BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES
Vol. 57, No. 3, 2009
DOI: 10.2478/v10175-010-0121-4

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

T. WITKOWSKI [1]*, P. ANTCZAK [2], and A. ANTCZAK [2]

[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 weighted tardiness, mean earliness, mean weighted 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 mataheuristics used for optimization schedules and the approaches that use domination method, fuzzy method, and analytic hierarchy proccess (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 performancr 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 achive 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 catogorized 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-multi 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 decison 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 onlyon one predetermined machine. Though the JSP has benn well studied, its aplication to real-word 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 the JSP by allowing each operations to be processed on more than machine. With this extension, we are now confronted with two subtask: assignment of each operation to an appriopriate 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 Π_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 $\{O_{ij}\}_{j=\xi}^{H_i}$. Then for Z_i, we can write $Z_i = (\Pi_i\{O_{ij}\}_{j=\xi}^{H_i})$, where $O_{ij} = (G_{ij}, t_{ij(N)})$ is the j-th operation of processing the i-th group of parts; ξ_i is the number of operation of the production process at which one should start the processing the i-th group of parts; H_i

*e-mail: tawit@poczta.onet.pl

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 $||\{Z_i\}||$; $t_{ij}(N)$ 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 referred to as makespan ($C_{\max}$). To evaluate schedules we will use performance measures or optimality criteria [15]:
schedule length (makespan)

$$C_{\max} = \max\{C_j\}, \tag{1}$$

mean flow time

$$\overline{F} = \frac{1}{n}\sum_{j=1}^{n} F_j, \tag{2}$$

or mean weighted flow time

$$\overline{F} = \sum_{j=1}^{n} w_j F_j / \sum_{j=1}^{n} w_j, \tag{3}$$

max. lateness

$$L_{\max} = \max\{L_j\}, \tag{4}$$

mean tardiness

$$\overline{D} = \frac{1}{n}\sum_{j=1}^{n} D_j, \tag{5}$$

mean weighted tardiness

$$\overline{D} = \sum_{j=1}^{n} w_j D_j / \sum_{j=1}^{n} w_j, \tag{6}$$

mean earliness

$$\overline{E} = \frac{1}{n}\sum_{j=1}^{n} E_j, \tag{7}$$

mean weighted earliness

$$\overline{E} = \sum_{j=1}^{n} w_j E_j / \sum_{j=1}^{n} w_j, \tag{8}$$

number of tardy tasks

$$U = \sum_{j=1}^{n} U_j, \tag{9}$$

$$\text{where} \quad U_j = \begin{cases} 1 & \text{when} \quad C_j > 0 \\ 0 & \text{when} \quad C_j \leq 0 \end{cases}, \tag{10}$$

or weighted number of tardy tasks

$$U_w = \sum_{j=1}^{n} w_j U_j. \tag{11}$$

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 increasingly popular for the resolution of complex combinatorial optimization problems. Often classified as so called mataheuristics [16–30] with the most prominent examples of Simulated 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 algorithm, differing from conventional research techniques, start with an inital set of random solutions (population). The chromosomes evolve through succesive iterations (generations). The next generation produced with the following operations: crossover, mutation, and reproduction.

The choice of the method of repesentation 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 representation 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 seperate operations. Let us consider it on an example (Fig. 1).

1	2	1	3	2	1	2	2	1	Number of part
7	2	6	4	1	3	8	4	3	Number of machine

Fig. 1. Chromosome with two related lists

The first list (Fig. 1) contains natural number. Before computation the program attaches to every batch of parts (a production 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 lengh 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 machines. We shall call it the list of location of machines (MLST). For the 1-st operation on the 1-st part, i.e. $O_{11}(O_{ij}$

is the j-th operation of processing of the i-th part) the 1-st machine S_7 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 desiging 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.

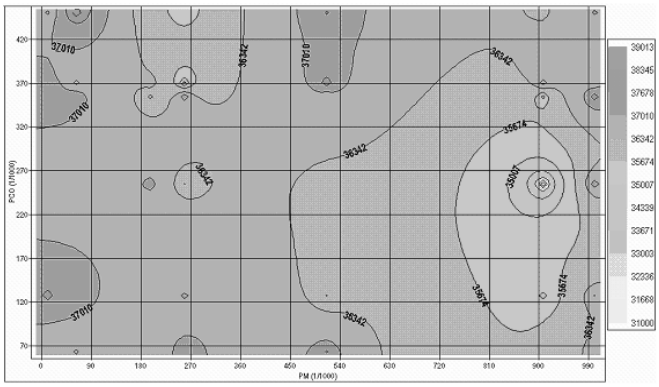

Fig. 2. Example of search space with AGHAR after Ref. [31]

Table 1
Example of experiment results for the AGHAR algorithm

PM	PC	Makespan values for different samples [in minutes]						
		1	2	3	4	5	6	7
0.512	0.128	37545	36724	38155	37585	35637	39430	38822
1.000	0.450	38926	38543	37084	37065	38862	38588	40597
0.192	0.256	37368	40725	41188	38902	40709	39457	38531
0.096	0.353	37729	38707	40275	37866	39706	39514	39915
0.256	0.450	40018	35124	39795	38777	37631	38515	38777
0.064	0.256	40359	38339	36397	37939	38109	38610	39853
0.128	0.064	39775	38181	40018	37210	40112	39615	38968
0.016	0.256	41088	38055	40519	40566	39857	37301	39629
0.008	0.128	40200	41239	39281	40422	39025	40047	41556
0.008	0.256	40942	38210	38390	40132	39879	39890	35470
0.008	0.450	36868	39628	39560	39932	39959	38842	40078
0.032	0.450	37158	40589	40653	37440	37855	38031	39183
0.032	0.256	39961	38647	41262	40165	39269	35166	39483
0.032	0.128	40567	40193	39226	39579	40423	38843	39825
0.064	0.128	42941	38274	39981	39491	40111	39908	37428
0.064	0.256	38145	38874	36901	39209	38915	40416	39649
0.064	0.450	39013	39672	39344	40236	39826	39775	41053
0.128	0.353	38671	38903	39186	39422	37540	36506	38223
0.512	0.353	40968	39619	40180	39583	38870	40559	36429
0.192	0.256	37466	39257	38295	38501	38132	40304	39477
0.096	0.256	39444	38595	38905	38402	38774	38134	39817

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.

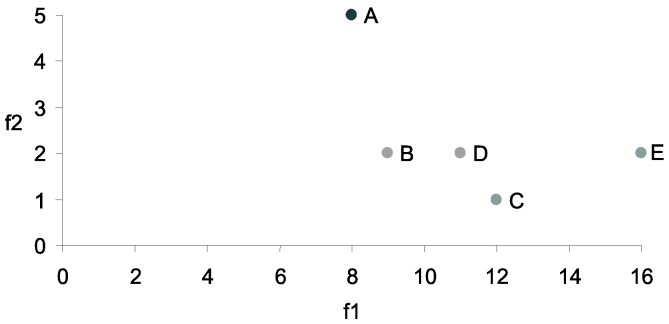

Fig. 3. Representation of the solutions in the plane f_1, f_2

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.

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

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

Table 2
Classification of solutions

	A	B	C	D	E
A		(−,−)	(−,−)	(−,−)	(−,−)
B	(+,+)		(−,−)	(−,=)	(−,=)
C	(+,+)	(+,+)		(+,+)	(−,+)
D	(+,+)	(+,=)	(−,−)		(−,=)
E	(+,+)	(+,=)	(+,−)	(+,=)	

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 3
Classification of solutions od rank 2

	A	B	D
A		(−, −)	(−, −)
B	(+, +)		(−, =)
D	(+, +)	(+, =)	

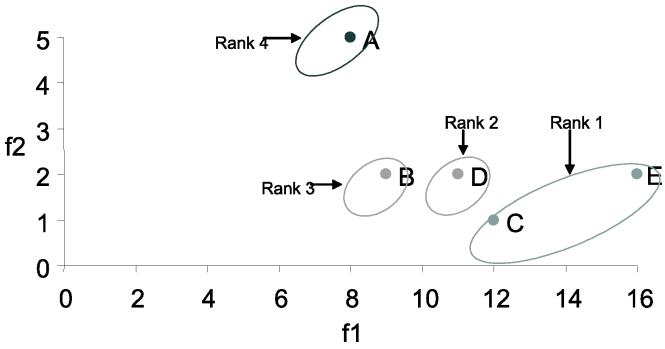

Fig. 4. Solutions and their Pareto ranks

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 μ_G: $X_{op} \to [0, 1]$. 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 μ_C: $X_{op} \to [0, 1]$. 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]:

$$D = G_1 \cap ... \cap G_n \cap C_1 \cap ... \cap C_m, \tag{12}$$

while

$$\mu_D(x) = T\{\mu_{G1}(x), ..., \mu_{Gn}(x), \mu_{C1}(x), ..., \mu_{Cm}(x)\} \tag{13}$$

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

$$\mu_D(x^*) = \max \mu_D(x), \qquad \text{for} \qquad x \in X. \tag{14}$$

Table 4
Average evaluations for fuzzy sets MC and AC (and their membership degree values in parantheses)

Schedule	Criterion							
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
x_1	4.8(1)	5.0(1)	4.7(0.8)	4.4(0.2)	4.7(1)	5.0(1)	4.7(1)	4.3(0.25)
x_2	4.4(0.2)	4.7(0.8)	4.8(1)	4.4(0.2)	4.4(0.5)	4.7(1)	4.3(0.25)	4.4(0.5)
x_3	4.9(1)	4.9(1)	4.6(0.6)	4.9(1)	4.9(1)	4.9(1)	4.4(0.5)	4.7(1)
x_4	4.5(0.4)	4.8(1)	4.5(0.4)	5.0(1)	4.5(0.75)	4.5(0.75)	4.7(1)	4.4(0.5)
x_5	5.0(1)	4.6(0.6)	4.7(0.8)	4.4(0.2)	5.0(1)	4.7(1)	4.7(1)	4.4(0.5)
x_6	4.9(1)	4.5(0.4)	4.7(0.8)	4.4(0.2)	4.9(1)	4.4(0.5)	4.4(0.5)	4.5(0.75)

The intersection of fuzzy sets may be more generalny defined as:

$$\mu_{A\ldots B}(x) = \min(\mu_A(x), \mu_B(x)) = T(\mu_A(x), \mu_B(x)), \quad (15)$$

where the function T is the so-called t-norm.

We analized 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:

$$\mu_{\mathrm{MC}}(x) = \begin{cases} 0 & \text{for } 1 \le x \le 4.3 \\ (x - 4.3)/0.5 & \text{for } 4.3 < x \le 4.8 \\ 1 & \text{for } 4.8 < x \le 5 \end{cases} \quad (16)$$

and

$$\mu_{\mathrm{AC}}(x) = \begin{cases} 0 & \text{for } 2 < x < 4.2 \\ (x - 4.2)/0.4 & \text{for } 4.2 < x < 4.6 \\ 1 & \text{for } 4.6 < x < 5 \end{cases} \quad (17)$$

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" = $G_1 = 1/x_1 + \mathbf{0.2}/x_2 + 1/x_3 + \mathbf{0.4}/x_4 + 1/x_5 + 1/x_6$

"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" = $G_4 = \mathbf{0.2}/x_1 + \mathbf{0.2}/x_2 + 1/x_3 + 1/x_4 + \mathbf{0.2}/x_5 + \mathbf{0.2}/x_6$

"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" = $G_7 = 1/x_1 + 0.25/x_2 + \mathbf{0.5}/x_3 + 1/x_4 + 1/x_5 + 0.5/x_6$

"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:

$$D = 0.2/x_1 + 0.2/x_2 + \mathbf{0.5}/x_3 + 0.4/x_4 + 0.2/x_5 + 0.2/x_6.$$

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).

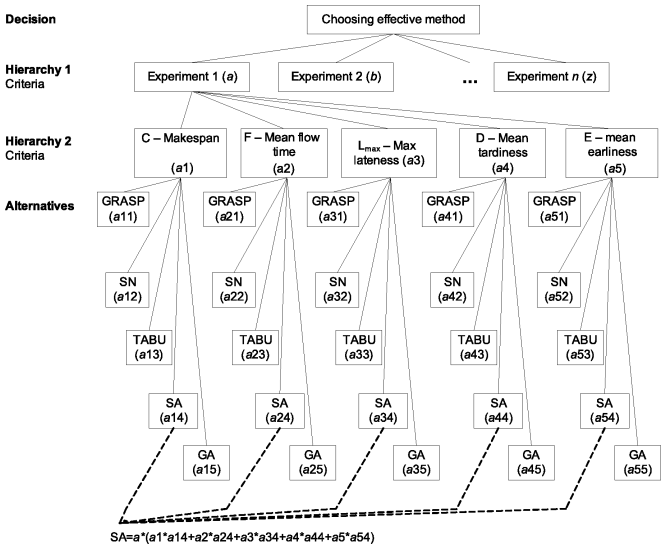

Fig. 5. Structure of decision process for the problem

Table 5
Values of criteria optimization (for Simulated Annealing)

Criteria of optimization	Values of criteria	Number of job									
		1	2	3	4	5	6	7	8	9	10
Makespan	50242.2										
w_j		3	3	3	3	2	2	2	1	1	1
Mean F	39459.6										
$w_j^* F_j$		61463.7	108592.2	135174.0	160727.0	77987.2	70579.2	16252.8	33742.6	49423.4	47034.5
Mean weighted F	38617.9										
$L_{\max} = \max\{L_j\}$	487.9										
Mean D	149.1										
$w_j^* D_j$		1463.7	592.2	174.3	0.0	0.0	579.2	0.0	0.0Y	423.4	34.5
Mean weighted D	156.6										
Mean E	189.5										
$w_j^* E_j$		0.0	0.0	0.0	2273.4	12.8	0.0	1747.2	257.4	0.0	0.0
Mean weighted E	204.3										
U	6										
$w_j^* U_j$		3	3	3	0	0	2	0	0	1	1

The first hierarchy represents several experiments (weights these experiments are equal a=b=...=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.

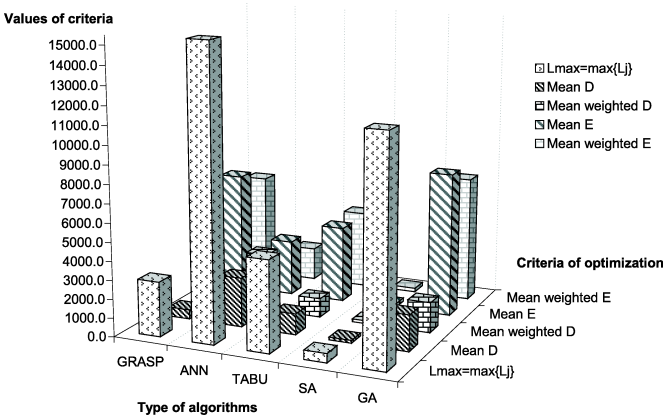

Fig. 6. Values of criteria optimization for other algorithms

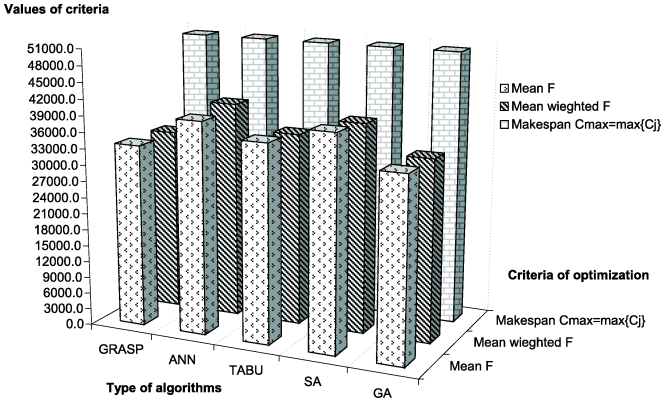

Fig. 7. Values other of criteria optimization for algorithms

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 $i (i = 1, 2, ..., n)$ 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 $a_{31} = 1/4 = 0.25$.

$$A = \begin{array}{c} \\ C \\ F \\ L \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} C & F & L & D & E \end{array} \\ \begin{pmatrix} 1 & 3.000 & 4.000 & 2.000 & 4.000 \\ 0.333 & 1 & 1.333 & 0.667 & 1.333 \\ 0.250 & 0.750 & 1 & 0.500 & 1.000 \\ 0.500 & 1.500 & 2.000 & 1 & 2.000 \\ 0.250 & 0.750 & 1.000 & 0.500 & 1 \end{pmatrix} \end{array}.$$

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.

$$N = \begin{array}{c} \\ C \\ F \\ L \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} C & F & L & L & E \end{array} \\ \begin{pmatrix} 0.429 & 0.429 & 0.429 & 0.429 & 0.429 \\ 0.143 & 0.143 & 0.143 & 0.143 & 0.143 \\ 0.107 & 0.107 & 0.107 & 0.107 & 0.107 \\ 0.214 & 0.214 & 0.214 & 0.214 & 0.214 \\ 0.107 & 0.107 & 0.107 & 0.107 & 0.107 \end{pmatrix} \end{array}.$$

The columns of N are identical, a characteristic that occurs only when the decision maker exhibits perfect consistency in specifying the entries of the comparison matrix A.

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
Values for different algorithms

	GRASP	ANN	TS	SA	GA
Makespan					
$C_{\max} = \max\{C_j\}$	50242.2	50242.2	50242.2	50242.2	50242.2
Mean F	33601.7	39156.7	36514.8	39459.6	33734.4
$L_{\max} = \max\{L_j\}$	2918.6	15433.8	4848.8	487.9	11790.5
Mean D	500.1	2639.9	1126.9	149.1	1922.2
Mean E	6398.4	2983.2	4112.1	189.5	7687.8

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

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

5. From matrix A we determine the matrice 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:

$$N_F = \begin{array}{c} \\ GRASP \\ ANN \\ TS \\ SA \\ GA \end{array} \begin{array}{c} \begin{array}{ccccc} GRASP & ANN & TS & SA & GA \end{array} \\ \begin{pmatrix} 0.216 & 0.216 & 0.216 & 0.216 & 0.216 \\ 0.186 & 0.186 & 0.186 & 0.186 & 0.186 \\ 0.199 & 0.199 & 0.199 & 0.199 & 0.199 \\ 0.184 & 0.184 & 0.184 & 0.184 & 0.184 \\ 0.215 & 0.215 & 0.215 & 0.215 & 0.215 \end{pmatrix} \end{array}.$$

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:

$$(w_{F(GRASP)},\ w_{F(ANN)},\ w_{F(TS)},\ w_{F(SA)},\ w_{F(GA)}) = \\ = (0,216;\ 0,186;\ 0,199;\ 0,184;\ 0,215).$$

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

7. Finaly we classify each method (algorithm), which is show in the calculations below as well as in Fig. 8.

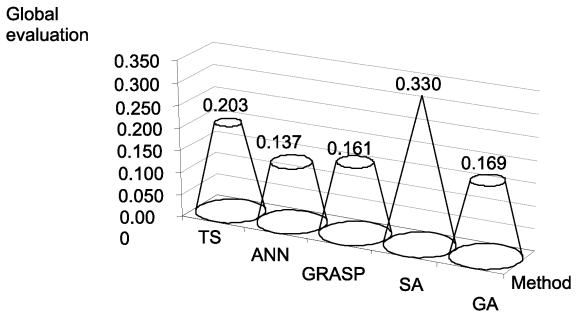

Fig. 8. Comparison results of algorithms

$$GRASP = 1^{*}(0.429^{*}0.200 + 0.143^{*}0.216 + 0.107^{*}0.125 + \\ + 0.214^{*}0.191 + 0.107^{*}0.299) = \mathbf{0.203}$$

$$SN = 1^{*}(0.429^{*}0.200 + 0.143^{*}0.186 + 0.107^{*}0.024 + \\ + 0.214^{*}0.036 + 0.107^{*}0.140) = \mathbf{0.137}$$

$$TABU = 1^{*}(0.429^{*}0.200 + 0.143^{*}0.199 + 0.107^{*}0.075 + \\ + 0.214^{*}0.085 + 0.107^{*}0.192) = \mathbf{0.161}$$

$$SA = 1^{*}(0.429^{*}0.200 + 0.143^{*}0.184 + 0.107^{*}0.746 + \\ + 0.214^{*}0.639 + 0.107^{*}0.009) = \mathbf{0.330}$$

$$WG = 1^{*}(0.429^{*}0.200 + 0.143^{*}0.215 + 0.107^{*}0.031 + \\ +0.214^{*}0.050 + 0.107^{*}0.360) = \mathbf{0.169}$$

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 showed 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 $J^T \triangleq \{1, 2, ..., n\}$ and the one that is inverse (opposite) with respect to it, $J^{-T} \triangleq \{n, n-1, ..., 1\}$. Because of the partial ordering of Σn 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 σ_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 $\{O_l(1),\ Q_l(j_c),\ Q_l(M)\} \triangleq \{Q_l\}$, is specified in definite way, where $l = \overline{1, l_{\max}}$.

In algorithms 1 and 2 for the intermediate operations ($1 < j < j_c$ 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).

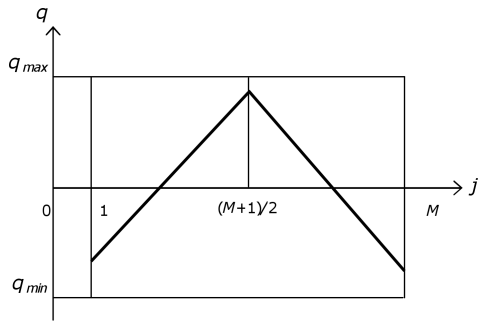

Fig. 9. Example function $q(j)$

Algorithm 1 possesses some "redundancy". For example, the function $q(j)$ (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 choosen 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 τ is defined as a fuzzy set of membership functions μ_τ: $[0,1] \rightarrow [0,1]$. Moreover, the truth value is assumed to be a linguistic variable whose set of terms $T(\tau)$ is an 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 τ, $\tau \subseteq$ [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 $(u = R)$ = "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: μ is "true" $(v) = v$ and μ is "not true" $(v) = 1 - v$, $\forall v \in [0, 1]$. Other more complicated forms of fuzzy truth values are defined as follows (in a plane with x, y, as the axes, where $x = v$, $y = \mu(v)$):
μ is "very true" $(v) = (\mu$ "true" $(v))^2 = v^2$, $\forall v \in [0, 1]$;
μ is "very not true" $(v) = (\mu$ "not true" $(v))^2 = (1 - v)^2$, $\forall v \in [0, 1]$;
μ is "sufficiently true" $(v) = (\mu$ is "true" $(v)^{1/2} = \sqrt{v}$, $\forall v \in [0, 1]$;
μ is "sufficiently not true" $(v) = (\mu$ is "not true" $(v))^{1/2} = \sqrt{1 - v}$, $\forall v \in [0, 1]$;
μ is "absolutely true" $(v) = 1$, $v = 1$ and $(v) = 0$, $\forall v \in [0, 1]$;
μ is "absolutely not true" $(v) = 1$, $v = 0$ and $(v) = 0$, $\forall v \in [0, 1]$;
μ 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 ϕ shown in Fig. 10.

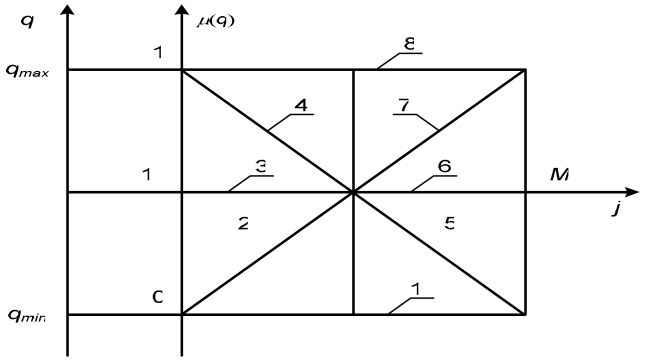

Fig. 10. Variants of the structure of sequence of jobs in the function $q(j)$ and fuzzy values of verity for the proposition φ

Let us introduce the definition of the membership function of a fuzzy set μ_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 $q_{\max}$ to q_{av}, $\forall_j \in [1, M/2]$; 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 ϕ "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 7 for true, 3 and 6 also for true, 1, 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 mataheuristic) many times (e.g. 1000) in the clustering process. Structures for some of sequences shown in Fig. 11.

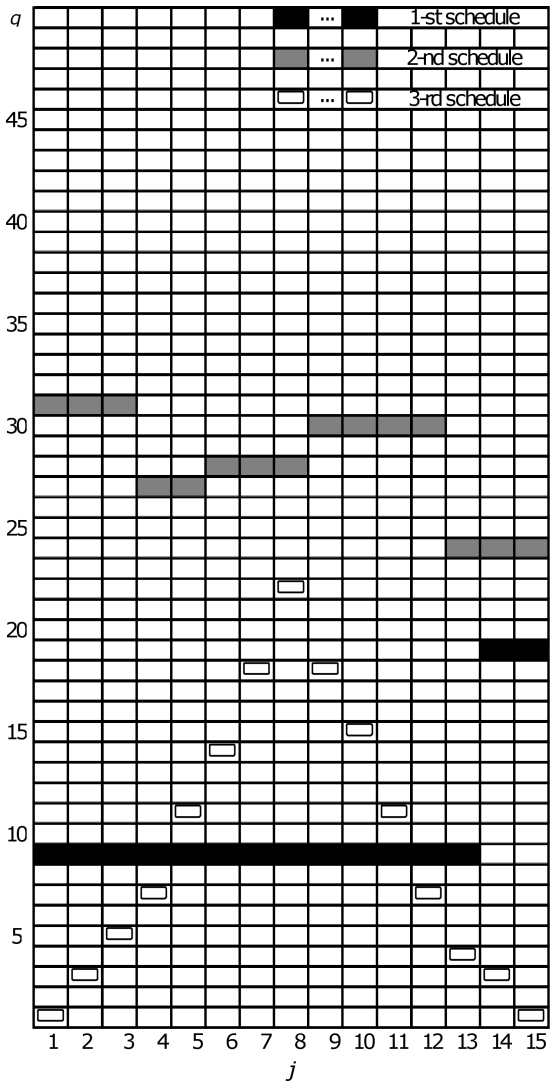

Fig. 11. Example some of schedules

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 $15^*3/750 = 0.06$ 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 lenght n are choosen with probability $1/n!$. Probability of success (that is, the event that p has property A) is definied 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 $= P$ [no cluster | observed data] / P [cluster | observed data].

If this number is large, typically much greater than one, we would be inclined to reject the hypothesis of a cluster, and otherwise, to accept it. We can use Bayes' theorem to evaluate the odds ratio by letting $B = \{\text{cluster}\}$ and $A = \{\text{observed data}\}$. Then,

$$odds = \frac{P[B^c|A]}{P[B|A]} = \frac{P[A|B^c]P[B^c]}{P[A|B]P[B]}. \tag{18}$$

To evaluate this we need to determine $P[B]$, $P[A|B^c]$, $P[A|B]$. The first probability $P[B]$ is the prior probability of a cluster. Since we belive a cluster is quite unlikely, we assign a probability of 10^{-6} to this. Next we need $P[A|B^c]$ or the probability of the observed data if there is no cluster. Since each cell can take on only one of two values, either a hit or no hit, and if we assume that the outcomes of the various cells are independent of each other, we can model the data as a Bernoulli sequence. For this problem, we 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:

$$P[k] = \begin{pmatrix} M \\ k \end{pmatrix} p^k (1-p)^{M-k}. \tag{19}$$

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 belive 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 $59/200 = 0.295$. Thus,

$$P[A|B^C] = P[\text{observed data}|\text{no cluster}] =$$
$$= \begin{pmatrix} M \\ k \end{pmatrix} p_{nc}^k (1-p_{nc})^{M-k} = \begin{pmatrix} 200 \\ 59 \end{pmatrix} 0.147^{59}(0.853)^{141},$$

$$P[A|B] = P[\text{observed data}|\text{cluster}] =$$
$$= \begin{pmatrix} M \\ k \end{pmatrix} p_c^k (1-p_c)^{M-k} = \begin{pmatrix} 200 \\ 59 \end{pmatrix} (0.295)^{59}(0.705)^{141},$$

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

which results odds < 1.

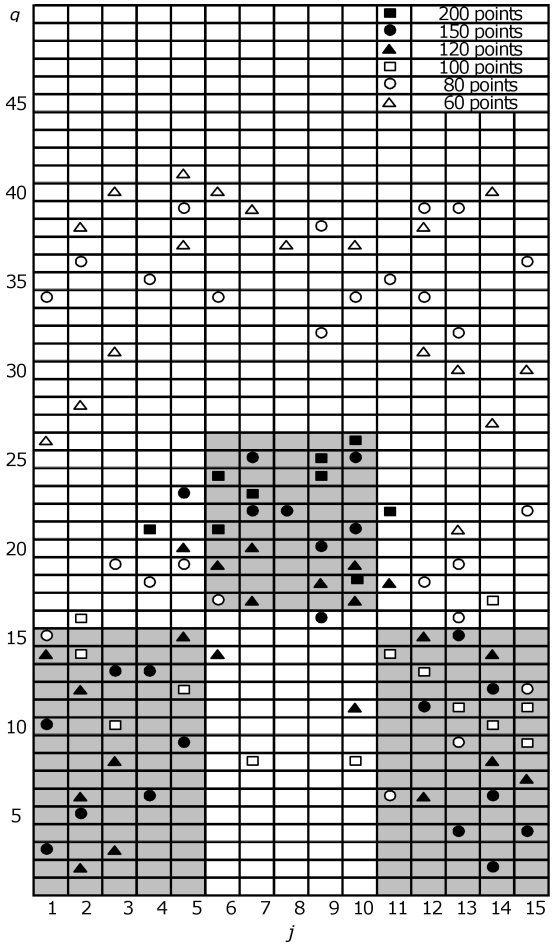

Fig. 12. View in the jq-plane

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 circe 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 loose interesting rules. On the other hand, association rules are among the most popular representation for local paterns in data mining. In these rules the target is not predifined and the right-hand side of such a rule may be a conjuction 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 reletively 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 cluster, computed by the clustering algorithm, is essentially lost.

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, $q_{av}i$, $q(M)i$) and $C2$ (q (1)i, $q_{av}i$, $q(M)i$) be the centers (centrids) 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 about hedge reflects the compactness of the cluster. Compact clusters have larger (wider) expactencies, 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 $f(x) = (y_1,\ y_2,\ ...,\ y_R)^T$. The mapping takes place between an n-dimensional solution vector and an R-dimensional objective vector.

The search space is shown in a system of coordinates (j, k, q) too, where j – the number of the operation, $k = n/m$ – the coefficient of the operation/machine type (n – number of parts, m – number of machines), q – sequence of parts in j-operation [$q^T = \{1, ..., n\}$; $q^{-T} = \{n, ..., 1\}$] (Fig. 13).

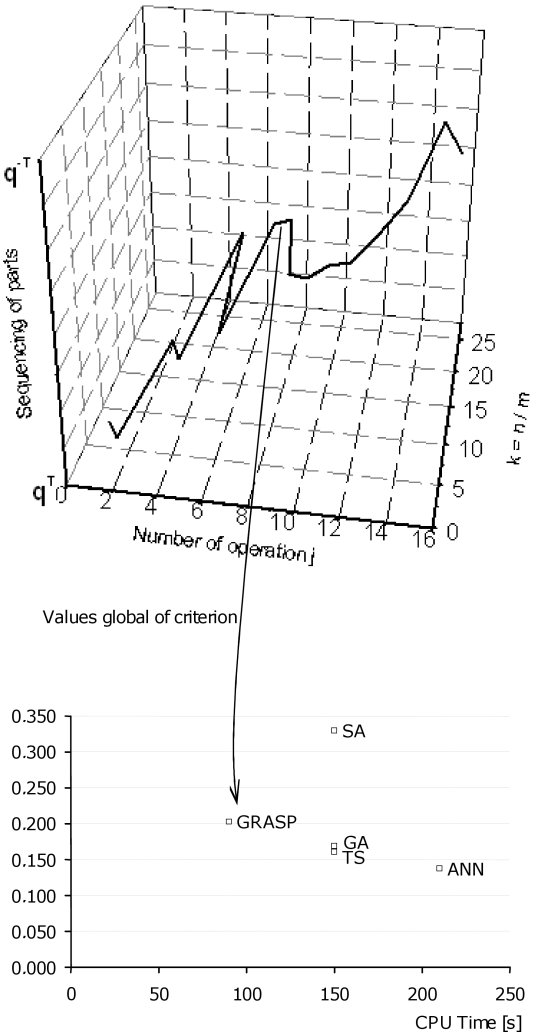

Fig. 13. The search space (decision variable space) and evaluation of schedules in objective space

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].

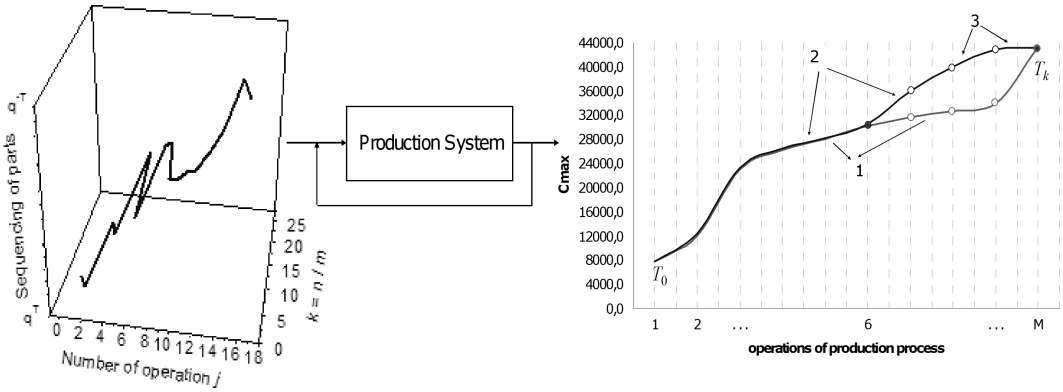

Fig. 14. A production system experiment (1. trajectory from schedule algorithm, 2. trajectory from system, 3. control trajectory)

In our case we have:
IF (q_1, k_1; ...; q_{av}, k_{av}; ...; q_M, k_M), THEN (y_1, y_2, ..., y_R)
or IF algorithm a, THEN (y_{1a}, y_{2a}, ..., y_{Ra})
...
IF algorithm n THEN (y_{1n}, y_{2n}, ..., 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 unidiscovered 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 particulalarly 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 (ie. for each j operation specify q, n/m 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 $[q^T, q^{-T}]$.

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, particulary 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 approache to the adaptive schedule parameter optimization is discribed in [44] and can be used in our case. This approache include the following (Fig. 15).

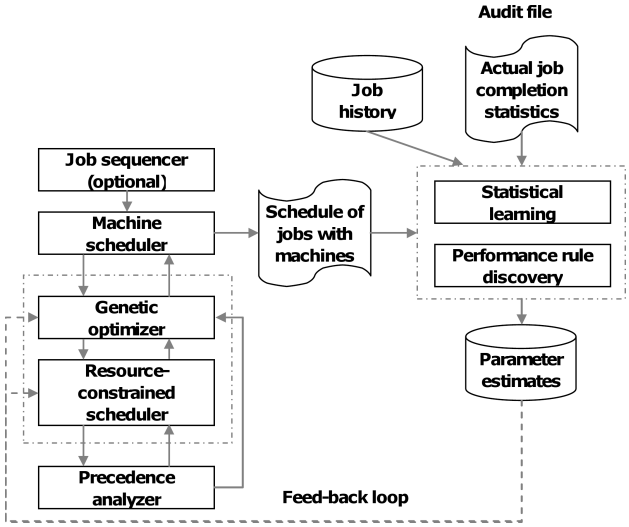

Fig. 15. The adaptive schedule parameter optimization (on base after Ref. [44])

- The job sequencer (this facility is used to order jobs that have predecosser relationships);
- The machine scheduler (manages all constraint conditions and objective functions for the current schedule process);
- The genetic optimizer (solves the underlying multiconstraint, multiobjective schedule through the parallel exploration and convergence through a large n-dimensional space of candidate schedules – the genetic optimizer explores this space by creating a random of machine-feasible and time-feasible schedules);
- The resource-constrained scheduler (generates a complete feasible schedule and takes a candidate-time and machine-feasible and turns it into a working schedule by attempting to start each job at its earliest possible time);

- The precedence analyzer (manages the topological relationships among the various jobs);
- The auditing facility (allows the analyst to expose a considerable amount of the fine-grain details in all scheduling components);
- The current schedule (is generated from the mixture of machines and jobs based on the set of constraint parameters, such as estimated job duration times, machine avalability);
- The actual job completion statistics (in most cases the acquisition of job completion statistics is fairly straightforward and is derived form the field service report associated with the job);
- The job history repository (provides the foundation for learning about the past and predicting the future – each current schedule is transformated into a set of descriptive statistics that is used by the learning machanism);
- Parameter estimates (this is the result of the statistical learning and rule extraction – a collection of estimates for each of the underlying parameters associated with the schedule. The parameter estimates are designed to coincide with the actual schedule parameters);
- The feedback loop (the current generation of parameter estimates is fed back into the genetic scheduler and the resource constraint scheduler to produce the next-generation schedule. The actual completion statistics for this schedule are then used to compute or refine the parameters, thus completing the feedback loop).

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 reqiurements 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 o knowlegde 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 learnig. 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 mataheuristics 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

[1] R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley, New York, 1976.

[2] P. Dasgupta and P.P. Chakrabarti, *Multiobjective Heuristic Search*, Vieweg&Soon, Braunschweig/ Wiesbaden, 1999.

[3] J. Gupta, Jc. Ho, and S. Webster, "Bi criteria optimization of the makespan and mean flow time on two identical parallel machine", *J. Operational Research Society* 51 (11), 1330–1339 (2000).

[4] C.J. Lio, W.C. Yu, and C.B. Joe, "Bi-criteria scheduling in two machine flow shop", *Int. J. Production Research* 53 (9), 1004–1015 (1997).

[5] A. Singh, N.K. Mehta, and P.K. Jain, "A multicriterion approach for dynamic scheduling", *15 th Int. DAAAM Symposium*, 419–420 (2004).

[6] H. Liu, A. Abraham, O. Choi, and S.H. Moon, "Variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems", *LNCS* 4247, 197–204 (2006).

[7] P. Brandimarte, "Routing and scheduling in a flexible job shop by taboo search", *Annals of Operations Research* 41 (3), 157–183 (1993).

[8] K. Deb, "Multi-objective optimization", in: *Search Methodologies*, eds. E.K. Burke and G. Kendall, pp. 273–316, Springer, London, 2005.

[9] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems", *IEEE Trans. on Systems, Man, Cybernetics* 1, 1–13 (2002).

[10] M. Mastrolilli and L.M. Gambardella, "Effective neighborhood functions for the flexible job shop problem", *J. Scheduling* 3 (1), 3–20 (2000).

[11] K.S.N. Ripon, "Hybrid evolutionary approach for multi-objective job-shop scheduling problem", *Malaysian J. Computer Science* 20 (2), 183–198 (2007).

[12] W.J. Xia and Z.M. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems", *Computers and Industrial Engineering* 48, 409–425 (2005).

[13] L-N. Xing, Y-W. Chen, and K-W. Yang, "Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling", *Applied Soft Computing*, 362–379 (2009).

[14] Y.J. Xing, Z.Q. Wang, J. Sun, and J.J. Meng, "A multi-objective fuzzy genetic algorithm for job-shop scheduling problem", *J. Achievements in Materials and Manufacturing Engineering* 17 (1–2), 297–300 (2006).

[15] J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt, and J. Węglarz, *Handbook Scheduling*, Springer, Berlin, 2007.

[16] E. Alba, *Parallel Metaheuristics*, Wiley, New Jersey, 2005.

[17] M.E. Aydin and T.C. Fogarty, "A simulated annealing algorithm for multi-agents systems: A job shop scheduling", *J. Intelligent Manufacturing* 15 (6), 805–814 (2004).

[18] E.K. Burke and G. Kendall, *Search Methodologies*, Springer, New York, 2005.

[19] T.A. Feo and M.G.C. Resende, "Greedy randomized adaptive search procedures", *J. Global Optimization* 6, 109–133 (1994).

[20] J. Gao, M. Gen, and L. Sun, "A hybrid of genetic algorithm and bottleneck shifting for flexible job shop scheduling problem", *Proc. GECCO*' 06, 1157–1164 (2006).

[21] F. Glover, "Tabu search – Part I", *ORSA J. Computing* 1 (3), 190–206 (1989).

[22] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning,* Addison-Wesley, New York, 1989.

[23] T.F. Gonzales (ed.), *Handbook of Approximation Algorithms and Metaheuristics*, Chapman and Hall/CRC, New York, 2007.

[24] E. Hart and P. Ross, "The evolution and analysis of a potentional antibody library for job shop scheduling", in: *New Ideas in Optimisation*, eds. D. Corne, M. Dorigo and F. Glover, pp. 185–202, McGraw-Hill, 1999.

[25] A. Kusiak and M. Chen, "Expert systems for planning and scheduling manufacturing systems", *Eur. J. Operational Research* 34, 113–130 (1988).

[26] Z.X. Ong, J.C. Tay, and C.K. Kwoh, "Applying the clonal selection principle to find flexible job shop schedules", *LNCS* 3627, 442–455 (2005).

[27] Z. Othman, K. Subari, and N. Morad, "Application of fuzzy inference systems and genetic algorithms in integrated process planning and scheduling", *Int. J. Computer, Internet and Management* 10 (2), 81–96 (2002).

[28] C.C. Ribeiro and P. Hansen, *Essays an Surveys in Metaheuristics*, Kluwer Academic Publishers, London, 2002.

[29] K.C. Tan, E.F. Khor, and T.H. Lee, *Multiobjective Evolutionary Algorithms and Applications*, Springer-Verlag, London, 2005.

[30] S. Yang and S.D. Wang, "Constraint satisfaction adaptive neural network and heuristics combined approaches for generalized job-shop scheduling", *IEEE Trans. Neural Networks* 11 (2), 474–486 (2000).

[31] T. Witkowski, A. Antczak, and P. Antczak, "Random and evolution algorithms of tasks scheduling and the production scheduling", *Proc. Int. Joint Conf. on Fuzzy Systems – FUZZ – IEEE* 2, 727–732 (2004).

[32] T. Witkowski, P. Antczak, and A. Antczak, "Tabu search and GRASP used in hybrid procedure for optimize the flexible job shop problem", *Proc. 11th Int. Fuzzy Systems Association – World Congress: Fuzzy Logic, Soft Computing and Computional Intelligence, IFSA* III, 1620–1625 (2005).

[33] T. Witkowski, P. Antczak, and A. Antczak, "The application of simulated annealing procedure for the flexible job shop scheduling problem", *Proc. 11th Int. Conf. Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU*, 21–26 (2006).

[34] Y. Collette and P. Siarry, *Multiobjective Optimization. Principles and Case Studies*, Springer-Verlag, Berlin, 2004.

[35] L. Rutkowski, *Computational Intelligence. Methods and Techiques*, Springer-Verlag, Berlin, 2005.

[36] T.L. Saaty, *Fundamentals of Decision Making*, RWS Publications, Pittsburgh, 1994.

[37] H.A. Taha, *Operations Research. An Introduction*, Pearson Prentice Hall, New Jersey, 2007.

[38] T. Witkowski, A. Antczak, and P. Antczak, "Schedule cluster recognition with use conditional probability", *Proc. IEEE Int. Workshop on Intelligent Data Acquistion and Advanced Computing Systems: Technology and Applications*, 413–418 (2007).

[39] G.J. Klir and T.A. Folger, *Fuzzy Sets, Uncertainty, and Information*, Prentice Hall, New Jersey, 1988.

[40] J. Han and M. Kamber, *Data Mining*, Elsevier, New Jersey, 2006.

[41] P.N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Education, New York, 2006.

[42] S. Theodoridis and K. Koutroumbas, *Patern Recognition*, Academic Press, San Diego, 2006.

[43] S. Kay, *Intuitive Probability and Random Processes Using MATLAB*, Springer, New York, 2006.

[44] E. Cox, *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, Morgan Kaufman, New York, 2005.

[45] D.T. Larose, *Discovering Knowledge in Data*, Wiley-Interscience, New Jersey, 2005.

[46] E.F. Khor, K.C. Tan, and T.H. Lee, "Tabu-based exploratory evolutionary algorithm for effective multiobjective optimization", *First Conf. on Evolutionary Multi-Criterion Optimization*, 344–358 (2001).