

Effective Techniques Applying via Genetic Algorithm Approach

Stefania GALLOVA, *Member, IAENG*

Abstract— The relevant idea is to consider the search for the best genetic algorithm approach as an optimisation problem and use another genetic algorithm approach to solve it. A methodology calculation is based on the idea of measuring the increase of fitness and fitness quality evaluating created by two methodologies with secondary genetic algorithm approach using. Performance results for finding the best genetic algorithm for the complex real problem of optimal machinery equipment operation and predictive maintenance are presented. We illustrate two interesting solutions approaches within genetic algorithm environment.

Index Terms— genetic algorithm, fuzzy set, fitness, crossover, mutation.

I. INTRODUCTION

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. The purpose of the solved system is also to detect and localize in time possible abnormalities and faults of machinery equipment and thus prevent the damage of technological process.

A solved diagnostic system is very complex. There are the following topics:

- equipment condition monitoring
- data acquisition for equipment reliability modeling
- database for equipment state evaluation
- knowledge-based system for equipment fault diagnosis
- impact of plant operation on maintenance activities.

II. PROBLEM SOLVING

A measurement of the vibrations and temperatures, analysis

Manuscript received March 22, 2008. This work was supported by the Slovak Republic's Ministry of Education under Scientific Grant Project: *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; stefania.gallova@tuke.sk

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. 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.

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 be either a „1“ or „0“ at each point of the string. A solved genetic algorithm starts to work with a set of domain knowledge structures that 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 (production rules representation). Diagnostic rules are in the following form, which are easily realised also within expert (knowledge-based) system environment:

$$IF (S_1 \wedge S_2 \wedge \dots \wedge S_n) \quad THEN \quad (F_i) \quad (1)$$

This formula states that if symptoms S_1 to S_n are present then the i_{th} fault (F_i) 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*. Each symptom in the condition part of rule is coded by a 2-bit binary, where „00“ stands for symptom *decrease*, „01“ stands for symptom *steady*, „10“ stands for symptom *increase*, „11“ stands for symptom *neutral*. Symptom *neutral* means that the corresponding symptom is not important. For example, a *normal* rule applying 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.

The quality of a genetic algorithm approach seems to be dependent on a few important parameters and operator variants. Fitness function, as well as the parameters of the fitness function, can affect the result of learning process. The fitness of an individual structure is a measure indicating how fitted the structure is.

We illustrate a genetic algorithm scheme:

```
Genetic_Algorithm ()  
( <Initialize population>  
  while <Not (Stop condition)> do(  
    <Fitness evaluation>  
    <Selection>  
    <Reproduction and Mutation>  
  )  
  <Choose final solution>  
)
```

A genetic algorithm is a stochastic computational model that seeks the optimal solution to an objective function. The search is performed through an iterating procedure applied to a „population of individuals“, i.e. a set of feasible solutions. Searching strategy is similar to biological evolution, i.e. better solutions are reproduced, whereas worse solutions are discarded. Thus, the search strategy is based on the possibility to discriminate between elements in order to resolve which is the good solution of the fitness function and therefore has a good chance of reproducing and generating new elements with its genetic inheritance.

Experiments with different approaches to solve fitness function models evaluation lead to the different results. Earlier experiments with a single population and simple (classical) based genetic algorithm approach converged prematurely to solutions of poor quality. A genetic algorithm has to maintain a balance between the preservation of good combinations of genes, and the exploration of new combinations. We adopt a successful strategy for achieving this balance which has been to combine a highly explorative, or disruptive crossover with elitism, in which a fraction of the best individuals found so far survive into the next generation. Elitism gives better individuals more chances of mating to produce fit offspring, an advantage when their offspring will frequently be poor.

This (solved) system is a good illustration of how a genetic algorithm approach to a complex practical combinatorial problem can provide an extremely robust solution with several practical advantages. The main difficulty in the problem is that there typically is a multitude of local extreme, which happen to be located close to a bounding constraints, conventionally imposed at the given threshold, and that anyway has to be imposed out of safety considerations.

The aim of the research (two methodologies) described here was to investigate the factors involved in designing a genetic algorithm with respect to the overall objective of robustness and utility as a practical tool. By robustness, we mean the ability of the program to produce good solution in reasonable time, independently of any user interaction in the form of careful set-up or run-time intervention, possibly requiring considerable expertise.

Given the time constraint within which the program must work, within a genetic algorithm approach the problem devolves into the design of a number of components: genetic representation of candidate solutions, selection of mating pairs and recombination of their genetic material, random mutation of genetic material, measure of fitness for a given population, and population distribution.

We solve a non-linear fitness function, which has been constructed (and optimized) to direct the search efficiently in

the presence of the many local optima that result for the constraint on solutions. The contribution describes the design of an efficient and robust genetic algorithm for the diagnostic system problem – a complex combinatorial, multimodel optimisation. A technical computing environment is Matlab programming tool.

A. First methodology approach

The primary genetic algorithm is applied to find the best genetic algorithm for a secondary problem, which closely resembles the properties of the primary problem, see Fig.1.

Each of the secondary runs of genetic algorithm independently to produce a solution of the problem considered. The fitness of the solution influences the operation of the primary genetic algorithm. Such a problem would require (for example) real valued genes, the possibility to treat logical subgroups of genes as an atomic unit and a sufficiently complex search space with multiple suboptimal peaks. The values obtained for the best secondary genetic algorithm in this scenario are then copied and used as the parameter settings of the primary genetic algorithm for optimizing the secondary genetic algorithm for the particular problem to be solved.

On the bottom level, the secondary genetic algorithm approach operates on a population of gene strings that represent possible solutions of the problem to be solved. On the top level, the primary genetic algorithm approach works on a population of secondary genetic algorithm approach, each of which is represented as a separate gene string.

Each of the secondary genetic algorithm approach runs independently to produce a solution of the problem considered, and the *fitness* of the solution influences the operation of the primary genetic algorithm approach. The number of generations created on the two levels is independent of each other. The string with the highest fitness in the last primary generation is expected to be the best genetic algorithm for the original problem. Generally, a priori information determines the kinds of genetic operators and their parameters settings for the primary genetic algorithm approach. The genetic operators and parameters values are initially determined by the designer (domain expert) and by the Internet technology approach.

The genes as a decision in the string are numerically represented by the probability that a particular variant of a genetic operator is selected among a limited number of variants of that operator. The genes as parameters in the string specify a real-coded value associated with the selected variant.

We define many parameters and operators within solved genetic algorithm approach. There are some more relevant parameters. *Elitist component* decides if the best string generated up to time t should be included in the population of generation $t+1$. The distance between two strings is measured and their fitness is modified in the sense that strings in the same neighbourhood are forced to share their fitness among another, which effectively limits the uncontrolled growth of particular species within a population. We use three various crossover operators: *order crossover*, *cycle crossover* and *partially matched crossover*. We have also a *distance of crossover points* parameter, which determines the maximal

distance between two crossover points in the used crossover operator for reordering problems. The *crossover unit* reflects the decision if the crossover operator (as usual) should consider genes as the smallest atomic entity. Another possibility considers if it should be applied such that logical subgroups of genes stay together as a structural unit [2], [3].

The crowding technology has been introduced to induce niche like behaviour in genetic algorithm search in order to maintain diversity in the population. *Mutation probability* parameter determines the probability of applying the mutation operator to a gene. We also realise the decision if the mutation amount of real-valued genes is determined according to a normal distribution or an exponential distribution. Special parameters determine the density function of the normal distribution of the mutation amount and the mean value of the exponential distribution of the mutation amount. *Mutation value replacement* reflects the decision of the mutation operator should overwrite the old gene [4].

We explicitly distinguish between decisions and parameters values and provide several alternatives for the decision components. In the absence of an a priori known best or worst fitness of a string, the solved methodology of fitness calculation is based on the idea of measuring the increase of fitness created by the secondary genetic algorithm approach. We have a parameter F_n^m , which is the best fitness of a string in the secondary population for test problem example m after generation n . A parameter p_t is the number of test problems. The *fitness* of the secondary genetic algorithm GA_{FS} after generation n is given by:

$$Fitness[GA_{FS}] = \frac{1}{p_t} \sum_{m=1}^{p_t} \frac{\max(0, F_n^m - F_0^m)}{1 - F_0^m} \quad (2)$$

By this way, we will avoid distorted fitness values arising from possibly different degrees of complexity of test problem examples. The fitness of an individual structure is a measure indicating how fitted the structure is [5].

We realize many experiments with various changes of genetic algorithms operators. For example, we implemented mutation as follows. A single mutate step with two new (created) children consists of randomly choosing two bits in the string of length 1,660 and interchanging their values. The probability that a single mutate will actually modify an individual is 0.29.

Fig.2 shows the performance profile with indicators of the most significant decisions. They become important after the 70th primary generation, when other decisions and parameters have already been appropriately determined. The optimal crossover probability evolved as 0.51. This value remained nearly constant for all strings lengths investigated. The optimal mutation probability value increases with increasing numbers of genes in a string.

The fitness of the best secondary genetic algorithm approach increases with increasing primary generations. The best secondary genetic algorithm is selected after every 6th primary generation, applied it to 60 randomly generated test problems with different numbers of weights parameters to optimize (for 120 secondary generations) and measured its performance.

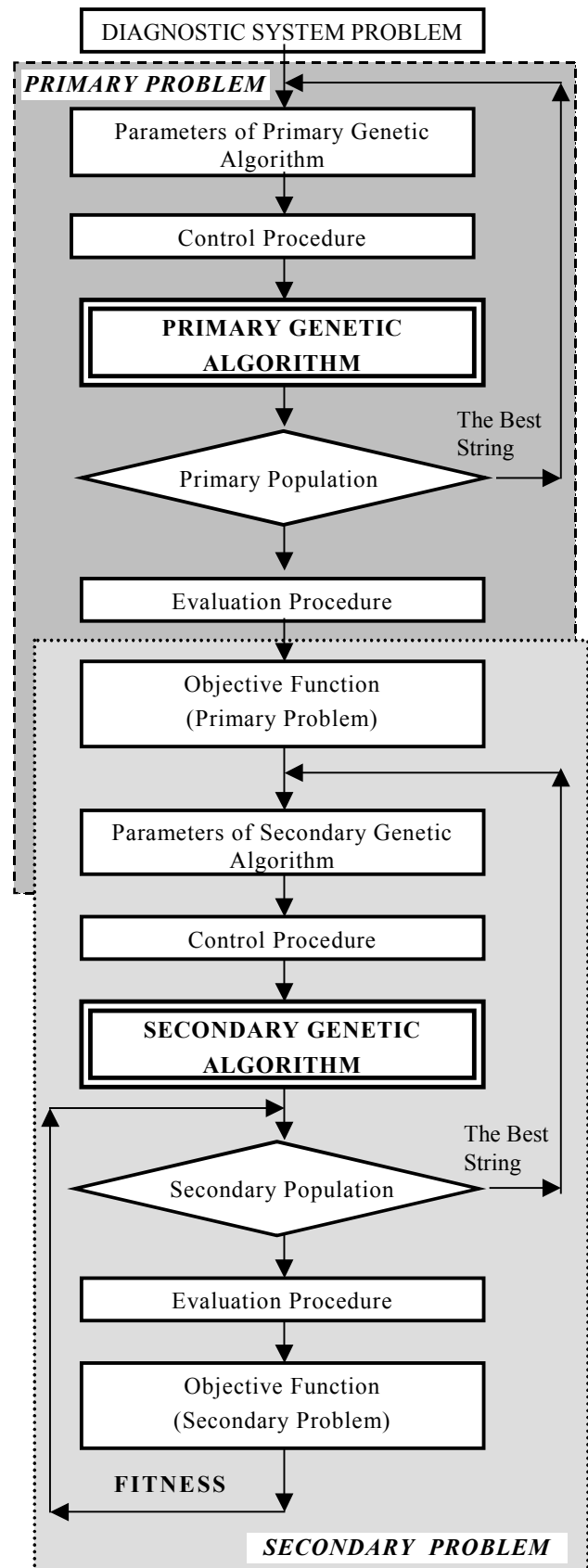


Fig. 1 Optimization problem architecture (two-level approach)

The total computation time required for performing the whole cycle was 38 hours. It involves 115 primary generations and a total of 79000 secondary generations for the 42 test problems every 6th primary generation. This

complex procedure was repeated six times. The final values of fitness performance profile are averages over the six experiments, see Fig.6. Up to primary generation 22, the increase in quality is slow, but then it significantly gets larger during a few primary generations. In the third phase, the increase again slows down. The main reason for this type of behaviour is that the quality of a genetic algorithm methodology seems to be dependent on a few important parameters and operator variants [5], [6].

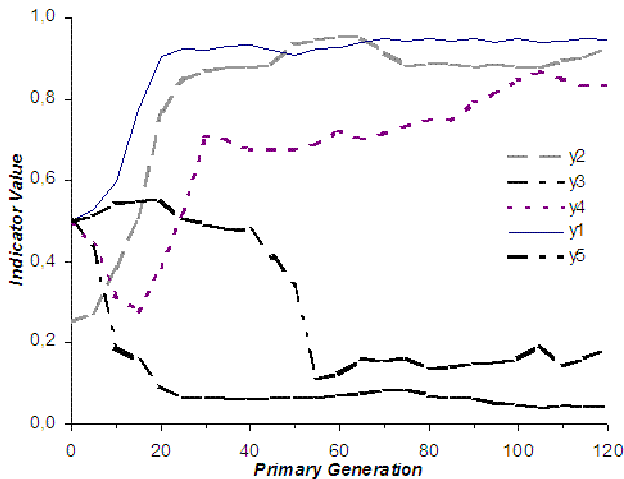


Fig. 2 The Most Significant Decisions Indicators:
y1 = crossover units course; y2 = selection method course;
y3 = elitist model course; y4 = mutation function course;
y5 = value replacement course

The solved genetic algorithm approach is feasible by implementing it in a multi-transputer environment. Performance results for finding the best genetic algorithm for the problem of optimal operation parameters in solved diagnostic system have demonstrated the quality of our implementation.

B. Second methodology approach

If we have a solved diagnostic signal, which is more complex, vague, uncertain, we realize a second methodology approach (we observe and analyze more signal parameters). The reason for this approach is to implement a real predictive maintenance system.

Fuzzy rules based system with an optimization by genetic algorithms approach can be effectively used to obtain relevant results in real diagnostic system problem solving. Fig.3 illustrates a fuzzy rules handling procedure.

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.

We have a primary expert system (Spel-expert 4.0) and secondary GA approach.

The aim is to detect changes of the current process behaviour and to generate analytical symptoms. The diagnosis task is accomplished by fuzzy evidential approximate reasoning scheme to handle different kinds of uncertainty that are inherently present in many real world processes, and to make decision under conflicting data or knowledge.

The diagnostic system serves several purposes. It identifies

the optimal decision boundaries between the different faulty states with as many details as possible or needed even in the presence of noise and uncertainties.

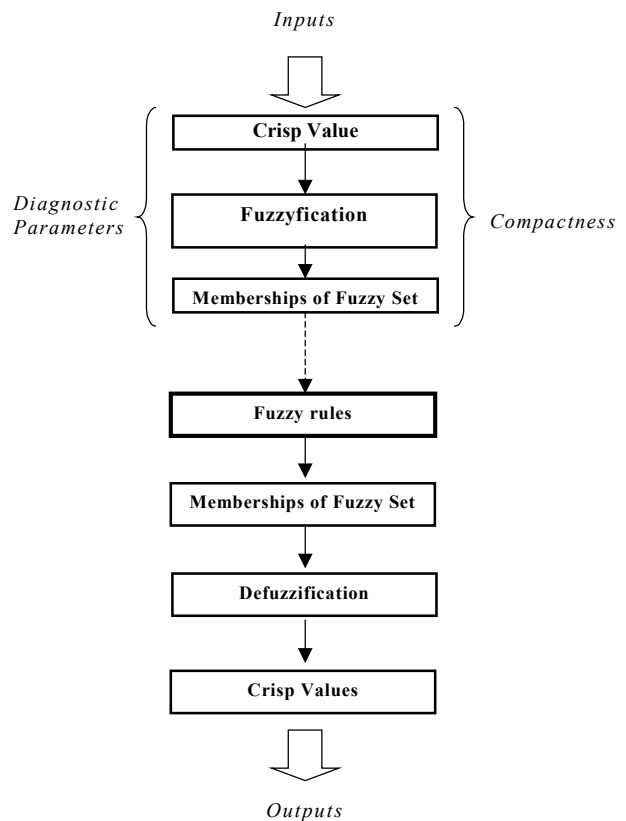


Fig. 3 A Fuzzy Rules Handling Procedure

The diagnostic system can be improved and can be more effective when taking into account aspects like simplicity, transparency, uncertainty and conflict management. The diagnostic process includes first of all trees of fault symptoms and fuzzy rule basis that use explicit knowledge to treat heuristic symptoms, which are mostly obtained by inspections through a human operator and by stating the statistics history of the monitored process. The appropriate heuristic rules for the fault diagnosis are then gathered through consulting experts [1].

Diagnostic process includes also measurement data to generate analytical symptoms. We have to solve the lack of sufficient information and the existence of uncertainty. We solve an iterative hierarchical optimization problem.

Procedure of parameter optimization is realised by following way. The identification procedure of obtained parameters leads to optimization and tuning procedures. We solve the forward procedure. The functional models FM_{ij} , $i=1, \dots, m; j=1, \dots, l$ are identified by solving a least square problem. The backward procedures fix the functional models. The parameters of the membership functions p_{ik}, q_{ik} ; $i=1, \dots, m; k=1, \dots, n$ are updated by an effective non-linear gradient descent optimization technique. It requires the computation of derivatives of the objective function to be minimized with respect to the parameters p_{ik}, q_{ik} . We apply the optimization algorithm with variable learning rates process using.

We have a set $S = (x^p, s^p)_{p=1}^N$, such that $x^p \in X \subset R^l$; $s^p \in Y \subset R^l$, the objective is to find subsystem $y_j(x^p)$ in the form:

$$y_j(x) = \frac{\sum_{i=1}^m FM_{ij} \prod_{k=1}^n e^{-\frac{(x_k - p_{ik})^2}{q_{ik}^2}}}{\sum_{i=1}^m \prod_{k=1}^n e^{-\frac{(x_k - p_{ik})^2}{q_{ik}^2}}} \quad (3)$$

We minimize the function of the mean squared error:

$$E_r = \frac{1}{2} \sum_{j=1}^l (y_j - s_j^p)^2 \quad (4)$$

where $x^p \in S$.

We solve the mean p_{ik} , variance q_{ik} (i.e. ellipsoidal functions) and the adjustment of the FM_{ij} . We also assume that $p_{ik} \in X_i, q_{ik} > 0; FM_{ij} \in Y_j$. We solve a complex non-linear multi-input and multi-output relationship with $x=(x_1, x_2, \dots, x_n)^T \in X \subset R^n$. Parameter x is the vector of input variables. We have also $y \in Y \subset R^l$. Parameter y is the vector of output variables. Output y and E_r depend on p_{ik}, q_{ik} only through equation (3). We have the following equations:

$$y_j = \sum_{i=1}^m FM_{ij}$$

$$y_j(x) = \frac{\sum_{i=1}^m FM_{ij} \prod_{k=1}^n e^{-\frac{(x_k - p_{ik})^2}{q_{ik}^2}}}{\sum_{i=1}^m \prod_{k=1}^n e^{-\frac{(x_k - p_{ik})^2}{q_{ik}^2}}} \quad (5)$$

We have a substitution:

$$U = \prod_{k=1}^n e^{-\frac{(x_k - p_{ik})^2}{q_{ik}^2}}$$

We realize derivatives of E_r :

$$\frac{\partial E_r}{\partial p_{ik}} = \frac{\partial E_r}{\partial U} \cdot \frac{\partial U}{\partial p_{ik}} = \sum_{j=1}^l \left(\frac{\partial E_r}{\partial y_j} \cdot \frac{\partial y_j}{\partial U} \right) \cdot \frac{\partial U}{\partial p_{ik}} = \left[\sum_{j=1}^l \frac{(y_j - s_j) \cdot (FM_{ij} - y_j)}{\sum_{i=1}^m U} \right] \cdot \left[2 \cdot U \cdot \frac{(x_k - p_{ik})}{q_{ik}^2} \right] \quad (6)$$

$$\frac{\partial E_r}{\partial q_{ik}} = \frac{\partial E_r}{\partial U} \cdot \frac{\partial U}{\partial q_{ik}} = \sum_{j=1}^l \left(\frac{\partial E_r}{\partial y_j} \cdot \frac{\partial y_j}{\partial U} \right) \cdot \frac{\partial U}{\partial q_{ik}} = \left[\sum_{j=1}^l \frac{(y_j - s_j) \cdot (FM_{ij} - y_j)}{\sum_{i=1}^m U} \right] \cdot \left[2 \cdot U \cdot \frac{(x_k - p_{ik})^2}{q_{ik}^3} \right] \quad (7)$$

Above-mentioned optimization procedure of parameters that is obtained by the identification procedure uses an effective training methodology. The used learning process is performed in two stages. A clustering algorithm, first of all, finds a course model that roughly approximates the underlying input-output relationship. Then the procedure of

parameter optimization is performed for a better tuning of the initial structure. If an appropriate structure is identified, the learning task can be accomplished by any suitable training algorithm such as the classical backpropagation algorithm.

Above-mentioned procedure can be very effectively implemented within genetic algorithm environment with expert system tool using. The configuration of the solved problem is illustrated in Fig. 4. Some interesting results have been achieved in real machinery equipment conditions experiments.

An approach for on-line parameters identification and measurements via internet technology uses the following designed structure of sentence transmitting, which represents some communication protocol between data server application and client application in Applet tool. We have:

```
//Received string structure
//[dt0.10][n0monitoringCourseName_ymin_vmax]0_51.33_
4:5_22.0
//sample.timenum.graphNameofgraph_ymin_vmaxnum.grap
h_value.
```

A dynamic model of the solved system has been developed using results presented in above-mentioned methodologies. The dynamic simulation of this system is capable of investigating the operation of the process under normal conditions as well as various faulty conditions.

The Most Significant Decisions Indicators are illustrated in Fig.5 (Second methodology).

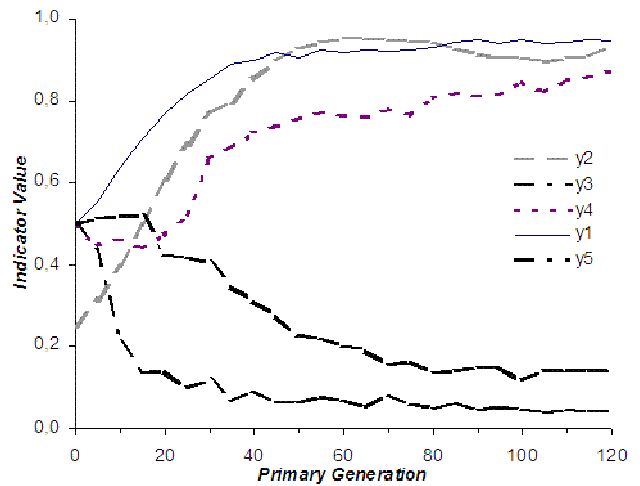


Fig. 5 The most Significant Decisions Indicators:

y1 = crossover units course; y2 = selection method course;
y3 = elitist model course; y4 = mutation function course;
y5 = value replacement course

We compare *Fitness evaluation* for both solved methodologies approaches. Judged input parameters for experimental simulations have been the same for both cases. The result is illustrated in Fig. 6.

C. Self-learning of diagnostic rules experiments

We illustrate the following example of many realised experiments. There are nine measurements and five controller outputs (to the manipulated machinery working places) in the solved diagnostic system. It determines that there should be fourteen symptoms in the condition parts of the rules.

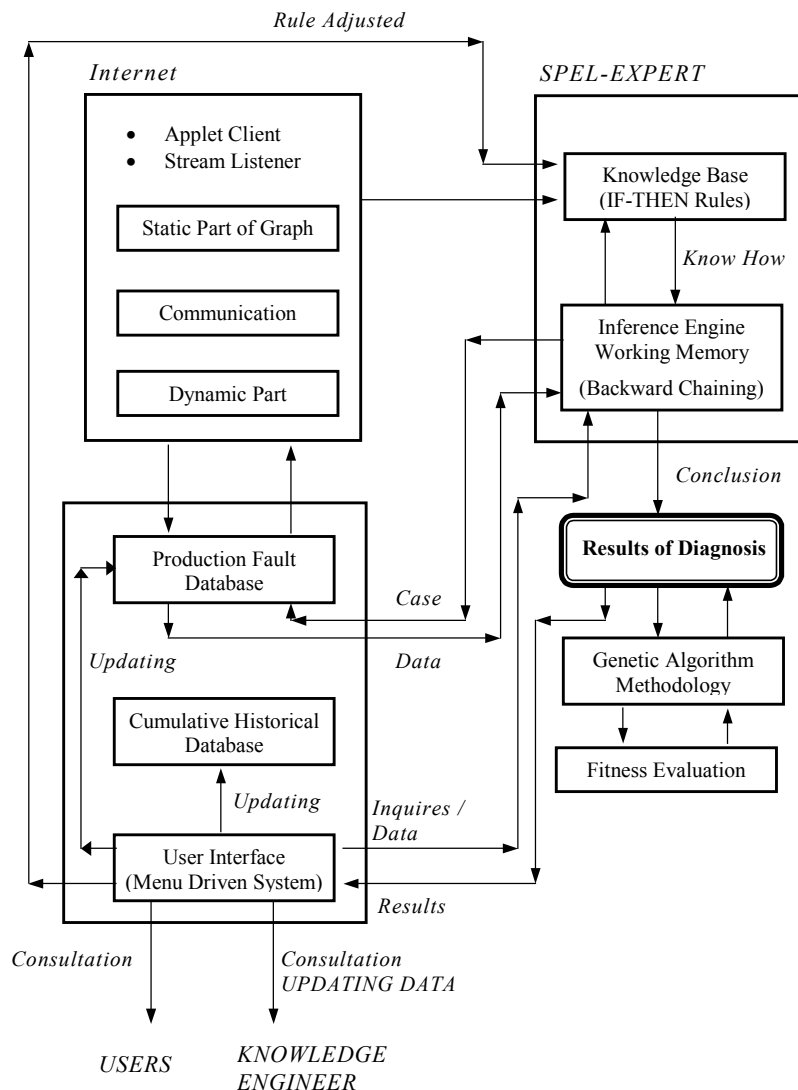


Fig. 4 The configuration of combination of genetic algorithm methodology with expert system approach.

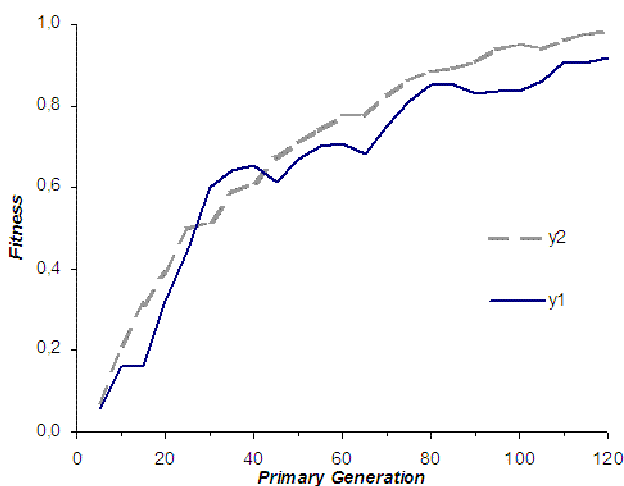


Fig. 6 Fitness Evaluation Course:
 y1 = Fitness curve for the First Methodology
 y2 = Fitness curve for the Second Methodology

Training data are generated for the faults (nine faults have been simulated in spectra vibrations). Training data are divided into ten groups, each containing eighteen sets of data, where one group represents the behaviour of the process under normal operating conditions and the other nine groups

correspond to the nine faults (above mentioned).

The training data are obtained by simulating the process under various faults and with noisy measurements. Corresponding to the nine faults, there are nine groups of rules. Each solved group contains thirty-seven rules. These rules are coded as binary strings for genetic algorithm implementation. There are some relevant parameters:

- the probability of crossover procedure was set to 0.88
- the probability of mutation procedure was set to 0.029
- the parameters used in the fitness function (fitness by the secondary genetic algorithm approach in first described case) were set from 0.003 to 3.16.

The fitness of an individual structure is a measure indicating how fitted the solved 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 performance of the learning system is shown in Table 1. The largest and average fitness of each group at the initial generation and the final generation are provided. It can be seen that in most of the cases significant improvements have been obtained.

Table 1. Largest and average fitness of rules in initial and final generations.

Rule	Initial Generation		Final Generation	
	Largest	Average	Largest	Average
1	0.74	0.20	0.98	0.92
2	0.55	0.05	0.92	0.88
3	0.70	0.12	0.94	0.85
4	0.67	0.14	0.88	0.80
5	0.55	0.11	0.87	0.79
6	0.60	0.10	0.85	0.76
7	0.72	0.13	0.94	0.88
8	0.71	0.25	0.97	0.91
9	0.50	0.14	0.92	0.88

An example of largest and average fitness in typical learning rule for solved diagnostic machinery system is illustrated in Fig. 7.

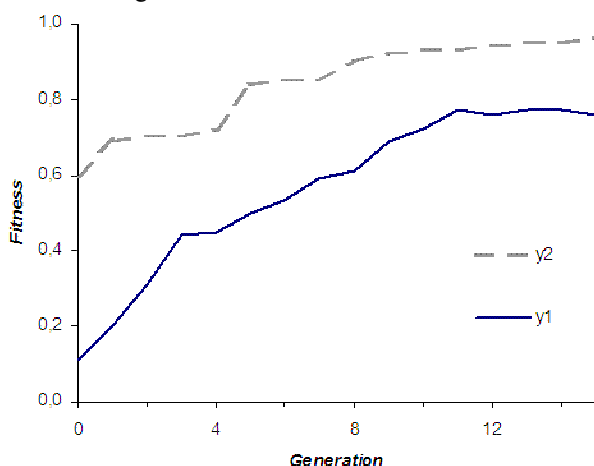


Fig. 7 An example of largest fitness (curve y2) and average fitness (curve y1) in typical learning rule.

The genetic algorithm used here is the three operator approach with some modifications, which are required in cases where the best structure in the new generation (after reproduction, crossover and mutation) is not as good as the best one of the previous generation in preserving the best structure. In such situations, the worst rule in the current generation is replaced by the best rule in the previous generation (we consider a preservation of the best rule). By such means, there will be no such dangers that the best rule ever discovered will be lost.

This mechanism with the strength and specificity rules management can be effectively assimilated to a genetic operator. So it may be interesting to compare this solution with above mentioned genetic algorithm approach.

The learnt rules have been tested on the real process. All simulated faults were successfully diagnosed by the corresponding rules and no incorrect diagnosis occurred.

III. CONCLUSION

The quality of a genetic algorithm approach seems to be dependent on a few important parameters and operator variants. Fitness function, as well as the parameters of the

fitness function, can affect the result of learning process.

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

There is some possible future extending. The fitted structures are selected and combined in a structured yet randomised way to produce more fitted structures, whereas, in a combinatorial search, all the possible structures will be evaluated with the same possibility. From the achieved experimental results, a promising performance of solved genetic learning approach can be expected when it is applied to more complicated tasks.

We solve also genetic algorithm methodology applications to the self-learning of diagnostic rules for industrial processes. Self-learning of diagnostic rules can facilitate knowledge acquisition effort and is more desirable in these cases where certain knowledge is unavailable. The solved genetic algorithm approach is feasible by implementing it in a multi-transputer environment. Performance results for finding the best genetic algorithm for the problem of optimal operation parameters in solved diagnostic system have demonstrated the quality of our implementation.

Future research will be focused first of all on improving the runtime performance of solved implementation, including other genetic operators in the architecture and investigating the results of further test problems in more detail. 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 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 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 research reported on this paper is supported by following scientific project of Slovak Grant Agency VEGA (Ministry of Education): *Intelligent Approach to Automated Diagnostic Problem Solving of Machinery Equipment* (Number: VEGA 1/4133/07).

REFERENCES

- [1] P. Ballé, *Fuzzy Model-based Parity Equations for Fault Isolation*. Control Engineering Practice, No7, 1999, pp.261-270.
- [2] J. Zhang, P.D. Roberts, *On-line Process Fault Diagnosis Using Neural Network Techniques*. Institute of Measurement and Control, 1992.
- [3] D. Montana, L. Davis, *Training Feedforward Neural Networks using Genetic Algorithms*. In Proceedings of 11th Int. Joint Conference on Artificial Intelligence, 1989, pp.762-767.
- [4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, reading, Massachusetts, 1989.
- [5] S. Gallova, *A maximum Entropy Inference within Uncertain Information Reasoning*, in Proc. of Information Processing and Management of Uncertainty in Knowledge-based Systems, Les Cordeliers, Paris, 2006, 1803-1810.
- [6] S. Gallova, *Fault Diagnosis of Manufacturing Processes via Genetic Algorithm Approach*, IAENG Engineering Letters, 2007, 15, Issue 2, 349-355.