Multi-objective decision making and search space for the evaluation of production process scheduling

T. WITKOWSKI, P. ANTCZAK, and A. ANTCZAK

1 The Faculty of Management, Warsaw University of Technology, 85 Narbutta St., 02-524 Warsaw, Poland
2 The Faculty of Production Engineering, Warsaw University of Technology, 85 Narbutta St., 02-524 Warsaw, Poland

Abstract. Over the years, various approaches have been proposed in order to solve the multi-objective job-shop scheduling problem – particularly a hard combinatorial optimization problem. The paper presents an evaluation of job shop scheduling problem under multiple objectives (mean flow time, max lateness, mean tardiness, mean earliness, number of tardy tasks). The formulation of the scheduling problem has been presented as well as the evaluation schedules for various optimality criteria. The paper describes the basic metaheuristics used for optimization schedules and the approaches that use domination method, fuzzy method, and analytic hierarchy process (AHP) for comparing schedules in accordance with multiple objectives. The effectiveness of the algorithms has been tested on several examples and the results have been shown. New search space for evaluation and generation of schedules has been created. The three-dimensional space can be used for the analysis and control of the production processes.

Key words: production scheduling, multi-objective decision making, heuristics, data mining.

1. Introduction

In multicriteria decision making, it is assumed that a small set of alternatives are available from which a selection must be made on the basis of multiple factors. Often Multi-Attribute Utility Theory [1] is used to create a scalar-valued criterion for selecting from the decision set. Since the individual preferences of the decision maker is of prime importance in multicriteria decision making, many researches have approached the problem using interactive methods [2].

The feasibility of schedules is evaluated for various performance criteria, which may be single or multiple. A solution that is optimal for a given criterion may not be optimal for some other criterion. In many practical situations, it be thus desirable to achieve a solution that is best with respect to a number of different criteria simultaneously. The research on bi-criteria and multi-criteria scheduling can be categorized in four different types of models, viz: single machine-bi-criteria scheduling, single machine-multiple criteria scheduling, multiple machine-bi-criteria scheduling and multiple machine-multiple criteria scheduling. Most of the research done so far on multiple criteria scheduling involves only a single machine or two-machine flow shop [3, 4]. In [5] a multi-criteria dynamic scheduling algorithm by swapping of dispatching rules that minimizes/maximizes several performance measures simultaneously have been described. In [6] a hybrid metaheuristic, the Variable Neighborhood Particle Swarm Optimization have been introduced. The proposed method is used for solving the multi-objective Flexible Job Shop Scheduling Problems (FJSP). Other approaches to multi-objective decision making are shown in [7–14].

The paper is organized as follows: Sec. 2 formulates the flexible job shop problem (FJSP). Section 3 introduces recent research in solving the scheduling problems with the application of metaheuristics. In Sec. 4, the Pareto approach, the fuzzy method and the analytic hierarchy process (AHP) are used for the evaluation of the FJSP problem. In Sec. 5, schedule cluster recognition and evaluation of dependencies according to the data mining approach is analyzed. The representation of schedules in search space is presented. Section 6 formulates some concluding remarks.

2. Formulation of the scheduling problem

In the job-shop scheduling problem (JSP), there are \( n \) jobs and \( m \) machines, each job is to be processed on a group of machines satisfying precedence constraints. Each operation of job is to be processed only on one predetermined machine. Though the JSP has been well studied, its application to real-world scenarios is often undermined by the constraint of the one-to-one mapping of operations to machines. Hence, the flexible job-shop scheduling problem extends the JSP by allowing each operation to be processed on more than one machine. With this extension, we are now confronted with two subtask: assignment of each operation to an appropriate machine and sequencing operations on each machine.

The FJSP is formulated as follows. There is a set of jobs \( Z = \{Z_i\}, i \in I \), where \( I = \{1, 2, ..., n\} \) is an admissible set of parts, \( U = \{u_k\}, k \in 1, m \), is a set of machines. Each job \( Z_i \) is a group of parts \( \Pi_i \) of equal partial task \( p_i \) of a certain range of production. Operations of technological processing of the \( i \)-th part are denoted by \( O_{ij} \). Then for \( Z_i \), we can write \( Z_i = (\Pi_i\{O_{ij}\})_{i=\xi} \), where \( O_{ij} = (G_{ij}, t_{ij}(N)) \) is the \( j \)-th operation of processing the \( i \)-th group of parts; \( \xi \) is the number of operation of the production process at which one should start the processing the \( i \)-th group of parts; \( H_i \)
is the number of the last operation for a given group; \( G_{ij} \) is a
group of interchangeable machines that is assigned to the
operation \( O_{ij} \); \( G \) is a set of all groups of machines arose
in the matrix \( \{ \{ Z_{ij} \} \} \); \( t_{ij}(N) \) is an elementary duration
of the operation \( O_{ij} \) with one part \( d_i \) that depends on the
number of machine \( N \) in the group (on the specified operations);
\( t'_{ij} \) is the duration of set up before the operation \( O_{ij} \); \( N_{gr} \)
is the number of all groups of machines. The most widely
used objective is to find feasible schedules that minimize the
completion time of the total production program, normally re-
ferred to as makespan \( (C_{max}) \). To evaluate schedules we will
use performance measures or optimality criteria [15]:

\[
schedule length (makespan)
\]

\[
C_{max} = \max \{C_j\}, \quad (1)
\]

mean flow time

\[
F = \frac{1}{n} \sum_{j=1}^{n} F_j, \quad (2)
\]

or mean weighted flow time

\[
F = \sum_{j=1}^{n} w_j F_j / \sum_{j=1}^{n} w_j, \quad (3)
\]

max. lateness

\[
L_{max} = \max \{L_j\}, \quad (4)
\]

mean tardiness

\[
T = \frac{1}{n} \sum_{j=1}^{n} D_j, \quad (5)
\]

mean weighted tardiness

\[
T = \sum_{j=1}^{n} w_j D_j / \sum_{j=1}^{n} w_j, \quad (6)
\]

mean earliness

\[
E = \frac{1}{n} \sum_{j=1}^{n} E_j, \quad (7)
\]

mean weighted earliness

\[
E = \sum_{j=1}^{n} w_j E_j / \sum_{j=1}^{n} w_j, \quad (8)
\]

number of tardy tasks

\[
U = \sum_{j=1}^{n} U_j, \quad (9)
\]

where

\[
U_j = \begin{cases} 
1 & \text{when } \ C_j > 0 \\
0 & \text{when } \ C_j \leq 0
\end{cases}, \quad (10)
\]

or weighted number of tardy tasks

\[
U_w = \sum_{j=1}^{n} w_j U_j. \quad (11)
\]

Flow time \( F_j = C_j - r_j \); lateness \( L_j = C_j - d_j \); tardiness
\( D_j = \max \{C_j - d_j, 0\} \); earliness \( E_j = \max \{d_j - C_j, 0\} \); \( r_j \)
= arrival time; \( d_j \) – due date, \( T_j \) – task; \( w_j \) – priority, which
expresses relative urgency of \( T_j \).

### 3. Metaheuristics used for the optimization
of schedules

In the past decades, local search approaches have become in-
creasingly popular for the resolution of complex combinatorial
optimization problems. Often classified as so called meta-
heuristics [16–30] with the most prominent examples of Sim-
ulated Annealing, Tabu Search, and Evolutionary Algorithms.
These methods organize modification and improvement steps
for alternatives with the ultimate goal of identifying a global
optimal solution.

Own and implemented methods (partly modified because
of its specific character) have been used for analysis. The
results (with different criteria) for algorithms: 1. artificial
neural networks (ANN), 2. genetic algorithm (GA), 3. greedy
randomized adaptive search procedure (GRASP), 4. taboo
search (TS), 5. simulated annealing (SA) have been achieved
[31–33].

Below we present results a genetic algorithm AGHAR [31]
for finding efficient solutions to the FJSP problem. Genetic al-
gorithm, differing from conventional research techniques, start
with an initial set of random solutions (population). The chro-
osomes evolve through successive iterations (generations). The
next generation produced with the following operations: crosso-
over, mutation, and reproduction.

The choice of the method of representation of solutions for
scheduling problems is one of main stages while GA design.
The use of GA allows the choice of the certain method of
problem coding at which the specific properties of problem
solution (phenotype) are reflected in the most natural manner
using the representation (genotype).

There are different kinds of chromosome representation
for solving the FJSP. One of the approaches for representa-
tion of solutions in the problem of scheduling is the use of
permutations with repetitions. Permutations with repetitions
are connected with genes which code foreground location of
machines for separate operations. Let us consider it on an
example (Fig. 1).

![Fig. 1. Chromosome with two related lists](image)

The first list (Fig. 1) contains natural number. Before com-
putation the program attaches to every batch of parts (a pro-
duction order) one natural number. This number is repeated
in the list so many as the number of production operations
are fulfilled on this part. The length of this list is equal to the
list of sequence of operations (SEQ). The list is read from
left to right. So, the first number in the SEQ list (1 in this
case) means the 1-st part. Since this number is in the first
on the left position, it means simultaneously the 1-st operation
on this part.

The second list is a natural number which determines
a machine from the group o technologically changeable ma-
chines. We shall call it the list of location of machines
(MLST). For the 1-st operation on the 1-st part, i.e. \( O_{11}(O_{ij}

196

Bull. Pol. Ac.: Tech. 57(3) 2009
is the \( j \)-th operation of processing of the \( i \)-th part) the 1-st machine \( S_1 \) is chosen, for the next \( O_{ij} \) – second machine \( S_2 \).

The accepted in this work operator of mutation of the sequence of operations may be classified as order based mutation. Mutation of the sequence of operations consists in random replacement of two numbers in the SEQ list. On the first, on the basis of the matrix of production operations one forms a sequence of the length \( nM \), where \( n \) is a number of parts, \( M \) is a number of operations. Every number from the interval \((1,M)\) appears so many times as production operations are fulfilled on the given part. On the second stage the corresponding number from the interval \((1,N)\), where \( N \) is a number of all operations for \( n \) parts, is assigned randomly to every operation \( O_{ij} \). Mutation of machines location for the given production operation is connected with a random choice of the triplet (the place in a list, part, machine) from the MLST list and random replacement by another triplet so that the chosen machine should satisfy the processing restrictions.

Translating the concepts of a GA into a working engine involves not only designing ways to represent the basic data structures but ways of setting the principal properties or parameters of the genetic algorithm: population size, population generation, maximum number of generations, type of crossover, type of mutation, crossover rate, mutation rate.

Selecting GA parameters like mutation rate, and population size, is very difficult due the many possible variations in the algorithm and cost function. A GA relies on random number generators for creating the population, mating, and mutation. A different random number seed produces different results. In addition there are various types of crossovers and mutations, as well as other possibilities.

Figure 2 presents the best values of \( F(H) \) criterion (makespan) in relation to \( p_m \) and \( p_c \) (for 60 generations and 500 chromosomes in the population; serial – parallel route). The results of the experiments for the AGHAR algorithm are also presented in Table 1.

Genetic algorithms form a family of directed optimization and search techniques that can solve highly complex and often highly nonlinear problems. They can be used to explore very large problem spaces and find the best solution based on multiobjective functions under a collection of multiple constraints.

4. Some approaches to the evaluation of schedules

4.1. Solutions and their Pareto ranks. For a understanding of what is a domination relation, we shall work with an example [8, 34]. We consider a problem of evaluation schedules with two objectives: to maximize \( f_1 \) and to minimize \( f_2 \). For this problem, we find a set of solutions. This set of solutions is represented in a plane with \( f_1, f_2 \) as the axes (Fig. 3). We present comparisons between various solutions (e.g. method GRASP – point \( A \), ANN – point \( E \), TS – point \( B \), SA – point \( C \), GA – point \( D \)) in the Fig. 3.

The comparison between two solutions, say \( P \) and \( Q \), is represented by a pair of symbols. This pair is made up of two symbols, associated with the two objectives \( f_1 \) and \( f_2 \); each of these symbols can take three values, +, − or =, according whether \( P \) is better than, worse than or equal to \( Q \), considering the objective with which it is associated.

![Fig. 2. Example of search space with AGHAR after Ref. [31]](image)

![Table 1. Example of experiment results for the AGHAR algorithm](table)

<table>
<thead>
<tr>
<th>PM</th>
<th>PC</th>
<th>Makespan values for different samples [in minutes]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.512</td>
<td>0.128</td>
<td>37545 36724 38155 37585 35637 39430 38822</td>
</tr>
<tr>
<td>1.000</td>
<td>0.450</td>
<td>38926 38543 37084 37065 38862 38588 40597</td>
</tr>
<tr>
<td>0.192</td>
<td>0.256</td>
<td>37368 40725 41188 38902 40709 39457 38531</td>
</tr>
<tr>
<td>0.096</td>
<td>0.353</td>
<td>37229 38070 40275 37866 39706 39514 39915</td>
</tr>
<tr>
<td>0.256</td>
<td>0.450</td>
<td>40018 35124 39795 38777 37631 38515 38777</td>
</tr>
<tr>
<td>0.064</td>
<td>0.256</td>
<td>40359 38339 36397 37939 38109 38610 39853</td>
</tr>
<tr>
<td>0.128</td>
<td>0.064</td>
<td>39775 38181 40018 37210 40112 39615 39868</td>
</tr>
<tr>
<td>0.016</td>
<td>0.256</td>
<td>41088 38055 40519 40566 39857 37301 39629</td>
</tr>
<tr>
<td>0.008</td>
<td>0.128</td>
<td>42941 38274 39981 39491 40111 39908 37428</td>
</tr>
<tr>
<td>0.008</td>
<td>0.256</td>
<td>38688 39628 39560 39932 39959 38842 40078</td>
</tr>
<tr>
<td>0.032</td>
<td>0.450</td>
<td>37158 40589 40653 37440 37855 38031 39183</td>
</tr>
<tr>
<td>0.032</td>
<td>0.256</td>
<td>39961 38647 41262 40165 39269 35166 39483</td>
</tr>
<tr>
<td>0.032</td>
<td>0.128</td>
<td>40567 40193 39226 39579 40423 38843 39825</td>
</tr>
<tr>
<td>0.064</td>
<td>0.128</td>
<td>42941 38274 39981 39491 40111 39908 37428</td>
</tr>
<tr>
<td>0.064</td>
<td>0.256</td>
<td>38145 38874 36901 39209 38915 40416 39649</td>
</tr>
<tr>
<td>0.064</td>
<td>0.450</td>
<td>39013 39672 39344 40236 39826 39775 41053</td>
</tr>
<tr>
<td>0.128</td>
<td>0.353</td>
<td>38671 38903 39186 39422 37540 36506 38223</td>
</tr>
<tr>
<td>0.512</td>
<td>0.353</td>
<td>38671 38903 39186 39422 37540 36506 38223</td>
</tr>
<tr>
<td>0.192</td>
<td>0.256</td>
<td>37466 39257 38295 38501 38132 40304 39477</td>
</tr>
<tr>
<td>0.096</td>
<td>0.256</td>
<td>39444 38595 38905 38402 38774 38134 39817</td>
</tr>
</tbody>
</table>
We recall that we want to maximize $f_1$ (solution quality – that is minimize CPU Time) and minimize $f_2$ (makespan). Therefore, a point $P$ is better than a point $Q$ considering objective $f_1$ if $f_1(P)$ has a higher value than $f_1(Q)$; a point $P$ is better than a point $Q$ considering objective $f_2$ if $f_2(P)$ has a smaller value than $f_2(Q)$.

The following is an example of how to proceed with points $A$ and $B$ of Table 2. Let us perform the comparisons:

- $A$ is worse than $B$ considering objective $f_1$. Therefore, the first element of the pair in the cell $[A, B]$ is the sign $\sim$.
- $A$ is worse than $B$ considering objective $f_2$. Therefore, the second element of the pair in the cell $[A, B]$ is the sign $\sim$.

If we refer to the preceding, definition, we can conclude that solution $B$ dominates solution $A$.

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification of solutions</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
</tbody>
</table>

Obviously, pairs in the cells of Table 2 which are symmetric about the principal diagonal are “complementary”.

We shall now try to extract nondominated solutions.

1. Consider point $A$.

- This point is dominated by the following points: $B$ (pair $(-,-)$ at the intersection of row $A$ and column $B$), $C$ (pair $(-,-)$ at the intersection of row $A$ and column $C$), $D$ (pair $(-,-)$ at the intersection of row $A$ and column $D$) and $E$ (pair $(-,-)$ at the intersection of row $A$ and column $E$).

2. Consider point $B$.

- This point is dominated by the following points: $C$ (pair $(-,-)$ at the intersection of row $B$ and column $C$), $D$ (pair $(-,-)$ at the intersection of row $B$ and column $D$) and $E$ (pair $(-,-)$ at the intersection of row $B$ and column $E$).

- It dominates point $A$ (pair $(+,+)$ at the intersection of row $B$ and column $A$).

We can say that point $A$ does not belong to the set of nondominated solutions of rank 1, because it is possible to find a point ($B$ in this case) which is better than the point $A$ for all objectives.

After considering point $C$, $D$ and $E$ we can say that points $E$ and $C$ are nondominated, and $D$ is dominated. These points $E$ and $C$ dominate points $A$, $B$ and $D$ but do not dominate themselves.

Now, we take these two points ($E$ and $C$) and remove them from the table: they belong to the set of nondominated points. We can sort these solutions using the rank of domination. In our example, we attribute rank 1 (because we have finished the first comparison) to points $E$ and $C$, because they dominate all the other points but do not dominate themselves. So, these points are Pareto optimal solutions of rank 1.

We now go back to the start and apply this rule again to the remaining elements of the table. The remaining solutions after points $E$ and $C$ have been removed are represented in Table 3. This process stops when the set of points to be compared is empty. Figure 4 represents the various points and their ranks.

<table>
<thead>
<tr>
<th>Table 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification of solutions of rank 2</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

![Fig. 4. Solutions and their Pareto ranks](image)

### 4.2. The fuzzy method for evaluation of schedules

The fuzzy sets theory allows to make decisions in the so-called fuzzy environment, which is made of fuzzy objectives, fuzzy constraints and a fuzzy decision.

Let us consider a certain set of options (also referred to as choices or variants) notated using $X_{op} = \{x\}$. A fuzzy objective is defined as a fuzzy set $G$ defined in the set of options $X_{op}$. The fuzzy set $G$ is described by the membership function $\mu_G(x)$ in the set of options $X_{op}.$ The function $\mu_G(x) \in [0,1]$ for a given $x$ defines the membership degree of option $x \in X_{op}$ to the fuzzy set $G$ (fuzzy objective). A fuzzy constraint is defined as a fuzzy set $C$ also defined in the set of options $X_{op}$. The fuzzy set $C$ is described by the membership function $\mu_C(x) \in [0,1]$. The function $\mu_C(x) \in [0,1]$ for a given $x$ defines the membership degree of option $x \in X_{op}$ to the fuzzy set $C$ (fuzzy constraint). A fuzzy decision $D$ is a fuzzy set created as result of intersection of the fuzzy objective and fuzzy constraint [35]:

$$D = G_1 \cap \ldots \cap G_n \cap C_1 \cap \ldots \cap C_m,$$  \hspace{1cm} (12)

while

$$\mu_D(x) = T\{\mu_{G_1}(x), \ldots, \mu_{G_n}(x), \mu_{C_1}(x), \ldots, \mu_{C_m}(x)\}$$ \hspace{1cm} (13)

for each $x \in X_{op}$. A maximization decision is the option $x^* \in X$, such as

$$\mu_D(x^*) = \max \mu_D(x), \quad \text{for} \quad x \in X.$$ \hspace{1cm} (14)
The intersection of fuzzy sets may be more generally defined as:

\[
\mu_{A \land B}(x) = \min(\mu_A(x), \mu_B(x)) = T(\mu_A(x), \mu_B(x)),
\]

where the function \(T\) is the so-called \(n\)-norm.

We analyzed schedule on the following criteria: (a) schedule length, (b) mean flow time, (c) max. lateness, (d) mean tardiness, (e) mean weighted tardiness, (f) mean weighted earliness, (g) number of tardy tasks, and (h) weighted number of tardy tasks.

The word the best is a linguistic value, which was described separately for main (a)-(d) criteria (MC) and auxiliary (e)-(h) criteria (AC), assuming that the interval of marks is \(< 2.5 >\).

The membership functions of fuzzy sets MC and AC are the following:

\[
\mu_{MC}(x) = \begin{cases} 
0 & \text{for } 1 \leq x \leq 4.3 \\
(x - 4.3)/0.5 & \text{for } 4.3 < x \leq 4.8 \\
1 & \text{for } 4.8 < x \leq 5 
\end{cases}
\]

and

\[
\mu_{AC}(x) = \begin{cases} 
0 & \text{for } 2 < x < 4.2 \\
(x - 4.2)/0.4 & \text{for } 4.2 < x < 4.6 \\
1 & \text{for } 4.6 < x < 5 
\end{cases}
\]

The set for assessment of schedules is \(X_{op} = \{x_1, x_2, x_3, x_4, x_5, x_6\}\). By substituting the average of schedules’ marks in main criterion (a)-(d) to formula (16), we obtain membership degrees to the fuzzy set set MC. Similarly, by substituting the average of schedules’ marks in auxiliary criteria (e)-(h) to formula (17), we obtain membership degrees to the fuzzy set AC. Table 4 contains the average evaluations and (values of membership degrees) to fuzzy sets MC and AC.

The next step is to create fuzzy sets corresponding to the data included in Table 4.

“The best in schedule length” = \(G_1 = 1/x_1 + 0.2/x_2 + 1/x_3 + 0.4/x_4 + 1/x_5 + 1/x_6\)

“The best in mean flow time” = \(G_2 = 1/x_1 + 0.8/x_2 + 1/x_3 + 1/x_4 + 0.6/x_5 + 0.4/x_6\)

“The best in max. lateness” = \(G_3 = 0.8/x_1 + 1/x_2 + 0.6/x_3 + 0.4/x_4 + 0.8/x_5 + 0.8/x_6\)

“The best in mean tardiness” = \(G_4 = 0.2/x_1 + 0.2/x_2 + 1/x_3 + 1/x_4 + 0.2/x_5 + 0.2/x_6\)

“The best in mean weighted tardiness” = \(G_5 = 1/x_1 + 0.5/x_2 + 1/x_3 + 0.75/x_4 + 1/x_5 + 1/x_6\)

“The best in mean weighted earliness” = \(G_6 = 1/x_1 + 1/x_2 + 1/x_3 + 0.75/x_4 + 1/x_5 + 0.5/x_6\)

“The best in number tardy task” = \(G_7 = 1/x_1 + 0.25/x_2 + 0.5/x_3 + 1/x_4 + 1/x_5 + 0.5/x_6\)

“The best in weighted number of tardy tasks” = \(G_8 = 0.25/x_1 + 0.5/x_2 + 1/x_3 + 0.5/x_4 + 0.5/x_5 + 0.75/x_6\)

By substituting the data to formula (12), we obtain

\[
D = G_1 \cap G_2 \cap G_3 \cap G_4 \cap G_5 \cap G_6 \cap G_7 \cap G_8.
\]

The fuzzy decision of the minimum type has the form:

\[
D = 0.2/x_1 + 0.2/x_2 + 0.5/x_3 + 0.4/x_4 + 0.2/x_5 + 0.2/x_6.
\]

The schedule \(x_3\) is characterized by the greatest membership-degree and therefore he will be accepted.

4.3. Decision making using analytic hierarchy process (AHP). AHP is designed for situations in which evaluations are quantified to provide a numeric scale for prioritizing the alternatives [36].

The multiobjective decision making method enables one to take into account many different criteria which were used for the result evaluation, obtained with a number of methods in a number of experiments.

The structure of the decision problem is summarized in Fig. 5. The problem involves two hierarchies with five criteria and five decision alternatives (methods – GRASP, ANN, TS, SA, and GA).

The problem requires two hierarchies with five criteria and five decision alternatives (methods – GRASP, ANN, TS, SA, and GA). 

![Fig. 5. Structure of decision process for the problem](image-url)
Table 5
Values of criteria optimization (for Simulated Annealing)

<table>
<thead>
<tr>
<th>Criteria of optimization</th>
<th>Values of criteria</th>
<th>Number of job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan</td>
<td>50242.2</td>
<td></td>
</tr>
<tr>
<td>$w_j$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean $F$</td>
<td>39459.6</td>
<td></td>
</tr>
<tr>
<td>$w_j^*F_j$</td>
<td>61463.7</td>
<td>108592.2</td>
</tr>
<tr>
<td>Mean weighted $F$</td>
<td>38617.9</td>
<td></td>
</tr>
<tr>
<td>$L_{\text{max}} = \max {L_j}$</td>
<td>487.9</td>
<td></td>
</tr>
<tr>
<td>Mean $D$</td>
<td>149.1</td>
<td></td>
</tr>
<tr>
<td>$w_j^*D_j$</td>
<td>1463.7</td>
<td>592.2</td>
</tr>
<tr>
<td>Mean weighted $D$</td>
<td>156.6</td>
<td></td>
</tr>
<tr>
<td>Mean $E$</td>
<td>189.5</td>
<td></td>
</tr>
<tr>
<td>$w_j^*E_j$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Mean weighted $E$</td>
<td>204.3</td>
<td></td>
</tr>
<tr>
<td>$U$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$w_j^*U_j$</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

The first hierarchy represents several experiments (weights these experiments are equal $a_1=a_2=\ldots=a_z$). The second hierarchy demonstrates criteria: $C$ – schedule length (makespan), $F$ – mean flow time, $L$ – max. lateness, $D$ – mean tardiness, and $E$ – mean earliness.

Values of criteria optimization for SA and other algorithms are described in Table 5 and Figs. 6–7.

These criteria represent different weights (from $a_1$ to $a_5$) because some of them are more important than others, e.g. the most important is the makespan being 3 times as important as the mean flow time.

For any schedule generated by each method, values of the above mentioned schedule evaluation criteria are calculated. The alternatives have weights from $a_{11}$ to $a_{55}$. The weights of individual hierarchies must add up to 1, e.g.

$$a_1 + a_2 + a_3 + a_4 + a_5 = 1,$$

$$a_{11} + a_{12} + a_{13} + a_{14} + a_{15} = 1.$$ 

The crux of AHP is determination of the relative weights to rank the decision alternatives. Assuming that we are dealing with $n$ criteria at a given hierarchy, the procedure establishes an $n \times n$ pairwise comparison matrix, $A$, that quantifies the decision maker’s judgment regarding the relative importance of the different criteria. The pairwise comparison is made such that the criterion in row $i$ ($i=1, 2, \ldots, n$) is ranked relative to every other criterion. Letting $a_{ij}$ define the element $(i, j)$ of $A$, AHP uses a discrete scale from 1 to 9 in which $a_{ij}=1$ signifies that $i$ and $j$ are of equal importance, $a_{ij}=5$ indicates that $i$ is strongly more important than $j$, and $a_{ij}=9$ indicates that $i$ is extremely more important than $j$. Other intermediate values between 1 and 9 are interpreted correspondingly. Consistency in judgement requires that $a_{ij}=k$ automatically implies that $a_{ji}=1/k$. Also, all the diagonal elements $a_{ii}$ of $A$ must equal 1, because they rank a criterion against itself.

Stages of AHP method are as follows [37]:

1. Determination of the comparison matrix (for the criteria). To show how the comparison matrix is determined for this decision problem, we start with the hierarchy dealing with criteria (Fig. 5). In expert’s judgment, is that $C$ – makespan, is strongly more important than $L$ – max. lateness, and hence $a_{13}=4$, and this assignment automatically implies that $a_{31}=1/4 = 0.25$. 

![Fig. 6. Values of criteria optimization for other algorithms](image)

![Fig. 7. Values other of criteria optimization for algorithms](image)
2. The relative weights of criteria \( C, F, L, D \) and \( E \) can be determined from \( A \) by normalizing it into a new matrix \( N \). The process requires dividing the elements of each column by the sum of the elements of the same column. In our example the sum of elements for each column of matrix \( A \) is equal: 2.33; 7.00; 9.33; 4.67 and 9.33. The process requires dividing the elements of column 1 by 2.33, and those of column 2 by 7.00 etc. and normalized matrix is determined.

3. The relative weights \( w \) of the criteria are then computed as the row average

\[
(w_C, w_F, w_L, w_D, w_E) =
(0.429; 0.143; 0.107; 0.214; 0.107).
\]

4. Determinate of the comparison matrix (for alternatives). At this stage when ranking an alternative (of the method) in relation to any other one we will use the values of the results obtained by different algorithms. For instance we will consider the values of mean flow time criterion – \( F \) for ANN (\( F = 39156.70 \)) and for SA (\( F = 39459.57 \)). It can be seen that (flow time for SA consist 0.992 flow time value for ANN (39156.70/39459.57 = 0.992) and this value will be written into the created comparison matrix \( A_F \) as \( a_{42} \). The inverse value \( a_{24} = 1.008 \) will be obtained by calculating 1/0.992. Values for different algorithms shows Table 6.

### Table 6

<table>
<thead>
<tr>
<th></th>
<th>GRASP</th>
<th>ANN</th>
<th>TS</th>
<th>SA</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C_{\max} = \max{C_j} )</td>
<td>50242.2</td>
<td>50242.2</td>
<td>50242.2</td>
<td>50242.2</td>
<td>50242.2</td>
</tr>
<tr>
<td>Mean ( F )</td>
<td>33601.7</td>
<td>39156.7</td>
<td>36514.8</td>
<td>39459.6</td>
<td>33734.4</td>
</tr>
<tr>
<td>( L_{\max} = \max{L_j} )</td>
<td>2918.6</td>
<td>15433.8</td>
<td>4848.8</td>
<td>487.9</td>
<td>11790.5</td>
</tr>
<tr>
<td>Mean ( D )</td>
<td>500.1</td>
<td>2639.9</td>
<td>1126.9</td>
<td>149.1</td>
<td>1922.2</td>
</tr>
<tr>
<td>Mean ( E )</td>
<td>6398.4</td>
<td>2983.2</td>
<td>5112.1</td>
<td>189.5</td>
<td>7687.8</td>
</tr>
</tbody>
</table>

Out of the comparison matrices \( A, F, L, D \) and \( E \), matrix \( A_F \) is shown:

\[
A_F = \begin{pmatrix}
GRASP & 1 & 1.165 & 1.087 & 1.174 & 1.004 \\
ANN & 0.858 & 1 & 0.933 & 1.008 & 0.862 \\
TS & 0.920 & 1.072 & 1 & 1.081 & 0.924 \\
SA & 0.852 & 0.992 & 0.925 & 1 & 0.855 \\
GA & 0.996 & 1.161 & 1.082 & 1.170 & 1
\end{pmatrix}
\]

5. From matrix \( A \) we determine the matrix of the normalized values \( N \) for the alternatives. It will be done similar to stage 2, i.e. by dividing each element of matrix \( A \) by the sum of the elements in the same column. Using the values of sums of individual matrices, we divide for example the elements of the first column in matrix \( A_C \) by 5, of the second column by 5, and so on, obtaining a matrix of normalized values – matrix \( N_C \). In a similar way we obtain the other \( N \) matrices. One of the normalized matrices is shown below:

\[
N_F = \begin{pmatrix}
GRASP & 0.216 & 0.216 & 0.216 & 0.216 \\
ANN & 0.186 & 0.186 & 0.186 & 0.186 \\
TS & 0.199 & 0.199 & 0.199 & 0.199 \\
SA & 0.184 & 0.184 & 0.184 & 0.184 \\
GA & 0.215 & 0.215 & 0.215 & 0.215
\end{pmatrix}
\]

6. At stage 6 (just as at stage 3) we calculate the relative weights \( w \) for the alternatives as the average for each row of the normalized values matrix:

\[
(w_{F(\text{GRASP})}, w_{F(\text{ANN})}, w_{F(TS)}, w_{F(SA)}, w_{F(GA)}) = \frac{(0.216; 0.186; 0.199; 0.184; 0.215)}{5}
\]

As an example we calculated one of the weights of matrix \( N_F \) for the fourth alternative. It is \( w_F = 0.184 \).

7. Finally we classify each method (algorithm), which is show in the calculations below as well as in Fig. 8.
5. Schedule representation in search space

5.1. Generation of job sequences using the function \( q(j) \).

Some methods used to generation of the permutations are shown in [38].

In the partially ordered set of all the permutations there exist two “extreme” permutations, one of which precedes all of them and the second one comes after all of them. They are, respectively, the identical permutation \( J^T = \{ 1, 2, ..., n \} \) and the one that is inverse (opposite) with respect to it, \( J^{-T} = \{ n, n-1, ..., 1 \} \). Because of the partial ordering of \( \Sigma \) we cannot control the permutation choice in a determined manner varying \( Q(j) \). However, some “partial controllability” is possible; in this case:

a) the lower is progression ratio \( Q \), the higher is the probability of choice of the permutation \( J^T \),

b) when \( Q = 1 \), all the subintervals are of equal lengths and \( \sigma_j^R \) will be yielded as “absolutely random”,

c) the greater is \( Q \), the higher is the probability to obtain the inverse permutations \( J^{-T} \).

In the first two algorithms in the matrix of route \( ||O_{ij}|| \) whose rows are the “generalized” operations three “supporting” operations are selected: first, intermediate and final \((j = 1, j = j_{av}, j = M)\), where \( j_{av} = (M + 1)/2 \) under which the sequence of the triples \( \{ Q_0(1), Q_1(j_{av}), Q_1(M) \} \) is specified in definite way, where \( l = 1 \) \( \text{max} \).

In algorithms 1 and 2 for the intermediate operations \((1 < j < j_{av} \text{ and } j_{av} < j < M)\) the function \( Q(j) \) can be calculated by the linear or exponential interpolation.

Now let us speak about the differences between these algorithms. The first algorithm chooses \( Q(j) \) where \( j \in \{ 1, j_{av}, M \} \) is some discretization of segment \([q_{min}, q_{max}]\), where \( q_{min} = \ln Q_{min}, q_{max} = \ln Q_{max} \). In this case the algorithm performs the complete sorting-out of all the triples \( q \in \Xi \times \Xi \times \Xi \).

In order to make it possible to take the discretization \( \Xi \times \Xi \times \Xi \) as the uniform one we shall work not with the value \( Q \) but with the value \( q = \ln Q \). The linear interpolation for \( q(j) \) corresponds to the exponential one used when constructing \( Q(j) \) (Fig. 9).

![Fig. 9. Example function \( q(j) \)](image)

Algorithm 1 possesses some “redundancy”. For example, the function \( q_j(j) \) (broken lines) that are located near \( q(j) \) \( \sim 0 \) can generated schedules with similar characteristics, because the permutations during every operation are obtained as “purely random”.

We proposes the modification for algorithm 1 when for three base operations \( j = 1, j_{av} = (M + 1)/2, j = M \) the limits \( Q_{min} \) and \( Q_{max} \) are chosen as the different ones, i.e. the “framing hexagon” is obtained instead of the rectangle as in algorithm 1.

Algorithm 2 possesses the lower “redundancy”. In this algorithm the correct specification of the lower an the upper limits of changes of \( q \) is also important. Also, in this algorithm the functions \( q(j) \) are generated each of which completely fills the segment \([q_{min}, q_{max}]\) with its values; here \( q_{min} = \ln Q_{min}, q_{max} = \ln Q_{max} \). It is assumed that the correct choice of the segment \([q_{min}, q_{max}]\) should be made in the algorithm teaching process.

Algorithm 3, as opposed to the previous ones, constructs the parabolic function \( q(j) \). It is not less time-saving when compared with algorithm 2 but essentially differs from it by the way of parametrization and parameter \( q \) variation. Thus, we look for the parabol a each of which fills the interval \([1, M]\) with its values on the segment \([q_{min}, q_{max}]\). Since the rectangle bounded by the lines \( q = q_{max}, q = q_{min}, j = 1, j = M \) has two axes of symmetry, then proceeding from each constructed parabola we can receive three more parabolas by way of symmetric transformation:

a) with respect to the axis \( q = 0 \),
b) with respect to the axis \( j = (M + 1)/2 \),
c) with respect to both axes simultaneously.

Assume that it is required to determine the best sequence of jobs for the performance based on an accepted test the choice. Let a proposition (statement) of the form “a structure of sequence of jobs \( q(1), q(j_{av}), q(M) \) is the best for the accepted test \( F \)” be given. Depending on the control parameter \( q \), the corresponding fuzzy truth values can be assigned to this proposition.

Consider the definition of the standard fuzzy truth values. According to [39], a fuzzy truth value \( \tau \) is defined as a fuzzy set of membership functions \( \mu_\tau : [0, 1] \rightarrow [0, 1] \). Moreover, the truth value is assumed to be a linguistic variable whose set of terms \( T(\tau) \) is the enumerated set of the form \( T(\tau) = \{ “true”, “not true”, “very false”, “to some extent true”, “true by all likelihood”, “not very true”, “not very false”, “false”, ..., \} \). Each element of this set is identified with a certain fuzzy set in the subset of truth values, i.e., within \([0, 1]\). If a defined proposition with a fuzzy truth value has properly the form \( u = R \), where \( R \) is a certain fuzzy set, \( R \subseteq X \), and its fuzzy truth value is equal to \( \tau \), \( \tau \subseteq [0, 1] \), then we can write \( (u = R) = \tau \).

Consider the Baldwin approach [39] to the definition of the standard fuzzy truth values. In the approach, membership functions of the main fuzzy truth values (such as “true”, “not true”, “very true”, “certainly true”, etc.) are defined as some functions \([0, 1] \rightarrow [0, 1]\); moreover, this author uses the
following arguments. If we have the proposition \((u = R) = \text{“true”}\), then it should be “logically” equivalent to the proposition \(u = R\). Then the fuzzy truth value that corresponds to the term “true” should be defined using the membership function: \(\mu\) is “true” \((v) = v\) and \(\mu\) is “not true” \((v) = 1 - v\), for all \(v \in [0, 1]\). Other more complicated forms of fuzzy truth values are defined as follows (in a plane with \(x, y\), as the axes, where \(x = v, y = \mu(v)\): 

- \(\mu\) is “very true” \((v) = (\mu \text{ “true”} (v))^2 = v^2\), for all \(v \in [0, 1]\);
- \(\mu\) is “very not true” \((v) = (\mu \text{ “not true”} (v))^2 = (1 - v)^2\), for all \(v \in [0, 1]\);
- \(\mu\) is “sufficiently true” \((v) = (\mu \text{ “true”} (v)^{1/2} = \sqrt{v}\), for all \(v \in [0, 1]\);
- \(\mu\) is “sufficiently not true” \((v) = (\mu \text{ “not true”} (v))^{1/2} = \sqrt{1 - v}\), for all \(v \in [0, 1]\);
- \(\mu\) is “absolutely true” \((v) = 1, v = 1\) and \((v) = 0, v \in [0, 1]\);
- \(\mu\) is “absolutely not true” \((v) = 1, v = 0\) and \((v) = 0, v \in [0, 1]\);
- \(\mu\) is “indefinite” \((v) = 1, \forall v \in [0, 1]\).

Variants of the job sequence structure in the function \(q(j)\) and fuzzy verity values for the proposition \(\phi\) shown in Fig. 10.

Let us introduce the definition of the membership function of a fuzzy set \(\mu_q\) of the type “ordering” of jobs from the sequence \(J^T\) to the inverse one \(J^{-T}\) whose linguistic variables have the form (“according to the given permutation”, “very close to the given permutation”, “close to the given permutation”, “far from the given permutation”, “close to the reverse permutation”, “very close to the reverse permutation”, “the permutation that is reverse to the given one”), where \(X = \{(1, 2, 3, ... n - 1, n), (n, n - 1, ... 3, 2, 1)\}\).

If in the process of choice of the sequence jobs (for a definite structure of real data such as the performance time of operation, the number of machines, etc.), we assign the values equal to the values in Fig. 10, that is 1. \(q_{\min}\), \(\forall j \in [1, M]\); 2. from \(q_{\min}\) to \(q_{av}\), \(\forall j \in [1, M/2]\); 3. \(q_{av}\), \(\forall j \in [1, M/2]\); 4. from \(q_{av}\) to \(q_{\max}\), \(\forall j \in [1, M/2]\); 5. from \(q_{av}\) to \(q_{\min}\), \(\forall j \in [M/2, M]\); 6. \(q_{av}\), \(\forall j \in [M/2, M]\); 7. \(q_{av}\) to \(q_{\max}\), \(\forall j \in [M/2, M]\); 8. \(q_{\max}\), \(\forall j \in [1, M]\). Then the fuzzy values of truthness for the proposition \(\phi\) “a structure of sequence of jobs that is the best for process production with the minimal total time” take the values (depending on the structure of the data) corresponding linguistic values from the set of terms: very true, true, ambiguous, sufficiently true, not very true, false, very false; in our case, they are respectively 2 and 5 for very true, 2 and 5 for false, 2 and 5 also for true, 3 and 5 for sufficiently true, 3 and 7 also for true, 4 and 5 for not very true, 4 and 7 for false, 8 for very false.

5.2. Schedule cluster recognition for job shop problem and creating the rules. Inverse problem to generation permutations is representation of results simulation process. We wish to determine if a cluster of events has occurred. By cluster, we mean that more occurrences of an event are observed than would normally be expected [40–42].

To construct a grid we use following approach. Denote by \(J^T\) and \(J^{-T}\) the identical and inverse permutations. We introduce the area, where \(y = q(j) \) – permutations, and \(x = j\) – process operation. We connect three points corresponding to \(q_{\min}\) and \(q_{\max}\) on the all operations by a curve (it is sequence). Sequences are generated by the optimization algorithm (for example simulated annealing, taboo method, and other matalheuristic) many times (e.g. 1000) in the clustering process. Structures for some of sequences shown in Fig. 11.

![Fig. 10. Variants of the structure of sequence of jobs in the function \(q(j)\) and fuzzy verity values of verity for the proposition \(\phi\)](image)

![Fig. 11. Example some of schedules](image)
An example can be an area which is divided into a grid of $15 \times 50 = 750$ cells as shown in Fig. 11.

Figure 11 shows two sets of two-dimensional points. In this example (one schedule consists of 15 points) three schedules occur $15^3 \times 3/750 = 0.06$ of the time 6%. One might be inclined to call this shaded area a cluster. But how probable is this cluster? And how can we make a decision to either accept the hypothesis that this area is a cluster or to reject it?

When we say that we select $n$ permutations at random, we mean that each of the $n!$ permutations of length $n$ are chosen with probability $1/n!$. Probability of success (that is, the event that $p$ has property $A$) is defined as the number of favorable outcomes of our random choice divided by the number of all outcomes.

To arrive at a decision we use a Bayesian approach [43]. It computes the odds ratio against the occurrence of a cluster (or in favor of no cluster), which is defined as odds $= P \{ \text{no cluster} \} / P \{ \text{cluster} \}$. Probability of success is $P \{ \text{hit} \} = 0.06$. One might be tempted to call it a Bernoulli array but the determination of the probabilities will of course proceed as usual. If $M$ cells are contained in the supposed cluster area (shown as shaded in Fig. 12 with $M = 200$ i.e. three grey area), then the probability of $k$ hits is given by the binomial law:

$$ P[k] = \binom{M}{k} p^k (1 - p)^{M-k}. \quad (19) $$

Next must assign values to $p$ under the hypothesis of a cluster present and no cluster present. From Fig. 11 in which we did not suspect a cluster, the relative frequency of hits was about 0.147 so that assume $p_{nc} = 0.147$ when there is no cluster. When we believe a cluster is present, we assume that $p_c \sim 0.3$ in accordance with the relative frequency of hits in the shaded area of Fig. 12, which is 50/200 = 0.295. Thus,

$$ P[A|B^c] = P\{ \text{observed data|no cluster} \} = \binom{M}{k} \; p_{nc}^k (1 - p_{nc})^{M-k} = \left( \frac{200}{59} \right) 0.147^{59} (0.853)^{141}, $$

$$ P[A|B] = P\{ \text{observed data|cluster} \} = \binom{M}{k} \; p_c^k (1 - p_c)^{M-k} = \left( \frac{200}{59} \right) (0.295)^{59} (0.705)^{141}, $$

$$ \text{odds} = \frac{P[A|B^c] \; P[B^c]}{P[A|B] \; P[B]} = \frac{0.147^{59} (0.853)^{141} (1 - 10^{-6})}{(0.295)^{59} (0.705)^{141} (10^{-6})} = 0.66, $$

which results odds $< 1$.

Since the posterior probability of no cluster is 0.66 times larger than the posterior probability of a cluster, we would accepted the hypothesis of a cluster present.

Figure 12 show the points plotted in the the $jq$-plane. When we consider the $q$ axis, we see four sets of points. One is from the circle and triangle (white points) that do not form a set in the full space, one consists of the square and triangle (black points) for average $j$, and two consists of the circle and triangle (black points) and square (white points) for first and last $j$.

Clustering is often performed as a preliminary step in a data mining process, with the resulting clusters being used as further inputs into a different technique (e.g. neural networks). Data mining may be defined as the discovery of unexpected
relationships by analyzing such large volumes of data that automated processes are necessary. The extracted knowledge is expressed as a model or a pattern sets of rules or clusters for instance.

Numerous algorithm have been proposed for rule induction from data in the machine learning literature. One technique for generating a set of individually interesting and useful rules is to build a classification tree and to evaluate each of the branches as individual rules according to specific targeted quality criteria. But some rules produced by standard classification make no sense to the user since they use biases and specific heuristic to generate the classifier. Thus this kind of technique may lose interesting rules. On the other hand, association rules are among the most popular representation for local patterns in data mining. In these rules the target is not predefined and the right-hand side of such a rule may be a conjunction of attribute-value terms.

There are a wide variety of methods for converting clusters into rules. Many of these approaches attempt to generate rules from fuzzy clusters through a reductionist approach that treats the clusters and the data cloud around them as binary classification points. Much of the underlying algorithmic work is concerned with inducing a membership framework from the cluster centers so that the control rules can be induced. These approaches generally ignore the more straightforward use of approximation hedges to convert cluster centroids into fuzzy numbers with finely tuned expectancy (width) values. By using hedges we can treat one dimension of the data space as an outcome and the remaining dimensions as rule predicates.

Cluster centers as fuzzy numbers is relatively easy the approach (when we treat the center of a cluster as the center of a bell-shaped fuzzy set). The closer a point is to the center of the cluster the higher its membership in the center’s fuzzy set. In this approach, the actual degree of membership in the clusters is concerned with inducing a membership framework from the cluster centers so that the control rules can be induced. These rules may lose interesting rules. On the other hand, association rules are among the most popular representation for local patterns in data mining. In these rules the target is not predefined and the right-hand side of such a rule may be a conjunction of attribute-value terms.

If the data N dimensions and there are K clusters, we can induce K rules with N-1 predicates that is, each cluster forms a fuzzy rule in the data classification space. To illustrate, let \( q(1), q_{av}, \) and \( q(M) \) be data vectors. Let \( C1(q(1)i, q_{av1}, q(M)i) \) and \( C2(q(1)i, q_{av2}, q(M)i) \) be the centers (centroids) of the clustering of \( q(1), q_{av}, \) and \( q(M) \). From the clustering we can induce the following rules:

- If \( q(1) \) is about \( C1(q(1)) \) and \( q_{av} \) is about \( C1(q_{av}) \), then \( q(M) \) is near \( C1(q(M)) \).
- If \( q(1) \) is about \( C2(q(1)) \) and \( q_{av} \) is about \( C2(q_{av}) \), then \( q(M) \) is near \( C2(q(M)) \).

The expectancy of the above hedge reflects the compactness of the cluster. Compact clusters have larger (wider) expectancies, whereas less compact clusters have larger (wider) expectancies.

5.3. Evaluation of scheduling processes in the objective space. One other difference between single-objective and multi-objective (MO) optimization is that in multi-objective optimization (MOO) the objective functions constitute a multi-dimensional space, in addition to the usual decision variable space common to all optimization problems. This additional space is called the objective space, \( Z \). For each solution \( x \) in the decision variable space, there exists a point in objective space, denoted by \( f(x) = (y_1, y_2, ..., y_n)^T \). The mapping takes place between an \( n \)-dimensional solution vector and an \( R \)-dimensional objective vector.

Fig. 13. The search space (decision variable space) and evaluation of schedules in objective space

Let \( D \) be the set of sequences, where each sequence \( q \) in \( D \) represents a set of sequences \( q(j) \) contained in \( I \), where \( j \in 1, M \). Suppose that we have a particular set of sequences \( q(j) \) (e.g., \( q(1) = q_{min}, q(2) = q_{av}, \) and \( q(3) = q_{max} \)) and another set of items \( B \) (e.g., \( q(1) = q_{av}, q(2) = q_{max} \), and \( q(3) = q_{min} \)). Then an association rule takes the form if \( A \), then \( B \) (i.e., \( A \Rightarrow B \)), where the antecedent \( A \) and the consequent \( B \) are proper subsets of \( I \), and \( A \) and \( B \) are mutually exclusive [44, 45].
In our case we have:

IF \((q_1, k_1; \ldots; q_{av}, k_{av}; \ldots; q_M, k_M)\), THEN \((y_1, y_2, \ldots, y_R)\)

or IF algorithm \(a\), THEN \((y_{1a}, y_{2a}, \ldots, y_{Ra})\)

... IF algorithm \(n\) THEN \((y_{1n}, y_{2n}, \ldots, y_{Rn})\).

If standard algorithms are involved, the rule selection according to multiple criteria is made as postprocessing the set of rules extracted first by the algorithm. This approach may result in undiscovered interesting rules. Few non-standard approaches have been proposed in order to apply a multiple criteria selection. In the context which we consider, i.e. the data mining and particularally the extraction of rules, few works related to metaheuristics in muti-objective optimization exist.

For a given data structure \(\{t_{ij}, t'_{ij}, m, Z_i\}\) we can specify trajectory carried out by the schedule algorithm (i.e. for each \(j\) operation specify \(q, n/m, k\) values). Similarly, as we can generate with a relatively high probability of a particular sequence of parts (as shown above), it is also inversely with specific sequence parts can most likely determine its degree of belonging to the scope \([T_0, T_k]\).

At the each period time (operation \(j\)) \(q\) and \(n/m\) control variables (decision variables) determinate sure the state of the production system (scope \([T_0, T_k]\)). For the transition production system by the state in point \(T_0\) to the state in point \(T_k\) should be determined the transition trajectory (Fig. 14).

The result of the process control is achieved by the production system a particular goal referred to the final point \(T_k\) of the desired trajectory (in our case, the characteristics of time).

Although the the search process of an algorithm takes place on the decision variables space, many algorithms, particular multi-objective evolutionary algorithms (MOEAs), use the objective space information in their search operators. However, the presence of two different spaces introduces a number of interesting flexibilities in designing a search algorithm for MOO.

5.4. The adaptive schedule parameter optimization and adaptive search space. A schedule relies on the accuracy of its constraints and its parameters. Many of the parameters in schedule are often difficult to predict and establish with a high degree of certainty. The relative precision of parameter values affects not only the outcome of the schedule but the way in which optimization is achieved. The are several critical factors (or parameters) that are especially important in evolving a workable and effective job schedule. These include the following: job duration times, machine efficiencies, available machine times, number machine and job prioritization.

One of approach to the adaptive schedule parameter optimization is described in [44] and can be used in our case. This approach include the following (Fig. 15).
The machine learning facilities are the core of parameter optimization. The first component of this system is based on statistical learning theory and extracts, from the historical database and from the differences between the planned and actual schedules (the value for each parameter over time). The second component of the machine learning system is a very advanced and very deep pattern recognition processor.

Using a form of data mining, this process extracts behavior rules in the form of fuzzy logic from the historical database. These rules are stored in a knowledge base and used to make fine-grain classification and categorization decisions.

One of the requirements in evolutionary optimization (EO) is that the boundary of all feasible regions in the parameter domain must be predefined a priori to an evolution process. Genetic operations such as crossover and mutation can be viewed as the basis of inductive learning in a human brain, where the induction and recombination of knowledge take place [46]. If the candidate solutions are regarded as knowledge stored in a human brain, the evolution toward the global optimum is similar to the route of the investigation process, while the parameter search space in EO can be viewed as the experimental region interest in human brain. The region of interest may change through the deductive learning process, and the search space in EO can be dynamic and learned in a deductive manner where analysis and reasoning take place. In the general structure of inductive-deductive learning for EO, at each generation, the genetic evolution performs an inductive learning where knowledge in the form of candidate solutions is induced through the genetic operations for previous solutions. Then statistical information is acquired from the distribution of induced candidates and applied to the deductive learning process. In deductive learning, analysis and reasoning are performed based on heuristic rules to determine the next search space of interest in the parameter domain for inductive learning. Since fitness information of the evolved candidates is not required in the updating rule, the adaptive search space approach can be directly applied to most evolutionary algorithm, for both single-objective and multiobjective optimization problems.

6. Conclusions

In this paper we have outlined the domination method, fuzzy method and AHP methodology used in supporting the decision maker in solving multi-objective problem. The basic metaheuristics applied for schedule optimization have been described. Schedule evaluation, using various optimality criteria has been presented. We demonstrated interdependencies between several objectives (makespan, mean flow time, mean weighted flow time, max. lateness, mean tardiness etc).

An approach for presenting the search space and schedule evaluation has been proposed. The three-dimensional space can be used for the analysis and control of the production processes. We can identify the areas most relevant to performing production tasks of a certain data structure. By choosing the appropriate control value we can pursue the production process in accordance with the required trajectory describing the process characteristics (such as time etc.). Many real-world phenomena cannot be modeled by one single model but require a set of complementary models that together are able to describe the whole process. In a multimodel, depending on the state the process is in, one out of the sets of models will be applicable and will describe the production process appropriately. Solutions of the scheduling problem will probably be more perfect along with the development of such sciences as data mining, pattern recognition, theory of modeling and simulation, and many others.

REFERENCES


J. Han and M. Kamber, Data Mining, Elsevier, New Jersey, 2006.


