Decision making under interval uncertainty - selected models and algorithms

Jerzy Józefczyk

Division of Intelligent Decision Support Systems
Wrocław University of Technology, Wrocław, POLAND

e-mail: Jerzy.Jozefczyk@pwr.edu.pl
## Figures

<table>
<thead>
<tr>
<th>Category</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields of study</td>
<td>39</td>
</tr>
<tr>
<td>Students</td>
<td>35,252</td>
</tr>
<tr>
<td>Academic staff</td>
<td>2,000</td>
</tr>
<tr>
<td>Professors</td>
<td>395</td>
</tr>
<tr>
<td>Associate professors</td>
<td>240</td>
</tr>
<tr>
<td>PhD holders</td>
<td>1,223</td>
</tr>
<tr>
<td>Campus area (ha)</td>
<td>87.6</td>
</tr>
<tr>
<td>Buildings</td>
<td>251</td>
</tr>
</tbody>
</table>

### Budget:
- budget grant for teaching and research activities
- own income from teaching and research activity
Faculties

Faculty of Architecture
Faculty of Civil Engineering
Faculty of Chemistry
Faculty of Electronics
Faculty of Electrical Engineering
Faculty of Geoengineering, Mining and Geology
Faculty of Environmental Engineering

Faculty of Computer Science and Management
Faculty of Mechanical and Power Engineering
Faculty of Mechanical Engineering
Faculty of Fundamental Problems of Technology
Faculty of Microsystems Electronics and Photonics
Faculty of Mathematics

Regional Branches in Jelenia Góra, Legnica, Wałbrzych
Division of Intelligent Decision Support Systems

Problems

- Modelling of decision making (management) systems
- Complex decision making problems
  
  **Decision making for uncertain systems**

- Big data processing and machine learning

Methods

- **Classic methods of** modelling and optimization
- Methods of computational intelligence
- Methods of multi-criteria decision making.
- Methods of **uncertain** (non-deterministic) **knowledge representation**, e.g. interval uncertainty.

Applications

- **Computer systems and networks**, e.g. routing, rate control, distribution of data.
- Complex manufacturing and multi-robot systems, e.g. logistic systems and supply networks, wireless sensor networks.
- Biomedical systems.
1. Decision making for input-output plants.
2. Parameter uncertainty and its selected representations.
3. Determinization (substantiation) of evaluation function.
4. Selected decision making problems under interval uncertainty.
   a. Time-optimal resource allocation.
   b. Task scheduling minimizing the sum of completion times.
   c. Task scheduling minimizing the makespan.
   d. Permutational flow-shop
5. Final remarks
Decision making for input-output plants

$u$ – decision

$y = F(u,a)$ – evaluation of the decision $u$

**Special cases of decision making problems**

**Case A** \( \varphi(y) : y = y' \) (or \( y \in Y \))

Given: \( F, \varphi(y) \),

Find: \( u' \) to fulfil \( \varphi(y) \)

**Case B** \( \varphi(y) = \varphi(F(u,a)) \triangleq Q(u,a) \)

Given: \( F, Q \),

Find \( u' \) to minimise (maximise) \( Q \), i.e.

\[
   u' = \arg \min_{u \in U} Q(u,a) \quad [u' = \arg \max_{u \in U} Q(u,a)]
\]

– optimisation problem
Parameter uncertainty and its selected representations (1/2)

\[
\text{decision} \quad \text{System (Process)} \quad \text{evaluation of decision}
\]

\(u\) – decision / dependent on parameter \(a\) /

\(U(a)\) – set of feasible decisions / dependent on parameter \(a\) /

\(F(u,a)\) – evaluation of the decision \(u\)

\(a = [a^{(1)}, a^{(2)}, ..., a^{(k)}, ..., a^{(m)}]^{T} - m\) dimensional unknown parameter \(a\)

\(a \in A \subseteq \mathbb{R}^{m} - s\) et of feasible values of parameter \(a\)

In particular, \(a^{(k)} \in [a^{(k)}, \bar{a}^{(k)}], \ a^{(k)} \leq \bar{a}^{(k)} , \ A = \times_{k=1}^{m} [a^{(k)}, \bar{a}^{(k)}] \)

In discrete case, \(a^{(k)} \in \{a_{1}^{(k)}, a_{2}^{(k)}, ..., a_{n_{k}}^{(k)}\}, \ A = \times_{k=1}^{m} \{a_{1}^{(k)}, a_{2}^{(k)}, ..., a_{n_{k}}^{(k)}\} \)

It is assumed that set \(A\) is the only information on uncertain parameter \(a\).

A specific realization of parameter \(a\) is often called a \textbf{scenario}. 
Parameter uncertainty and its selected representations (2/2)

The most known representations of the uncertainty:

- Probabilistic
- Fuzzy
- Grey
- Uncertain variables /credibility
- Possibilistic

**Determinization of the evaluation function (1/8)**

**Calculation of the evaluation function /in the presence of interval uncertainty/**

The basis for such calculation may be directly evaluation function \( F(u,a) \) or any term built on it. Additionally, different operators for aggregation can be used.

**Aggregation of \( F(u,a) \) values**

\[
\begin{align*}
\max_{a \in A} F(u,a) \\
\min_{a \in A} F(u,a) \\
\beta \max_{a \in A} F(u,a) + (1 - \beta) \min_{a \in A} F(u,a), \quad \beta \in [0,1] \\
\int_{A} F(u,a) da
\end{align*}
\]
Determinization of the evaluation function (2/8)

Aggregation of terms containing $F(u,a)$

$F(u,a) - F'(a), \ a \in A$ for a given $u$ – regret (opportunity loss)

where $F'(a) = \min_{u \in U} F(u,a)$, $U$ – set of feasible decisions

Regret (opportunity loss) is a difference between the cost of a specific decision and the corresponding cost of the optimal decision – for any given realization of unknown parameter.

$\frac{F(u,a)}{F'(a)}, \ a \in A$ – relative regret

We are interested in decisions acceptable irrespective of a specific scenario $a$. The notion ‘robustness’ is used.

Decision is robust if it is ‘good’ or at least ‘not so bad’ for all scenarios.

Decision is robust if its realization prevents undesirable or deteriorating behavior of decision object.
Determinization of the evaluation function (4/8)

*p*-robustness

Due to uncertain \( a \) we do not seek for optimal decisions.

\[
F(u, a) - F'(a) < p \rightarrow u \in U(a) - p - \text{optimal decision}
\]

\[
U_p(a) = \{ u \in U(a) : F(u, a) - F'(a) < p \} - \text{set of all } p - \text{optimal decisions}
\]

If we require the inequality for all scenarios, we have \textbf{p-robust decision} /used mainly for discrete \( A \)/.

\[
\forall_{a \in A} F(u, a) - F'(a) < p, \quad \text{and} \quad u \in \bigcap_{a \in A} U_p(a)
\]

In general, the \textbf{set of p-robust decisions} can be obtained:

\[
\overline{U}_p = \{ u \in U : \forall_{a \in A} F(u, a) - F'(a) \leq p \}
\]

The solution is close to the optimal one for all scenarios.
Determinization of the evaluation function (5/8)

Minmax regret approach (*worst-case regret approach*)

Instead of considering all scenarios, it is enough to take into account only the worst scenario, i.e.

\[ z(u) = \max_{a \in A} [F(u, a) - F'(a)], \quad u \in U \]

Instead of postulating \( z(u) < p \) seeking for the minimal \( p \), i.e.

\[ \min_{u \in U} z(u) = \min_{u \in U} \max_{a \in A} [F(u, a) - F'(a)] \]

\[ z(u) = \max_{a \in A} [F(u, a) - F'(a)] \quad \text{– measure of robustness} \]

\[ u^*, \quad z(u^*) = \min_{u \in U} z(u) \quad \text{– results of solving minmax regret problem} \]
Determinization of the evaluation function (6/8)

Other measures of robustness are encountered in literature.

\((b,w) \rightarrow \text{robustness},\) parametric robustness, used for discrete \(A\) [proposed by B. Roy],

\(b, w \ (b < w)\) – non-negative parameters

Robust decision maximizes the number of scenarios for which the value of evaluation (e.g. the value of regret) does not exceed \(b\) as well as this value for all scenarios does not exceed \(w\).

\[
\max_{u \in U} \sum_{i=1}^{N} 1[b - (F(u,a_i) - F'(a_i))], \ N \rightarrow \text{number of all scenarios}
\]

s. t. \(F(u,a_i) \leq w, \ i = 1, 2, ..., N\)
**Lexicographic \( \alpha \)– robustness**, parametric robustness, used for discrete \( A \)

Non-increasing sequence of the evaluation function values is determined for all scenarios, i.e.:

\[
F(u, a_i) \geq F(u, a_{i+1}), \quad i = 1, 2, \ldots, N - 1, \quad N - \text{number of all scenarios}
\]

Set of \( \alpha \) – robust lexicographic decisions is derived.

\[
\tilde{U}(\alpha) = \{u \in U : \forall j \leq n \quad F(u, a_j) - F'(a_j) \leq \alpha\}
\]

Lexicographic \( \alpha \) – robustness implies \( p \)-robustness but not vice versa.
Determinization of the evaluation function (8/8)

Other minmax \textit{(non-parametric)} approaches

\[
\max_{a \in A} F(u, a) \quad \text{– robust (worst-case) criterion}
\]

\[
\max_{a \in A} [F(u, a) - F'(a)] \quad \text{– robust (worst-case) regret criterion}
\]

\[
\max_{a \in A} \frac{F(u, a)}{F'(a)} \quad \text{– robust (worst-case) relative regret criterion}
\]

Robust optimal decision making

\[
\min_{u \in U} \max_{a \in A} F(u, a)
\]

\[
\min_{u \in U} \max_{a \in A} [F(u, a) - F'(a)]
\]

\[
\min_{u \in U} \max_{a \in A} \frac{F(u, a)}{F'(a)}
\]

Challenge: Three nested optimization problems.


Time-optimal resource allocation (1/5)

Special case of continuous functions of the form \( F(u,a) = \max_{i=1,2,...,n} a_i u_i \)

\[
T_i = \frac{1}{p_i c_i}, \quad i = 1, 2, ..., n \quad \text{where} \quad p_i > 0 \quad \text{– parameters} \quad /a_i = \frac{1}{p_i}, \quad u_i = \frac{1}{c_i} /
\]

\( C \) – global amount of resources \( D_c = \{ c \Delta [c_1, c_2, ..., c_n]^T : c_i \geq 0, \ i = 1, 2, ..., n, \ \sum_{i=1}^{n} c_i \leq C \}

\[
Q(p,c) = \max_{i=1,2,...,n} \frac{1}{p_i c_i} \quad \text{where} \quad p = [p_1, p_2, ..., p_I]^T
\]

Deterministic time optimal resource allocation problem \( Q(p,c') \Delta Q'(p) = \min_{c \in D_c} Q(p,c) \)

Analytical solution: \( c_i' = C \frac{p_i}{\sum_{m=1}^{I} \frac{1}{p_m}}, \ i = 1, 2, ..., I, \quad Q'(p) = \frac{1}{C} \sum_{m=1}^{I} \frac{1}{p_m} \)
Let us find such an index \( s \) that 
\[
\frac{1}{\bar{p}_s u_s} \geq \frac{1}{p_i c_i}, \quad s \neq i.
\]
Then
\[
z(c) = \max_{p \in P} \left[ \frac{1}{p_s c_s} - \frac{1}{C \sum_{m=1}^{n} \frac{1}{p_m}} \right] = \max_{p \in P} \left[ \frac{1}{p_s c_s} - \frac{1}{p_s C} - \frac{1}{C \sum_{m=1, m \neq s}^{n} \frac{1}{p_m}} \right] = \max_{p \in P} \left[ \frac{C - c_s}{p_s c_s C} - \frac{1}{C \sum_{m=1, m \neq s}^{n} \frac{1}{p_m}} \right].
\]

It is easy to see that \( p^c = [\bar{p}_1, \ldots, \bar{p}_s, \ldots, \bar{p}_n]^T \) is the worst scenario and
\[
z(c) = \frac{1}{p_s c_s} - Q'(p^w) = \frac{1}{p_s c_s} - \frac{1}{C \left( \frac{1}{p_s} + \sum_{m=1, m \neq s}^{n} \frac{1}{\bar{p}_m} \right)}.
\]


Theorem. The solution $c^*$ of the uncertain time-optimal resource allocation problem is optimal if and only if the conditions

$$\sum_{i=1}^{n} c_i^* = C$$

$$\frac{1}{p_i c_i} = \frac{1}{p_s c_s}, \quad s = 1, 2, ..., n, \quad s \neq i$$

are fulfilled, and $s$ is the longest lasting operation.
Time-optimal resource allocation (4/5)

Optimal polynomial solution algorithm

The optimal resource allocation $c^*$ for operations $i = 1, 2, \ldots, n$, $i \neq s$ is as follows:

1. Calculate $\hat{c}_i = C \frac{1}{\bar{p}_i + \sum_{m=1}^{n} \frac{1}{p_m}}$, $i = 1, 2, \ldots, n$ and find index $s$ and $c_s^*$ such that $c^*_s = \max_i \hat{c}_i$.

2. Determine $c^*_i = C \frac{p_i}{\bar{p}_s + \sum_{m=1}^{n} \frac{1}{p_m}}$, $i = 1, 2, \ldots, n$, $i \neq s$.

The optimal value of the absolute regret $z(c^*) = \frac{1}{C} \sum_{m=1}^{n} \frac{\bar{p}_s p_m - p_s p_m}{p_s p_m \bar{p}_m}$.

(A)
Time-optimal resource allocation (5/5)

Numerical example: \( n = 2, \quad p_1 \in [1, 3], \ p_2 \in [2, 4], \ c_1 + c_2 = 10 \quad (C = 10) \)

Step 1. \[ \hat{c}_1 = C \frac{1}{\frac{1}{p_1} + \frac{1}{p_2}} = 10 \frac{1}{\frac{1}{3} + \frac{1}{2}} = 4, \quad \hat{c}_2 = C \frac{1}{\frac{1}{p_2} + \frac{1}{p_1}} = 10 \frac{1}{\frac{1}{4} + \frac{1}{1}} = 2, \]

Therefore, \( s = 1, \) and \( c_1^* = 4. \)

Step 2. \( c_2^* = 6 \)

Moreover, \( z(c^*) = \frac{1}{8}. \)
Task scheduling minimizing the sum of completion times (1/5)

\[
R \mid p_{i,j} \leq p_{i,j} \leq \overline{p}_{i,j} \mid \sum C_j \quad \text{notation of the problem}
\]

\[
J = \{J_1, J_2, \ldots, J_j, \ldots, J_n\} \quad \text{set of } n \text{ tasks}
\]

\[
M = \{M_1, M_2, \ldots, M_i, \ldots, M_m\} \quad \text{set of } m \text{ machines (executors)}
\]

\[
p = [p_{i,j}]_{i=1,2,\ldots,m}^{j=1,2,\ldots,n} \quad \text{matrix of task execution times}
\]

\[
x_{j,k,i} = \begin{cases} 
1, & \text{\(j\) th task is performed as the } k \text{ th from the end by } i \text{ th machine} \\
0, & \text{otherwise}
\end{cases}
\]

\[
\sum_{i=1}^{m} \sum_{k=1}^{n} x_{j,k,i} = 1, \ j = 1,n \quad \text{each task must be executed once}
\]

\[
\sum_{j=1}^{n} x_{j,k,i} \leq 1, \ i = 1,m, \ j = 1,n \quad \text{each machine can perform at most one task at every time moment}
\]

\[
F_1(p, x) = \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{k=1}^{n} k p_{i,j} x_{j,k,i}
\]
Problem formulation (deterministic version):

Given: \( J, M, p \)

Find: the optimal schedule \( x' \), i.e.

\[
F_1'(p) \triangleq F_1(p, x') = \min_x F_1(p, x) = \min_x \sum_{j=1}^{n} \sum_{i=1}^{m} \sum_{k=1}^{n} p_{i,j} x_{j,k,i}
\]
Task scheduling minimizing the sum of completion times (3/5)

\[ p_{i,j} \in [\underline{p}_{i,j}, \bar{p}_{i,j}], \quad p_{i,j} \leq \bar{p}_{i,j} \] – interval execution times

\[ p \in \mathbf{P} = [\underline{p}_{1,1}, \bar{p}_{i,j}] \times \ldots \times [\underline{p}_{m,n}, \bar{p}_{m,n}] \]

\[ F_1(p, x) - F'_1(p) \] – regret

\[ z_1(x) = \max_{p \in \mathbf{P}} [F_1(p, x) - F'_1(p)] \] – evaluation function (robust regret criterion)

**Problem formulation (uncertain version):**

Given: \( J, M, p \)

Find: the optimal schedule \( x^* \), i.e.

\[ z_1^* \triangleq z_1(x^*) = \min_x z_1(x) \]

\[ z_1(x) = \max_{p \in \mathbf{P}} [F_1(p, x) - F'_1(p)] = F_1(p^x, x) - F'_1(p^x), \text{ where } p^x - \text{ the worst scenario for } x \]
Important results:

• Reducing the calculation of the worst scenario to the weighted bipartite matching problem solved polynomially by the Hungarian algorithm.

• Proofing of the 2-approximate property for the mid-point scenario (MIH).

• Elaboration and experimental evaluation of heuristic Scatter Search (SS) algorithm.

\[ p_{i,j}^{\text{MIH}} = \frac{p_{i,j} + \bar{p}_{i,j}}{2} \quad \text{- the mid-point scenario} \quad \rightarrow \quad z_1(x^{\text{MIH}}) \leq 2z_1(x^*). \]
### Task scheduling minimizing the sum of completion times (5/5)

**Task Scheduling Problem**

When scheduling tasks on unrelated machines, the goal is to minimize the sum of completion times. This problem is addressed in the context of interval processing times and the total flow time criterion.

#### Example

Let's consider a scenario where there are 10 tasks ($n = 10$) and 2 machines ($m = 2$). We will evaluate the performance of different scheduling algorithms using the metrics $z_1(x^*)$, $z_1(x^{\text{SS}})$, and $z_1(x^{\text{MIH}})$.

<table>
<thead>
<tr>
<th>$C$</th>
<th>$z_1(x^*)$</th>
<th>$z_1(x^{\text{SS}})$</th>
<th>$z_1(x^{\text{MIH}})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>57</td>
<td>57</td>
<td>64</td>
</tr>
<tr>
<td>30</td>
<td>95</td>
<td>97</td>
<td>111</td>
</tr>
<tr>
<td>50</td>
<td>220</td>
<td>220</td>
<td>248</td>
</tr>
<tr>
<td>70</td>
<td>197</td>
<td>202</td>
<td>232</td>
</tr>
<tr>
<td>100</td>
<td>322</td>
<td>330</td>
<td>372</td>
</tr>
<tr>
<td>150</td>
<td>497</td>
<td>507</td>
<td>565</td>
</tr>
</tbody>
</table>

**Correction Factor**

The correction factor $\delta_1$ is calculated as:

$$\delta_1 = \frac{z_1(x^{\text{SS}}) - z_1(x^{\text{MIH}})}{z_1(x^{\text{MIH}})} \times 100\%$$

<table>
<thead>
<tr>
<th>$n$</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_1$</td>
<td>9.90</td>
<td>8.13</td>
<td>7.66</td>
<td>5.60</td>
</tr>
</tbody>
</table>

**References**

Task scheduling minimizing the makespan (1/4)

\[ R \mid p_{i,j} \leq p_{i,j} \leq \overline{p}_{i,j} \mid C_{\text{max}} \] – notation of the problem

\[ J = \{J_1, J_2, \ldots, J_j, \ldots, J_n\} \] – set of \( n \) tasks

\[ M = \{M_1, M_2, \ldots, M_i, \ldots, M_m\} \] – set of \( m \) machines (executors)

\[ p = [p_{j}]_{j=1,2,\ldots,n} \] – vector of execution times

\[ c_{j,i} = \begin{cases} 1, & j \text{ th task is performed by } i \text{ th machine} \\ 0, & \text{otherwise} \end{cases} \]

\[ c = [c_{j,i}]_{i=1,2,\ldots,m} \]

\[ \sum_{i=1}^{m} c_{j,i} = 1, \ j = 1, \ldots, n \] – each task must be executed once

\[ C_{\text{max}}(p, c) = \max_{i} \sum_{j=1}^{n} p_{j} c_{j,i} \]

Problem formulation (deterministic version): Given: \( J, M, p \)
Find: the optimal schedule \( c' \), i.e.

\[ C'_{\text{max}}(p) \triangleq C_{\text{max}}(p, c') = \min_{c} \max_{i} \sum_{j=1}^{n} p_{j} c_{j,i} \]
Task scheduling minimizing the makespan (2/4)

\[ p_j \in [\underline{p}_j, \bar{p}_j], \underline{p}_j \leq \bar{p}_j \] – interval execution times

\[ p \in \mathbf{P} = [\underline{p}_1, \bar{p}_1] \times \ldots \times [\underline{p}_n, \bar{p}_n] \]

\[ C_{\text{max}}(p,c) - C'_{\text{max}}(p) \] – regret

\[ z_2(c) = \max_{p \in \mathbf{P}} [C_{\text{max}}(p,c) - C'_{\text{max}}(p)] \] – evaluation function (robust regret criterion)

Problem formulation (uncertain version):

Given: \( \mathbf{J}, \mathbf{M}, p \)

Find: the optimal schedule \( c^* \), i.e.

\[ z_2^* \overset{\Delta}{=} z_2(c^*) = \min_c z_2(c) \]

\[ z_2(c) = \max_{p \in \mathbf{P}} [C_{\text{max}}(p,c) - C'_{\text{max}}(p)] = C_{\text{max}}(p^c, x) - C'_{\text{max}}(p^c) \]

where \( p^c \) – the worst scenario for \( c \)
Important results:

• Determination of the algorithm for calculating of lower and upper bounds of $z_2(c)$ for a given $c$.

• Development of heuristic algorithms (SS-based and MIH) as well as their experimental evaluation /lack of any approximate algorithm/.
### Task scheduling minimizing the makespan (4/4)

**n = 10, m = 2**

<table>
<thead>
<tr>
<th>$C$</th>
<th>$z_2(c^*)$</th>
<th>$z_2(x_{SS})$</th>
<th>$z_2(x_{MIH})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>21</td>
<td>21</td>
<td>31</td>
</tr>
<tr>
<td>50</td>
<td>38</td>
<td>40</td>
<td>58</td>
</tr>
<tr>
<td>70</td>
<td>56</td>
<td>56</td>
<td>75</td>
</tr>
<tr>
<td>100</td>
<td>114</td>
<td>119</td>
<td>155</td>
</tr>
<tr>
<td>150</td>
<td>135</td>
<td>136</td>
<td>195</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n$</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_2$</td>
<td>44.60</td>
<td>27.48</td>
<td>19.19</td>
<td>9.77</td>
</tr>
</tbody>
</table>

$$\delta_2 = \frac{z_2(x_{SS}) - z_2(x_{MIH})}{z_2(x_{MIH})} \times 100\%$$

Permutational flow-shop problem (1/8)

General scheme of the flow-shop problem (*tasks move along stationary executors*)

- set of tasks
- \( M_1 \)
- \( M_2 \)
- \( \ldots \)
- \( M_m \)

\[ J = \{J_1, J_2, \ldots, J_j, \ldots, J_n\} \] – set of \( n \) tasks

\[ M = \{M_1, M_2, \ldots, M_i, \ldots, M_m\} \] – set of \( m \) machines (executors)

\[ J_j = (O_{1,j}, O_{2,j}, \ldots, O_{m,j}) \] – operations for \( j \)th task

\[ p = [p_{i,j}]_{i=1,2,\ldots,m \atop j=1,2,\ldots,n} \] – matrix of operation execution times

\[ \pi = (\pi_1, \pi_2, \ldots, \pi_j, \ldots, \pi_n) \in \Pi \] – permutation of tasks (**decision**) where

\[ \pi_j \in \{1, 2, \ldots, n\}, \, \Pi = \{\pi : \pi_j \neq \pi_k, \, j, k \in \{1, 2, \ldots, n\}, \, j \neq k\} \]
Evaluation function (criterion)

$$C_{\text{max}}(p, \pi) = C_{m, \pi_n}(p, \pi), \text{ where}$$

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

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

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

Problem formulation:

Given: $J, M, p$

Find: the optimal permutation $\pi'$, i.e.

$$C'_{\text{max}}(p) \triangleq C_{\text{max}}(p, \pi') = \min_{\pi \in \Pi} C_{\text{max}}(p, \pi)$$
Permutational flow-shop problem (3/8)

\[ p_{i,j} \in [p_{i,j}, \bar{p}_{i,j}], \quad p_{i,j} \leq \bar{p}_{i,j} \] - interval execution times

\[ p \in \mathcal{P} = [p_{1,1}, \bar{p}_{i,j}] \times \cdots \times [p_{m,n}, \bar{p}_{m,n}] \]

\[ C_{\text{max}}(p, \pi) - C_{\text{max}}'(p) \] - regret

\[ z_3(\pi) = \max_{p \in \mathcal{P}} [C_{\text{max}}(p, \pi) - C_{\text{max}}'(p)] \] - evaluation function (robust regret criterion)

**Problem formulation:**

Given: \( J, M, p_{i,j}, \bar{p}_{i,j}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n \)

Find: the optimal permutation \( \pi^* \), i.e.

\[ z_3(\pi^*) = \min_{\pi \in \Pi} (\max_{p \in \mathcal{P}} [C_{\text{max}}(p, \pi) - C_{\text{max}}'(p)]) \]

This problem is at least NP-hard.

Moreover, three nested optimizations.
Uncertain flow-shop is NP-hard.

Case of two machines $m = 2$

Theorem. For mid-point scenario

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

$$ z_3(\pi^{\text{MIH}}) \leq 2 z_3(\pi^*). $$
Permutational flow-shop problem (5/8)

\[
\min(\max_{\pi \in \Pi} [C_{\max}(p, \pi) - \min_{\sigma \in \Pi} C_{\max}(p, \sigma)]) = \min(\pi (\pi))
\]

Inner minimization

\[
z_{3, LB}(\pi) = \max_{\pi \in \Pi} (C_{\max}(p, \pi) - C_{\max}(p, \sigma^{NEH})) - \text{lower bound of } z(\pi)
\]

\[
z_{3, UB}(\pi) = \max_{\pi \in \Pi} (C_{\max}(p, \pi) - C_{\max, LB}(p)) - \text{upper bound of } z(\pi)
\]

where

\[
C_{\max, LB}(p) = \max_{k=1,m} \left( \min_{j=1,n} \sum_{i=1}^{k-1} p_{i,j} + \sum_{j=1}^{n} p_{k,j} + \min_{l=1,n} \sum_{i=k+1}^{m} p_{i,l} \right)
\]

Worst-case scenario analysis (maximization)

It is easy to justify that \( \forall i, j : p_{i,j} = p_{i,j} \lor p_{i,j} = \overline{p}_{i,j} \) /extreme scenarios only/

\[2^{mn} - \text{number of all scenarios} \quad \frac{(m+n-2)!}{(m-1)!(n-1)!} = \text{number of scenarios to check}\]
Permutational flow-shop problem (6/8)

Representation of a feasible solution (chromosome):
Sequence of numbers from 1 to \( n \).

Fitness function:
\( z_{3,UB}(\pi) \)

Initial population:
10 random permutations + solution determined by constructive MIH algorithm and its 9 random versions.

Crossover:
Order crossover operator (Goldberg) giving feasible offsprings.

Mutation:
Two random genes are swapped.

Selection:
The current population is sorted non-increasingly. First two chromosomes are removed from the list and the result of their crossover is added to the new population. The process is repeated until the list is empty.
Middle interval heuristic (MIH)
Application od NEH constructive algorithm for

\[
p_{i,j}^{\text{MIH}} = \frac{p_{i,j} + \bar{p}_{i,j}}{2}
\]

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

Final remarks (1/2)

Current work:

• **Considering of multi-criteria problems**

• **Investigation of computational complexity and seeking for approximate algorithms for basic problems**

• **Developing of heuristic algorithms for problem instances more close to real-world applications**


Applications
Non-repetitive and unique processes (systems) as well as non-available and(or) non-existing) historical data.

Example
Producing of software by small/medium computer company:

Tasks: specific programming activities with changing execution times.
Aim: Minimization of the total execution time.