Evolutionary Programming to Adjust the Process in High Precision Foundries

Javier Nieves, Igor Santos and Pablo G. Bringas
S^3Lab, DeustoTech - Computing, University of Deusto, Bilbao, Spain
Email: {jnieves, isantos, pablo.garcia.bringas}@deusto.es

Abstract*—Foundry is one of the activities that has contributed to evolve the society; however, the manufacturing process is carried out in the same manner as it was many years ago. Therefore, several defects may appear in castings when the production process is already finished. One of the most difficult defect to detect is the *microshrinkage*: tiny porosities that appear inside the casting. Another important aspect that foundries have to control is the attributes that measure the faculty of the casting to withstand several loads and tensions, also called *mechanical properties*. Both cases need specialised staff and expensive machines to test the castings and, in the case of mechanical properties, destructive inspections that render the casting invalid are also required. In our previous research, we have modelled the foundry process to apply machine-learning techniques to foresee these defects before making the castings. Nevertheless, these approaches do not correct what is happening in the plant. In this paper, we extend our approach proposing a genetic algorithm that employs meta-classifiers and cosine similarity measure to find the optimal parameters that allow us to change the foundry production, avoiding the aforementioned defects on-line.

I. Introduction

The manufacturing process is an important part of the current society. It is considered as one of the main axes that has allowed the society to evolve. Thanks to it, currently, consumers can have different products and services. Specifically, the casting production, also known as foundry process, is considered one of the main factors that influences the development of the world economy.

The result of this process is the input for other industries. The manufactured castings end up being part of other complex systems. In fact, the actual capacity of the casting production of the world, which is higher than 60 million metric tons per year, is strongly diversified [1]. In the year 2010, China, which is the biggest producer of castings in the world, produced 39.6 million tons of castings and Europe, the second producer, made 14.29 million tons of castings.

Due to current manufacturing trends, it is really easy to produce castings and suddenly discover that every single one is faulty. Currently, the techniques to assure the failure-free process are expensive and they only achieve good results in *a posteriori* manner. Taking into account the amount of casting produced by the countries, these techniques are incapable of preventing the defects, which means an extra cost. If *ex-ante* methods were developed to foresee several defects and the way of managing the manufacturing parameters to avoid them, the quality of the final castings will be improved and the process of remaking faulty parts will be removed.

Two of the most difficult targets to detect in ductile iron castings are *the microshrinkages* and *the mechanical properties*. The first one, also called secondary contraction, consists in tiny porosities that appear inside the casting when it is cooling down. For the second one, the mechanical properties, we have selected the *ultimate tensile strength* that is the force, which a casting can withstand until it breaks, in other words, it is the maximum stress any material can withstand when is subjected to tension.

In previous research, we employed *machine-learning* classifiers to predict the results of the castings [2], [3], [4], [5], [6], [7], [8]. We decided to apply these ideas due to they have been successfully applied in similar domains, for instance, for fault diagnosis [9], malware detection [10] or for cancer diagnosis [11]. Nevertheless, this task is very difficult because: (i) a huge amount of data, not prioritised or categorised in any way, is required to be managed, (ii) it is very hard to find cause-effect relationships between the variables of the system, and (iii) the human knowledge used in this task usually tends to be subjective, incomplete and not subjected to any test.

Despite the fact that with our previous approach we overcame the aforementioned problems and achieving sound results, several topics of discussion appear and remain unsolved. In particular, by using the classifiers as a stand-alone solution: (i) we cannot be completely sure that the selected classifier is the best one to generalise the manufacturing process, (ii) the learning algorithms employed for creating some of the machine-learning classifiers only find a local maximum and, hence, the final result is not optimal and (iii) by using a single classifier, we should generate a classifier close to the process nature (linear or non-linear). Hence, we solved all these problems developing and testing several methods to combine heterogeneous classifiers [8]. This new approach was safer because we used all the classifiers instead of selecting one of them and we can approximate their behaviour to the optimal one. Once finished this research, we were able to detect the defects in an *ex-ante* method, but we were not able to modify the parameters of the plant to solve them online.

Against this background, we present here our approach that makes use of a genetic algorithm to search the optimal parameters that avoid the predicted defect without changing a huge amount of variables. Specifically, our genetic algorithm is supported by the previous research. The fitness function, which quantifies how good the parameters are, is based on: (i) the previously tested meta-classifiers [8], which are responsible for discovering whether the generated solution is faulty or not, and (ii) the cosine similarity, a measure that shows how close 2 vectors are taking into account the angle between them and the method to assure that the final solution is the one that changes less variables of the process.

Our contributions are: (i) we describe how to adapt evolutionary programming to generate the data that we are going to use as feedback to the plant, (ii) we provide a genetic algorithm with concurrent calculations for the fitness function, (iii) we show how to combine the meta-classification process and the cosine similarity to calculate the suitability of each proposed solution and, finally, (iv) we evaluate empirically our approach.

II. High Precision Foundry and Defects

A. Foundry Process

The foundry, as it is defined in Cambridge Dictionary, is a factory where metal is melted and poured into specially shaped containers to produce objects such as wheels and bars. In this research we focus on foundries which produce 'near-net shape' components. In order to obtain the final casting, metals, in our case iron metals, have to pass through several stages in which raw materials are transformed [12]:

- **Pattern making.** In this step, moulds (exteriors) or cores (interiors) are produced in wood, metal or resin for being used to create the sand moulds.
- **Sand mould and core making.** The specialised machines create the two halves of the mould (sand mixed with clay and water or other chemical binders) and join them together to provide a container in which the metals are poured into.
- **Metal melting.** Raw materials are molten and mixed in a furnace. Depending on the choice of the furnace, the quality, the quantity and the throughput of the melt change.
- **Casting and separation.** The molten material is poured onto the sand mould. Later, the metal begins to cool. Finally, the casting is separated from the sand recovering it for further uses.
- **Removal of runners and risers.** Some parts of the casting that had been used to help in the previous processes are removed.
- **Finishing.** Some actions are usually performed to finish the process, e.g., cleaning the residual sand, heat treatment and rectification of defects by welding.

The complexity of optimising this process avoiding several defects stems from the huge amount of variables to manage along the whole foundry process and, therefore, the way in which these variables influence the final design of a casting. In consequence, we have simplified the manufacturing process to solve this problem. In this way, the main variables to control can be classified into the following categories:

- **Metal-related variables**
  - *Composition:* Type of treatment, inoculation and charges [13].
  - *Nucleation potential and melt quality:* Obtained by means of a thermal analysis program [14], [15], [16].
  - *Pouring:* Duration of the pouring process and temperature.
- **Mould-related variables**
  - *Sand:* Type of additives used, sand-specific features and carrying out of previous test or not.
  - *Moulding:* Machine used and moulding parameters.

Furthermore, we included several variables to control the dimension and geometry of the casting. In addition, we took into account other parameters regarding the configuration of each machine working in the manufacturing process [17]. Therefore, we can represent the castings with 24 different variables [2].

B. Microshrinkages

Michroshrinkage is a defect that usually appears during the cooling phase of the metal. This defect cannot be discovered until the casting is finished. Particularly, this flaw consists of a filamentary shrinkage in which the cavities are very small but large in number. This imperfection commonly appears due to metals are less dense as a liquid than a solid. Hence, along the solidification process, the volume of the casting decreases and the density of the metal increases in parallel, this might be rendered in the apparition of microscopically undetectable interdendritic voids.

The most widely techniques that allow us to analyse castings in order to detect the microshrinkages are X-rays and ultrasonic emissions. Although these methods are not destructive, unfortunately, they require suitable devices, specialised staff and quite a long time to analyse all the parts. Therefore, post-production inspection is not an economical alternative to the pre-production detection of microshrinkages.

Although we have already obtained overall significant results through a supervised machine-learning-based approach predicting those imperfections [2], [5], [6], [8], these approaches do not completely solve the fault and a correction step is required.

C. Mechanical Properties

When the casting is finished and is placed in a more complex system, it will be subject to several forces (loads). Therefore, it is important to recognise how mechanical properties influence iron castings [14]. Specifically, the most important mechanical properties of foundry materials are the following ones [18]: strength (there are many kinds of strength such as ultimate strength and ultimate tensile strength), hardness, toughness, resilience, elasticity, plasticity, brittleness, ductility and malleability.

Currently, there are several standard procedures for measuring the performance of the materials regarding the mechanical properties; unfortunately, the only way is employing destructive inspections. In addition, this complex process also requires suitable devices, specialised staff and quite a long time to analyse the materials.

Actually, the ultimate tensile strength, on which we focus here on, is examined as follows. First, a scientist prepares a testing specimen from the original casting. Second, the specimen is placed on the tensile testing machine. Finally, the machine pulls the sample from both ends and measures the force required to pull the specimen apart and how much the sample stretches before breaking.

We have already presented several work applying machine-learning-based classifiers with the aim of predicting these features [5], [8], [3], [4], [7]. Nevertheless, as for the microshrikages, these approaches only allow us to know that our production is going to be spoiled before doing it and no intervention or suggestion is given by the systems.

III. Evolutionary programming

Evolutionary programming, through the employment of *evolutionary algorithms*, tries to solve problems that require an adaptation, search or optimisation. To this end, evolutionary programming is based on well-established laws of the biology, such as the Darwin's Theory Of Evolution [19].

The evolutionary algorithms [20], hereafter referred to as genetic algorithms, is an iterative process of evolution, which keeps constantly a population between tens and hundreds individuals created, originally, in a random manner or under a known heuristic. Along each iteration, called generation, the individuals are evaluated to determine how closer are to the solution of the initial problem. To generate a new generation, they are selected with a probability proportional to their fitness to the solution. In this way, the number of times that a individual is chosen is approximately proportional to their relative performance in the population. The best individuals are more likely to reproduce and generate the new generation. The algorithm ends when it reaches a predefined number of generations or when the optimal solution is found.

The genetic algorithm developed in this work is based on these fundamentals. However, the selected domain, the foundry, adds some constraints to the problem. The most important one is the available time to find the optimal solution. In order to reduce the execution time, we employed the next evolution of genetic algorithms: *Evolutionary Parallel Algorithms* [21].

Hence, we detected that the fitness calculation was the bottleneck of our algorithm. To solve it, we chose to evaluate simultaneously all the individuals splitting them into the different cores of our computer[1]. In our case, we selected a population of 10 individuals to evaluate 10 fitness functions at the same time and keep 2 cores to process other tasks.

Below we explain each of the tasks of the genetic algorithm. Specifically, we will extend the information of the initialization process of the genomes, the generation of new individuals and the selected way for the evaluation of their fitness.

A. Initialisation

This task allows us to create the initial population that is going to be evolved by the genetic algorithm and in which will be the final solution. As in every genetic algorithm, the initialisation is done through a random process. Nevertheless, we have tuned this process to avoid the apparition of values out of the constraints defined in the foundry process. Then, when we start the process, the constructor employs the prediction models to extract the lower and the upper bounds. This means that we are not going to give the feedback to the plant outside the prefixed limits and these will be updated when we change the prediction models.

Furthermore, there are some variables that cannot be changed. Hence, we set the value of the variable and during the initialisation stage these values can only achieve one value.

B. Creating New Generations

The process for generating new individuals is one of the most important in genetic algorithms. In this section we detail how each of the genetic operations works in the method adopted in our approach.

1) Crossover: This is the first operation. For this operation, given 2 known individuals, we will generate 2 new individuals, who will be members of the next generation. This process will be performed if a number generated randomly exceeds a fixed *crossover rate*, which, in our case, is 80%.

When the rate is surpassed, one of the simplest methods is applied [22]. In this way, given 2 parents, we select one gene randomly and we swap the genes from that point to the end in both parents.

2) Mutation: The second operation is the mutation. This operation is required because if we work only combining the population, we will always generate solutions of the same space and we will find a local optimum. Nevertheless, if we change some genes randomly, we will expand the solution area allowing us to find the final solution.

A mutation will be produced if a randomly generated number surpasses the mutation rate of 5%. A mutation for us is a change of one gene modifying its value via the random number generation but fulfilling the constraints defined by the foundry process.

3) Selecting the individuals: This task is focused on selecting the best individuals to combine their genes, allowing finding the optimal solution. In our approach, we have selected the *Fitness Proportionate Selection*, also known as *Roulette-wheel selection*. In this method, each of the fitness values has assigned one probability of selection tightly related to its fitness value.

This selecting method has a high probability of choosing the best individuals, but without rejecting the worst ones because this fact might be useful to generate a better solution. We combine this method with the *elitism*, keeping the best individual between generations.

C. Calculating the Fitness of each Individual

This operation allows us to assess how close the solution is to the final solution. The evaluation is done through the *fitness function* [23]. Our calculation is carried out as is defined below.

For the predictions, we employed the best meta-classifiers for each defect [8] (i.e., *stacking* with Bayesian Networks learnt with Tree Augmented Naïve (TAN) for the structural

[1]Intel® Core^{TM} i7-3930 CPU 3,20 GHz, 24GB of RAM, Operating System: Windows 7 (64 bits)

Algorithm 1: Fitness function evaluation.

```
Input: Current genome, $Gn$
Input: Faulty genome, $Gn_e$
Output: Fitness value, $f \in [0,1]$
prediction ← PredictStationaryState($Gn$)
if ErrorExists(prediction) then
    $f \leftarrow 0$;
end
else
    $\overrightarrow{v_1} \leftarrow$ GetVector($Gn$)
    $\overrightarrow{v_2} \leftarrow$ GetVector($Gn_e$)
    $f \leftarrow$ CosineSimilarity($\overrightarrow{v_1}$, $\overrightarrow{v_2}$)
end
return $f$
```

learning to predict the microshrinkages and *grading* with Bayesian Networks learnt with TAN as structural learning algorithm to foresee the ultimate tensile strength) and the cosine similarity [24] for measuring the proximity between solutions and the current state of the plant, in order to modify the fewest number of variables.

IV. Experimental Results

To test our approach, we have collected data from a real foundry specialised in safety and precision components for the automotive industry, principally in disk-brake support with a production over $45,000$ tons a year. These experiments are focused exclusively on finding new parameters to solve the aforementioned targets: (i) microshrinkages and (ii) the ultimate tensile strength.

Specifically, we have conducted the next methodology in order to evaluate properly our genetic algorithm:

- **Evaluation of instances through predictive models.** In this step, each evidence has been analysed by the predictive model. If the meta-classifier discovered a defect in the casting, we would employ our genetic algorithm to solve it.
- **Finding the optimal solution.** Once the system detected the faulty casting, we run the genetic algorithm. As we said before, our genetic algorithm has a crossover rate of 80%, a mutation rate of 5%, our population has 10 individuals and it will finish when it accomplishes 2,000 generations. We also maintain the best of the individuals, using the *elitism*. The genetic algorithm is the same for both defects; the only difference is that we employed a different prediction model for each defect.
- **Evaluation of results.** Finished the execution of the searching process, we evaluated the results achieved. In this case we have focused in the followings:
  - *System accuracy.*
  - *Number of modified variables.*
  - *Process of genetic algorithm.*
  - *Execution time.*

After applying the aforementioned methodology, we have obtained the following results. In order to make easier the readability, we have divided the results by the classification target.

A. Microshrinkages

As we mentioned before, there are several approaches based on evolutionary programming to generate non-linear models for controlling the manufacturing process [25], [26], [27], [28]. Nevertheless, these approaches are specific and focused on reducing or solving one objective. Hence, we present the results of our generic genetic algorithm to solve the first objective, the faulty castings flawed with microshrinkages.

TABLE I
SUMMARY OF THE RESULTS RELATED TO SOLVE CASTINGS FLAWED WITH MICROSHRINKAGES.

Corrections	
Total castings	951
Faulty castings	261
Corrected castings	261

In this way, Table I shows the behaviour of the software and how accurate it is. As we can notice, 261 out to 951 castings were faulty and applying our approach we could detect and fix the 100% of them.

We also measured the number of changes needed to redirect the process to the optimum. Table II illustrates with the results achieved by the genetic algorithm. More accurately, we can notice that the lowest number of modified variables is 1. Although the algorithm changes 2 variables in the majority of the executions, there are some cases that we need to fix more than 5 variables, a 5.75%. In fact, the worst result implies to change 19 variables. However, as we can distinguish in the Table II, several statistic measures (average, mode, median) shows that the most representative number of changes is 2.

TABLE II
SUMMARY OF THE RESULTS RELATED TO MODIFICATIONS FOR SOLVING CASTINGS FLAWED WITH MICROSHRINKAGES.

Variable modifications	
Minimum number of modifications	1
Maximum number of modifications	19
Average number of modifications	2.67
Standard deviation	1.94
Mode number of modifications	2
Median number of modifications	2
Number of modification involving 1 variable	37
Number of modification involving de 2 variables	113
Number of modification involving 3 variables	87
Number of modification involving 4 variables	9
Number of modification involving 5 variables	0
Number of modification involving more than 5 variables	15

Thirdly, we gathered the information related to the number of generations (i.e., iterations of our genetic algorithm) that are required to obtain the optimal solution in Table III. As we said before, the condition to finish the algorithm is to reach a number of iterations. Henceforth, after 2000 generations, we realise that our genetic algorithm works in a wide range of generations, [115, 1900]. But if we reduce the number of them, we still maintain a huge amount of optimums. Specifically, the system is able to achieve a 56.32% of optimums using 500 generations, with 1000 generations an 83.91%, and with 1500 a 93.87%. These results imply that we could manage the foundry process using a suboptimal algorithm that achieves a high number of optimal solutions.

TABLE III
SUMMARY OF THE RESULTS RELATED TO REQUIRED GENERATIONS TO GET THE OPTIMUM FOR SOLVING CASTING FLAWED WITH MICROSHRINKAGES.

Generations needed for the optimum	
Minimum number of generations	115
Maximum number of generations	1900
Average number of generations	587.35
Standard deviation	433.62
Median number of generations	449
Optimums achieved with 500 generations	147
Optimums achieved with 1000 generations	219
Optimums achieved with 1500 generations	245

TABLE IV
SUMMARY OF THE RESULTS RELATED TO THE EXECUTION TIME FOR SOLVING CASTINGS FLAWED WITH MICROSHRINKAGES.

Execution Time	
Minimum time	00:02:40
Maximum time	00:04:25
Average time	00:03:21
Average time per generation	0,1005 secs.
Average time for 500 generations	00:00:50
Average time for 1000 generations	00:01:40
Average time for 1500 generations	00:02:31

Finally, we evaluated the execution time of the system (shown in Table IV). These results are really interesting due to this time could be the reason to reject the algorithm. Nevertheless, the achieved time is enough to solve the manufacturing process in a high precision foundry. More accurately, the average time is 03:21 minutes. Moreover, we calculated the average time for the suboptimal algorithms, and these results show that if we work with the suboptimal genetic algorithm that employs 500 generation, we can reduce the time until 50 seconds.

B. Mechanical Properties

Following the same methodology and executing the same genetic algorithm, we just changed the predictive model; we have tested our approach for another problem, the ultimate tensile strength.

Firstly, we tested the performance of the system (shown in Table V). In this case, our approach is not as accurate as for microshrinkages. Regarding the ultimate tensile strength, 244 out of 889 castings have low values for this mechanical property. Our genetic algorithm was able to detect and to solve 232 castings, a 95% of all the evidences. Despite this result, the system still maintains a great control over the production.

Secondly, as for the previous experiment, we measured the number of changes needed to fix the manufacturing process. Table VI illustrates that the most repeated number of changes is 2. The results are as good as in the previous experiment, in fact, only the 9.28% of castings needs to change more than 5 variables. Again, mode, average and median tell us that the number of modifications is close to 2.

TABLE V
SUMMARY OF THE RESULTS RELATED TO SOLVE CASTINGS WITH BAD VALUES OF THE ULTIMATE TENSILE STRENGTH.

Correcciones	
Total castings	889
Faulty castings	244
Corrected castings	232

TABLE VI
SUMMARY OF THE RESULTS RELATED TO MODIFICATIONS FOR SOLVING CASTINGS WITH BAD VALUES OF THE ULTIMATE TENSILE STRENGTH.

Variable modifications	
Minimum number of modifications	1
Maximum number of modifications	12
Average number of modifications	2.68
Standard deviation	2.16
Mode number of modifications	2
Median number of modifications	2
Number of modification involving 1 variable	44
Number of modification involving 2 variables	126
Number of modification involving 3 variables	32
Number of modification involving 4 variables	5
Number of modification involving 5 variables	3
Number of modification involving more than 5 variables	22

Thirdly, we registered the number of generations needed to obtain the optimal solutions. Moreover, we have measured how many optimums can be reached using suboptimal algorithms. This solution can reduce the time and also the computational complexity of the algorithm. Table VII shows the obtained results. For the ultimate tensile strength, the range of the needed generations is similar to the range for microshrinkages, [21, 1993]. And for our suboptimal system, using only 500 generations, the algorithm can solve the 76.37% in the optimal way, 94.41% for 1000 generations and 94.51% for 1500 generations.

TABLE VII
SUMMARY OF THE RESULTS RELATED TO REQUIRED GENERATIONS TO GET THE OPTIMUM FOR SOLVING CASTINGS WITH BAD VALUES OF THE ULTIMATE TENSILE STRENGTH.

Generations needed for the optimum	
Minimum number of generations	21
Maximum number of generations	1993
Average number of generations	337.40
Standard deviation	385.67
Median number of generations	168
Optimums achieved with 500 generations	181
Optimums achieved with 1000 generations	219
Optimums achieved with 1500 generations	224

Finally, the last information collected from the experiments is the execution time. Table VIII displays how much time our algorithm employs to fix the process. In order to solve the ultimate tensile strength, we need more time because the predictive model is more complicated and the duration of the fitness function is longer, nevertheless, the execution time is really close to the previous one. Specifically, the average time is 04:29 minutes. Nonetheless, if we reduce the number of generations, we are going to reduce the execution time, hence, a suboptimal algorithm can require from 01:07 to 03:22 minutes.

TABLE VIII
SUMMARY OF THE RESULTS RELATED TO THE EXECUTION TIME FOR SOLVING CASTINGS WITH BAD VALUES OF THE ULTIMATE TENSILE STRENGTH.

Execution time	
Minimum time	00:04:00
Maximum time	00:04:59
Average time	00:04:29
Average time per generation	0,1345 secs.
Average time for 500 generations	00:01:07
Average time for 1000 generations	00:02:15
Average time for 1500 generations	00:03:22

V. CONCLUSIONS

On the one hand, microshrinkages are tiny porosities that appear when the casting is cooling down. On the other hand, ultimate tensile strength is the capacity of a metal to resist deformation when subject to a certain load. Our previous research [2], [3], [4], [5], [6], [7], [8] pioneers the application of Artificial Intelligence to the prediction of these two features. However, this approach does not solve the problems in the plant. Specifically in this paper, we have extended the predictive model with a genetic algorithm to find the optimal parameters to give the feedback to the plant. For both problems, our algorithm was very accurate detecting the faulty castings and correcting them with few changes in the process, 2 variables.

Accordingly, future work will be focused on 2 main directions. First, we plan to extend our analysis for solving other defects in order to develop a global system of incident analysis. And, second, we plan to test other searching methods that can reduce the execution time reached by the genetic algorithm.

REFERENCES

[1] "45th census of world casting production," Modern Casting, Tech. Rep., 2011, monthly report, edited annually with the data concerning the number of casting houses and the world casting production in the year preceding the issue.

[2] I. Santos, J. Nieves, Y. K. Penya, and P. G. Bringas, "Optimising machine-learning-based fault prediction in foundry production." in *Proceedings del 2 International Symposium on Distributed Computing and Artificial Intelligence (DCAI)*, 2009, in press.

[3] J. Nieves, I. Santos, Y. K. Penya, S. Rojas, M. Salazar, and P. G. Bringas, "Mechanical properties prediction in high-precision foundry production." in *Proceedings de la 7ł IEEE International Conference on Industrial Informatics (INDIN 09)*, 2009, pp. 31–36.

[4] J. Nieves, I. Santos, Y. K. Penya, F. Brezo, and P. G. Bringas, "Enhanced foundry production control," in *Proceedings of the 21st International Conference on Database and Expert Systems Applications (DEXA). Lecture Notes in Computer Science 6262, Springer-Verlag Berlin Heidelberg*, 2010, pp. 213–220.

[5] I. Santos, J. Nieves, P. G. Bringas, and Y. K. Penya, "Machine-learning-based defect prediction in highprecision foundry production," in *Structural Steel and Castings: Shapes and Standards, Properties and Applications*, L. M. Becker, Ed. Nova Publishers, 2010, pp. 259–276.

[6] I. Santos, J. Nieves, and P. G. Bringas, "Enhancing fault prediction on automatic foundry processes," in *World Automation Congress (WAC), 2010*. IEEE, 2010, pp. 1–6.

[7] I. Santos, J. Nieves, Y. K. Penya, and P. G. Bringas, "Machine-learning-based mechanical properties prediction in foundry production," in *In Proceedings of ICROS-SICE International Joint Conference (ICCAS-SICE)*, 2009, pp. 4536–4541.

[8] J. Nieves, I. Santos, , and P. G. Bringas, "Combination of machine-learning algorithms for fault prediction in high-precision foundries," in *Database and Expert Systems Applications - 23rd International Conference, DEXA 2012*, ser. Lecture Notes in Computer Science, S. W. L. al., Ed. Springer Berlin / Heidelberg, 2011, vol. 6861, pp. 56–70.

[9] J. Yang, Y. Zhanga, and Y. Zhu, "Intelligent fault diagnosis of rolling element bearing based on svms and fractal dimension," *Mechanical Systems and Signal Processing*, vol. 1, pp. 2012–2024, 2007.

[10] I. Santos, Y. K. Penya, J. Devesa, and P. G. Bringas, "N-grams-based file signatures for malware detection," in *Proceedings de la 11ł International Conference on Enterprise Information Systems (ICEIS), Volume AIDSS*, 2009, pp. 317–320.

[11] P. Lisboa and A. Taktak, "The use of artificial neural networks in decision support in cancer: a systematic review," *Neural Networks*, vol. 19, no. 4, pp. 408–415, 2006.

[12] K. S. and S. Schmid, *Manufacturing, Engineering and Technology*. Pentice Hall, 2005.

[13] C. J. F. and R. Ríos, "A fracture mechanics study of nodular iron," *Revista de Metalurgía*, vol. 35, no. 5, pp. 279–291, 1999.

[14] R. Gonzaga-Cinco and J. Fernández-Carrasquilla, "Mecanical properties dependency on chemical composition of spheroidal graphite cast iron," *Revista de Metalurgia*, vol. 42, pp. 91–102, 2006.

[15] M. Hecht and F. Condet, "Shape of graphite and usual tensile properties of sg cast iron: Part 1," *Fonderie, Fondeur d'aujourd'hui*, vol. 212, pp. 14–28, 2002.

[16] P. Larraaga, J. Sertucha, and R. Suárez, "Análisis del proceso de solidificación en fundiciones grafíticas esferoidales," *Revista de Metalurgia*, vol. 42, pp. 244–255, 2006.

[17] J. Sertucha, R. Suárez, J. Legazpi, and P. Gacetabeitia, "Influence of moulding conditions and mould characteristics on the contraction defects appearance in ductile iron castings," *Revista de Metalurgia*, vol. 43, pp. 188–195, 2007.

[18] C. Lung and N. March, *Mechanical Properties of Metals: Atomistic and Fractal Continuum Approaches*. World Scientific Pub Co Inc, Julio 1992.

[19] C. Darwin, *The Origin of Species*. Collier, 1937.

[20] J. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan press, 1975, no. 53.

[21] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs paralleles, reseaux et systems repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[22] M. Tomassini, "A survey of genetic algorithms," *Annual Reviews of Computational Physics*, vol. 3, no. 2, pp. 87–118, 1995.

[23] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, USA, 1996.

[24] P. Tan, M. Steinbach, V. Kumar *et al.*, *Introduction to Data Mining*. Pearson Addison Wesley Boston, 2006.

[25] C. Onnen, R. Babuska, U. Kaymak, J. Sousa, H. Verbruggen, and R. Isermann, "Genetic algorithms for optimization in predictive control," *Control Engineering Practice*, vol. 5, no. 10, pp. 1363–1372, 1997.

[26] H. Al-Duwaish and W. Naeem, "Nonlinear model predictive control of hammerstein and wiener models using genetic algorithms," in *Control Applications, 2001.(CCA'01). Proceedings of the 2001 IEEE International Conference on*. IEEE, 2001, pp. 465–469.

[27] H. Sarimveis and G. Bafas, "Fuzzy model predictive control of non-linear processes using genetic algorithms," *Fuzzy sets and systems*, vol. 139, no. 1, pp. 59–80, 2003.

[28] J. Yang, M. Liu, and C. Wu, "Genetic algorithm based nonlinear model predictive control method," *Control and Decision*, vol. 18, no. 2, pp. 141–144, 2003.