Sampling Method for the Flow Shop with Uncertain Parameters

Pawel Rajba, Mieczyslaw Wodecki

Sampling method for the flow shop with uncertain parameters

Paweł Rajba and Mieczysław Wodecki

Institute of Computer Science, University of Wrocław
Joliot-Curie 15, 50-383 Wrocław, Poland
mieczyslaw.wodecki@uwr.edu.pl
pawel.rajba@uwr.edu.pl

Abstract. In the classic approach for optimization problems modelling well defined parameters are assumed. However, in real life problems we find ourself very often in a situation where parameters are not defined precisely. This may have many sources like inaccurate measurement, inability to establishing precise values, randomness, inconsistent information or subjectivity.
In this paper we propose a sampling method for solving optimization problems with uncertain parameters modeled by random variables. Moreover, by applying confidence intervals theory, the execution time has been significantly reduced. We will also show an application of the method for the flowshop problem with deadlines and parameters modeled by random variables with the normal distribution.

Keywords: flowshop with deadlines, uncertain parameters, tabu search, normal distribution

1 Introduction

Practical machine scheduling problems are numerous and varied. They arise in diverse areas such as flexible manufacturing systems, production planning, communication, computer design, etc. A scheduling problem consists in finding sequences of jobs on given machines with the objective of minimizing some function. In a simpler version of the problem, the flow shop scheduling, all jobs pass through all machines in the some order. In this paper, we deal with another specific version of the problem called a permutation flow shop scheduling problem where each machine processes the jobs in the same order ($F||w_iU_i$).

Research concerning problems of algorithms arrangement refers mainly to deterministic models ([1]). To solve such problems, which belong in the majority of cases to the NP-strongly hard class, rough algorithms are applied successfully ([2], [6], [8]). 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) very often it appears that certain parameters (e.g. the task completion time) are different from

the initial ones. By the lack of the solutions stability in the fixed schedule there may occur a big mistake, which makes such a schedule unacceptable. That's why a necessity exists 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 ([4], [9], [15]).

Problems of arrangements with uncertain data may be solved using methods based on elements of probability calculus ([13], [14], [5]). In this work we deal with the flow shop problem of tasks arrangement with the latest completion times and the minimalisation of the costs sum of tardy tasks ([7], [10]). On the basis of this problem the resistance to a random variable of constructive solutions of parameters according to tabu search metaheuristics is examined.

In this study, permutation flow shop scheduling problem of the typical situation of the flexible production systems which occupy a very important place in recent production systems are taken into consideration with random variables due dates.

2 Flowshop problem

Let $\mathcal{J} = \{1, \ldots, N\}$ be a set of jobs to be executed on M machines from the set $\mathcal{M} = \{1, \ldots, m\}$. At any given moment a specific machine can execute at most one job and all jobs needs to be executed without the preemption. Any job $j \in \mathcal{J}$ needs to be executed in sequence on every machine and if a job is being executed on the machine k then it means that it has been executed on machine the $k-1$ ($k = 2, 3, \ldots, M$). Jobs are executed in a given order determined by a permutation with a constraint that the permutation is applied on all machines. The execution of a job on a machine is named operation. In this section we consider flowshop problem with due dates defined as a set $(p_{i,j}, w_i, \widetilde{d}_i)$ ($i = 1, \ldots, N$, $j = 1, \ldots, M$) where $p_{i,j}$ are processing times of operations, w_i are weights for all jobs and $\widetilde{d}_i$ are due dates for all jobs, but they are defined as random variables with the distribution $N(d_i, c \cdot d_i)$.

Let Π be the set of all permutations of the set $\mathcal{J}$. For every permutation $\pi \in \Pi$ we define

$$C_{\pi(i),j} = \begin{cases} \sum_{k=1}^{i} p_{\pi(i),j}, & \text{dla } j = 1, \\ C_{\pi i,j-1} + p_{\pi(i),j}, & \text{dla } i = 1, j > 1, \\ \max\{C_{\pi i,j-1}, C_{\pi i-1,j}\} + p_{\pi(i),j}, & \text{dla } i > 1, j > 1, \end{cases}$$

as a completion time of execution job i on machine j in reference to permutation π.

The cost of execution of operations determined by permutation π is as follows

$$\sum_{i=1}^{n} w_{\pi(i)} \tilde{U}_{i\pi(i)}.$$

where

$$\tilde{U}_{i\pi(i)} = \begin{cases} 0 & \text{dla} \quad C_{\pi(i)} \leqslant \widetilde{d}_{i\pi(i)}, \\ 1 & \text{dla} \quad C_{\pi(i)} > \widetilde{d}_{i\pi(i)}. \end{cases}$$

We consider the optimization problem where the goal is to find a permutation $\pi^* \in \Pi$ which minimizes cost of execution of all operations:

$$\widetilde{W}(\pi^*) = \min_{\pi \in \Pi} \left(\sum_{i=1}^{n} w_{\pi(i)} \tilde{U}_{i\pi(i)} \right).$$

3 Tabu search

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 the 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 1 Tabu Search

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

3.1 Movement and Neighborhood

Let $\pi = (\pi(1), \ldots, \pi(n))$ be any permutation from the set Π. By π_l^k $(l = 1, 2, \ldots, k-1, k+1, \ldots, n)$ we denote the permutation obtained from π by a change of positions $\pi(k)$ with $\pi(l)$. We say, in such a case, that a π_l^k permutation was generated from π by a type of a swap move s_l^k (i.e. a permutation $\pi_l^k = s_l^k(\pi)$). Then, let $M(\pi(k))$ be a set of swap moves of an element $\pi(k)$, a set of all such movements

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

The neighborhood of an element $\pi \in \Pi$ is a set of permutations

$$\mathcal{N}(\pi) = \left\{ s_l^k(\pi) \colon s_l^k \in \mathcal{M}(\pi) \cap L(\pi) \right\},$$

where $L(\pi) = \{\pi(i) \colon C_{\pi(i)} > d_{\pi(i)}\}$ is a set of delay solutions in π.

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

3.2 The Tabu Moves List

To prevent a cycle from arising some attributes of each movement are put on the list of tabu moves. It's served by means of the FIFO queue. Performing a movement $s_j^r \in \mathcal{M}(\pi)$ (i.e. generating it from $\pi \in \Pi$ the π_j^r permutation) on the tabu list L_{TS} attributes of this movement, i.e. the triple $(\pi(r), j, \mathcal{F}(\pi_j^r))$ are put down.

Assuming that we examine a movement $s_l^k \in \mathcal{M}(\beta)$ generating from $\beta \in \Pi$ a permutation β_l^k. If on the list L_{TS} there is a triple (r, j, Ψ) such that $\beta(k) = r$, $l = j$ and $\mathcal{F}(\beta_l^k) \leq \Psi$, then such a movement is forbidden and removed from the set $\mathcal{M}(\beta)$.

4 Robustness

Due to the fact that we consider uncertain environment and the actual values are not known at the moment of the algorithm execution, we need a way to measure the quality of solutions. We assume that we have a set of reference test data and there are two algorithms: the examined one and the reference one (classic in our paper). The scenario of verification is as follows. For a specific test instance both algorithms propose solutions π_p (examined) and π_d (reference) which we expect to be robust. Then we generate a set of disturbed subinstances based on the test instance and for every subinstance we calculate the cost of execution with reference to π_p (cost w_p) and π_d (cost w_d). We also calculate an "almost optimal" solution for the subinstance w^*. Having that we calculate a relative error for all subinstances, then calculate relative error for all instances and by that we are able to take conclusion about the algorithm. We do that for both algorithms and compare the final values.

More formally, let define the basic robustness coefficient as a relative distance between examined and the reference solution, i.e. let w be a cost of "robust" solution (w_p or w_d) and w^* be the reference "almost optimal" solution cost. Then relative error

$$\delta = \frac{w - w^*}{w^*} 100\%$$

and it shows how many percent w is worse than w^*.

In some scenarios we need to compare the sets of values based on the disturbed data, so we propose an extension to the basic error definition. Let consider

s disturbed data instances, $w_1, \ldots, w_s$ be cost values obtained by examined algorithm and $w_1^*, \ldots w_s^*$ be reference cost values. Then we define extended relative error as follows:

$$\Delta = \frac{\frac{w_1+\ldots+w_n}{n} - \frac{w_1^*+\ldots+w_n^*}{n}}{\frac{w_1^*+\ldots+w_n^*}{n}} = \frac{(w_1 + \ldots + w_n) - (w_1^* + \ldots + w_n^*)}{w_1^* + \ldots + w_n^*}$$

Let ψ be a data instance, $\mathfrak{D}(\psi)$ be a set of disturbed data subinstances obtain from ψ based on the random variable $\widetilde{d}_i$ and

- A_{ref} be an algorithm which find the reference solution,
- A be the examined algorithm,
- $\pi_{M,x}$ be a solution obtained by algorithm $M \in \{A, A_{ref}\}$ for the problem instance x,
- $W(\pi_{M,x}, y)$ be the cost of instance y calculated by applying a solution $\pi_{M,x}$.

Then

$$\Delta(A, \psi, \mathfrak{D}(\psi)) = \frac{\sum_{\varphi \in \mathfrak{D}(\delta)} W(\pi_{A,\psi}, \varphi) - \sum_{\varphi \in \mathfrak{D}(\delta)} F(\pi_{A_{ref},\varphi}, \varphi)}{\sum_{\varphi \in \mathfrak{D}(\delta)} W(\pi_{A_{ref},\varphi}, \varphi)},$$

we define as solution robustness $\pi_{A,\psi}$ (obtained by the algorithm A for instance ψ) based on set of disturbed data $\mathfrak{D}(\psi)$.

Let Ω be a set of test data for the examined problem. Then by

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

we define as the robustness coefficient for the algorithm A on the set of test data Ω. The less the value is, the better the algorithm is, i.e. solutions obtained by the examined algorithm are more robust and random changes in the actual data don't affect significantly the final execution cost.

5 Sampling method

The idea of the method is as follows. In every tabusearch algorithm iteration we are testing different candidate solutions from the neighbourhood to find the best one and improve the current global best solution. Let assume an instance $(p_{i,j}, w_i, \widetilde{d}_i)$ and that we examine the candidate solution, a permutation π. Due to the fact that $\widetilde{d}_i$ is defined as random variable, we don't know the actual data that may come. What we propose in the sampling method is to simulate this actual scenario by testing the candidate solution on a sample of disturbed data generated from $\widetilde{d}_i$. We can describe the method in the following main steps:

1. Generate l vectors $\overline{d}^k = (\overline{d}_1^k, \ldots, \overline{d}_N^k)$, where $k \in \{1, \ldots, l\}$. By that we get l deterministic instances $(p_{i,j}, w_i, \overline{d}^k)$.

2. For every deterministic instance a cost is calculated based on the candidate solution π. By that we obtain sample $W_1^\pi, \ldots, W_l^\pi$.
3. We calculate a mean $\overline{x}$ and a standard deviation from s the sample which are used in the comparison by tabusearch. Of course less is better.

One can easily notice that in the above description we are missing the size of the sample, i.e. the value of l. We want the l to be as small as possible and meaningful on the other hand. To determine that we apply confidence intervals theory with the standard significance level $\alpha = 5\%$. Please note that we don't know the distribution of the sample $W_1^\pi, \ldots, W_l^\pi$. By that we apply the following variant of significance level formula:

$$\overline{x} - \mu_\alpha \frac{s}{\sqrt{l}} < m < \overline{x} + \mu_\alpha \frac{s}{\sqrt{l}}$$

where l is a sample size (at least 30), $\overline{x}$ is the sample mean, s is the sample standard deviation and μ_α is the value of random variable $N(0,1)$ under the condition:

$$\Phi(\mu_\alpha) = 1 - \frac{\alpha}{2}$$

what, according to the assumptions, provide $\mu_\alpha = 1,96$.

To sum up, the comparison criteria in the tabusearch method needs to be extended by the following code:

Function 2 Extention for comparison criteria in tabu search

1: $z \leftarrow 0$ ▷ current number of generated vectors
2: $l \leftarrow 30$
3: Generate $l - z$ vectors ($\overline{d}^k = (\overline{d}_1^k, \ldots, \overline{d}_N^k)$, where $k \in \{1, \ldots, l\}$. By that we have l instances $(p_{i,j}, w_i, \overline{d}^k)$
4: For every new instance calculate cost in context of candidate solution π. We obtain sample $W_1^\pi, \ldots, W_l^\pi$.
5: Calculate mean $\overline{x}$ and standard deviation s from sample.
6: $z \leftarrow l$
7: **if** $d \leqslant 5\%\overline{x}$ **or** $l > N \cdot M$ **then**
8: return $(\overline{x}, s)$
9: **else**
10: $l \leftarrow l + 10\%l$
11: Go to point 3

A remark: random sample don't need to be generated with every calculation of comparison criteria function. It is enough to generate it once for a specific data instance and this way it has been implemented in the computational experiments.

6 Computational experiments

In this section we describe the method for generating random data and elaborate the efficiency of the proposed method. The tabu search algorithm presented

in section 3 has been appropriately applied. As a reference algorithm we use classic deterministic implementation of tabusearch which we compare with the adaptation of tabusearch for the sampling method. Moreover, the following customization has been applied:

- start permutation: $\pi = (1, 2, \ldots, n)$,
- tabu list size: n,
- algorithm's iterations count: n.

In order to measure the efficiency of the proposed method we examine the computational complexity by checking samples' size and the robustness.

6.1 Test data

Both implemented algorithms have been examined on the commonly used reference test data which comes from [12] where he hired variants with jobs' numbers $N = 20, 50, 100$ and machines' numbers $M = 5, 10, 20$ what give 9 combinations. For every combination 10 examples are available so in total we have 90 test examples. Due to the fact that published examples consist of only processing times we needed to complete those examples by generating weights (w_i) and due dates (d_i). We applied the following schema: w_i has been generated uniformly from the range $[1, 10]$ and d_i has been generated uniformly from the range $[P(1 - T - R/2), P(1 - T + R/2)]$ where P denotes the best known value for the makespan for the $C_{\max}$ problem, $T = 0.3$ and $R = 0.3$ ([11]).

Based on the reference data we create the random instances $(p_{i,j}, w_i, \widetilde{d}_i)$ as formulated in the problem definition in section 2 where $p_{i,j}$ and w_i values come from previously described examples and $\widetilde{d}_i \sim N(d_i, c \cdot d_i)$ where $c \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. Having that for every random instance we generated 100 disturbed deterministic subinstances according to distribution of the random variable $\widetilde{d}_i$, in total we obtained $90 \cdot 5 \cdot 100 = 54000$ subinstances. The robustness coefficient 1 has been determined for both algorithms and results are presented in the next section.

6.2 Results

Before performing the complete set of tests we have checked whether sample measure on the mean only is good enough or is it worth introduce the standard deviation as well. It turned out the introducing standard deviation has a negligible influence on the final result, so all the tests have been executed with applying the mean only.

We executed tests for two main algorithms, but we examined the proposed method in more details to have a better insight into value that it brings. In the Tables 1, 2, 6.2, 6.2, 6.2, 6.2, 6.2, 6.2 there is a complete summary of all tested variants with main results. A quick observation leads to the conclusion that the proposed method gives much better results than the classic approach. Moreover, in all cases the results obtained by applying the sampling method are better

in the sense of statistical significance than results obtained by the classic way. The only thing which is puzzling is the fact that the more value c is the less advantage of sampling method is.

Table 1. Complete results summary for confidence intervals sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,79	0,26	2,04	75,48	89
0,10	1,36	0,66	1,08	79,26	89
0,15	2,00	1,14	0,75	83,17	89
0,20	2,62	1,64	0,59	90,16	89
0,25	3,31	2,27	0,46	89,54	88
0,30	3,66	2,60	0,41	91,67	88
Average	2,29	1,43	0,89	84,88	88,7

Table 2. Complete results summary for NM sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,82	0,25	2,25	661,11	90
0,10	1,38	0,63	1,21	661,11	90
0,15	1,99	1,08	0,85	661,11	89
0,20	2,63	1,59	0,66	661,11	90
0,25	3,24	2,11	0,53	661,11	90
0,30	3,61	2,47	0,46	661,11	88
Average	2,28	1,35	0,99	661,11	89,5

Another general observation is that for almost all test instances the sampling method gives better robustness than the classic approach. When the sample size is based on confidence intervals theory, we obtain the level $98,5\%$ of advantage. The best one is for the sample size $N \cdot M$ when we have the value $99,4\%$. Even for a small sample size $N \cdot M \cdot 0.03$ the sampling method gives the level $92,3\%$ of advantage, finally we lose the advantage for a very small sample size $N \cdot M \cdot 0.01$.

Let's take a closer look at the relationship between the classic approach, the sampling one with sample size NM and the sampling one with sample size based on confidence intervals based sample size. On the chart one can see the robustness level (Figure 1). We can easily observe that for all values of parameter

Table 3. Complete results summary for $0.3 \cdot NM$ sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,82	0,26	2,11	198,33	87
0,10	1,37	0,64	1,14	198,33	90
0,15	2,00	1,11	0,80	198,33	89
0,20	2,66	1,66	0,60	198,33	90
0,25	3,25	2,15	0,51	198,33	89
0,30	3,63	2,50	0,45	198,33	88
Average	2,29	1,39	0,94	198,33	88,8

Table 4. Complete results summary for $0.15 \cdot NM$ sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,81	0,27	2,00	99,22	89
0,10	1,37	0,65	1,12	99,22	90
0,15	2,00	1,14	0,75	99,22	89
0,20	2,64	1,69	0,57	99,22	89
0,25	3,25	2,21	0,47	99,22	88
0,30	3,67	2,60	0,41	99,22	88
Average	2,29	1,43	0,89	99,22	88,8

Table 5. Complete results summary for $0.1 \cdot NM$ sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,80	0,28	1,86	66,11	87
0,10	1,36	0,67	1,03	66,11	90
0,15	1,98	1,17	0,70	66,11	89
0,20	2,63	1,70	0,55	66,11	89
0,25	3,26	2,26	0,44	66,11	87
0,30	3,69	2,67	0,38	66,11	86
Average	2,29	1,46	0,83	66,11	88

Table 6. Complete results summary for $0.05 \cdot NM$ sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,79	0,30	1,66	33,11	88
0,10	1,36	0,72	0,89	33,11	88
0,15	1,99	1,26	0,58	33,11	85
0,20	2,68	1,84	0,46	33,11	85
0,25	3,30	2,41	0,37	33,11	83
0,30	3,77	2,89	0,30	33,11	83
Average	2,32	1,57	0,71	33,11	85,3

Table 7. Complete results summary for $0.03 \cdot NM$ sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,78	0,33	1,38	19,89	87
0,10	1,36	0,77	0,76	19,89	85
0,15	1,99	1,31	0,52	19,89	82
0,20	2,66	1,92	0,39	19,89	84
0,25	3,32	2,53	0,31	19,89	84
0,30	3,77	3,00	0,26	19,89	80
Average	2,31	1,64	0,6	19,89	83,7

Table 8. Complete results summary for $0.01 \cdot NM$ sample size

c	Classic	Sampling	Relatively	Sample size	#Better
0,05	0,74	0,48	0,52	6,67	73
0,10	1,33	1,01	0,32	6,67	72
0,15	1,98	1,61	0,23	6,67	70
0,20	2,66	2,26	0,18	6,67	71
0,25	3,31	2,96	0,12	6,67	65
0,30	3,82	3,52	0,08	6,67	62
Average	2,31	1,97	0,24	6,67	68,8

c the advantage of the sampling method is indisputable. We can also observe that we can get the best robustness when the sample size is NM.

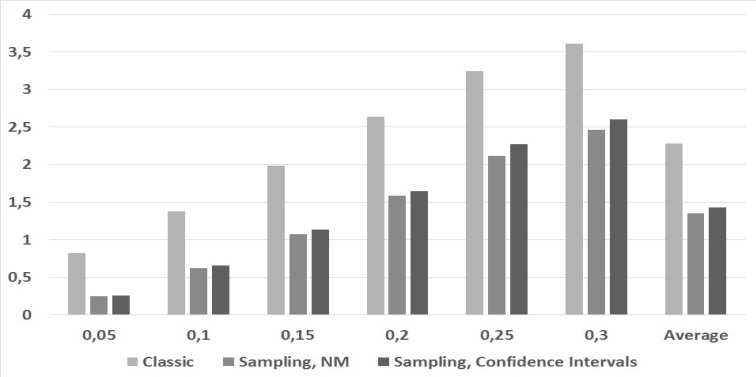

Fig. 1. Comparison of the robustness level with reference to the main methods

Finally, let's discuss the relationship between the algorithms' results with different sample size with reference to the robustness level (Figure 2). We can see that within the range $[1..0.1] \cdot NM$ the robustness levels are very close to each other. Only when the sample size is getting smaller (from $0.05NM$), the robustness level is getting significantly worse.

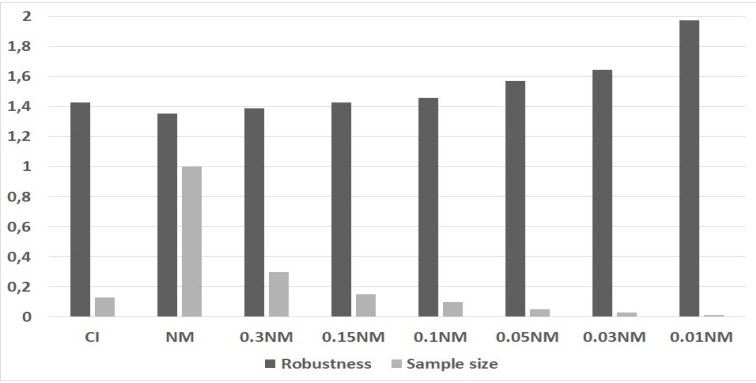

Fig. 2. Comparison of the robustness level with reference to the sample size

7 Conclusions

In this paper we proposed a sampling method to solve optimization problems with uncertain parameters modeled by random variables. By applying confidence intervals we wanted to keep a very good balance between the execution time and the robustness level. We have seen an application of the method for the flowshop problem with deadlines and parameters modeled by random variables with the normal distribution. Based on the performed computational experiments we can conclude that the proposed method gives substantially more robust solutions than the classic approach and by applying confidence interval theory we achieve the goal of keeping balance between the execution time and the robustness level.

References

1. Aarts A., Lenstra J.K., Local search in combinatorial optimization, John Wiley and Sons, 1997.
2. Bożejko W., Wodecki M., Solving Flow Shop Problem by Parallel Simulated Annealing, LNCS, Springer-Verlag, 2328, 2002, 236–244.
3. Bożejko W., Wodecki M., On the theoretical properties of swap multimoves, Operations Research Letters, 35/2, 2007, 227–231.
4. Bożejko W. Rajba P. Wodecki M., Scheduling Problem with Uncertain Parameters in Just in Time System, LNCS, Springer-Verlag, 2014, 456-467.
5. Dean B.C., Approximation algorithms for stochastic scheduling problems, PhD thesis, MIT, 2005
6. Grabowski J., Wodecki M., A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion, Computers and Operations Research, 31, 2004, 1891–1909.
7. Jang W., Klein C.M., Minimizing the expected number of tardy jobs when processing times are normally distributed, Operations Research Letters, 30, 2002, 100–106.
8. Nowicki E., Smutnicki C., A Fast tabu search algorithm for permutation flow shop problem, European Journal of Operational Research, 91, 1996, 160–175.
9. Rajba P., Wodecki M., Stability of scheduling with random processing times on one machine, Applicationes Mathematicae, 39, 2, 2012, 169–183.
10. Righter R., Stochastic Scheduling, in Stochastic Orders. M. Shaked and Shandhkumar (eds.), Academic Press, San Diego, 1994.
11. Sioud A., Gagné C., Gravel M., Minimizing Total Tardiness in a Hybrid Flexible Flowshop with Sequence Dependent Setup Times. In INFOCOMP 2014: The fourth International Conference on Advanced Communications and Computation, 2014, 13-–18
12. Taillard E., Benchmarks for basic scheduling problems, EJOR 64/2, 1993, 278–285.
13. Van den Akker M., Hoogeveen H., Minimizing the number of late jobs in a stochastic setting using a chance constraint, Journal of Scheduling 11, 2008, 59–69.
14. Vondrák J., Probabilistic methods in combinatorial and stochastic optimization. PhD thesis, MIT, 2005.
15. Zhu X., Cai X., General Stochastic Single-Machine Scheduling with Regular Cost Functions, Math. Comput. Modelling, Vol. 26, No. 3, 1997, 95–108.