DETECTING THE FAULT FROM SPECTROGRAMS BY USING GENETIC ALGORITHM TECHNIQUES

Amin A. E.[1], El-Geheni A. S.[2], and El-Hawary I. A[**]. El-Beali R. A.[3]

[1]Mansoura University, Textile Department
[2]Prof. Dr. Alexandria University, Textile Department
[3]Prof. Dr. Mansoura University, Textile Department

Abstract:

During the last thirty years there has been a rapidly growing interest in the field of genetic algorithms (GAs). The field is at a stage of tremendous growth, as evidenced by the increasing number of conferences, workshops, and papers concerning it, as well as the emergence of a central journal for the field. With their great robustness, genetic algorithms have proven to be a promising technique for many optimisation, design, control, and machine learning applications. This paper presents a new technique for detecting the source of fault in spinning mills from spectrograms by using genetic algorithm.

Key words:

Textile spectrogram, genetic algorithm

1. Introduction

The problem of faults and defects produced during the mechanical spinning of yarns is as old as the industry itself. In earlier times, much skill and experience were required in order to detect and remove these faults.

In this paper, an attempt is made at providing a key to enable the source of faults to be located. It is understood that only the more common defects can be covered here, since there is no end to the list of possible reasons. In any event, there is no complete replacement for skill and experience. An important point to be borne in mind is:

- that machines generally create short term variations, and
- that when ultimately drafted, these appear as long-term or count variations.

Figure 1 illustrates the types of fault in the spinning mill.

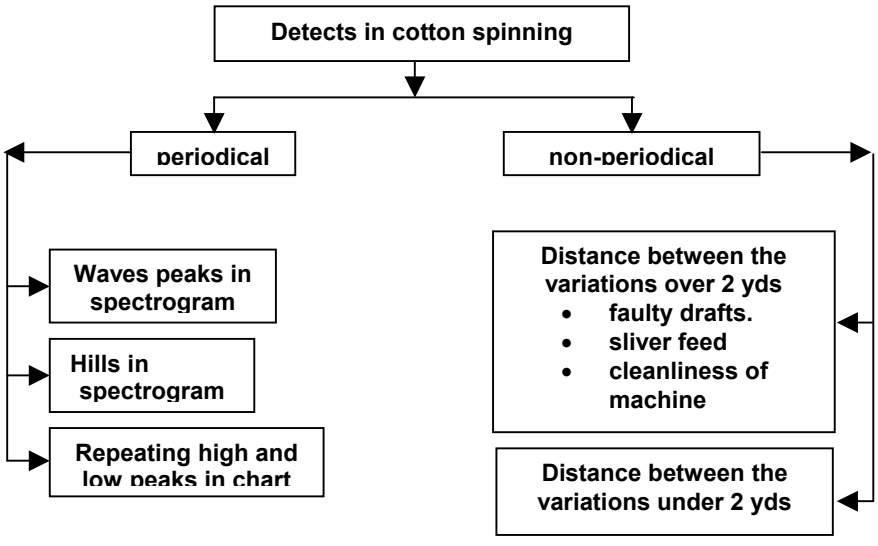


Figure 1. Chart illustrating types of fault.

2. Wavelength range of machine's faults [1]

Table 1 (1.A and 1.B) presents a specification of machine elements and machine sets at which faults may occur, together with the dimensions and formulas related, as well as their positions at the spectrograms.

Table 1.A. Machine parts, schematic drawings and dimensions specification.

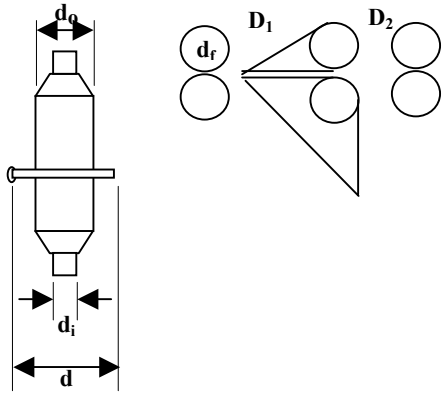
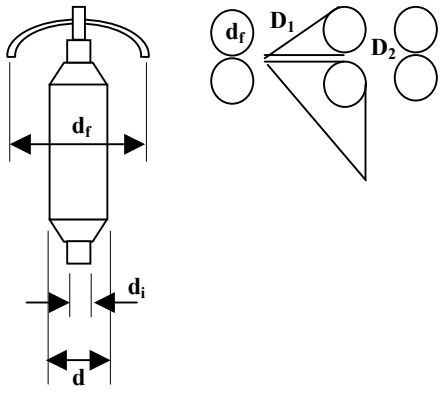
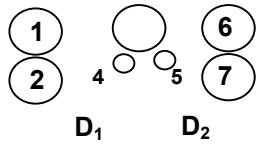
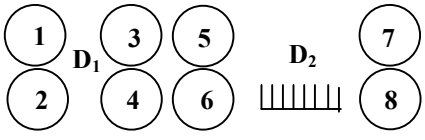
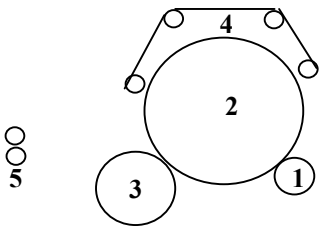
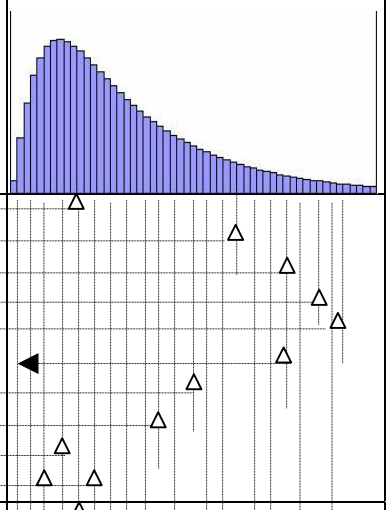
Machine part	Geometrical arrangement of the particular machine elements end sets	Specification of dimensions and other parameters
Ring spinning		d_f = front roller diameter. d_o = outer diameter of bobbin. d_i = inner diameter of bobbin. d_T = traveller diameter. D_1= front draft. D_2= back draft. l1 = length of top apron. l2 = length of bottom apron.
Speed frame		d_f = front roller diameter. d_o = outer diameter of bobbin. d_i = inner diameter o bobbin. d_T = traveller diameter. D_1= front draft. D_2= back draft. l1 = length of top apron. l2 = length of bottom apron.
Draw frame		1,2 = front roller diameter. 3,4,5 = middle roller diameter. 6,7= back roller diameter. D_1= front draft. D_2= back draft.
Comber		1,2 = front roller diameter. 3,4 = middle roller diameter. 5,6 = back roller diameter. 7,8 = detaching roller diameter. D_1= front draft. D_2= back draft.
Card		1 = licker-in diameter. 2= cylinder diameter. 3 = doffer diameter. 4 = lifting-of of flats. 5 = calender roller diameter.

Table 1.B. Machine parts, formulations of wavelengths and positions at the spectrograms.

Machine	Source	Calculation of wave length	
Ring Spinning	Front roller Middle roller Back roller Top apron Bottom apron Drawbox drive Ring traveller Empty spindle Full spindle Drafting waves	$\lambda_1=d_1*\pi$ $\lambda_2=d_1*\pi*D_1$ $\lambda_3=d_1*\pi*D_{total}$ $\lambda_4=l_1*D_1$ $\lambda_5=l_2*D_1$ λ_6 ▶ shorter than λ_3 $\lambda_7=d_T*\pi$ $\lambda_8=d_i*\pi$ $\lambda_9=d_o*\pi$ $\lambda_{10}\sim 2.75*l$	
Speed Frame	Front roller Middle roller Back roller Top apron Bottom apron Drawbox drive Empty spindle Full spindle Drafting waves	$\lambda_1= d_1*\pi$ $\lambda_2= d_1*\pi*D_1$ $\lambda_3= d_1*\pi*D_{total}$ $\lambda_4=l_1*D_1$ $\lambda_5=l_2*D_1$ λ_6 ▶ shorter than λ_3 $\lambda_7=d*\pi$ $\lambda_8=d*\pi$ $\lambda_9\sim 3.5*l$	
Draw Frame II/I	Front roller Front roller Middle roller Middle roller Back roller Back roller Drawbox drive Drafting waves	$\lambda_1= d_1*\pi$ $\lambda_2= d_2*\pi$ $\lambda_3= d_1*\pi*D_1$ $\lambda_{4,5}=d*\pi*D_1$ $\lambda_6=d*\pi*D_{total}$ $\lambda_7=d*\pi*D_{total}$ λ_8 ▶ shorter than λ_6 $\lambda_9\sim 4*l$	
Comber	Front roller Middle roller Middle roller Back roller Drawbox drive Drafting waves	$\lambda_{1,2}=d_{1,2}*\pi$ $\lambda_{3,5,6}=d_{3,4,5}*\pi*D_1$ $\lambda_4=d_4*\pi*D_1$ $\lambda_{7,8}=d*\pi * D_{total}$ λ_9 ▶ shorter than λ_6 $\lambda_{10}\sim 4*l$	
Card	Licker-in Cylinder Doffer Card flats Calender rollers	$\lambda_1 = \frac{d_1*\pi*Vp}{d_1*\pi*Vp}$ $\lambda_2 = \frac{d_2*\pi*Vp}{d_2*\pi*Vp}$ $\lambda_3 = \frac{d_3*\pi*Vp}{d_3*\pi*Vp}$ $\lambda_4 = \frac{l*Vp}{l*n_4}$ $\lambda_5 = d_5*\pi$	

3. Genetic algorithms (GA)

A genetic algorithm (GA) is a probabilistic search and optimisation technique based on the principles of biological evolution, natural selection, and genetic recombination, simulating the survival of the fittest in a population of potential solutions of individuals. GAs are capable of globally exploring a solution space, pursuing potentially fruitful paths while also examining random points to reduce the likelihood of settling for a local optimum [Goldberg, 1989]. They are conceptually simple yet computationally powerful, making them attractive for use in complex domains, and have been demonstrated on a wide variety of problems.

GAs and evolutionary computation, which use the twin metaphors of natural evolution and survival of the fittest, derive their strength by working on populations of solutions and are therefore best exploited in a parallel computing environment.

GAs function by iteratively refining a population of encoded representations of solutions. 'Parents' from a population are 'selected' for 'crossover' and 'mutation'. The 'offspring' form the new population. The 'selection' process is controlled via a fitness measure that is closely related to the optimisation objective. This generation cycle is repeated until a solution with the desirable quality is located.

GA evolution emerges from the following simple rules:

1. Initiate: Create a population of a some specific size.
2. Evaluate fitness: Calculate a fitness value for each individual in the population by evaluating how good solution the individual is to the given problem.
3. Reproduce: Let the best individuals have offspring with each others by choosing two individuals at a time. The probability for choosing an individual is weighted by the individual's, fitness value so that the good ones in the population reproduce many time more likely than the bad ones. The offspring are created by cutting both of the two individuals at the same randomly selected point and then switching their ends, as shown in Figure 2. Crossover is done to a predefined fraction of the population.

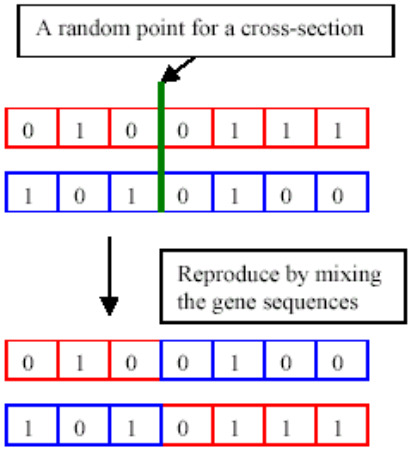


Figure 2. Genetic algorithmic reproduction.

4. Mutate: Choose a small, predefined percentage of the individuals randomly and randomly change one gene in each of them.
5. Copy the best individual of the last generation on one individual in the new generation to prevent possible degeneration.

By repeating these simple rules, the population evolves very fast and finds a good or the optimum solution from huge search spaces by searching only a small fraction of the search space. By searching only 10^{-32} %–fraction of a search space, a GA can still find the optimum solution in more than 10% of the searches [2]. Of course this depends on the structure of the problem. If not the optimum, a GA generally still finds very good solutions.

The author applied the GA to the physically based jumping problem. The system learned to jump much higher than expected in only 50 generations. The population size was 400, mutations were

made to 10% of the population, and only the 10 best individuals were allowed to make offspring. The fitness function used simulated the system with each string of the population, and gave a simple rating relative to how high the system jumped.

In his Siggraph 94 article [2], Karl Sims proposed a technique for creating evolving virtual creatures in a simulated physical environment. He used genetic algorithms for both, creating the bodies of the creatures as well as evolving behaviour for them. The behaviours that he created realised swimming, locomotion, jumping and following a light. The behaviours created were interesting, goal-oriented and life-like; they could not easily be predicted. The GA learns how to use the body it is given by knowing what it wants to accomplish with it (fitness evaluation). The GA basically tries everything the body is capable of, and improves the ways that lead towards the goal (better fitness value).

GAs are a very good tool for finding solutions from huge search spaces. They are good for many problems such as scheduling, DNA fragment assembly, edge colouring, minimum set-covering, 'travelling salesman' and other NP-hard (Non-deterministic Polynomial-time hard) problems [3]. NP-hard problems are solvable in polynomial time. Usually this means that these problems grow in complexity so fast, as a function of the order of the problem, that no matter how much the computers evolve, the problems cannot be solved with brute-force methods. An example of this is the chess search problem. With a problem branching factor of 35, the problem grows so fast as the function of the planning depth that the exponential evolving speed of computers does not help much in essentially deeper planning.

In relations to computer games, GAs embody one challenging problem. They search for solutions to a static problem, one that remains the same. The GA cannot iterate to find a maximum if the systems' behaviour changes between generations. Thus, one cannot really make the GA learn while playing against the human user, as the human user does not act in the same way every time.

However, there are many things, game AI wise, that can be done with GAs. For an example, one can make a population of individuals with different FSM behavioural probability distributions and rate their success as a fitness value. The best ones can then reproduce as the game proceeds. This would lead the game AI to slowly improve against the particular player. The choice of the fitness function is essential to how well a GA evolves.

3.1. Genetic algorithm procedure

Figure 3 explains how the GA can be used to obtain the best chromosome which maximizes $\psi_k(\chi_i)$.

Begin
{
Assume the fitness function as $\psi_k(\chi_i)$
Create a chromosome structural as follows;
{
Generate a number of slots equal to I, which represent input vector X.
Put a random value 0 or 1 in each slot.
}
G=0, where G is the number of the generation.
Create the initial population, P of T chromosomes, $P(t)^G$, where t=1 to T.
Evaluate the fitness function according to $P(t)^G$.
While termination conditions not satisfied, Do
{
G=G+1
Select number of chromosomes from $P(t)^G$ according to the roulette wheel procedure.
Recombine between them using crossover and mutation.
Modify the population from $P(t)^{G+1}$ to $P(t)^G$.
Evaluate the fitness function according to $P(t)^G$.
}
show the best chromosome which satisfies the conditions
}
End

Figure 3. Genetic algorithmic procedure.

For extracting a rule belonging to class K, the best chromosome must be decoded as follows:

- The best chromosome is divided into N segments.
- Each segment represents one attribute, A_n(n=1,2,….,N), and has a corresponding bit length m_n which represents their values.
- The attribute values are existed if the corresponding bits in the best chromosome equal one and vice versa.
- The operators OR and AND are used to correlate the existing values of the same attribute and the different attributes respectively.

The overall methodology of rule extraction is shown in Figure 4.

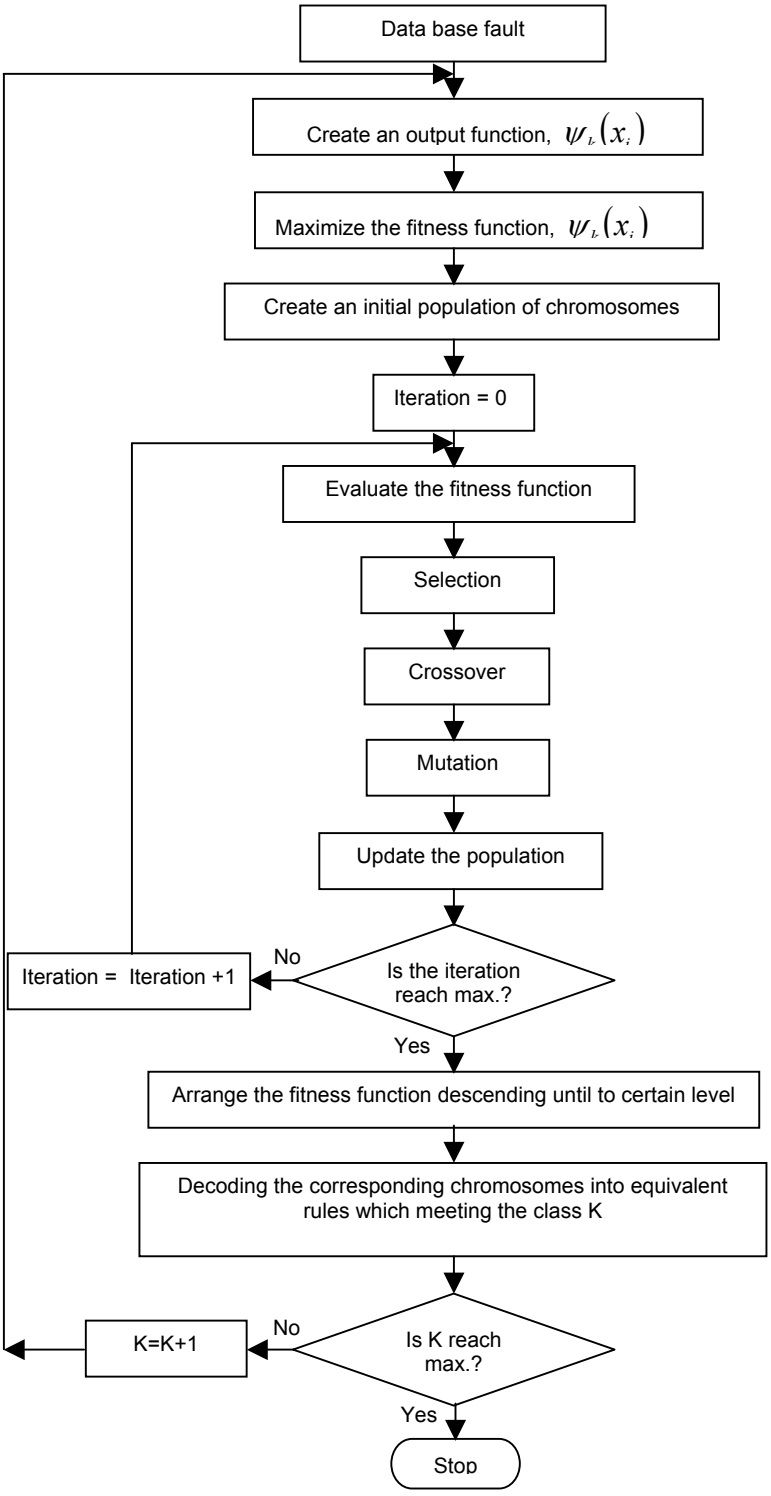


Figure 4. The overall methodology for rule extraction.

4. Illustrative example

The authors applied this technique in a virtual draft system simulation model [VDSS] [4]. The VDSS was given a database with 4 attributes and one output class, as shown in Table 2. The linguistic values of each attribute and the output class are summarised as follows;

Fault type = A_1 = (False Real)
Spectrum shape = A_2 = (Mechanical Draft waves)
Fault category = A_3 = (periodical Non periodical)
Harmony = A_4 = (odd even odd & even Non)
Fault source = A_T = (Ambient Coiling Drafting)

Table 2. Example for target concept draft system fault.

Wave length	Harmony	Source zone	Defect Source
0.12 - 0.25	no	no	Ambiant condition
0.6 - 1.2	no	no	Ambiant condition
0.12 - 0.25	no	no	Ambiant condition
0.12 - 0.25	no	no	Ambiant condition
0.1 - 0.15	no	no	
0,17	no	coiling	
0,16	no	coiling	
0,94	no	coiling	
0,16	no	coiling	
0,16	no	coiling	
0,165	no	coiling	
0,13	even	front draft	1st bot roller in draft zone
0,13	even	front draft	1st top roller in draft zone
0.12 - 0.15	no	front draft	1st top roller in draft zone
0.12 - 0.15	no	front draft	1st top roller in draft zone
0.08 - 0.1	no	front draft	2nd top roller in draft zone
0.08 -0.1	even	front draft	2nd top roller in draft zone
0.2 - 0.3	even	middle draft	
0.2 - 0.3	even	middle draft	
0.2 - 0.3	no	middle draft	
0.4 - 0.6	no	middle draft	
0.9 -0.95	no	back draft	
1 - 1.05	no	back draft	
0.9 - 0.95	even	back draft	
1 - 1.05	even	back draft	
0.06 - 0.23	no	main draft	
0.06 -0.23	even	main draft	
0.09 - 0.35	no	break draft	
0.09 - 0.35	even	break draft	
0.05 - 0.1	no	front draft	
0.05 - 0.1	no	front draft	
0.05 - 0.1	no	middle draft	
0.05 - 0.1	no	back draft	

The encoding values of the given database are shown in Table 3. Table 4 shows the result from the expert database by using the genetic algorithm (GA).

Table 3. Encoded database.

Fault Type	harmony	Source zone	Defect Source																			
			Draft Zone													coiling zone					Ambiant condition	other
			Roller							Distance		Pressure				Driver		coiling part		Guide rollers		
			top				bottom															
			1st	2nd	3rd	4th	1st	2nd	3rd	D1	D2	1st	2nd	3rd	4th	Can	Coiler	Can	Coiler			
periodic	no	no	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	no
periodic	no	no	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	no
periodic	no	no	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	no
periodic	no	no	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	no
periodic	no	no	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	no
periodic	no	coiling	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	no
periodic	no	coiling	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	no
periodic	no	coiling	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	no
periodic	no	coiling	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	no
periodic	no	coiling	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	no
periodic	no	coiling	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Calender roller
periodic	even	front draft	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	even	front draft	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	front draft	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	front draft	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	front draft	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	even	front draft	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	even	middle draft	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	even	middle draft	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	middle draft	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	middle draft	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	back draft	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	back draft	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	even	back draft	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	even	back draft	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	main draft	0	0	0	0	0	0	0	narrow	0	0	0	0	0	0	0	0	0	0	0	no
periodic	even	main draft	0	0	0	0	0	0	0	great	0	0	0	0	0	0	0	0	0	0	0	no
periodic	no	break draft	0	0	0	0	0	0	0	0	narrow	0	0	0	0	0	0	0	0	0	0	no
periodic	even	break draft	0	0	0	0	0	0	0	0	great	0	0	0	0	0	0	0	0	0	0	no
periodic	no	front draft	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	no
periodic	no	front draft	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	no

Table 4. Extracted ruler from expert database.

Rule No.	Fitness	Xi vector from GA										Directly Extracted Rule
		A1		A2		A3		A4				
		X1	X2	X1	X2	X1	X2	X1	X2	X3	X4	
1	0,99988	1	0	0	1	1	0	0	0	0	0	if Fault type is false And Spectrum shape is draft waves Then fault source is ambient condition
2	0,99995	0	1	1	0	1	0	0	1	0	0	if Fault type is real And Spectrum shape is mechanical And fault category is periodical and harmony is even Then fault source is drafting zone

5. Conclusion

This paper studies AI techniques for detecting the spinning fault resource. This technique involves extracting the fault resource from the expert database by using genetic algorithms. The expert

database consists of spectrogram specification such as fault category (periodic OR non-periodic), fault type (false OR real), spectrogram shape (mechanical OR Drafting wave), … etc.

References:

1. *Application handbook for evenness testers of the Uster type; Determination of periodic mass variations (spectrum).*
2. *Karl Sims, Evolving Virtual Creatures, http://www.genarts.com/karl/papers/siggraph94.pdf (7.2002).*
3. *Sami Khuri, 'Genetic Algorithms', course handouts, Helsinki University of Technology, August 2002.*
4. *Amin. A. E., 'Development of a simulation draft system model applied in virtual educational environment', under publication.*

∇Δ