Management and Production Engineering Review
Volume 6 • Number 3 • September 2015 • pp. 3–9
DOI: 10.1515/mper-2015-0021

DE GRUYTER OPEN

mper

EVOLUTIONARY ALGORITHM FOR MINMAX REGRET FLOW-SHOP PROBLEM

Michał Ćwik, Jerzy Józefczyk

Wroclaw University of Technology, Department of Informatics, Poland

Corresponding author:
Michał Ćwik
Wroclaw University of Technology
Department of Informatics
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
phone: (+48) 71 320-29-79
e-mail: Michal.Cwik@pwr.edu.pl

Received: 1 June 2015
Accepted: 30 July 2015

Abstract
The uncertain flow-shop is considered. It is assumed that processing times are not given a priori, but they belong to intervals of known bounds. The absolute regret (regret) is used to evaluate a solution (a schedule) which gives the minmax regret binary optimization problem. The evolutionary heuristic solution algorithm is experimentally compared with a simple middle interval heuristic algorithm for three machines instances. The conducted simulations confirmed the several percent advantage of the evolutionary approach.

Keywords
manufacturing, flow-shop, interval uncertainty, minmax regret, heuristic algorithms, evolutionary algorithms, simulation.

Introduction

The flow-shop, being an important task scheduling problem with many versions and a variety of vital applications, has been investigated for many decades. The vast literature exists in this area; see [1] as an example. The basic version of the classic flow-shop problem assumes the existence of a finite set of complex tasks which undergo the performance by means of a finite set of machines in such a way that each machine contributes to the elaboration of each task. A part of the task carried out by an individual machine is called an operation. So, each task consists of the same number of operations which is equal to the number of machines. The problem consists in the determination of the solution (schedule) of tasks in such a way to minimize the criterion evaluating the solutions. The makespan as the completion time of the last operation of the last task can serve as the criterion. It is assumed for classical flow-shop problems that all execution times of the operations are a priori known. This assumption occurs too restrictive for many real-world applications where it is extremely difficult to have exact values of these times. Abandoning this presumption leads to so called nondeterministic (non-classical) versions of the flow-shop problem. Different knowledge representations can be used to tackle with the non-fully known execution times. Probabilistic and fuzzy approaches are the most known (see [2, 3]). However, they require the existence and the availability of probability distributions and membership functions, respectively. Both characteristics imposed on the possible unknown values of execution times contain additional information. This extra information can be difficult or impossible to reach for a number of real-world cases, first of all for jobbing processes and activities. In the paper, the representation of known execution times in the form of intervals is supposed. It means that values of each execution time belong to the interval of known bounds. This interval representation of the uncertainty can be considered as the basic knowledge representation which does not require any additional characteristics of the set of possible values of execution times.

It is not possible to evaluate directly a solution for the uncertain version of flow-shop. In a consequence, the form of criterion is a second important

issue which ought to be resolved for the uncertain flow-shop problem. In fact, the majority of methods and techniques encountered in literature use criteria defined for deterministic versions and propose their different determinization pertaining to uncertain parameters to have the deterministic criteria as a consequence. The determinization can refer directly to criteria for deterministic versions or to some terms based on them. The absolute regret hereafter referred to regret, as such expression, is used in the paper to evaluate the uncertainty (see [4]). The determinization consisting in taking into account the worst possible values of execution times for the particular solution, performed by means of the operator 'maximum', gives, in a consequence, so called minmax regret optimization problems which are not farther on the uncertain problem. Many results, both general and particular, have been developed since that time. Let us only mention selected works: [5–9] where some general results as well as those related to many particular operations research optimization problems are reported. Some results have been also achieved for minmax regret flow-shop [10, 11]. However, the authors failed to find any approaches solving general flow-shop problem (or its permutation counterpart).

The reminder of the paper is organized as follows. Section 2 comprises the flow-shop problem formulation both for deterministic and uncertain version as well as presents some analytical properties of the latter version. In Sec. 3 we discuss the evolutionary based algorithm developed to solve the problem. Section 4 covers the algorithm tuning as well as its comparison to one of the most common approach to interval data uncertain problems. We conclude and define directions of further research in Sec. 5.

Problem statement and its properties

The mathematical model of the permutation version of flow shop problem is given in this section both for deterministic and uncertain case. The analysis of the worst-case scenario is also discussed.

Problem statement

Let us consider the flow-shop problem with a set $\mathbf{M} = \{M_1, M_2, \ldots, M_i, \ldots, M_m\}$ of m machines and a set $\mathbf{J} = \{J_1, J_2, \ldots, J_j, \ldots, J_n\}$ of n jobs. Indices i, j denote the current machine M_i, job J_j, correspondingly. Every job J_j is composed of m operations being its parts and performed by consecutive machines. Operation $O_{i,j}$ refers to the part of job J_j, which is performed by machineM_i. Operations within particular job J_j are performed by machines in the fixed order and constitute a sequence $(O_{1,j}, O_{2,j}, \ldots, O_{m,j})$. The order of machines denoted as $(M_1, M_2, \ldots, M_m)$ is given unlike the order of jobs undergoing the decision. The permutation version of the problem with unlimited buffers is considered. It means that every machine processes tasks in the same order and after being completed by the current machine any operation can leave it and wait if necessary for the service by the next machine at the buffer located there. Execution times $p_{i,j}$ of operations $O_{i,j}$ constitute matrix $p = [p_{i,j}]\ {}^{i=1,2,...,m}_{j=1,2,...,n}$. Solving the considered flow shop problem consists in the determination of permutation of jobs $\pi = (\pi_1, \pi_2, \ldots, \pi_j, \ldots, \pi_n) \in \Pi$, where $\pi_j \in \{1, 2, \ldots, n\}$ is the index of job performed as the jth in turn, and $\Pi = \{\pi : \pi_j \neq \pi_k,\ j,k \in \{1, 2, \ldots, n\},\ j \neq k\}$ is the set of all $n!$ permutations. The makespan $C_{\max}(p, \pi)$, being the time moment when the last job is completed, evaluates the permutation π. It can be calculated recurrently. Let us denote by $C_{i,\pi_j}(p, \pi)$ the time moment when machine M_i completes the execution of job π_j. This time moment can be calculated for $i = 2, 3, \ldots, m$ and $j = 2, 3, \ldots, n$ using the following recursive formula:

$$C_{i,\pi_j}(p,\pi) = p_{i,\pi_j} + \max[C_{i-1,\pi_j}(p,\pi), C_{i,\pi_{j-1}}(p,\pi)]. \tag{1}$$

Moreover, for j =1 and $i = 1, 2, \ldots, m$:

$$C_{i,\pi_1}(p,\pi) = \sum_{k=1}^{i} p_{k,\pi_1}. \tag{2}$$

Similarly for $i = 1$ and $j = 1, 2, \ldots, n$:

$$C_{1,\pi_j}(p,\pi) = \sum_{k=1}^{j} p_{1,\pi_k}. \tag{3}$$

So, $C_{\max}(p, \pi) = C_{m,\pi_n}(p, \pi)$. In a consequence, the deterministic version of the considered flow shop problem consists in the determination of permutation of tasks $\pi \in \Pi$ minimizing $C_{\max}(p, \pi)$ for given matrix of execution times p. The solution in the form of the optimal permutation π' and the corresponding optimal makespan $C'_{\max}(p)\Delta = C_{\max}(p, \pi')$ is sought.

In the paper the uncertain version of this problem is considered when it is assumed that the execution times are not given, but they belong to the intervals of known bounds. Namely, $p_{i,j} \in [\underline{p}_{i,j}, \overline{p}_{i,j}]$, $\underline{p}_{i,j} \leq \overline{p}_{i,j}$. Matrix p with all specified values of $p_{i,j}$ is now called a scenario. Each scenario is the element of the Cartesian product $P = [\underline{p}_{1,1}, \overline{p}_{i,j}] \times \ldots \times [\underline{p}_{m,n}, \overline{p}_{m,n}]$. To evaluate the decision π in the presence of uncertain processing times (parameters), the determinization is applied basing on the regret

$$C_{\max}(p,\pi) - C'_{\max}(p). \tag{4}$$

The operator 'max' is used for the determinization which leads to the performance index

$$z(\pi) = \max_{p\in P}[C_{\max}(p,\pi) - C'_{\max}(p)]. \tag{5}$$

The resulted discrete optimization problem can be formulated as follows:

For given: $\mathbf{M}$, $\mathbf{J}$, $\underline{p}_{i,j}$, $\overline{p}_{i,j}$, $i = 1, 2, ..., m$, $j = 1, 2, ..., n$, find: the optimal permutation π^* for which

$$z(\pi^*) = \min_{\pi\in\Pi}\left(\max_{p\in P}\left(C_{\max}(p,\pi) - C'_{\max}(p)\right)\right). \tag{6}$$

The problem (6) is at least NP-hard due to the fact that its deterministic counterpart is NP-hard for $m > 2$ (see [12]). The NP-hardness of (6) for $m = 2$ has been also proved in [13].

The NP-hardness of deterministic version of the problem implies the lack of approximate algorithm for (6). Indeed, let us assume that there exists k-approximate solution algorithm for (6) giving as the result the permutation $\widetilde{\pi}$ and $z(\widetilde{\pi})$. Then, $z(\widetilde{\pi}) \leq kz(\pi^*)$ which is true also for a special-case where: $\underline{p}_{i,j} = \overline{p}_{i,j}$ for $i = 1, 2, ..., m$, $n = 1, 2, ..., n$. For such scenario, $z(\pi^*) = 0$ and the uncertain problem (6) comes down to its deterministic counterpart which would be solved in polynomial time. It is not true unless P = NP.

Worst-case scenario analysis

Set $\mathbf{P}$ contains infinitely many scenarios. It is easy to show that the scenario p^π fulfilling the maximization in (5), referred to as the worst case scenario, is an extreme point scenario holding the following property:

$$\forall i,j\,; p_{i,j} = \underline{p}_{i,j} \vee p_{i,j} = \overline{p}_{i,j}. \tag{7}$$

We can observe that for any p and π the problem of finding $C_{\max}(p,\pi)$ can be presented as the problem of determining the longest path between vertexes $v_{0,0}$ and v_{m,π_n} in directed graph G(V, A) which is defined as follows. Number of vertexes in set V is equal to $mn + 1$. There is a vertex corresponding to each of mn operations in the problem (v_{i,π_j} is related to i-th operation of job scheduled as j-th). A dummy vertex $v_{0,0}$ is added to set V. Number of arcs in set A is $(m-1)(n-1)+1$. Graph G contains $(m-1)(n-1)+1$ arcs which weights in the form of the execution times are defined in Table 1.

Let us denote $S(p,\pi)$ as the set of tuples (i, j) which identify the weights that sum up to the longest path between vertexes $v_{0,0}$ and v_{m,π_n} (makespan of schedule π) under scenario p. Then:

$$C_{\max}(p,\pi) = \sum_{(i,j)\in S(P,\pi)} p_{i,\pi_j}. \tag{8}$$

Table 1
Arcs of graph G.

i	j	arcs	weights
0	0	$(v_{0,0}, v_{1,\pi_1})$	p_{1,π_1}
$1 \leq i < m$	$1 \leq j < n$	$(v_{i,\pi_j}, v_{i,\pi_{1+1}})$ $(v_{i,\pi_j}, v_{i+1,\pi_j})$	$p_{i,\pi_{j+1}}$ $p_{i,\pi_{j+1}}$
m	$1 \leq j < n$	$(v_{i,\pi_j}, v_{i,\pi_{1+1}})$	$p_{i,\pi_{j+1}}$
$1 \leq i < m$	n	$(v_{i,\pi_j}, v_{i+1,\pi_j})$	p_{i+1,π_j}
m	n	none	n/a

We can denote the regret using (8):

$$\begin{aligned} &C_{\max}(p,\pi) - C_{\max}(p,\pi') \\ &= \sum_{(i,j)\in S(p,\pi)} p_{i\pi} - \sum_{(i,j)\in S(p,\pi')} p_{i\pi'_j}. \end{aligned} \tag{9}$$

Therefore, we can see that scenario p^π maximizing the above difference has to be the extreme point scenario.

Moreover, p^π has to have the following form:

$$\begin{aligned} &\forall (i,j) \in S(p^\pi,\pi)\backslash S(p^\pi,\pi') : p_{i\pi_j} = \overline{p}_{i,\pi_j}, \\ &\forall (i,j) \in S(p^\pi,\pi')\backslash S(p^\pi,\pi) : p_{i\pi_j} = \underline{p}_{i,\pi_j}. \end{aligned} \tag{10}$$

Following the above considerations, no polynomial algorithm of determining the worst-case scenario has been identified. However, we can observe that the worst-case scenario can be denoted as:

$$\begin{aligned} &\forall (i,j) \in \widetilde{S}(p,\pi) : p_{i\pi_j} = \overline{p}_{i,\pi_j}, \\ &\forall (i,j) \notin \widetilde{S}(p,\pi) : p_{i\pi_j} = \underline{p}_{i,\pi_j}. \end{aligned} \tag{11}$$

where $\widetilde{S}(p,\pi)$ is a set that identifies weighs of one of the possible paths between vertexes $v_{0,0}$ and v_{m,π_n}. Since no polynomial algorithm of identifying this path has been developed, all of the possible paths need to be checked to determine the maximum regret for each schedule. Looking at the structure of graph G, we can observe that the number of paths that need to be checked is $((m+n-2)!) / ((m-1)!(n-1)!)$.

This number will grow exponentially according to the size of the problem. However, it is significantly smaller than 2^{mn} as the number of all extreme-point scenarios. For instance, for the problem with 3 machines and 4 jobs there are 4096 possible extreme-point scenarios, while there are only 10 possible paths between vertexes $v_{0,0}$ and v_{m,π_n}.

Lower bound and upper bound for the maximum regret

As stated above, solving the deterministic case of the flow shop problem is NP-hard. Therefore, we will

estimate the optimal value of $z(\pi)$ using lower bound $z_{LB}(\pi)$ and upper bound $z_{UB}(\pi)$ functions. To calculate the lower bound of $z(\pi)$, we use the NEH heuristic elaborated for solving the deterministic permutation flow shop problem. NEH is the simple procedure which constructs good quality solutions (see [14]) in a time-effective manner.

Let us denote π^{NEH} as the solution returned by the above heuristic. Then we can see that the $C_{\max}(p, \pi^{\mathrm{NEH}})$ is the upper bound of $C'_{\max}(p)$, and we can define the lower bound of $z(\pi)$:

$$z_{LB}(\pi) = \max_{p \in \mathbf{P}} \left(C_{\max}(p, \pi) - C_{\max}(p, \pi^{\mathrm{NEH}}) \right). \quad (12)$$

In order to define the upper bound of $z(\pi)$, we estimate the lower bound of $C'_{\max}(p)$. We can observe that for each machine k the following inequality holds:

$$\sum_{j=1}^{n} p_{k,j} \leq C'_{\max}(p). \quad (13)$$

Moreover, before the k-th machine starts processing operation at least one job needs to be processed on all machines indexed from 1 to $k-1$, unless $k = 1$. The processing time of these operations is $\min_j \sum_{i=1}^{k-1} p_{i,j}$. On the other hand, after the i-th machine completes the processing, there is at least one job that needs to be processed on machines indexed from $k+1$ to m, unless $k = m$. The processing time of these operations is $\min_j \sum_{i=k+1}^{m} p_{i,j}$. Thus, we can observe that for each k the following inequality holds:

$$\min_j \sum_{i=1}^{k-1} p_{ij} + \sum_{j=1}^{n} p_{kj} + \min_j \sum_{i=k+1}^{m} p_{ij} \leq C'_{\max}(p). \quad (14)$$

To obtain even tighter bound, we observe that the job which is processed before the k-th machine must not be the same as the job that will be processed after the k-th machine completes processing operations. Secondly, as the fact that (14) holds for all machines we can take the highest value. Therefore, we can define the lower bound of the deterministic flow-shop problem:

$$C_{\max,LB}(p) = \max_{k=\overline{1,m}} \cdot \left(\min_{j=\overline{1,n}} \left(\sum_{i=1}^{k-1} p_{i,j} \right) + \sum_{j=1}^{n} p_{k,j} + \min_{\substack{l=\overline{1,n} \\ l \neq j}} \left(\sum_{i=k+1}^{m} p_{i,l} \right) \right). \quad (15)$$

So, we can define the upper bound of $z(\pi)$:

$$z_{UB}(\pi) = \max_{p \in \mathbf{P}} \left(C_{\max}(p, \pi) - C_{\max,LB}(p) \right). \quad (16)$$

Proposed solution method

In this section, we present the algorithm proposed to solve the uncertain problem considered. All elements of the evolutionary approach are discussed in detail and chosen subprocedures are also described. The input of the algorithm is two matrices of size $n \times m$ containing values of lower/upper bounds of all uncertainty intervals. The output is a permutation of jobs π^{EVO}. The problem is not only NP-hard, but calculating the objective function for any solution requires solving NP-hard deterministic problems. Therefore, it is impossible to efficiently solve this problem using any exact algorithms e.g. solvers or branch and bound methods even for small instances. Because of that, the following evolutionary algorithm has been proposed.

A solution is represented by a sequence of numbers from 1 to n. The order of these numbers implies the permutation it represents.

A common approach to building evolutionary based algorithms is using the objective function of the problem as the fitness function. In this case, as stated before, the objective function $z(\pi)$ is too complex to consider it as the fitness function. We use its upper bound $z_{UB}(\pi)$ instead.

To generate the initial population, we generate 10 random permutations. In addition, we solve the given problem using the MIH algorithm described in Subsec. 4.3. The obtained solution undergoes 9 random mutations to obtain another 10 solutions. Therefore, we obtain initial population composed of 10 random permutations, one MIH-solution and its 9 mutations.

Standard crossover algorithms cannot be applied, because the permutation representation for chromosomes is used. Therefore, a well described in literature order crossover operator was chosen, see Golberg (1989). This operator returns a pair of child chromosomes for each pair of parent chromosomes given and ensures that the children also represent a feasible permutation. Therefore, no repair algorithm is required. For the crossover operator, there is a P_{cross} parameter defined being the probability of crossing over two selected chromosomes. This parameter requires tuning to ensure the best algorithm performance.

A simple mutation operator has been introduced. Firstly, it is randomly determined if the chromosome undergoes the mutation. The probability of the mutation P_{mut} is considered another parameter of the algorithm which is tuned. If the chromosome is decided to undergo the mutation, two random genes are swapped.

All chromosomes from the population are selected to generate a new generation. The population is sorted by nonincreasing fitness function values. First two chromosomes are removed from the list and the result of their crossover is added to the new population. This process is repeated until the list is empty.

The algorithm terminates when after 5 subsequent iterations no correction is being observed between the best chromosomes from each population. The best chromosome of the last generation is returned as the solution.

Computational experiments

In this section, we introduce the method of generating random instances of the problem. Then, we present the comparison of the evolutionary algorithm (EVO) to the middle-interval heuristic (MIH).

Instance generation

There are no benchmark problems defined in the literature. Problem instances are generated by using two constants. For each of mn uncertain parameters, the lower bound $\underline{p}_{i,j}$ is randomly generated from the finite interval $[0,K]$ with uniform distribution. The upper bound $\overline{p}_{i,j}$ is than generated from the finite interval $[\underline{p}_{i,j}, \underline{p}_{i,j} + C]$ also with uniform distribution.

Algorithm tuning

There are two parameters for this algorithm that were tuned: crossover probability P_{cross} and mutation probability P_{mut}. These parameters were tuned one at a time with fixed value of the other. The algorithm was tuned using small instances of the problem ($m = 3$ and $n = 6$). Values of K and C were fixed to $K = 100$ and $C = 50$. All parameters were tuned using 10 instances generated in the described way. The proposed algorithm is random, therefore for each problem instance and each fixed value of tuned parameter, it was executed 5 times. Average values of $z(\pi^{\text{EVO}})$ are considered.

The probability of mutation was tuned first with the P_{cross} fixed to 0.75. According to the obtained results given in Fig. 1, $P_{\text{mut}} = 0.15$ was used as the tuned value and a fixed value for tuning P_{cross}. Taking into account the results presented in Figs. 1 and 2, the parameters of the evolutionary algorithm were fixed to $P_{\text{mut}} = 0.15$ and $P_{\text{cross}} = 0.85$.

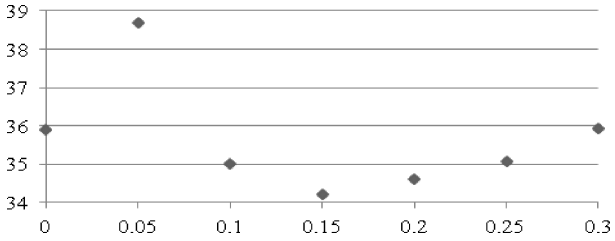

Fig. 1. Average $z(\pi^{\text{EVO}})$ for different values of P_{mut}.

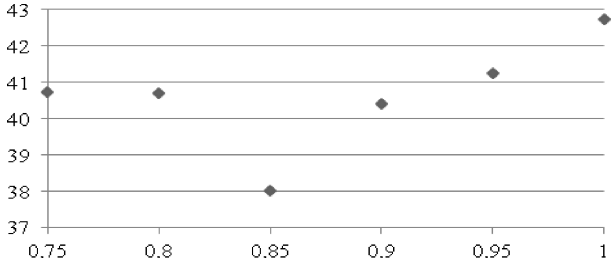

Fig. 2. Average $z(\pi^{\text{EVO}})$ for different values of P_{cross}.

Algorithm evaluation

The proposed evolutionary algorithm was compared to the middle-interval heuristic (MIH) which is a common approach to deal with interval data uncertainty [7]. The input of this algorithm is two matrices of size $n \times m$ containing values of lower/upper bounds of all uncertainty intervals. The output is a permutation of jobs π^{MIH}. This algorithm yields a solution to any interval data uncertain problem by solving the deterministic counterpart which is generated by substituting all uncertain parameters of the problem with its intervals middle points:

$$p_{i,j}^{\text{MIH}} = \frac{\underline{p}_{i,j} + \overline{p}_{i,j}}{2}.$$

The resulted deterministic problem is also NP-hard. Therefore, the NEH algorithm is applied as the solution tool. The solution of deterministic problem π^{MIH} is returned as the solution of the uncertain problem.

To compare the proposed evolutionary algorithm with the MIH algorithm, we generated 10 random instances with the manner described in Subsec. 4.1 for $m = 3$, $n = 25$, $K = 100$, $C = 50$. Instances of the problem with less jobs were generated by considering only a fixed number of jobs from each generated instance. Problems with more than three machines were not considered in the experiments. We did not also consider instances smaller than 6 jobs (smaller instances can be solved with an exact algorithm in reasonable time). Every instance was solved using both algorithms. Because the evolutionary algorithm is random, it was repeated 5 times for each instance. Table 2 and Fig. 3 present obtained results.

As we considered 10 random instances of the problem for each value of n, we calculated through all 10 instances the average values of $z_{LB}(\pi^{\text{MIH}})$ and $z_{UB}(\pi^{\text{MIH}})$ denoted respectively by z_{LB}^{MIH} and z_{UB}^{MIH}. Due to the fact that the evolutionary algorithm was additionally repeated 5 times for each instance, we denote respectively by z_{LB}^{EVO} and z_{UB}^{EVO} the average value of $z_{LB}(\pi^{\text{EVO}})$ and $z_{UB}(\pi^{\text{EVO}})$ calculated through all 5 executions of all 10 instances. We also present the corresponding average running times T^{MIH} and T^{EVO} in seconds. The other conducted experiment consisted in finding out the relationship between uncertainty characteristic of the problem and the difference in the quality of solutions. Therefore, for a fixed value of $n = 10$, we generated 10 random instances of the problem using different values of $C \in \{10, 20, 30, ..., 200\}$. The C value is correlated with the width of the uncertainty interval, therefore instances generated with smaller values of C are considered as "less uncertain" than other ones.

Table 2
Computational results for different values of n.

n	z_{LB}^{MIH}	z_{UB}^{MIH}	z_{LB}^{EVO}	z_{UB}^{EVO}	T^{MIH}	T^{EVO}
6	52.9	91.7	41.2	68.0	<0.01	0.58
7	59.1	101.8	46.7	69.5	< 0.01	0.97
8	60.1	89.5	40.5	65.9	< 0.01	1.49
9	86.9	118.7	48.4	69.3	< 0.01	2.78
10	74.3	103.8	53.5	73.9	0.01	3.93
11	70.7	97.1	53.5	71.6	0.01	4.80
12	86.8	107.2	61.3	79.2	0.02	7.78
13	59.6	84.8	50.7	69.9	0.03	9.32
14	84.6	105.6	56.6	72.8	0.03	12.36
15	84.4	109.6	56.3	76.4	0.05	18.78
16	99.7	125.3	65.8	83.1	0.06	23.55
17	89.2	110.9	64.7	82.2	0.07	30.63
18	110.4	128.6	67.1	84.7	0.10	43.59
19	123.1	146.6	74.9	89.0	0.13	66.51
20	97.1	111.9	67.8	81.4	0.16	65.76
21	103.6	119.6	71.8	85.0	0.19	83.20
22	96.1	107.3	66.9	79.6	0.23	99.20
23	111.7	132.3	73.8	89.3	0.28	125.42
24	102.4	125.7	82.8	95.5	0.33	134.93
25	99.4	130.3	77.5	90.4	0.41	171.16

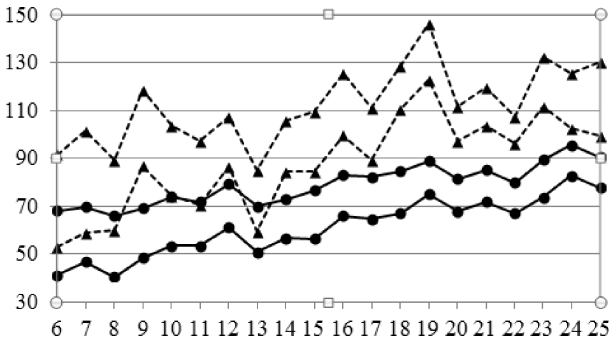

Fig. 3. Average upper and lower bounds of z returned by MIH heuristic (triangles) and EVO algorithm (dots) for different values of n.

The values of the lower and upper bounds of obtained solutions were averaged in the similar manner as previously. The results are given in Table 3 and Fig. 4.

Table 3
Computational results for different values of n.

C	z_{LB}^{MIH}	z_{UB}^{MIH}	z_{LB}^{EVO}	z_{UB}^{EVO}	T^{MIH}	T^{EVO}
10	11.0	33.3	8.34	28.2	0.01	4.28
20	35.2	57.6	28.24	47.84	0.01	4.12
30	47.0	81.4	39.68	67.18	0.01	4.91
40	67.2	110.1	52.32	83.44	0.01	4.85
50	79.8	114.1	61.04	81.68	0.01	5.59
60	93.8	121.5	59.38	79.56	0.01	5.64
70	108.2	139.7	61.34	83.7	0.01	5.14
80	128.6	164.2	81.2	103.26	0.01	5.20
90	142.7	173.1	78.7	97.46	0.01	5.14
100	168.6	204.9	92.12	114.64	0.01	5.76
110	219.2	254.3	90.68	112.12	0.01	6.35
120	190.6	227.2	97.04	115.34	0.01	6.05
130	205.5	239.5	102.2	124.00	0.01	5.25
140	244.6	275	104.2	122.94	0.01	5.72
150	284.8	326.6	109.5	125.46	0.01	6.17
160	307.4	338.1	109.6	130.08	0.01	6.58
170	281	301.8	101.1	117.94	0.01	6.19
180	301.5	338.1	104.6	119.46	0.01	6.21
190	214.5	241.8	108.3	124.5	0.01	5.28
200	281.2	314.0	120.2	138.64	0.01	5.67

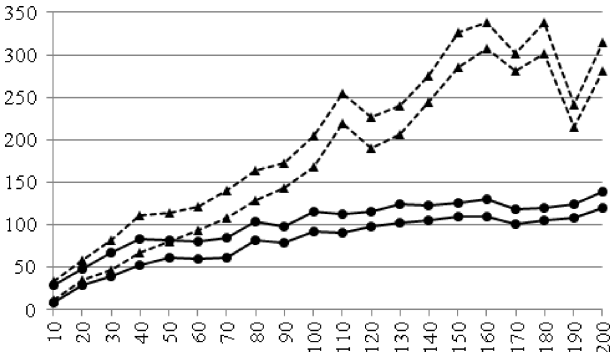

Fig. 4. Average upper and lower bounds of z returned by MIH heuristic (triangles) and EVO algorithm (dots) for different values of C.

Conclusions

The performed experiments confirmed the usefulness of the evolutionary algorithm for minmax regret flow-shop with three machines ($m = 3$). The algorithms were compared for the worst case which means that the upper bounds for the EVO algorithm and the lower bounds for the MIH algorithm were taken into account. The average relative differences for both performed experiments are equal 7% and 18% for different values of n and C, respectively. These values are means of $((z_{LB}^{\text{MIH}} - z_{UB}^{\text{EVO}})/z_{UB}^{\text{EVO}})100\%$ calculated for all n and C used

in the experiments. One can see also that EVO algorithm works better for bigger uncertainty (Fig. 4). Moreover, the lack of clear trends in Fig. 3 allows suspecting that the performance of both algorithms is highly dependent on specific problem instance characteristics. The most important challenge for further works deals with the effective determination of the worst-case scenarios which will result in the development of successful heuristic algorithms for $m > 3$.

References

[1] Pinedo M.L., *Scheduling – Theory, Algorithms and Systems*, Springer, 2008.

[2] Dutt L.S., Kurian M., *Handling of Uncertainty – A Survey*, International Journal of Scientific and Research Publications, 3, 2250–315, 2013.

[3] Pinedo M.L., Schrage L., *Stochastic shop scheduling: A survey*, Dempster M.A.H., Lenstra J.K., Rinooy Kann A.H.G. [Eds.], Deterministic and Stochastic Scheduling, Reidel, Dordrecht, 1982.

[4] Kouvelis P., Yu G., *Robust Discrete Optimization and its Applications*, Kluwer Academic Publishers, Dordrecht-Boston-London, 1997.

[5] Conde E., *A 2-approximation for minmax regret problems via a mid-point scenario optimal solution*, Operations Research Letters, 38 (4), 326–327, 2010.

[6] Averbakh I., *Minimax regret solutions for minimax optimization problems with uncertainty*, European Journal of Operational Research, 27 (2), 57–65, 2000.

[7] Kasperski A., Zielinski P., *A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion*, Operations Research Letters, 42, 343–344, 2008.

[8] Aissi H., Bazgan C., Vanderpooten D., *Min-max and min-max regret versions of combinatorial optimization problems: A survey*, European Journal of Operational Research, 197 (2), 427–438, 2009.

[9] Siepak M., Józefczyk J., *Solution algorithms for unrelated machines minmax regret scheduling problem with interval processing times and the total flow time criterion*, Annals of Operations Research, 222, 517–533, 2014.

[10] Kasperski A., Kurpisz A., Zielinski P. *Approximating a two-machine flow shop scheduling under discrete scenario uncertainty*, Journal of Operational Research, 217, 36–43, 2012.

[11] Averbakh I., *The minmax regret permutation flowshop problem with two jobs*, Operations Research Letters, 69 (3), 761–766, 2006.

[12] Garey M.R., Johnson D.S., Sethi R., *The complexity of flowshop and jobshop scheduling*, Mathematics of Operations Research, 1, 117–129, 1976.

[13] Lebedev V., Averbakh I., *Complexity of minimizing the total flow time with interval data and minmax regret criterion*, Discrete Applied Mathematics, 154, 2167–2177, 2006.

[14] Nawaz M., Enscore Jr. E., Ham I., *A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem*, The International Journal of Management Science, 11, 91–95, 1983.