlier than in σ_i, and at some point, later. Therefore, some released dates will eventually increase, and so $LB(I_i)$ is a lower bound to the schedules associated with such nodes. If at least one job in H_1 has higher priority than in N_i, we can apply the previous theorems and therefore, we have already seen that these nodes can be eliminated.

In this way, we visit some nodes in the search tree and eliminate all of the remaining ones by knowing that they do not lead to better schedules. Once the remaining nodes have been eliminated, we can conclude that the best schedule found is optimal. ∇

Theorem 12: The early-late algorithm solves the one-machine scheduling problem with a chain of jobs, $1\left|pmtn, r_j, q_j, chain\left(l_{ij}\right)\right|L_{\max}$, in $O(hn)$ calls of the modified preemptive EDD rule. Therefore, it runs in $O\left(hn^2 \log n\right)$ time.

Proof. There are at most h late-active schedules. If σ_i is early-active and J_k is the early-active job, then there cannot exist another early-active schedule with J_k as the early-active job and J_c as the critical job. This implies that the algorithm does at most $O(hn)$ iterations.∇

4. Example

In this section, we present an example of the application of the $1\left|pmtn, r_j, q_j, chain\left(l_{ij}\right)\right|L_{\max}$.

We have outputed one instance of the one-machine scheduling problem during the execution of the branch-and-bound method by Applegate and Cook (1991) applied to the instances of the job-shop scheduling problem ABZ5, as well as the corresponding directed graph of the precedence constraints and all time intervals obtained.

Table 1: An instance of the one-machine scheduling problem.

Jobs	Release date	Processing time	Delivery time
1	947	92	0
2	316	50	648
3	924	77	0
4	627	54	347
5	0	69	747
6	601	79	88
7	365	96	639
8	171	82	845
9	274	81	565
10	0	50	1019

In Tables 1 and 2, we present an instance of the one-machine scheduling problem with time-lags obtained during the branch-and-bound method for the instance ABZ5 of the job-shop scheduling problem. Considering all lags to be zero, the optimal value when we applied the Horn’s rule is 1101; if we do not allow preemption and apply Carlier's algorithm, Carlier (1982), the optimal value is 1108. But these lower bounds can be improved to 1116 just by considering the chain $J_9 < J_8 < J_5$, their respective time intervals and allowing preemption. These, and other examples like this, lead us to think about how to use the one-machine scheduling problem with time-lags to obtain new lower bounds to the job-shop scheduling problem.

Table 2: Precedence constraints and time-lags for instance in Table 1.

Jobs	**Precedence Constraints and Time-Lags**			
1	T_2, 540	T_3, 243		
4	T_0, 432	T_2, 559	T_3, 262	T_5, 232
6	T_0, 181			
7	T_0, 402	T_2, 576	T_3, 279	T_5, 262
8	T_2, 457			
9	T_0, 704	T_2, 762	T_3, 504	T_5, 224

5. Computational Results

In the section, we will present a computational experiment and the results obtained when several methods are applied to different versions of the one machine scheduling problems. We would like to point out that the main aim of this paper is to present the polynomial algorithm for the $1\left|pmtn, r_j, q_j, chain\left(l_{ij}\right)\right|L_{\max}$, however the objective of this section is to shown the potential applicability of the early-late algorithm.

We consider 10 instances of the one-machine scheduling problem obtained by relaxations of the job-shop scheduling problem. These instance were obtained throughout the running of teh Applegate and Cook (1991) branch-and-bound method. The methods applied to solve each version are:

- No time lags, pmtn $\left(1\left|pmtn, r_j, q_j, prec\right|L_{\max}\right)$. Horn's Rule
- Chain time-lags, pmtn $\left(1\left|pmtn, r_j, q_j, chain\left(l_{ij}\right)\right|L_{\max}\right)$. Early-Late algorithm
- All time lags, pmtn $\left(1\left|pmtn, r_j, q_j, prec\left(l_{ij}\right)\right|L_{\max}\right)$. Enumerative method
- No time lags, no pmtn $\left(1\left|r_j, q_j, prec\right|L_{\max}\right)$. Carlier's algorithm.

- Chain time-lags, no pmtn $\left(1\middle|r_j, q_j, chain\left(l_{ij}\right)\middle|L_{\max}\right)$. Enumerative method
- All time lags, no pmtn $\left(1\middle|r_j, q_j, prec\left(l_{ij}\right)\middle|L_{\max}\right)$. Enumerative method

With respect to these few examples we can make some observations:

- The release dates and the delivery times are strong enough, so the schedule obtained by applying the early-late algorithm does not violate the time-lag constraints, and in many cases the same thing happens even when we apply the Horn's rule. By considering all of the lags, in only one case, Example 1, we improve the lower bound with respect to the chain version of the problem.
- When we do not allow preemption, we observe that the instances were usually easily solved. If we apply Carlier's algorithm to solve this version (no lags/no pmtn), many times only one node of the search tree was needed, usually 2 or 3 and very few times 4 or more, and it runs very fast. Note that, we expected Carlier's algorithm to be fast, since the algorithm is known to perform well even in very large scale instances, and all instances for this problems have 10 jobs.
- When solving (all lags/no pmtn) version of the problem, we observed again that we need only to resolve very few conflicts in the order of the jobs, usually involving 2 jobs at most, and therefore we construct about 1 or 2 schedules. For this case of (all lags/ no pmtn), in four out of ten examples the bound was improved with respect to the case of (chain/pmtn). But, we need to solve a NP-complete problem, and the question is whether it is worthwhile spending more time to obtain a stronger lower bound.
- In many of the schedules obtained, very few preemptions occur and most of them can be eliminated without affecting the length of the schedule, i.e., if the preemption does not occur, the length of the schedule is not changed.
- We pay special attention to the MT10 since for this instance, the bound improved only two times. Note that the only cases in Table 10, in which the (all lags/no pmtn) does not help are examples from the MT10 instance. For this case, even if preemption is not allowed, it looks like the lower bound does not improve; in 6 examples, we get an improvement in only two.

Table 3: Results for 10 instances of the one-machine scheduling problem

Examples	**no time-lags pmtn.**	**Chain time-lags pmtn**	**all time-lags pmtn**	**no time-lags no pmtn**	**chain time-lags no pmtn**	**all time-lags no pmtn**
1 (LA19)	798	807	813	807	807	832
2 (MT10)	911	911	911	911	911	911
3 (MT10)	917	917	917	917	917	917
4 (MT10)	836	836	836	836	836	836
5 (MT10)	884	884	884	892	892	892
6 (ABZ5)	1101	1116	1116	1108	1116	1116
7 (LA19)	735	752	752	747	755	755
8 (ABZ5)	1147	1157	1157	1147	1157	1157
9 (MT10)	884	884	884	892	892	892
10 (MT10)	918	918	918	918	918	918

We have performed more tests, by integrating the early-late algorithm to obtain lower-bounds in the branch-and-bound algorithm developed by Brucker, Jurish and Sievers (1994). To improve the method, we changed the branching scheme to take advantage of the one-machine scheduling problems with time-lags. In Bruno and Lourenço (1998) two methods were proposed to this branching scheme, CMC and CTC methods, and extensive computational results are presented.

The CMC branching scheme arranges the blocks according to non-increasing length of the longest chain, if exist, associated to each block, instead of the non-decreasing cardinality as in Brucker, Jurish and Sievers (1994), say the BJS scheme. Meanwhile, the CTC branching scheme sorts the block according to the total length of all chains associated with the blocks. Note that, both rules try to take advantage of the existence of time-lags , where the first nodes to be evaluated are associated with larger chains.

In tables 4, we present some results obtained for the job-shop test problems.

We can observe that, in general, when the new rules were used, the branch-and-bound method search for fewer nodes than when using the BJS scheme. This happens in special for the larger instances. However, since it takes more time to calculate the new lower-bound, the use of this one not always lead to an improve in the running times. Between the schemes CMC and CTC, we can observe that the CMC usually needs less search-nodes than the CTC leading to better running times.

Table 4: Computational Results for job-shop instances.

			BJS			*CMC*			*CTC*		
	Dimensão	*Solução óptima*	*Solução obtida*	*nº de nodos*	*tempo (seg.)*	*Solução obtida*	*nº de nodos*	*tempo (seg.)*	*Solução obtida*	*nº de nodos*	*tempo (seg.)*
MT10		930	930	4249	513	930	11724	1493	930	4959	545
ABZ5		1234	1234	2147	244	1234	972	120	1234	972	103
ABZ6	10×10	943	943	135	15	943	168	18	943	153	14
ORB1		1059	1059	3124	37643	1059	3764	4983	1059	45159	5083
ORB2		888	888	2317	262	888	1284	156	888	1284	134
LA21		1046	*1057	765206	225890	*1046	208073	249365	*1046	242062	84934
LA22		927	927	10529	3488	927	6365	2041	927	7993	2169
LA23	15×10	1032	1032	6619	1829	1032	519	145	1032	3157	863
LA24		935	935	136630	45524	935	92736	33479	935	76831	24517
LA25		977	977	429524	135658	977	406705	136075	977	408647	118233
LA26		1218	1218	56639	24076	*1266	484788	313729	*1292	23394	11305
LA27		1235-1256	*1270	185116	116043	*1293	314154	201967	*1391	543788	332863
LA28	20×10	1216	*1273	20042	7889	*1309	27141	16803	*1278	7821	3050
LA29		1120-1164	*1202	326106	182291	*1195	453193	291783	*1225	628420	331613
LA30		1355	1355	368	126	*1379	437	187	*1399	16296	6635
LA31		1784	1784	8	3	*1797	11232	19339	*1788	20	9
LA32		1850	1850	1	0	1850	1	0	1850	1	0
LA33	30×10	1719	1719	77	41	*1753	2	1	*1729	1	0
LA34		1721	1721	15	6	*1798	3	1	*1823	10567	16321
LA35		1888	1888	24	11	1888	8	4	1888	7	3

6. Conclusions

In this work, we have considered the one-machine scheduling with time-lags. The main contribution of this work is the presentation of a polynomial algorithm, the early-late algorithm, that solves to optimality the special case where all time-lags have a chain form and preemption is allowed. This algorithm can be used to obtain lower-bounds to other complex scheduling problems, as the job-shop problem. We also present examples of this application and some computational testing. From the computational experiment, we can conclude the potential application of the early-late algorithm.

Further developments of this work are related to the extensions of the one-machine scheduling problem with chain time-lags and preemption, and their application to obtain lower bounds to more complex scheduling problems. We observed that the directed graph of the precedence constraints was frequently an almost-bipartite graph, i.e., there are two subsets of jobs containing most of the jobs such that a job in the first group has only to be processed before some of the jobs in the second group. The previous characteristic is associated with a one-machine scheduling problem with several chains. Also, we would like to study if it is worth to solve the problem without preemption. Therefore, as an extension of this research, we would like to develop efficient algorithms to solve the above problems. Even for NP-complete problems, we should study the following question: does it pay off to spend more time to obtain the stronger lower-bounds or not?

References

[1] Applegate D and Cook W (1991), A computational study of the job-shop scheduling problem, *ORSA - Journal on Computing*, **3**, 149-156.

[2] Balas E, Lenstra JK and Vazacoupolos A (1995), One machine scheduling with time-lags, *Management Science*, **41** (1), 94-109.

[3] Bruno PM and Lourenço HR (1998), New ramification rules in a branch-and-bound method to solve the job-shop scheduling problem, *Investigação Operacional*, **18**, 3-16.

[4] Brucker P, Hilbig T and Hurink J (1997), A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags, *Working paper*, Fachbereich Mathematik/Informatik, Universität Osnabrück, Germany.

[5] Brucker P, Jurish B and Sievers B (1994), A branch and bound algorithm for the job-shop scheduling problem, *Discrete Applied Mathematics*, **49**, 107-127.

[6] Brucker P and Knust S (1998), Complexity results for single-machine problems with positive finish-start time-lags, *Working paper*, Fachbereich Mathematik/Informatik, Universität Osnabrück, Germany.

[7] Carlier J (1982), The one-machine sequencing problem, *European Journal of Operational Research*, **11**, 42-47.

[8] Horn WA (1974), Some scheduling algorithms, *Naval Res. Logist. Quart*. **21**, 177-185.

[9] Lageweg BJ, Lenstra JK and Rinnooy Kan AHG (1976), Minimizing maximum lateness on one machine: computational experience and some applications, *Statistica Neerlandica*, **30**, 25-41.

[10] Lenstra JK, Rinnooy Kan AHG and Brucker P (1977), Complexity of machine scheduling problems, *Annals of Discrete Mathematics*, **1**, 343-362.

[11] Lourenço HR (1993), *A computational study of job-shop and the flow-shop scheduling problems*, Ph.D. Thesis, School of OR&IE, Cornell University, Ithaca, NY, USA.