Chapter 4
Stable Scheduling with Random Processing Times

Wojciech Bożejko, Paweł Rajba, and Mieczysław Wodecki

Abstract. In this work stability of solutions determined by algorithms based on tabu search method for a certain (NP-hard) one-machine arrangement problem was examined. The times of tasks performance are deterministic and they also constitute random variables of the standard or the Erlang's schedule. The best results were obtained when as a criterion to choose an element from the neighborhood convex combinations of the first and the second moments of the random goal function were accepted. In this way determined solutions are stable, i.e. little sensitive to parameters random changes.[1]

4.1 Introduction

Research concerning problems of algorithms arrangement refers mainly to deterministic models. To solve such problems, which belong in the majority of cases to the strongly NP-hard class, rough algorithms are applied successfully. They are mainly based on local optimalization methods: simulated annealing, tabu search and a genetic algorithm. Determined by these algorithms solutions only slightly differ from best solutions. However, in practice, in the course of a process realisation (according to the fixed schedule) it appears very often that certain parameters (e.g. the task completion time) are different from the initial ones. By the lack of the solutions

Wojciech Bożejko
Institute of Computer Engineering, Control and Robotics,
Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland
e-mail: `wojciech.bozejko@pwr.wroc.pl`

Paweł Rajba · Mieczysław Wodecki
Institute of Computer Science, University of Wrocław, Joliot-Curie 15,
50-383 Wrocław, Poland
e-mail: `{pawel.rajba,mwd}@ii.uni.wroc.pl`

[1] The work was supported by MNiSW Poland, within the grant No. N N514 232237.

stability in the fixed schedule there may occur a big mistake, which makes such a schedule unacceptable. That's why there is a necessity to construct such models and methods of their solutions that would take into account potential changes in the course of parameters process realisation and generate stable solutions.

Problems of arrangements with uncertain data may be solved using methods based on elements of probability calculus (van den Akker and Hoogeveen [25], Vondrák [30], Dean [5], Cai [4] or fuzzy sets theories (Prade [22], Itoh and Ishii [10]). They make it possible to consider uncertainties already at the stage of the mathematical model construction or directly in algorithms being constructed.

In this work we deal with the one-machine problem of tasks arrangement with the latest completion times and the minimalisation of the costs sum of tardy tasks. Times of tasks completion are deterministic and constitute random variables of a standard or the Erlang's schedule. On the basis of this problem the resistance to a random variable of constructive solutions of parameters according to tabu search metaheuristics is examined. The paper constitutes a continuation of work presented in Bożejko et al. [2].

4.2 Problem Definition and Method of Its Solution

Algorithms based on the tabu search method have been applied successfully to solve NP-hard problems of combinatorial optimalization to date. They are simple to be implemented and comparative results in literature show that determined by these algorithms solutions only slightly differ from the best ones.

4.2.1 Single Machine Scheduling Problem

In this problem each task from the set $\mathcal{J}=\{1,2,\dots,n\}$ should be performed uninterruptedly on a machine which in any moment can do at most one task. For a task i, let p_i, d_i,w_i be the execution time, the expected completion time and the penalty for a delay of a task. Such a sequence of tasks' performance should be determined in a way that the penalty sum is minimal.

Let Π be a set of permutations of elements from $\mathcal{J}$. For any permutation $\pi\in \Pi$ by $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)}$ we denote the completion time of i-th task] in a permutation π. Then

$$U_{\pi(i)} = \begin{cases} 0, & \text{if } C_{\pi(i)} \le d_{\pi(i)}, \\ 1, & \text{otherwise,} \end{cases} \tag{4.1}$$

we call the task delay, $w_{\pi(i)} \cdot U_{\pi(i)}$ the *penalty* for delay and

$$F(\pi) = \sum_{i=1}^{n} w_{\pi(i)} U_{\pi(i)} \tag{4.2}$$

the permutation cost.

The problem of Minimization of the Total Weighted of Late jobs (abbrev. MTWL) consist in determining such a permutation $\pi^* \in \Pi$ that

$$F(\pi^*) = \min\{\mathscr{F}(\beta) : \beta \in \Pi\}.$$

In literature (see [9]) this problem is indicated as $1||\sum w_i U_i$ and it belongs to the strongly NP-hard problems class (Karp [13]). Such problems heve 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 [17] 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 [18] with $O(n \ln n)$ complexity. Lawler [15] 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. [12] 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 the set of tasks, the MTWL problem is strongly NP-hard even when the task realization times are unities (Garey and Johnson [6]). Lenstra and Rinnoy Kan [16] have proven that if a partial order relation is a union of independent chains, the problem is also strongly NP-hard.

Optimal algorithms of this solution based on the dynamic programming method were presented by Lawler and Moor [14]; a pseudo-polynomial algorithm with the computational complexity $O(n \min\{\sum_j p_j, \max_j\{d_j\}\})$ and based on the branch and bound method - by Potts [20], Sourd [26] and Wodecki [27]. Exact algorithms make it possible to determine optimal solutions effectively only if the number of tasks does not exceed 50 (80 in a multi-processor neighborhood, [27]).

Therefore, in practice only rough algorithms are applied (mainly metaheuristics) constituting adaptation of algorithms of the problem solution $1||\sum w_i T_i$, ([3]). We can refer to the models and algorithms of Józefowska [11] for a detailed survey on the models and algorithms developed in this area.

4.3 Problem Description and Preliminaries

Each schedule of jobs can be represented by permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ on set $\mathscr{J}$. Let Π denote the set of all such permutations. The total cost of $\pi \in \Pi$ is $\sum_{i=1}^{n} w_{\pi(i)} U_{\pi(i)}$, where $C_{\pi(i)} = \sum_{j=1}^{i} p_{\pi(j)}$ is a completion time of the job $\pi(i)$.

Job $\pi(i)$ is considered as *early* one, if it is completed before its due date (i.e. $C_{\pi(i)} \leq d_{\pi(i)}$), or *tardy* if the job is completed after its due date (i.e. $C_{\pi(i)} > d_{\pi(i)}$).

Each permutation $\pi \in \Pi$ is decomposed into m $(m \le n)$ subsequences $B_1, B_2, \ldots,$ B_m, called *blocks* in π, each of them contains the jobs having in common specific properties, where

1. $B_k = (\pi(f_k), \pi(f_k+1), \ldots, \pi(l_k-1)), \pi(l_k)), \qquad l_{k-1}+1 = f_k \le l_k,$
   $k = 1, 2, ..., m, \quad l_0 = 0, \quad l_m = n.$
2. All the jobs $j \in B_k$ satisfy the following condition:
   either
   $$d_j \ge C_{\pi(l_k)}, \qquad \textbf{(C1)},$$
   or
   $$d_j < S_{\pi(f_k)} + p_j, \qquad \textbf{(C2)},$$
   where $S_{\pi(f_k)}$ is a *starting time* of the job $\pi(f_k)$, i.e. $S_{\pi(f_k)} = C_{\pi(f_k)} - p_{\pi(f_k)}$. Clearly, each job $j \in B_k$ satisfying Condition $C1$ (or $C2$) is a early (or tardy) one in π.
3. B_k is maximal subsequence of π in which all the jobs satisfy either Condition $C1$ or Condition $C2$.

Jobs $\pi(f_k)$ and $\pi(l_k)$ in B_k are the *first* and *last* ones, respectively. Note that a block can contain only one job, i.e. $|B_k| = 1$, and then $f_k = l_k$. Note that all the blocks are connected "in series" in permutation π. By definition, there exist two type of blocks implied by either $C1$ or $C2$. To distinguish them, we will use the *E-block* and *T-block* notions (or alternatively B_k^E and B_k^T), respectively (see Figure 4.1).

Example 1. Let us consider the n=10 jobs' instance that is specified in Table 4.1.

Table 4.1. Data for the instance

i	1	2	3	4	5	6	7	8	9	10
p_i	2	3	1	2	3	2	3	3	2	4
d_i	12	19	12	9	5	1	17	24	19	3
w_i	3	1	2	5	3	3	4	2	4	5

Let $\pi = (1,2,3,4,5,6,7,8,9,10)$. Permutation π contains five blocks, (i.e. $m = 5$), $l_1 = 1$, $f_1 = 4$, $l_2 = 5$, $f_2 = 6$, $l_3 = f_3 = 7$, $l_4 = f_4 = 8$, $l_5 = 9$ and $f_5 = 10$. Blocks $B_1 = (1,2,3,4)$, $B_2 = (5,6)$, $B_3 = (7)$, $B_4 = (8)$ and $B_5 = (9,10)$. There are: three E-blocks: B_1, B_3, B_4, and two T-blocks: B_2 and B_5. These blocks are shown on Figure 4.1.

Let
$$F_k(\pi) = \sum_{j \in B_k} w_j U_j,$$

be a *partial value* of the objective associated with the block B_k in π. It is clear that by the definition of blocks in π, we have

$$F(\pi) = \sum_{k=1}^{m} F_k(\pi).$$

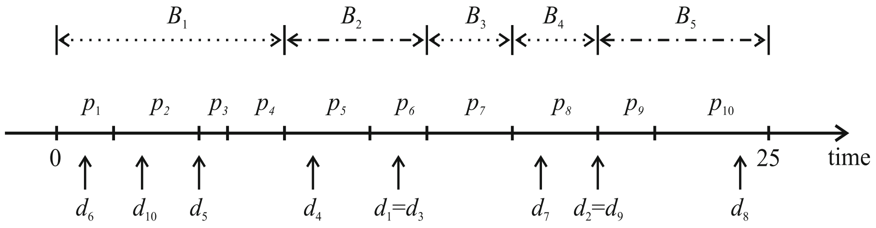

Fig. 4.1. Blocks in permutation π

It is evident that by Condition $C1$, for any permutation of jobs within an E-block B_k^E of π, (i.e. in the positions $f_k, f_k+1, ..., l_k-1, l_k$ of B_k^E) , we have

$$F_k(\pi) = \sum_{j \in B_k^E} w_j U_j = 0. \tag{4.3}$$

With respect to T-blocks B_k^T in π, it should be noticed that by Condition $C2$, for any permutation of jobs within B_k^T (i.e. in the positions $f_k, f_k+1, ..., l_k-1, l_k$ of B_k^T), all the jobs are tardy. Therefore, an optimal sequence of the jobs within B_k^T of π can be obtained, using well-known Weighted Shortest Processing Time (WSPT) rule, proposed by Smith [24]. The WSPT rule creates an optimal sequence of the jobs in the non-increasing order of the ratios w_j/p_j.

Let $\overrightarrow{B}_k^T$ denote a T-block of the jobs from B_k^T of π ordered by the WSPT rule, then we have

$$F_k(\pi) = \sum_{j \in B_k^T} w_j U_j \geq \sum_{j \in \overrightarrow{B}_k^T} w_j U_j = \overrightarrow{F}_k(\pi). \tag{4.4}$$

where the jobs from B_k^T are ordered in any permutation.

Fundamental Block Properties of the MTWL problem are derived from the following Theorem.

Theorem 4.1. *Let $\pi \in \Pi$ be any permutation with blocks $B_1, B_2, ..., B_m$, and let the jobs of each T-block of π to be ordered according to the WSPT rule. If the permutation β has been obtained from π by an interchange of jobs that $F(\beta) < F(\pi)$, then in β*

(i) at least one job from B_k precedes at least one job from blocks $B_1, B_2, ..., B_{k-1}$, for some $k \in \{2, 3, \dots, m\}$, or
(ii) at least one job from B_k succeeds at least one job from blocks $B_{k+1}, B_{k+2}, ..., B_m$ for some $k \in \{1, 2, \dots, m-1\}$.

Proof. Without loss of generality and for the simplicity of denotation one can assume that $\pi(i) = i$. Thus π takes the form

$$\pi = (1, 2, ..., l_1, f_2, f_2+1, ..., l_2, ..., f_k, f_k+1, ..., l_k, ..., f_m, f_m+1, ..., n),$$

where

$$B_k = (f_k, f_k+1, ..., l_k), \qquad k = 1,2,...,m,$$

and

$$F(\pi) = \sum_{k=1}^{m} F_k(\pi) = \sum_{k=1}^{m} \sum_{j=f_k}^{l_k} w_j U_j.$$

Suppose that the theorem is false, and let β be an arbitrary permutation of the following form

$$\beta = (x_1^1, x_2^1, ..., x_{t_1}^1, x_1^2, x_2^2, ..., x_{t_2}^2, ..., x_1^k, x_2^k, ..., x_{t_k}^k, ..., x_1^m, x_2^m, ..., x_{t_m}^m), \qquad (4.5)$$

where $X_k = (x_1^k, x_2^k, ..., x_{t_k}^k)$, $k = 1,2,...,m$, is any permutation of $(f_k, f_k+1, ..., l_k)$, i.e. any permutation of the jobs from block B_k of π, then it is easy to verify that the thesis of the theorem does not hold for β.

Now, for permutation β, we have

$$F(\beta) = \sum_{k=1}^{m} F_k(\beta) = \sum_{k=1}^{m} \sum_{j \in X_k} w_j U_j.$$

Since $\{x_1^k, x_2^k, ..., x_{t_k}^k\} = \{f_k, f_k+1, ..., l_k\}$ for $k = 1,2,...,m$, then

$$F_k(\pi) = F_k(\beta)$$

whenever the permutations of jobs within both X_k and B_k are the same. Hence, since the jobs from X_k of β are ordered in any permutation, then, for each T-block $\overrightarrow{B}_k^T$ of π, using (4.4), we get

$$F_k(\beta) = \sum_{j \in X_k} w_j U_j \geq \sum_{j \in \overrightarrow{B}_k^T} w_j U_j = \overrightarrow{F}_k(\pi),$$

where the jobs from $\overrightarrow{B}_k^T$ are ordered by the WSPT rule, according to the assumption of the theorem.

Further, for each E-blocks B_k^E of π, using (4.3), we have

$$F_k(\beta) = F_k(\pi) = 0.$$

Therefore, for each block B_k of π, we get

$$F_k(\beta) \geq F_k(\pi).$$

Hence, for any permutation β given by (4.5), we get

$$F(\beta) = \sum_{k=1}^{m} F_k(\beta) \geq \sum_{k=1}^{m} F_k(\pi) = F(\pi),$$

which contradicts the assumption of the theorem. □

Note that Theorem 4.1 provides the necessary condition to obtain a permutation β from π such that $F(\beta) < F(\pi)$.

In any permutation of jobs from T-block B^E, each job is early in permutation π. Using this property we present the algorithm of determining the first E-block in π.

Algorithm AE-block

```
  Input: permutation $\pi = (\pi(1), \pi(1), \dots, \pi(n))$;
  Output: subpermutation ($E$-block) $B^E = (\pi(l), \pi(l+1), \dots, \pi(k-1), \pi(k))$;
      Let $\pi(l)$ be the first job in $\pi$ such, that $C_{\pi(l)} \le d_{\pi(l)}$;
      $B^E \leftarrow \pi(l)$;  $k \leftarrow l$;
      while $|B^E| = k - l + 1$ and $k < n$ do
      begin
        if $(C_{\pi(k+1)} \le d_{\pi(k+1)})$ and
        (if (for all $\pi(i) \in B^E$,  $C_{\pi(i)} \le d_{\pi(k+1)}$)) then $B^E \leftarrow B^E \cup \{\pi(k+1)\}$;
        $k \leftarrow k + 1$
      end.
```

Computational complexity of this algorithm is $O(n)$.

In any permutation of jobs from T-block B^T, each job in permutation π is late. Similarly, like for T-block, on the basis of the above definition, we announce the algorithm of determining the first T-block in the permutation π.

Algorithm AT-block

```
  Input: permutation $\pi = (\pi(1), \pi(1), \dots, \pi(n))$;
  Output: subpermutation ($T$-block) $B^T = (\pi(v), \pi(v+1), \dots, \pi(r-1), \pi(r))$;
      Let $\pi(t)$ be the first job in $\pi$ such, that $C_{\pi(v)} > d_{\pi(v)}$.
      $B^T \leftarrow \pi(v)$;   $P_{first} \leftarrow C_{\pi(v)} - p_{\pi(v)}$;   $r \leftarrow t$;
      while $|B^T| = r - v + 1$ and $r < n$ do
      begin
        if $P_{first} + p_{\pi(r+1)} > d_{\pi(r+1)}$ then $B^T \leftarrow B^T \cup \{\pi(r+1)\}$;
        $r \leftarrow r + 1$
      end.
```

Computational complexity of the above algorithm is $O(n)$.

Considering in turn, jobs in permutation π (beginning from $\pi(1)$) and applying respectively algorithm AE-block or AT-block, we will break π into E and T blocks. Computational complexity of this break is $O(n)$.

4.3.1 The Tabu Search Method

Rough algorithms are used mainly to solve NP-hard problems of discrete optimization. Solutions determined by these algorithms are found to be fully satisfactory (very often they differ from the best known solutions approximately less than a few percent). One of realizations of constructive methods of these algorithms is tabu search, whose basic elements are

- *movement* – a function which transforms one task into another,
- *neighborhood* – a subset of acceptable solutions set,
- *tabu list* – a list which contains attributes of a number of examined solutions.

Let $\pi \in \Pi$ be a starting permutation, L_{TS} a tabu list and π^* the best solution found so far.

Algorithm Tabu Search (*TS*)

```
1   repeat
2      Determine the neighborhood $\mathcal{N}(\pi)$ of permutation $\pi$;
3      Delete from $\mathcal{N}(\pi)$ permutations forbidden by the list $L_{TS}$;
4      Determine a permutation $\delta \in \mathcal{N}(\pi)$, such that
5            $F(\delta) = \min\{F(\beta): \ \beta \in \mathcal{N}(\pi)\}$;
6      if ( $F(\delta) < F(\pi^*)$ ) then
7            $\pi^* := \delta$;
8      Place attributes $\delta$ on the list $L_{TS}$;
9      $\pi := \delta$
10  until (the completion condition).
```

4.3.2 Movement and Neighborhood

Let us notice that Theorem 1 provides the necessary condition to obtain a permutation β from π such that $F(\beta) < F(\pi)$.

Let $\mathcal{B} = [B_1, B_2, \ldots, B_v]$ be an ordered partition of the permutation $\pi \in \Pi$ into blocks. If a job $\pi(j) \in B_i$ $(B_i \in \mathcal{B})$, therefore existang moves, which can improve goal function value, consist in reordering a job $\pi(j)$ before the first or after the last job of this block. Let $\mathcal{M}_j^{bf}$ and $\mathcal{M}_j^{af}$ be sets of such moves (obviously $\mathcal{M}_1^{bf} = \mathcal{M}_v^{af} = \oslash$). Therefore, the neighborhood of the permutation $\pi \in \Pi$ has the form of

$$\mathcal{M}(\pi) = \bigcup_{j=1}^{n} \mathcal{M}_j^{bf} \cup \bigcup_{j=1}^{n} \mathcal{M}_j^{af}. \tag{4.6}$$

The *neighborhood* of the π is a set of permutations

$$\mathcal{N}(\pi) = \{m(\pi): \ m \in \mathcal{M}(\pi)\}. \tag{4.7}$$

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.

By implementing an algorithm from the neighborhood permutations whose attributes are on the tabu list L_{TS} are removed.

Generally, in our algorithm, for the given initial permutation, we identify the blocks (if there is more than one partition of the permutation into blocks, any of them can be used), and order the jobs of each T-block according to the WSPT rule. Then, for the resulting (basic) permutation π, we calculate $F(\pi)$, create the set of moves ME, compound move $\widehat{v}$, and the permutation $\pi_{\widehat{v}}$. Next, the search process of algorithm is repeated for the new initial permutation $\pi_{\widehat{v}}$ until a given number of iterations is reached. According to the philosophy of tabu search, the compound move cannot contain the single moves with a status tabu; these moves are not allowed.

4.3.3 The Tabu Moves List

To prevent a cycle from arising some attributes of each movement are put on the list of tabu moves.

In our algorithm we use the cyclic tabu list defined as a finite list (set) L_{TS} with length $LengthT$ containing ordered triplets. The list L_{TS} is a realization of the short-term search memory. If a move $v = (x, y)$ is performed on permutation π, then, a triplet $(\pi(x), y, F(\pi_v))$ is added to L_{TS}. If the compound move $\widehat{v}$ is performed, then the triplet corresponding to each move from $\widehat{v}$ is added to the tabu list. Each time before adding a new element to L_{TS}, we must remove the oldest one. With respect to a permutation π, a move $v = (x, y)$ is forbidden i.e. it has *tabu* status, if there is a triplet (r, s, ϕ) in L_{TS} such that $\pi(x) = r$, $y = s$, and $F(\pi_v) \geq \phi$.

As mentioned above, our algorithm uses a tabu list with dynamic length. This length is changed, as the current iteration number $iter$ of algorithm increases, using a "pick" that can be treated as a specific disturbance (diversification).

This kind of tabu list was employed on those very fast tabu search algorithms proposed by Grabowski and Wodecki, where it was successfully applied to the classical flow shop and job shop problems [7, 8]. Here, we extend this component of algorithm in the original form [7], to the problem considered. In this tabu list, length $LengthT$ is a cyclic function shown in Figure 4.2, and defined by the expression

$$LengthT = \begin{cases} LTS, & \text{if } W(l) < iter \leq W(l) + h(l), \\ LTS + \psi, & \text{if } W(l) + h(l) < iter \leq W(l) + h(l) + H, \end{cases}$$

where $l = 1, 2, \ldots$ is the number of the cycle, $W(l) = \sum_{s=1}^{l} h(s-1) + (l-1) \times H$ (here $h(0) = 0$). Further, H is the width of the pick equal to ψ, and $h(l)$ is the interval between the neighbour pick equal to $3 \times LTS$. If $LengthT$ decreases, then a suitable number of the oldest elements of tabu list L_{TS} is deleted and the search process is continued. The LTS and ψ are tuning parameters which are to be chosen experimentally.

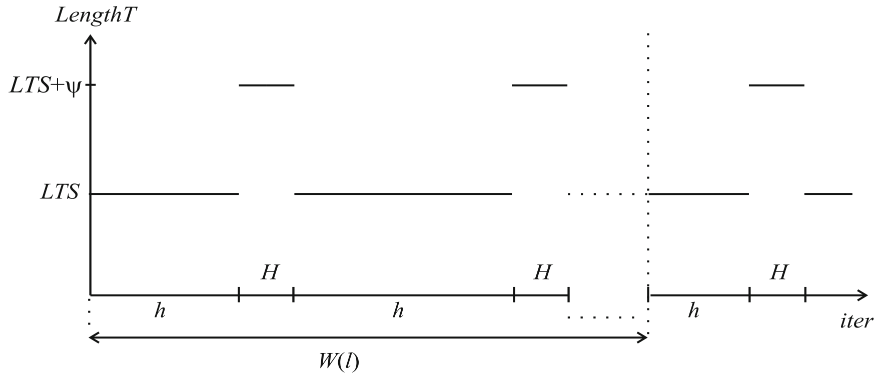

Fig. 4.2. Dynamic tabu list

4.4 Stochastic Processing Times

In literature there has been explored a problem of arrangement with random times tasks, mainly with a normal or a uniform distribution (van den Akker Hoogeveen [25]) and an exponential one (Pinedo [19]).

We consider a set of n jobs $J = \{1, 2, \dots, n\}$ to be processed on a single machine. The job processing times $\widetilde{p}_i$ $(i = 1, 2, \dots, n)$ are independent random variables. Then for a determined sequence of occurring tasks in a permutation π the completion time of a task performance $\widetilde{C}_{\pi(i)} = \sum_{j=1}^{i} \widetilde{p}_{\pi(j}$ is a random variable. Random variables are delays $\widetilde{U}_{\pi(i)} = 0$ when $\widetilde{C}_{\pi(i)} \leq d_{\pi(i)}$ and $\widetilde{U}_{\pi(i)} = 1$ when $\widetilde{C}_{\pi(i)} > d_{\pi(i)}$ as well as the goal function

$$\widetilde{\mathscr{F}}(\pi) = \sum_{i=1}^{n} w_{\pi(i)} \widetilde{U}_{\pi(i)}. \tag{4.8}$$

In the tabu search (TS) algorithm selecting the best element from an environment (instruction 5) is comparable with the goal function values. As (4.8) is a random variable we replace it with a convex combination of the expected value and the standard deviation

$$\mathscr{W}(\pi) = c \cdot E(\widetilde{\mathscr{F}}(\pi)) + (1 - c) \cdot D(\widetilde{\mathscr{F}}(\pi)) \;\; (c \in [0, 1]). \tag{4.9}$$

In the probabilistic version of an algorithm in a place of a goal function $\mathscr{F}$ (instructions 5 and 6) there should be placed a function W defined in (4.9).

4.4.1 Normal Distribution

Let's assume that the times of tasks performance $\tilde{p}_i$, $i \in J$ are independent random variables with a normal distribution with an average m_i and a standard deviation σ_i i.e. $\tilde{p}_i \sim N(m_i, \sigma_i)$. Then, the time of completion of a task performance $\tilde{C}_i$ (performed as i-th in a sequence) is a random variable with a normal distribution with an average $m^{(i)} = \sum_{j=1}^{i} m_j$ and a standard deviation $\sigma^{(i)} = \sqrt{\sum_{j=1}^{i} \sigma_j^2}$.

The delay of a task $\tilde{U}_i$ (according to (4.6)) is also a random variable whose expected value is:

$$E(\tilde{U}_i) = \sum_x x * P(\tilde{U}_i = x) = 0 * P(\tilde{C}_i \le d_i) + 1 * P(\tilde{C}_i > d_i) = 1 - \Phi(\delta^i),$$

where Φ is a cumulative distribution function of a normal distribution $N(0,1)$, and a parameter $\delta^i = \frac{d_i - m^{(i)}}{\sigma^{(i)}}$.

In that case, the expected value of a goal function (4.8) equals

$$E(\widetilde{\mathscr{F}}(\pi)) = E(\sum_{i=1}^{n} w_i * \tilde{U}_i) = \sum_{i=1}^{n} w_i * E(\tilde{U}_i) = \sum_{i=1}^{n} w_i * (1 - \Phi(\delta^i)). \qquad (4.10)$$

It's easy to notice that $E(\widetilde{U}_i^2) = 1 - \Phi(\delta^i)$, therefore

$$D^2(\widetilde{U}_i) = E(\widetilde{U}_i^2) - (E(\widetilde{U}_i)^2 = \Phi(\delta^i)(1 - \Phi(\delta^i)). \qquad (4.11)$$

Making use of (4.8) and (4.11) we get

$$D^2(\widetilde{\mathscr{F}}(\pi)) = \sum_{i=1}^{n} w_i \Phi(\delta^i)(1 - \Phi(\delta^i)) + 2 \sum_{i<j} w_i w_j \mathrm{cov}(\widetilde{U}_i, \widetilde{U}_j).$$

In line with the covariance definition

$$\mathrm{cov}\left(\widetilde{U}_i, \widetilde{U}_j\right) = E\left(\widetilde{U}_i, \widetilde{U}_j\right) - E\left(\widetilde{U}_i\right) E\left(\widetilde{U}_j\right) =$$

$$E\left(\widetilde{U}_i, \widetilde{U}_j\right) - \left(1 - \Phi\left(\delta^i\right)\right)\left(1 - \Phi\left(\delta^j\right)\right).$$

It's possible to prove easily that the expected value

$$E(\widetilde{U}_i, \widetilde{U}_j) = (1 - \Phi(\delta^i))(1 - \Phi(\gamma^j)),$$

where $\gamma^j = \frac{1}{\sqrt{1-\rho^2}} \left(\frac{d_j - m^{(j)}}{\sigma^{(j)}} - \rho \frac{d_i - m^{(i)}}{\sigma^{(i)}} \right)$ and $\rho^2 = \frac{(\sigma^{(i)})^2}{(\sigma^{(j)})^2}$. Thus,

$$\mathrm{cov}\left(\widetilde{U}_i, \widetilde{U}_j\right) = \left(1 - \Phi\left(\delta^i\right)\right)\left(1 - \Phi\left(\gamma^j\right)\right) - \left(1 - \Phi\left(\delta^i\right)\right)\left(1 - \Phi\left(\delta^i\right)\right) =$$

$$\left(1 - \Phi\left(\delta^i\right)\right)\left(\Phi\left(\delta^i\right) - \Phi\left(\gamma^j\right)\right).$$

Summarizing the goal function variance

$$D^2\left(\mathscr{F}(\pi)\right) = \sum_{i=1}^{n} w_i \Phi\left(\delta^i\right)\left(1 - \Phi\left(\delta^i\right)\right) + 2\sum_{i<j} w_i w_j \mathrm{cov}\left(\widetilde{U}_i, \widetilde{U}_j\right) =$$

$$\sum_{i=1}^{n} w_i \Phi\left(\delta^i\right)\left(1 - \Phi\left(\delta^i\right)\right) + 2\sum_{i<j} w_i w_j [\left(1 - \Phi\left(\delta^i\right)\right)\left(\Phi\left(\delta^i\right) - \Phi\left(\gamma^j\right)\right)]. \quad (4.12)$$

Therefore, to calculate the value $\mathscr{W}(\pi)$ determined in (4.9), (4.10) and (4.12) should be used.

4.4.2 The Erlang's Distribution

Let's assume that the time of tasks' performance has the Erlang's distribution $\widetilde{p}_i \sim \mathscr{E}(\alpha_i, \lambda),\ i \in J$. Then, the time of completing the task execution (in a permutation $\pi = (1, 2, \ldots, n)$) $\widetilde{C}_i = \sum_{j=1}^{i} \widetilde{p}_j \sim \mathscr{E}(\alpha_1 + \ldots + \alpha_i, \lambda)$.

Let $F_i(x) = F_{\tilde{p}_1 + \ldots + \tilde{p}_i}(x)$ be the cumulative distribution function of the time of completion of the i-th task $\widetilde{C}_i$ execution. The expected value

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

and

$$E(\widetilde{\mathscr{F}}(\pi)) = E\left(\sum_{i=1}^{n} w_i \widetilde{U}_i\right) = \sum_{i=1}^{n} w_i E\left(\widetilde{U}_i\right) = \sum_{i=1}^{n} w_i (1 - F_i(d_i)). \quad (4.13)$$

It's easy to observe that $E(\widetilde{U}_i^2) = 1 - F_i(d_i)$, that's why a variance

$$D^2(\widetilde{U}_i) = D^2(\sum_{i=1}^{n} w_i \widetilde{U}_i) = E(\widetilde{U}_i^2) - (E(\widetilde{U}_i))^2 = F_i(d_i)(1 - F_i(d_i)).$$

Thus,

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

The covariance $\mathrm{cov}(\widetilde{U}_i, \widetilde{U}_j)$ between variables $\widetilde{U}_i$ and $\widetilde{U}_j$ is calculated according to the formulae

$$\mathrm{cov}(\widetilde{U}_i, \widetilde{U}_j) = E(\widetilde{U}_i \widetilde{U}_j) - E(\widetilde{U}_i) E(\widetilde{U}_j).$$

Finally,

$$D^2(\widetilde{\mathscr{F}}(\pi)) = \sum_{i=1}^{n} w_i (F_i(d_i)(1 - F_i(d_i))) +$$

$$2\sum_{i<j} w_i w_j (FI + SI - (1-F_i(d_i))(1-F_j(d_j))), \tag{4.14}$$

where

$$FI = \int_{d_i}^{d_j}\int_{d_j-x}^{\infty} f_i(x)f_j(y)dydx, \quad \text{and} \quad SI = \int_{d_j}^{\infty}\int_{0}^{\infty} f_i(x)f_j(y)dydx.$$

As a result, in order to calculate the value of a function $\mathscr{W}(\pi)$ determined in (4.9) the formulae (4.13) and (4.14) should be applied.

4.5 The Algorithms' Stability

In this section we shall introduce a certain measure which let us examine the influence of the change of tasks' parameters on the goal function value (4.2) i.e. the solution stability.

Let $\delta = ((p_1, w_1, d_1), \ldots, (p_n, w_n, d_n))$ be an example of data (deterministic) for the MTWL problem. By $D(\delta)$ we denote a set of data generated from δ by a disturbance of times of tasks performance. A disturbance consists in changing these times on random determined values. Disturbed data $\gamma \in D(\delta)$ take the form of $\gamma = ((p'_1, w_1, d_1), \ldots, (p'_n, w_n, d_n))$, where the time of execution p'_i $(i = 1, \ldots, n)$ is a realization of a random variable $\tilde{p}_i$ in the Erlang's distribution $\mathscr{E}(\lambda, \alpha_i)$ (see Section 4.4.2), and thel expected value is $E\tilde{p}_i = p_i$.

Let $\mathscr{A} = \{\mathscr{AD}, \widetilde{\mathscr{AN}}\}$ where $\mathscr{AD}$ i $\widetilde{\mathscr{AN}}$ is the deterministic and the probabilistic algorithm respectively (i.e. solving examples with deterministic or random times of tasks' performance) for the MTWL problem. By π_δ we denote a solution (a permutation) determined by the algorithm $\mathscr{A}$ for a data δ. Then, let $\mathscr{F}(\mathscr{A}, \pi_\delta, \varphi)$ be the cost of tasks' execution (4.2) for the example φ in a sequence determined by a solution (a permutation) π_δ determined by the algorithm $\mathscr{A}$ for data δ. Then,

$$\Delta(\mathscr{A}, \delta, D(\delta)) = \frac{1}{|D(\delta)|}\sum_{\varphi \in D(\delta)} \frac{\mathscr{F}(\mathscr{A}, \pi_\delta, \varphi) - \mathscr{F}(\mathscr{AD}, \pi_\varphi, \varphi)}{\mathscr{F}(\mathscr{AD}, \pi_\varphi, \varphi)},$$

we call *the solution stability* π_δ (of an example δ) determined by the algorithm $\mathscr{A}$ on the set of disturbed data $D(\delta)$.

Because in our studies on the π_φ determining as the starting solution in the $\mathscr{A}$ algorithm (tabu search) we have adopted π_δ, we have

$$\mathscr{F}(\mathscr{A}, \pi_\delta, \varphi) - \mathscr{F}(\mathscr{A}, \pi_\varphi, \varphi) \geq 0\ .$$

Let Ω be a set of deterministic examples for the problem of tasks' arrangement. The *stability* rate of the algorithm $\mathscr{A}$ on the set Ω is defined in the following way:

$$S(\mathscr{A}, \Omega) = \frac{1}{|\Omega|}\sum_{\delta \in \Omega} \Delta(\mathscr{A}, \delta, D(\delta)). \tag{4.15}$$

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

4.6 The Calculation Experiments

Presented in this work algorithms were examined on many examples. The deterministic data were generated randomly (for a problem $1||\sum w_iT_i$ (see [27])) For a fixed number of tasks n ($n = 40, 50, 100, 150, \ldots, 500, 1000$) we have determined n tuples (p_i, w_i, d_i), $i = 1, 2, \ldots, n$, 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 critic lines are drawn from the range $[P(1-TF-RDD/2, P(1-TF+RDD/2]$ depending on the parameters RDD (relative range of due dates) and TF (average tardiness factor) from the set $\{0.2, 0.4, 0.6, 0.8, 1.0\}$, whereas $P = \sum_{i=1}^{n} p_i$. For every couple of parameters RDD, TF (there are 25 such couples) 5 examples have been generated. The whole deterministic data setΩ contains 1500 examples (125 for every n). On the basis of each example of deterministic data (p_i, w_i, d_i), $i = 1, 2, \ldots, n$, an example of probabilistic data $(\widetilde{p}_i, w_i, d_i)$ was determined, where $\widetilde{p}_i$ is a random variable (with the normal/standard or the Erlang's schedule) representing the time of task performance (detailed description in Section 4.4.2). The set we denote by $\widetilde{\Omega}$.

By initiating of each algorithm the starting permutation was the natural permutation $\pi = (1, 2, \ldots, n)$. In addition, the following parameters were used:

- the length of list of tabu moves: $\ln n$,
- the maximal number of iterations of an algorithm (*the completion condition*): $n/2$ or n,
- in the formula (4.9) parameter $c = 0.8$.
- dynamic parameters of tabu list: $h = \lceil n/4 \rceil$, $H = \lceil n/10 \rceil$, $LTS = \lceil \sqrt(n) \rceil$ and $\psi = \lceil \sqrt(n/4) \rceil$.

Calculations of the deterministic algorithm $\mathscr{AD}$ were made on the examples from the set Ω, whereas the probabilistic algorithm $\widetilde{\mathscr{AN}}$ (normal distribution) and $\widetilde{\mathscr{AE}}$ (Erlang's distribution) on examples from the set $\widetilde{\Omega}$.

First of all, the quality designated by the algorithms solutions was examined. In order to do that, for every solution of the example of deterministic data designated by algorithm $A = \{\mathscr{AD}, \widetilde{\mathscr{AN}}, \widetilde{\mathscr{AE}}\}$ relative error was calculated

$$\varepsilon(A) = \frac{\mathscr{F}_A - \mathscr{F}^*}{\mathscr{F}^*}, \tag{4.16}$$

where $\mathscr{F}_A$ is a value of a solution set by the algorithm A, and $\mathscr{F}^*$ a value of the solution set by a very good algoritm presented in the work [28]. The average relative error (for every set of data) was presented in Table 4.2. According to the expectations, the best appeared to be a deterministic algorithm $\mathscr{AD}$ (for every group of data the mean error equals 6%)). The error for the two left $\widetilde{\mathscr{AN}}$ and $\widetilde{\mathscr{AE}}$ algorithms is similar and equals 11% and 12% respectively.

In order to examine the algorithms stability (i.e. the determination of parameter's value (4.15)) for each example of deterministic data from the set Ω, 100 examples of disturbed data were generated (the way of generating is described in Section 4.5). Each of these examples was solved by the algorithm $\mathscr{AD}$. On the basis of these calculations the stability rates of all three algorithms were determined. The comparable results are presented in Table 4.2. After completing n iterations the average stability rate of an algorithm $S(\mathscr{AD},\Omega) = 0.36$, and the probabilistic coefficient(for times with the Erlang's schedule) $S(\widetilde{\mathscr{AE}},\Omega) = 0.07$.

It means that the disturbance of a solution determined by an algorithm $\mathscr{AD}$ causes worsening of the goal function value on average by about 36%. In case of the algorithm $\mathscr{AD}$ the worsening equals on average only 7%. As a result, we can state that the average deviation of the deterministic algorithm is about 5 bigger than the average deviation of the probabilistic algorithm. For the algorithm $\mathscr{AN}$ (with times with the normal distribution) the rate $S(\widetilde{\mathscr{AN}},\Omega) = 0.24$. It means it's smaller than the rate of the deterministic algorithm, however, nearly 4 times bigger than the probabilistic algorithm $\widetilde{\mathscr{AE}}$. The maximal mistake of an algorithm $\widetilde{\mathscr{AE}}$ does not exceed 41%, and the deterministic algorithm $\mathscr{AD}$ equals over 124%. Calculations for a bigger number of iterations were made as well. The stability rate of algorithms improved slightly. The n number of iterations of the algorithm based on the tabu

Table 4.2. Algorithms' deviation $\varepsilon(A)$ and stability $S(\mathscr{A},\Omega)$)

n	*Algorithm* $\mathscr{AD}$ (deterministic)		*Algorithm* $\widetilde{\mathscr{AN}}$ (normal distribution)		*Algorithm* $\widetilde{\mathscr{AE}}$ (Erlang's distribution)	
	Deviation	*Stability*	*Deviation*	*Stability*	*Deviation*	*Stability*
40	0.03	0.62	0.08	0.26	0.05	0.08
50	0.01	0.34	0.07	0.30	0.011	0.07
100	0.04	0.53	0.09	0.19	0.08	0.06
150	0.06	0.31	0.11	0.17	0.09	0.09
200	0.05	0.28	0.07	0.28	0.12	0.09
250	0.03	0.30	0.06	0.11	0.09	0.06
300	0.08	0.36	0.14	0.26	0.28	0.07
350	0.06	0.27	0.09	0.18	0.08	0.02
400	0.05	0.32	0.16	0.25	0.15	0.08
450	0.07	0.25	0.05	0.21	0.06	0.09
500	0.11	0.17	0.16	0.26	0.16	0.07
1000	0.14	0.48	0.16	0.36	0.17	0.09
Average	0.06	0.36	0.11	0.24	0.12	0.15

search method is very small. Due to this fact, the average time of calculations of one example, on a personal computer with a processor Pentium 2.6 GHz, does not exceed one second.

The calculational experiments proved explicitly that solutions determined by the probabilistic algorithm with times of tasks performance with the Erlang's distribution are the most stable.

4.7 Conclusion

In this work we presented a method of modelling uncertain data by means of random variables with the normal and the Erlang's distribution. The Erlang's distribution, as much better than other ones, represents times of tasks performance which in the course of realization are exposed to change (extending). Algorithms based on the tabu search method were presented for solving a certain single machine problem of tasks arrangement. The calculations experiments proved that an algorithm in which times of tasks performance are random variables with the Erlang's distribution is very stable. The relative average deviation for disturbed data by a small number of iterations and a short calculation time does not exceed 7%.

References

1. Beasley, J.E.: OR-Library: distributing test problems by electronic qmail. Journal of the Operational Research Society 41, 1069–1072 (1990), `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`
2. Bożejko, W., Rajba, P., Wodecki, M.: Arrangement Algorithms Stability with Probabilistic Parameters of Tasks. In: Proceedings of the 14th International Asia Pacific Conference on Computer Aided System Theory, Sydney, Australia, February 6-8 (2012)
3. Bożejko, W., Grabowski, J., Wodecki, M.: Block approach-tabu search algorithm for single machine total weighted tardiness problem. Computers & Industrial Engineering 50, 1–14 (2006)
4. Cai, X., Zhou, X.: Single machine scheduling with expotential processing times and general stochastic cost functions. Journal of Global Optimization 31, 317–332 (2005)
5. Dean, B.C.: Approximation algorithms for stochastic scheduling problems. PhD thesis, MIT (2005)
6. Garey, M.R., Johnson, D.S.: Scheduling tasks with nonuniform deadlines on two processor. Journal of ACM 23, 461–467 (1976)
7. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Computers and Operations Research 31, 1891–1909 (2004)
8. Grabowski, J., Wodecki, M.: A very fast tabu search algorithm the job shop problem. In: Rego, C., Alidaee, A. (eds.) Metaheuristic Optimization via Memory and Evolution; Tabu Search and Scatter Search, pp. 117–144. Kluwer Academic Publishers, Boston (2005)

9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling. Annals of Discrete Mathematics 5, 287–326 (1979)
10. Itoh, T., Ishii, H.: Fuzzy due-date scheduling problem with fuzzy processing times. Int. Trans. Oper. Res. 6, 639–647 (1999)
11. Józefowska, J.: Just-in-time scheduling. Springer, Berlin (2007)
12. Kise, H., Ibaraki, T., Mine, H.: A solvable case of the one-machine scheduling problem with ready times and due times. Operations Research 26, 121–126 (1978)
13. Karp, R.M.: Reducibility among Combinatorial Problems. In: Millerand, R.E., Thatcher, J.W. (eds.) Complexity of Computations, pp. 85–103. Plenum Press, NY (1972)
14. Lawler, E.L., Moore, J.M.: A Functional Equation and its Applications to Resource Allocation and Sequencing Problems. Management Science 16, 77–84 (1969)
15. Lawler, E.L.: A "pseudopolinomial" algorithm for sequencing jobsto minimize total tardiness. Annals of Discrete Mathematics 1, 331–342 (1977)
16. Lenstra, J.K., Rinnoy Kan, A.H.G.: Complexity results for scheduling chains on a single machine. European Journal of Operational Research 4, 270–275 (1980)
17. Monma, C.I.: Linear-time algorithms for scheduling on parallel processor. Operations Research 30, 116–124 (1982)
18. Moore, J.M.: An n-job, one machine sequencig algorithm for minimizing the number of late jobs. Menagement Science 15, 102–109 (1968)
19. Pinedo, M.: Stochastic scheduling with release dates. Operation Research 31, 559–572 (1983)
20. Potts, C.N., Van Wassenhove, L.N.: A Branch and Bound Algorithm for the Total Weighted Tardiness Problem. Operations Research 33, 177–181 (1985)
21. 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), 843–858 (1988)
22. Prade, H.: Using fuzzy set theory in a scheduling problem. Fuzzy Sets and Systems 2, 153–165 (1979)
23. Sahni, S.K.: Algorithms for Scheduling Independent Jobs. J. Assoc. Comput. Match. 23, 116–127 (1976)
24. Smith, W.E.: Various Optimizers for Single-Stage Production. Naval Research Logist Quartely 3, 59–66 (1956)
25. Van den Akker, M., Hoogeveen, H.: Minimizing the number of late jobs in a stochastic setting usinga chance constraint. Journal of Scheduling 11, 59–69 (2008)
26. Sourd, F., Kedad-Sidhoum, S.A.: A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem. Journal of Scheduling 11, 49–58 (2008)
27. Wodecki, M.: A Branch-and-Bound Parallel Algorithm for Single-Machine Total Weighted Tardiness Problem. J. Adv. Manuf. Tech. 37(9-10), 996–1004 (2008)
28. Wodecki, M.: A block approach to earliness-tardiness scheduling problems. Advanced Manufacturing Technology 40, 797–807 (2009)
29. Villareal, F.J., Bulfin, R.L.: Scheduling a Single Machine to Minimize the Weighted Number of Tardy Jobs. IEE Trans. 15, 337–343 (1983)
30. Vondrák, J.: Probabilistic methods in combinatorial and stochastic optimization. PhD thesis, MIT (2005)