Scheduling problem with uncertain parameters

by

Wojciech Bożejko[1,3], Paweł Rajba[2], Mieczysław Wodecki[2,3]

[1] Wrocław University of Technology, Poland,
[2] University of Wrocław, Poland,
[3] College of Management "Edukacja" Wrocław, Poland

Corresponding author:
Mieczysław Wodecki
Institute of Computer Science, University of Wrocław
ul. Joliot-Curie 15, 50-383 Wrocław, Poland
phone: (+48 71) 375 78 00
e-mail: mwd@ii.uni.wroc.pl

ABSTRACT

In the paper we consider a *strong NP-hard* single-machine scheduling problem with deadlines and minimizing the total weight of late jobs on a single machine. Processing times are deterministic values or random variables having normal distributions. For this problem we study the tolerance to random parameter changes for solutions constructed according to tabu search metaheuristics. We also present a measure (called stability) that allows an evaluation of the algorithm based on its resistance to random parameter changes.

Keywords: scheduling, weight tardiness, normal distribution, tabu search, stability.

1. INTRODUCTION

In many applications serious difficulties occur while indicating parameters or when the data comes from inaccurate measurement equipment. Due to short realization terms, short series and production elasticity there are no comparative data and no possibility to conduct experimental studies that would enable one to determine explicit values of certain parameters. Furthermore, in many economy branches like tourism, agriculture, commerce, building industry, etc., the processes that occur have by their nature random character (they depend on weather, market conditions, accidents, etc.). Making decisions in the conditions of uncertainty (lack of exact values of parameters) becomes quotidian.

Poblems of taking decisions under uncertainty are solved by application of probabilistic method or through fuzzy sets theory. In the first case (Dean [3], Vondrák [18]) knowledge of

distribution of random variables is of crucial importance. Some pocesses are characterised with randomity by nature. They depend on weather conditions, traffic intensity, number of accidents, geological conditions, device's failure, etc. If they, nevertheless, posses certain "history", it is possible to define their distribution on the basis of statistical data.

In many issues the uncertainty of data is not of random nature but it results from uniqueness of a process, errorr in measurement, etc. In such a case a natural method of representing uncertainty are fuzzy numbers (Iscibuchi et all. [6], Ishii [7]). In this case a huge problem is posed by a proper choice of membership function and deffuzification method. They have crucial influence on the quality of taken solutions.

In this paper we examine a scheduling problem on a single machine with the latest possible processing times and cost minimizing for the belated jobs. The delays needed to accomplish the jobs are deterministic or random variables with normal distribution. In this case we study the resistance to random parameter changes on solutions constructed according to the tabu search metaheuristics. We also present a certain measure (called stability) that allows one to evaluate the resistance of solutions to random data perturbations.

2. PROBLEM DEFINITION

In this paper we consider the scheduling problem on a single machine. The machine can perform only one job at a time. For job i $(i=1,...,n)$, let p_i, w_i, d_i be: the processing time, a weight function of costs and the deadline expected. If for a given sequencing the deadline of job i exceeds d_i, the delay U_i is 1, if not, U_i is 0. The problem of minimizing the total weight of late jobs on a single machine (TWLJ) consists in finding a job sequence that minimizes the sum of delay costs, i.e. $\sum_{i=1}^{n} w_i U_i$. The problem can be written as $1||\sum w_i U_i$, and though its formula is so simple, it is *NP-hard* (Karp [8]). Such problems have been studied for quite long together with many variations, especially with polynomial computational complexity.

For the problem $1|p_i=1|\sum w_i U_i$ (all the processing times are identical) Monma [13] has presented an algorithm with $O(n)$ complexity. Similarly, for the problem $1|w_i=c|\sum U_i$, (where the cost function factors are identical) there is the Moore algorithm [14] with $O(n\ln n)$ complexity. Lawler [11] has adapted the Moore algorithm to solve the problem $1|p_i<p_j \Rightarrow w_i \geq w_j|\sum w_i U_i$. Problems with the earliest starting times compose another

group r_i. Kise et al. [9] have proven that even the problem of late jobs minimization ($1|r_i|\sum U_i$ without the cost function weight) is *strongly NP-hard*. They have also presented a polynomial algorithm that has computational complexity $O(n^2)$ for a particular example, the $1|r_i < r_j \Rightarrow d_i \leq d_j|\sum U_i$ problem. If a partial order relation is given on the set of jobs, the TWLJ problem is *strongly NP-hard* even when the job realization times are unities. Lenstra and Rinnoy Kan [12] have proven that if a partial order relation is a union of independent chains, the problem is also *strongly NP-hard*.

There have been only a few exact algorithms solving the TWLJ problem published. They are based on the dynamic programming method (Lawler and Moore [10] – with $O(n\min\{\sum p_i, \max\{d_i\}\})$ complexity and Sahni [17] – with $O(n\min\{\sum p_i, \sum w_i,\ \max\{d_i\}\}))$ complexity and on a limitation and division method (Villarreal and Bulfin [18], Potts [15], Potts and Van Wassenhowe [14], Bożejko, Grabowski and Wodecki [1], Bożejko and Wodecki [2] and Wodecki [20]). The last one is a parallel algorithm.

The scheduling problem on a single machine can be formulated as follows:

The problem: There is a set $J = \{1, 2, ..., n\}$ of jobs that have to be processed without interruptions on a machine that can work on one job at a time. The job can start at time zero. For job $i \in J$ let p_i be the processing time, d_i the expected deadline, and w_i costs function weight. We want to determine a job sequence that minimizes the weight of late jobs.

For a given sequence let C_i be the date of accomplishing of job $i \in J$. Then $f_i(C_i) = w_i U_i$ is the cost (penalty) of a late job, where

$$U_{\pi(i)} = \begin{cases} 0, & \text{if } C_i \leq d_i, \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

Let Φ be the set of permutations of J. The cost of the permutation $\pi \in \Phi$ is a defined as follows:

$$W(\pi) = \sum_{i=1}^{n} w_{\pi(i)} U_{\pi(i)}, \tag{2}$$

where $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(i)}$ is the processing time of the job $\pi(i) \in J$. The problem of minimizing the total weight of late jobs (TWLJ) boils down to finding an optimal permutation $\pi^* \in \Phi$ which satisfies

$$W(\pi^*) = \min_{\pi \in \Phi} \left(\sum_{k=1}^{n} w_{\pi(i)} U_{\pi(i)} \right).$$

Exact efficient algorithms to solve the TWLJ problem only exist when the number of jobs does not exceed 50 (80 in a multiprocessor environment Wodecki [20]). That is why in practice we use approximate algorithms (essentially the correction type).

3. TABU SEARCH METHOD

In solving *NP-hard* problems of discrete optimization we almost always use approximate algorithms. The solutions given by these algorithms are satisfactory applications (they often differ from the best known solutions by less than 1al search methods.

The tabu search method (TS - proposed by Glover [4] and [5]) is a metaheuristic approach designed to find a near-optimal solution of combinatorial optimization problems. The basic version of TS starts from an initial solution x^0. The elementary step of the method performs, for a given solution x^i, a search through the neighborhood $N(x^i)$ of x^i. The neighborhood $N(x^i)$ is define by move (transitions) performed from x^i. A move transforms a solution into another solution. The aim of those elementary search is to find in $N(x^i)$ a solution x^{i+1} with the lowest cost functions. Then the search repeats from the best found, as a new starting solution, and the process is continued. In order to avoid cycling, becoming trapped to a local optimum, and more general to conduct the search in “good regions” of the solution space, a memory of the search history is introduced. Among many classes of the memory introduced for tabu search (see. Glover [4]), the most frequently used is the short term memory, called the tabu list. This list recorded, for a chosen span of time, solutions or selected attributes of these solutions (or moves).The search stops when a given number of iterations or current neighborhood is empty

In this section we present some properties which are the base of a new neighborhood’s construction and, further, very fast tabu search algorithm.

3.1. CLASSIC TABU SEARCH ALGORITHM

In solving *NP-hard* problems of discrete optimization we almost always use approximate algorithms. The solutions given by these algorithms are, in their appliance, fully satisfying (they often differ from the best known solutions by less then 1%). Most of them belong to the local search methods group. Their acting consists in viewing in sequence a subset of a set of

acceptable solutions, and in pointing out the best one according to a determined criterion. One of this method realizations is the tabu search, whose basic criterions are:

- *neighborhood* - a subset of a set of acceptable solutions, whose elements are rigorously analyzed,
- *move* - a function that converts one solution into another one,
- *tabu list* - a list containing the attributes of a certain number of solutions analyzed recently,
- *ending condition* - most of the time fixed by the number of algorithm iterations.

Let $\pi \in \Phi$ be any (starting) permutation, L_{TS} a tabu list, W costs function, and π^* the best solution found at this moment (the starting solution and π^* can be any permutation).

Algorithm Tabu Search (TS)

repeat

- Determinate the neighborhood $N(\pi)$ of the permutation π;
- Remove from $N(\pi)$ the permutations forbidden by the L_{TS} list;
- Determinate the permutation $\delta \in N(\pi)$, in which

$$W(\delta) = \min\{W(\beta) : \beta \in N(\pi)\};$$

- ***if*** $(W(\delta) < W(\pi^*))$ ***then***
- Include δ parameters on the L_{TS} list;
- $\pi := \delta$

until (*ending_condition*).

The computational complexity of the algorithm depends mostly on the way the neighborhood is generated and viewed. Below we present in details the basics elements of the algorithm.

3.2. THE MOVE AND THE NEIGHBORHOOD

Let $\pi = (\pi(1),\ldots,\pi(n))$ be any permutation from the Φ, and

$$L(\pi) = \{\pi(i) : C_{\pi(i),m} > d_{\pi(i),m}\},$$

a *set of late jobs* in π.

By π_l^k $(l=1,2,\ldots,k-1,k+1,\ldots,n)$ we mark a permutation received from π by changing in π the element $\pi(k)$ and $\pi(l)$. We can say at that point that the permutation π_l^k was generated from π by a *swap move* (*s-move*) s_l^k (it means that the permuta-tion $\pi_l^k = s_l^k(\pi)$). Then, let $M(\pi(k))$ be a set of all the *s*-moves of the $\pi(k)$ element. By

$$M(\pi) = \bigcup_{\pi(k)\in L(\pi)} M(\pi(k)),$$

we mean an *s-moves* set of the late elements π in the pemutation. The power of the set $M(\pi)$ is top-bounded by $n(n-1)/2$.

The neighborhood $\pi \in \Phi$ is the permutation set

$$N(\pi) = \left\{ s_l^k(\pi) : s_l^k \in M(\pi) \right\}.$$

While implementing the algorithm, we remove from the neighborhood the permutations whose attributes are on the forbidden attributes list L_{TS}.

3.3. TABU LIST

To prevent from arising cycle too quickly (returning to the same permutation after some small number of iterations of the algorithm), some attributes of each move are saved on so-called tabu list (list of the prohibited moves). This list is served as a FIFO queue, see Bożejko, Grabowski and Wodecki [1]. Making a move $i_j^r \in M(\pi)$, (that means generating permutation π_j^r from $\pi \in \Phi$) we save attributes of this move, triple $(\pi(r), j, F(\pi_j^r))$, on the tabu list. Let us assume that we consider a move $i_l^k \in M(\beta)$ which generates permutation β_l^k. If there is a triple (r, j, Ψ) such that $\beta(k) = r, l = j$ on the tabu list, and $F(\beta_l^k) \geq \Psi$, then such a move is eliminated (removed) from the set $M(\beta)$.

The dynamic length *lTS* of tabu list L_{TS} is a cyclic function defined by the expression:

$$lTS(iter) = \begin{cases} low, & \text{if } S(k) < iter \leq S(k)+h, \\ low+\alpha, & \text{if } S(k)+h < iter \leq S(k+1), \end{cases}$$

where: *iter* is the number of iteration of the algorithm, $k = 1, 2, \ldots$ is the number of the cycle. Integer number $\alpha > 0$, $S(k) = (k-1)(h+H)$, here $S(0) = 0$. *Low* is the standard length of the L_{TS} list (by h iterations of the algorithm) and H is the width of the pick equal $low+\alpha$.

If *lTS* decreases then a suitable number of the oldest elements of tabu list L_{TS} is deleted and the search process is continued. All parameters of length of the tabu list are empirical, based on preliminary experiments. Changing the tabu list's length causes diversification of the search process.

4. PROBABILISTICS JOBS TIMES

Let $\theta = \langle p, d, w \rangle$ be an example of deterministic data for TWLJ problem. We assume that times of execution of jobs $i \in J$ are independent random variables with a normal distribution, i.e. $\tilde{p}_i \sim N(p_i, \sigma_i)$. The expected value of times $E(\tilde{p}_i) = p_i$. Then data $\tilde{\theta} = \langle \tilde{p}, d, w \rangle$, where $\tilde{p} = [\tilde{p}_i]_{1,2,...,n}$ is a matrix of random variables, we call a probabilistic data, and the problem - probabilistic (TWLJP in short).

Let $\pi \in \Phi$ be some sequence of jobs execution at objects. In order to simplify the calculations we assume that moments of completion of separate works have also a normal distribution

$$\tilde{C}_{\pi(i)} \sim N(\sum_{j=1}^{i} m_j, \sqrt{\sum_{j=1}^{i} \sigma_j^2}).$$

The equivalent of delays (1) are random variable

$$\tilde{U}_{\pi(i)} = \begin{cases} 1, & if\ \tilde{C}_{\pi(i)} > d_{\pi(i)}, \\ 0, & if\ \tilde{C}_{\pi(i)} \le d_{\pi(i)}. \end{cases}$$

The mean of random variable $\tilde{U}_{\pi(i)}$, $i = 1, 2, ..., n$,

$$E(\tilde{U}_{\pi(i)}) = \sum_{x} x * P(\tilde{U}_{\pi(i)} = x) =$$

$$0 * P(\tilde{C}_{\pi(i)} \le d_{\pi(i)}) + 1 * P(\tilde{C}_{\pi(i)} > d_{\pi(i)}) = 1 - \Psi(\frac{d_{\pi(i)} - \sum_{j=1}^{i} m_{\pi(j)}}{\sum_{j=1}^{i} \sigma_{\pi(j)}^2}),$$

where Ψ is distribuant of random variable with a normal distribution $N(0,1)$.

By solving a TWLJ problem (with a random times of jobs execution) for a cost function (4) we assume

$$FP(\pi) = E(F(\pi)) = E(\sum_{i=1}^{n} w_{\pi(i)} * \tilde{U}_{\pi(i)}) =$$

$$\sum_{i=1}^{n} w_{\pi(i)} * E(\tilde{U}_{\pi(i)}) = \sum_{i=1}^{n} w_{\pi(i)} * (1 - \Psi(\frac{d_{\pi(i)} - \sum_{j=1}^{i} m_{\pi(j)}}{\sum_{j=1}^{i} \sigma_{\pi(j)}^2}). \quad (3)$$

Tabu search algorithm of solving TWLJP problem (with a goal function (3)) we call a probabilistic one, **TSP** in short.

5. SUSTAINABILITY OF ALGORITHMS

Sustainability is some property which enables estimating of the influence of data perturbation on changes of goal function values. We present a method of generating a set of instances as the fist priority.

Let $\delta = \langle p, d, w \rangle$, where: $p = [p_i]_{i=1,2,...,n}$, $d = [d_i]_{i=1,2,...,n}$ and $w = [w_i]_{i=1,2,...,n}$ are respectively, the matrix: of work execution times, completion times and penalty coefficient, will be some instances of (deterministic) data for TWLJ problem. By $D(\theta)$ we denote a set of data generated from θ through perturbation of time execution. This perturbation consists in changing of $p = [p_i]_{i=1,2,...,n}$, elements into randomly determined values (i.e. numbers generated in accordance with certain distribution, for instance monotonous, etc.). Any element of $D(\theta)$ set takes form of $\langle p', d, w \rangle$ where perturbed elements of matrix $p' = [p_i']_{i=1,2,...,n}$, are determined randomly. Thus, $D(\theta)$ set includes instances of deterministic data for TWLJ problem**,** different from one another only by values of jobs' execution times.

Let $\boldsymbol{A} = \{\boldsymbol{TS}, \boldsymbol{TSP}\}$, where ***TS*** and ***TSP*** algorithms are: deterministic, fuzzy and probabilistic respectively. By $\pi_\delta^{\boldsymbol{A}}$ we denote a solution (permutation) determined by $\boldsymbol{A}$ algorithm for δ data. The value of expression $W(\pi_\delta^A, \varphi)$ is cost (4) for an instance of deterministic φ data, when objects are executed in a sequence of (permutations) $\pi_\varphi^{\boldsymbol{A}}$ (i.e. in a sequence defined by ***A*** algorithm for δ) data. Then

$$\Delta(A, \delta, D(\delta)) = \frac{1}{|D(\delta)|} \sum_{\varphi \in D(\delta)} \frac{W(\pi_\delta^A, \varphi) - W(\pi_\varphi^{TS}, \varphi)}{W(\pi_\varphi^{TS}, \varphi)},$$

We call a sustainability of $\pi_\delta^{\boldsymbol{A}}$ solution determined by $\boldsymbol{A}$ algorithm on a set of $D(\delta)$ perturbed data. Determining $\pi_\varphi^{\boldsymbol{TS}}$, for a starting solution of ***TS*** algorithm $\pi_\varphi^{\boldsymbol{A}}$ was denoted and next

$$W(\pi_\delta^A, \varphi) - W(\pi_\varphi^{TS}, \varphi) \geq 0.$$

Thus, $\Delta(A, \delta, D(\delta)) \geq 0$. The value of expression $\Delta(A, \delta, D(\delta))$ is an average relative deviation of the best $\pi_\delta^{\boldsymbol{A}}$ solution for the best set solutions, for every instances of perturbed data $\varphi \in D(\delta)$.

Let Ω be some set of deterministic instances for TWLJ problem. *Sustainability* coefficient of ***A*** algorithm on a Ω set we define as follows:

$$S(A,\Omega)=\frac{1}{|\Omega|}\sum_{\delta\in\Omega}\Delta(A,\delta,D(\delta)). \tag{4}$$

The smaller the coefficient, the more sustainable the solutions set by ***A*** algorithm i.e. small changes in value of data cause small changes of goal function value.

6. NUMERICAL EXPERIMENTS

The algorithms presented in this paper have been tested on many examples. Deterministic data for one machine problem with delay cost minimization $1||\sum w_iT_i$ were generated in a randomized way (see [20]), and are available on the OR-Library. For a given number of n jobs ($n=40,50,100,250,500$) we have determined n triples (p_i, w_i, d_i), $i=1,...,n$, where the processing time p_i and the cost w_i are the realization of a random variable with a uniform distribution, respectively from the range [1, 100] and [1, 10]. Similarly, the critic lines are drawn from the range $[P(1-TF-RDD/2), P(1-TF+RDD/2)]$ depending on the parameters $RDD, TF = 0.2, 0.4, 0.6, 0.8, 1.0$, while $P=\sum_{i=1}^{n} p_i$. For every couple of parameters *RDD*, *TF* (there are 25 such couples) 5 examples have been generated. The whole deterministic data set Ω contains 525 examples (125 for every n).

For every deterministic data example (p_i, w_i, d_i), $i=1,...,n$, we have defined a probabilistic data example ($\tilde{p}, w_i, d_i$), $i=1,...,n$, where $\tilde{p}$ is a random variable with normal distribution representing the processing time (the exact description in Section 4). We denote the set of examples by Ω.

The deterministic ***TS*** and probabilistic ***TSP*** algorithms were started from identity permutation. Moreover, we have adopted the following parameters:

1. dynamic length of tabu list (*lTS*(*iter*):
   $h=\lceil n/4\rceil$, $H=\lceil n/10\rceil$, $low=\lceil\sqrt{n}\rceil$, $\alpha=\lceil\sqrt{n/4}\rceil$,
2. the maximum number of algorithm iterations: $n/2$ or n.

The deterministic algorithm ***TS*** has been performed on Ω, and the probabilistic algorithm ***TSP*** – on Ω. In order to evaluate the stability coefficient (4) of both algorithms, 100 examples of perturbed data have been generated for every deterministic data example from Ω (we have presented the way of generating these examples in Section 4). Then, all these

examples have been solved by the ***TS*** algorithm. Based on these calculations, we have determined the stability coefficient of both algorithms. The results are presented in Tables 1.

Table 1. Stability coefficient (relative average error $S(A,\Omega)$) for $n/2$ and n iterations.

Number of jobs n	Deterministic algorithm ***TS***		Probabilistic algorithm ***TSP***	
	$n/2$ iterations	n iterations	$n/2$ iterations	n iterations
40	0,094	0,111	0,033	0,039
50	0,118	0,139	0,042	0,051
100	0,289	0,303	0,046	0,057
250	0,403	0,362	0,053	0,069
500	0,415	0,487	0,076	0,094
avg.	**0,261**	**0,280**	**0,050**	**0,062**

The average stability coefficient ($n/2$ iterations) for the deterministic algorithm is, $S(TS,\Omega)=0,261$ and for the random algorithm $S(TSP,\Omega)=0,050$. This means that the perturbation of the solution determined by the ***TS*** algorithm causes a target function value deterioration of about 26%. In the ***TSP*** algorithm the deterioration is only about 5%. So the medium error for the deterministic algorithm is more than 5 times that for the probabilistic algorithm.

Table 1 contains too the results of a two times bigger for *n* iterations. The fact that the stability of both algorithms has slightly deteriorated is a little surprising. The stability difference is more advantageous for the random algorithm in $n/2$ iterations, even if this algorithm is still significantly more stable than the deterministic one. In this case the medium error of the ***TS*** algorithm is more than 10 times that for the ***TSP*** algorithm. Moreover, the data perturbation causes (for the solution determined by the probabilistic algorithm) a target function value deterioration of around 6,2%.

We have also made calculations for more iterations ($n\log n$, n^2). The stability coefficient for both algorithms have slightly deteriorated, as well as the stability difference between the deterministic and the probabilistic algorithm (even if the random algorithm maintains its superiority). The number of $n/2$ iterations in the tabu search method is very small (usually we make n^2 iterations). Based on the results obtained, we can say that in this

case it is not only sufficient, but even optimal. For this reason the medium calculation time for one example, on a personal computer with a 2,6 GHz Pentium processor is very short and does not exceed one second.

The experiments conducted have shown without doubt that solutions determined by the probabilistic algorithm are very stable. The perturbation (change) of the processing time causes a medium deterioration of a few percent (maximum about a 11%). From the point of view of its utility in practice, this is completely satisfactory.

7. CONCLUDING REMARKS

In this paper we have presented a method of modeling uncertain data using random variables with normal distribution. We have presented an algorithm based on the tabu search method in order to solve a certain scheduling problem on a single machine. For this problem, we have evaluated the solution stability, which means its resistance to random changes in job parameters. The experiments have shown that the algorithm in which the processing times are random variables with normal distribution, is very stable. The medium relative error for perturbed data does not exceed 5% when the iteration number is small, and the calculation time is short.

References

[1] Bożejko W., Grabowski J., Wodecki M., Block approach-tabu search algorithm for single machine total weighted tardiness problem, Computers & Industrial Engineering, Elsevier Science Ltd,. Volue 50. Issue1/2, 2006, 1-14.
[2] Bożejko W., Wodecki M., A parallel metaheuristics for the single machine total weighted tardiness problem with sequence-dependent setup times, Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications (MISTA 2007, Paris), 96-103.
[3] Dean B.C., Approximation algorithms for stochastic scheduling problems, PhD thesis, MIT, 2005.
[4] Glover F., Tabu search . Part I.ORSA Journal on Computing, 1, 1989, 190-206.
[5] Glover F., Tabu search. Part II.ORSA Journal on Computing, 2, 1990, 4-32.
[6] Ishibuschi H., Murata T., Scheduling with Fuzzy Duedate and Fuzzy Processing Time, in: Scheduling Under Fuzziness, R.Słowiński and M.Hapke (eds), Springer-Verlag, 2000, 113–143.
[7] Ishii H., Fuzzy combinatorial optimization, Japanese Journal of Fuzzy Theory and Systems, Vol. 4, no. 1, 1992, 31–40.
[8] Karp R.M., Reducibility among Combinatorial Problems, Complexity of Computations, R.E. Millerand and J.W. Thatcher (Eds.), Plenum Press, New York, 1972, 85-103.
[9] Kise H., Ibaraki T., Mine H., A solvable case of the one-machine scheduling problem with ready times and due times, Operations Research, 26, 1978, 121-126.
[10] Lawler E.L., Moore J.M., A Functional equation and its applications to resource allocation and sequencing problems, management science, 16, 1969, 77-84.
[11] Lawler E.L., A "pseudopolinomial" algorithm for sequencing jobs to minimize total

tardiness, Annals of Discrete Mathematics, 1, 1977, 331-342.
[12] Lenstra J.K., Rinnoy Kan A.H.G., Complexity results for scheduling chains on a single machine, European Journal of Operational Research, 4, 1980, 270-275.
[13] Monma C.I., Linear-time algorithms for scheduling on parallel processor, Operations Research, 30, 1982, 116-124.
[14] Moore J.M., An n-job, one machine sequencig algorithm for minimizing the number of late jobs, Menagement Science, 15, 1968, 102-109.
[15] Potts C.N., Van Wassenhove L.N., A Branch and Bound Algorithm for the Total Weighted Tardiness Problem, Operations Research, Vol. 33, 1985, 177-181.
[16] Potts C.N., Van Wassenhove L.N., Algorithms for Scheduling a Single Machine to Minimize the Weighted Number of Late Jobs, Management Science, 34, 7, 1988, 843-858.
[17] Sahni S.K., Algorithms for Scheduling Independent Jobs, J.Assoc. Comput. Match., 23, 1976, 116-127.
[18] Villareal F.J., Bulfin R.L., Scheduling a Single Machine to Minimize the Weighted Number of Tardy Jobs, IEE Trans., 15, 1983, 337-343.
[19] Vondrák J., Probabilistic methods in combinatorial and stochastic optimization, PhD, MIT, 2005.
[20] Wodecki M., A Branch-and-Bound Parallel Algorithm for Single-Machine Total Weighted Tardiness Problem, Advanced Manufacturing Technology, 37, 2007, 996-1004.