Control and Cybernetics

vol. **42** (2013) No. 3

Scatter Search based algorithms for min-max regret task scheduling problems with interval uncertainty*

by

Jerzy Józefczyk and Marcin Siepak

Institute of Informatics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
Jerzy.Jozefczyk@pwr.wroc.pl, Marcin.Siepak@pwr.wroc.pl

Abstract: Uncertain versions of three task scheduling problems: $P\|C_{\max}$, $F2\|C_{\max}$, $R\|\sum C_j$ are investigated. Parametric uncertainty is only considered which is represented by intervals. It is assumed that values of execution times of tasks are not a priori given, and they belong to the intervals of known bounds. No distributions additionally characterizing the uncertain parameters are assumed. The regret is used as the basis for a criterion evaluating the uncertainty. In a consequence, min-max regret combinatorial problems are solved. Heuristic algorithms based on Scatter Search are proposed. They are evaluated via computational experiments and compared to a simple middle intervals heuristics and to exact solutions for small instances of the problems considered.

Keywords: task scheduling, interval uncertainty, min-max regret, branch and bound.

1. Introduction

Uncertain versions of optimization problems, in particular the combinatorial optimization problems, are being developed for many decades. This concerns also task scheduling. The term 'uncertainty' can be understood and characterized in many ways and can concern diverse aspects of such problems. Moreover, uncertainty can be differently represented and evaluated. The paper is focused on a selected and very specific case which, however, is amply present in the literature. Investigations are limited to parametric uncertainty when exact values of some parameters in the objective functions to be optimized are not known. Different approaches are used to represent uncertainty, in general and for the optimization problems, in particular, see Ayyub and Klir (2006), Klir (2006), Bubnicki (2004) for surveys. Stochastic and fuzzy approaches are undoubtedly

*Submitted: July 2012; Accepted: May 2013.

most popular. Both of them assume the existence of some distributions concerning uncertain parameters in their domains, i.e.: probability distributions and membership functions, respectively. Unfortunately, application of both approaches is often impossible or at least not recommended due to many reasons. The most important are: scarce representative empirical data for the whole domain of uncertainty to determine probability distributions and lack of appropriate credible experts to elaborate membership functions. In the paper we assume that the only available information about uncertain parameters is limited to sets of their values in the form of intervals of known bounds. Specific values of uncertain parameters are called scenarios. Many authors used such representation also for task scheduling, e.g. Kasperski and Zielinski (2008), Lebedev and Averbakh (2006), Jozefczyk and Siepak (2011), or for other combinatorial optimization problems, e.g.: Assi et al. (2005), Conde (2009), Kasperski et al. (2013), Volgenant and Duin (2010).

Apart from the representation of uncertainty, its evaluation is a second important issue which has to be decided when solving uncertain versions of optimization problems. The majority of methods and techniques which can be encountered in the literature, in fact, use criteria defined for deterministic versions and propose their different determinization (substantiation) with respect to uncertain parameters to have the deterministic criteria as a consequence. Mean value for the stochastic approach is a good example of the criterion after determinization. The determinization can refer directly to criteria for deterministic versions or to some terms based on them. Absolute regret or equivalently absolute opportunity loss called hereafter regret, being such an expression, is employed in the paper to assess the uncertainty. The regret is subject to determinization by the use of different corresponding operators. The notion of 'regret' has been proposed in Savage (1951), and it was first used in management science. Kouvelis and Yu in their seminal book (Kouvelis and Yu, 1997) summarized previous results and indicated directions of further work in the area of regret based methods with interval representation of uncertainty and their application to operations research problems. Many results, both general and particular, have been developed since that time. Let us only mention selected works: Conde (2010), Averbakh (2000), Kasperski (2008), Aissi et al. (2009) where some general results as well as those related to many particular operations research optimization problems are reported. Averbakh (2000) and Conde (2010) showed the classes of easy continuous optimization problems. The results can be used for many particular applications, see e.g. Jozefczyk (2008), Jozefczyk and Siepak (2013) where min-max regret allocation problems with interval uncertainty are discussed. It is worth noting that majority of studies are focused on very simple and rather easy, elementary deterministic problems, treated as the basis for investigation of their uncertain counterparts. The authors are mainly concentrated on approximate solution algorithms and on other analytical properties of the uncertain problems, e.g. Conde (2007, 2009), Gutierrez et al. (1996). Due to the fact that uncertain problems are more complex and more difficult to solve in comparison with corresponding deterministic versions, it is obvious that expected results are possible only for basic problems far from real-world applications. Therefore, it is extremely important to broaden investigations on time efficient solution algorithms which could be practically used (see Averbakh and Pereira, 2011, and Kasperski et al., 2012, where such approach is applied). However, it is necessary to bear in mind that for many cases heuristic algorithms the are only time efficient solution tools that can be developed.

In the paper a class of combinatorial optimization problems is considered which deals with the determination of integer optimization vector variables $u \in D_u$ dependent on vector parameters $p \in D_p$, where D_u, D_p are sets of feasible optimization variables, parameters, respectively. Variables u are evaluated by a function $F(p,u)$ undergoing minimization to get optimal variables u', i.e. $F'(p) = min_{u \in D_u} F(p,u)$. For the uncertain counterpart considered, it is assumed that values of every element $p^{(i)}$ of p belong to an interval of known bounds, i.e. $p^{(i)} \in \left[\underline{p}^{(i)}, \overline{p}^{(i)}\right]$ where $\underline{p}^{(i)} \leq \overline{p}^{(i)}$. All intervals constitute a set P of all possible scenarios for interval parameters p. The evaluation function for such uncertain case is based on the regret being the difference $F(p,u) - F'(p)$. The worst-case with respect to unknown parameters p is considered which leads to the following deterministic min-max problem $min_{u \in D_u} \max_{p \in D_p} [F(p,u) - F'(p)] = min_{u \in D_u} z(u)$.

Three selected min-max regret task scheduling problems with interval uncertainty are considered. They are uncertain counterparts of: $P\|C_{\max}$, $F2\|C_{\max}$, $R\|\sum C_j$. The first one has not been yet investigated. Some results on the uncertain flow-shop can be found in Kouvelis et al. (2000). The latter problem is the generalization of $1\|\sum C_j$ discussed in Kasperski and Zielinski (2008). In consecutive sub-sections of Section 2, formulations as well as the most important properties for both deterministic and uncertain versions are presented. The main purpose of the paper is to elaborate and analyse time efficient heuristic solution algorithms of all three uncertain problems. The common platform for all algorithms in the form of Scatter Search metaheuristics has been used and presented in Section 3. In the subsequent section, results of simulation experiments evaluating the heuristic algorithms are given. Final remarks complete the paper.

2. Selected task scheduling problems

For task scheduling problems considered in this section, n tasks are assigned to m machines. The execution time of jth task on ith machine is denoted $p_{i,j}$. For the flow-shop problems, whose example is investigated in Sub-section 2.2, execution times $p_{i,j}$ refer to the execution of ith operation of jth task. All execution times $p_{i,j}$ form matrix $p = [p_{i,j}]_{i=1,\ldots,m; j=1,\ldots,n}$. For uncertain interval problems, it is assumed that exact values of $p_{i,j}$ are not known, but they belong to intervals

$\left[\underline{p}_{i,j}, \overline{p}_{i,j}\right]$ of known bounds, $\underline{p}_{i,j} \leq \overline{p}_{i,j}$. Matrix parameter p is called a scenario, and it is an element of a set of feasible scenarios P, being the Cartesian product of all intervals. Additionally, C_j and $C_{\max} = \max_j C_j$ denote the completion time of task j and the makespan, respectively.

2.1. Problem $P\|C_{\max}$

Problem $P\|C_{\max}$ consists in assigning each task to one of m parallel and identical machines. The execution time p_j of jth task is machine independent. Each machine can process maximally one task at a time. The optimal assignment minimizes the makespan which is the execution time of a machine working the longest. The order of tasks executed on each machine is arbitrary, as it does not affect the quality criterion. Let us denote by $c = [c_{i,j}]_{i=1,\ldots,m;j=1,\ldots,n}$ the binary assignment matrix such that $c_{i,j} = 1$ if jth task is scheduled on ith machine, and 0, otherwise. The set D_c specifies the constraints which have to be fulfilled by any feasible solution c:

$$D_c = \left\{ c_{i,j} \in \{0,1\} : \quad \forall j = 1,2,\ldots,n \quad \sum_{i=1}^{m} c_{i,j} = 1 \right\}. \tag{1}$$

The makespan can be expressed analytically as $F_1(p,c) = \max_{i=1,\ldots,m} \sum_{j=1}^{n} p_j c_{i,j}$. Then, problem $P\|C_{\max}$ deals with the determination of such a matrix c, feasible with respect to D_c, that minimizes $F_1(p,c)$, i.e. $F_1^{'}(p) \triangleq F_1(p,c^{'}) \triangleq \min_{c\in D_c} \max_{i=1,\ldots,m} \sum_{j=1}^{n} p_j c_{i,j}$.

For the uncertain $P\|C_{\max}$, the processing times of tasks are assumed to be imprecise, i.e. $p_j \in \left[\underline{p}_j, \overline{p}_j\right]$. The worst-case regret, being the objective function for this version of the problem, takes the form $z_1(c) = \max_{p\in P}\left[F_1(p,c) - F_1^{'}(p)\right] = F_1(p^c,c) - F_1^{'}(p^c)$ where p^c is the worst-case scenario for a given assignment matrix c. The optimal solution $c^* \in D_c$ minimizes the value of z_1. According to our best knowledge, this uncertain problem has not been yet investigated.
The deterministic version of $P\|C_{\max}$ is NP-hard as shown in Garey and Johnson (1978). The popular approximate algorithm for solving this problem is the LPT rule (see, e.g. Pinedo, 2008). The complexity of $P\|C_{\max}$ implies directly the NP-hardness of its uncertain counterpart for which the following property is satisfied:

Property 2.1 : *NP-hardness of $P\|C_{max}$ implies impossibility of developing any polynomial approximation algorithm for the uncertain version of this problem.*

Proof: Let c^* be the optimal solution for the uncertain problem and let us assume that a k-approximation algorithm exists ($k > 1$), which returns solution $\hat{c}$

such that $z_1(\hat{c}) \leq kz_1(\hat{c})$. Such an approximation algorithm could be applied to the deterministic problem, where $\forall j \quad \underline{p}_j = \overline{p}_j$, and must return then solution $\hat{c}$ such that $z_1(\hat{c}) = 0$. Matrix $\hat{c}$ must be, therefore, the optimal solution for $P\|C_{\max}$, which is not possible if P $\neq$ NP. ■

NP-hardness of the uncertain $P\|C_{\max}$ causes also that it is not possible to calculate efficiently the exact value of $F_1^{'}(p^c)$, required to have $z_1(c)$ for a given c. Therefore, $z_{1,\mathrm{LB}}$ and $z_{1,\mathrm{UB}}$, being the lower and the upper bounds of z_1 are obtained instead. Let us denote by c_{pmtn} and c_{LPT} the approximate solutions of $P\|C_{\max}$ relaxed by allowing tasks preemption and applying LPT rule, respectively. To calculate $z_{1,\mathrm{LB}}$ and $z_{1,\mathrm{UB}}$, the following procedure is applied (Jozefczyk and Siepak, 2011):

```
Algorithm 1: Lower and upper bounds calculation
  Data: Input solution $c$.
  Result: Lower bound $z_{1,\mathrm{LB}}$ and upper bound $z_{1,\mathrm{UB}}$ of the worst-case
          regret $z_1$.
1 Fix $\overline{p} = [\overline{p}_1, \dots, \overline{p}_n]$. For $\overline{p}$ find index $r$ of the executor that works the
  longest, i.e. $r = \arg\max_{i=1,\dots,m} \sum_{j=1}^{n} \overline{p}_j c_{ij}$.
2 Obtain the worst-case scenario $p^c = [p_1^c, \dots, p_n^c]$ such that $p_j^c = \overline{p}_j$,
  $j = 1, 2, \dots, n$ when $c_{r,j} = 1$ and $p_j^c = \underline{p}_j$, otherwise.
3 For $p^c$ solve the deterministic problem $P\|C_{\max}$ by applying LPT rule and
  obtain $F_1(p^c, c_{\mathrm{LPT}})$.
4 Calculate $F_{1,\mathrm{pmtn}}(p^c) = \frac{1}{m} \sum_{j=1}^{n} p_j^c$.
5 Calculate $z_{1,\mathrm{LB}}(c) = F_1(p^c, c) - F_1(p^c, c_{\mathrm{LPT}})$ and
  $z_{1,\mathrm{UB}}(c) = F_1(p^c, c) - F_{1,\mathrm{pmtn}}(p^c)$.
6 return $z_{1,\mathrm{LB}}$, $z_{1,\mathrm{UB}}$.
```

2.2. Problem $F2\|C_{\max}$

Considerations in this section refer to preliminary results given in Kouvelis et al. (2000). It is assumed that $m = 2$ dedicated machines and a set of n tasks are given where each task consists of two operations. The ith operation of each task needs to be performed on the ith machine ($i = 1, 2$) and the second operation of task j cannot start until the first operation is completed. Each machine is allowed to perform maximally one operation at each time moment. The objective is to find the schedule which minimizes the makespan. The optimization variables are elements of permutation $\sigma = (\sigma_1, \dots, \sigma_n) \in \Phi$ which specifies the task schedule, i.e. σ_j is the number of task performed at the jth position of permutation, where Φ is the set of all feasible permutations. Matrix p contains now two rows. The value of objective function $F_2(p, \sigma)$, which for given p and σ expresses the makespan, is the completion time moment of the second operation of task σ_n, i.e.

$$F_2(p, \sigma) = C_{2,\sigma_n}(p). \tag{2}$$

We denote by C_{i,σ_j} the completion time of ith operation of task σ_j. The value of C_{i,σ_j} can be expressed by the following set of equations Pinedo, (2008):

$$C_{i,\sigma_1}(p) = \sum_{k=1}^{i} p_{k,\sigma_1}, \quad i = 1, 2, \tag{3}$$

$$C_{1,\sigma_j}(p) = \sum_{k=1}^{j} p_{1,\sigma_k}, \quad j = 1, \ldots, n, \tag{4}$$

$$C_{2,\sigma_j}(p) = \max\left(C_{2,\sigma_{j-1}}, C_{1,\sigma_j}\right) + p_{2,\sigma_j}, \quad j = 2, \ldots, n. \tag{5}$$

The following constraints are imposed on the optimization variables:

$$\forall j \quad \sigma_j \in \{1, \ldots, n\} \quad \wedge \quad \forall j \neq k \quad \sigma_j \neq \sigma_k. \tag{6}$$

The deterministic problem deals with the minimization of F_2, with respect to σ, i.e.

$$F_2^{'}(p) \triangleq F_2(p, \sigma^{'}) = \min_{\sigma \in \Phi} F_2(p, \sigma). \tag{7}$$

This problem is polynomially solvable and the optimal solution can be obtained using Johnson's algorithm (Johnson, 1954).

For the uncertain version of $F2\|C_{\max}$, $p_{i,j} \in \left[\underline{p}_{i,j}, \overline{p}_{i,j}\right]$. The optimal schedule σ^* minimizes the value of $z_2(\sigma) = \max_{p \in P}\left[F_2(p,\sigma) - F_2^{'}(p)\right]$, i.e. $z_2^* \triangleq z_2(\sigma^*) = \min_{\sigma \in \Phi} z_2(\sigma)$. The uncertain version of $F2\|C_{\max}$ is NP-hard (Kouvelis et al., 2000).

2.3. Problem $R\|\sum C_j$

The task scheduling problem $R\|\sum C_j$ deals with the scheduling of n independent tasks on m parallel and unrelated machines to minimize the sum of completion times $\sum C_j$. The solution of the problem can be characterized by matrix of binary optimization variables $x = [x_{i,k,j}]_{i=1,\ldots,m; j,k=1,\ldots,n}$, where $x_{i,k,j} = 1$ if jth task is scheduled on machine i as kth to the last, and 0, otherwise. For given scenario p, being now the m rows matrix, the criterion is expressed as follows

$$F_3(p, x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} k p_{i,j} x_{i,k,j}. \tag{8}$$

The following constrains are imposed on optimization variables $x_{i,k,j}$. Each task can be performed on exactly one position of exactly one machine

$$\sum_{i=1}^{m} \sum_{k=1}^{n} x_{i,k,j} = 1, \quad j = 1, 2, \ldots, n. \tag{9}$$

At most one task can be performed on each position of each machine

$$\sum_{j=1}^{n} x_{i,k,j} \leq 1, \quad i = 1, 2, \ldots, m, \quad k = 1, 2, \ldots, n. \tag{10}$$

The elements of matrix x are binary optimization variables

$$x_{i,k,j} \in \{0, 1\}, \quad i = 1, 2, \ldots, m, \quad j, k = 1, 2, \ldots, n. \tag{11}$$

The deterministic problem deals with the minimization of F_3 with respect to x subject to constraints (9), (10) and (11), i.e. $F_3^{'}(p) \triangleq F_3(p, x^{'}) = \min_x F_3(p, x)$.

For the uncertain $R\|\sum C_j$, i.e. when $p_{i,j} \in \left[\underline{p}_{i,j}, \overline{p}_{i,j}\right]$, the worst-case regret takes the following form: $z_3(x) = \max_{p \in P} \left[F_3(p, x) - F_3^{'}(p)\right] = F_3(p^x, x) - F_3^{'}(p^x)$, where p^x is the worst-case scenario for a feasible solution x. The optimal solution x^* minimizes z_3 with respect to (9), (10) and (11). Results presented in this sub-section are generalizations of similar outcomes obtained for $1\|\sum C_j$ and presented in Kasperski and Zielinski (2008).

Problem $R\|\sum C_j$ is equivalent to the weighted bipartite assignment problem, see Pinedo (2008), polynomially solvable by the Hungarian algorithm (Jungnickel, 2008). The uncertain version of $R\|\sum C_j$ is NP-hard which results from the NP-hardness of the uncertain version of $1\|\sum C_j$ (Lebedev and Averbakh, 2006). We will show now how to calculate the worst-case scenario p^x and regret $z_3(x)$ for a given solution x. Let us notice that considering (8) the criterion for the uncertain problem can be expressed by the following equation:

$$z_3(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} \left(k p_{i,j}^{x} x_{i,k,j} - k p_{i,j}^{x} x_{i,k,j}^{'} \right). \tag{12}$$

For any solution y, let k_j^y and i_j^y denote, respectively, the index of position before the last one and the machine on which task j is scheduled, i.e. $y_{i_j^y, k_j^y, j} = 1$. Therefore, the regret can be rewritten as:

$$z_3(x) = \sum_{j=1}^{n} \left(k_j^x p_{i_j^x, j}^{x} - k_j^{x^{'}} p_{i_j^{x^{'}}, j}^{x} \right) \tag{13}$$

where k_j^x and i_j^x are known for a given solution x. It is easy to see that if $i_j^x \neq i_j^{x^{'}}$, then we can increase the processing time $p_{i_j^x, j}^{x}$ to $\overline{p}_{i_j^x, j}$ and decrease $p_{i_j^{x^{'}}, j}^{x}$ to $\underline{p}_{i_j^{x^{'}}, j}$. Otherwise, i.e. for $i_j^x = i_j^{x^{'}}$, the execution time of task j depends on the indexes of positions before the last one where it is scheduled in x and $x^{'}$, i.e. $p_{i_j^x, j}^{x} = p_{i_j^{x^{'}}, j}^{x} = \overline{p}_{i_j^x, j}$ if $k_j^x > k_j^{x^{'}}$ and $p_{i_j^x, j}^{x} = p_{i_j^{x^{'}}, j}^{x} = \underline{p}_{i_j^x, j}$ if $k_i^x \leq k_i^{x^{'}}$.

Consequently, we can now express the regret for x as:

$$\begin{aligned} z_3(x) &= \sum_{\left\{j: i_j^x \neq i_j^{x'}\right\}} \left(k_j^x \overline{p}_{i_j^x,j} - k_j^{x'} \underline{p}_{i_j^{x'},j} \right) \\ &+ \sum_{\left\{j: i_j^x = i_j^{x'} \wedge k_j^x > k_j^{x'}\right\}} \left(k_j^x \overline{p}_{i_j^x,j} - k_j^{x'} \overline{p}_{i_j^{x'},j} \right) \\ &+ \sum_{\left\{j: i_j^x = i_j^{x'} \wedge k_j^x \leq k_j^{x'}\right\}} \left(k_j^x \underline{p}_{i_j^x,j} - k_j^{x'} \underline{p}_{i_j^{x'},j} \right). \end{aligned} \tag{14}$$

Observe that the worst-case scenario can be immediately obtained from (14) if we know $x^{'}$, which, as a result, would allow us to know $k_j^{x^{'}}$ and $i_j^{x^{'}}$. We show now that the computation of $x^{'}$ can be done by solving weighted bipartite matching problem.

Let us define binary variables $z_{i,k,j} \in \{0,1\}$, $i = 1,\ldots,m$; $j,k = 1,\ldots,n$, where $z_{i,k,j}$ takes 1 only if $k = k_j^{x^{'}}$ and $i = i_j^{x^{'}}$, that is, task j is scheduled as the kth to the last on machine i in solution $x^{'}$. Since each task has to be scheduled on exactly one position of exactly one machine, and each position on each machine is taken by at most one task, variables $z_{i,k,j}$ fulfil the following requirements

$$\sum_{i=1}^{m}\sum_{k=1}^{n} z_{i,k,j} = 1; \quad j = 1,\ldots,n, \tag{15}$$

$$\sum_{j=1}^{n} z_{i,k,j} \leqslant 1; \quad i = 1,\ldots,m; \quad k = 1,\ldots,n, \tag{16}$$

$$z_{i,k,j} \in \{0,1\}, \quad i = 1,\ldots,m; \quad j,k = 1,\ldots,n. \tag{17}$$

The regret (14) can be now rewritten as:

$$\begin{aligned} z_3(x) &= \max_{z_{i,k,j}} \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n} \left(k_j^x p_{i_j^x,j}^x - k p_{i,j}^x \right) z_{i,k,j} \\ &= \max_{z_{i,k,j}} \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{i,k,j}^x z_{i,k,j} \end{aligned} \tag{18}$$

where maximization is taken with respect to all $z_{i,k,j}$ fulfilling (15), (16), (17), and $c_{i,k,j}^x$ are fixed coefficients having the following form:

$$c_{i,k,j}^x = \begin{cases} k_j^x \overline{p}_{i_j^x,j} - k\underline{p}_{i,j} & \text{if } i_j^x \neq i, \\ k_j^x \overline{p}_{i_j^x,j} - k\overline{p}_{i,j} & \text{if } i_j^x = i \wedge k_j^x > k, \\ k_j^x \underline{p}_{i_j^x,j} - k\underline{p}_{i,j} & \text{if } i_j^x = i \wedge k_j^x \leq k. \end{cases} \tag{19}$$

The problem of maximization in (18) s.t. (15) — (17) is the weighted bipartite assignment problem with n tasks and mn positions and can be solved in a polynomial time using the well known Hungarian algorithm, which returns the solution $z^{'}_{i,k,j}$. Then, the absolute regret z_3 is calculated using the formula:

$$z_3(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} c^{x}_{i,k,j} z^{'}_{i,k,j} \tag{20}$$

where $c^{x}_{i,k,j}$ is determined according to (19). Observe that knowledge of $x^{'}$ allows us to determine the individual values $p^{x}_{i,j}$ of the matrix representing the worst-case scenario p^{x}.

3. Scatter Search based solution algorithms

Scatter Search is an evolutionary method, which has been successfully applied to hard optimization problems, see, e.g. Corberan et al. (2002), Nowicki and Smutnicki (2006), Jozefczyk and Siepak (2001), Xu et al. (2001). The fundamental concepts of this method were first proposed in the 1970s and were based on formulations for combining decision rules and problem constraints; however, its current state was described in Glover (1997), see also Glover, Laguna and Marti (2000). This algorithm uses strategies for search diversification and intensification in order to avoid stopping at local optima. The quality of each solution is based on its objective function value and is also characterized by the value of the diversity measure which is specified for the problem. Five separate subprocedures can be distinguished within the Scatter Search basic algorithm (Laguna and Marti, 2003):

A. Diversification Generation Method generates a collection of $MSize$ initial high diversity solutions, which allows for analyzing different points of solution space. The solutions generated do not necessarily need to be feasible.

B. Improvement Method transforms (if necessary) the input solution in order to fulfil the feasibility conditions and uses local search algorithms for improving solution quality. If the original input solution is feasible and no improvement was achieved, then this solution is returned as a result.

C. Reference Set Update Method updates and maintains the reference set $RefSet$ which contains solutions particularly valuable for the purpose of the optimum solution exploration and to avoid stopping of the algorithm at local optima. This set consists of at most b_1 high quality solutions and at most b_2 high diversity solutions. At a single iteration of the Scatter Search procedure, a set of so called candidate solutions is generated, each of these solutions being verified whether it can be added to $RefSet$. The verification process requires defining problem specific method of obtaining the distance $d(x, y)$ between two candidate solutions x and y. The value of d is calculated between the candidate solution and each solution in the $RefSet$. The smallest d found determines the distance between the candidate solution x

and $RefSet$ and is calculated according to the formula:

$$\tilde{d}(x, RefSet) = \min_{i=1,\ldots,|RefSet|} d(x, s_i) \tag{21}$$

where s_i is the ith solution in the $RefSet$. Each of the solutions added to this set is marked as high quality solution or high diversity solution. The Reference Set Update Method is presented as Algorithm 2, where it is assumed that we are dealing with the problem of minimization, and $z(x)$ is the value of the objective function z for a given solution x.
Algorithm 2 consists of 3 conditions. If any of them is fulfilled, then the candidate solution x is automatically added to $RefSet$:
Condition 1: Is a number of high quality solution in $RefSet$ lower than $b1$?
Condition 2: Is the quality of x better than the quality of the worst solution in $RefSet$ marked as the high quality solution?
Condition 3: Is the diversity of x higher than the diversity of the least diverse solution in $RefSet$ marked as the high diversity solution?

D. Subset Generation Method produces all r-element subsets of $RefSet$ where r is the input parameter of the procedure. In most of practical cases it is assumed that $r = 2$ which means that the total number of subsets equals $0.5(b^2 - b)$ where $b = b_1 + b_2$.
E. Solution Combination Method combines all input solutions and generates the new output solution as a result.

Scatter Search heuristics consists of five above subprocedures which are called for iteratively. Adaptation of this algorithm to solve specific optimization problem requires adjusting its subprocedures in order to take into consideration problem properties. It is noteworthy that the Subset Generation Method is generally problem independent and the Reference Set Update Method only requires defining distance d between two problem solutions x and y. All other subprocedures are problem specific and have to be fully adapted to the properties of a problem. The general Scatter Search procedure introduced in Laguna and Marti (2003) is presented as Algorithm 3.
It is easy to see that its duration depends significantly on the number of repetitions of the **while** loop in Line 11. The algorithm stops when the $RefSet$ has not been updated at least once while executing the loop in Lines 14—22, because none of the solutions obtained as a result of using Combination and Improvement Methods (Line 16) was of higher quality or higher diversity than any of the $RefSet$ solutions. The highest quality solution of $RefSet$ is then returned as a result. Additionally, one can set a time restriction for the algorithm which would end the loop in Line 11 after the predefined period of time, being an input parameter of the procedure.

Algorithm 2: Reference Set Update method

```
   Data: Current set of reference solutions $RefSet$, candidate solution $x$ to
         $RefSet$, $L$ and $R$ — current number of solutions marked as high
         quality and high diversity solutions, respectively.
   Result: Updated set of reference solutions $RefSet$.
1  Generate vector $s = [s_1, \ldots, s_L]$ in $RefSet$ marked as high quality
   solutions. Sort $s$ non-descending according to the value of $z$.
2  if $L < b_1$ then
3      Add $x$ to $RefSet$ and mark $x$ as high quality solution.
4      return $RefSet$.
5  else if $z(x) < z(s_L)$ then
6      Remove solution $s_L$ from $RefSet$.
7      Add $x$ to $RefSet$ and mark $x$ as high quality solution.
8      return $RefSet$.
9  else
10     Generate vector $s = [s_1, \ldots, s_R]$ in $RefSet$ marked as high diversity
       solutions. Sort $s$ non-ascending according to the value of
       $\tilde{d}(s_i, RefSet)$, $i = 1, \ldots, R$.
11     if $\tilde{d}(x, RefSet) > \tilde{d}(s_R, RefSet)$ then
12         if $R$ equals $b_2$ then
13             Remove task $s_R$ from $RefSet$.
14         end
15         Add $x$ to $RefSet$ and mark $x$ as high diversity solution.
16     end
17 end
18 return $RefSet$.
```

Algorithm 3: General Scatter Search scheme

```
Data: $b_1$, $b_2$, $MSize$, $r$.
Result: $x_{\text{best}}$ — the highest quality solution found in RefSet.
Auxiliary variables: $M = \phi$ — set of initial solutions, $SubSet$ — set of
all $r$-element subsets of $RefSet$, $IsRefSetUpdated$ — boolean variable
specifying whether $RefSet$ has been updated.
1  while $|M| < MSize$ do
2      Use Diversification Generation Method to construct an initial solution
       $x$ and apply Improvement Method to get the improved solution $y$ as a
       result.
3      if $y \notin M$ then
4          Add $y$ to $M$.
5      end
6  end
7  while $|RefSet| < b$ do
8      Use the Reference Set Update Method trying to add the consecutive
       solution $y$ of set $M$ to $RefSet$.
9  end
10 $IsRefSetUpdated = $ true
11 while $IsRefSetUpdated$ do
12     $IsRefSetUpdated = $ false
13     Use Subset Generation Method in order to obtain $SubSet$.
14     while $|SubSet| > 0$ do
15         Get the consecutive element $s$ of $SubSet$.
16         Apply the Combination Method to $s$ and generate the new trial
           solution $x$. Apply the Improvement Method to $x$ to get the
           improved solution $y$ as a result.
17         Use the Reference Set Update Method for $y$ as the candidate
           solution.
18         if $y$ was added to $RefSet$ then
19             $IsRefSetUpdated = $ true
20         end
21         Remove $s$ from $SubSet$.
22     end
23 end
24 return $x_{\text{best}} \in RefSet$.
```

3.1. Uncertain version of $P||C_{\max}$

In this sub-section, we present how Scatter Search subprocedures were adapted to the uncertain version of $P||C_{\max}$. Diversification Generation Method is an iterative subprocedure which uses sequence s of tasks non-assigned yet to any of the machines. Its pseudo-code is presented as Algorithm 4. In ith iteration of this algorithm, value of the auxiliary parameter k is randomly selected from the set $\{2, 3, \dots, \lfloor n/2 \rfloor\}$. Then, all tasks, with indexes corresponding to every kth element of the sequence s, are assigned for processing on machine i and their indexes are removed from s. The procedure ends its execution in mth iteration by assigning all unallocated tasks to mth machine.

Algorithm 4: Diversification Generation method for uncertain $P||C_{\max}$

```
   Data: $m$, $n$.
   Result: $c$ — generated diverse solution.
   Auxiliary variables: $s = (s_1, \dots, s_L)$, $k$.
 1 Generate zero matrix $c = [c_{i,j}]_{i=1,\dots,m;j=1,\dots,n}$.
 2 Generate sequence $s = (1, \dots, n)$.
 3 for $i \leftarrow 1$ to $m - 1$ do
 4     Select randomly the value of $k$ from set $\{2, 3, \dots, \lfloor n/2 \rfloor\}$.
 5     for $j \leftarrow L$ down to $1$ do
 6         if $(j - 1)$ mod $k$ equals $0$ then
 7             $c_{i,s_j} = 1$.
 8             Remove $j$th element of sequence s.
 9         end
10     end
11 end
12 for $j \leftarrow 1$ to $L$ do
13     $c_{I,s_j} = 1$.
14 end
15 return $c$.
```

Algorithm 5: Improvement method for uncertain $P||C_{\max}$

```
   Data: $c$, $m$.
   Result: $c$ — improved solution.
   Auxiliary variables: $w = (1, \dots, m)$, $L_i$ — number of tasks assigned to
   machine $i$.
 1 For given solution $c$, generate the worst-case scenario $p^c$.
 2 Sort elements of $w$ non-ascending according to the sum of task execution times
   on each machine.
 3 for $i \leftarrow 1$ to $\lfloor m/2 \rfloor$ do
 4     for $j \leftarrow 1$ to $L_{w_i}$ do
 5         Generate temporary solution $\tilde{c}$ by moving the shortest execution time
           task scheduled on machine $w_i$ to machine $w_{m-i+1}$.
 6         if $z_{1,\text{UB}}(\tilde{c}) < z_{1,\text{LB}}(c)$ then
 7             $c = \tilde{c}$.
 8             For given solution $c$, generate the worst-case scenario $p^c$.
 9         end
10     end
11 end
12 return $c$.
```

Improvement Method presented as Algorithm 5 tries to improve iteratively the quality of input solution c. The procedure starts by obtaining the worst-case scenario p^c and a sequence w of machine indexes sorted non-ascending according to the sum of execution times of tasks performed on each machine. In iteration i ($i = 1, \dots, \lfloor m/2 \rfloor$), a temporary solution $\tilde{c}$ is generated by moving the shortest execution time task performed on machine w_i to machine w_{m-i+1}. If upper bound of z for solution $\tilde{c}$ is smaller than the lower bound of z for c, then a new improved solution was found, which is marked as the current solution, i.e. $c = \tilde{c}$, and the new worst-case scenario p^c is calculated. The algorithm terminates after analyzing all tasks assigned to the first $\lfloor m/2 \rfloor$ machines of sequence w. The improved solution c is returned as a result.

In Reference Set Update Method, the distance between any two input solutions $c^{'}$ and $c^{''}$ is defined as a sum of the absolute differences between values of $c^{'}_{i,j}$ and $c^{''}_{i,j}$, i.e.

$$d\left(c^{'}, c^{''}\right) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left| c^{'}_{i,j} - c^{''}_{i,j} \right|. \tag{22}$$

Apart from defining the distance between two input solutions, this procedure is problem independent and works according to the scheme presented as Algorithm 2.

Solution Combination Method combines two input solutions $c^{'}$, $c^{''}$ and generates the new output solution $\hat{c}$ as a result. This procedure extends the results presented in Laguna and Marti (2003). It starts from generating zero matrix $\hat{c}$ and a sequence $s = (s_j)_{j=1,\dots,n} = (1, \dots, n)$ of tasks not assigned yet to any of machines. At ith iteration this algorithm tries to assign selected tasks of sequence s to machine i. The selection occurs by calculating for each element s_j the following score

$$\varsigma\left(i, s_j\right) = \frac{z_{1,\mathrm{UB}}\left(c^{'}\right) c^{'}_{i,s_j} + z_{1,\mathrm{UB}}\left(c^{''}\right) c^{''}_{i,s_j}}{z_{1,\mathrm{UB}}\left(c^{'}\right) + z_{1,\mathrm{UB}}\left(c^{''}\right)}, \tag{23}$$

and by randomly selecting the value of an auxiliary parameter $\rho \in [0, 1]$ according to the uniform probability distribution. If $\rho < \varsigma\left(i, s_j\right)$, then task j is assigned to machine i, otherwise such assignment is not performed, i.e.

$$\hat{c}_{i,s_j} = \begin{cases} 1, & \text{when } \rho < \varsigma\left(i, s_j\right), \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

All unallocated tasks in mth iteration are automatically assigned to machine m. The pseudocode of this procedure is presented as Algorithm 6.

Algorithm 6: Solution Combination method for uncertain $P||C_{\max}$

```
   Data: $c^{'}, c^{''}$.
   Result: $\hat{c}$ — output combined solution.
   Auxiliary variables: $s = (s_j)_{j=1,\dots,L} = (1,\dots,n)$.
1  Generate zero matrix $\hat{c} = [\hat{c}_{l,r}]_{l=1,\dots,m;r=1,\dots,n}$
2  for $i \leftarrow 1$ to $m-1$ do
3      for $j \leftarrow L$ down to 1 do
4          Calculate $\varsigma(i, s_j)$ according to (23).
5          Select randomly $\rho \in [0,1]$.
6          if $\varsigma(i, s_j) > \rho$ then
7              $\hat{c}_{i,s_j} = 1$.
8              Remove $j$th element of sequence s.
9          end
10     end
11 end
12 for $j \leftarrow 1$ to $L$ do
13     $\hat{c}_{m,s_j} = 1$.
14 end
15 return $\hat{c}$.
```

3.2. Uncertain version of $F2||C_{\max}$

Scatter Search subprocedures to the uncertain $F2||C_{\max}$ is presented in this sub-section. Diversification Generation Method is an iterative procedure which in jth iteration randomly selects one of the unallocated tasks and assigns it to jth position of permutation σ (Algorithm 7).

Algorithm 7: Diversification Generation method for uncertain $F2||C_{\max}$

```
   Data: $n$.
   Result: $\sigma$ — generated diverse solution.
1  Generate permutation $\sigma = (\sigma_1,\dots,\sigma_n) = (0,\dots,0)$.
2  for $j \leftarrow 1$ to $n$ do
3      Assign one of randomly selected unallocated tasks to position $\sigma_j$.
4  end
5  return $\sigma$.
```

Improvement Method, presented as Algorithm 8, iteratively tries to swap positions occupied by each pair of tasks and checks whether this improves the value of the regret. Algorithm terminates after analyzing all pairs of tasks.

In Reference Set Update Method, the distance between any two input solutions σ and δ is defined according to the following formula:

$$d(\sigma, \delta) = \sum_{j=1}^{n} |j_\sigma - j_\delta|, \tag{25}$$

where for any feasible input solution γ, the value of j_γ determines the position occupied by jth task in γ.

Algorithm 8: Improvement method for uncertain $F2\|C_{\max}$

```
  Data: $\sigma$ — input solution.
  Result: $\sigma$ — improved solution.
1 for $j \leftarrow 1$ to $n$ do
2     for $w \leftarrow (j+1)$ to $n$ do
3         Generate permutation $\tilde{\sigma}$ by iteratively swapping task $\sigma_j$ with task
          $\sigma_w$.
4         if $z_2(\tilde{\sigma}) < z_2(\sigma)$ then
5             $\sigma = \tilde{\sigma}$.
6         end
7     end
8 end
9 return $\sigma$.
```

Solution Combination Method generates iteratively permutation ($\epsilon = \epsilon_1 \ldots, \epsilon_n$) which is a result of combination of two input solutions $\sigma = (\sigma_1, \ldots, \sigma_n)$ and $\sigma = (\sigma_1, \ldots, \sigma_n)$. In jth iteration of this procedure, one of the still unallocated tasks corresponding to jth position of either schedule σ or δ is assigned to position j of solution ϵ. In order to specify the exact task index for the assignment, parameter ρ is introduced with value randomly generated from the interval $[0, 1]$ according to the uniform probability distribution. If $\rho > 0.5$, then checking is performed whether task σ_j has already been allocated to any of positions $1, \ldots, j-1$ of permutation ϵ. If such an allocation was not performed before, then $\epsilon_j = \sigma_j$. Otherwise, i.e. if task σ_j has been already allocated before in any of the previous positions of solution ϵ, then a similar checking is performed in order to determine whether task δ_j has been already allocated to any of positions $1, \ldots, j-1$ of permutation ϵ. The lack of such an assignment causes performing allocation of task δ_j into solution at jth position of solution ϵ. If both tasks σ_j and δ_j have been already assigned in the previous positions of ϵ, then ϵ_j is left unallocated and the procedure proceeds to iteration $j+1$. In case when $\rho < 0.5$, the procedure works analogously, but $\epsilon_j = \delta_j$ is attempted to be performed first, and if that violates the constrains, then $\epsilon_j = \sigma_j$ is checked. If both tasks δ_j and σ_j have been previously assigned, then the algorithm proceeds to the next iteration. After processing n iterations, all unassigned tasks are randomly allocated to the unassigned positions of ϵ. The pseudocode of this procedure is presented as Algorithm 9.

Algorithm 9: Solution Combination method for uncertain $F2||C_{\max}$

```
Data: $\delta$ $\sigma$.
Result: $\epsilon$ — output combined solution
Auxiliary variables: $x$, $y$, $\rho$.
1  for $j \leftarrow 1$ to $n$ do
2      Select randomly $\rho \in [0, 1]$.
3      $x = y = 0$.
4      if $\rho > 0.5$ then
5          $x = \sigma_j$, $y = \delta_j$.
6      else
7          $x = \delta_j$, $y = \sigma_j$.
8      end
9      if task $x$ has not been assigned to $\epsilon$ in any of previous iterations
       then
10         $\epsilon_j = x$.
11     else
12         $\epsilon_j = y$.
13     end
14 end
15 Randomly allocate all unassigned yet tasks to unallocated positions of $\epsilon$.
16 return $\epsilon$.
```

3.3. Uncertain version of $R||\sum C_j$

In this sub-section, we present how Scatter Search subprocedures were adapted to the uncertain version of $R||\sum C_j$. Diversification Generation Method generates a feasible solution x by randomly assigning tasks to machines and then randomly ordering tasks performed on each machine. The computational experiments performed showed that Scatter Search returns solutions of better quality when using the random procedure for generating input solutions rather than any of the tested deterministic methods.

Algorithm 10: Diversification Generation method for uncertain $R||\sum C_j$

```
Data: $m$, $n$.
Result: $x$ — generated diverse solutions.
1  for $j \leftarrow 1$ to $n$ do
2      Assign $j$th task to the randomly selected machine index $i = 1, \ldots, m$
       according to the uniform probability distribution.
3  end
4  for $i \leftarrow 1$ to $m$ do
5      Randomly order tasks assigned to $i$th machine according to the uniform
       probability distribution.
6  end
7  Transform obtained solution to fulfil Property 3.1.
8  return $x$.
```

Algorithm 11: Improvement method for uncertain $R\|\sum C_j$

```
   Data: $x$, $m$, $t$.
   Result: $x$ — improved solution.
   Auxiliary variables: $w = (w_i)$, $i = 1, \dots, m$, $k_{w_i}$, $\tilde{x}$, $\hat{x}$.
 1 Generate the sequence of machines $w = (w_i)$, $i = 1, \dots, m$ and sort it
   non-ascending according to the sum of tasks completion times on each machine.
 2 for $i \leftarrow 1$ to $\lfloor m/2 \rfloor$ do
 3     Obtain the total number of task $k_{w_i}$ assigned to machine $w_i$.
 4     for $k \leftarrow 1$ to $k_{w_i}$ do
 5         Generate temporary solution $\tilde{x}$ obtained by moving task performed as
           $k$th to the last on machine $w_i$ and scheduling it as the last one on
           machine $w_{m-i+1}$.
 6         if $z_3(\tilde{x}) < z_3(x)$ then
 7             $x = \tilde{x}$.
 8         else
 9             Generate temporary solution $\hat{x}$ by ordering tasks in $\tilde{x}$
               non-descending according to the worst-case scenario $p^{\tilde{x}}$.
10             if $z_3(\hat{x}) < z_3(x)$ then
11                 $x = \hat{x}$.
12                 $i = 1$.
13                 if current execution time $> t$ then
14                     return $x$.
15                 end
16                 Go to Line 1.
17             end
18         end
19     end
20 end
21 return $x$.
```

The initial schedule generated is modified at the end of the procedure in order to fulfil the following property, which improves the total completion time value and is true for each optimal solution (e.g. Pinedo, 2008):

Property 3.1 *: If task j is assigned to position $k > 1$ on machine i, then there is also a task assigned to position $k-1$ on the same machine. Otherwise, scheduling task j on position $k-1$ would improve the total assignment cost.*

The pseudo-code for the Diversification Generation Method is presented as Algorithm 10.

Improvement Method tries to improve input solution x by moving tasks between machines and by changing the execution order of tasks on each machine. The procedure starts by generating a sequence $w = (w_1, \dots, w_m)$ of machine indexes and sorting it non-ascending according to the sum of task completion times on each machine. Then, machines corresponding to the first $\lfloor m/2 \rfloor$ elements of m are analyzed iteratively. For machine w_i, $i = 1, \dots, \lfloor m/2 \rfloor$, the procedure determines k_{w_i}, which is the total number of tasks assigned to w_i

and then iteratively generates temporary solution $\tilde{x}$ by moving the task performed there as kth to the last $(k = 1, \ldots, k_{w_i})$ and scheduling it as the last one on machine w_{m-i+1}. If the quality of $\tilde{x}$ is better than of x, then the new temporary solution becomes the best solution currently found (i.e. $x = \tilde{x}$). Otherwise, the tasks on each machine are sorted non-descending to $p^{\tilde{x}}$, which gives new solution $\hat{x}$. If $z_3(\hat{x}) < z_3(x)$, then $\hat{x}$ is marked as the new best solution found, i.e. $x = \hat{x}$, and the procedure starts again by going to Line 1. The algorithm terminates when analysis of the first $\lfloor m/2 \rfloor$ machines of the sequence w did not produce any improved solution during any of the improvement attempts. An alternative stop condition for the procedure is the execution time limit t (being the input parameter), exceeding of which causes automatically the termination of procedure and returning the best solution found as a result. The scheme of this algorithm is presented as Algorithm 11.

Algorithm 12: Solution Combination method for uncertain $R\|\sum C_j$

```
   Data: $x$, $y$.
   Result: $v$ — combined output solution.
 1 for $j \leftarrow 1$ to $n$ do
 2     $i_j^v = \lfloor 0.5\left(i_j^x + i_j^y\right)\rfloor$.
 3     $k_j^v = \lfloor 0.5\left(k_j^x + k_j^y\right)\rfloor$.
 4     if position $k_j^v$ on machine $i_j^v$ is free then
 5         $v_{i_j^v, k_j^v, j} = 1$.
 6     else
 7         Find the closest available position to $k_j^v$ on machine $i_j^v$ and assign
           there $j$th task.
 8     end
 9 end
10 Modify $v$ in order to fulfil Property 3.1.
11 return $v$.
```

In Reference Set Update Method, the distance between any two input solutions x and y is defined as the sum of the absolute differences between values of $x_{i,k,j}$ and $y_{i,k,j}$, i.e.

$$d(x, y) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{n} |x_{i,k,j} - y_{i,k,j}|. \tag{26}$$

Solution Combination Method generates a new solution $v = [v_{i,k,j}]_{i=1,\ldots,m; j,k=1,\ldots,n}$, which is the result of combination of input solutions x and y. Let k_j^w and i_j^w denote respectively the index of position to the last and the machine index, where task j is scheduled on, in a given solution w, i.e. $w_{i_j^u, k_j^u, j} = 1$. In the jth iteration of the algorithm, task j is assigned to machine i_j^v, whose index is the arithmetic average of machine indices, where task j is scheduled in solutions x and y. The position k_j^v, where task j is going to be performed on machine i_j^v, is also the arithmetic average of the positions, where this task is performed on, in x and y. If position k_j^v on machine i_j^v is already occupied by any other task, then the algorithm looks for the closest available position index on this machine and assigns there the jth task. At the end, the solution generated is modified in order to fulfil Property 3.1. The pseudo-code describing this procedure is presented as Algorithm 12.

4. Computational experiments

The proposed Scatter Search (SS) algorithms were experimentally evaluated in terms of their quality and execution times. Exact algorithms (EX) based on a simple enumeration, tested for small data instances, as well as Middle Interval algorithms (MI) serve as the basis for evaluation. The latter algorithms consist in determining the middle intervals scenarios p^{mid} such that $p_{i,j}^{\mathrm{mid}} = 0.5\left(\underline{p}_{i,j} + \overline{p}_{i,j}\right)$ or $p_j^{\mathrm{mid}} = 0.5\left(\underline{p}_j + \overline{p}_j\right)$ and then in solving the resulting deterministic problems for p^{mid} using Hungarian algorithm for $R\|\sum C_j$, Johnson's algorithm for $F2\|C_{\max}$ or LPT rule for $P\|C_{\max}$. For the uncertain $F2\|C_{\max}$ and $R\|\sum C_j$, such algorithms keep the property of 2-approximation according to criteria z_2 and z_3, respectively. For the uncertain $P\|C_{\max}$, this is an heuristic procedure.

Due to the lack of benchmark test data instances, a method of generating bounds of the interval execution times was proposed. The parameter $C \in \mathbb{N}$ was introduced and the values of $\underline{p}_{i,j}$ (or $\underline{p}_j$ for the uncertain version of $P\|C_{\max}$) and $\overline{p}_{i,j}$ (or $\overline{p}_j$) were selected randomly according to the uniform distribution within intervals $[0, C]$ and $\left[\underline{p}_{i,j}, \underline{p}_{i,j} + C\right]$ (or $\left[\underline{p}_j, \underline{p}_j + C\right]$), respectively. Parameter C determines the maximum range of intervals where the uncertain parameters belong to. Therefore, it can be treated as the numerical characteristics of uncertainty.

For each problem, the test data instances were generated, i.e. $n \in \{10, 20, 50, 100\}$, $m = 5$ for both the uncertain $P\|C_{\max}$ and $R\|\sum C_j$ as well as $n \in \{9, 11, 13, 80, 120, 160\}$ for the uncertain $F2\|C_{\max}$. Moreover, first two problems were additionally solved for $n = 10$, $m = 2$ where SS and MI were compared with EX. All experiments were performed for $C \in \{10, 30, 50, 70, 100, 150\}$. For each configuration of C, m, n, a single problem instance of the problem was generated. SS was repeated 5 times for each instance due to the randomness of Diversification Generation Method and Combination Method procedures. All execution times of algorithms launched in the experiments are expressed in seconds.

Let u_{MI}, $\tilde{u}_{\max}$ and $\tilde{u}_{\min}$ be MI solution, SS best and SS worst quality solutions, respectively. The following general performance indices were proposed to evaluate the relative percentage difference between SS and MI for the worst and the best results generated by SS, respectively, i.e.:

$\delta_{\max} = \frac{z(u_{\mathrm{MI}}) - z(\tilde{u}_{\max})}{z(\tilde{u}_{\max})} \cdot 100\%$ which expresses, for a given problem instance, how much, the quality of u_{MI} is worse than the worst quality solution $\tilde{u}_{\max}$ generated by SS,
$\delta_{\min} = \frac{z(u_{\mathrm{MI}}) - z(\tilde{u}_{\min})}{z(\tilde{u}_{\min})} \cdot 100\%$ which expresses, for a given problem instance, how much the quality of u_{MI} is worse than the best quality solution $\tilde{u}_{\min}$ generated by SS,
and decisions specific for every particular problem considered stand for u_{MI}, $\tilde{u}_{\max}$ and $\tilde{u}_{\min}$. Similar general performance index $\gamma(u) = \frac{z(u) - z(u^*)}{z(u^*)} \cdot 100\%$ was proposed for small problem instances to compare solution u with optimal solution u^*.

Tables 1–19 present the results of computational experiments for all three problems. Column C denotes the value of parameter C. Symbols T_{EX}, T_{MI} and T_{avg} denote the execution times of EX, MI and the average execution times of SS, while running it 5 times, respectively. In order to simplify the presentation, all values of the regret criteria are rounded to the nearest integer.

4.1. Uncertain version of $P\|C_{\max}$

As a result of tuning the parameters of SS, we assumed $MSize = 50$, $b_1 = 5$, $b_2 = 5$. The results of computational experiments are presented in Tables 1–6. Columns $z_{1,\mathrm{UB}}(\tilde{c}_{\min})$, $z_{1,\mathrm{UB}}(\tilde{c}_{\mathrm{avg}})$ and $z_{1,\mathrm{UB}}(\tilde{c}_{\max})$, denote, respectively, the minimum, the average and the maximum value of the upper bound of z_1. Moreover, $z_{1,\mathrm{LB}}(c_{\mathrm{MI}})$ expresses lower bound of z_1 obtained for the solution c_{MI} generated by MI. For small instances of the problem, the exact value of z_1, presented in $z_1(c^*)$ column, was calculated for the optimal solution c^*. The performance indices have now the following particular forms:
$\delta_{1,\max} = \frac{z_{1,\mathrm{LB}}(c_{\mathrm{MI}}) - z_{1,\mathrm{UB}}(\tilde{c}_{\max})}{z_{1,\mathrm{UB}}(\tilde{c}_{\max})} \cdot 100\%$,
$\delta_{1,\min} = \frac{z_{1,\mathrm{LB}}(c_{\mathrm{MI}}) - z_{1,\mathrm{UB}}(\tilde{c}_{\min})}{z_{1,\mathrm{UB}}(\tilde{c}_{\min})} \cdot 100\%$,
$\gamma_1(c) = \frac{z_1(c) - z_1(c^*)}{z_1(c^*)} \cdot 100\%$ where $c \in \{\tilde{c}_{\min}, \tilde{c}_{\max}, c_{\mathrm{MI}}\}$.

The results for $n = 10$, $m = 2$ (Table 5) show that the lowest quality SS solutions are at most by 8.8% worse than the corresponding optimal solutions, while the highest quality SS solutions are worse than the corresponding optimal ones by not more than 2.6%, which is expressed by $\gamma_1(\tilde{c}_{\min})$ and $\gamma_1(\tilde{c}_{\max})$, respectively. MI solutions are at least by 33.9% worse than the optimal ones, which is reported by $\gamma_1(c_{\mathrm{MI}})$. Both heuristics work fast. It is easy to see that all solutions generated by SS are better than the corresponding MI solutions. The maximum obtained value of both $\delta_{1,\max}$ and $\delta_{1,\min}$ equals 66.7% (for $n = 10$, $C = 10$). The average values of $\delta_{1,\max}$ and $\delta_{1,\min}$, i.e. $\delta_{1,\max,\mathrm{avg}}$ and $\delta_{1,\min,\mathrm{avg}}$ for $m = 5$ and $n = 10, 20, 50, 100$ are presented in Table 6. General corollary holds that the increase of number of tasks n leads to the decrease of both $\delta_{1,\max,\mathrm{avg}}$ and $\delta_{1,\min,\mathrm{avg}}$. The results confirm also the low sensitivity of indices $\delta_{1,\max}$ and $\delta_{1,\min}$ to changes of parameter C. This can be explained by the similar change rate of z_1, while increasing C for both SS and MI.

Execution times of SS even for the largest tested problem instance do not exceed 0.79 (Table 4: $n = 100$, $m = 5$, $C = 150$). Corresponding times of MI are less than 0.01 for all tested problem instances. Thus, we recommend applying SS for all cases, apart from the case when the generation of solutions is time critical. Then, we suggest using MI, whose execution times are smaller than 0.01 for all tested problem instances.

Table 1. Results of SS and MI for $n = 10$, $m = 5$.

C	SS				MI		$\delta_{1,\max}$	$\delta_{1,\min}$
	T_{avg}	$z_1(\tilde{c}_{\min})$	$z_1(\tilde{c}_{\text{avg}})$	$z_1(\tilde{c}_{\max})$	T_{MI}	$z_1(c_{\text{MI}})$		
10	< 0.2	3	3	3	< 0.01	5	66.7	66.7
30	< 0.2	12	12	12	< 0.01	19	58.3	58.3
50	< 0.2	32	32	32	< 0.01	43	34.4	34.3
70	< 0.2	76	77	79	< 0.01	106	34.2	39.5
100	< 0.2	70	71	73	< 0.01	103	41.1	47.1
150	< 0.2	91	91	91	< 0.01	121	33.0	33.0

Table 2. Results of SS and MI for $n = 20$, $m = 5$.

C	SS				MI		$\delta_{1,\max}$	$\delta_{1,\min}$
	T_{avg}	$z_1(\tilde{c}_{\min})$	$z_1(\tilde{c}_{\text{avg}})$	$z_1(\tilde{c}_{\max})$	T_{MI}	$z_1(c_{\text{MI}})$		
10	< 0.25	13	13	13	< 0.01	18	38.5	38.4
30	< 0.25	30	30	30	< 0.01	43	43.3	43.3
50	< 0.25	77	77	77	< 0.01	96	24.7	24.7
70	< 0.25	93	95	96	< 0.01	110	14.7	18.3
100	< 0.25	163	167	169	< 0.01	214	26.6	31.3
150	< 0.25	202	207	215	< 0.01	252	17.2	24.7

4.2. Uncertain version of $F2\|C_{\max}$

Small ($n \in \{9, 11, 13\}$) and large ($n \in \{80, 120, 160\}$) instances of the problem were generated as the basis of experiments. We assumed $MSize = 50$, $b_1 = 5$, $b_2 = 5$ as parameters of SS. The results are presented in Tables 7–13. Columns $z_2(\tilde{\sigma}_{\min})$, $z_2(\tilde{\sigma}_{\text{avg}})$ and $z_2(\tilde{\sigma}_{\max})$ denote, respectively, the minimum, the average and the maximum value of z_2. Moreover, $z_2(\sigma^*)$ and $z_2(\sigma_{\text{MI}})$ denote, respectively, the quality of optimal solution σ^* and 2-approximate MI solution σ_{MI}.

Analogously as in the previous subsection, the following performance indices are used:
$\delta_{2,\max} = \frac{z_2(\sigma_{\text{MI}}) - z_2(\tilde{\sigma}_{\max})}{z_2(\tilde{\sigma}_{\max})} \cdot 100\%$,
$\delta_{2,\min} = \frac{z_2(\sigma_{\text{MI}}) - z_2(\tilde{\sigma}_{\min})}{z_2(\tilde{\sigma}_{\min})} \cdot 100\%$,

Table 3. Results of SS and MI for $n = 50$, $m = 5$.

C	SS				MI		$\delta_{1,\max}$	$\delta_{1,\min}$
	T_{avg}	$z_1(\tilde{c}_{\min})$	$z_1(\tilde{c}_{\text{avg}})$	$z_1(\tilde{c}_{\max})$	T_{MI}	$z_1(c_{\text{MI}})$		
10	< 0.3	36	36	36	< 0.01	44	22.2	22.2
30	< 0.3	132	136	140	< 0.01	159	13.6	20.4
50	< 0.3	200	202	204	< 0.01	251	23.0	25.5
70	< 0.3	298	298	298	< 0.01	365	22.5	22.5
100	< 0.3	443	449	462	< 0.01	517	11.9	16.7
150	< 0.3	579	591	598	< 0.01	729	21.9	25.9

Table 4. Results of SS and MI for $n = 100$, $m = 5$.

C	SS				MI		$\delta_{1,\max}$	$\delta_{1,\min}$
	T_{avg}	$z_1(\tilde{c}_{\min})$	$z_1(\tilde{c}_{\text{avg}})$	$z_1(\tilde{c}_{\max})$	T_{MI}	$z_1(c_{\text{MI}})$		
10	0.72	68	69	70	< 0.01	83	18.6	22.1
30	0.75	231	231	231	< 0.01	250	8.2	8.2
50	0.77	365	374	384	< 0.01	441	14.8	20.8
70	0.77	541	544	546	< 0.01	577	5.7	6.6
100	0.78	753	763	779	< 0.01	835	7.2	10.9
150	0.79	1211	1237	1266	< 0.01	1318	4.1	8.8

Table 5. Results of EX, SS and MI for $n = 10$, $m = 2$.

C	EX		SS				MI		$\delta_{1,\max}$	$\delta_{1,\min}$	$\gamma_1(\tilde{c}_{\min})$	$\gamma_1(\tilde{c}_{\max})$	$\gamma_1(c_{\text{MI}})$
	T_{EX}	$z_1(c^*)$	T_{avg}	$z_1(\tilde{c}_{\min})$	$z_1(\tilde{c}_{\text{avg}})$	$z_1(\tilde{c}_{\max})$	T_{MI}	$z_1(c_{\text{MI}})$					
10	5.47	4	< 0.1	4	4	4	< 0.01	6	50	50	0.0	0.0	50.0
30	5.73	21	< 0.1	21	21	21	< 0.01	31	47.6	47.6	0.0	0.0	47.6
50	5.84	38	< 0.1	39	40	40	< 0.01	58	45	48.7	2.6	5.3	52.6
70	5.87	56	< 0.1	56	56	56	< 0.01	75	33.9	33.9	0.0	0.0	33.9
100	5.71	114	< 0.1	115	119	124	< 0.01	155	25	34.8	0.9	8.8	36.0
150	5.95	135	< 0.1	135	136	137	< 0.01	195	42.3	44.4	0.0	1.5	44.4

Table 6. The average values of $\delta_{1,\max}$ and $\delta_{1,\min}$ for different n.

n	10	20	50	100
$\delta_{1,\max,\text{avg}}$	44.60	27.48	19.19	9.77
$\delta_{1,\min,\text{avg}}$	46.49	30.13	22.21	12.92

$\gamma_2(\sigma) = \frac{z_2(\sigma)-z_2(\sigma^*)}{z_2(\sigma^*)} \cdot 100\%$ where $\sigma \in \{\tilde{\sigma}_{\min}, \tilde{\sigma}_{\max}, \sigma_{\text{MI}}\}$.

From Tables 8–10 one can see that for all small tested problem instances, SS returns optimal solutions. Moreover, running SS independently five times does not improve the quality of solutions, i.e. $z_2(\tilde{\sigma}_{\min}) = z_2(\tilde{\sigma}_{\text{avg}}) = z_2(\tilde{\sigma}_{\max})$, and $\delta_{2,\max} = \delta_{2,\min}$ for these problem instances. MI solutions are at least by 21.2% worse than the optimal ones (see $\gamma_2(\sigma_{\text{MI}})$). It is also worth noting that T_{avg} is not substantially sensitive to changes of C, i.e. the increase of T_{avg} is not greater than 6.71% while increasing C for fixed n (Table 7, $C = 10$, $C = 150$). The execution times of MI do not exceed 0.1 for any of the tested problem instances. Such time for the largest problem instance solved by SS ($n = 160$) is approximately one hour. Values of $\delta_{2,\max}$ vary in the range between approximately 21% and 34% for small problem instances ($n \in \{9, 11, 13\}$) and between 9% and 15% for large problem instances, while values of $\delta_{2,\min}$ change in the range between 12% and 20%. Average values of $\delta_{2,\max}$ and $\delta_{2,\min}$ are presented in Table 13. The corollary is similar to that for uncertain $P\|C_{\max}$. Namely, the increase of n results in decreasing of both $\delta_{2,\max,\text{avg}}$ and $\delta_{2,\min,\text{avg}}$.

As a consequence of the experiments, we recommend using SS when the quality of solutions is more important, and MI when having the solution in a short time is preferred.

Table 7. Results of SS and MI for $n = 80$.

C	SS				MI		$\delta_{2,\max}$	$\delta_{2,\min}$
	T_{avg}	$z_2(\tilde{\sigma}_{\min})$	$z_2(\tilde{\sigma}_{\text{avg}})$	$z_2(\tilde{\sigma}_{\max})$	T_{MI}	$z_2(\sigma_{\text{MI}})$		
10	1133	199	202	209	< 0.1	235	12.4	18.1
30	1161	292	298	306	< 0.1	342	11.8	17.1
50	1173	339	350	356	< 0.1	405	13.8	19.5
70	1187	473	477	483	< 0.1	542	12.2	14.6
100	1191	644	659	669	< 0.1	761	13.8	18.2
150	1209	977	993	1011	< 0.1	1131	11.9	15.8

4.3. Uncertain version of $R\|\sum C_j$

Experiments were conducted for $MSize = 70$, $b_1 = 7$, $b_2 = 7$ as tuned parameters of SS. The results are presented in Tables 14–19. Columns $z_3(\tilde{x}_{\min})$,

Table 8. Results of EX, SS and MI for $n = 9$.

C	EX		SS				MI		$\delta_{2,\max}$	$\delta_{2,\min}$	$\gamma_2\,(\tilde{\sigma}_{\min})$	$\gamma_2\,(\tilde{\sigma}_{\max})$	$\gamma_2\,(\sigma_{\mathrm{MI}})$
	T_{EX}	$z_2\,(\sigma^*)$	T_{avg}	$z_2\,(\tilde{\sigma}_{\min})$	$z_2\,(\tilde{\sigma}_{\mathrm{avg}})$	$z_2\,(\tilde{\sigma}_{\max})$	T_{MI}	$z_2\,(\sigma_{\mathrm{MI}})$					
10	1.31	11	0.43	11	11	11	< 0.01	14	27.3	27.3	0.0	0.0	27.3
30	1.32	26	0.45	26	26	26	< 0.01	34	30.8	30.8	0.0	0.0	30.8
50	1.34	44	0.46	44	44	44	< 0.01	59	34.1	34.1	0.0	0.0	34.1
70	1.35	70	0.46	70	70	70	< 0.01	89	27.1	27.1	0.0	0.0	27.1
100	1.37	125	0.47	125	125	125	< 0.01	163	30.4	30.4	0.0	0.0	30.4
150	1.37	221	0.49	221	221	221	< 0.01	276	24.9	24.9	0.0	0.0	24.9

Table 9. Results of EX, SS and MI for $n = 11$.

C	EX		SS				MI		$\delta_{2,\max}$	$\delta_{2,\min}$	$\gamma_2\,(\tilde{\sigma}_{\min})$	$\gamma_2\,(\tilde{\sigma}_{\max})$	$\gamma_2\,(\sigma_{\mathrm{MI}})$
	T_{EX}	$z_2\,(\sigma^*)$	T_{avg}	$z_2\,(\tilde{\sigma}_{\min})$	$z_2\,(\tilde{\sigma}_{\mathrm{avg}})$	$z_2\,(\tilde{\sigma}_{\max})$	T_{MI}	$z_2\,(\sigma_{\mathrm{MI}})$					
10	216	8	1.11	8	8	8	< 0.01	10	25.0	25.0	0.0	0.0	25.0
30	216	23	1.13	23	23	23	< 0.01	29	26.1	26.1	0.0	0.0	26.1
50	218	66	1.16	66	66	66	< 0.01	80	21.2	21.2	0.0	0.0	21.2
70	219	87	1.19	87	87	87	< 0.01	109	25.3	25.3	0.0	0.0	25.3
100	219	121	1.21	121	121	121	< 0.01	148	22.3	22.3	0.0	0.0	22.3
150	220	186	1.23	186	186	186	< 0.01	228	22.6	22.6	0.0	0.0	22.6

Table 10. Results of EX, SS and MI for $n = 13$.

C	EX		SS				MI		$\delta_{2,\max}$	$\delta_{2,\min}$	$\gamma_2\,(\tilde{\sigma}_{\min})$	$\gamma_2\,(\tilde{\sigma}_{\max})$	$\gamma_2\,(\sigma_{\mathrm{MI}})$
	T_{EX}	$z_2\,(\sigma^*)$	T_{avg}	$z_2\,(\tilde{\sigma}_{\min})$	$z_2\,(\tilde{\sigma}_{\mathrm{avg}})$	$z_2\,(\tilde{\sigma}_{\max})$	T_{MI}	$z_2\,(\sigma_{\mathrm{MI}})$					
10	33845	19	3.11	19	19	19	< 0.01	25	31.6	31.6	0.0	0.0	31.6
30	33884	35	3.26	35	35	35	< 0.01	47	34.3	34.3	0.0	0.0	34.3
50	33889	62	3.22	62	62	62	< 0.01	77	24.2	24.2	0.0	0.0	24.2
70	33904	69	3.41	69	69	69	< 0.01	84	21.7	21.7	0.0	0.0	21.7
100	33915	70	3.54	70	70	70	< 0.01	88	25.7	25.7	0.0	0.0	25.7
150	33926	161	3.65	161	161	161	< 0.01	198	23.0	23.0	0.0	0.0	23.0

Table 11. Results of SS and MI for $n = 120$.

C	SS				MI		$\delta_{2,\max}$	$\delta_{2,\min}$
	T_{avg}	$z_2\,(\tilde{\sigma}_{\min})$	$z_2\,(\tilde{\sigma}_{\text{avg}})$	$z_2\,(\tilde{\sigma}_{\max})$	T_{MI}	$z_2\,(\sigma_{\text{MI}})$		
10	2284	226	229	236	< 0.1	265	12.3	17.3
30	2293	456	466	482	< 0.1	532	10.4	16.7
50	2295	495	499	506	< 0.1	563	11.3	13.7
70	2312	584	593	609	< 0.1	676	11.0	15.8
100	2323	857	876	892	< 0.1	995	11.5	16.1
150	2341	1621	1657	1695	< 0.1	1857	9.6	14.6

Table 12. Results of SS and MI for $n = 160$.

C	SS				MI		$\delta_{2,\max}$	$\delta_{2,\min}$
	T_{avg}	$z_2\,(\tilde{\sigma}_{\min})$	$z_2\,(\tilde{\sigma}_{\text{avg}})$	$z_2\,(\tilde{\sigma}_{\max})$	T_{MI}	$z_2\,(\sigma_{\text{MI}})$		
10	3612	596	604	621	< 0.1	690	11.1	15.8
30	3625	849	861	874	< 0.1	966	10.5	13.8
50	3642	1449	1491	1523	< 0.1	1669	9.6	15.2
70	3651	2035	2056	2086	< 0.1	2299	10.2	13.0
100	3653	2937	2989	3028	< 0.1	3354	10.8	14.2
150	3666	3842	3849	3855	< 0.1	4297	11.5	11.8

Table 13. The average values of $\delta_{2,\max}$ and $\delta_{2,\min}$ for different n.

n	9	11	13	80	120	160
$\delta_{2,\max,\text{avg}}$	29.09	23.75	26.75	12.63	11.01	10.42
$\delta_{2,\min,\text{avg}}$				17.20	15.68	13.96

$z_3(\tilde{x}_{\text{avg}})$ and $z_3(\tilde{x}_{\max})$ denote, respectively, the minimum, the average and the maximum value of z_3. Additionally, $z_3(x^*)$ and $z_3(x_{\text{MI}})$ denote, respectively, the quality of optimal solution x^* and of 2-approximate MI solution x_{MI}. The performance indices are now expressed as:
$\delta_{3,\max} = \frac{z_3(x_{\text{MI}})-z_3(\tilde{x}_{\max})}{z_3(\tilde{x}_{\max})} \cdot 100\%$,
$\delta_{3,\min} = \frac{z_3(x_{\text{MI}})-z_3(\tilde{x}_{\min})}{z_3(\tilde{x}_{\min})} \cdot 100\%$,
$\gamma_3(x) = \frac{z_3(x)-z_3(x^*)}{z_3(x^*)} \cdot 100\%$ where $x \in \{\tilde{x}_{\min}, \tilde{x}_{\max}, x_{\text{MI}}\}$.

Experiments for $n = 10$, $m = 2$ (Table 14) enable us to compare SS and MI solutions with optimal ones. The result is promising, i.e.: the lowest and the highest quality SS solutions are at most by 4.3% and 1.1% worse than the corresponding optimal ones, respectively (see $\gamma_3(\tilde{x}_{\min})$ and $\gamma_3(\tilde{x}_{\max})$). The quality of MI solutions differs by at least 12.3% from the optimal solutions ($\gamma_3(\tilde{x}_{\text{MI}})$). According to the results presented in Tables 15–18, values of $\delta_{3,\max}$ and $\delta_{3,\min}$, for $m = 5$ as well as different n and C, belong to intervals $[4.7\%, 11.8\%]$ and $[7\%, 15.7\%]$, respectively. The average values of $\delta_{3,\max}$ and $\delta_{3,\min}$, which are presented in Table 19, descend for increasing values of n.

The execution time of SS for the largest tested problem instance exceeds 3 hours (Table 18: $n = 100$, $m = 5$, $C = 150$), while for MI it is equal 60.4. This is caused by the necessity to solve the deterministic problem many times, especially by the Improvement Method procedure.

Concluding, we recommend applying SS when the quality of solutions is critical and MI when it is necessary to have solutions in a short time.

5. Final remarks

The paper focuses on heuristic algorithms solving uncertain min-max regret task scheduling problems with interval uncertainty. It is assumed that values of execution times of tasks belong to intervals of known bounds. All uncertain problems refer to their deterministic counterparts of different properties, namely to: $R\|\sum C_j$ and $F2\|C_{\max}$, which are polynomially solvable, and to NP-hard $P\|C_{\max}$.

The uncertain problems investigated have various properties and differ with respect to the existence of approximate schemes; however, all of them turned out NP-hard irrespective of the time complexity of their deterministic counterparts. Therefore, Scatter Search based heuristic algorithms have been elaborated to solve them efficiently. It is worth noting that when the deterministic counterpart is NP-hard then it is not possible to determine any approximate algorithms for the corresponding uncertain version. Heuristics are only possible as time efficient solution procedures. For uncertain problems with easy deterministic counterparts, i.e. for $R\|\sum C_j$ and $F2\|C_{\max}$, it is possible to search

Table 14. Results of EX, SS and MI for $n = 10$, $m = 2$.

C	EX		SS				MI		$\delta_{3,\max}$	$\delta_{3,\min}$	$\gamma_3\left(\tilde{\sigma}_{\min}\right)$	$\gamma_3\left(\tilde{\sigma}_{\max}\right)$	$\gamma_3\left(\sigma_{\mathrm{MI}}\right)$
	T_{EX}	$z_3\left(x^*\right)$	T_{avg}	$z_3\left(\tilde{x}_{\min}\right)$	$z_3\left(\tilde{x}_{\mathrm{avg}}\right)$	$z_3\left(\tilde{x}_{\max}\right)$	T_{MI}	$z_3\left(x_{\mathrm{MI}}\right)$					
10	24740	57	1.88	57	57	57	< 0.01	64	12.3	12.3	0.0	0.0	12.3
30	24774	95	1.87	96	97	99	< 0.01	111	12.1	15.6	1.1	4.2	16.8
50	24791	220	1.96	220	220	220	< 0.01	248	12.7	12.7	0.0	0.0	12.7
70	24812	197	1.99	199	202	205	< 0.01	232	13.2	16.6	1.0	4.1	17.8
100	24819	322	2.25	322	330	336	< 0.01	372	10.7	15.5	0.0	4.3	15.5
150	24835	497	2.92	497	507	518	< 0.01	565	9.1	13.7	0.0	4.2	13.7

Table 15. Results of SS and MI for $n = 10$, $m = 5$.

C	SS				MI		$\delta_{3,\max}$	$\delta_{3,\min}$
	T_{avg}	$z_3\left(\tilde{x}_{\min}\right)$	$z_3\left(\tilde{x}_{\mathrm{avg}}\right)$	$z_3\left(\tilde{x}_{\max}\right)$	T_{MI}	$z_3\left(x_{\mathrm{MI}}\right)$		
10	3.43	87	88	90	< 0.01	98	8.9	12.6
30	3.58	98	99	101	< 0.01	111	9.9	13.3
50	3.66	79	80	81	< 0.01	89	9.9	12.7
70	3.75	152	156	160	< 0.0	175	9.4	15.1
100	3.76	132	134	136	< 0.01	149	9.6	12.9
150	3.82	172	175	178	< 0.01	199	11.8	15.7

Table 16. Results of SS and MI for $n = 20$, $m = 5$.

C	SS				MI		$\delta_{3,\max}$	$\delta_{3,\min}$
	T_{avg}	$z_3\left(\tilde{x}_{\min}\right)$	$z_3\left(\tilde{x}_{\mathrm{avg}}\right)$	$z_3\left(\tilde{x}_{\max}\right)$	T_{MI}	$z_3\left(x_{\mathrm{MI}}\right)$		
10	67	327	333	335	< 0.1	356	6.3	8.9
30	74	219	223	225	< 0.1	244	8.4	11.4
50	80	319	324	325	< 0.1	356	9.5	11.6
70	78	304	306	309	< 0.1	335	8.4	10.2
100	81	265	272	276	< 0.1	296	7.2	11.7
150	83	304	309	316	< 0.1	344	8.9	13.2

Table 17. Results of SS and MI for $n = 50$, $m = 5$.

C	SS				MI		$\delta_{3,\max}$	$\delta_{3,\min}$
	T_{avg}	$z_3\,(\tilde{x}_{\min})$	$z_3\,(\tilde{x}_{\text{avg}})$	$z_3\,(\tilde{x}_{\max})$	T_{MI}	$z_3\,(x_{\text{MI}})$		
10	5209	1509	1528	1542	3.31	1667	8.1	10.5
30	5236	1587	1595	1602	3.36	1712	6.9	7.9
50	5304	1747	1756	1769	3.34	1915	8.3	9.6
70	5401	1841	1871	1896	3.54	2026	6.9	10.0
100	5385	1730	1752	1768	3.41	1916	8.4	10.8
150	5473	1977	1998	2033	3.57	2186	7.5	10.6

Table 18. Results of SS and MI for $n = 100$, $m = 5$.

C	SS				MI		$\delta_{3,\max}$	$\delta_{3,\min}$
	T_{avg}	$z_3\,(\tilde{x}_{\min})$	$z_3\,(\tilde{x}_{\text{avg}})$	$z_3\,(\tilde{x}_{\max})$	T_{MI}	$z_3\,(x_{\text{MI}})$		
10	12878	4936	5001	5055	52.7	5350	5.8	8.4
30	12895	7097	7166	7221	59.3	7745	7.3	9.1
50	12913	7102	7140	7174	56.2	7598	5.9	7.0
70	12974	7637	7700	7731	59.9	8091	4.7	5.9
100	12926	8174	8292	8375	57.8	8772	4.7	7.3
150	13051	8295	8376	8462	60.4	8903	5.2	7.3

Table 19. The average values of $\delta_{3,\max}$ and $\delta_{3,\min}$ for different n.

n	10	20	50	100
$\delta_{3,\max,\text{avg}}$	9.90	8.13	7.66	5.60
$\delta_{3,\min,\text{avg}}$	13.71	11.16	9.89	7.51

for approximate schemes. The conducted computational experiments confirmed that Scatter Search algorithms are competitive, useful and outperform the quality of results generated by simple middle intervals heuristics, which turned out to be the 2-approximate solution algorithms for $R\|\sum C_j$ and $F2\|C_{\max}$. The comparison of Scatter Search algorithms with exact solutions for limited number of small instances is also promising. The drawback of the Scatter Search algorithms is connected with their longer execution times, especially for uncertain $R\|\sum C_j$. Another important aspect of the uncertain problems investigated deals with the determination of the worst-case scenarios and, in a consequence, the efficient calculation of the values of criteria. This can be a difficult task regardless of the complexity of deterministic counterparts. The Scatter Search heuristic algorithms proposed in the paper are a good basis for the development of this research direction both for the problems considered and for other uncertain task scheduling problems. In particular, other metaheuristics, e.g. Tabu Search, are worth investigating. More profound comparison with exact solutions is also desirable, especially with application of a professional solver.

References

Ayyub, B.M. and 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. (2005) Complexity of the min-max and min-max regret assignment problem. *Operations Research Letters* **33** (6), 634-640.

Aissi,H., Bazgan, hspace-2ptC., 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**(2), 57-65.

Averbakh, I. and Pereira, J. (2011) Exact and heuristic algorithms for the interval data robust assignment problem. *Computers & Operations Research*, **38**(8), 1153-1163.

Bubnicki, Z. (2004) *Analysis and Decision Making in Uncertain Systems.* Springer Verlag, Berlin-London-New York.

Conde, E. (2007) Minmax regret location—allocation problem on a network under uncertainty. *European Journal of Operational Research*, **179**(3), 1025-1039.

Conde, E. (2009) A minmax regret approach to the critical path method with task interval times. *European Journal of Operational Research*, **197** (1), 235-242.

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

Corberan, A., Fernandez, A.E., Laguna, M., Marti, R. (2002) Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society*, **53**, 427-435.

Garey, M.R. and Johnson, D.S. (1978) Strong NP-completeness results: motivation, examples, and implications. *J. Assoc. Comput. Mach.* **25**(3), 499-508.

Glover, F. (1997) A template for scatter search and path relinking. In: *Proceeding AE '97 Selected Papers from the Third European Conference on Artificial Evolution* Nimes, France. IEEE Computer Society, 3-54.

Glover, F., Laguna, M. and Marti, R.(2000) Fundamentals of scatter search and path relinking. *Control & Cybernetics*, **29**(3), 653-684.

Gutierrez, G.J., Kouvelis, P., Kurawarwala, A.A. (1996) A robust approach to uncapacitated network design problems. *European Journal of Operational Research*, **94** (2), 362-376.

Johnson, S.M. (1954) Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, **1**(1), 61-68.

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

Jozefczyk, J. and Siepak, M. (2011) Minmax regret algorithms for uncertain $P\|C_{\max}$ problem with interval processing times. In: H. Selvaraj and D. Zydek, eds., *Proceedings of 21'st International Conference on Systems Engineering*, Las Vegas, USA. IEEE Computer Society, Los Alamitos, 115–119.

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

Jungnickel, D. (2008) *Graphs, Networks and Algorithms.* Springer, Berlin-Heidelberg-New York.

Kasperski, A. (2008) *Discrete Optimization with Interval Data: Minmax Regret and Fuzzy Approach. Studies in Fuzziness and Soft Computing.* Springer, Berlin-Heidelberg-New York.

Kasperski, A. and 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., Makuchowski, M., Zielinski, P. (2012) A tabu search algorithm for the minmax regret minimum spanning tree problem with interval data. *Journal of Heuristics*, **18**(4), 593-625.

Kasperski, A., Kurpisz, A., Zielinski, P. (2013) Approximating the minmax (regret) selecting items problem. *Information Processing Letters*, **113**(1-2), 23-29.

Klir, G.J. (2006) *Uncertainty and Information: Foundations of Generalized Information Theory.* Wiley.

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

Kouvelis, P., Daniels, R.L., Vairaktaris, G. (2000) Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions*, **32**(5), 421-432.

Laguna, M. and Marti, R. (2003) *Scatter Search–Methodology and Implementations.* Kluwer Academic Publishers, Dortrecht-Boston-London.

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

Nowicki, E. and Smutnicki, C. (2006) Some aspects of scatter search in the flow-shop problem. *European Journal of Operational Research*, **169**, 654-666.

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

Savage, L.J. (1951) The theory of statistical decision. *Journal of American Statist. Assoc.,* **46**, 55-67.

Volgenant, A. and Duin, C.W. (2010) Improved polynomial algorithms for robust bottleneck problems with interval data. *Computers & Operations Research*, **37**(5), 909-915.

Xu, J., Chiu, S.Y., Glover, F. (2001) Tabu Search and Evolutionary Scatter Search for 'Tree-Star' Network Problems, with Applications to Leased-Line Network Design. In: *Telecommunications Optimization: Heuristic and Adaptive Techniques*, Chapter 4. John Wiley & Sons.