

Single Machine and Parallel Machine Scheduling Problems with a Common Due Date to Minimize Total Weighted Tardiness

Nguyen Huynh Tuong, Ameer Soukhal, Jean-Charles Billaut

Laboratoire d'Informatique, Université François Rabelais Tours, France, nguyen.huynh@etu.univ-tours.fr,
{ameur.soukhal,jean.billaut}@univ-tours.fr

This paper deals with a common due date parallel machines scheduling problem in which each job has a different tardiness penalty. The objective is to minimize the total weighted tardiness. The scheduling problem of minimizing the total weighted tardiness with a common due date on a single machine is known to be ordinary NP-hard. This is also the case for problem $Pm|d_i = d|\sum w_i T_i$. A new dynamic programming algorithm is proposed to solve the $1|d_i = d|\sum w_i T_i$ scheduling problem and a fully polynomial time approximation scheme (FPTAS) is deduced from this algorithm. We show that the complexity of this FPTAS is better than existing one. These results are generalized to the parallel machines scheduling case.

Keywords: Multi-processor Scheduling; Weighted Tardiness; Dynamic Programming Algorithm; Approximation Scheme.

1 Introduction

Lawler and Moore [5] are the first ones who paid attention to tardiness scheduling problems for the last 40 years. In this paper, we consider the scheduling problem where n jobs J_1, \dots, J_n have to be scheduled without preemption on m identical parallel machines ($m \geq 1$). Each job is described by a processing time p_i , a positive integer tardiness penalty w_i , and a common due date $d_i = d, d \geq 0$. All jobs are available at time zero. We denote by S_i the starting time of J_i and by C_i its completion time.

As defined in [2], a job J_i can belong to one of the following three states:

1. early job if $C_i < d$,
2. fully-tardy job if $S_i > d$,
3. straddling job if $S_i < d$ and $C_i \geq d$.

The tardiness of J_i , denoted by T_i is defined by $T_i = \max(0, C_i - d)$. So, if J_i is completed before the due date ($C_i < d$) there is no penalty. Otherwise, there is a job-dependent tardiness penalty given by $w_i \times T_i$. The objective is to minimize the sum of tardiness penalties ($\sum_{i=1}^n w_i T_i$). According to the standard scheduling notation, the considered problem is denoted by $Pm|d_i = d|\sum w_i T_i, m \geq 1$.

If $m = 1$, the problem $1|d_i = d|\sum w_i T_i$ is proved to be NP-hard in the ordinary sense [8]. In [5], the authors propose a dynamic programming algorithm with complexity $O(n^2 d)$. Kolliopoulos and Steiner [3] determine a fully polynomial-time approximation scheme (FPTAS) for the problem $1|d_i = d|\sum w_i (T_i + d)$. So, even an optimal sequence for $1|d_i = d|\sum w_i (T_i + d)$ is also optimal in the case of minimizing $\sum w_i T_i$ but an ϵ -approximation for the first cannot guarantee the same approximation scheme for the second.

For problem $1|d_i = d|\sum w_i T_i$, Kellerer and Strusewich [2] determine a FPTAS with complexity $O((n^6 \log W)/\epsilon^3)$ (W is the sum of weights).

In the case of m identical parallel machines, the problem $Pm|d_i = d|\sum w_i T_i$, for a fixed m ($m \geq 2$), is NP-Hard and does not accept a polynomial ρ -approximation algorithms. In fact, the special case $Pm|d_i = d|\sum T_i$ where $w_i = 1$, ($\forall i, i = 1, \dots, n$), is NP-Hard [1] and does not accept any polynomial ϵ -approximation algorithms with $\epsilon < \infty$ unless $P = NP$ [4]. However, in [4] the authors propose two approximation algorithms with guaranteed performances. They show that the value of the solution given by one of the two approximation algorithms, denoted by X^0 , satisfies the following inequality $(X^0 - X^*)/(X^* + d) \leq \epsilon$ (X^* corresponds to optimal solution value).

In this study, we propose both dynamic programming algorithms and FPTAS for the single machine and parallel machine scheduling problem. However, the FPTAS for m parallel machines is valid under condition $P > md$ where P is the total processing time.

This paper is organized as follows. Section 2 is dedicated to the single machine scheduling problem ($m = 1$). In this section, we describe the new proposed dynamic programming algorithm with complexity $O(n^2d)$. Then, we will show how to transform this algorithm to a FPTAS, which can be runned in $O(h(W, P, d) \times n^3/\epsilon)$ time where W is the sum of weights, and $h(W, P, d)$ is a polynomial function of W , P and d . In Section 3, the DP algorithm and the approximation algorithm are extended to the parallel machine scheduling problem.

2 Single machine scheduling problem

This section deals with problem $1|d_i = d|\sum w_i T_i$. We remark that if $d = 0$, the problem is equivalent to $1||\sum w_i C_i$ that can be solved in $O(n \log n)$. In the case where $d \neq 0$ we propose an optimal pseudo-polynomial time algorithm with a complexity $O(n^2d)$.

The two following properties are well known [5] or trivial:

Property 1 : There always exist an optimal schedule where:

- there is no idle times between jobs,
- the early jobs are scheduled in an arbitrary order,
- the fully-tardy jobs are scheduled in a non-decreasing order of p_i/w_i .

Property 2: There is an optimal solution where the early jobs are scheduled according to a non-decreasing order of p_i/w_i .

To find an optimal solution we have to determine the straddling job and the set of early jobs. Finding the early job set can be viewed as solving a knapsack problem, where the objective is to maximize the penalty cost. So, we propose a dynamic programming algorithm where an optimal solution is calculated according to the following three steps.

1. at iteration k , let J_k be the straddling job.
2. we consider an initial solution where all jobs except J_k are scheduled after date d , i.e. there is no early job. To minimize the sum of weighted tardiness penalty of this initial solution, the jobs are scheduled according to the non-decreasing order of p_i/w_i , called WSPT.
3. we determine an optimal set of early jobs of total length strictly less than d .

It means that we have n choices for the straddling job. To illustrate the DP algorithm, let us study the following example. At the k th iteration, the job J_k is considered as the straddling job.

Let J_{i_1} , J_{i_2} and J_{i_3} be the only three jobs that can be early (the processing time of the other jobs exceeds d) (see Fig. 1). Let $Z^{(k)}(i)$ be the cost of job J_i in the initial solution and $F^{(k)}(i)$ be the reduction cost that corresponds to moving J_i to the early set. Let $Z^{(k)}$ be the total cost that we have to maximize at iteration k . The final cost is denoted by $F^{(k)}(n)$.

In the initial solution, let $n1$ be the number of jobs scheduled between J_k and J_{i_1} , $n2$ be the number of jobs between J_{i_1} and J_{i_2} , $n3$ be the number of jobs between J_{i_2} and J_{i_3} , $n4$ be the number of jobs after J_{i_3} . We have $n = n1 + n2 + n3 + n4 + 4$.

Let $W_{n1}^{(k)}$ be the sum of the weights of the $n1$ jobs, $W_{n2}^{(k)}$ be the sum of the weights of the $n2$ jobs, $W_{n3}^{(k)}$ be the sum of the weights of the $n3$ jobs, etc.

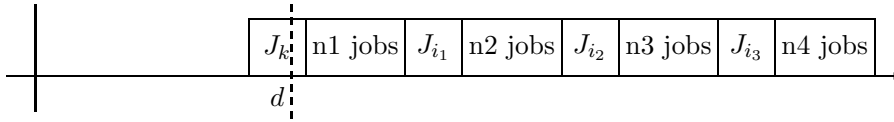


Figure 1: example - first step

First, if J_{i_1} is early, it means that this job is scheduled without tardy penalty (see Fig. 2) and the reduction cost is given by the following formula:

$$F^{(k)}(J_{i_1}) = Z^{(k)}(J_{i_1}) + p_{J_{i_1}} \times (W_{n2}^{(k)} + w_{J_{i_2}} + W_{n3}^{(k)} + w_{J_{i_3}} + W_{n4}^{(k)})$$

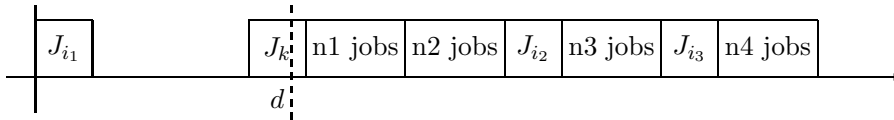


Figure 2: example - second step

By moving J_{i_2} in the early job set (see Fig. 3), we have:

$$F^{(k)}(J_{i_2}) = F^{(k)}(J_{i_1}) + Z^{(k)}(J_{i_2}) + p_{J_{i_2}} \cdot (W_{n3}^{(k)} + w_{J_{i_3}} + W_{n4}^{(k)}) - p_{J_{i_1}} \cdot w_{J_{i_2}}$$

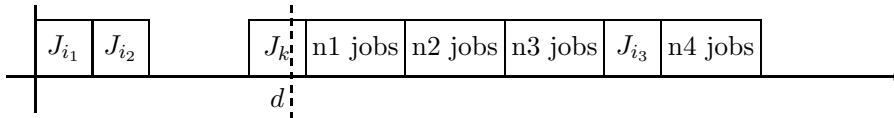


Figure 3: example - third step

Finally, $F^{(k)}(J_{i_3}) = F^{(k)}(J_{i_2}) + Z^{(k)}(J_{i_3}) + p_{J_{i_3}} W_{n4}^{(k)} - w_{J_{i_3}} (p_{J_{i_1}} + p_{J_{i_2}})$ (see Fig. 4).

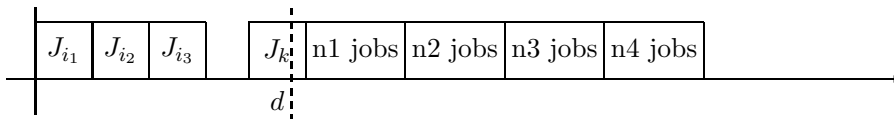


Figure 4: example - last step

To solve the problem, we propose the following algorithm, noted WTSMCDD (**W**eighted **T**ardiness scheduling problem in a **S**ingle **M**achine with **C**ommon **D**ue **D**ate).

For each $k, k = 1, \dots, n$, we have $(n-1) \times d$ iterations. Consequently, the algorithm complexity is $O(n^2d)$. This algorithm gives a global optimal solution since we can determine a local optimal solution for each sub-problem corresponding to the straddling job k . Of course, at each iteration k and at any time $0 \leq t < d$, $F^{(k)}(n, t)$ defines a sub-set of early jobs, sequenced according to the WSPT rule. The reduction given by this sub-set corresponds to the large reduction of the total cost of solution k . So, we can compare all the possible cases for $S_k = t$ where $d - p_k \leq S_k < d$. This comparison can be done according to the following formula:

$$Z^{(k)} - F^{(k)}(n, t) + (S_k + p_k - d)w_k + (S_k + p_k - d)W'^{(k)}(n, t) = Z^{(k)} - F^{(k)}(n, t) + (t + p_k - d)(W'^{(k)}(n, t) + w_k)$$

where $Z^{(k)} - F^{(k)}(n, t)$ is the weighted tardiness of the late job; $(S_k + p_k - d)w_k$ is the weighted tardiness of the straddling job J_k , and $(S_k + p_k - d)(W'^{(k)}(n, t))$ is the weighted tardiness defined by the late job set's movement at the time $(S_k + p_k - d)$.

Let's remind that in Lawler's solution, the early jobs are scheduled according to WLPT rule (**W**eighted **L**ongest **P**rocessing **T**ime). However, in our solution the early jobs are scheduled according to the WSPT rule. This DP algorithm allows finding a FPTAS.

In the following section, we propose a FPTAS based on the dynamic programming algorithm. In [2], the authors propose an FPTAS with complexity $O((n^6 \log W)/\epsilon^3)$. Even the complexity of our DP algorithm have the same one to the DP of Lawler and Moore [5], the FPTAS we propose has a complexity in $O(Wn^3/\epsilon)$.

Algorithm 1 WTSMCDD: $1|d_i = d|\sum w_i T_i$

```

1: // The jobs are supposed to be sorted according to WSPT //
2: for k = 1 to n do
3:   // Let J_k be the straddling job //
4:   Z(k)(i) = wiTi; Z(k) = ∑(i=1...n, i≠k) Z(k)(i)
5:   F(k)(0, 0) = 0 ; F(k)(0, t) = -∞, if t ≠ 0
6:   W(k)(0) = ∑j=1n(wj) - wk ; W'(k)(0, t) = W(k)(0)
7:   for i = 1 to n, i ≠ k do
8:     Z(k)(i) = wiTi; W(k)(i) = W(k)(i - 1) - wi
9:     for t = 0 to d - 1 do
10:      F(k)(i, t) = max(F(k)(i - 1, t), F(k)(i - 1, t - pi) + Z(k)(i) + piW(k)(i) - wi(t - pi))
11:      if (F(k)(i, t) = F(k)(i - 1, t)) then
12:        | W'(k)(i, t) = W'(k)(i - 1, t)
13:      else
14:        | W'(k)(i, t) = W'(k)(i - 1, t - pi) - wi
15:      endif
16:    endfor
17:  endfor
18:  S(k) = mint=d-pkd-1 (Z(k) - F(k)(n, t) + (t + pk - d)(W'(k)(n, t) + wk)
19: endfor
20: The optimal schedule corresponds to the sequence minimizing S(k).

```

Figure 5: Algorithm WTSMCDD

2.1 Lower bound

To prove that the considered problem accepts a FPTAS, we first calculate a lower bound. Let LB be the lower bound. We have $LB = w_{min} \times (P - d)$, where $w_{min} = \min_{i=1, \dots, n} \{w_i\}$ and $P = \sum_{i=1, \dots, n} p_i$ (the total processing time).

It is easy to show that LB is a lower bound. Of course, if $P \leq d$ we have $LB \leq 0 = \sum(w_i T_i)$. Otherwise, there exists at least one late job (a straddling job). The last job in a given sequence must be penalized by $w_{[n]}(P - d) \geq w_{min}(P - d)$. This penalty is less than or equal to the total weighted tardiness of the sequence.

2.2 Approximation algorithm

For a feasible schedule π , let $T_i(\pi)$ ($C_i(\pi)$, $S_i(\pi)$) be the tardiness of job J_i (the completion time of job J_i , and the starting time of job J_i) in sequence π ; let $T(\pi)$ ($X(\pi)$) be the total tardiness cost (the total weighted tardiness cost) of sequence π .

Following the idea presented in [7] to determine a FPTAS for knapsack problem, we define a new instance as follows:

- Given $\epsilon > 0$, let $W = \sum_{i=1}^n (w_i)$ and

$$K = \frac{\epsilon LB}{Wn} = \frac{\epsilon w_{min}(P - d)}{Wn} \quad (1)$$

- For each job J_i , $i = 1, \dots, n$, we define a new processing time as follows: $p'_i = \lfloor \frac{p_i}{K} \rfloor$.
- Let $d' = \frac{d}{K}$.
- Let $X'(\pi_A)$ be the total weighted tardiness cost that corresponds to the optimal solution π_A given by the previous algorithm WTSMCDD applied on the new processing times p'_i .

Let π^* be the optimal schedule and let $X(\pi^*)$ be its value. We have

$$X'(\pi_A) \leq X'(\pi^*) \quad (2)$$

and

$$X(\pi_A) \geq X(\pi^*) \quad (3)$$

For any job J_i , we have:

$$Kp'_i \leq p_i \quad (4)$$

and

$$p_i \leq K(p'_i + 1) \quad (5)$$

From (4), we have: $\Rightarrow K \sum_{j=1}^n (p'_j) \leq \sum_{j=1}^n (p_j)$

Therefore, since there is no iddle time between jobs, for the sequence π , we have:

$$\begin{aligned} \Rightarrow K C'_i(\pi) &\leq C_i(\pi) \\ \Rightarrow K C'_i(\pi) - d &\leq C_i(\pi) - d \\ \Rightarrow K T'_i(\pi) &\leq T_i(\pi) \\ \Rightarrow K w_i T'_i(\pi) &\leq w_i T_i(\pi) \\ \Rightarrow K X'_i(\pi) &\leq X_i(\pi) \quad (6) \end{aligned}$$

From (5) : $\Rightarrow \sum_{j=1}^n (p_j) \leq K \sum_{j=1}^n (p'_j + 1)$

$$\Rightarrow \sum_{j=1}^n (p_j) \leq Kn + K \sum_{j=1}^n (p'_j)$$

$$\begin{aligned} &\Rightarrow C_i(\pi) \leq Kn + KC'_i(\pi) \\ &\Rightarrow C_i(\pi) - d \leq Kn + KC'_i(\pi) - Kd' \\ &\Rightarrow T_i(\pi) \leq Kn + KT'_i(\pi) \\ &\Rightarrow w_i T_i(\pi) \leq w_i Kn + w_i KT'_i(\pi) \\ &\Rightarrow X(\pi) \leq WKn + KX'(\pi) \end{aligned}$$

Precisely, for the sequence π_A we have: $\Rightarrow X(\pi_A) \leq KX'(\pi_A) + WKn$

From (2) $\Rightarrow X(\pi_A) \leq KX'(\pi^*) + WKn$

From (6) $\Rightarrow X(\pi_A) \leq X(\pi^*) + WKn$

From (1) $\Rightarrow X(\pi_A) \leq X(\pi^*) + \epsilon w_{min}(P - d)$

$\Rightarrow X(\pi_A) \leq X(\pi^*) + \epsilon LB$

$\Rightarrow X(\pi_A) \leq (1 + \epsilon)X(\pi^*)$

Thus, the running time of the algorithm is $O(n^2 d') = O(n^2 d/K) = O(\frac{dW}{w_{min}(P-d)} \cdot \frac{n^3}{\epsilon})$ which is polynomial in n and $(1/\epsilon)$. So, we have a fully polynomial time approximation scheme.

Remark: If $d \leq \frac{Pw_{min}}{w_{min}+1}$, the complexity becomes more simple and corresponds to $O(\frac{Wn^3}{\epsilon})$.

3 Identical parallel machines scheduling problem

In this section, the scheduling problem $Pm|d_i = d|\sum w_i T_i$ with fixed m is studied. It is easy to show that there exists an optimal solution with at least one job scheduled on each machine ($n \geq m$). Consequently, we have the following property:

Property 3 : There exists an optimal solution with exactly m straddling jobs.

Proof: Let π^* be an optimal sequence. In π^* we suppose that there is not straddling job on a machine M_j . Let J_k be the last job from the early job set scheduled on M_j in π^* (i.e. $C_k < d$). Then we can move the job J_k to the right until $C_k = d$ without increasing the tardiness penalty. By definition J_k becomes the j th straddling job.

So, it is possible to construct an optimal solution with idle times between early job set and the straddling job without increasing the total tardiness penalties.

Let's consider the case $m = 2$. As in the case of a single machine, we assume that the jobs are sorted according to WSPT rule. Firstly we choose two straddling jobs and the remaining jobs are executed on the first machine after the completion time of the straddling job assigned to the first machine. It means that all jobs are late. Then, we try to assign each job to one machine in order to be scheduled in the time interval $[0, d]$. If it is not possible, the current job is assigned to the second machine and scheduled after the completion date of the straddling job in the second machine. So the complexity of this algorithm is $O(n^3 d^2 p_{max}^2 P)$ where $p_{max} = \max_{p_i=1}^n \{p_i\}$ and $P = \sum_{p_i=1}^n \{p_i\}$ (see Algorithm 6).

Hence, following the same idea and for any fixed $m \geq 2$ we can determine a DP algorithm with a complexity $O(n^{m+1} d^m p_{max}^m P^{m-1})$, which is pseudo-polynomial.

In [4] the authors show that there is no polynomial ϵ -approximation algorithm for problem $Pm|d_i = d|\sum_{i=1}^n T_i$ with $\epsilon < \infty$ unless $P = NP$.

However, if $P > md$ then we can show that the considered problem accepts a FPTAS. It is obtained by following the same idea of the FPTAS proposed in the case of a single machine scheduling problem. However, K is given by: $K = \frac{\epsilon LB_m}{Wn}$, where LB_m is a lower bound given by $LB_m = w_{min}Z$ where Z corresponds to the optimal value of $Pm||C_{max}$.

Algorithm 2 WTP2CDD: $P2|d_i = d|\sum w_i T_i$

```

1:  Order the jobs according to increasing ratios  $p_i/w_i$ 
2:  for  $k \in \{1 \dots n\}$  and  $l \in \{1 \dots n\}$  and  $k \neq l$  do
3:      /*Let  $J_k, J_l$  be the straddling jobs in the first and second machine, respectively.*/
4:      for  $S_k \in \{d - p_k, \dots, d - 1\}$  and  $S_l \in \{d - p_l, \dots, d - 1\}$  do
5:          Calculate the tardiness cost  $T^{(k,l,S_k,S_l)}(i)$   $i \notin \{k, l\}$  such that  $J_i$  begins after  $C_k$ 
6:           $Z^{(k,l,S_k,S_l)}(i) = w_i T^{(k,l,S_k,S_l)}(i)$ 
7:           $Z^{(k,l,S_k,S_l)} = \sum_{(i=1, i \neq k, l)}^n Z^{(k,l,S_k,S_l)}(i)$ 
8:           $F^{(k,l,S_k,S_l)}(0, 0, 0, S_l + p_l - d) = 0$ 
9:           $F^{(k,l,S_k,S_l)}(0, t_1, t_2, t_3) = -\infty$ , if  $t_1 \neq 0$ ,  $t_2 \neq 0$  and  $t_3 \neq (S_l + p_l - d)$ 
10:          $W^{(k,l,S_k,S_l)}(0) = \sum_{j=1}^n (w_j) - w_k - w_l$ 
11:          $W'^{(k,l,S_k,S_l)}(0, t) = W^{(k,l,S_k,S_l)}(0)$ 
12:         for  $i \in \{1, \dots, n\}, i \neq k, i \neq l$  do
13:              $Z^{(k,l,S_k,S_l)}(i) = w_i T_i$ 
14:              $W^{(k,l,S_k,S_l)}(i) = W^{(k,l,S_k,S_l)}(i - 1) - w_i$ 
15:             for  $t_1 \in \{0, \dots, S_k\}$  and  $t_2 \in \{0, \dots, S_l\}$  and  $t_3 \in \{0, \dots, P\}$  do
16:                  $m_0 = F^{(k,l,S_k,S_l)}(i - 1, t_1, t_2, t_3)$ 
17:                  $m_1 = F^{(k,l,S_k,S_l)}(i - 1, t_1 - p_i, t_2, t_3) + Z^{(k,l,S_k,S_l)}(i) + p_i W^{(k,l)}(i) - w_i(t_1 + t_2 + t_3 - p_i)$ 
18:                  $m_2 = F^{(k,l,S_k,S_l)}(i - 1, t_1, t_2 - p_i, t_3) + Z^{(k,l,S_k,S_l)}(i) + p_i W^{(k,l)}(i) - w_i(t_1 + t_2 + t_3 - p_i)$ 
19:                  $m_3 = F^{(k,l,S_k,S_l)}(i - 1, t_1, t_2, t_3 - p_i) + Z^{(k,l,S_k,S_l)}(i) + p_i W^{(k,l)}(i) - w_i(t_1 + t_2 + t_3 - p_i) - w_i t_3$ 
20:                  $F^{(k,l,S_k,S_l)}(i, t_1, t_2, t_3) = \max\{m_0, m_1, m_2, m_3\}$ 
21:             endfor
22:         endfor
23:          $S^{(k,l)}(S_k, S_l) = \min_{t_1=0}^{S_k} \min_{t_2=0}^{S_l} \min_{t_3=0}^P \left( Z^{(k,l,S_k,S_l)} - F^{(k,l,S_k,S_l)}(n, t_1, t_2, t_3) \right)$ 
24:          $S(k, l) = \min_{S_k=d-p_k \dots (d-1), S_l=d-p_l \dots (d-1)} S^{(k,l)}(S_k, S_l)$ 
25:         endfor
26:     endfor
27:     The optimal schedule corresponds to the sequence defined by the straddling jobs  $J_k$  and  $J_l$  in which  $S$  is minimized.

```

Figure 6: Algorithm WTP2CDD

So, $LB_m \leq w_{\min}(P - md) > 0$ since $P > md$. Hence, the FPTAS's complexity is bounded by $O(H \frac{n^{3m}}{\epsilon^{2m-1}})$ with $H = d^m p_{\max}^m P^{m-1} \left(\frac{W}{w_{\min}(P - md)} \right)^{2m-1}$.

In the case where $d = 0$, the DP algorithm can solve the $Pm || \sum w_i C_i$ problem.

4 Conclusion

This paper deals with single machine and identical parallel machines scheduling problems with a common due date. The objective is to minimize the total weighted tardiness penalties. A new

DP algorithm is developed for the single machine scheduling problem with a complexity identical to the complexity of Lawler's DP. The DP allows defining a FPTAS with complexity less than the best one of the literature. We also show how to transform these results in order to solve the case with m identical parallel machines.

References

- [1] M.R. Garey, D.J. Johnson (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco.
- [2] H. Kellerer, Vitaly A. Strusevich (2006), A fully polynomial approximation scheme for the single machine weighted total tardiness problem with a common due date, *Theoretical Computer Science* **369**, 230 – 238.
- [3] S.G. Kolliopoulos, G.Steiner (2005), Approximation algorithms for scheduling problems with a modified total weighted tardiness objective, *Computing and Software Technical Reports 2004-2005*.
- [4] M. Y. Kovalyov, Frank Werner (2002), Approximation Schemes for Scheduling Jobs with Common Due Date on Parallel Machines to Minimize Total Tardiness, *Journal of Heuristics* **8**, 415 – 428.
- [5] E.L. Lawler, J.M. Moore (1969), A functional equation and its application to resource allocation and sequencing problems, *Management Science* **16**, 77 – 84.
- [6] J.K. Lenstra, A.H.G. Rinnooy Kan (1978), Complexity of scheduling under precedence constraints, *Operations Research* **26**, 22 – 35.
- [7] V. V. Vazirani (2003), *Approximation Algorithms*, Springer-Verlag, Berlin, Heidelberg.
- [8] J. Yuan (1992), The NP-hardness of the single machine common due date weighted tardiness problem, *Systems Science and Mathematical Sciences* **5**, 328 – 333.