Fault Diagnosis of Manufacturing Processes via Genetic Algorithm Approach

Stefania GALLOVA

***Abstract*—Instantaneous detection and diagnosis of various faults and break-downs in industrial processes is required to reduce production losses and damage to equipments. A solved knowledge-based system and developed techniques within artificial intelligence principles are showing its capability in assisting process operators to detect and diagnose faults and anomalies in industrial processes, especially in chosen machine tools environment. In this contribution, there is investigated a self-learning of diagnostic rules through genetic algorithms approach. A success of a genetic algorithm is crucially connected with the adaptive plan which controls „mutation“ and „crossover“ actions, and hence the production of the new solutions. The maximum entropy principle with minimum cross-entropy updating, provides a way of making assumptions about the missing specification that minimises the additional information assumed. Solved genetic algorithm is also a stochastic computational model that seeks the optimal solution to an objective function.**

***Index Terms*—genetic algorithm, self-learning, probability, rule**

I. Introduction

A solved knowledge-based diagnostic system involves the measurements of machine tool vibrations to detect faults. When combining and propagating uncertain information, each inference systems must, at least by implication, make certain assumptions about correlations not explicitly specified. The maximum entropy principle with minimum cross-entropy updating, provides a way of making assumptions about the missing specification that minimises the additional information assumed, and thus offers a standard against which the other uncertain inference systems can be compared. From this point of view was solved, compared and designed an appropriate uncertain inference system for created diagnostic expert system for machine tools.

Manuscript received March 22, 2007. This work was supported by the Slovak Republic's Ministry of Education under Scientific Grant Projects:

1. *Diagnostic System Building of Machinery Equipment with Knowledge-based System and Chaos Theory Applying* (VEGA 1/1084/04).
2. *Intelligent Approach to Automated Diagnostic Problem Solving of Machinery Equipment* (Number: VEGA 1/4133/07).

Stefania GALLOVA. Author is with the Technical University of Kosice, Letna 9, SK-042 00 Kosice, Slovak Republic; phone: +421-904-584-426; e-mail: stefania.gallova@zoznam.sk

Both the inductively derived rules and the expert-derived rules were tested using the same testing events. The experiment involved the application of several genetic algorithm approaches techniques. There is investigated a self-learning of diagnostic rules through genetic algorithms approach. The training data is divided into various sets corresponding to various faults and the normal operating conditions. These rules are evaluated by a genetic algorithm approach, with contains three basic operators: reproduction, crossover and mutation. This genetic methodology approach can be easily integrated with knowledge-based or expert systems [3], [4], [6], [9]. Self-learning process can facilitate knowledge acquisition effort and is more desirable in these cases where certain knowledge is unavailable. A genetic algorithm based learning can efficiently discover new rules fitted with the training data and provide a means for developing diagnostic rules without any deep knowledge about the process.

II. Problem Solving

A solved diagnostic knowledge-based system uses a combination of knowledge inference processes with genetic algorithm approach and maximum entropy principles for diagnostic rules handling within applied domain of deliberated machinery systems. Maximum entropy principles are used in these special cases, where some diagnostic rules are not suitable for direct using within genetic algorithm approach environment.

Further, we assume a concrete case, that probability values come in at the lowest level of the hierarchy. The initial problem is to propagate these probabilities upward. That is given a hypothesis h_{ii}, supported by evidence – „sub-hypotheses“ $h_1, h_2, ..., h_n$. An idea is to compute $p(h_{jj})$. There is an assumption, that h_{jj} is at the $(m+1)$ – level of the hierarchy, and that the values $p(h_i)$ have already been accrued at the $(m\text{-}th)$ – level. H has been associated to the h_i, based on a prior model of h_{jj}, which requires $(n+k)$ component hypotheses, of which n have been observed, namely the set $(h_1, h_2, ... h_n)$. Let b_m be the number of $(m+1)$ – level hypotheses associated to each level m hypothesis. It is done from mathematical reasoning, that this mentioned value is constant at level m. There is defined:

$$p(H) = \left(\frac{1}{n+k}\right)\left(\frac{1}{b_m}\right)\sum_{i=1}^{n} p(h_i) \qquad (1)$$

We claim, that this is a probability function on any set of level *(m+1)* hypotheses supported by evidence at level *m*. This fact may be mathematically proved. Suppose the probabilities *p* for the concrete hypotheses:

$p(h_{11}) = 0.55 \quad p(h_{22}) = 0.18 \quad p(h_{33}) = 0.22 \quad p(h_{44}) = 0.05$

These values correspond to evidences:

$h_1 = 0.7$	*being relatively certain*
$h_2 = 0.1$	*being relatively uncertain*
$h_3 = 0.2$	*being relatively uncertain,*

then $b_m = 2$.

From this – following a „strongest hypothesis first" strategy - we obtain a global interpretation of (h_1, h_2, h_3, h_4). Assume without loss of generality, that all the h_i have precisely two components required in their model. Then we compute:

$$p(h_{11}) = (2/2) . (1/2) . (0.7 + 1) = 0.400$$
$$p(h_{22}) = (2/2) . (1/2) . (0.1 + 0.2) = 0.150$$
$$p(h_{33}) = (1/2) . (1/2) . (0.7) = 0.175$$
$$p(h_{44}) = (1/2) . (1/2) . (0.2) = 0.050$$

These values are normalising. Given a hierarchical hypothesis space with conflicts, we invoke the rule, that all conflicts must be propagated upward. This increases the generation of alternative hypotheses, which do not claim conflicting evidence. Conflicts should be searched for at the time of hypothesis generation AND/OR associations. All consistent interpretations of the hypothesis space should be computed and explicitly represented in the hypothesis space.

There were made several experiments with the solved problem modelled. In order to evaluate the eventual dependence of the system's performances on the number of the sensors and on the range of their possible output (discrete) values, we made various different simulations (from 3 to 9 sensors) with specific output values. Trying to evaluate the system's sensibility we ran some simulations for example with only one deteriorated sensor.

A solved genetic algorithm starts to work with a set of domain knowledge structures which are coded into binary strings. The diagnostic rules, which are to be evaluated through genetic algorithm, should then be coded. A solved system used here has the diagnostic rules. Diagnostic rules are in the following form, which is easily realised within expert (knowledge-based) system environment.

$$IF(S_1 \wedge S_2 \wedge \ \ldots \ \wedge S_n) \quad THEN \quad (F_i) \qquad (2)$$

This formula states that if symptoms S_1 to S_n are present then the i_{th} fault (*Fi*) occurs. The used symptoms S_1 to S_n correspond to *„n"* different on-line information sources, which could be on-line measurements and controller outputs. Each symptom is considered to take one of the following values: - *„increase", „steady", „decrease", „neutral"*. Symptom *„neutral"* means that the corresponding symptom is not important.

When applying a rule: *„normal"* can match with any values. By introducing this symptom, the condition parts of all the diagnostic rules will be of the same length and the corresponding parts of the rules will represent the same observations.

A genetic algorithm is also an adaptive problem. A researcher J.H.Holland [10] identified the main topics as requirements of an adaptive system: - an algorithm contains the system environment, a set of structures, an adaptive plan, and a some function for measuring the performance of each solved structure. A success of a genetic algorithm is crucially connected with the adaptive plan. The adaptive plan controls „mutation" and „crossover", and hence the production of the new solutions, i.e."childs", from old ones, i.e."parents". Created adaptive plan solves some problems about ensuring a constant population size, the number of times any particular solution can be used as a parent and so on.

There are many possible ways to determine the selection of solutions as effectively as possible.
There is an *Adaptive plan* within a Genetic algorithm scheme:

```
Genetic_Algorithm ()
    1.  Choose the multitude of population.
    2.  Set t=0 and randomly select N possible solutions.
    3.  Define evolution process end criterion.
        ( <Initialise population>
          while<Not (stop condition)> do
        (
    4.  Choose evaluation criterion and evaluate each
        elementary solution.
    5.  For each solution $k_n(t)$, select at random a set of r
        solutions without replacement, $K_r(t)$, from the current
        population, K(t), where $1<r\leq N$.
                <Fitness Evaluation>
                <Selection>
    6.  Select the fittest solution, $k^*(t) \in K_r(t)$.
                <Reproduction>
    7.  Apply the crossover operator to $k_n(t)$ and $k^*(t)$, and
        store the result in the next generation as $k_n(t+1)$.
                <Crossover>
    8.  Apply the mutation operator to $k_n(t+1)$ with
        probability $P_m$
                <Mutation>
    9.  Increment t by one.
    10. Identify the best (and the worst) solution.
    11. Repeat steps two to six until T=t, where T is the
        number of generation to be run.
        )
  <Choose final solution>
        )
```

A solved genetic algorithm was run with a population of 89 over 96 generations with r=2. The crossover operator was allowed to operate over the full length of solutions and the probability of mutation was kept low at 0.01. The implementation does show that a genetic algorithm using adaptive plan can be successful.

An analysis of the change in the relative proportions of schema instances leads to an explanation of the success of genetic algorithm. Since the adaptive plan controls the selection and recombination of solutions it is vital to the success of a solved genetic algorithm.

Set of solutions is represented by a string of numbers. The use of a binary representation means that each point on the string can be occupied by one of only two „alleles“. There are can e either a „1“ or „0“ at each point of the string.

Measurement of the vibrations and temperatures, analysis of input signals and their processing is an important part of this work. Sensors are connected to a processing unit. In the second phase we transform based signal features, which will be used as the inputs to domain knowledge-based system. A basic used genetic approach is simply illustrated in Fig.1.

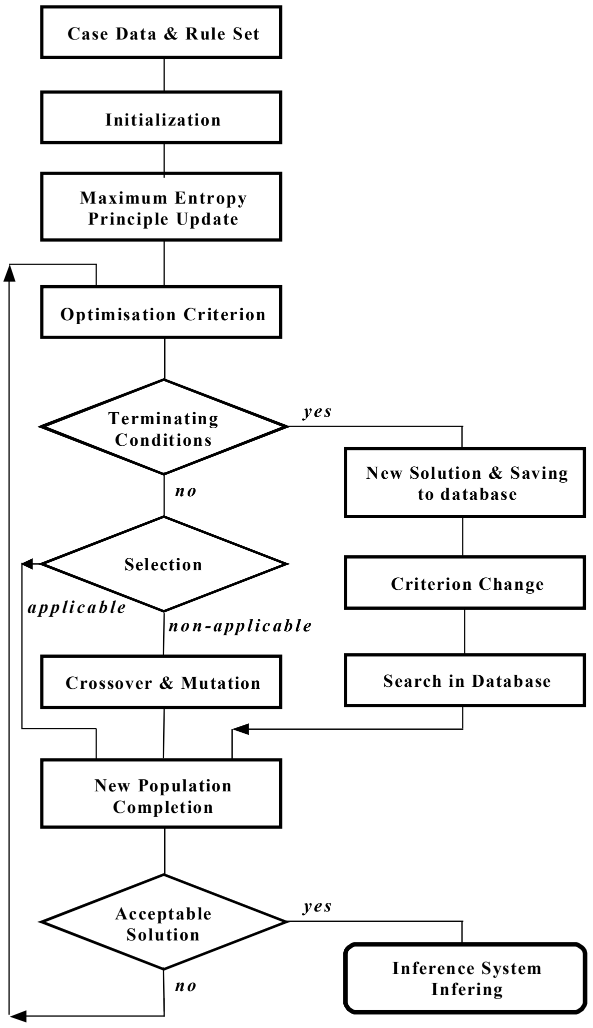

Fig. 1: A basic used genetic approach

An advantage of solved adaptive plan is fact, that each solution is guaranteed at least one share in the next generation regardless of how low its fitness value may be. This ensures that no solution is completely neglected during a single generation. The better solution donate more genetic material than the poorer ones. Since the environment is time independent, the information is invariant and need only be read in once at the beginning of a run of the genetic algorithm.

Decision tree coding is introduced to improve efficiency and accuracy on large graphs.

Genetic algorithm is also a stochastic computational model that seeks the optimal solution to an objective function. The search within clustering process is performed through an iterating procedure applied to a „population“ of „individuals“, i.e. a set of feasible solutions.

The characteristic feature of genetic algorithm is that the searching strategy is similar to biological evolution. It means that the better solutions are reproduced. Worse solutions are discarded. The search strategy is based on the possibility to discriminate between elements in order to resolve which is a good solution of the fitness function and therefore has a good chance of reproducing and generating new elements with its genetic inheritance.

There is the problem-dependent data structure representation, and cost function, i.e. fitness, evaluation, and the robust reproduction phase, which are functionally separated and may be common to each application. That's reason for creating a good and relevant genetic model for clustering and adopting efficient operators for the optimisation.

Each symptom in the condition part of a rule is coded by a *„2-bit“*binary, where *„00“* stands for symptom *„decrease“*, *„01“* stands for symptom *„“steady“*, *„10“* stands for symptom *„increase“* and *„11“* stands for *„neutral“*. By such a means, a rule whose condition part includes *„n“* symptoms will be coded into a *„2n-bit“*binary string consisting of *„0s“* and *„1s“*. Each set of training data represents the qualitative states of the process under the corresponding non-normal state or fault with different severities. A set of training data representing normal operating conditions is also required. Training data could be obtained from simulation studies or from previous operating experience of the process. The fitness of an individual structure is a measure indicating how fitted the structure is. In the self-learning of diagnostic rules, a better rule should have more applications when tested by the training data corresponding to this rule and fewer incorrect applications when tested by the other training data.

The fitness *„FIT“* of the rule is calculated by following way:

$$FIT = \frac{k_1\, O}{k_2\, P + 1} + k_3\, Q \qquad (3)$$

where *„O“*.....................is a number of successful applications of a rule when tested by the training data corresponding to this rule,

„P“......................is a number of incorrect applications of the rule when tested by the rest of the training data,

„Q“.....................is a number of *„neutral's“* in the condition part of the rule.

„k_1, k_2, k_3“...........are positive weighting coefficients,

The requirement that *„FIT“* should not be negative is determined by the genetic algorithm, which is used here in solved problem. In the case of the reproduction phase, an individual rule is selected with a probability, which is equal to the ratio of the fitness of the rule and the sum of the fitnesses of all the rules in the generation. This is also the reason for conclusion, that the fitness of a rule should not be less than zero.

This fitness function (3) is not a linear function. By inspecting the first order partial derivatives, if we suppose that *„O, P, Q“* could change continuously, it is found that the fitness function (3) is preferable to the first one. Differentiating fitness function (3) provides following equations:

$$\frac{\partial\, FIT}{\partial\, O} = \frac{k_1}{\partial\, P + 1} \tag{4a}$$

$$\frac{\partial\, FIT}{\partial\, P} = \frac{-k_1 k_2 O}{(k_2 P + 1)^2} = \frac{-k_2 O}{k_2 P + 1} \cdot \frac{\partial\, FIT}{\partial\, O} \tag{4b}$$

$$\frac{\partial\, FIT}{\partial\, Q} = k_3 \tag{4c}$$

A non-linear fitness function is more sensitive to changes in *„O, P“* when a rule is close to the desired rule, with large *„O“* and small *„P“*. The values of the parameters *„k_1, k_2, k_3“* in relation (3) can affect the result of learning. The choices of *„k_1, k_2, k_3“* are determined by the quantity of training data and the user's objectives.

Relation (4b) shows that a large *„k_2“* will impose a larger penalty on incorrect applications. The experiments have shown that a large *„k_2“* will produce rules, which are more „cautious“ in that they rarely have incorrect applications, but they could miss some faults, while a small *„k_2“* will produce rules, which generally will not miss the corresponding faults but could produce incorrect diagnoses [2], [5].

The probability of a rule being selected is defined as follows:

$$P(i) = \frac{FIT(i)}{\sum_{j=1}^{n} FIT(j)} \tag{5}$$

where *P(i)* is the probability that the *„i_{th}“* rule will be selected.

„FIT(i)“	is the fitness of the *„i_{th}“*rule,
„N“	is the total number of rules in the current generation,
„j“	is a running index taking values from *1* to *N*.

By such means, more fitted rules will be selected more frequently and produce more copies in the new generation. After the reproduction phase, the rules are randomly mated to undergo the crossover operations. Through the crossover operation with the probability, which is often large (about 0.8), the 2 rules in a pair will exchange a certain amount of knowledge (information). This procedure produces new knowledge structures. Then is realised the mutation operation, which performs bit by bit changes in the coded rules – from *„0“* to *„1“* or from *„1“* to *„0“* with a probability which is typically small (0.001 to 0.01).

The clustering problem within genetic algorithm approach may be characterised as follows.

We have: - „n“-dimensional feature space named „S“
- „B“, which is subset of „S“ with „N“ elements „X_j“ represented with „n“ coordinates in „S“.

$$B \subseteq S$$
$$B = \{X_j \,/\, X_j \in S, j=1,...,N\} \tag{6}$$

We solve the problem to search the best „T“ partition of „B“ in „m“ clusters, such as:

$$T = \{C_i,\ i=1\ ,...,\ m\}$$
$$C_i = \{X_j \,/\, X_j \in B, j=1,..,N_i\}$$
$$\cap\, C_i = 0$$
$$\cup\, C_i = B \text{ or } \Sigma_{(i=1,...,m)} N^i = N \tag{7}$$

and the „best“ idea is correlated to the concept of „similarity“ between objects „X_j“.

Coding is realised as the first step, which is efficient data representation with a binary code. Each „T“ solution is coded as a chromosome, composed of a set of variables or genes. A first formulation can consider N genes (one to each object) which can assume a district value between „1“ and „m“ to indicate the belonging cluster. The „m“ could be fixed or indicate the maximum group number. Therefore the binary code needs „N“multiply „Int(log2(m))“ bits. As example we divide 30 objects into 5 classes. Each chromosome has a 60-bit length. The code call to modify the reproduction task in order to generate only consistent solutions.

In phase selection the reproducible solutions are selected. It is convenient to search for a good compromise between selecting only the best elements to increase the algorithm convergency and avoid a block around local minima of the cost function. The classical „Monte Carlo“ statistic selection has been adopted for problem solving. A sort of „elitist selection“ has been investigated to conserve the best solutions achieved in each generation by introducing a given number of „clones“ in the new generation. The clonation percentage has to be carefully detected in order not to limit the genetic model and concentrate the research only in one direction. The best percentage seems to be 24% maximum because a higher value demonstrates an average fitness of more than one, suggesting that several tests do not find the best solution and are trapped in local optima. Variable clonation could be a good proposal but depends also on the application domain characteristic.

Reproduction method approach has the „crossover“ as a basic operator to reproduce new solutions from those selected in a given generation. A proposal could be to mix two approaches, using the standard coupled reproduction with clones and "asexual" generation. In all reproduction phases the mutation operator has been applied with an experimentally set value from 0.01 to 0.001 variable percentages to permit an alternative manifold hyperplane exploration and avoid premature convergence.

Then, there is the fitness model, which must exalt the difference between applicable and less applicable solutions in order to converge to the optimum solution, accepting also unbalanced clusters or clusters with a single element. On the contrary, in some applications, there are may be many acceptable solutions. In this case the goal could be to promote a balanced distribution of objects in a given number of clusters. In this context the „fuzzy theory approach“ may be combined with the genetic model, for example by putting a value between „0“ and „1“ in the Boolean cluster code to act as object belonging probability.

The problem searches the best feasible solution named „U“ in „m“ clusters. The „cost function“ to be minimised was experimentally given by (see relations (6) and (7)):

$$f(U) = \sum_{i=1}^{m} \frac{1}{N^i} \sum_{j=1}^{N^i} d(X_j, \overline{X^i}) \tag{8}$$

This problem aims exploit the effectiveness of the solved genetic algorithm mostly in clustering, especially in a context where the data dimensions have not been supported by exhaustive solutions and the problem generality does not always find a usable solution with heuristic techniques. The aim is to create a robust model for good clustering in analysis related to the application goals.

After the learning and testing phases, „n“ rules are probabilistically selected according to their judged „strength“. The „crossover operator“ generates two children from each pair of selected parents. It is a one-point crossover. The resulting offsprings are added to the population in case they were not. The „parent rules“ are not deleted – reproduction operator. The „crossover probability P“ is in the form:

$$P = p_S \text{ “multiply“ } (1- p) \tag{9}$$

There is a system's parameter „p_S“, which has in our experiment the value: „0.77“. From this reason, the „crossover probability“ decreases as the success rate increases. To determine the „strength“ of new rules, we bear in mind the „strength“ and the specificity of the „parents“. We will obtain the „children strength“ by the following way. We assume two „parent“ rules „R_1“ and „R_2“. Next, we consider a „one-point crossover operator“ randomly set in condition part of solved parent rules. For each „parent rule“ we consider the two parts „L“ and „R“, which correspond to left and right position, on both sides of the crossover point (random mating). There are „specificities“ of the left and right position. The left side specificity of „R_1“ or „R_2“ is marked by „$c_L(R_1)$“ or „$c_L(R_2)$“. The „$c_R(R_1)$“ or „$c_R(R_2)$“ is the specificity of right side of „R_1“ or „R_2“.The „$S(R_1)$“ or „$S(R_2)$“ is „strength“ of „R_1“ or „R_2“. We also consider „R_{F1}“ and „R_{F2}“, which are the two „offspring rules“. The below mentioned relations allow a „child rule“ to inherit the „parent rule's strength“ proportionally to the specificity of the inherited part, i.e. number of instanced attributes to divide with number of attributes of left or right part. It caused this fact, that the new rule's strength is all the higher as the rule inherits of a high specificity (10).

$$S(R_{F1}) = \frac{c_L(R_1)\,\text{"multiply"}\,S(R_1)\,\text{"plus"}\,c_R(R_2)\,\text{"multiply"}\,S(R_2)}{c_L(R_1)\,\text{"plus"}\,c_R(R_2)}$$

$$S(R_{F2}) = \frac{c_R(R_1)\,\text{"multiply"}\,S(R_1)\,\text{"plus"}\,c_L(R_2)\,\text{"multiply"}\,S(R_2)}{c_R(R_1)\,\text{"plus"}\,c_L(R_2)} \tag{10}$$

Similarly, above mentioned form is equal for others rules in solved generations.

We carried out several series of tests to study the best choices. For each test, used genetic operators are turned off. A test is evaluated mostly on a single cycle, i.e. one „learning“ and „testing“, by examining the success rate, the partial failure rate, the total failure rate, the final population size, the maximal strength of the rules, the maximal specificity of the rules and the average specificity of the rules. The partial failure on a testing set example is when the best rule in the matching set does not have the correct consequent, and the total failure is when the matching set is empty.

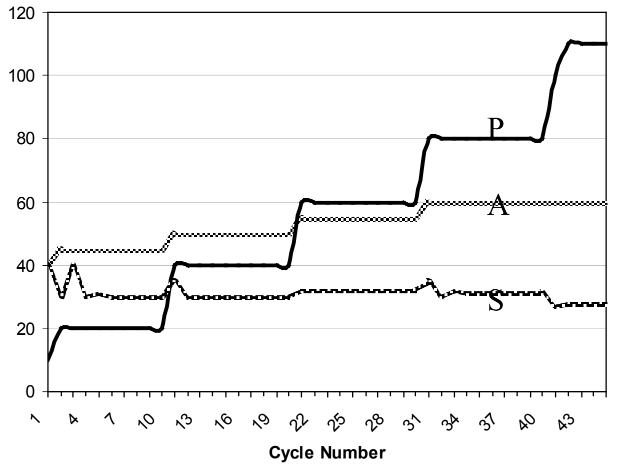

Fig. 2 System evolution experiment 1

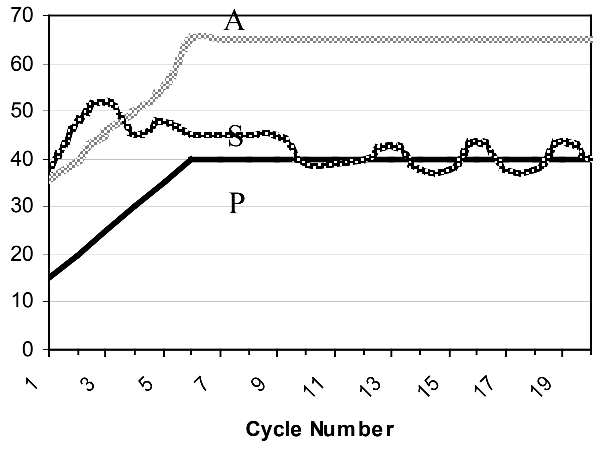

Fig. 3 System evolution experiment 2

Fig. 2 shows the solved system evolution experiment over twenty-four cycles. Twenty rules used in the testing phase are created during the first 5 or 6 cycles. The success rate decreases as the specificity of the rules used increases. From cycle about seven, the create operator is ineffective. For this reason the new rules cannot be created and the population size remains equal to 399. More rules have been created in the case of crossover operator with the „p_S“ value equal to 0.77. The success rate and average specificity have lower values than in the case of above mentioned experiment, but we have more rules (455). Eleven rules were used during the last testing phase. Three rules were created by he create/specialise operator and eight rules by crossover operator. In the case of experiment with thirteen used rules during the last cycle, six rules were created with create operator and seven by crossover operator (see Fig. 2 and Fig. 3, curves „A“,„P“,„S“). The strength of the new created rules is updated during the following cycles by reinforcement algorithm. Fig. 3 shows experimental results with create operator and crossover operator application.

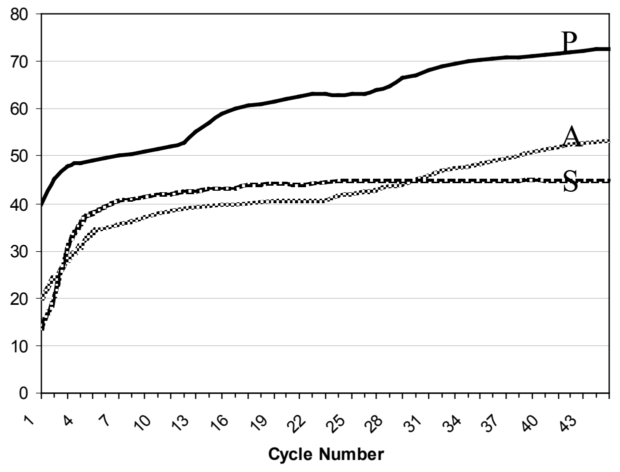

Fig. 4 System evolution experiment 3

„Crossover“ in the first experiment is turned off and special operator is invoked at each cycle, and also the crossover approach is involved every 9 cycles. The whole number of rules is 555, the average specificity over all the rules is 0.41. The success rate is 39%. The „crossover“ in the second experiment (p_S=0.77) and special operators are used every 9 cycles. The whole number of rules is 998, the average specificity over all the rules is 0.56. The average specificity of the used rules is 0.44 and the success rate is 41%.

Fig. 4 shows experimental results achieved with mutation operator application. The mutation operation has been applied in these cases, where the create operator was ineffective. The new rules can be created with new quality. Each hidden unit is susceptible with a small probability (from 0.01 to 0.1) to a random perturbation in its weights with each generation. This procedure was implemented in two ways. A small random change to each weight was realised, and then a rerandomisation of a single weight in the hidden unit was raised. The mutation activity was realised by mostly experimentally tested methodology. This approach may be considered more reliable.

Experiments and the performance of the solved system have been studied on a complex diagnostic environment, which is also uncertain. To better know the learning data used, we will make use of classical classification techniques over the data (first of all some statistical approaches), and we will compare the obtained data by various methodologies approaches. The developed system gives us a lot of information not only on the solved system's behaviour, but also on the component of each rule.

III. Process of Learning and Self-Learning of Diagnostic Rules

There is as an example of measurements, which determines that the condition part of a diagnostic rule should contain nine symptoms each corresponding to an on-line information source. The training data are obtained by simulating the process under various faults and abnormalities and with noisy measurements. Rules are coded as binary strings. There are some relevant values: - *„the probability of crossover „* (for searched diagnostic system) was set to 0.91, *„the probability of mutation“*(0.002), *„fitness function parameters“* (*„k_1“*= 5.15, *„k_2“*=0.22, *„k_3“*=2.). These parameters are chosen through trial and error but are roughly guided by relations (6a), (6b), (6c). The performance of the realised learning system is illustrated in Table I. Experiments have shown that in many cases have been obtained significant improvements. There are best and average fitness of rules in initial and final generations (Table 1).

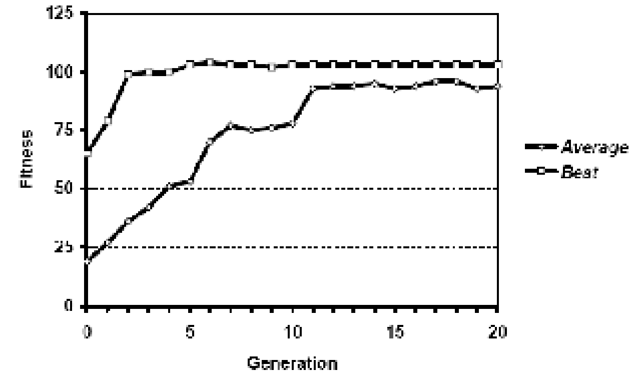

Fig.5: Characteristic learning rule with average and best fitness

The learnt rules have been tested on the real process. A sample of the results of approximation on training data from sensors placed on machine tool equipment and the results of approximation on testing data (i.e. on unknown data) are shown in fig.5. The approximation by genetic algorithm gives the acceptable accuracy. The determination of machinery equipment state and manufacturing or technological conditions states, which is done first of all in prediction of relevant signal course and its analysis by solved predictive methodology on genetic algorithm basis will be sufficiently more accurate, correct and precise than others used approaches. Using of the artificial genetic algorithms for predictive machine tool equipment maintenance approximation seems to have better performance than the classical approach.

The rules could be improved by filtering out the bad training data prior to learning. The rules could also be improved by including additional features, such as the magnitudes of deviations in measurements, in their condition parts to increase their resolution.

Table I. Fitness of rules in initial and final generations

Rule	Generations	Average	Best
1	0	19	65
2	1	27	79
3	2	36	99
4	3	42	100
5	4	51	100
6	5	53	103
7	6	70	104
8	7	77	103
9	8	75	103
10	9	76	102
11	10	78	103
12	11	93	103
13	12	94	103
14	13	94	103
15	14	95	103
16	15	93	103
17	16	94	103
18	17	96	103
19	18	96	103
20	19	93	103
21	20	94	103

IV. Conclusion

The aim of solved system is to develop an effective system, which is able first of all to provide intelligent advice for solving complex technical (diagnostic) problems efficiently and precisely with its relevant domain expertise or skills. Performance of the approach was verified on a real process of machine tool conditioning. Real-time diagnostic results show that each component correctly represents examined states or faulty states of the system.

A dynamic model and dynamic simulation of this system is capable of investigating the operation of the process under normal conditions as well as various faulty conditions. There are can be distinguished also the following possibilities. In the case of a dangerous process state the system automatically initiates an appropriate counteraction. Features are determined and a fault diagnosis is performed in advanced part of solved knowledge-based system. The purpose of the solved system is to detect and localize in time possible abnormalities and faults of machinery equipment and thus prevent the damage of technological process. A solved system will be extended in future to include various complicated forms of rules, for example with more complicated condition parts of the rules, which will include combined qualitative deviations in on-line information with various features such as the magnitudes of the deviations, the shapes in measurement curvatures and so on. There are will be investigated self-learning approaches of diagnostic rules through more advanced genetic and evolutionary algorithms and modified chaos theory principles. Nowadays, some achieved results seem to be very interesting.

The developed system gives us a lot of information not only on the solved system′s behaviour, but also on the component of each rule. Genetic algorithm will have been the main rule discovery algorithm. It will concern to obtain self-organisation of a kind of communication protocol among a solved population.

The main feature of genetic algorithm is that the searching strategy is similar to biological evolution, i.e. better solutions are reproduced,

The quality of a genetic algorithm approach seems to be dependent on a few important parameters and operator variants. There are a number of others issues for future research, such as improving the runtime performance of our implementation, including other genetic operators in the architecture. The fuzzy theory may be combined with the genetic model, for instance by putting a value between „0“ and „1“ in the Boolean cluster code to act as object belonging to probability.

The research reported on this paper is supported by following scientific projects of Slovak Grant Agency VEGA (Ministry of Education):

1. *Diagnostic System Building of Machinery Equipment with Knowledge-based System and Chaos Theory Applying*. (Number: VEGA 1/1084/04)
2. *Intelligent Approach to Automated Diagnostic Problem Solving of Machinery Equipment.* (Number: VEGA 1/4133/07).

References

[1] S. K. Anderson, K. G. Olesen, F. V. Jensen, *Hugin – a Shell for Building Bayesian Belief Universes for Expert Systems.* In: Shaffer, W & Pearl, J. (eds.) Readings in Uncertain Reasoning, Morgan Kaufma, USA, 1990.

[2] S. A. Goldman, R. L. Rivest: *A Non-iterative Maximum Entropy Algorithm.* In: Koval, L. N. & Lemmu, F. J. (eds.) Ucertainty in Artficial Intelligence, Vol.2, pages 133-148, North-Holland, 1988.

[3] R. L. Chilausky: *An Experimental Comparison of Several Many-valued Logic Inference Techniques in the Context of Computer Diagnosis of Soybean Disease.* International Journal of Man-Machine Studies. Department of Computer Science, pages 178-162, University of Illinois, Illinois, 1980.

[4] P. A. Cohen: *A Theory of Heuristic Reasoning about Uncerainty.* The AI Magazine, Summer 1983, Report 84.

[5] R. W. Hamming, *Coding and Information Theory.* Prentice Hall Inc., Englewood Cliffs, N.J., 1980

[6] E. Nissan, *Intelligent Technologies for Nuclear Power Systems.* Heuristics and Neural Tools. In: Expert Systems. 1998

[7] J. A. Shore, R. W. Johnson, *Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross-entropy.* IEEE Transactions on Information Theory, pages 26-37, Vol.IT-26, 1980.

[8] K. Watanabe, I. Matsura, M. Abe, M. Kubota, D. Himmelbian, *Incipient Fault Diagnosis of Chemical process via Artificial Neural Network's.* AIChE Journal Vol 35 No 11, 1989

[9] J. Zhang, P. D. Roberts, J. E. Ellis, *Fault Diagnosis of a Mixing Process Based on Qualitative Representation of Physical Behaviours.* In: Journal of Intelligent and Robotic Systems, 1987.

[10] J. H. Holland, *Adaptation in Naturaland Artificial Systems.* University of Michigan Press, 1975