Tuned Forest Fire Prediction: Static Calibration of the Evolutionary Component of 'ESS'

Germán Bianchini[a] **and Paola Caymes Scutari**[a,b]
[a]Laboratorio de Investigación en Cómputo Paralelo/Distribuido,
Departamento de Ingeniería en Sistemas de Información
Facultad Regional Mendoza - Universidad Tecnológica Nacional
Mendoza, CP (M5502AJE) Argentina
{*gbianchini,pcaymesscutari*}*@frm.utn.edu.ar*
[b]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)

Abstract

Forest fires are a major risk factor with strong impact at eco-environmental and socio-economical levels, reasons why their study and modeling are very important. However, the models frequently have a certain level of uncertainty in some input parameters given that they must be approximated or estimated, as a consequence of diverse difficulties to accurately measure the conditions of the phenomenon in real time. This has resulted in the development of several methods for the uncertainty reduction, whose trade-off between accuracy and complexity can vary significantly. The system ESS (Evolutionary-Statistical System) is a method whose aim is to reduce the uncertainty, by combining Statistical Analysis, High Performance Computing (HPC) and Parallel Evolutionary Algorithms (PEAs). The PEAs use several parameters that require adjustment and that determine the quality of their use. The calibration of the parameters is a crucial task for reaching a good performance and to improve the system output. This paper presents an empirical study of the parameters tuning to evaluate the effectiveness of different configurations and the impact of their use in the Forest Fires prediction.

Keywords: Parameters Calibration, Tuning, Uncertainty Reduction, Evolutionary Algorithms, High Performance Computing

1 Introduction

Modeling is a mathematical representation and application of methods for the analysis of complex real-world problems in order to make predictions about what should happen according to certain actions based on the values assumed by the system. Furthermore, through simulation, the models are implemented in the form of computer programs, with input parameters that describe the variables at the modeled system and allow for the interpretation of such a model to obtain a result that represents the expected behavior of the system under such starting conditions. However, it is common for simulation tools produce results certainly far from the actual behavior of the simulated phenomenon, as there are several sources of error (in terms of accuracy): errors, assumptions or limits in the model formulation, algorithmic or logical errors in the computational version, constraints related to numerical representation, precision, rounding and truncation, etc. Unfortunately, even though the formulation of the model and the algorithm implementation are free of errors, it is very difficult to provide a simulator ever faithful to the actual situation to be represented, given the uncertainty introduced by both mathematical expressions and by the computer numerical representation, as both have limitations intrinsic to the difficulty or impediment to measure, quantify and/or represent all aspects or parameters involved in the system under study. There are different methods for reducing uncertainty (where uncertainty is the lack of clear and certain knowledge about something, in this case we are referring to the real values of the input parameters of the model), which can diminish the negative effects caused by the many worlds or scenarios, mathematical and computationally representable. Thus, the reduction of uncertainty is a very important process to achieve more consistent predictions to reality. At the same time, it can provide reliable tools to assist decision support systems. This paper focuses on a method for reducing uncertainty, called ESS, which is based on four pillars: statistics, evolutionary computing, high

performance computing and tuning. The statistics provide the opportunity to study trends in the behavior of the system under various conditions. Evolutionary computation can guide the search to obtain a more accurate prediction. Parallel computing provides the infrastructure to exploit and expand the search space, in reasonable times. Tuning is a process to modify, adjust, improve and/or calibrate the various parameters and aspects involved in the system. This process can achieve quality results in a shorter time and with a more efficient resource utilization. While the ESS method can be applied in any system of prediction and prevention that involves simulation, in this paper we present results obtained for the prediction of forest fires.

This paper is structured as follows: Section 2 describes basics concepts about forest fires, their causes and effects, and the relationship with the simulators. Section 3 presents a classification of prediction methods and provide details about the ESS method. In Section 4 we define the most important issues related to tuning or calibration process considered for the experimental design. Section 5 shows experiments with their results and finally, Section 6 presents the conclusions.

2 Forest Fires

Forest fires, which can be defined as a fire that spreads freely by vegetation with unwanted effects for the same (not to be confused with controlled burns) can cause great ecological damage in vast tracts of land, the flora, the wildlife, water resources and soils. Forest fires produce economic damage on timber and non-timber products such as fences, sheds, yards, homes, etc., while ravaging the landscape (damage especially negative for the tourist areas) [1]. The risk increases when weather conditions are extreme, with dry seasons, high temperatures and strong winds. Southern Europe is a clear example of regions where conditions are at high risk of fire, and that every year suffer losses of various magnitudes. Also in Argentina forest fires occur with some frequency, affecting native forests and inevitably impacting on the ecosystem. A clear example is the series of fires that affected the area of Tierra del Fuego in the course of only two months in early 2012, including Bahía Torito, Bahía San Pablo, El Diamante lagoon and Corazón de la Isla (Figure 1). More recently (in September 2013), there were a series of fires that spread through six provinces (Córdoba, San Luis, Salta, Tucuman, Río Negro and Neuquén). This phenomenon devasted 95,000 hectares, affecting more than 40 homes and causing death of thousands of animals [2] (see Figure 2).

Figure 1: Fire brigades fight fire in Bahía Torito (Ushuaia, Argentina) [3]

Unfortunately, although there are natural causes that originate fires (lightning, volcanic activity, etc.), most of the fires are provoked by human causes [4]: deforestation, burning trash, deliberate fires, accidents, etc. That is why the process of firefighting attempts to make use of different types of tools and resources for prevention, prediction, detection and monitoring, management and firefighting itself, so as to speed up the fight against the flames to minimize the negative effects caused by the fire.

Simulation of forest fire propagation is a challenge from the computational point of view, given the complexity of the models involved, the need for numerical methods and the difficulty in reaching an efficient resource management in order to obtain good results. In this context, the class of methods presented in this paper is an important tool for the prevention and prediction of forest fires, since it provides more complete information about the potential fire behavior and areas at greatest risk of ignition. The level of risk depends on static factors such as the slope of the land or the particular type of vegetation, but also depends on dynamic factors such as vegetation moisture and wind speed and direction. When a fire occurs, it is very difficult (if not impossible) to know in advance the specific conditions under which the fire started. This

Figure 2: Uncontrollable fire in Calamuchita (Córdoba, Argentina) [Telam]

problem makes it very difficult to estimate the velocity of propagation of fire or flame intensity in order to predict the behavior of a fire. Therefore, the calibration of the parameters of the method is essential to reduce the sources of uncertainty that attenuate the accuracy of predictions.

3 Classification of Prediction Methods

This section provides a classification of prediction methods applied to the forest fires management, starting from what is known as Classical Prediction or classic application of a forest fire prediction simulator, through Data-Driven methods of one solution and multiple overlapping solution. In the last class, lies the Statistical Evolutionary Statistical System (ESS) on which we have carried out the parameters tuning described in this paper.

3.1 Classical Prediction

Basically, classical prediction is the application of any fire simulator to assess the fire front position after a certain time interval. The simulator must be fed with all required parameters (vegetation, weather information, area of ignition, etc.). Then it is put into operation to predict the fireline in the next time instant. This scheme is shown in Figure 3, where FS corresponds to the fire behavior model, which is considered as a black box. RFL0 is the real state of the fire in time t_0 (initial fire front) while RFL1 state corresponds to the real fire front at time t_1. After applying the parameters and RFL0 to FS, the state of the fire front predicted is represented by PFL, which is expected to match RFL1. However, due to the complexity of the fire behavior model, this scheme of work is not usually enough to provide a result close to reality as to be considered a reliable working tool. Some examples of Classical prediction are [5, 6, 7, 8, 9, 10, 11].

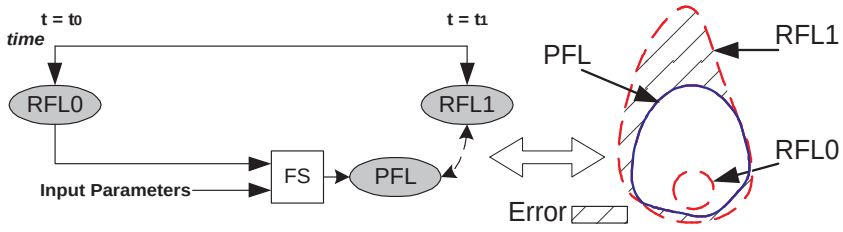

Figure 3: Diagram of Classical Prediction of forest fire propagation (FS: Fire Simulator; PFL: Predicted Fire Line; RFLX: Real Fire Line on time X)

One reason for this discrepancy between the real and the simulated spread arises from the difficulty of feeding the input model with precise values.

3.2 Data-Driven Methods

There are multiple algorithms designed to examine a search space and find a solution to a problem (eg. Exhaustive Search, Local Search, Greedy Algorithms, Divide and Conquer, Branching and pruning, etc.). Moreover, problems are constantly changing and therefore algorithms should change too.

Some algorithms or strategies mentioned guarantee to find a solution, some not, but all share one thing in common: or guarantee to find the global solution to a very high cost (for instance in processing time), or tend to fall into an local optimun.

For the case of propagation prediction of physical or environmental phenomena, the search space can be identified with the set of possible combinations of values in the model input parameters, while a solution (optimal or acceptable) would be a combination of values that satisfies the requirements and can achieve a solution whose quality is acceptable.

For these reasons, it is interesting the application of another modern heuristics, which are frequently used in the geosciences (e.g. in predicting climate and hydrology, to name a couple of them). In general, the methods developed [12, 13, 14, 15] operate over a large number of input values and, through some kind of optimization, they focus on finding a single set of values that describe the behavior of the best possible way. That is, the objective of the optimization focuses on finding a set of values, such that if these values are applied to the model in question (which normally could have been implemented in a simulator), it would be possible to correctly describe the previous behavior, i.e. the behavior that has been used to calibrate or to find the set of parameters. Therefore, it is expected that the same set of values can be used to explain the behavior of the immediate future, i.e. some kind of temporal and spatial locality is assumed. Systems that use this type of methodology are known as Data-Driven Methods (DDM).

Schematically, the DDM operate on a stage called calibration step (CS). Figure 4 shows how they work on large amounts of values (different combinations of inputs that produce various scenarios). This feature is what explains the extra time required to compute all the information. Finally, a very important difference between DDM and classical method is that DDM apply some kind of calibration or optimization to find the most suitable parameter set.

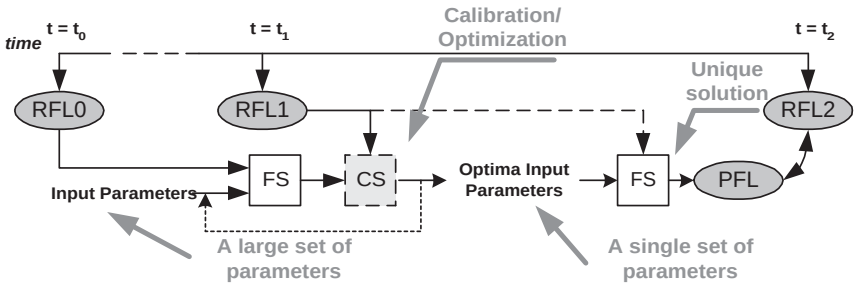

Figure 4: Diagram of Data-Driven Prediction of forest fire propagation (FS: Fire Simulator, CS: Calibration Stage, PFL: Predicted Fire Line, RFLX: Real Fire Line on time X)

Here are some specific cases of DDM applied to the problem of predicting the behavior of forest fires, including three heuristic optimization methods implemented under a scheme of parallel processing within the Black-Box Optimization Framework (BBOF) [12], which works iteratively advancing step by step from an initial set of assumptions to reach the final expected value (in this case, the set of input parameters).

- **BBOF - Taboo Search:** Taboo search (TS) [16] is a metaheuristic search method employing local search methods used for mathematical optimization. Taboo search uses a local or neighborhood search procedure to iteratively move from one potential solution x to an improved solution x' in the neighborhood of x, until some stopping criterion has been satisfied (generally, an attempt limit or a score threshold). Local search procedures often become stuck in poor-scoring areas or areas where scores plateau. In order to avoid these pitfalls and explore regions of the search space that would be left unexplored by other local search procedures, tabu search carefully explores the neighborhood of each solution as the search progresses. The solutions admitted to the new neighborhood, $N^*(x)$, are determined through the use of memory structures (taboo list). Using these memory structures, the search progresses by iteratively moving from the current solution x to an improved solution x' in $N^*(x)$.

- **BBOF - Simulated Annealing:** Simulated annealing (SA) [17, 18] is a method for solving unconstrained and bound-constrained optimization problems. The method models the physical process of heating a material and then slowly lowering the temperature to decrease defects, thus minimizing the system energy. For certain problems, simulated annealing may be more efficient than exhaustive enumeration –provided that the goal is merely to find an acceptably good solution in a fixed amount of time, rather than the best possible solution–. At each step, the SA heuristic considers some neighbouring state s' of the current state s, and probabilistically decides between moving the system to state s'

or staying in state s. These probabilities ultimately lead the system to move to states of lower energy. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted.

- **BBOF - Genetic Algorithm:** A genetic algorithm (GA) [19] is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover (over generations, the populations evolve in nature according to the principles of natural selection and survival of the fittest, proposed by Darwin [20]). In imitation of this process, in a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

- **GLUE (Generalized Likelihood Uncertainty Estimations):** GLUE is another statistical method for quantifying the uncertainty of model predictions. The method has been introduced by Beven and Binley [21] for hydrological problems. The basic idea of GLUE is that given our inability to represent exactly in a mathematical model how nature works, there will always be several different models that mimic equally well an observed natural process (such as river discharge). Such equally acceptable or behavioral models are therefore called equifinal. The methodology deals with models whose results are expressed as probability distributions of possible outcomes, often in the form of Monte Carlo simulations [22], and the problem can be viewed as assessing, and comparing between models, how good these representations of uncertainty are. There is an implicit understanding that the models being used are approximations to what might be obtained from a thorough Bayesian analysis of the problem if a fully adequate model of real-world processes were available. An implementation-oriented forest fire prediction was made in [23] and another, following a parallel processing scheme in [24].

However, the results obtained with DDM suffer the same problem that classical predictions: they find a single set of values, and for those parameters that dynamically change, the value found in general are not useful to describe correctly the behavior of the model in question.

3.3 Data-Driven Methods with Multiple Overlapping Solution

Next, we describe two methods which try to solve the problem caused by uniqueness set of input parameters that, as explained, DDM use. Both methods belong to a new branch of DDM that perform prediction considering the overlapping of several cases or combinations of parameters.

- **Statistical System for Forest Fire Management:** S^2F^2M [25, 26] is in a branch of Data-Driven Methods with Multiple Overlapping Solution. Although S^2F^2M belongs to the Data-Driven type, it generates predictions based on all the proposed cases, rather than on a single case, such as GLUE and BBOF. S^2F^2M considers a large set of scenarios to carry out the search of the forest fire behavior (we define the concept of scenario as a particular setting of the set of parameters). Unlike the methods of unique solution, S^2F^2M does not make a distinction between the best and worst cases (or at least it does not try to make any type of classification): the interesting fact of this methodology is that every case (i.e. every scenario), helps, with its particular characteristics, to find a better prediction to describe the behavior pattern. All possible scenarios are generated in a discrete manner considering a certain domain by a factorial experiment [27] and then the model is evaluated for each set of values. The results are aggregated to determine the trend in the behavior of the model, fitting with the current observation of the fireline, in the following way: We obtain a matrix with a value associated to each cell that represents the probability of each cell to be caught by the fire (P_{ign}, defined as n_A/n where n is the total number of scenarios and n_A is the number of scenarios in which the cell A was burned). The set of cells, whose P_{ign} value is higher than or equal to a certain particular value P_K, where $0 \leq P_K \leq 1$, constitutes what we call the probability map with probability P_K, defined in the expression (1):

$$\{x : P_{ign}(x) \geq \frac{K_{ign}}{n} \mid K_{ign} \in N\} \tag{1}$$

with n equal to the number of scenarios and $P_{ign}(x)$ varying from K_{ign}/n to 1, i.e. the set of cells (x) which have been burned at least K_{ign} times. Once we have obtained the output matrix, which includes

all the probability maps, the next step consists in comparing the real fire against this matrix. The objective of such a comparison is to search for a particular value of P_{ign}, whose associated probability map provides the best matching with the real fire propagation. In other words, we are interested in finding what we refer to as a Key Ignition number (K_{ign}). The pattern found is then considered for the prediction of the next step.

This way of work requires a large number of operations, resulting very demanding in execution time. For this reason, S^2F^2M has been implemented under a scheme of parallel/distributed computation. In this scheme we use many resources working in parallel with the Master-Worker paradigm [28], because the main processor may compute and send parameters combinations (scenarios) to the workers, who then work on different domains. Therefore, given the characteristics of the method, it has been classified as a new branch that extends to the previously mentioned classification, which has been called Data Driven Methods with Multiple Overlapping Solution (DDMMOS).

Figure 5 shows a schematic diagram of the system S^2F^2M. The calibration stage (CS) consists of three successive steps: a statistical stage (SS) which feeds on the partial results offered by Workers working in parallel, the stage SK (search key value K_{ign}, to be used in the following prediction time) that is responsible for finding a pattern model, and a fitness function (FF) used to assess the suitability of the results. The SK output in time t_i is combined with the output of SS in time t_{i+1} to generate the prediction in stage FP. Note that calibration for time instant t_{i+1} is overlapped in t_i with prediction calculations.

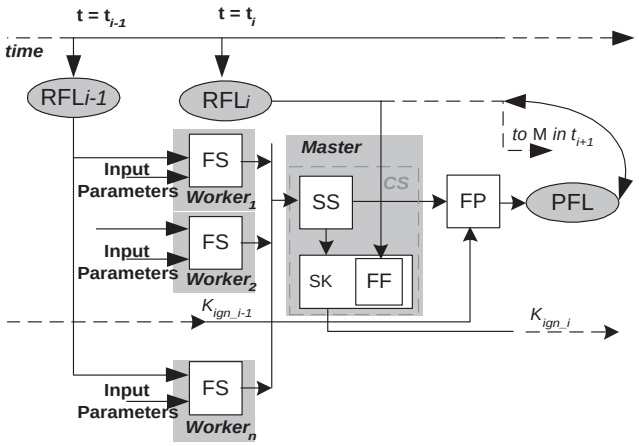

Figure 5: Schematic diagram of the S^2F^2M (RFLX: Real Fire Line on time X, FS: Fire Simulator, CS: Calibration Stage, SS: Statistical Stage, SK: Search K_{ign} Stage, FF: Fitness Function, FP: Fire Prediction, PFL: Predicted Fire Line)

- **Evolutionary-Statistical System:** This method combines the strength of four components: Statistics, Evolutionary Algorithms, Parallelism and Tuning, which has been called Evolutionary-Statistical System (ESS) [29]. It corresponds to an improvement of the S^2F^2M related to the introduction of Parallel Evolutionary Algorithms features [19] in the calibration step. As seen, the stage of the statistical methodology includes all the results from a number of cases that arise as a combination of the possible resulting values (within valid ranges) of parameters whose values exhibit uncertainty. It is clear that there are a certain percentage of cases that do not add significant value to the result, whether they are now redundant, or because they stray too far from reality. To solve this problem we decide to apply evolutionary algorithms (EAs), which are optimization methods inspired by the principles of biological evolution. EAs maintain a set of entities that represent possible solutions, which are mixed, and compete with each other, so that the fittest are able to prevail over time, evolving into better solutions.

  As shown in Figure 6, the system is divided into two general stages: one Optimization Stage (OS) that implements the Parallel Evolutionary Algorithm (PEA box), and a Calibration Stage (CS) that is responsible for the application of the statistical method.

  There are two basic approaches to implement an EA on a parallel scheme: In the first approach, the sequential EA model is implemented on a parallel computer. This is usually done by dividing the task of evaluating the population (and/or the application of genetic operators) among several processors. In the second case, the full population exists in a distributed form. Either multiple independent or

interacting populations exist (coarse-grained or distributed EA), or there is only one population, divided in subpopulations, where each individual interacting only with a limited set of neighbors (fine-grained EA) [30]. In this work, we use the standard parallel aproach, which mantains a single population and the evaluation of the individuals are done in parallel. Following this scheme, OS iterates until the population reaches a certain level of quality. Each individual is then applied to FS and their fitness values are calculated in parallel. This is because there are two kinds of phases PEA: the ESS architecture is based on the Master-Worker paradigm. In each iteration, the Master distributes an individual for each Worker; the simulation of the model and fitness evaluation are applied to each individual (tasks carried out by the Workers, who return the results to the Master). This process is repeated until every individual in the population has been treated. Finally, the PEA of Master evolves the population. These results constitute the input of Statistical Stage (SS box). Similarly to S^2F^2M, SS output (a probability map) has a dual purpose. First, the probability maps are used as input stage SK (search for K_{ign}) to find the current value of K_{ign}, to be used in the following prediction time. At this stage, we use a fitness function (FF) to evaluate the probability map. Moreover, the SS output also enters the stage of prediction (FP), which is responsible for generating the prediction map taking into account the K_{ign} found in the previous time. This process is repeated while the system is fed with information about the different states of the model along the time.

Since the simulator uses an approximation based on cells, the fitness function is defined as a quotient, which is shown in the expression (2).

$$Fitness = \frac{(\#cells \bigcap - \#IgnitionCells)}{(\#cells \bigcup - \#IgnitionCells)} \tag{2}$$

Where $\#cells \bigcap$ represents the number of cells in the intersection between the simulation results and the real map, $\#cells \bigcup$ is the number of cells in the union of the simulation results and the real situation, and $\#IgnitionCells$ represents the number of burned cells before starting the simulation. According to this expression, a fitness value equal to one corresponds to the perfect prediction because it means that the predicted area is equal to the real burned area. On the other hand, a fitness equal to zero indicates the maximum error: the experiment did not coincide with reality at all.

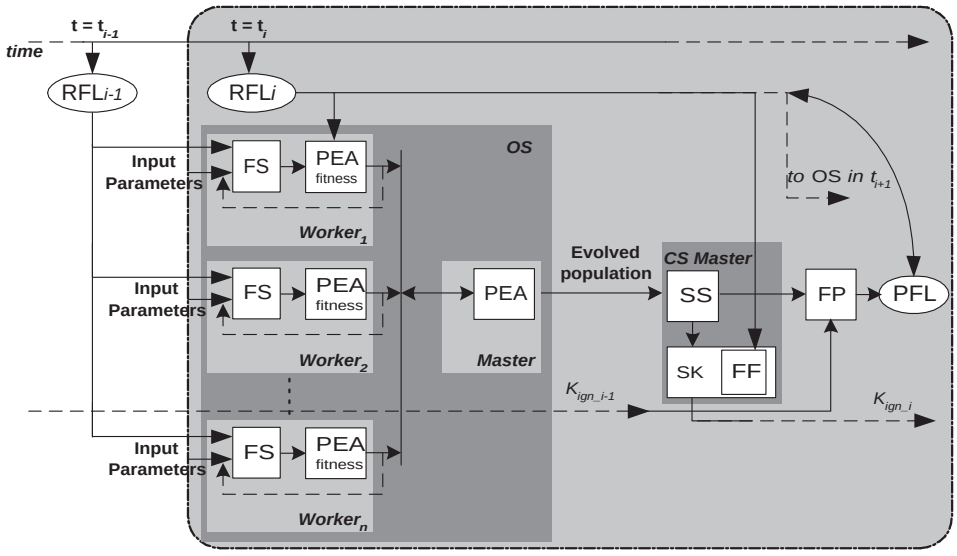

Figure 6: Schematic diagram of the Evolutionary-Statistical System (RFLX: Real Fire Line on time X, FS: Fire Simulator, OS: Optimization Stage, CS: Calibration Stage, PEA: Parallel Evolutionary Algorithm, SS: Statistical Stage, SK: Search K_{ign} Stage, FF: Fitness Function, FP: Fire Prediction, PFL: Predicted Fire Line)

4 Performance Tuning

The tuning process, which corresponds to the adjustment of certain features to improve the performance and operation of the program, covers several successive phases: code instrumentation to collect information,

monitoring of such information, analysis of collected metrics and modification of the application. While any application –sequential or parallel– can be tuned, theory and tuning technology software have been developed in strong connection to the HPC, given the demand for greater power computational both from computational science and engineering, and supported by a relentless progress of technology architectures of high performance [31, 32]. Since the high performance computing is one of the pillars of the ESS presented in this paper, it is appropriate to tune in this case the behavior of the algorithm, in order to achieve better performance while adjusting the method to obtain a prediction more adequate to reality. In this paper, we present a first approach to the performance tuning of ESS in terms of the quality of prediction. So we've followed a classical approach tuning, i.e. the measurements, analysis and modifications have been statically carried out on the program:

- Instrumentation: the program is augmented with additional instructions, inserted for the purpose of data collection, event detection, etc.
- Monitoring: When program execution reaches the instrumentation instructions, the information is obtained and collected, and it is systematized in a trace file. In this file is stored the information collected throughout the execution of the application, for further analysis.
- Analysis: When the execution of the program ends, the data obtained during monitoring are available for analysis, i.e. the analysis is performed *post-mortem*. This analysis may involve more than one execution. Depending on the results and according to the evaluation criteria considered, it is possible identify several changes to improve the program behavior.
- Tuning: the program is modified according to the results provided by the analysis stage. Sometimes the change only affects parameters values, while on other occasions it is necessary to modify the algorithm and/or the instrumentation.

Note that classical tuning requires repeating experiments under different conditions to provide to the analysis stage with adequate information for making decisions. That is why from a set of executions of the application with different operating parameters (and hence a set of trace files), it is possible to conclude what is the best combination of parameters that allow tuning the operation of the program to the runtime characteristics. Below we provide some details about each of the parameters that are subjected to variation. In section related to experiments the values achieved are documented. The assessment of each execution (which is carried out at the analysis phase) takes as parameters of comparison the execution time, the quality of the prediction obtained (given by the fitness value and the speed with which the result is achieved (in terms of number of generations and simulations necessary to calibrate the prediction). This allows us to appreciate the usefulness of including characteristics of evolutionary algorithms to the method, while reducing the search space and the consequent achievement of a result. Although EAs have many operating parameters, the analysis should be limited to the controlled variation of certain parameters that are of interest. This is because each execution of experiments is costly and in turn the variation of each parameter can greatly affect the behavior of the application. Therefore variations are performed on a single parameter at a time to finally find a suitable combination of parameters. The interdependence of parameters is the subject of future work, as well as modeling of behavior determined by the interplay of parameters, which subsequently can be used as knowledge to automate the tuning phase. For the present work we have selected a subset of parameters so as to tune the overall performance of the algorithm [33]:

- Fitness Threshold: indicates the minimum quality expected in the solution that yields the algorithm in at least one individual per generation. The lower the threshold, the more quickly the solution will be found, at the risk of finding a solution far from the optimal one. By contrast, very high thresholds hinder the convergence to the solution.
- Population Size: The population size affects the performance and the efficiency of the evolutionary algorithm. In very small populations, EAs tend to be useless given the lack of variety in the sample of individuals. When growing the size of the population, it is possible to obtain a greater variety of points of the search space at the expense of increasing the number of evaluations needed, slow convergence speed, and even increasing the probability of stagnation at local optima.
- Number of Iterations: controls the level of evolution of the population. The smaller the number of iterations, the faster the termination of the program, at the expense of poor exploration of the search space given by the small evolution of the population. By contrast, a large number of iterations can cause great loss of time when the search is not well oriented, or the migration rate is not enough to expand the search. In this case of study, the number of iterations is limited by the simulation step, so

that the algorithm processes a number of scenarios at most comparable to the amount that would be processed without the evolutionary calibration step of ESS.

The incidence of each of these parameters will be considered during the analysis of results to find the best combination of values in order to achieve better results more efficiently. In the particular case of the application of the method to the prediction of forest fires, the tuning process will allow to evaluate the utility of each parameter in the search for a line of fire that more accurately predicts the actual fire behavior.

As mentioned before, in this work the calibration of the evolutionary parameters of ESS has been carried out following a static or classic tuning. The analysis and comparison of the information collected along the successive executions allows us to determine which could be the best combination of values in the operation parameters of the method, trying to find the results with a certain trade-off between the quality reached and the execution time, but the main focus of this work is related to the improvement of the quality of the ESS prediction in terms of similarity to reality. The aspects strictly related to the improvement of ESS in terms of execution time and efficiency are out of the scope of this article, but they constitute our following object of study to calibrate the method. In particular, we are starting to work in the improvement of the efficiency of the computational resources involved along the execution, with the aim of avoiding the use of unnecessary resources, the overload of some resources or the idleness of some of them. In that case we are considering a dynamic tuning approach to vary or adapt the amount of the underlying computational resources as soon as the data set or the load of the execution environment vary along the time, requiring more or less computational power to cover some minimal efficiency expectations. Along the execution of an application, the data set to be managed could vary in dimensions, such as in the case of forest fires prediction, where the perimeter of the fireline gets bigger as the time pass (and in consequence, the number of calculus necessary to predict the next position of each point proportionally increases), and these variations could cause the overload of the parallel system (when using just a few nodes and the set of data increases) or periods of idleness of it (when using several resources and the data set decreases). The states of overload and idleness could be also provoked by external reasons to the application: an heterogeneous execution environment, or a time-sharing environment. In either case, the availability for computing at each parallel node varies according to the characteristics of the machine and the demand of other users for executing their own applications. This dynamism in the execution conditions difficults to reach to an efficient execution. This is why we are now addressing (out of the scope of this article) the dynamic tuning of the amount of computational resources involved in the execution of ESS, basing on a performance model [34] that describes the optimal number of workers necessary to cover the computational power requirements, without fall into overloaded or idle machines.

Figure 7: Plot533 during the tenth minute of the fire

5 Experimental Results

Below are the results obtained after the experiments on a real case study. For this purpose we used the plot called Plot533, corresponding to a section of land dedicated to controlled burns, which were conducted in the field, located in Serra de Louçã (Gestosa, Portugal). The plot has a width of 95 meters, a length of 123 meters and a slope of 21º (see Fig. 7). Throughout the development of the burnings were defined discrete steps to represent the advance of the fire front. Therefore, were considered different instants of time t_0, t_1,

t_2... etc. For this particular job, such discretization allows the application of the calibration step of the ESS prediction (CS in Figure 6).

Table 1: Results obtained by varying the threshold of fitness (time in seconds)

Fitness Threshold	Execution Time	Speed of Convergence	Quality of Prediction	Step 1	Step 2	Step 3	Step 4
0.55	183.882358	1600	0.397754	0.36409	0.39521	0.386855	0.44486
0.6	375.483171	4000	0.470107	0.374214	0.467302	0.533749	0.505163
0.65	1771.805855	40800	0.560218	0.507692	0.556782	0.601804	0.574595
0.7	1804.738509	41800	0.529901	0.507692	0.556782	0.46036	0.594771
0.73	1813.430580	43000	0.578395	0.507692	0.545578	0.662806	0.597502
0.75	7266.791839	160200	0.690629	0.654605	0.716854	0.721223	0.669834
0.77	7301.521006	163000	0.702796	0.654605	0.716854	0.721223	0.718502
0.8	7767.837747	200000	0.702796	0.654605	0.716854	0.721223	0.718502
0.85	7795.684511	200000	0.702796	0.654605	0.716854	0.721223	0.718502

Table 2: Results obtained by varying the population size (time in seconds)

Population Size	Execution Time	Speed of Convergence	Quality of Prediction	Step 1	Step 2	Step 3	Step 4
50	1467.113662	14450	0.564108	0.470968	0.629565	0.525128	0.630769
75	3199.063588	60150	0.663458	0.635379	0.711538	0.677316	0.629597
100	4223.299043	83400	0.696012	0.688312	0.718121	0.72205	0.655565
125	5433.940280	122125	0.554085	0.23913	0.694143	0.65897	0.624096
150	6473.469596	120150	0.674066	0.704167	0.633663	0.661429	0.697005
175	6616.830495	146300	0.720124	0.755102	0.71619	0.72381	0.685393
200	7266.791839	160200	0.690629	0.654605	0.716854	0.721223	0.669834
300	6609.219695	248400	0.702796	0.654605	0.716854	0.721223	0.718502
400	9673.281719	293200	0.700928	0.730612	0.706573	0.703404	0.663121
500	9412.967396	353000	0.702811	0.654616	0.716831	0.721229	0.718571

The results were obtained by executing both systems on a LINUX cluster (12 processors AMD64 2G RAM and Gigabit Ethernet 1000 Mbps) under an MPI environment, and the experiments were conducted on the basis of a classical configuration of the evolutionary parameters: population composed of 200 individuals, 80% of breeders, 60% probability of crossover, 50% probability of mutation, maximum of 200 iterations per step. The seed for generating the random numbers has been set to a fixed value so as to achieve more comparable experiments in terms of similarity of cases. As mentioned above, several experiments were performed varying the value of a single parameter at a time. The results obtained are presented in Tables 1, 2 and 3, and also shown graphically in Figures 8, 9 and 10, in which it is possible appreciate the difference in the values studied. In each column of these tables are documented the most relevant aspects for this work, which are:

- Case study: the first column identifies the experiment, and the type of parameter that was varied (fitness threshold, population size or number of iterations per step).
- Execution Time: indicates the time (in seconds) to perform the experiment.
- Speed of Convergence: counts the number of simulations that were conducted to obtain the results obtained.
- Quality of Prediction: represents the average quality of the prediction carried out by the method. This is useful to compare the overall quality of the different experiments.

Table 3: Results obtained by varying number of iterations per simulation step (time in seconds)

Number of Iterations	Execution Time	Speed of Convergence	Quality of Prediction	Step 1	Step 2	Step 3	Step 4
100	3365.813957	79000	0.692095	0.699153	0.697826	0.711921	0.659478
150	4907.613884	122000	0.687837	0.717213	0.672269	0.704545	0.657321
200	7266.791839	160200	0.690629	0.654605	0.716854	0.721223	0.669834
250	6998.951520	111800	0.580086	0.240682	0.714617	0.743134	0.621912
300	8109.458388	191400	0.706197	0.7375	0.722892	0.671545	0.692849
400	9727.599406	164200	0.613422	0.39834	0.664122	0.724913	0.666314
500	12426.756830	323400	0.709407	0.7375	0.728311	0.708609	0.663206

- Step 1, ..., Step 4: The last four columns provide complementary information to the column 'Quality of Prediction', because this average value is computed from them. In such columns are shown the quality of prediction achieved by the method ESS in each simulation step, i.e. in four predefined time instants.

Analyzing the results in Table 1 indicates that the increasing in the fitness threshold improves the quality of the predictions. However, such improvement significantly increases also the run time, due to, in order to achieve improvements, the method needs an extensive search. Nevertheless, it also shows that beyond the increase in the execution time, the quality achieved stagnates around a value of 0.70, possibly because it explores all possible scenarios covered by the method without finding more favorable results. Similarly, Table 2 shows that the population growth significantly penalizes time without yielding significant improvements in the quality of prediction, resulting 175 the population size with best results. Finally, Table 3 shows very similar results in terms of quality. According to the three aspects evaluated, the most suitable cases are 200 and 300 iterations as a maximum.

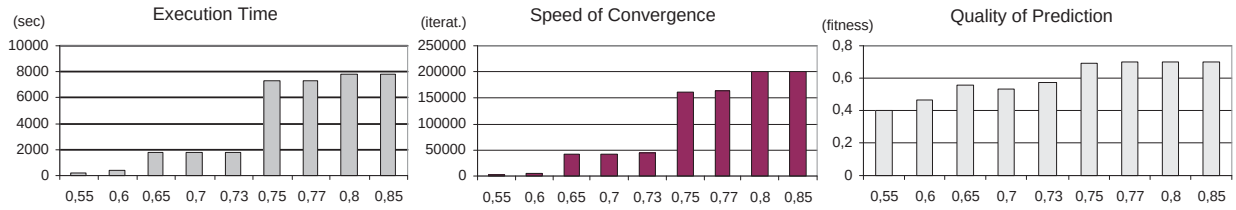

Figure 8: Graphical comparison of the values shown in Table 1 (varying the Fitness Threshold)

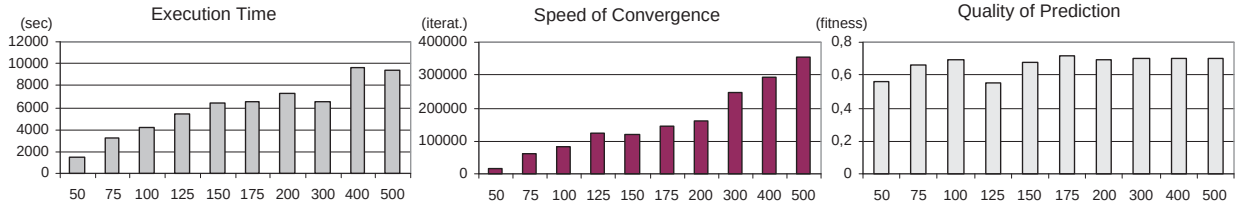

Figure 9: Graphical comparison of the values shown in Table 2 (varying the Population Size)

Given these results, we designed eight new experiments, but this time with the tuned parameters. The new cases generated are the result of the possible combinations of the best values found for each parameter. Such cases, referred to as Case A, Case B, ..., to Case H, are listed in Table 4. The results obtained after application of the generated combinations are presented in Figure 11. It can be observed that considering the execution time, cases that require less time to solve the problem are the B and H, which is directly related to the speed of convergence, because they are the cases that require fewer iterations to find the solution. Moreover, by paying attention to the quality of prediction found, can be seen that the case with higher fitness is A, followed by cases B, C, D and E, which obtain similar and good values. Combining all aspects, we conclude that the case which offers better relation between quality and time is the case B, due to it presents a good level of fitness and it takes a shorter execution time than the other combinations.

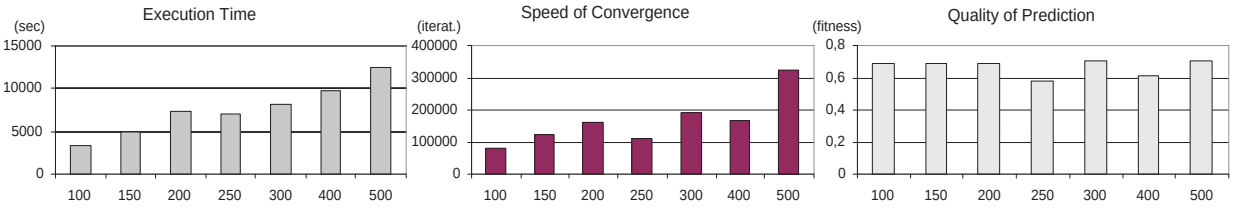

Figure 10: Graphical comparison of the values shown in Table 3 (varying the Iterations Number)

Table 4: Cases resulting obtained from the study of tuned parameters

Case	Fitness Threshold	Population Size	Max. Iterations
A	0.77	300	300
B	0.77	300	200
C	0.77	175	300
D	0.77	175	200
E	0.75	300	300
F	0.75	300	200
G	0.75	175	300
H	0.75	175	200

Given that the case B offers the best trade-off between quality of prediction and execution time, we establish such setting of parameters as the best option for this real plot. At this point, we also designated 6 complementary experiments in order to validate the efficacy of the settings of the case B when applied to other plots. In this set of experiments, we consider the plots 534, 751 and 752 (also from Gestosa controlled burns), whose description and parameters setting is detailed in Table 5. For each one of them, we conducted two different experiments under different scenarios:

- In first place, we executed the application with the classical setting of parameters.
- In second place, we executed the application using the setting of parameters of case B (in Table 5 we denote these cases as *#plot*-B).

The results obtained are presented in Figure 12. Some bar columns of the graphic have an arrow at top, representing that the column bar has a bigger magnitude that the maximum scale in the graphic. Moreover, we include small labels with the value of those big columns.

Table 5: Plot characteristics and parameters settings for each experiment

Plot	Width (m)	Length (m)	Slope (°)	Fitness Threshold	Population Size	Max. Iterations
534	75	126	19	0.7	200	200
534-B	75	126	19	0.77	300	200
751	20	30	6	0.7	200	200
751-B	20	30	6	0.77	300	200
752	20	30	6	0.7	200	200
752-B	20	30	6	0.77	300	200

It can be observed that the settings of the case B provides a subtle improvement in the quality of prediction of the plot 751 in a similar execution time. In contrast, for plots 534 and 752, even though the quality of prediction remains steady, the execution time increases considerably, because the increase in the population size and the fitness threshold. Such settings require a considerably bigger amount of evaluations and in consequence the processing time is also increased, but a deeper search does not always allow for reaching an improvement in the prediction, given the particularities of each terrain. This constitutes a key and encouraging factor to address the dynamic tuning of ESS, in order to make the system more flexible, according to the own characteristics of each case of study.

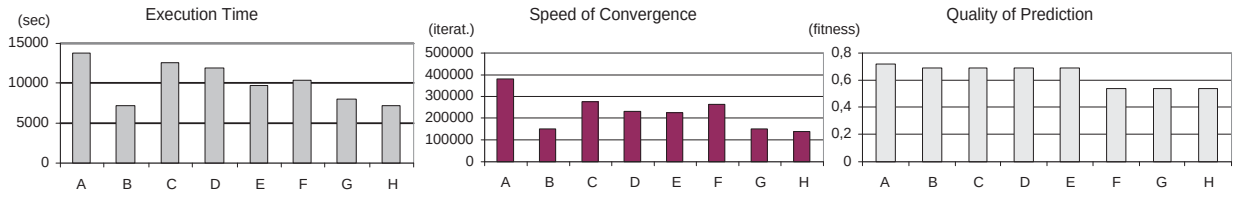

Figure 11: Results obtained for the case studies with tuned parameters

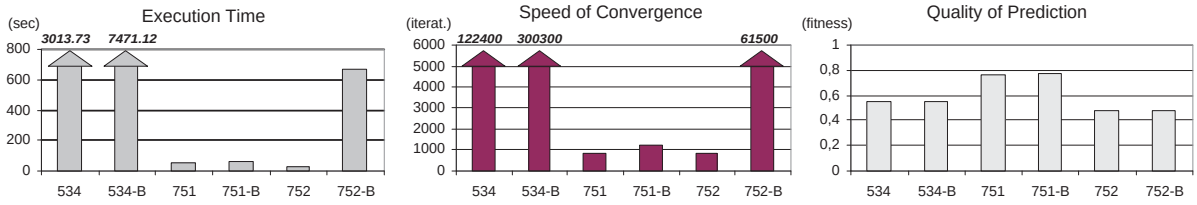

Figure 12: Results obtained for the Plots 534, 751 and 752 (classical parameters and case B)

6 Conclusions

The behavior of an evolutionary algorithm for a given optimization problem is usually addressed by their operators (crossover, mutation, selection, etc.) and selected values for the parameters of the algorithm itself. Calibration of these parameters is a difficult task because they can take different values and thus the number of possible combinations could be very high. Therefore, when using an EA, there is an additional optimization problem that needs to be solved: the calibration of evolutionary algorithm parameters, i.e. the analysis of the operating parameters of an evolutionary algorithm is a process required for its use, since its effectiveness and behavior are very dependent of the problem.

In this paper, we have performed an analysis of the calibration of some of the parameters pertaining to the stage that operates with evolutionary algorithms in ESS. In particular, we considered parameters such as the fitness threshold, the population size and the maximum number of iterations. From the experiments, we obtained a set of cases resulting from the combination of the best values found for each parameter. From these we can conclude that a size population comprised 300 individuals with a maximum number of iterations set in 200 and a fitness threshold equal to 0.77, can reach a solution that satisfies the quality requirements expected in a runtime not excessive for the burning of the plot under study. Moreover, the values used in this case are broadly consistent with the values specified as classics (parameters values are considered classic when such values work for most application problems, allowing to reach solutions that at least meet the minimum requirements of quality). However, the complementary experiments show that some adjustments in the parameters could be needed when the system with the setting of case B is applied to other study cases. Therefore, it is necessary the incoporation of dynamic tuning to ESS as evidenced by these results. Finally, beyond the specific results found, it is interesting to see the impact on the output of the system (both in terms of quality of prediction as in performance of system) the variation of the parameters under analysis, which demonstrates the importance of tuning and calibration stage in this kind of systems. We have already been working in the tuning of the quality of prediction of ESS, and it is now necessary to address the quality of the execution. Our following challenge is then provide ESS with new abilities to be also benefitted by dynamic tuning in order to improve the efficiency with which resources are used, according to the variable execution conditions given by the data size or the load of the system.

Acknowledgment

This work was supported by project PIP 11220090100709 (CONICET), project UTN1194 and UTN1585 (UTN), and by project PICT PRH 2008-00242 (ANPCyT).

References

[1] P. Morgan, C. Hardy, T.W. Swetnam, M.G. Rollins, D.G. Long, “Mapping fire regimes across time and space: Understanding coarse and finescale fire patterns”, *International Journal of Wildland Fire*, 10, pp. 329–342, 2001.

[2] Diario Clarín. (2013) "Se reavivó el fuego en Córdoba". [Online].
Available: http://www.clarin.com/sociedad/reavivo-fuego-Cordoba_0_995900784.html

[3] El Diario del Fin del Mundo. (2012) "Sector Este controlado, el Oeste sigue complicado". [Online].
Available: http://www.eldiariodelfindelmundo.com/noticias/leer/40784/sector-este-controlado-el-oeste-sigue-complicado.html

[4] M.A. Cochrane, *Se extienden como un reguero de pólvora*, Publicado por el Programa de las Naciones unidas para el Medio Ambiente (PNUMA), 2002.

[5] G.L. Ball, D.P. Guertin, "FIREMAP - fire and the environment: ecological and cultural perspectives", Knoxville, TN, USDA Forest Service, pp. 215–218, (Ashecille, NC), 1991.

[6] G. Wallace, "A numerical fire simulation model", *Int. J. Wildland Fire*, 3 (2), pp. 111–116, 1993.

[7] M.A. Finney, "FARSITE: Fire Area Simulator-model development and evaluation", Res. Pap. RMRS-RP-4, U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, pp. 47, (Ogden, UT), 1998.

[8] J. Jorba, T. Margalef, E. Luque, J. Campos da Silva André, D.X. Viegas, "Parallel Approach to the Simulation of Forest Fire Propagation", *Proc. 13 Internationales Symposium "Informatik für den Umweltshutz" der Gesellshaft Für Informatik (GI)*, pp. 69–81, 1999.

[9] P.L. Andrews, C.D. Bevins, R.C. Seli, "BehavePlus Fire Modeling System, Version2,0: User's Guide", Gen. Tech. Rep. RMRS-GTR-106WWW, Department of Agriculture, ForestService, Rocky Mountain Research Station, Ogden, UT, 2003, pp. 132, 2003.

[10] A.M.G. Lopes, M.G. Cruz, D.X. Viegas, "FireStation - An integrated software system for the numerical simulation of wind field and fire spread on complex topography", *Environmental Modelling & Software*, 17 (3), pp. 269–285, 2002.

[11] L.M. Ribeiro, D.X. Viegas, A.G. Lopes, P. Mangana, P. Moura, "Operational application of a decision support tool in fire management in Portugal", *Forest Ecology and Management*, 234 (Supplement 1), pp. S243, 2006.

[12] B. Abdalhaq, *A methodology to enhance the Prediction of Forest Fire Propagation*, Ph. D Thesis, Universitat Autònoma de Barcelona (Spain), 2004.

[13] K.J. Beven, J. Freer, "Equifinality, data assimilation, and uncertainty estimation in mechanistic modelling of complex environmental systems", *Journal of Hydrology*, 249, pp. 11–49, 2001.

[14] J. Mandel, L.S. Bennethum, M. Chen, J.L. Coen, C.C. Douglas, L.P. Franca, C.J. Johns, M. Kim, A.V. Knyazev, R. Kremens, V. Kulkarni, G. Qin, A. Vodacek, J. Wu, W. Zhao, A. Zornes, "Towards a Dynamic Data Driven Application System for Wildfire Simulation", LNCS, 3515, pp. 632–639, 2005.

[15] S. Thorndahl, K.J. Beven, J.B. Jensen, K. Schaarup-Jensen, "Event based uncertainty assessment in urban drainage modelling, applying the GLUE methodology", *Journal of Hydrology*, 357 (3-4), pp. 421–437, 2008.

[16] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, 1997.

[17] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, 220 (4598), pp. 671-680, 1983.

[18] V. Cerný, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm", *Journal of Optimization Theory and Applications*, 45, pp. 41–51, 1985.

[19] D.E. Goldberg, "Genetic and evolutionary algorithms, Come of age", *Communications of the ACM*, 37(3), pp. 113–119, 1994.

[20] C. Darwin, *On the Origin of Species by Means of Natural Selection*, Murray, London, 1859.

[21] K. Beven, A. Binley, "The future of distributed models: model calibration and uncertainty prediction", *Hydrological Processes*, 6, pp. 279–298, 1992.

[22] C.P. Robert, G. Casella, *Monte Carlo Statistical Methods*, Springer, 2004.

[23] J. Piñol, R. Salvador, K. Beven, "Model Calibration and uncertainty prediction of fire spread", *Forest Fire Research & Wildland Fire Safety*, On CD-ROM, Millpress, 2002.

[24] G. Bianchini, *Wildland Fire Prediction based on Statistical Analysis of Multiple Solutions*, Ph. D Thesis, Universitat Autònoma de Barcelona (Spain), 2006.

[25] G. Bianchini, M. Denham, A. Cortés, T. Margalef, E. Luque, "Wildland Fire Growth Prediction Method Based on Multiple Overlapping Solution", *Journal of Computational Science* (JOCS), 1 (4), pp. 229–237, 2010.

[26] G. Bianchini, M. Denham, A. Cortés, T. Margalef, E. Luque, "Improving forest-fire prediction by applying a statistical approach", *Forest Ecology and Management*, 234 (supplement 1), pp. S210, 2006.

[27] D. Montgomery, G. Runger, *Probabilidad y Estadística aplicada a la ingeniería*, Limusa Wiley, 2002.

[28] T. Mattson, B. Sanders, B. Massingill, *Patterns for Parallel Programming*, Addison-Wesley, 2004.

[29] G. Bianchini, P. Caymes Scutari, "Uncertainty Reduction Method Based on Statistics and Parallel Evolutionary Algorithms", Proceedings of High-Performance Computing Symposium - 40 JAIIO (HPC 2011, ISSN: 1851-9326) pp. 1–4, 2011.

[30] E. Cantú Paz, "A Survey of Parallel Genetic Algorithms". *Calculateurs Parallèles, Reseaux et Systems Repartis*, 10(2), pp. 141–171, 1998.

[31] K. Naono, K. Teranishi, J. Cavazos, R. Suda, *Software Automatic Tuning - From Concepts to State-of-the-Art Results*, Springer, 2010.

[32] R. Buyya, *High Performance Cluster Computing - Architectures and Systems*, Prentice Hall, 1999.

[33] J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms", *IEEE Transactions on Systems, Man and Cybernetics*, 16 (1), pp. 122-128, 1986.

[34] E. César, J. Sorribes, E. Luque, "Modeling Master-Worker Applications with POETRIES", IEEE 9th International Workshop HIPS 2004, IPDPS, pp. 22–30, 2004.