APPLICATIONES MATHEMATICAE
39,2 (2012), pp. 169–183

Paweł Rajba and Mieczysław Wodecki (Wrocław)

STABILITY OF SCHEDULING WITH RANDOM PROCESSING TIMES ON ONE MACHINE

Abstract. We consider a strong NP-hard single-machine scheduling problem with deadlines and minimizing the total weight of late jobs on a single machine ($1 \parallel \sum w_i U_i$). Processing times are deterministic values or random variables having Erlang 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. Our experiments prove that random model solutions are more stable than the deterministic model ones.

1. Introduction. Many problems related to the process of making decisions are indeed issues of some scheduling problems. The relevant research concentrates on deterministic models, in which the main assumption is the explicitness of all parameters. Many effective approximation algorithms have been elaborated to solve this type of problem, mostly NP-hard. They are principally based on local optimization methods: simulated annealing, tabu search and genetic algorithms. Solutions indicated by those algorithms are only slightly different from the optimal ones. However, in practice, while the process is running (according to the new schedule) it occurs frequently that some of the parameters (for example the times needed to complete the operations) differ from the initially adopted ones. If no possible solution for stability occurs, the determined schedule can be wrong and is not acceptable.

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

2010 *Mathematics Subject Classification*: Primary 90C59.
Key words and phrases: scheduling, weight tardiness, Erlang distribution, tabu search.

DOI: 10.4064/am39-2-5

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 random character by their nature (they depend on weather, market conditions, accidents, etc.). Making decisions in the conditions of uncertainty (lack of exact values of parameters) becomes quotidian.

Scheduling uncertain data problems can be solved by using methods of probability theory (Shaked and Shanthikumar [21], Zhou and Cai [25], Van den Akker and Hoogeveen [1], [2], Vondrák [23], Dean [3]) or fuzzy set theory (Prade [19], Ishii [6], Ishibuchi et al. [5], Itoh and Ishii [7]). They allow one to take the uncertainty into account at the stage of construction of a mathematical model and directly in the algorithms constructed. The exactness of such an algorithm is not determined according to a certain problem instance (as in the deterministic data case), but with a certain family of randomized examples, according to a certain probability distribution. We call the exactness determined in that way algorithm stability (or, more generally, problem solving stability).

In this paper we examine a scheduling problem on a single machine with the latest possible processing times and cost minimizing for the belated tasks. The delays needed to accomplish the tasks are deterministic or random variables with Erlang 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.

The paper is organized as follows: in Sect. 2 we review the problem of minimizing the number of late jobs with deterministic processing times. The main elements of the tabu search method are described in Sect. 3. The next two sections are the principal part of the paper. In Sect. 4 we study the scheduling problem with non-deterministic task deadline times, and in the next section we present a method of examining the stability of the algorithms. The description of the computational experiments conducted is presented in Sect. 6. Finally, we draw some conclusions in Sect. 7.

2. Problem definition and preliminaries. In this paper we consider the scheduling problem on a single machine. The machine can perform only one task at a time. For task i ($i = 1, \ldots, n$), 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 task 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 (TWLJ) on a single machine 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 $1 \parallel \sum w_i U_i$, 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 [10] 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 tasks 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 a set of tasks, the TWLJ problem is strongly NP-hard even when the task realization times are unities (Garey and Johnson [4]). Lenstra and Rinnooy Kan [12] have proven that if a partial order relation is a union of independent chains, the problem is also strongly NP-hard.

Only a few exact algorithms solving the TWLJ problem have been published. They are based on the dynamic programming method (Lawler and Moore [11] with $O(n \min\{\sum p_i, \max\{d_i\}\})$ complexity, and Sahni [20] with $O(n \min\{\sum p_i, \sum w_i, \max\{d_i\}\})$ complexity) and on a limitation and division method (Villarreal and Bulfin [22], Potts and Van Wassenhowe [17], [18] and Wodecki [24]). 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 $\mathcal{J} = \{1, \ldots, n\}$ of tasks that have to be processed without interruptions on a machine that can work on one task at a time. The task can start at time zero. For task $i \in \mathcal{J}$ let p_i be the *processing time*, d_i the *expected deadline*, and w_i a *cost function weight*. We want to determine a task sequence that minimizes the weight of late tasks.

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

$$U_i = \begin{cases} 0 & \text{if } C_i \leq d_i, \\ 1 & \text{otherwise.} \end{cases}$$

We have to determine a task sequence that minimizes

$$\sum_{i=1}^{n} w_i U_i \tag{2.1}$$

Let Φ_n be the set of permutations of $\mathcal{J}$. The cost of the permutation $\pi \in \Phi_n$

is defined as follows:

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

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

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

Exact efficient algorithms to solve the TWLJ problem only exist when the number of tasks does not exceed 50 (or 80 in a multiprocessor environment [24]). That is why in practice we use approximate algorithms (essentially the correction type).

3. The 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 1%). Most of them use local search methods. These consist in viewing a subset of acceptable solutions in sequence, and pointing out the best one according to a given criterion. One of these methods is the *tabu search*, whose basic criterions are:

(i) *neighborhood*: a subset of acceptable solutions, whose elements are rigorously analyzed;
(ii) *move*: a function that converts one solution into another one;
(iii) *tabu list*: a list containing the attributes of a certain number of solutions analyzed recently;
(iv) *ending condition*: most of the time fixed by the number of algorithm iterations.

Let $\pi \in \Phi_n$ be any (starting) permutation, L_{TS} a tabu list, and π^* the best solution found up to the present time (the starting solution and π^* can be any permutation).

Tabu Search Algorithm

repeat
  Define the neighborhood $\mathcal{N}(\pi)$ of the permutation π;
  Remove from $\mathcal{N}(\pi)$ the permutations forbidden by the L_{TS} list;
  Determine the permutation $\delta \in \mathcal{N}(\pi)$ for which
    $W(\delta) = \min\{W(\beta)\colon \beta \in \mathcal{N}(\pi)\}$;
  if $(W(\delta) < W(\pi^*))$ ***then***
    $\pi^* := \delta$;

8 Include the parameters of δ on the L_{TS} list;
9 $\pi := \delta$
10 ***until*** (*ending_ condition*).

The computational complexity of the algorithm depends mostly on the way the neighborhood is generated and viewed. Below we detail the basic elements of the algorithm.

3.1. The move and the neighborhood. Let $\pi = (\pi(1), \ldots, \pi(n))$ be any permutation from Φ_n, and

$$L(\pi) = \{\pi(i)\colon C_{\pi(i)} > d_{\pi(i)}\}$$

be the *set of late tasks* in π.

Let π_l^k $(l = 1, \ldots, k-1, k+1, \ldots, n)$ be the permutation obtained from π by interchanging $\pi(k)$ and $\pi(l)$. We can say that π_l^k is generated from π by a *swap move* (*s-move*) s_l^k (i.e. $\pi_l^k = s_l^k(\pi)$). Let $M(\pi(k))$ be the set of s-moves of the element $\pi(k)$. Define

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

the set of *s-moves* of all late elements π in the permutation. The cardinality of $M(\pi)$ is bounded above by $n(n-1)/2$.

The neighborhood of $\pi \in \Phi_n$ is the set of permutations

$$\mathcal{N}(\pi) = \{s_l^k(\pi)\colon s_l^k \in M(\pi)\}.$$

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

3.2. The tabu list. In order to avoid generating a cycle (by returning to the same permutation after a small number of algorithm iterations), some attributes of every move are saved in a tabu list. It is operated according to the FIFO queue rule. We put the move's attribute, the tuple $(\pi(r), j, W(\pi_j^r))$, on the tabu list L_{TS} when making the $s_j^r \in M(\pi)$ move (generating from $\pi \in \Phi_n$ the permutation π_j^r).

Suppose that we analyze the move $s_l^k \in M(\beta)$ that generates from $\beta \in \Phi_n$ the permutation β_l^k. If the tuple (r, j, Ψ) such that $\beta(k) = r$, $l = j$ and $W(\beta_l^k) \geq \Psi$ is on the L_{TS} list, such a move is forbidden and removed from the set $M(\beta)$. The only parameter of this list is its length, the number of elements it contains. There are many realizations of the tabu list presented in the given references.

4. Stochastic processing times. In this section we study the problem of minimizing the delay cost sum, when the processing times are random

variables with Erlang distribution. In the literature, scheduling problems with random processing times have been analyzed, principally with normal or uniform distribution (van den Akker and Hoogeveen [2]) or with exponential distribution (Pinedo [16]). The Erlang distribution $\mathcal{E}(\alpha, \lambda)$ is much more efficient compared with the other distributions in modeling realistic events (for example queues, financial operations analysis, agriculture, services, transport, construction, etc.) with uncertain parameters. It is even more important, because in practice processing times tend to get longer, and not shorter.

Because the starting point for our study is a deterministic instance of the problem (p_i, w_i, d_i), $i = 1, \ldots, n$, we present a method of its randomization. In order to simplify the notation we suppose that the analyzed solution (the task permutation) π is the identity permutation.

The randomization consists in finding random variables with Erlang distribution $\widetilde{p_i} \sim \mathcal{E}(\alpha_i, \lambda)$ $(i = 1, \ldots, n)$ such that the expected value $E(\widetilde{p_i})$ is p_i. Define

$$\lambda = \max\left\{\frac{2}{\min_{1\le i\le n} p_i}, 1\right\} \quad \text{and} \quad \alpha_i = p_i\lambda \quad (i = 1, \ldots, n).$$

The random version of the problem is described by n tuples $(\widetilde{p_i}, w_i, d_i)$, where $\widetilde{p_i}$ is a random variable representing the processing time, and w_i and d_i are respectively the cost function weight and the deadline.

As in the case of the deterministic problem, the completion time for task i (according to its place in the permutation π) is $\tilde{C}_i = \sum_{j=1}^{i} \widetilde{p_i}$. Then the delay $\tilde{U}_i$ is 0 if $\tilde{C}_i \le d_i$, and $\tilde{U}_i$ is 1 if $\tilde{C}_i > d_i$. The processing times as well as the delays are random variables. The target function is also a random variable:

$$\widetilde{W}(\pi) = \sum_{i=1}^{n} w_i \tilde{U}_i. \tag{4.1}$$

In the tabu search method the target function is used to determine the best element (the task permutation) from the neighborhood. In order to compare permutations (their target function values (4.1)) we have to attribute some numerical values to them. Because the expression (4.1) is a random variable, we can replace it by a particular moment or a combination of moments. Experiments have shown that the best results were obtained when the first and the second central moments (the mean and the variance) were used to compare solutions. Finally, we study two functions:

1. $\mathcal{W}_1(\pi) = c \cdot E(\widetilde{W}(\pi)) + (1 - c) \cdot D(\widetilde{W}(\pi))$ $(c \in [0, 1])$,
2. $\mathcal{W}_2(\pi) = \sum_{i=1}^{n} w_i E(\tilde{U}_i) + \sum_{i=1}^{n} w_i D^2(\tilde{U}_i)$.

The first one is a convex combination (depending on the parameter c) of the mean and the standard deviation of the target function (4.1). The other is a weighted sum of the means and the variances. We now present methods that allow one to calculate the values of both these functions.

4.1. Calculating $\mathcal{W}_1$. Since the processing time has the Erlang distribution $\widetilde{p_i} \sim \mathcal{E}(\alpha_i, \lambda)$, $i = 1, \dots, n$, the completion time of the task (for the identity sequence π) is

$$\tilde{C}_i = \sum_{j=1}^{i} \widetilde{p_j} \sim \mathcal{E}(\alpha_1 + \cdots + \alpha_i, \lambda).$$

The target function studied in this subsection is of the form

$$\mathcal{W}_1(\pi) = c \cdot E(\widetilde{W}(\pi)) + (1 - c) \cdot D(\widetilde{W}(\pi)). \tag{4.2}$$

Therefore we have to calculate its expected value $E(\widetilde{W}(\pi))$ and standard deviation $D(\widetilde{W}(\pi))$.

Let $F_i(x) = F_{\tilde{p}_1 + \cdots + \tilde{p}_i}(x)$ be the cumulative distribution function (CDF) of the ith job processing time. Then

$$E(\tilde{U}_i) = 0 \cdot P(\tilde{C}_i \le d_i) + 1 \cdot P(\tilde{C}_i > d_i) = 1 - F_i(d_i)$$

and

$$E(\widetilde{W}(\pi)) = E\Big(\sum_{i=1}^{n} w_i \tilde{U}_i\Big) = \sum_{i=1}^{n} w_i E(\tilde{U}_i) = \sum_{i=1}^{n} w_i (1 - F_i(d_i)). \tag{4.3}$$

In order to calculate the variation $D^2(\widetilde{W}(\pi))$ we use the formula

$$D^2(\widetilde{W}(\pi)) = D^2\Big(\sum_{i=1}^{n} w_i \tilde{U}_i\Big).$$

Since $E(\widetilde{U_i}^2) = 1 - F_i(d_i)$, we see that

$$D^2(\tilde{U}_i) = E(\tilde{U_i}^2) - (E(\tilde{U}_i))^2 = F_i(d_i)(1 - F_i(d_i)).$$

Therefore

$$D^2(\widetilde{W}(\pi)) = \sum_{i=1}^{n} w_i \left(F_i(d_i)(1 - F_i(d_i))\right) + 2 \sum_{i<j} w_i w_j \operatorname{cov}(\tilde{U}_i, \tilde{U}_j),$$

where the covariance $\operatorname{cov}(\tilde{U}_i, \tilde{U}_j)$ is calculated from the formula

$$\operatorname{cov}(\tilde{U}_i, \tilde{U}_j) = E(\tilde{U}_i \tilde{U}_j) - E(\tilde{U}_i) E(\tilde{U}_j).$$

We notice that the random variables $\tilde{U}_i$ and $\tilde{U}_j$ are not independent. In order to calculate $E(\tilde{U}_i \tilde{U}_j)$ we study the two-dimensional random variable $(\tilde{C}_i, \tilde{C}_j - \tilde{C}_i)$, where $\tilde{C}_i = \tilde{p}_1 + \cdots + \tilde{p}_i$ and $\tilde{C}_j - \tilde{C}_i = \tilde{p}_{i+1} + \cdots + \tilde{p}_j$. The

variables $\tilde{C}_i$ and $\tilde{C}_j - \tilde{C}_i$ are independent. To fix ideas suppose that $d_i < d_j$ (otherwise interchange d_i and d_j). We have

$$\begin{aligned} E(\tilde{U}_i\tilde{U}_j) &= E(\tilde{U}_i(\tilde{U}_i + (\tilde{U}_j - \tilde{U}_i))) = P(\tilde{C}_i > d_i, \tilde{C}_i + (\tilde{C}_j - \tilde{C}_i) > d_j) \\ &= \iint\limits_{\{(x,y)\colon x>d_i,\, x+y>d_j\}} f_i(x)f_j(y)\,dx\,dy \\ &= \int\limits_{d_i}^{d_j}\int\limits_{d_j-x}^{\infty} f_i(x)f_j(y)\,dy\,dx + \int\limits_{d_j}^{\infty}\int\limits_{0}^{\infty} f_i(x)f_j(y)\,dy\,dx, \end{aligned}$$

where $f_i(x)$ and $f_j(y)$ are the density functions of the distributions $\mathcal{E}(\alpha_1 + \cdots + \alpha_i, \lambda)$ and $\mathcal{E}(\alpha_{i+1} + \cdots + \alpha_j, \lambda)$.

We set

$$\mathrm{FI} = \int\limits_{d_i}^{d_j}\int\limits_{d_j-x}^{\infty} f_i(x)f_j(y)\,dy\,dx, \qquad \mathrm{SI} = \int\limits_{d_j}^{\infty}\int\limits_{0}^{\infty} f_i(x)f_j(y)\,dy\,dx.$$

To calculate FI and SI we make use of the fact that the CDF of the Erlang distribution has the form

$$F_{c,\lambda}(x) = 1 - e^{-\lambda x}\sum_{m=1}^{c-1}\frac{(\lambda x)^m}{m!}.$$

In order to simplify the formula, let $\beta_1 = \alpha_1 + \cdots + \alpha_i$ and $\beta_2 = \alpha_{i+1} + \cdots + \alpha_j$. Then

$$\begin{aligned} \mathrm{FI} &= \int\limits_{d_i}^{d_j}\int\limits_{d_j-x}^{\infty} f_{\beta_1,\lambda}(x)f_{\beta_2,\lambda}(y)\,dy\,dx = \int\limits_{d_i}^{d_j} f_{\beta_1,\lambda}(x)\Big[\int\limits_{d_j-x}^{\infty} f_{\beta_2,\lambda}(y)\,dy\Big]\,dx \\ &= \int\limits_{d_i}^{d_j} f_{\beta_1,\lambda}(x)(1 - F_{\beta_2,\lambda}(d_j - x))\,dx \\ &= \int\limits_{d_i}^{d_j} \frac{\lambda^{\beta_1}x^{\beta_1-1}e^{-\lambda x}}{(\beta_1-1)!}e^{-\lambda(d_j-x)}\sum_{k=1}^{\beta_2-1}\frac{\lambda^k(d_j-x)^k}{k!}\,dx \\ &= \sum_{k=1}^{\beta_2-1}\frac{\lambda^{\beta_1}}{(\beta_1-1)!}\frac{\lambda^k}{k!}e^{-\lambda d_j}\int\limits_{d_i}^{d_j} x^{\beta_1-1}(d_j-x)^k\,dx \\ &= \frac{\lambda^{\beta_1}}{(\beta_1-1)!}e^{-\lambda d_j}\sum_{k=1}^{\beta_2-1}\frac{\lambda^k}{k!}\int\limits_{d_i}^{d_j} x^{\beta_1-1}\sum_{m=0}^{k}\binom{k}{m}(-1)^m x^m d_j^{k-m}\,dx \end{aligned}$$

$$= \frac{\lambda^{\beta_1}}{(\beta_1-1)!}e^{-\lambda d_j}\sum_{k=1}^{\beta_2-1}\frac{\lambda^k}{k!}\sum_{m=0}^{k}\binom{k}{m}(-1)^m d_j^{k-m}\int_{d_i}^{d_j} x^{\beta_1+m-1}\,dx$$

$$= \frac{\lambda^{\beta_1}}{(\beta_1-1)!}e^{-\lambda d_j}\sum_{k=1}^{\beta_2-1}\frac{\lambda^k}{k!}\sum_{m=0}^{k}\binom{k}{m}(-1)^m d_j^{k-m}\left(\frac{d_j^{\beta_1+m}}{\beta_1+m}-\frac{d_i^{\beta_1+m}}{\beta_1+m}\right)$$

$$= \frac{\lambda^{\beta_1}}{(\beta_1-1)!}e^{-\lambda d_j}\sum_{k=1}^{\beta_2-1}\lambda^k\frac{1}{k!}\,\frac{k!}{m!(k-m)!}(-1)^m\frac{d_j^{k-m}}{\beta_1+m}(d_j^{\beta_1+m}-d_i^{\beta_1+m})$$

$$= \frac{\lambda^{\beta_1}}{(\beta_1-1)!}e^{-\lambda d_j}\sum_{k=1}^{\beta_2-1}\sum_{m=0}^{k}\frac{(-1)^m}{m!(k-m)!}\,\frac{d_j^k}{\beta_1+m}\left[d_j^{\beta_1}-d_i^{\beta_1}\left(\frac{d_i}{d_j}\right)^m\right]$$

$$= \frac{\lambda^{\beta_1}}{(\beta_1-1)!}e^{-\lambda d_j}\sum_{k=1}^{\beta_2-1}(\lambda d_j)^k\sum_{m=0}^{k}\frac{(-1)^m}{m!(k-m)!(\beta_1+m)}\left[d_j^{\beta_1}-d_i^{\beta_1}\left(\frac{d_i}{d_j}\right)^m\right],$$

$$\mathrm{SI} = \int_{d_j}^{\infty} f_{\beta_1,\lambda}(x)\,dx\cdot\int_0^{\infty} f_{\beta_2,\lambda}\,dy = 1-F_{\beta_1,\lambda}(dj) = e^{-\lambda d_j}\sum_{m=1}^{\beta_1-1}\frac{(\lambda d_j)^m}{m!}.$$

Just as SI, the factor FI can be easily transformed into a form involving CDF functions, which may simplify the implementation:

$$\mathrm{FI} = \sum_{k=1}^{\beta_2-1} P(X=\beta_1+k)(1-F_{B(\beta_1,k+1)}(d_i/d_j)),$$

where X has the Poisson distribution $\mathcal{P}(\lambda d_j)$, which is of the form

$$P(X=m) = \frac{e^{-\lambda d_j}(\lambda d_j)^m}{m!} \qquad (m=0,1,\ldots).$$

Eventually

$$D^2(\widetilde{W}(\pi)) = \sum_{i=1}^{n} w_i(F_i(d_i)(1-F_i(d_i))) + 2\sum_{i<j} w_i w_j(\mathrm{FI}+\mathrm{SI}-(1-F_i(d_i))(1-F_j(d_j))). \tag{4.4}$$

So, to calculate $\mathcal{W}_1(\pi)$ for $\pi\in\Phi_n$, we use formulas (4.3) and (4.4).

4.2. Calculating $\mathcal{W}_2$. The other function analyzed is of the form

$$\mathcal{W}_2(\pi) = \sum_{i=1}^{n} w_i E(\tilde{U}_i) + \sum_{i=1}^{n} w_i D^2(\tilde{U}_i). \tag{4.5}$$

Using the above considerations on $\mathcal{W}_1$ we can easily prove that

$$\mathcal{W}_2(\pi) = \sum_{i=1}^{n} w_i(1 - F_i(d_i)) + \sum_{i=1}^{n} w_i F_i(d_i)(1 - F_i(d_i)),$$

where F_i is the CDF of the processing time of task i in the permutation π.

Both functions $\mathcal{W}_1$ and $\mathcal{W}_2$ can be used in the tabu search algorithm in order to determine the best element in the neighborhood.

To obtain the probabilistic version of the tabu search algorithm presented in Section 3 we have to replace the target function W (lines 5 and 6) respectively by the functions $\mathcal{W}_1$ and $\mathcal{W}_2$.

5. Algorithm stability. In this section we introduce a certain measure which allows us to examine the impact of a change in task parameters on the target function value.

Let $\delta = ((p_1, w_1, d_1), \ldots, (p_n, w_n, d_n))$ be an example of (deterministic) data for the TWLJ problem. Denote by $\mathfrak{D}(\delta)$ the set of data generated from δ by perturbing processing times. The perturbation consists in substituting random values for the original times. The perturbed data $\gamma \in \mathfrak{D}(\delta)$ have the form $\gamma = ((p'_1, w_1, d_1), \ldots, (p'_n, w_n, d_n))$, where the processing time p'_i $(i = 1, \ldots, n)$ is the realization of a random variable $\tilde{p}_i$ with Erlang distribution $\mathcal{E}(\alpha_i, \lambda)$ (both distribution parameters have been determined in Section 4), while the expectetd value $E(\tilde{p}_i)$ is p_i.

Let $\mathbb{A}$ and $\tilde{\mathbb{A}}$ be the respective deterministic and probabilistic algorithms, based on the tabu search method (see Section 3) for the TWLJ problem. We introduce the following:

(i) π_δ: the best solution (permutation) determined for data δ by the algorithm $\mathbb{A}$,
(ii) $\tilde{\pi}_\delta$: the best solution determined for randomized data δ by the algorithm $\tilde{\mathbb{A}}$,
(iii) π_φ: the best solution determined for data $\varphi \in \mathfrak{D}(\delta)$ by the algorithm $\mathbb{A}$.

Let $W(\mathcal{A}, \pi, \delta)$ be the processing cost (2.1) incurred when using an algorithm $\mathcal{A} \in \{\mathbb{A}, \tilde{\mathbb{A}}\}$. Then we call

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

the *stability* of $\mathcal{A}$ on the perturbed data set $\mathfrak{D}(\delta)$. Because in our studies on the determination of π_φ we have adopted π_δ as the starting solution in the algorithm $\mathcal{A}$ (tabu search), we have $W(\mathcal{A}, \pi_\delta, \varphi) - W(\mathcal{A}, \pi_\varphi, \varphi) > 0$.

Let Ω be a deterministic instances set for the scheduling problem. We define the stability of the algorithm $\mathcal{A}$ on Ω as follows:

$$\mathbb{S}(\mathcal{A}, \Omega) = \frac{1}{\Omega} \sum_{\delta \in \Omega} \Delta(\mathcal{A}, \delta, \mathfrak{D}(\delta)). \tag{5.1}$$

In the following section we will present numerical experiments that allow comparisons of the deterministic stability coefficient $\mathbb{S}(\mathbb{A}, \Omega)$ with the probabilistic stability coefficient $\mathbb{S}(\tilde{\mathbb{A}}, \Omega)$.

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 $1 \parallel \sum w_i T_i$ were generated in a randomized way (see [24]), and are available on the OR Library website [15]. For a given number of n tasks ($n = 40, 50, 100$) we have determined n tuples (p_i, w_i, d_i), $1, \ldots, n$, 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 critical lines are drawn from the range $[P(1 - \mathrm{TF} - \mathrm{RDD}/2), P(1 - \mathrm{TF} + \mathrm{RDD}/2)]$ depending on the parameters $\mathrm{RDD}, \mathrm{TF} = 0.2, 0.4, 0.6, 0.8, 1.0$, 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 375 examples (125 for every n).

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

The deterministic $\mathcal{AD}$ and probabilistic $\widetilde{\mathcal{AP}}$ algorithms were started from the identity permutation. Moreover, we have adopted the following parameters:

(i) the tabu list length: n,
(ii) the maximum number of algorithm iterations (*ending_ condition*): $n/2$ or n.

The deterministic algorithm ($\mathcal{AD}$) has been performed on Ω, and the probabilistic algorithm on $\widetilde{\Omega}$. In order to evaluate the stability coefficient (5.1) 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 5). Then all these examples have been solved by the $\mathcal{AD}$ algorithm. Based on these calculations, we have determined the stability coefficient of both algorithms. The results are presented in Tables 1 and 2.

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

n	Algorithm	
	deterministic $\mathcal{AD}$	random $\widetilde{\mathcal{AP}}$
40	0.775739356	0.053583636
50	0.520483832	0.053137101
100	0.596875969	0.032746328
avg	**0.631033052**	**0.046489022**

Table 1 contains the results obtained for $n/2$ iterations. The average stability coefficient for the deterministic algorithm is $\mathbb{S}(\mathcal{AD}, \Omega) = 0.631$, and for the random algorithm $\mathbb{S}(\widetilde{\mathcal{AP}}, \Omega) = 0.046$. This means that the perturbation of the solution determined by the $\mathcal{AD}$ algorithm causes a target function value deterioration of about 63%. In the $\widetilde{\mathcal{AP}}$ algorithm the deterioration is only about 5%. So the medium error for the deterministic algorithm is more than 12 times that for the probabilistic algorithm.

Table 2 contains the results of twice as many 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 $\mathcal{AD}$ algorithm is more than 10 times that for the $\widetilde{\mathcal{AP}}$ algorithm. Moreover, the data perturbation causes (for the solution determined by the probabilistic algorithm) a target function value deterioration of around 7.4%.

Table 2. Stability coefficient for n iterations

n	Algorithm	
	deterministic $\mathcal{AD}$	random $\widetilde{\mathcal{AP}}$
40	0.852279479	0.085561408
50	0.674076051	0.070146136
100	0.805104581	0.066410047
avg	**0.77715337**	**0.074039197**

We have also made calculations for more iterations ($n \log n$, n^2). The stability coefficient for both algorithms has 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.

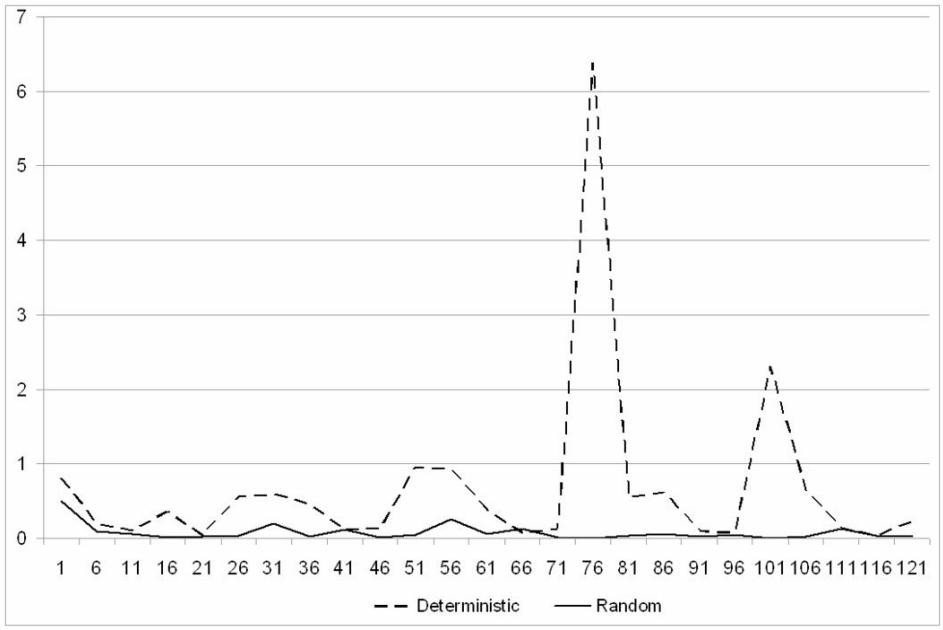

Fig. 1. Solution stability graphs for the algorithms $\mathcal{AD}$ and $\widetilde{\mathcal{AP}}$ for the examples with $n = 50$ jobs and n iterations

The effect of the number of iterations on the solution stability is also presented in Figure 1. For simplicity we only give results for the example numbers $1, 6, 11, \ldots, 121$. Figure 1 contains stability graphs for both algorithms for n iterations (and number of tasks $n = 50$). It is important to notice that for the major part the random algorithm is significantly more stable, and for the deterministic algorithm there are two dominant "peaks" (examples 76 and 101) that are not reflected by the random algorithm. The maximum error for the probabilistic algorithm does not exceed 49%, while for the deterministic algorithm it is over 600%.

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 12%). 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 by using random variables with Erlang distribution. This distribution represents, in a significantly better way than other distributions, the processing times that can vary during the process. 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 task parameters. The experiments have shown that the algorithm in which the processing times are random variables with Erlang distribution is very stable. The medium relative error for perturbed data does not exceed 6% when the iteration number is small, and the calculation time is short.

Acknowledgements. The work was supported by MNiSW of Poland, within the grant no. NN514 232237.

References

[1] M. van den Akker and H. Hoogeveen, *Minimizing the number of late jobs in case of stochastic processing times with minimum success probabilities*, Technical Report, Inst. Information and Computation Sci., Utrecht Univ., 2004.

[2] —, —, *Minimizing the number of late jobs in a stochastic setting using a chance constraint*, J. Sched. 11 (2008), 59–69.

[3] B. C. Dean, *Approximation algorithms for stochastic scheduling problems*, PhD thesis, MIT, 2005.

[4] M. R. Garey and D. S. Johnson, *Scheduling tasks with nonuniform deadlines on two processors*, J. Assoc. Comput. Mach. 23 (1976), 461–467.

[5] H. Ishibuchi, N. Yamamoto, S. Misaki and H. Tanaka, *Local search algorithm for flow shop scheduling with fuzzy due-dates*, Int. J. Production Economics 33 (1994), 53–66.

[6] H. Ishii, *Fuzzy combinatorial optimization*, Japan. J. Fuzzy Theory Systems 4 (1992), 111–122.

[7] T. Itoh and H. Ishii, *Fuzzy due-date scheduling problem with fuzzy processing times*, Int. Trans. Oper. Res. 6 (1999), 639–647.

[8] R. M. Karp, *Reducibility among combinatorial problems*, in: Complexity of Computer Computations, R. E. Miller and J. W. Thatcher (eds.), Plenum Press, New York, 1972, 85–103.

[9] H. Kise, T. Ibaraki and H. Mine, *A solvable case of the one-machine scheduling problem with ready times and due times*, Oper. Res. 26 (1978), 121–126.

[10] E. L. Lawler, *A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness*, in: Studies in Integer Programming, Ann. Discrete Math. 1, North-Holland, Amsterdam, 1977, 331–342.

[11] E. L. Lawler and J. M. Moore, *A functional equation and its application to resource allocation and sequencing problems*, Management Sci. 16 (1969), 77–84.

[12] J. K. Lenstra and A. H. G. Rinnooy Kan, *Complexity results for scheduling chains on a single machine*, Eur. J. Oper. Res. 4 (1980), 270–275.

[13] C. L. Monma, *Linear-time algorithms for scheduling on parallel processors*, Oper. Res. 30 (1982), 116–124.

[14] J. M. Moore, *An n job, one machine sequencing algorithm for minimizing the number of late jobs*, Management Sci. 15 (1968), 102–109.

[15] OR Library, http://www.ms.ic.ac.uk/info.html.

[16] M. Pinedo, *Stochastic scheduling with release dates and due dates*, Oper. Res. 31 (1983), 559–572.

[17] C. N. Potts and L. N. Van Wassenhove, *A branch and bound algorithm for the total weighted tardiness problem*, ibid. 33 (1985), 363–377.

[18] —, —, *Algorithms for scheduling a single machine to minimize the weighted number of late jobs*, Management Sci. 34 (1988), 843–858.

[19] H. Prade, *Using fuzzy set theory in a scheduling problem: A case study*, Fuzzy Sets Systems 2 (1979), 153–165.

[20] S. K. Sahni, *Algorithms for scheduling independent tasks*, J. Assoc. Comput. Mach. 23 (1976), 116–127.

[21] M. Shaked and J. G. Shanthikumar (eds.), *Stochastic Orders and Their Applications*, Academic Press, San Diego, 1994.

[22] F. J. Villarreal and R. L. Bulfin, *Scheduling a single machine to minimize the weighted number of tardy jobs*, IIE Trans. 15 (1983), 337–343.

[23] J. Vondrák, *Probabilistic methods in combinatorial and stochastic optimization*, PhD thesis, MIT, 2005.

[24] M. Wodecki, *A branch-and-bound parallel algorithm for single-machine total weighted tardiness problem*, Int. J. Advanced Manuf. Technol. 37 (2007), 996–1004.

[25] X. Zhou and X. Cai, *General stochastic single-machine scheduling with regular cost functions*, Math. Comput. Modelling 26 (1997), 95–108.

Paweł Rajba, Mieczysław Wodecki
Institute of Computer Science
University of Wrocław
50-383 Wrocław, Poland
E-mail: pawel@ii.uni.wroc.pl
mwd@ii.uni.wroc.pl

Received on 26.2.2010;
revised version on 8.9.2011 (2034)