Heuristic Algorithm for Uncertain Permutation Flow-shop Problem

Jerzy Józefczyk and Michał Ćwik
Faculty of Computer Science and Management, Wroclaw University of Technology,
Wybrzeze Wyspianskiego 27, Wroclaw, Poland

Keywords: Decision Making, Optimization, Uncertainty, Interval Data, Minmax Regret, Heuristic Algorithms.

Abstract: A complex population-based solution algorithm for an uncertain decision making problem is presented. The uncertain version of a permutation flow-shop problem with interval execution times is considered. The worst-case regret based on the makespan is used for the evaluation of permutations of tasks. The resulting complex minmax combinatorial optimization problem is solved. The heuristic algorithm is proposed which is based on the decomposition of the problem into three sequential sub-problems and employs a paradigm of evolutionary computing. The proposed algorithm solves the sub-problems sequentially. It is compared with the fast middle point heuristic algorithm via computer simulation experiments. The results show the usefulness of this heuristic algorithm for instances up to five machines.

1 INTRODUCTION

Investigation of uncertain versions of decision making problems has a long history. Such problems being closer to real-world applications are more complex and difficult to solve. Three issues (Is) are crucial when considering uncertain problems: the representation of uncertainty (I1), the evaluation of arisen uncertain decision making problems (I2), and the determination of corresponding solution algorithms (I3). A variety of approaches are presented and discussed in the literature on all mentioned issues. Their particular combinations lead to plenty specific and mostly difficult complex decision making problems. It is worth noting when referring to issue I1 that representations of uncertainty in the form of probability distributions and the ones based on fuzzy sets and logic seem to be predominant (Dutt and Kurian, 2013; Aayyub and Klir, 2006; Klir, 2006). It is assumed that a probability distribution exists over the space of all values of corresponding random variables. This representation is treated as the objective one as the probability distribution and derivative descriptions can be empirically verified. Substantial difficulties with the determination and (or) the estimation of mentioned probabilistic descriptions, which require considerable and credible empirical data, is the main disadvantage of this popular representation. Other important drawbacks are discussed in (Kouvelis and Yu, 1997). For the fuzzy approach, the availability of experimental data on the uncertainty can be replaced by experts' opinions in the form of corresponding membership functions. In a consequence, the quality of this subjective representation as well as of following activities based on it strongly depends on an expert's quality. Mentioned shortcomings of both popular representations have motivated many researchers to develop other approaches which cope with the uncertainty more adequately. One of such an approach is used in the paper. The main idea of the evaluation of uncertainty as the second mentioned issue (I2) consists first of all in the substantiation (determinization) of the uncertainty. The main idea of such a substantiation consists in transformation of an uncertain problem into its deterministic counterpart. Taking the expected value for the probabilistic representation, when selected probabilistic distribution reflects the uncertainty, is a good example of the substantiation. Other frequently used operators of the substantiation can be found e.g. in (Yager, 1988). Having the deterministic problem as the result the issue I3 arises. In fact, all feasible solution methods and resultant solution algorithms can be considered, i.e. exact, approximate as well as heuristic ones.

It is obvious that the choice and justification of the formalization approach and the solution algorithm for an unceratin decision making problem

Józefczyk, J. and Ćwik, M.
Heuristic Algorithm for Uncertain Permutation Flow-shop Problem.
In *Proceedings of the 1st International Conference on Complex Information Systems (COMPLEXIS 2016)*, pages 119-127
ISBN: 978-989-758-181-6

depend strongly on the problem per se as well as its complexity and computational difficulty. Considerations in the paper are limited to the area of uncertain combinatorial optimization problems with the parametric uncertainty. It means that for a combinatorial optimization problem not all parameters are known, precise, evident or given. The permutation flow-shop with unlimited buffers to minimize the makespan is considered, e.g. (Pinedo, 2008). It is one of the most important task scheduling problems with many applications mainly in manufacturing, production, logistic and service systems, but also in information and computing systems. Generally speaking, the problem deals in the execution of a finite number of complex tasks by a finite number of executors (machines, processors). Each task requires the carrying-out of the same number of operations, being parts of tasks, which equals the number of executors. The exact mapping of executors to operations within tasks is given. A permutation of tasks is sought minimizing defined criterion for a given execution times of operations by corresponding executors. The completion time of the last task last operation in the permutation plays often a role of such criterion. An assembly process of a product performed along a production line is a good example of the investigated flow-shop problem. Then, the order of carried-out products should be determined to minimize the total production time. The non-deterministic versions of flow-shop without full information on execution times are also a subject of many research works, e.g. (Pinedo and Schage, 1982; Kouvelis *et al*, 2000; Averbakh, 2006; Kasperski and Zielinski, 2008).

A specific junction of the mentioned three issues I1, I2 and I3 is proposed in the paper to solve the uncertain optimal decision making problem (optimization problem). Namely, it is assumed that execution times of tasks by machines are uncertain (not fully known). However, the information on their ranges in the form of intervals is only given. The uncertainty in execution times cause straightforwardly the uncertainty of the criterion being the deterministic evaluation of the flow-shop problem considered. The regret based approach is proposed to make possible the evaluation of resultant optimization problem. The notion 'regret' assesses the difference between the value of criterion for fixed realization (scenario) of uncertain parameters and the optimal value of criterion – for a given decision (optimization variable). The application of the regret based approach is recommended for the interval uncertainty (Kouvelis and Yu, 1997; Aissi *et al*, 2009), however, resulting deterministic combinatorial optimization problem is extremely complex and difficult. As it has been pointed out, the regret requires substantiation of the criterion evaluating a decision with respect to all feasible scenarios of uncertainty to have the deterministic evaluation of a decision. In the paper, the substantiation via maximization is proposed which expresses the utmost pessimism od a decision maker (in fact, a decision algorithm) with respect to scenarios of uncertainty (execution times for the considered flow-shop) which can occur but are not known while making a decision. It leads to worst-case i.e. robust decisions on the one hand but safe decisions on the other hand. The solution algorithm determined on such a basis will perform well irrespective of the actual scenario of uncertainty. However, it can work fairly when medium scenarios of uncertainty will take place. The substantiation via averaging seems to be more adequate for such cases which, however, can give poor results for extreme scenarios of uncertainty. The substantiation via maximization is used hereinafter.

In the paper, a bespoke hybrid heuristic solution algorithm is proposed to solve the uncertain problem (issue I3). As it is presented in following sections, the consequent unceratin flow-shop is extremelly difficult combinatorial optimization problem, at least NP-hard one. The rechearches have been focused on developing of time-effective solution algorithms appropriated for real-world applications. A hybrid heuristic algorithm is the result of presented investigations. The evolutionary computing as an important paradigm of the computational intelligence has been employed as the basis for the developed algorithm.

The uncertain flow-shop problem has been firstly stated in (Kouvelis and Yu, 1997). Then it has been investigated in some works. Its NP-hardness was proved in (Kouvelis *et al*, 2000) where a branch-and-bound algorithm and a heuristic procedure based on a local improvement were also developed. Particular attention has been paid in this paper to the elaboration of approximate and heuristic time-efficient solution algorithms. Some computational complexity properties were also investigated in (Kasperski *et al*, 2012) for the case with discrete bounded and unbounded scenario sets. The case of the problem with only two tasks and *m* machines was presented in (Averbakh, 2006) where a linear-time algorithm is given. The evolutionary heuristic algorithm for the case of three machines was considered in (Ćwik and Józefczyk, 2015).

The main contribution of the paper deals with proposing and experimentally evaluating of a time

effective hybrid heuristic solution algorithm for more than three machines. Former works on minmax regret problems, in general, and on minmax regret flow-shop problems, in particular, considered mainly theoretical issues for special cases, e.g. (Averbakh, 2000; Lebedev and Averbakh, 2006; Conde, 2010; Volgenant and Duin, 2010; Lu *et al*, 2012). The results, which can be found there, do not allow us to have constructive tools for solving real-world problems, in general, and permutation flow-shop problems with interval execution times to minimize the makespan, in particular. The algorithm presented in the paper fills this gap for the uncertain flow-shop and refers to analogous works where time-effective algorithms for other minmax regret problems with interval uncertainty are presented, e.g. (Józefczyk, 2008; Józefczyk and Siepak, 2013a,b; Siepak and Józefczyk, 2014; Averbakh and Pereira,. 2011; Pereira, 2016)

To sum up, the decision making under uncertainty is a subject of the paper. In particular, the complex combinatorial NP-hard optimization problem is solved via hybrid heuristic population-based solution algorithm. This algorithm can be treated as a complex tool of computational intelligence.

The reminder of the paper is organized as follows. Deterministic and uncertain versions of the flow-shop investigated are stated in Section 2 as the combinatorial optimization problems. The idea for the solution algorithm together with the detailed presentation of its three component sub-algorithms are given in Section 3. The next section presents results of computer simulation experiments evaluating the algorithm. Section 5 with conclusions completes the paper.

2 PROBLEM FORMULATION

The flow-shop task scheduling problem is investigated with a set $\mathbf{M} = \{M_1, M_2, \ldots, M_i, ..., M_m\}$ of m machines which are assigned for performing n tasks constituting a set of tasks $\mathbf{J} = \{J_1, J_2, \ldots, J_j, ..., J_n\}$. Each task from the set **J** needs to be sequentially carried out by all executors from the set **M**. For the permutation version of flow-shop, which is only investigated in the paper, each machine executes tasks in the same order. Moreover, the version with unlimited buffers is considered which means that the task after being completed by the current machine can leave it and wait if necessary for the service by the next machine at the buffer located there. Let us denote by p_{ij} execution times of task j by executor i for $i = 1, 2, ..., m$ and $j = 1, 2, .., n$ which form the matrix $p = [p_{ij}]_{\substack{i=1,2,...,m \\ j=1,2,...,n}}$.

A part of the task performed by a single machine is called often an operation. Then the problem consists in the determination of a permutation of tasks $\pi = (\pi_1, \pi_2, ..., \pi_j, ..., \pi_n)$ to minimize the makespan, i.e. the completion time of the last task from permutation π by the last machine. The current element π_j of π is the number of task executed as the jth in turn, and $\pi \in \mathbf{\Pi}$ where $\mathbf{\Pi}$ is the set of all $n!$ feasible permutations, i.e. $\mathbf{\Pi} = \{\pi : \pi_j \neq \pi_k, j, k \in \{1, 2, ..., n\}, j \neq k\}$. The makespan $C_{\max}(p, \pi)$ as the function of p and π can be written in different ways (e.g. Pinedo, 2008). However, the following recursive form seems to be the most popular. Let $C_{i,\pi_j}(p, \pi)$ be the time moment when ith machine finishes the execution of task π_j:

$$C_{i,\pi_j}(p,\pi) = p_{i,\pi_j} + \max[C_{i-1,\pi_j}(p,\pi), C_{i,\pi_{j-1}}(p,\pi)], \quad i = 2, 3, ..., m, \; j = 2, 3, ..., n. \tag{1}$$

Moreover,

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

and

$$C_{1,\pi_j}(p,\pi) = \sum_{k=1}^{j} p_{1,\pi_k}, \; j = 1, 2, ..., n. \tag{3}$$

In a consequence, $C_{\max}(p,\pi) = C_{m,\pi_n}(p,\pi)$. The deterministic flow-shop task scheduling problem deals with the determination of such $\pi \in \mathbf{\Pi}$ to minimize $C_{\max}(p,\pi)$. The optimal permutation π' as well as the optimal makespan $C'_{\max}(p) \triangleq C_{\max}(p,\pi')$ is obtained. This problem is NP-hard for $m > 2$. The polynomial optimal Johnson's algorithm exists for $m = 2$ (Garey *et al*, 1976).

Let assume now that crisp values of execution times p_{ij} are not known and given. Instead, intervals for p_{ij} with possible values of corresponding execution times are available as only information of these times. Namely, $p_{ij} \in [\underline{p}_{ij}, \overline{p}_{ij}]$, $\underline{p}_{ij} \leq \overline{p}_{ij}$. Now, matrix p called a

scenario is an element of the Cartesian product of all scenarios $\mathbf{P} = [\underline{p}_{11}, \overline{p}_{ij}] \times \ldots \times [\underline{p}_{mn}, \overline{p}_{mn}]$, i.e. $p \in \mathbf{P}$. The vagueness of p makes it impossible to directly use the makespan $C_{\max}(p,\pi)$ as the evaluation function of π. The approach based on regret is applied. The regret

$$C_{\max}(p,\pi) - C'_{\max}(p) = C_{\max}(p,\pi) - \min_{\sigma \in \mathbf{\Pi}} C_{\max}(p,\sigma) \quad (4)$$

is the difference between the value of makespan and the optimal value of the makespan for fixed p. It is calculated for current p and π. The regret requires substantiation according to scenarios. The worst-case with respect to p has been chosen in the form of the maximization which gives the criterion

$$z(\pi) = \max_{p \in \mathbf{P}} [C_{\max}(p,\pi) - \min_{\sigma \in \mathbf{\Pi}} C_{\max}(p,\sigma)]. \quad (5)$$

Consequently, the uncertain version of the flow-shop task scheduling consists in finding such optimal permutation π^* that

$$z(\pi^*) = \min_{\pi \in \mathbf{\Pi}} \left(\max_{p \in \mathbf{P}} [C_{\max}(p,\pi) - \min_{\sigma \in \mathbf{\Pi}} C_{\max}(p,\sigma)] \right) \quad (6)$$

for given **M**, **J**, $\underline{p}_{i,j}$, $\overline{p}_{i,j}$, $i = 1, 2, .., m$, $j = 1, 2, .., n$.

3 SOLUTION ALGORITHM

The problem (6) called hereinafter **P** is NP-hard even for $m = 2$ (Lebedev and Averbakh, 2006). It is easy to see that (6) is composed of three nested optimization sub-problems:

– **SP1**: inner minimization of $C_{\max}(p,\sigma)$ with respect to permutation σ, being the deterministic flow-shop,

– **SP2**: maximization with respect to feasible scenarios p, i.e. searching for so called worst-case scenario, and at the same time calculating the value of criterion for **SP3**,

– **SP3**: outer minimization with respect to permutation π.

The majority of results presented in the corresponding literature refer to cases where **SP1** is easy, i.e. solvable by polynomial algorithms as well as worst-case scenarios as the solution of **SP2** can be determined either in an intuitional way or by the reduction to easy optimization problems. Nevertheless, it is not a case of this paper's problem.

Now, the problem (6) is a composition of individually difficult optimization sub-problems making **P** extremely difficult issue. In a consequence, it is easy to justify the lack of any approximate algorithm for **P** (Ćwik and Józefczyk, 2015). So, heuristic algorithms are only possible to solve problem **P** in a reasonable time. Such an heuristic algorithm called ALG is proposed in the paper. To develop it, some simplifications have been assumed while solving **SP1**-**SP3**. First of all, **SP1**-**SP3** are solved independently and successively by corresponding individual sub-algorithms. As the result, we have admittedly a heuristic algorithm but working in a reasonable and acceptable time. The main idea for the algorithm ALG solving **P** is based on the assumption that **SP3** is solved by known metaheuristics with the dedicated way of calculating the value of $z(\pi)$ being the criterion for the metaheuristics. Let us present the sub-algorithms successively.

3.1 Sub-algorithm for SP1

The solution of **SP1** is replaced by its lower bound in the form

$$C_{\max,\mathrm{LB}}(p) = \max_{k=1,m} \left(\min_{j=1,n} \sum_{i=1}^{k-1} p_{i,j} + \sum_{j=1}^{n} p_{k,j} + \min_{\substack{l=1,n \\ l \neq j}} \sum_{i=k+1}^{m} p_{i,l} \right) \quad (7)$$

where first and third elements of the maximum reflect respectively the following properties:

– at least one task needs to be processed on all machines indexed from 1 to $k-1$ before the kth machine starts processing operation, unless $k = 1$,

– after ith machine completes the processing, there is at least one task that needs to be processed on machines indexed from $k + 1$ to m, unless $k = m$.

We can take the highest value with respect to all machines as the fact that (7) holds for all machines. The details on this lower evaluation of $C'_{\max}(p)$ can be found in (Ćwik and Józefczyk, 2015). Such a form of the lower bound is used by the procedure calculating approximate value of (5) in **SP2**.

3.2 Sub-algorithm for SP2

The purpose for **SP2** is to calculate the value of function z for the worst-case scenario. For many minmax regret problems, it is enough only to consider extreme point scenarios while searching the worst-case scenarios, i.e.:

$$\forall i,j \; (p_{ij} \in \{\underline{p}_{ij}, \overline{p}_{ij}\}). \quad (8)$$

Calculation of $C_{\max}(p,\pi)$ can be easily substituted with the determination of length of the longest path between v_{1,π_1} and v_{m,π_n} in a directed graph $G(\mathbf{V},\mathbf{A})$ defined by the solution π where **V** and **A** are set of vertexes and arcs, respectively. Each vertex v_{i,π_j} corresponds exactly to an individual execution time of task π_j executed by machine *i*. Therefore, its weight is equal to p_{i,π_j}. Set **A** consists of arcs connecting vertices corresponding to subsequent operations of the same task $(v_{i,\pi_j}, v_{i+1,\pi_j})$, for $0 < i < m$) and of operations performed subsequently on the same machine $(v_{i,\pi_j}, v_{i,\pi_{j+1}})$, for $1 < j < n$). Assuming that processing times of operations are unknown, all possible paths between v_{1,π_1} and v_{m,π_n} represent all possible ways of calculating $C_{\max}(p,\pi)$. Therefore, we can associate each of those paths with a separate scenario. Let us denote path *r* as a sequence of vertices that belong to it. Then, we can construct scenario p^r as follows:

$$p^r = \{p_{ij} : p_{ij} = \overline{p}_{ij}, \text{ for } v_{ij} \in r \wedge p_{ij} = \underline{p}_{ij}, \text{ for } v_{ij} \notin r\}. \quad (9)$$

As it has been shown in (Ćwik and Józefczyk, 2015), this way we can limit the search from checking all 2^{mn} extreme points scenarios to $\left(\frac{(m+n-2)!}{(m-1)!(n-1)!}\right)$ paths that exist in *G*. Then, a simple enumeration has been applied for such decreased number of paths, which consists in checking all feasible paths. This approach turned out effective only for three machines ($m = 3$). Therefore, the heuristic time-effective procedure is now proposed to enable its applicability for more than three machines. The idea is as follows. For each vertex v_{i,π_j} in graph *G*, we construct a path from vertex v_{1,π_1}. Obviously, there is only one possible path for $i = 1$ or $j = 1$. Therefore, the construction is trivial. For $i > 1$ and $j > 1$, we can see that the path to v_{i,π_j} needs to contain one of the two vertices: v_{i-1,π_j}, $v_{i,\pi_{j-1}}$. Assuming that the path to each of them has already been constructed, we add the last vertex v_{i,π_j} and choose the path which yields a greater value of *z* as a constructed path for vertex v_{i,π_j}. After processing all the vertexes in such a manner, the procedure returns constructed path for v_{m,π_n} together with the corresponding value of *z* denoted as $\tilde{z}$. In a consequence, sub-algorithm for **SP2** comprising this procedure is summarized by the following pseudo-code.

Input: Graph G(V,A) generated by feasible solution π.
Output: Heuristic value of $z(\pi)$ denoted as $\tilde{z}(\pi)$.

```
1: Set empty matrix cp
2:  for i≤n
3:   for j≤m
4:    if i==1 or j==1
5:     cp[i][j]:= p_r(v[1][1],v[i][j])
6:    else:
7:     prev1 := cp[i-1][j]
8:     prev2 := cp[i][j-1]
9:     c1 := prev1 + v[i][j]
10:    c2 := prev2 + v[i][j]
11:    if z(c1)>z(c2):
12:     cp[i][j]:=c1
13:    else:
14:     cp[i][j]:=c2
15:    endif
16    endif
17:  endfor
18: endfor
19: return z̃(cp[n][m])
```

3.3 Sub-algorithm for SP3

A standard simple evolutionary algorithm has been applied for solving **SP3** as it has been presented in (Ćwik and Józefczyk, 2015).

The values of permutation π play directly a role of a chromosome. Due to the complexity of $z(\pi)$, the outcome of the sub-algorithm for **SP2** in the form of $\tilde{z}(\pi)$ is used as the fitness function.

The initial population is constructed via the random generation of *N* permutations (chromosomes). The solution of the middle interval metaheuristics (MIH) is the basis for generation of a half of the population. Namely, a MIH-based permutation undergoes $(N/2)-1$ independent random mutations to have in total (*N*/2)–element part of the initial population. The remainder of the initial population is filled with random permutations generated uniformly from the search space. MIH consists in the brute conversion of the uncertain interval problem into its deterministic counterpart by assuming the middles of intervals as deterministic processing times, i.e. $p_{ij}^{\text{MIH}} = (\underline{p}_{ij} + \overline{p}_{ij})/2$.

The order crossover operator (Goldberg, 1989) has been employed to avoid the generation of

unfeasible permutations, so no repair algorithm is required. This operator is characterized by parameter P_{cross} being the probability of crossing over two selected chromosomes. Accordingly, $1 - P_{\text{cross}}$ is the probability of passing the two chromosomes to the next generation without any changes. The value of parameter requires tuning to ensure the best algorithm performance.

A simple mutation operator has been also used. 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 population. The population is sorted according to decreasing values of the fitness function. 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 $\tilde{\pi}$. The sub-algorithm for **SP3** is presented in the form of corresponding pseudo-code.

Input: Matrices $\underline{p}$ and $\overline{p}$ of size $m \times n$ containing respectively lower and upper bounds of processing times.

Output: Permutation $\tilde{\pi}$ and value of criterion $\tilde{z}(\tilde{\pi})$.

```
1: pop = Initialpopulation()
2:  SortByFitFunc(pop)
3:  repeat
4:   nextgen = list of chromosomes
5:   add(nextgen,pop[0])
6:   add(nextgen,pop[1])
7:   remove(pop,pop[0])
8:   remove(pop,pop[1])
9:   repeat:
10:   ind=RemvoveFromPop(pop[0],pop[1])
11:   Sibs = Crossover(ind[1],ind[2])
12:   Mutate(Sibs[1])
13:   Mutate(Sibs[2])
14:   add(nextgen,Sibs[1])
15:   add(nextgen,Sibs[2])
16:  until(pop is empty)
17:  pop = nextgen
18:  SortByFitFunc(pop)
19: until (Stop condition is fulfilled)
20: return(pop[0], z̃(pop[0]))
```

The algorithm for **P** referred to as ALG obtained after merging the described sub-algorithms solving **SP1**-**SP3** is in fact hybrid heuristic population-based one.

4 NUMERICAL EVALUATION OF THE ALGORITHM

The algorithm proposed is evaluated via computer simulation experiments performed using a PC with Intel Core i5 CPU processor of 2,53 GHz with 4GB of RAM. According to our best knowledge, there are no benchmarks in the literature for this problem which could be used for the comparison. Therefore, the own instances have been generated in the following way. For each of *mn* uncertain parameters (execution times), the lower bound $\underline{p}_{i,j}$ is randomly generated from the finite interval [0,*K*] with the uniform distribution. The upper bound $\overline{p}_{i,j}$ is then generated from the finite interval $\left[\underline{p}_{i,j}, \underline{p}_{i,j} + C\right]$ also with the uniform distribution where *K* and *C* are parameters of the experiments. The default values of algorithm parameters and problem parameters have been assumed as: $P_{\text{cross}} = 0.85$, $P_{\text{mut}} = 0.15$, $N = 20$, $K = 100$, $C = 200$.

Values of criterion (6) as well as run times of algorithms are the basis for evaluation. The algorithm ALG is compared with the heuristic deterministic algorithm MIH. As it has been mentioned, MIH generates a solution to any interval data uncertain problem by solving the deterministic counterpart which is obtained by substituting all interval execution times of the problem with its interval middle points. The arisen deterministic problem is also NP–hard. Therefore, the NEH algorithm is applied as the solution tool (Nawaz *et al*, 1983). The solution of deterministic problem π^{MIH} is returned as the solution of the uncertain problem together with the heuristic value of the criterion $\tilde{z}(\pi^{\text{MIH}})$ calculated by the sub-algorithm solving **SP2**. The results are presented in Tables 1–3. Every numerical result in tables is the mean value of five independent instances randomly generated for fixed *n* and *m* from given matrices $\underline{p}$ and $\overline{p}$ of the bounds of interval processing times. Due to randomness of the evolutionary algorithm each instance is additionally repeated for this algorithm five times and the mean result is taken for every of five instances. In a consequence, every result of

Table 1: Computational results for $m = 3$.

n	$\tilde{z}(\pi^{\mathrm{MIH}})$	$\tilde{z}(\tilde{\pi})$	T^{MIH}	T
5	447.4	351.36	<0.001	0.309
6	372.2	318.96	0.001	0.402
7	419.8	372.16	0.001	0.600
8	555.0	361.28	0.002	0.785
9	639.0	405.0	0.003	1.093
10	575.6	424.0	0.003	1.361
11	667.6	470.8	0.004	1.294
12	762.2	474.7	0.006	1.724
13	730.8	473.3	0.007	1.892
14	626.6	474.7	0.009	2.085
15	834.8	509.5	0.011	2.459
16	360.4	476.6	0.013	2.913
17	839.4	510.8	0.016	3.643
18	869.2	516.2	0.017	3.812
19	999.6	510.0	0.021	3.876
20	964.2	511.6	0.026	4.089
21	1057.8	522.7	0.029	5.371
22	918.0	538.6	0.032	5.623
23	892.2	548.7	0.037	5.895
24	922.8	522.3	0.040	5.273
25	965.4	537.1	0.044	6.977
26	1207.8	550.9	0.052	6.419
27	886.4	540.6	0.056	8.583
28	1252.8	553.6	0.063	8.548
29	1433.4	559.8	0.070	8.941
30	993.2	564.9	0.079	10.043

Table 2: Computational results for $m = 4$.

n	$\tilde{z}(\pi^{\mathrm{MIH}})$	$\tilde{z}(\tilde{\pi})$	T^{MIH}	T
5	488.6	392.4	<0.001	0.594
6	599.6	476.1	<0.001	0.772
7	765.8	643.4	0.002	1.147
8	696.6	605.3	0.002	1.293
9	667.4	548.3	0.003	1.686
10	789.8	650.2	0.004	2.116
11	791.6	621.2	0.005	2.941
12	815.4	702.4	0.007	2.929
13	1076.8	753.5	0.008	3.22
14	906.2	723.2	0.010	3.350
15	903.6	737.0	0.012	4.031
16	1042.6	772.8	0.014	5.109
17	1129.2	808.6	0.018	6.182
18	1173.8	779.4	0.020	6.729
19	1218	812.9	0.023	6.383
20	1242	831.4	0.029	7.312
21	1140.6	820.3	0.033	8.593
22	1336.6	811.2	0.036	10.425
23	1394.8	825.8	0.040	10.433
24	1468.2	847.5	0.045	12.819
25	1252.4	858.4	0.051	12.970
26	1645.8	891.4	0.059	14.852
27	1448.8	881.5	0.064	16.267
28	1255.4	898.4	0.072	18.853
29	1500.8	857.5	0.079	14.276
30	1463.8	856.4	0.087	19.002

Table 3: Computational results for $m = 5$.

n	$\tilde{z}(\pi^{\mathrm{MIH}})$	$\tilde{z}(\tilde{\pi})$	T^{MIH}	T
5	622.4	544.7	<0.001	0.790
6	742.2	629.8	0.002	0.987
7	786.6	677.0	0.002	1.619
8	917.8	790.0	0.003	1.758
9	848.4	753.2	0.003	2.428
10	909.4	806.1	0.005	2.607
11	1032.6	845.7	0.006	3.541
12	1051.4	879.6	0.007	4.259
13	1121.8	983.3	0.009	5.238
14	1073.2	940.6	0.012	5.007
15	1241.4	1018.6	0.014	5.668
16	1315.8	1088.4	0.017	7.327
17	1357.4	1099.3	0.020	9.004
18	1453.0	1068.6	0.024	10.628
19	1381.8	1113.3	0.028	12.190
20	1587.6	1153.0	0.030	12.271
21	1616.4	1148.0	0.036	16.190
22	1630.0	1158.2	0.044	18.645
23	1681.8	1155.0	0.046	18.397
24	1594.6	1175.2	0.054	17.910
25	1586.4	1180.0	0.063	20.780
26	1875.2	1214.9	0.067	23.18
27	1721.6	1199.9	0.074	24.570
28	1832.2	1201.7	0.083	26.622
29	1917.0	1237.2	0.092	32.713
30	2163.8	1207.3	0.104	40.000

ALG is, in fact, the mean value of 25 independent random instances. The values of criterion as well as the run times in seconds for MIH and ALG are respectively denoted in corresponding columns of tables as $\tilde{z}(\pi^{\mathrm{MIH}})$ and $\tilde{z}(\tilde{\pi})$ as well as $T^{\mathrm{MIH}}$and T.

The results confirmed the usefulness of ALG in comparison with MIH from both the criterion $\tilde{z}$ and the computational time points of view. It is necessary to point out that the values of criterion (5) for the resulting permutations π^{MIH} and $\tilde{\pi}$ were calculated by the sub-algorithm **SP2**, i.e. the values of $\tilde{z}$ instead of z are the basis for this comparison. It turned out that ALG is substantially better than MIH. Namely, the relative improvement calculated as the ratio

$$\frac{\tilde{z}(\pi^{\mathrm{MIH}})}{\tilde{z}(\tilde{\pi})} \tag{10}$$

fluctuates from 1,11 to 1.79 (except the instance for $m = 3$ and $n = 16$ when MIH is better). The mean improvement for all instances is equal to 1.36. The computational time of ALG is fully acceptable however much longer than for MIH.

5 CONCLUSIONS

The complex and extremely difficult combinatorial optimization problem, i.e. the uncertain permutation flow-shop with the makespan as criterion, has been investigated. The case of minmax regret with interval execution times has been considered. The nested three sub-problems have been solved independently by heuristic sub-algorithms. The algorithm of the whole optimization problem as the consecutive composition of all three sub-algorithms has been compared with the MIH metaheuristics which is often applied for minmax regret problems with interval data. MIH has good properties for many simple optimization problems. For example, it is 2-approximate algorithm for the special case of flow-shop problem considered in the paper with only two machines ($m = 2$). Therefore, this metaheuristics has been chosen as the basis for comparison. It is worth noting that the calculation of criterion for **SP3**, i.e. for the whole problem is also NP-hard. The comparison uses its approximate value $\tilde{z}$. Unfortunately, the relation between z and $\tilde{z}$ is not known. This issue requires more profound studies. The preliminary considerations show the usefulness of $\tilde{z}$ as the approximate value of the criterion for solution $\tilde{\pi}$. Namely, four very simple instances have been considered for $m = 3$ and 4 as well as $n = 4$ and 5. These instances have been solved by ALG as well as their optimal solutions have been derived by the brute force algorithm. The relative differences between both results in the sense of ratio $\tilde{z}(\tilde{\pi}) / z^*(\pi^*)$ analogous to (10) are presented in Table 4. The values of criterion $\tilde{z}$ are worse than z^* no more than 1.30.

The research of this problem is now continuing. For example, the instances for greater *m* and *n* are solved as well as more effective and faster sub-algorithm for **SP3** is verified.

Table 4: Values of $\tilde{z}(\tilde{\pi}) / z^*(\pi^*)$ for different *m* and *n*.

*m**n*	4	5
3	1.21	1.13
4	1.30	1.14

REFERENCES

Aayyub, B. M., Klir, G. J. 2006. *Uncertainty modeling and analysis in engineering and the sciences.* Chapman&Hall/CRC: Boca Raton, London, New York.

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

Averbakh, I. 2000. Minimax regret solutions for minimax optimization problems with uncertainty. *Operations Research Letters*, 27, 57-65.

Averbakh, I. 2006. The minmax regret permutation flow-shop problem with two jobs. *Operations Research Letters*, 169(3), 761-766.

Averbakh, I., Pereira, J. 2011. Exact and heuristic algorithms for the interval data robust assignment problem. *Computers & Operations Research*, 38, 1153-1163.

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

Ćwik, M., Józefczyk, J. 2015. Evolutionary algorithm for minmax regret flow-shop problem. *Management and Production Engineering Review*, 6(3), 3-9.

Dutt, L.S., Kurian, M. 2013. Handling of Uncertainty – A Survey. *International Journal of Scientific and Research Publications*, 3, 2250-3153.

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

Goldberg, D., E. 1989. *Genetic Algorithms in Search. Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Józefczyk, J. 2008. Worst-case allocation algorithms in a complex of operations with interval parameters. *Kybernetes*. 37, 652-676.

Józefczyk. J.. Siepak. M. 2013a. Worst-case regret algorithms for selected optimization problems with interval uncertainty. *Kybernetes*. 42 (3). 371-382.

Józefczyk, J., Siepak, M. 2013b. Scatter Search based algorithms for min-max regret task scheduling problems with interval uncertainty. *Control and Cybernetics*, 42(3), 667-698.

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

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

Klir, G. J. 2006. *Uncertainty and information: Foundations of generalized information theory*, Wiley.

Kouvelis, P., Daniels, R. L., Vairaktarakis, G. 2000. Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, 32, 421-432.

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

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

Lu, C. C., Lin, S. W., Ying, K., C. 2012. Robust scheduling on a single machine to minimize total flow time. *Computers & Operations Research*, 39, 1682-1691.

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

Pereira, J. 2016. The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective. *Computers & Operations Research*, 66, 141-152.

Pinedo. M. L. 2008. *Scheduling–Theory. Algorithms and Systems*. Springer.

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

Siepak, M. Józefczyk, J. 2014. 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.

Volgenant, A., Duin, C. W. 2010. Improved polynomial algorithms for robust bottleneck problems with interval data. *Computers & Operations Research*, 37, 909-915.

Yager, R.R. 1988. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Trans. on SMC*, 18, 183-190.