Planning the Spinning Process by Means of Neural Networks and Genetic Algorithms

F. Pynckels[+], S. Sette[*], L. Van Langenhove[*] and K. Impe[**]

[+] *Ministry of Finance, Treasury Department, Kunstlaan 30, 1040 Brussels, Belgium*
[*] *Department of Textiles, University of Ghent*
[**] *Ministry of The Flemish Community*

In two previous papers, a description was given how the spinnability of a given fiber quality on a rotor and ring spinning machine can be predicted with a reliability of 95% by means of a neural network, and how the yarn properties can be deducted from fiber properties and spinning machine settings with an accuracy of over 95%. This paper goes further. It describes how one can decide which fiber types and spinning machine settings are to be used to create a yarn with given characteristics. In other words, a method is described how to decide in which way a certain type of spinning machine must be set and what fibers must be used to spin a yarn with properties determined beforehand. The accuracy of the described method is surprisingly good, and a lot better than the currently used methods.

1. INTRODUCTION

The aim of the present study (which falls within the framework of a BRITE/EURAM project BREU0052) consists in developing a method that predicts the fiber properties and production conditions to use during the spinning process to obtain a yarn with given characteristics. The conditions are determined by the type of machines and their settings. By means of a neural network, as described in *Pynckels (1995)*, it is possible to determine the spinnability of fibers under given production conditions.

Once spinnability is assured, one can predict the yarn properties by means of multiple interconnected neural networks. As mentioned in *Pynckels (1995)*, statistical methods as for instance multiple linear regression are not suited to resolve this problem since they assume in advance that the factors are independent and linear. Both assumptions are false in the case of a spinning process. In other words, a method is needed that does not oversimplify the problem to reduce the mathematical complexity of the solution. It is especially for this kind of problems that neural networks have proven to be a valid alternative as described in *Ramesh (1995)*.

Once the spinning process can be simulated as *Pynckels (1997)* shows, one can construct a method that permits to optimize the input parameters of the spinning process simulation (fiber properties and production conditions) to obtain given yarn properties. Again, the classic mathematical methods very soon become very complex, or have the tendency to oversimplify the problem. In this case, genetic optimization algorithms give very good results, and hence will be used in the method described in this paper.

2. TEST SET-UP

First, a spinning process simulation had to be constructed. This was done as *Pynckels (1997)* describes:

> Twenty cotton types were selected based on their divergent properties. From the seventy-three important properties, a selection of 35 representative parameters was made using statistical methods. Afterwards, another 21 parameters were eliminated by means of neural network methods. The 14 fiber properties and the 5 machine parameters studied, together with the 9 yarn properties to predict are to be found in *Pynckels (1997), table I, table II and table III*. Yarn of 25, 30 and 50 tex was spun with twist factor α_{tex} of 3500, 4000 and 4500. Ring spinning was done on an SKF lab ring spinning machine. Rotor spinning was carried out on a Platt rotor spinner. For a total of 1382 spinnable cases, the resulting yarn properties were measured, and were kept, together with the corresponding fiber properties and machine parameters. Several interconnected neural networks were used to simulate the spinning process. To assess whether the used method can simulate the spinning process, and whether the methods accuracy level was acceptable, the data of the 1382 spinnable cases was used.

Once the spinning process simulation was available, yarn characteristics were selected. The optimization method was used to obtain *'optimal'* fiber properties and production conditions. These were fed back to a real production process to verify the accuracy of the used method.

3. METHOD

3.1 Introduction

The aim of the designed method is to predict the fiber qualities and the production conditions to use so that the spinning process will result in a yarn with properties that are given in advance. Since *Pynckels (1997)* and *Sette (1996)* describe methods to simulate the spinning process, the problem can be stated as:

> Given is a spinning process simulation function
>
> $$S : \Re^n \rightarrow \Re^m : \{I_i\} \rightarrow \{Y_j\}$$
>
> with i ∈ {1..n} where n is the sum of the number of fiber properties taken into account and the number of considered production conditions (say spinning machine settings), and with j ∈ {1..m} where m is the number of yarn properties that are predicted by the spinning process simulation. Further in this paper, the notation I will be used for the input vector $I(I_1, \dots, I_n)$.
>
> Given is also a vector $Y(Y_1, \dots, Y_m)$ that contains yarn properties that describe the desired yarn quality.
>
> If Y' is defined as $Y' = S(I)$ then find a I so that $\|Y - Y'\| \leq \varepsilon$ where ε is a given threshold value.

The method proposed in this paper will find such a vector I by using a genetic method. This principle is clarified in figure 1.

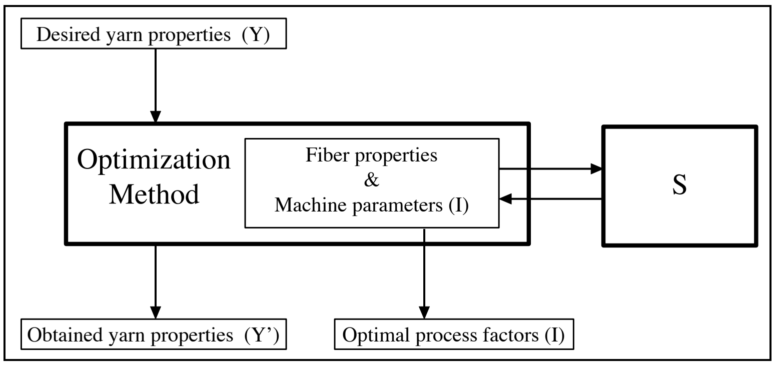
Desired yarn properties (Y)
Optimization Method
Fiber properties & Machine parameters (I)
S
Obtained yarn properties (Y')
Optimal process factors (I)

Figure 1: Optimization method

The optimization method gets a vector Y as input. Starting from this Y an estimation for the vector I (fiber properties and machine parameters) is calculated. This is done by feeding the vector repetitively to the spinning process simulation function S that calculates a corresponding vector Y' (yarn properties for input vector I). The vector Y' is compared with Y. If the difference is too large (larger than a chosen ε) and the number of optimization recursions has not exceeded a certain predetermined number, another estimation for I is decided upon and another recursion step is taken. After a certain number of recursions, a vector I is found that generates an acceptable approximate vector Y' for Y. The final vector I for which $Y' = S(I)$ is the result of the optimization process and can be used in a real spinning process. In our case this is done to cross-check the results of the optimization method, in a production environment, the vector I is used to start spinning the necessary quantities of a yarn with the demanded properties.

3.2 Genetic optimization

3.2.1 Introduction

To describe the method used to optimize the spinning process as shown in figure 1, a general description of *genetic algorithms* is given. Understanding this general description will help to see how it is applied to the specific spinning process optimization later in this paper.

The fundamental point of view of a genetic optimization method is that in nature the law of *the survival of the fittest* exists. Since each organism is generally spoken different than its parents, it will be better adapted to its environment, or less adapted for that matter. It is also reasonable to suppose that the environment itself is continuously changing. Only those organisms that are adapted to the environment they live in can survive. The well adapted organisms are also more likely to produce better adapted children, since their genetic material and its variations are closer to the ideal genetic pattern for the target environment. In that way, a race is created that is continuously evolving towards a better adaptation to the environment (the others can't survive in the long run). The ability to survive and improve (taken the environment into account) is called the *fitness* of the organism. It's clear that in nature, fitness can't be expressed by means of a single real number.

But how does an organism become more fit than its ancestors? Simply put: it must be lucky. Reproduction of organisms is a question of random reorganization of the genes of the parents. This means, as already stated, that the reorganized genes can be better suited to survive in a certain environment, but this is not always the case. The random reorganization is done in more than one way. Mostly by *reproduction* of the original genes and by *crossover* (part of the genes is taken from one parent, and the other part of the genes are taken from the other parent), and in some cases by *mutation* (the reproduction of the genes shows some ‘deficiencies’ that can purely by accident make the organism more fit).

A second thing to keep in mind when looking at the evolution of organisms is the fact that most of them feel comfortable in groups, as long as the groups do not become overcrowded. Once this is the case, a larger mortality is observed. One could say that the fitness of the individuals becomes smaller once the group they live in becomes too populated.

3.2.2 Mathematical implementation

So, where does all of this fit into the mathematical theory of optimization? Suppose that for some function $F : \Re^n \rightarrow \Re^+$, the absolute maximum must be found. In other words, a vector $I(I_1 , \dots , I_n) \in \Re^n$ must be found for which F(I) is larger than for all other vectors in $\Re^n$. The function F can be seen as a *fitness function* that expresses how fit a certain organism $I(I_1 , \dots , I_n)$ is within an environment. The values $I_1 , \dots I_n$ can be seen as one long string of values that characterize the organism (the *characteristic string*). In other words, its genes. So, by manipulating the genes from one or more organisms, new next-generation organisms can be created. Some of them will be fitter within the boundaries of the stated environment (function F) than others. By iterating through successive generations of organisms, whereby the fittest species are kept, and the weakest species are eliminated, one creates a population of organisms that come closer and closer to the points where the fitness function has an absolute maximum. The reader must know the fitness function can be time-dependent (changing environment).

The next problem to aboard is the translation of the vector I to a characteristic string. This can be done in many ways. An obvious method is to remap the vector I from $\Re^n$ to $\{0,1\}^N$. It should be clear that N must always be equal to or larger than n because of this *binary remapping*. However, there is always a certain amount of accuracy that gets lost, no matter how large N is chosen. For example, in most cases each value I_i can be remapped to a value between 0 and 255, those values in turn represent all possible 8-bit strings. Remapping the vector I this way would result in N = 8n. The resulting characteristic string is a binary string that can be easily manipulated by the *genetic operators* that are explained below.

The last problem to handle is the mathematical representation of the genetic inheritance, called reproduction, crossover and mutation. And the mathematical way to express that crowded populations become less fit. This is done by the already mentioned genetic operators that take one or more base characteristic strings and transform these into one or more new characteristic strings. Each of the mentioned genetic operators is explained in more detail in the next paragraphs.

3.2.3 Genetic operators

The discussion of the genetic operators used by the method, as described in this paper, is based on the representation of the organisms (I vectors) as N-bit strings. These strings are created as mentioned above. It should also be clear that one can invent all kinds of genetic operators, once the ‘modus operandi’ of such an operator is understood. In most cases however, the genetic operators described below will do just fine.

3.2.3.1 Reproduction operator

The reproduction operator, also called *selection operator*, simply copies the characteristic string of a member of a generation to a member of the next generation. The use of the reproduction operator is directly dependent on the fitness of a member of the population. In other words, the fitter a member, the higher the probability that its characteristic string will be reproduced into the next generation. A possible implementation of the reproduction operator is the *weighted roulette wheel algorithm*. This algorithm is clarified in figure 2. A roulette wheel is devided in as many sectors as there are members in the original population. The surface of each sector is proportional to the relative fitness (fitness/total population fitness) of the corresponding organism. A new population is generated by spinning the wheel as many times as there are members in the original population, and by using the characteristic string of the organism to which the sector where the wheel stops corresponds. Figure 2 shows the possible creation of a new generation. The organisms are described by a 4-bit characteristic string. In this case, the organism C did not survive in the environment described by fitness function F. Organism A on the other hand was duplicated two times.

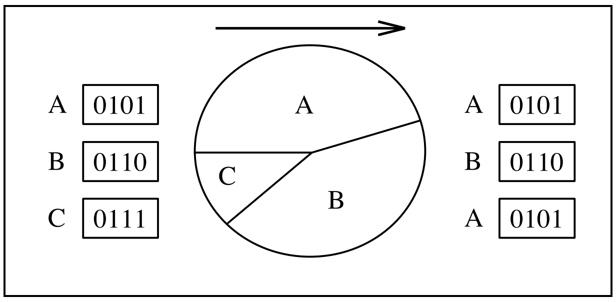

Figure 2 : Reproduction operator

3.2.3.2 Crossover operator

The crossover operator takes two members of the original generation, and creates two members of the next generation. The original members are chosen at random from the base population. One decides at which position the characteristic strings are cut in two. The respective two parts of both organisms are then joined with the other part of the characteristic string of the other organism. In figure 3 a crossover operation is shown between two 6-bit characteristic strings. The *cut* is chosen between the second and the third bit.

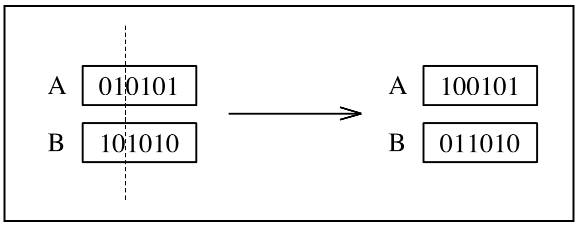

Figure 3 : Crossover operator

3.2.3.3 Mutation operator

The mutation operator is, as already mentioned, a less used operator. In reality, the possibility on a mutation is 1/30000. For the sake of convergence, this proportion is somewhat altered in the mathematical implementation, where 1/1000 seems a good proportion. The principle stays the same however. Some bits are chosen at random from all characteristic strings, and are binary reversed (0 becomes 1, 1 becomes 0).

Although this operator is used much less frequently than the reproduction and the crossover operator, its role is certainly not unimportant. Since the two previous operators can result in the loss of important information by transforming significant characteristic strings into worthless strings, a method is needed that randomly introduces new members in the next generation. This is exactly what the mutation operator does: it prevents a mathematical or genetic stall (no further convergence) or divergence.

3.2.4 Sharing function

The other problem to face when developing a genetic algorithm is the fact that the groups the organisms live in should not be to crowded. Mathematically, this means that we want to prevent that only one optimum is found if multiple optima exist. This is done by the so-called *sharing function*. This is a function that is superposed on the original fitness function, so that the fitness of an organism is reduced as more organisms come closer towards it. This means that, even if organisms approach an optimum, their fitness can be reduced. The consequence of this reduction is that several organisms will reproduce, crossover or mutate towards another optimum if one exists, or towards a variant that will not survive in the long run if no other optimum exists.

The superposition of the sharing function on the original fitness function can be done as *Sette (1996)* describes. Suppose that F(I) is the original fitness function, and that we define a distance function $d : \Re^n \times \Re^n \rightarrow \Re^+$ that calculates the distance between two characteristic strings. Suppose also that $sh : \Re^+ \rightarrow \Re^+$ is the sharing function that is discussed above. Then a new fitness function $f : \Re^n \rightarrow \Re^+$ can be defined as follows:

$$f(I) = \frac{F(I)}{\sum_J sh(d(I,J))} \qquad \forall\, J \in \{characteristic\ strings\}$$

In other words, to calculate f(I), the distance between the considered characteristic string $I(I_1, \ldots, I_n) \in \Re^n$ and all other characteristic strings $J(J_1, \ldots, J_n) \in \Re^n$ is taken into account.

Sette (1996) proposes to use the taxi-metric as distance function, and chooses a linear decreasing share function with function values 1 respectively 0 at the lower and upper boundaries of its domain.

3.2.5 Putting it all together

So, how does all the above fit into the big picture of finding the ideal fibers and machine parameters to spin a fiber with given characteristics. In this paragraph, an algorithm will be discussed that does just that. But first, it can be helpful to review all the components that play a role in the stated algorithm, together with their domain, range and function description.

As mentioned before, two types of vectors are used in the mathematical representation of a genetic algorithm, the characteristic strings (I) and the desired, respectively obtained yarn characteristics (Y and Y'):

$$I \in \Re^n$$
$$Y \in \Re^m \quad and \quad Y' \in \Re^m$$

Also described above are the functions used in the algorithm. These are the spinning process simulation function S, the fitness function F, the sharing function sh, a distance function d and the extended fitness function f. These functions can be described as follows:

$$S : \Re^n \to \Re^m : I \to Y'$$

$$F : \Re^n \to \Re^+ : I \to e^{-\frac{\|Y'-Y\|^2}{\sigma^2}} \qquad met\ \sigma \in \Re_0$$

$$d : \Re^n \times \Re^n \to \Re^+ : (I,J) \to \|I - J\| \qquad \forall\, I,J \in \Psi_t$$

$$sh : \Re^+ \to \Re^+ : x \to \alpha^{-x} \qquad met\ \alpha \in \Re\ en\ \alpha > 1$$

$$f : \Re^n \to \Re^+ : I \to \frac{F(I)}{\sum_J sh(d(I,J))} \qquad \forall\, J \in \Psi_t$$

It should be clear that the functions F, d, sh and f can be defined in another way. The above-mentioned formulas however produce satisfactory results. To describe the algorithm, we need one last notation: the collection of all characteristic strings at a certain moment t. The notation used for this collection is Ψ_t .

Given the above notations and formulas, and keeping the biological principles and their mathematical representation in mind, it is not difficult to grasp the following algorithm:

1. Fill Ψ_0 (being Ψ_t with t=0) with a chosen number of random characteristic strings
2. Choose a ε that represents the maximal approximation error that will be tolerated between the desired vector Y and the obtained vector Y'
3. Check the spinnability of all elements of Ψ_t . Replace all not-spinnable elements with random characteristic strings that are spinnable.
4. Calculate f(I) for all vectors I belonging to Ψ_t .
5. If $\|S(I) - Y\| < \varepsilon$ for the fittest vector I (the one with the largest f(I)), then continue with step 8.
6. Take the fittest vectors I from Ψ_t and use the genetic operators to create a set Ψ_{t+1} starting from these vectors.

7. Go further with step 3 until the number of executed iterations becomes larger than a chosen value Δ. In that case, no acceptable approximation can be found for the conditions ε and Δ.
8. Choose the fittest vector I from Ψ_t and calculate Y' = S(I). The chosen vector I is the optimal production process vector (fiber properties and machine settings), the vector Y' is the resulting yarn that can be expected.

It goes without saying that the result, obtained from the previous algorithm, is an ideal vector I, that will not always be available in the real world. This problem can arise if for instance the type of fibers are not available, or simply do not exist. If this is the case, a solution must be found. The obvious solution is the following: order all vectors in Ψ_t in descending order of fitness. Choose the first vector I' that approximates the available fibers (or combination of fibers, using commonly known mixing formulas) and machine settings. If for this vector the value $\left\| S(I') - Y \right\|$ is still acceptably small, a valid alternative for the ideal I is found.

A second, more correct method is described in paragraph 3.3. Other solutions are mentioned in paragraph 5, and are situated in the fields of fuzzy logic and biochemical genetic manipulation of the growth process. Further research must be conducted in these directions.

3.3 Further optimization

Till now, a method was described to find an optimal set of fiber characteristics and machine settings, that can generate a yarn with given properties. The next question is how the optimal set of fiber characteristics can be approximated by a mixture of existing fibers with known characteristics. The generally used formulas to calculate the resulting fiber characteristics, given the characteristics of the fibers that are mixed are (depending on the fiber characteristic) :

$$M_i = \frac{\sum_j c_j F_{ij}}{\sum_j \mathrm{c_j}} \quad \text{or} \quad \frac{1}{M_i} = \frac{\sum_j \frac{c_j}{F_{ij}}}{\sum_j \mathrm{c_j}}$$

where M_i is the ith characteristic of the resulting fiber-mix, F_{ij} is the ith characteristic of fiber quality j and c_j is the mixture coefficient of fiber quality j.

These formulas can be used as an extra part (a pre-filter) of the S function in figure 1. Doing so changes the optimization method to the one shown in figure 4. This figure shows that instead of optimizing the fiber properties and process factors, the mixture quantities and process factors are optimized. Doing so does not change anything to the described method. The only thing that must be considered is that the spinning simulation function S must be preceded by the mixture formulas (MF) that transform the mixture quantities in fiber properties.

Other methods to optimizing the mixture quantities are not ready for publication at this moment, but the conclusion already mentions some possibilities that can be seen as further research topics

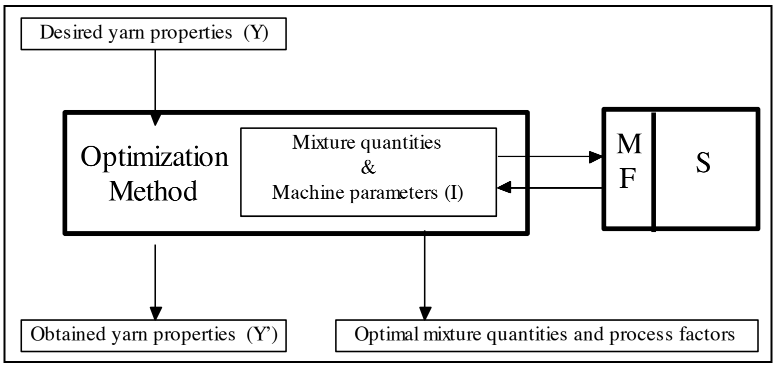
Desired yarn properties (Y)
Optimization Method
Mixture quantities & Machine parameters (I)
M F
S
Obtained yarn properties (Y')
Optimal mixture quantities and process factors

Figure 4 : Extended optimization method

4. RESULTS

Since the primary goal of this paper is to give a comprehensive explanation of the described method, the discussion of the results will be focused on the following problem: prediction of fiber characteristics and spinning machine settings to obtain optimal tenacity and elongation. Figure 5 shows the population of points and their respective predicted elongation and tenacity.

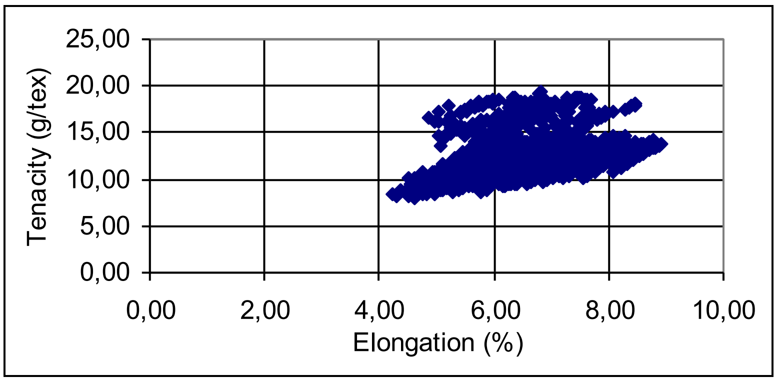

Figure 5 : Genetic algorithm population

To obtain an optimal population member, Pareto optimization was used. The optimal yarn was chosen from the Pareto front (elongation 8.46 and tenacity 18.08). A fiber quality was selected which was as close as possible to the predicted optimal fiber properties. The calculated mixture coefficients for the 20 distinct fiber qualities that were available is shown in figure 6.

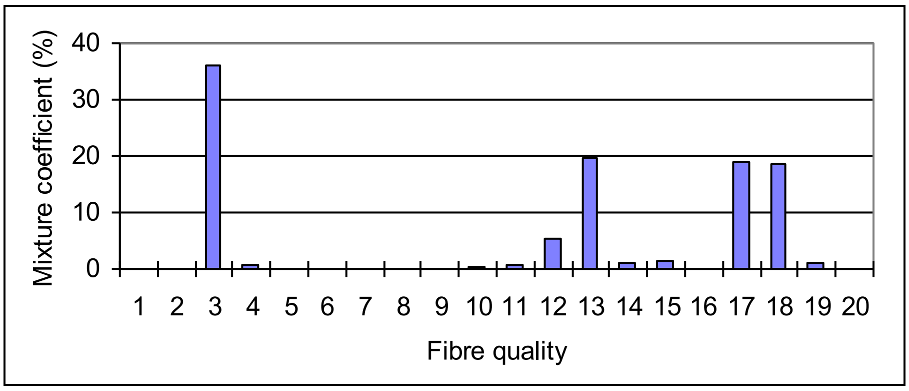

Figure 6 : Mixture coefficients for available fibre qualities

The optimal yarn was constructed using the predicted machine settings and using a simplified mixture of available fiber qualities (40% Q3, 20% Q13, 20% Q17, 20% Q18). The measured elongation is 7.7% (absolute error 0.76%) and the obtained tenacity equals 17.7 g/tex (absolute error 0.38 g/tex). An even better result can be expected in those cases where a more exact fiber quality mixture is used, and certainly in those cases where a more versatile number of fiber qualities will be available. Further tests are planned soon and will be available on demand.

5. CONCLUSION

It can be stated that combining neural networks and genetic algorithms results in a method that can predict what fiber types and what machine settings to use, so that the spinning process will result in a yarn with properties that are given in advance. It goes without saying that the given figures must be interpreted with the necessary caution, since although many tests were carried out, they were done on a very small scale. The method as described in this paper can be applied to the optimization of a spinning process using rotor or ring spinning machines. Therefore, the networks used in the method must be trained with a series of data on fiber properties, machine settings and yarn specifications. Although this means extra work, it doesn't change anything to the given method.

Further research is conducted in the field of coupling the described method with fuzzy expert systems. The final goal is to be able to decide more accurately which fiber qualities and quantities to choose from an available number of qualities and quantities to approximate the *'optimal'* fiber quality so that, although no exact match of fiber properties is available, the resulting yarn properties will still be as close as possible to the desired yarn properties.

Another topic of further research is the approximation of the desired fiber qualities by means of genetic manipulation of the growth process. In this way, a closer match of the *'optimal'* fiber qualities should be obtainable. The growth process depends on several controllable conditions, but then again, also on a few uncontrollable conditions (such as the weather, exact condition of the soil, …). The authors are convinced that, as is the case for the fiber to yarn problem, the controllable conditions to arrive at a desired fiber quality can be decided upon by use of a similar method as the one described in *Pynckels (1995), Pynckels (1997)* and this paper, considering that the uncontrollable conditions have an empirically proven statistical behavior (distribution of rainfall, …). A question that arises here is if statistical neural networks can be of any use in this context.

REFERENCES

Cheng (1995) L. Cheng and D.L. Adams. 'Yarn Strength Prediction using Neural Networks, Part I : Fiber Properties and Yarn Strength Relationship', *Textile Research Journal*, Volume 65, Number 9, 1995, pp. 495-500

Klimasauskas (I) C. Klimasauskas and P. Guiver. *Neural Networks-I (revision 2.00)*, NeuralWare Inc.

Klimasauskas (II) C. Klimasauskas and P. Guiver. *Neural Networks-II (revision 2.00)*, NeuralWare Inc.

Lippman (1987) R.P. Lippmann. 'An Introduction to Computing with Neural Networks', *IEEE ASSP Magazine*, 1987 April, pp. 4-22

Parker (TR-47) D.B. Parker. 'Learning Logic', *Report TR-47*, Cambridge, Massachusetts Institute of Technology, Center for Computational Research in Economics and Management Science.

Pynckels (1995) F. Pynckels, P. Kiekens, S. Sette, L. Van Langenhove and K. Impe 'Use of Neural Nets for Determining the Spinnability of Fibers', *Journal of the Textile Institute*, Volume 86, Number 3, 1995, pp. 425-437

Pynckels (1997) F. Pynckels, P. Kiekens, S. Sette, L. Van Langenhove and K. Impe 'Use of Neural Nets to Simulate the Spinning Process', accepted for publication in *Journal of the Textile Institute*

Ramesh (1995) M.C. Ramesh, R. Rajamanicham and S. Jayaraman. 'The prediction of Yarn Tensile Properties by using Artificial Neural Networks', *Journal of the Textile Institute*, Volume 86, Number 3, 1995, pp. 459-469

Sette (1996) S. Sette, L. Boullart, L. Van Langenhove and P. Kiekens. 'Optimizing the fiber-to-yarn production process with a combined Neural Network/Genetic Algorithm approach', *Jtextile Research Journal*, Volume 67, Number 2, 1997, pp. 84-93

ACKNOWLEDGEMENTS

The authors would like to thank the Commission of the European Community for its financial support in the research, which falls within the framework of a BRITE/EURAM project on the setting-up of an expert system for the cotton spinning mill. We would also like to thank the laboratories of the Institut Textile de France-Section Nord, Trützschler, Harlander Coats and Zellweger for their help in the analysis of the fibers.