Universitat Autònoma de Barcelona

Wildland Fire Prediction based on Statistical Analysis of Multiple Solutions

Departament d'Arquitectura d'Ordinadors

i Sistemes Operatius

Thesis submitted by **Germán Bianchini**
in fulfillment of the requirements for the degree of Doctor per la Universitat Autònoma de Barcelona

Barcelona (Spain), July 2006

Wildland Fire Prediction based on Statistical Analysis of Multiple Solutions

Thesis submitted by **Germán Bianchini** in fulfillment of the requirements for the degree of Doctor per la Universitat Autònoma de Barcelona. This work has been developed in the Computer Architecture and Operating Systems Department of the Universitat Autònoma de Barcelona and was advised by Dra. Ana Cortés Fité and Dr. Tomàs M. Margalef Burrull,

Bellaterra, July 2006

Thesis Advisor

Ana Cortés Fité

Tomàs M. Margalef Burrull

Acknowledgments

Firstly, I want to thank to my colleagues from DACSO. The help given by them always has been very useful. Thanks to Baker and Mónica for her participation in the experiments, and to Xiao for saving me when I was in a tight spot (talking in scientific terms). I also would like to thank to Ana Cortés and Tomàs Margalef for their corrections and guidance. Thank to Emilio Luque for his help and opinions.

Many thanks to the technical staff, Daniel Ruiz and Jordi Valls, for their job (very difficult job, really): solving the problems in the cluster and to help us to solve technical problems.

I also want to express my gratitude to the professor Domingos Xavier Viegas and his group of collaborators to allow us to carry out experiments in the laboratories of Lousã and by his hospitality during my stay in Coimbra.

It is true that phrase which says: "the friends can be counted with a single hand". I feel that my handful of friends helped me every day. Some of them are very far from Barcelona, but I feel their presence. I met new friends in Spain also. With them I shared many hours and ideas. I think that this cultural interchange has been very interesting for everybody not only because of the scientific aspect but also because of our development as people.

Luckily, one of my best friends has been to my side all this time: my wife Paola. Only she knows about the large hours of work, the tiredness, the sadness, and happiness which accompanied this doctoral thesis. Everything has been better thanks to her help, opinions and comradeship. Thanks, my dear Paola.

My wife, my parents and my brother are my family. I dedicate this work to them. Thanks.

Contents

List of Figures

List of Tables

Preface

In many different scientific areas, the use of models to represent the physical system has become a common strategy. These models receive some input parameters representing the particular conditions and provide an output representing the evolution of the system. Usually, these models are integrated in simulation tools that can be executed on a computer.

A particular case where models are very useful is the prediction of Forest Fire propagation. Forest fire is a very significant hazard that every year provokes huge looses from the environmental, economical, social and human point of view. Particularly dry and hot seasons seriously increase the risk of forest fires in the Mediterranean area. Therefore, the use of models is very relevant to estimate fire risk, and predict fire behavior.

However, in many cases models present a series of limitations. Usually, such limitations are due to the need of a large number of input parameters. In many cases such parameters present some uncertainty due to the impossibility to measure all of them in real time and must be estimated from indirect measurements. Moreover, in most cases these models cannot be solved analytically and must be solved applying numerical methods that are only an approach to reality (still without considering the limitations that present the translations of these solutions when they are carried out by means of computers).

In the particular case of forest fire propagation simulators, there are several input parameters that present a dynamic behavior and they cannot be estimated in a precise way in real time. Therefore, such simulators provide erroneous predictions in many cases due to this imprecision in input parameter estimation.

This is a significant drawback that should be overcome to improve the predictions. The most promising approach to reach this goal is to use real time data assimilation and apply some computational method to analyze the deviation of the prediction from the real behavior, determine the values of the parameters that reproduce the behavior of the fire and use these values in the next simulation step.

Several methods based on data assimilation have been developed to optimize the input parameters. In general, these methods operate over a large number of input parameters, and, by mean of some kind of optimization, they focus on finding a unique parameter set that would describe the previous behavior in the best form. Therefore, it is hoped that the same set of values could be used to describe the immediate future.

However, this kind of prediction is based on a single value of parameters and, as it has been said above, for those parameters that present a dynamic behavior the new optimized values cannot be adequate for the next step.

The objective of this work is to propose an alternative method. Our method, called Statistical System for Forest Fire Management, is based on statistical concepts. Its goal is to find a pattern of the forest fire behavior, independently of the parameters values. In this method, each parameter is represented by a range of values with a particular cardinality for each one of them. All possible scenarios considering all possible combinations of input parameters values are generated and the propagation for each scenario is evaluated. All results are statically aggregated to determine the burning probability of each area. This aggregation is used to predict the burned area in the next step.

To validate our method, we use a set of real prescribed burnings. Furthermore, we compare our method against two other methods. One of these methods was implemented by us for this work: GLUE (Generalized Likelihood Uncertainty Estimation). It corresponds to an adaptation of a hydrological method. The other method (Evolutionary method) is a genetic algorithm previously developed and implemented by our research team.

The proposed system requires a large number of simulations, a reason why we decide to use a parallel-scheme to implement them. This way of working is different from traditional scheme of theory and experiment, which is the common form of science and engineering. The scientific computing approach is in continuous expansion, mainly through the analysis of mathematical models implemented on computers. Scientists and engineers develop computer programs that model the systems under study. This methodology is creating a new branch of science based on computational methods that is growing very fast. This approach is called Computational Science.

Chapter 1

Introduction

> "The one advantage of playing with fire, Lady Caroline, is that one never gets even singed."
>
> *A woman of no importance*, Oscar Wilde

Using computers, scientists and engineers have made a large number of discoveries that they would not have made otherwise. During the last years, computers have revolutionized the way that many scientists do their work.

If we need to solve a simple equation or system of two equations and two unknowns, we do not precise a computer. However, we cannot solve a problem with millions of variables by hand [67].

Habitually, science was done in a laboratory environment as a combination of theory and physical experimentation (including hand calculations), but computers have made possible a new and powerful way of doing science —numerical simulation— that augments the old. Numerical simulation is the name for the process of modeling mathematically a physical phenomenon, and then running an experiment with the mathematical model. Computational mathematicians or computational scientists play an important role in this new way of doing science, creating, evaluating, and refining the mathematical models used to simulate the physical phenomena.

Simulation can be used when physical experiments are too costly, time consuming, dangerous, or even impossible. This way of work becomes a new way of doing science known as computational science.

1.1 Computational Science vs. Computer Science

Every day we witness how the advances grow in computer technology. At the same time, demand grows for high performance computers. In different areas, but mainly in the scientific field, a greater requirement is in increase. Most of the problems that scientists work on involve vast amounts of data and a large number of variables. This type of requirements are hardly provided by any sequential computer. This situation drove to the science evolution to a new step that has been called Computational Science. According to Tapia and Lanius it is possible to define "Computational Science" [67] as:

> *[...] an interdisciplinary field at the intersection of three domains: mathematics, computer science, and the biological and physical sciences. The computational scientist uses tools from computer science and mathematics to study problems from physical science, social science, engineering, etc.*

In figure 1.1 we can see where it is located this field respect to other sciences.

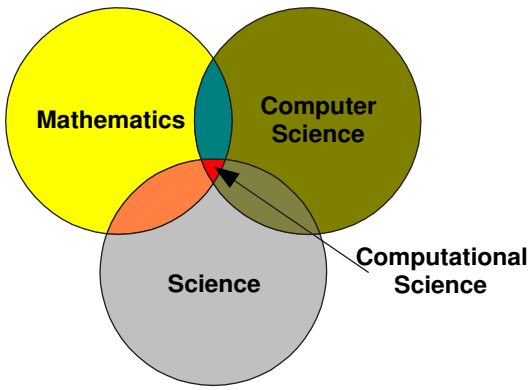

Figure 1.1: Interaction among Mathematics, Computer Science and Biological and Physical sciences [67]

Through the advances in computer technology and numerical methods, mathematicians and scientists are able to work together modeling and solving problems that were impossible to address several years ago.

Computational scientists do more than use computers to find good solutions to mathematical models developed from scientific problems, they develop new mathematical tools and theory and develop new numerical methods and they also improve the accuracy and speed of existing methods.

Although, Computational Science and Computer Science have very similar names, they refer to different areas of study. On one hand, Computational science uses knowledge and tools from both mathematics and computer science, but its main focus is designing tools that will be used on the computer to solve a scientific or engineering problem. On the other hand, Computer Science focuses on both the theory and design of computers and the phenomena depending on them. Some examples are distributed programming languages, systems, networking, software, parallel processing, etc.

1.1.1 Grand Challenge Problem

According to [11] a Grand Challenge Problem "*is a general category of unsolved problems*". However, this definition has a certain degree of inherent subjectivity surrounding what is, or is not, a Grand Challenge. In general, a Grand Challenge problem exhibits at least the following characteristics:

- The problem is demonstrably hard to solve, requiring several orders-of-magnitude improvement in the capability required to solve it.
- The problem cannot be unsolvable. If it probably can't be solved, then it can't be a Grand Challenge. Ideally, quantifiable measures that indicate progress toward a solution are also definable.
- The solution to a Grand Challenge problem must have a significant economic and/or social impact, such as biomedicine, the environment, economic competitiveness, and national security.

Many of the original Grand Challenge Problems involved simulations that, at the time the act passed, could not be completed fast enough or with enough accuracy.

Some of the Grand Challenges currently underway deal with climate modeling, convective turbulence and mixing in astrophysics, computational

biology, geophysical databases, condensed matter physics, and binary black holes. They also address problems in areas such as flow modeling, quantum chromo dynamics, ground water remediation, and contaminant containment.

1.1.2 HPC, Parallel Computing

A common theme in parallel computing, and in grand challenges in particular, is the use of the computer simulation paradigm. Computer simulation involves the use of a mathematical model to simulate a real world situation or problem, and then the use of computers to calculate the results of these mathematical models. Often the use of a simulation has many characteristics that makes it much more desirable than actually performing an experiment in the real world.

An even more interesting goal is to provide accurate solutions to these computer models in a reasonable period of time. A reasonable period of time can range from several days in certain models to several hours in other cases, as for example weather prediction.

An obvious way to make simulations run faster on a computer is to design faster computers, but faster isolated computers are not the solution. This method has specific limitations determined by light speed, thermodynamics laws and economic cost. Then, it is reasonable to present an alternative to this problem. The design of the algorithms that run on the computer is equally important, if not more. Inefficient algorithms can quickly cancel out faster computers.

Another option is based on using multiple computational resources working together. This class of systems are called parallel and distributed systems. This kind of systems allow to share tasks among different processors with the goal of reducing time [22].

As a direct consequence of the above mentioned systems appears the parallel process, which is based on the idea of "divide & conquer", it means to divide big problems into little sub problems that could be resolved in a parallel way.

Another achievement that supposes a step forward to the revolution of scientific problems is distributed computation. Under this scenario, a set

of connected computers work together on solving an individual large-scale problem by using some kind of middleware.

The most important factor in distributed computation is cost. Big MPP's (Massivelly Parallel Processors) are commonly very expensive, however, executing the same problem in a set of local computers is not very expensive. Nevertheless, a common element in the architecture between MPP's and distributed computation is the notion of message passing. This is a very known paradigm among others (as for example shared memory, parallelisant compilers, etc.) because it is supported by different multiprocessor and it is used by a lot of software and hardware systems [49].

As it was mentioned before, communities that need efficient fast computers are usually found in scientific environment, medical environment, academic environment, etc. In these areas, many people need to do huge calculus that could not be afforded using the traditional computation model and, the best way to treat with this requirement is using parallel process computers.

The main reasons why the mentioned disciplines require great resources and capacity of processing, are to obtain numerical solutions of complex problems and to use computerized simulations.

Essentially, simulations offer several advantages:

- Computer simulations are cheaper and faster than physical experiments.
- Computers can solve a wider range of problems than specific laboratory equipment can.
- Computational approaches are only limited by computer speed and memory capacity, while physical experiments have many practical constraints.

It is interesting to note that the majority of the top 500 supercomputers [68] are running one of the applications listed below which need parallel processing for obtaining a good performance.

- **Predictive Modelling and Simulations**: Multidimensional modelling of the atmosphere, the earth environment, outer space, and the

world economy has become a major concern of world scientists. Predictive Modelling is done through extensive computer simulation experiments, which often involve large-scale computations to achieve the desired accuracy and turnaround time.

- **Engineering Design and Automation**: Supercomputers have been in high demand for solving many engineering design problems. Industrial development also demands the use of computers to advance automation, artificial intelligence, and remote sensing of earth resources.

- **Energy Resources Exploration**: Computers play an important role in exploration of energetic resources: the discovery of oil and gas, the management of their recovery, development of workable plasma fusion energy and ensuring nuclear reactor safety. Using computers in the high-energy area results in less production costs and higher safety measures.

- **Medical, Military and Basic Research**: In the medical area, fast computers are needed in computer-assisted tomography (CAT scan), imaging, artificial heart design, liver diagnosis, brain damage estimation and genetic engineering studies. Military defence needs to use supercomputers for weapon design, effects simulation, and other electronic warfare. Almost all basic research areas demand fast computers to advance their studies.

- **Visualization**: Applications of this type are for example: Graphics generated by computers for animations or films, complex or voluminous data visualization, etc.

1.2 Forest Fire Prediction

Forest fires are a very serious hazard that, every year, causes significant damage around the world from the ecological, social, economical and human point of view [48]. These hazards are particularly dangerous when meteorological conditions are extreme with dry and hot seasons or strong wind. For example, fire is a recurrent factor in Mediterranean areas. In summer 2003

(a) (b)

Figure 1.2: (a) Volunteers work to extinguish forest fires in Portugal [29] (b) Castelo Branco Covihla (Portugal) [38]

the weather was very hot in this area and, in Portugal alone 420,000 hectares were burned and 20 people died (figure 1.2 shows volunteers fighting against fires). In October of the same year, the strong wind caused a large fire in California that burned 300,000 hectares, destroying 3,361 houses and killing 26 people. Spain also suffered the effects of devastating forest fires: in 2005 this country was victim of 17 big forest fires (figure 1.3).

(a) (b)

Figure 1.3: (a) Spain suffers 17 big forest fires in 2005 [29] (b) A forest fire kills 11 members of an extinction equipment in Spain (Photo Reuters) [74]

The causes that produce forest fires are many, and the great majority is related to one or another form of human factors (more than 90% of forest fires are provoked by human action); in addition, fires in degraded forests are worse than those that happen in the intact forests [28].

The forest fires usually can be exacerbated or usually are the result of:

- Deforestation (degraded zones)
- Drought and/or heat
- Previous fires
- Waste, bonfires or burn of rubbish
- Premeditated wildfire
- Accidents

The phenomenon of forest fires has not only given as a result an important loss of forests and damage in the economy, but it also has seriously affected human health and environment. The fire fighting should have at its disposal the most advanced resources and tools to help the use of available resources in the most efficient way to diminish fire effects as much as possible. It can be classified in:

- Prevention
- Fire management
- Forecast
- Detection and monitoring
- Fight against wildfire

Simulation of forest fires propagation is an important problem from the computational point of view due to the complexity of the involved models, the necessity of numerical methods and the required resources for calculation. Therefore, we can talk about a **Grand Challenge Problem**.

Reduction of those negative effects of fire requires to improve current fire risk assessment methods. In this context, the kind of systems presented in this work (forest fire simulators and alternative methods to apply them) becomes a very important tool for the authorities to prevent these accidents. Considering these methods is very important, because they can provide more

complete information to determine the possible behavior of a wildland fire and to determine those regions where an ignition is more dangerous.

This danger depends on static factors such as the slope of the terrain or the vegetation type in that particular region, but also on certain dynamic factors such as the moisture content in the vegetation or wind conditions. Since it is not possible to previously determine the current conditions under which a fire starts, it is not possible to evaluate beforehand the behavior or the effective rate of fire spread, or inclusive flame intensity in a real situation.

Several propagation models have been developed to predict fire behavior. These models can be used to develop simulators and tools for preventing and fighting forest fires [9, 10, 30, 31, 42]. These models require a set of input parameters, including vegetation type, moisture contents, wind conditions and so on, and provide the evolution of the fire line in simulation steps. However, this work focuses on the consideration that there is no set of input parameters to be applied to the propagation model because, as has been observed, it is not possible to know the value of each parameter when a fire starts.

Considering this uncertainty, the methods studied in this work, try to determine the possible fire behavior based on different principles: **evolutionary algorithms**, **uncertainty prediction** and **statistical analysis**.

1.2.1 Classical Prediction: Problems

Within the different areas of science we have mentioned, we found out that in many fields of environmental science the use of models has been increased to carry out predictions. The models offer a series of limitations due to the lack of precision or because of an illegal validation. Some of these restrictions can produce dramatic consequences. On the other hand, limitations are often imposed by the models. Commonly to operate they need an important number of input variables, and because of the mathematical model's own characteristics, these variables usually exhibit some degree of uncertainty. In addition, numerical solutions sometimes are only one approach to reality. Here it is where simulators (computation programs that represent some given model) arise as another tool to provide some facilities during the analysis

of the model characteristics. However, if we consider the limitations that present the carried out solutions by means of computers, we see that the breach between reality and the result of the simulation becomes even greater. Therefore, validation of models, as well as the verification and calibration become a truly arduous task.

As we mentioned previously, models require static parameters (topology of the land), parameters that can change very slowly (type of vegetation or also called "fuel"), parameters that can change most frequently (moisture, amount of load) and parameters that are completely dynamic (wind conditions). The precision of these parameters is the important point in prediction of the behavior, and in many cases it is impossible to carry out some type of measurement, and still worse in case it is not possible to consider the parameter in a real situation.

The prediction of the behavior can not be reliable due to the difficulty at the moment of estimating parameters, and the result of a single simulation can not be a reasonable base to make a decision. This is known as **Classical Prediction**.

Classical Prediction simply consists of using any existing fire simulator to evaluate the fire position after a certain time. The simulator is fed with all the required parameters (vegetation, meteorological conditions, ignition point, etc.). Then, the simulator is executed to predict the fire line after a certain period of time. Some examples of Classical Prediction are [9, 10, 30, 31, 42, 44]

Without this set of information the simulator will not properly work. Generally, the prediction obtained with this approach disagrees with reality. One reason for the disagreement between real and simulated propagation stems from the difficulty of feeding the model with accurate input values. Uncertainties in the input variables can have a substantial impact on the result errors and should be considered.

This classical approach is depicted in figure 1.4. In this scheme, FS corresponds to the underlying fire simulator, which will be seen as a black-box. $RFL0$ is the real fire line at time $t0$ (initial fire line), whereas $RFL1$ corresponds to the real fire line at $t1$. If the prediction process works, after executing FS (which should be fed with the corresponding input parameters

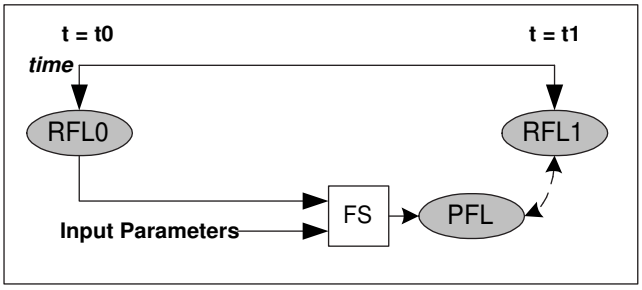

Figure 1.4: Classical prediction of wildland fire propagation

and $RFL0$) the predicted fire line at time $t1$ (PFL) should coincide with the real fire line ($RFL1$). Nevertheless, due to the complexity of the fire behavior modeling, the traditional prediction scheme fairly matches the real fire propagation, but the result is not as similar to reality as desired.

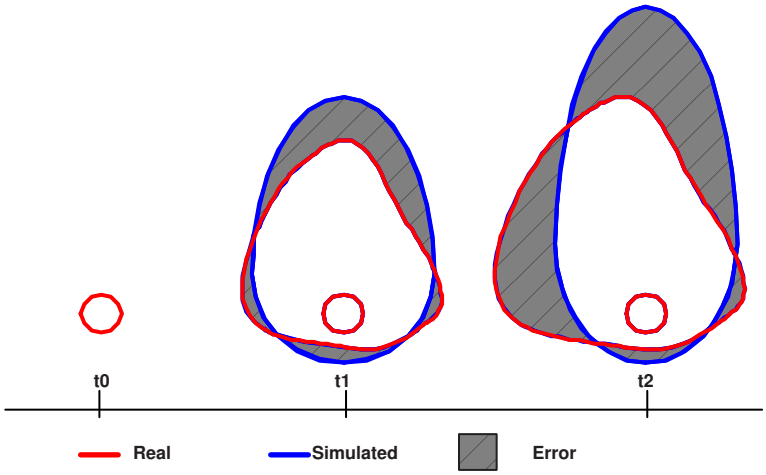

Figure 1.5: Error using classical prediction

In figure 1.5, we illustrate how classical prediction works by showing the prediction provided by this method in two time instants. Although the propagations depicted in figure 1.5 are only hypothetical cases, they clearly illustrate the error incurred by the classical prediction method.

1.2.2 Data Driven Prediction

According to the form of operating of forest fire prediction methods, we can make a classification of such methods according to figure 1.6. In a first level, we have the case previously commented: the *Classic Prediction Method*. We saw that this method simply consists of using any existing fire simulator behavior to evaluate the position of the fire after a certain initial period. It is necessary to feed the simulator with all the required parameters (weather data, vegetation, etc.). Next, the simulator is put into operation to predict the fire line after a certain time interval.

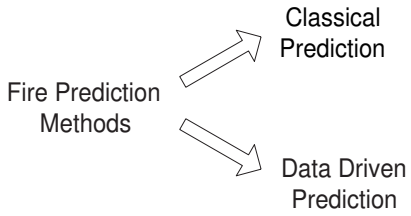

Figure 1.6: Forest fire methods classification

The classic prediction exhibits certain limitations, and the error that takes place when it is applied not only comes from the problems of the model (errors due to a poor definition of the conditions of limit and also to input data badly defined, associated errors to the measures used in the calibration of the model, errors due to the deficiencies in the structure of the model, etc.) but also from the implementation of the simulator. Even supposing that we had a perfect simulator, we would conclude that the large source of errors comes from the lack of precision of the entrance parameters, because we would have serious difficulties to provide the simulator high quality parameters values. There are certain parameters that cannot be measured directly, a reason why they must be estimated from indirect measures (an example is moisture content in vegetation). Wind offers a different situation to us: it can be measured in definite zones, but it is finally necessary to interpolate these values to be able to apply it to all the land. Furthermore, we have a problem with the wind, for citing only one case: it dynamically changes its values during the course of the fire, which modifies the previously calculated values.

As a second option within the classification, we found the methods of Data Driven prediction. Under this name we try to group those methods that, in search of a solution to the problem manifested by the classic methods, make use of optimization techniques with the purpose of calibrating the input parameters set. The optimization process objective is to find a set of input values that, if they feed the simulator, would describe the best form of the previous behavior. Therefore, we would hope that the same set of values could be used to describe the best possible form for the immediate future.

Schematically, the Data Driven methods operate in a stage that we will call Calibration Stage (CS box). In figure 1.7 we can appreciate this idea.

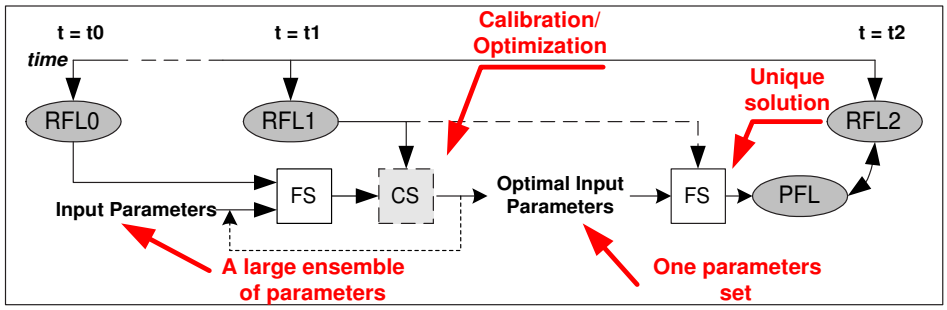

Figure 1.7: Data Driven prediction of wildland fire propagation

Fundamentally, beyond the type of calibration or optimization which the methods make in CS box, this class of methodologies agree in one point: they make the process look for a unique set of values (parameters set) to do the prediction in the following time. Therefore, we can say that Classical prediction methods and Data Driven methods give only one solution.

1.2.3 Thesis contribution

Our proposal is to extend the given classification, offering a new method which we called Statistical System for Forest Fire Management (S^2F^2M), which will be in a new branch of Data Driven methods with *Multiple Overlapping Solution*. In figure 1.8 we can see there is a new branch opened because the proposed method, although it belongs to the Data Driven, generates prediction based on the totality of the proposed cases.

The S^2F^2M objective is to look for a behavior pattern of the forest

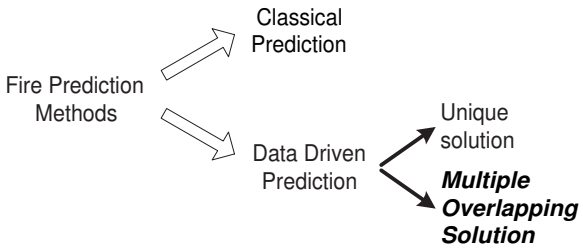

Figure 1.8: Forest fire methods: New classification

fire, independently of the parameters values. For this reason, the method operates with a large number of parameters sets, which we have denominated scenarios, and statistically tries to describe the behavior of the real fire in a near past. In this way, we hope that this behavior pattern stays still within a certain limit of time in order to use this criterion within such limit and to obtain a future prediction very similar to reality. Therefore, our method is based on a general behavior, obtained from the consideration of all scenarios evaluated. Thus, it proposes an alternative to those methods centered in strict observation of the values of parameters that take part in the simulation and it presents an option to the resolution of the problem by means of the use of multiple cases instead of doing a prediction from only one set of parameters.

In figure 1.9 we can observe how the operation scheme differs from the traditional Data Driven methods. The text in red indicates the fundamental differences. As in figure 1.7, the stage in charge of the processing of the method has been labeled CS box. This has been done to avoid details and only showing the stages in a conceptual form.

S^2F^2M considers at any moment the total set of scenarios to carry out the search of the forest fire behavior. Unlike the methods of unique solution, S^2F^2M does not make distinction between better and worse cases (or at least it does not try to make any type of classification), because the interesting fact of its methodology is that every case, every scenario, can contribute to find that the final result, the found pattern, is better. Finally, between the contributions of the present work, we also presented a comparison between two methods of Data Driven prediction and our method, with the objective

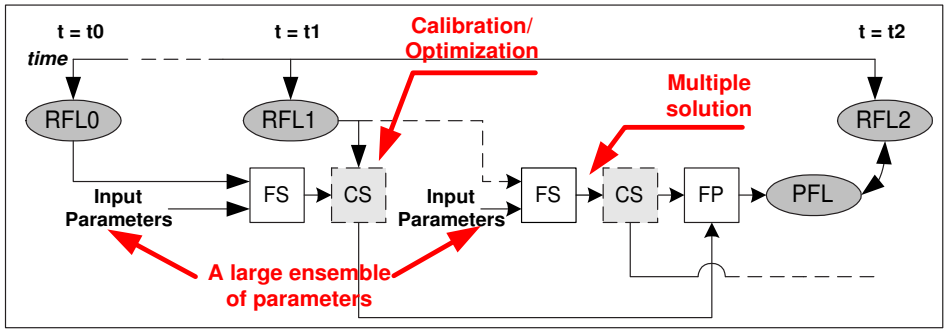

Figure 1.9: Data Driven prediction of wildland fire propagation using Multiple solution

of showing the effectiveness of our proposal of Data Driven prediction with Multiple Overlapping Solution during the application on real forest fires.

1.3 Work organization

This work is organized as follows: In the following chapter we give a general description about Fire Simulator Systems and summarize the main aspects of Rothermel Model. Then, we analyze some of the most important forest fire simulators at the present time. Finally, we comment some characteristics about the selected simulator to apply the methods to compare (Chapter 2).

In Chapter 3, we describe two Data Driven Prediction Methods. We will describe the method and implementation for each one of them.

In Chapter 4 we describe the methodology of statistical method and also explain its implementation. Furthermore, as a first example of experimental study, we show the results obtained when we apply the method on a synthetic forest fire.

Chapter 5 is dedicated to depict the experiments (terrain, platform, etc.). We compare the obtained results after applying the three methods on five different real forest fires.

Chapter 6 describes the results obtained after applying the statistical method on real situations (for example, generating risk maps).

In Chapter 7, we discuss possible ways to improve the statistical method to obtain better results and/or to arrive the result in a shortest time. In the

same manner, we explain the problems found when we apply these variants.

Finally, the main conclusions and open lines that can extend this research are reported in Chapter 8.

Chapter 2

Forest Fire Simulator

> "No pretendas apagar con fuego un incendio, ni remediar con agua una inundación."
>
> Confucio

A Forest Fire Spread Simulator is the last echelon of the stair composed by the construction of a mathematical model, the design of a numerical algorithm to solve the model and, finally, building the computer code that implements the algorithm. Once all this process is finished, we are in front of a forest fire spread simulator. In this chapter, we will describe the five most outstanding and used forest fire simulators at the current time. All of them are based on the propagation model proposed by Rothermel in 1972 [61]. This model is classified as a semi-physical model because it combines physical techniques and empirical correlations obtained from experimental data. The results of this type of model are trustworthy, although they have a rank of validity limited within the specific conditions in which the experiments were carried out. In the following section, we will analyze the general structure of a fire spread simulator. Next, we will introduce the basic equations of a Rothermel model and, subsequently, the BehavePlus, FARSITE, FOFEM, NEXUS and fireLib simulators will be described according to the general structure previously commented.

2.1 A General Fire Simulator System

The global and local models (these two models consider two different scales: the global model considers the fire line as a whole unit —geometrical unit— that evolves in time and space; the local models consider the small units —points, cells, etc.— that constitute the fire line) and the required environmental information must be integrated to obtain a simulation system that provides the space-time forest fire evolution. A scheme of an ideal structure is shown in figure 2.1.

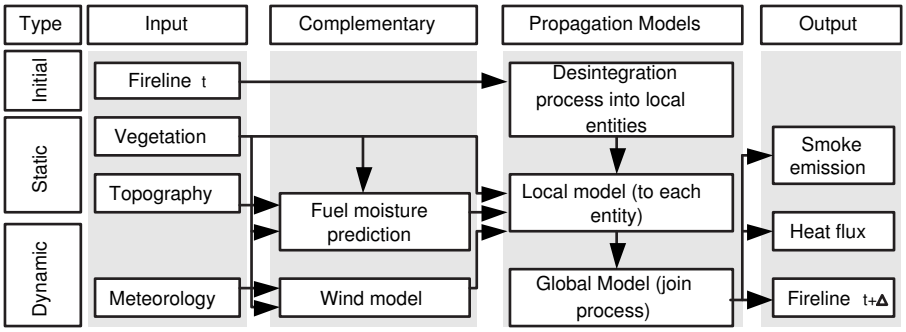

Figure 2.1: Components of an ideal fire simulation system

The main components are the following:

1. Input information databases, concerning the physical environment, including: ignition point or current status of the fire line, vegetation maps that include the characteristics of the vegetation of each region, topographic information of the terrain where the fire is burning, and meteorological information.

2. Propagation models: the global and local models.

3. Complementary models: these models include those parameters with a dynamic behavior.

4. Output: mainly is the prediction of forest fire propagation, in addition to others outputs like heat flux and smoke emission.

However, the current state of forest fire research does not allow us to include all the components in a real system. There is an active research in

all these fields [23, 35, 50, 58, 62, 69, 73], but there are no final results that can be included in the simulation systems. For this reason, the real current simulation systems have a simplified structure. This scheme can be seen in figure 2.2.

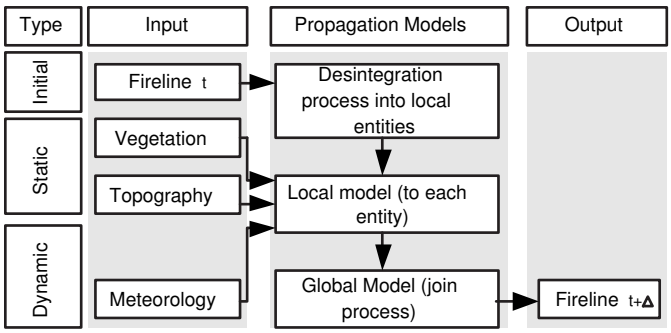

Figure 2.2: Components of a fire simulation system

2.2 Rothermel Model Summary

There are several models in literature to describe the behavior of forest fire propagation, which can be divided into three main categories [54]: physical, empirical and semi-physical models. Physical models rely on theoretical aspects of fire propagation such as the heat transfer process (for example [33, 6, 70]). However, though theoretical aspects can help on understanding the mechanisms of fire spread, physical models are not currently operational as they require information of many parameters almost impossible to measure in the field [12]. Moreover, the heat transfer processes modelled are neither spatially nor temporally constant.

However, there are authors who have developed empirical models to overcome the problems derived from the complexity of the theoretical approaches. These models have been largely developed in Australia ([46, 39, 26]) and Canada ([65, 45]) where extensive databases of wild and prescribed fires are available. Such empirical models are basically statistical models. They have a main drawback: they should be used with great caution under other conditions than those of the original fires.

Finally, the semi-physical option combines empirical and physical tech-

niques. This approach looks for operational models which can be used under a wide range of conditions. Concretely, while the main structure of the equations of the model is based on physical relationship, these equations are adjusted and modified considering the results of experimental studies in the laboratory. Therefore, due to their intermediate features, semi-physical models can be currently applied with predictive purposes in real events.

A good example of a semi-physical model is the Rothermel model [61]. This model is, by far, the most used semi-physical model in fire modellin. In fact, although there are several spatially explicit models being currently applied to predict the behaviour of wild and prescribed fires, such as FARSITE [30], FIRESTATION [31] or BehavePlus [10], the majority of them have in their core the set of equation given by Rothermel.

However, the Rothermel model has at least two big drawbacks:

1. Rothermel equations were derived in an empiric way in a laboratory at a spacial scale of a few square meters, but are now currently used to predict the fire behavior over wide areas [63]. Since exhaustive spatial information of variables involved is not available, a simplified approach based on maps has been usually used.

2. Data for many input variables is needed to apply the Rothermel equations, and some of these variables may be very costly and difficult to obtain. Hence, a reduction in the number of variables is highly advisable in order to achieve an operational use of the model [63].

The Rothermel model has two main outputs: the rate of spread of a point in the fire front (R; in $length * time^{-1}$) and the reaction intensity (I_R) which is the energy released per unit area for a specific period of time ($energy * time^{-1} * length^{-2}$).

Since 1972, several modifications have been proposed for the Rothermel basic model of fire spread. Some examples are the work of Albini [5], affecting the net fuel loading and the optimum reaction velocity, and those of Catchpole and Catchpole [25], based on Wilson [72], which eliminate the use of the moisture content of extinction and give a new formula for pre-ignition heat.

The equation for the rate of spread of a point in the fire front (R) is given by the following equation (see [61] for a detailed description of the original equation).

$$R = \frac{I_R * \xi * (1 + \phi_w + \phi_s)}{(\rho_b * \epsilon * Q_{ig})}$$

where ξ is the propagating flux ratio; ϕ_w the wind coefficient; ϕ_s the slope coefficient; ρ_b the oven-dry bulk density; ϵ the effective heating number; and Q_{ig} the heat of pre-ignition.

The I_R term also can be calculated as follows:

$$I_R = \Gamma' * W_n * h * \eta_M * \eta_s$$

where Γ' is the optimun reaction velocity, W_n is the fuel loading, h is the fuel particle low heat content, η_M the moisture dumping coefficient and η_s the mineral dumping coefficient.

All these terms are, in turn, functions of the 10 input variables listed in table 2.1 (see [63] and [61] for details).

Parameter	Symbol	Units
Fuel bed depth	f	feet
Surface area-to-volume ratio	s	sq ft fuel / cu ft fuel
Low heat of combustion	H	BTU/lb fuel
Ovendry particle density	ρ_p	D.W. g cm^{-3}
Moisture content of the live fuel	lm	lb water / lb fuel
Moisture content of the dead fuel	dm	lb water / lb fuel
Total silica content	St	lb silica / lb fuel
Slope	ϕ	$^\circ$
Wind speed at mid flame height	U	feet / min
Dead fuel loading vs. total loading	P	lb / feet^3

Table 2.1: Input parameters for the Rothermel model (Note: D.W. means dry weight)

2.3 Forest Fire Simulators

In this section, we will describe the relevant characteristics of the most widely used forest fire simulators.

2.3.1 BehavePlus

BEHAVE [9] was developed in the 80's to calculate a few projections of fire behavior simultaneously and to show the results in a table. At that moment, modeling the fire growth was a new interesting theme of research. The fire was simulated through a vector of land, fuel, and climatic conditions that varied spatially and temporarily. Such models made highly repetitive calculations from each starting point towards multiple destiny points, and this process was repeated for each point in the map and each time interval in the simulation. This resulted in millions of iterations. BEHAVE was formed by a set of five programs that were executed under DOS, three of them were available in 1984. This separation of programs was made because of computational restrictions of that moment.

BehavePlus [10] is the successor of BEHAVE system. The BehavePlus system is a program based on PC (Personal Computer) that groups models that describe the fire and its environment of development. It is a flexible system that produces tables and diagrams, and can be used for an ample range of applications of administration before fire situations. BehavePlus works on Windows and offers a friendly interface. A sample of diagram output can be seen in figure 2.3

Some of the capacities that BehavePlus offers are the following:

- Surface fire spread and intensity
- Safety zone size
- Spotting distance
- Crown scorch height
- Tree mortality

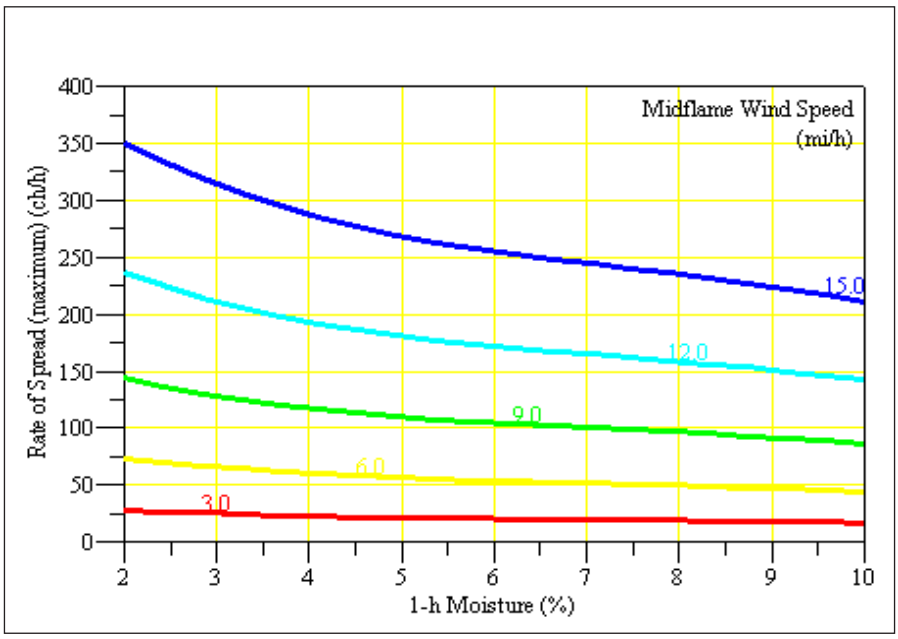

Figure 2.3: Rate of Spread graph with 1 hr dead fuel moisture and midflame wind speed as the independent variables. When up to two input variables have multiple values Behave Plus produces customizable graphs [32]

BehavePlus also offers other capacities: the user can define his own fuel models in addition to the standards. Between the operational capacities, it offers diverse diagrams, photographies of the Rothermel models, and user interface in different languages, among others.

2.3.2 FARSITE

FARSITE [30], also based on the model of Rothermel, automatically computes wildfire growth and behavior for long time periods under heterogeneous conditions of terrain, fuel, and weather. It uses existing fire behavior models for surface and crown fires, post-frontal combustion, and fuel moisture. It is a deterministic model, which means the user can relate simulation results directly to the inputs.

FARSITE produces outputs that are compatible with PC, Workstation graphics and GIS software (Geographic Information System) for later analysis and display. Furthermore, it can simulate air and ground suppression

actions. Finally, it accepts both GRASS[1] and ARC/INFO[2] GIS raster data. A screen shot can be seen on figure 2.4.

Version 4.0 is multi-threaded, which means the program can be divided in pieces that are executed separately. Multi-threading is used to separate the growth of the fire from the interface, to allow so computation takes place in more than one processor.

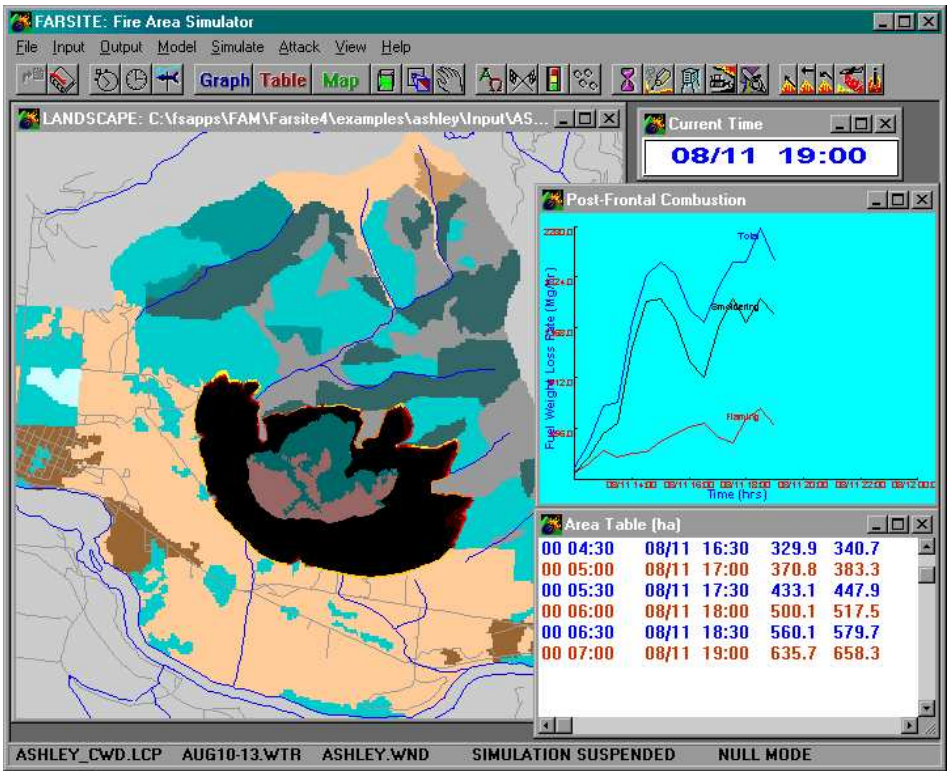

Figure 2.4: Screen shot of a FARSITE v4.00 simulation utilizing the post-frontal combustion model [32]

[1]GRASS (Geographic Resources Analysis Support System) is an open source GIS application. It is a good system to work with raster data and it is very extended in the academic scope.

[2]ARCINFO: ArcInfo Workstation includes tools for edition, publication, transactions management, database manager, etc.

2.3.3 FOFEM

First Order Fire Effects Model, FOFEM [60], is a computer program developed to fulfill the needs of resource managers, planners, and analysts in predicting and planning fire effects.

Quantitative predictions of fire effects are needed for planning prescribed fires that best accomplish resource needs, for impact assessment, and for long-range planning and policy development. The program FOFEM was developed to meet this information.

The FOFEM developers explain that FOFEM will be useful in a variety of situations. Some examples include: setting acceptable upper and lower fuel moistures for conducting prescribed burns; determining the number of acres that may be burned on a given day without exceeding particulate emission limits; assessing effects of wildfire; developing timber salvage guidelines following wildfire; and comparing expected outcomes of alternative actions.

First order fire effects are those that concern direct, indirect or immediate consequences of fire. First order fire effects form an important basis for prediction of secondary effects such as tree regeneration plant succession, and changes in site productivity, but these long-term effects generally involve interaction with many variables (for example, weather, animal use, insects, and disease) and are not directly predicted by this program. Currently, FOFEM provides quantitative fire effects information for tree mortality, fuel consumption, mineral soil exposure, smoke and soil heating.

2.3.4 NEXUS

NEXUS 2.0 [32] is a crown fire hazard analysis software that links separate models of surface and crown fire behavior to compute indices of relative crown fire potential. NEXUS is useful to compare crown fire potential for different stands, and to compare the effects of alternative fuel treatments on crown fire potential. It includes several visual tools useful for understanding how surface and crown fire models interact.

Crown fire hazard assessment and behavior prediction is an emerging science. NEXUS linked existing models of surface and crown fire behavior to produce a system to assess the potential for crown fires at the stand

level. In 1998 the modeling system was initially implemented in a Microsoft Excel spreadsheet. At that time the authors envisioned a short life-span for NEXUS as other more established programs incorporated similar modeling capabilities. In 2001, the authors updated the spreadsheet and produced an online user's guide, but still did not envision a long life for it. In 2003, the authors finally accepted that many of the concepts and modeling simulated in NEXUS are unique enough to warrant its maintenance independent of other programs. Therefore, the authors obtained funding to re-code NEXUS as a stand-alone computer program, shedding the Excel spreadsheet interface.

There are other modeling systems whose functions partially overlap with those of NEXUS. BehavePlus is the industry standard tool for modeling surface fire behavior. The current version simulates only surface fire spread, but later versions may include crown fire simulation capability.

2.3.5 fireLib

fireLib [15] is a C language function library for predicting the spread rate, intensity, flame length, and scorch height of free-burning surface fires. It is derived directly from the BEHAVE fire behavior algorithms for predicting fire spread in two dimensions, but it is optimized for highly iterative applications such as cell-based or wave-based fire growth simulation.

The fireLib library was developed to give fire growth modellers a simple, common, and optimized application programming interface (API) to use in their applications. It is written entirely in ANSI standard C and also compiles under a wide range of Kernighan & Ritchie [43] (pre-ANSI standard) C and C++ compilers on a variety of personal computers and work stations.

The library contains 13 functions, as few as 4 functions are required to create a simple yet efficient and functional fire growth simulator. The fire propagation and intensity computation can be imagined as a pipeline through which the sets of parameters are introduced.

Although the characteristics of the fuel vary through the land, these usually are considered invariant within the frame of time of a fire simulation. An improvement in benefit of the model is to make the computation and to store variables that depend on the intermediate fuel in each model.

2.4 Why fireLib?

In table 2.2 we summarize the main characteristics of the five forest fire simulators previously described. The features shown in this table have been divided in two categories: model and implementation characteristics. The first set of characteristics is related to the type of fire approach by the simulator. The rest of characteristics determine how the simulator code works. When we consider all these topics, it is easy to understand the reason for choosing fireLib as a kernel of our system. Although it does not supply simulation about crown fires or spotting fires, it offers freedom in respect of certain aspects related to implementation.

.	Model Characteristics			Implementation Characteristics			
Simulator	Surface	Crown	Spot	C.A.	Portability	Platform	Ouput
Behave	Yes	Yes	Yes	No	No	Windows	Tab./Graph
FARSITE	Yes	Yes	Yes	No	No	Windows	Graph
FOFEM	No	No	No	No	No	Windows	Graph
NEXUS	Yes*	Yes*	No	No	No	Windows	Tab./Graph
fireLib	Yes	No	No	Yes	Yes	Linux/W.	Text

Table 2.2: Summary of forest fire simulators characteristics. (Note: C.A. means Code Accessibility, Tab. means Tables , W. means Windows and * means that Nexus only produces ratio index, but not behavior)

fireSim is a simulator proposed by Colins D. Bevins, implemented using the fireLib library [15] previously commented. The simulator uses an approach based on cells. The land is divided into cells, and the relationship with the adjacent neighbors is used to evaluate if the cell has been burned and in which moment this happens.

The simulator accepts as inputs the map of the land, the characteristics of the vegetation, the wind and the ignition map. The output consists of a map of the land where each cell is labeled with its time of ignition.

Following, we enumerate the reasons why we decided to use the firesim as simulator in our experiments:

- fireSim is a simple and functional simulator. The power and benefit of this tool is in its simplicity. Its scheme of work, based on cells, offers enough freedom and the possibility of adjusting land units.

- fireSim was written in ANSI C, which is a general-purpose language with a good economy of expressions. Therefore, we obtain a portable simulator.
- fireSim has a solid base because it derives from another simulator (BEHAVE). fireSim has the same functionality of BEHAVE, and it has been optimized for highly iterative applications, such as simulation of fire propagation through cells.
- It is based in the Rothermel Model, what can be seen as an advantage, since it is one of the most used models within fire propagation models. However, it has the model's limitations commented in section 2.2.
- The source code is *open source* which simplifies its adaptation and modification.

2.4.1 FireSim

Once we have seen the functionality offered by fireLib, we will schematically describe the fireSim structure.

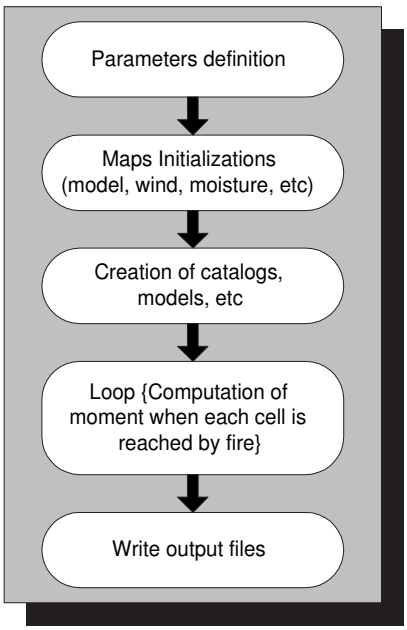

Figure 2.5: Structure of fireSim code

In figure 2.5 we can observe the simulator simplicity. The structure is divided in five blocks: *Parameters definition*, *maps initialization*, *creation of catalogs*, *main loop*, and *writing of output files*. The main loop deserves a special mention, since it operates through an algorithm of "Contagion" [15] applied to the eight neighboring cells of the analyzed cell in a given moment. By each burned cell, the algorithm analyzes the eight neighboring cells and determines the time of ignition for them, as long as conditions (humidity, wind, etc.) allow the fire arrive them.

2.4.2 Forest fire calculation

As figure 2.6 shows, calculation of fire spread under fireLib simulator can be represented as a pipeline compound by four stages. In the first stage, the characteristics fuel area, load, etc, are calculated. The `Fire_FuelCombustion()` function performs this stage.

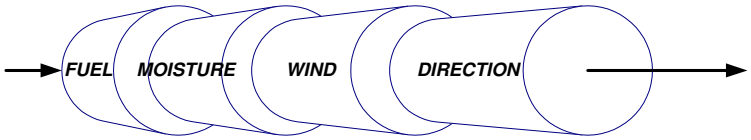

Figure 2.6: Pipeline structure of firelib

Fuel moisture, wind speed, and wind direction are more temporal. Fuel moisture is introduced in the second stage of the pipeline, because the Rothermel fire model uses fuel moisture to determine the heat sink term in deriving fire reaction intensity and the no-wind no-slope spread rate. The `Fire_SpreadNoWindNoSlope()` function performs the second stage computations.

Rothermel modelled the effect of slope on the fire behavior using the same mechanism as for wind; he calls the combined effect of slope and wind the *effective windspeed*. In the third stage the maximum spread rate and direction of maximum spread are calculated by `Fire_SpreadWindSlopeMax()` function.

Finally, in stage four, fire behavior in any compass direction is determined using the elliptical growth model developed by Anderson [7] and employed by BEHAVE [9]. The four vector-dependent fire behavior outputs of spread

rate, Byram's fire line intensity, flame length, and height are derived by the fireLib `Fire_SpreadAtAzimuth()` function.

Significant improvement in the fire model performance is realized by partitioning the fire behavior computation into these four stages [15].

2.4.3 Catalog

The fireLib API employ the concepts of fuel catalog, fuel model, and fuel particle. A fireLib application may contain one or more fuel catalogs. A fuel catalog is a collection of zero or more fire behavior fuel models, and each fuel model contain zero or more fuel particles.

- **Fuel Catalog Objects** is a collection of fuel models. In addition to fuel data, it serves as a container for state information on function status, error messages, and use of the optional flame length table. Each fuel catalog is identified by a handle returned by one of the fuel catalog creation functions. All other functions require this handle as an input argument. Example: The following call creates a Catalog called "**new**" with enough space to store from 0 to *MaxModels* fuel models. The handle is stored in *my_ catalog*, with type FuelCatalogPtr.

  ```
  my_catalog = Fire_FuelCatalogCreate(‘‘new’’, MaxModels);
  ```

- **Fuel Model** describes the fuel particles and fuel bed arrangement for a representative fuel condition. The fuel model object also serves as a container for all the current fire environment, fuel bed, and fire behavior state information relating the fuel model.

  Each model within a catalog is identified by a unique integer number. This number may range from zero to the maximum number of models allowed in the catalog. All fireLib fire behavior functions require the fuel catalog handle and the fuel model number as input arguments (in the example, *MaxModels*). The following example shows the creation of a model called "shortGrass" which has a height equal to 0.5 feet, moisture equal to 0.12 lb *water/lb fuel* and only one particle.

```
Fire_FuelModelCreate(my_catalog, 1,‘‘shortGrass’’,‘‘first
                     model’’, 0.5, 0.12, 1, 1);
```

This results in a model with the following characteristics:

- Model number: 1
- Name: shortGrass
- Description: first model
- Height: 0.5 feet
- Extinction Moisture: 0.12 lb water/lb fuel
- Particles: 1

- **Fuel Particle** is a specific type of fuel component found in a fuel model (needle, grass, or foliage litter, etc). Fuel particles are assigned an index number (starting with 0) as they are added to the fuel model.

  For example, the added particle to the model previously created could have the following values:

  - Particle index = 0
  - Type: 1 (Fire_Type_Herb)
  - Load: 0.034 lb fuel /sq feet
  - Ratio surface/volumen: 3500
  - Density: 35 lbs fuel/cu ft fuel
  - Combustion heat: 8000 BTU/lb fuel
  - Total Silica content: 0.0555 lb silica/lb fuel
  - Effective silica content: 0.01 lb silica/lb fuel

2.4.4 Accessing Object Properties using macros

fireLib functions return a success or failure code rather than computation result. Access to catalog, fuel model, and fuel particle properties, including all input variables, stored intermediates, and all fire behavior outputs, is

available to the programmer via C macros. These macros are used like C function calls to access or update current object properties.

A property macro consists of the "Fuel_" prefix followed by a descriptive label and 1, 2 or 3 arguments. The first argument is always the catalog handle. If two or more arguments, the second is always the fuel *modelNumber*. If three or more arguments, the third argument is either the fuel particle index, life class index, or moisture class index.

Some examples are:

```
/* to obtain the error message */
message = FuelCat_error(catalog);

/* returning the fuel height of model 11 within the catalog */
depth = Fuel_Depth(catalog, 11);

/* it recovers the load of particle 1 within the model 11 in
   the catalog */
load = Fuel_Load(catalog, 11, 1);
```

2.4.5 Functionality

There are eight functions for creating, querying, and destroying fuel catalog, fuel model, fuel particle, and flame length table objects, and four functions for determining fire behavior for fuel models.

It is possible to create a fuel catalog to include the 13 standard fire behavior, or to create an independent catalog and to append new models using calls to create models.

A catalog also can be destroyed.

On the other hand, the models consist of three functions: one for creating, one for destroying and the third to verify if a model exists. The particles only can be created. The form to eliminate them is destroying the model or the catalogue that contains them. In order to change something in their characteristics the user must resort to the use of the macros.

The functions of fire behavior calculation are interesting. We comment some of them:

```
int Fire_FuelCombustion (
    FuelCatalogPtr catalog,
    size_t         modelNumber)
```

Calculates all intermediate fuel bed and combustion variables that are solely dependent upon fuel bed characteristics (stage 1 in the pipeline). It is normally not called directly by the user as it is automatically called by `Fire_SpreadWind- NoSlope()` whenever fuel *modelNumber* has been defined or updated. The function returns `FIRE_STATUS_OK` on success or, on failure, the constant `FIRE_STATUS_ERROR`.

A large number of intermediate fuel and combustion variables are calculated and stored by this function. The original BEHAVE System spends approximately 80% of its computation time in the redundant calculation of these non-variant values. Isolating these computations within a function which is called only once per fuel model dramatically improves the performance.

```
int Fire_SpreadNoWindNoSlope (
    FuelCatalogPtr catalog,
    size-t         modelNumber,
    double         moisture[FIRE_MCLASSES])
```

Calculates reaction intensity, heat per unit area, and no-wind no-slope fire spread rate for fuel *modelNumber*. It automatically calls `Fire_FuelCombustion()` if needed for *modelNumber*. The `moisture[]` array contains the current fuel moistures of dead fuel. The functions returns `FIRE_STATUS_OK` on success or `FIRE_STATUS_ERROR` on failure.

```
int Fire_SpreadWindSlopeMax (
    FuelCatalogPtr catalog,
    size_t         modelNumber,
    double         windFpm,
    double         windDeg,
    double         slope,
    double         aspect)
```

Calculates the maximum spread rate and direction of maximum spread given the wind and terrain conditions. Once `Fire_SpreadNoWindNoSlope()` has been called to establish initial conditions for this fuel *modelNumber*, `Fire_SpreadWindSlopeMax()` may be called repeatedly with different input arguments to optimize simulation performance. The function returns `FIRE_STATUS_OK` on success or `FIRE_STATUS_ERROR` on failure.

```
int Fire_SpreadAtAzimuth (
    FuelCatalogPtr catalog,
    size_t         modelNumber,
    double         azimuth,
    size_t         whichOutputs)
```

Calculates spread rate along the requested compass *azimuth* for fuel *modelNumber*, an optionally calculates Byram's fire line intensity, flame length, and/or scorch height along *azimuth* as requested by *whichOutputs*. Once `Fire_SpreadWindSlopeMax()` has been called to establish initial conditions, `Fire_SpreadAtAzimuth()` may be called repeatedly to get fire behavior at multiple azimuths. The function returns `FIRE_STATUS_OK` on success or `FIRE_STATUS_ERROR` on failure.

```
int Fire_FlameScorch (
    FuelCatalogPtr catalog,
    size_t         modelNumber,
    size_t         whichOutputs)
```

Calculates the flame length or scorch height along the azimuth established by th most recent call to `Fire_SpreadAtAzimuth()`. The function returns `FIRE_STATUS_OK` on success or `FIRE_STATUS_ERROR` on failure.

Chapter 3

Data Driven Prediction Methods

> "PLAN: To bother about the best method of accomplishing an accidental result."
>
> *The Devil's Dictionary*, Ambrose Bierce

Nowadays, fire spread models are complex models designed to simulate forest fires occurring under heterogeneous conditions ([9, 10, 30, 31, 44]). These models are very data-demanding; they need a large quantity of information on topography, on some fuel characteristics (static and dynamic) and on wind speed and direction as it has been mentioned in Chapter 1. However, an important problem is that such data may not be entirely available. It is implicitly assumed that if all this information is correct, then the model will correctly predict the spread of fire. But this is not usually the situation with complex, distributed environmental models of any kind [13]. Therefore, in all circumstances, a calibration of the model using observed fire behavior data is essential [21, 24]. In fact, this is being done manually, in a non-systematic way, by fire analysts using fragmentary data on fire spread and behavior.

As we commented in Chapter 1, instead of using the Classical Prediction Method, it is possible to use a different approach, what we call Data Driven strategies. To summarize, this alternative methodology is based on executing a large number of simulations in order to select the best parameters set for

doing the prediction in a future time. Within these methods, we will analyze two cases:

- GLUE: This method is based on the Generalized Likelihood Uncertainty Estimation [13], which is a framework for estimating predictive uncertainty of complex environmental models. Originally, GLUE was developed for being used with hydrological models, but it has subsequently been applied to a wide range of environmental systems.
  GLUE framework supposes that all model structures, parameters, and observations must be in error. Then, there is no reason to look for a single 'true' set of parameter values. So, the GLUE methodology makes suggestions by considering a high number of possible sets of parameter values.

- Evolutionary Algorithms: They mimic the metaphor of natural biological evolution. Evolutionary algorithms operate a population of potential solutions applying the principle of survival of the fittest to produce even better approximations to a solution.
  Evolutionary algorithms model natural processes, such as selection, recombination, mutation, migration, locality and neighborhood. Evolutionary algorithms work on populations of individuals instead of single solutions, which allows to perform the search in a parallel manner.

In the following section, we will describe these methods in detail.

3.1 GLUE

Empirical calibration of parameters has become a usual practice to minimize some of the limitations when we use models. Most of the current methods consider the model as a representation of reality and, under this assumption, they try to find a parameter set to reproduce the available observational data as closely as possible. An alternative to this type of model is the GLUE framework [13] which supposes that all model structures, parameters, and observations must be in error. Therefore, there is no reason to look for a single set of parameter values. Instead of that, GLUE works with a large set

of parameters combinations assigning different weights (likelihoods) to them by analyzing their ability to make accurate predictions of the observed data. This process can be carried out sequentially (i.e. every time there is new information on the evolution of the fire event the likelihoods are updated). After comparing the different parameter sets with the first data available the prior likelihood is modified in accordance with the model performance and a posterior likelihood is calculated (a Bayesian approach). Those parameter sets that obtain higher likelihood are selected as good predictors and are used for future forecasting [56].

3.1.1 Method description

The GLUE method of Beven and Binley [13] is a Monte Carlo [55, 57] simulation based approach to model conditioning and uncertainty estimation. It rejects the idea that there is only one optimum parameter set in a model calibration. It considers that there are multiple parameter sets and even multiple model structures that may be acceptable in simulating the system under study. Therefore, it is possible to evaluate the relative likelihood of a given model and parameter set in reproducing the available data to test the models. Then, uncertainty in the predictions may be estimated by calculating a likelihood weighted cumulative distribution of a predicted variable based on the simulated values from all the retained simulations (those with a likelihood value greater than zero). Thus, for any model predicted variable, Z:

$$P(\hat{Z}_t < z) = \sum_{i=1}^{N} L\left[M(\Theta_i)|\hat{Z}_{t,i} < z\right]$$

where $P(\hat{Z}_t < z)$ are prediction quantiles, $\hat{Z}_{t,i}$ is the value of variable Z at time t simulated by model $M(\Theta_i)$ with parameter set Θ_i and likelihood $L[M(\Theta_i)]$. Then, the accuracy in estimating such prediction quantiles will depend on having a suitable sample of models to represent the behavioral part of the model. In this framework the parameter values are treated as a set with their associated likelihood value so that any interactions between parameter values in fitting the available observations are included implicitly in the conditioning process.

In this case, a fuzzy measure of goodness of fit has been used. Initial prior likelihoods were set to zero for all the parameter sets. The updating of likelihoods from one time step to the following one consisted in averaging the prior and the current likelihoods. In order to make the uncertainty limits converge when the current rate of spread did not change, this average can be raised, optionally, to a power p ($p \leq 1$):

$$L_p(M(\Theta_i)) = \frac{[L_0(M(\Theta_i)) + L(M(\Theta_i)|Y)]^p}{C}$$

where $L_0(M(\Theta_i))$ is the prior likelihood of the model M with the parameter set Θ_i; $L(M(\Theta_i)|Y)$ is the goodness of fit of the model M with the parameter set Θ_i to the latest observations Y; $L_p(M(\Theta_i))$ is the posterior likelihood of the model M with the parameter set Θ_i; and C is a constant which ensures the sum of the posterior likelihoods of all the parameters equals 1.

3.1.2 Method implementation

Figure 3.1 shows how the GLUE method has been applied for fire prediction purpose. Different from classical prediction, GLUE considers a high number of possible sets of parameters values which are stored in the PS box (Parameters Sets) where the likelihood of parameters sets are updated. To reduce the divergence between classical prediction and real-fire propagation, we need to evaluate the goodness of the results provided by the simulator. For this purpose, a Fitness Function must be included (FF box). This function will somehow determine the degree of matching between the predicted fire line and the real fire-line.

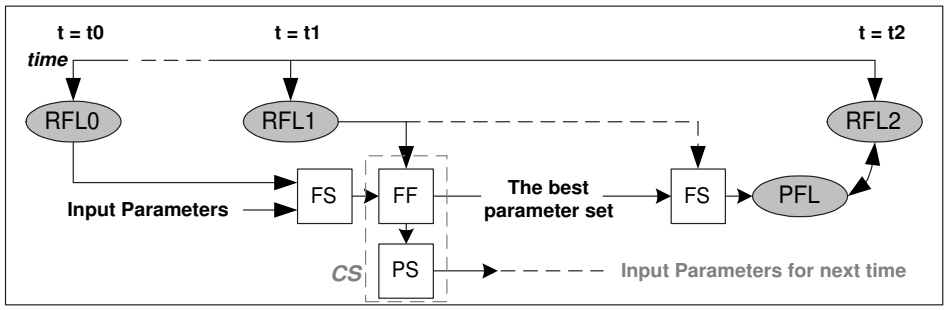

Figure 3.1: GLUE implementation

Then, according to their ability to make a proper prediction (evaluated by FF box), the likelihood of each parameter set is updated. This process is performed N times (being N the number of sets). Finally, once FF has been evaluated for the whole parameter sets, the best set of input parameters is used as the input set for the underlying fire simulator, in order to obtain the predicted position of the fire front (PLF) in the very near future (t_2 in figure 3.1). This process is repeated each time a new fire-line feeds the process in order to readjust the prediction as the fire goes on.

The GLUE software was coded in the C language. The program has a simple structure showed in figure 3.2.

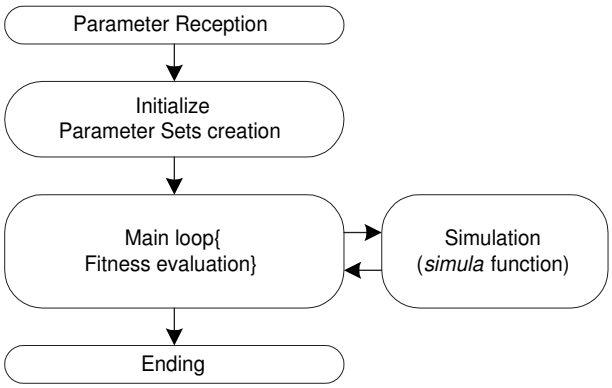

Figure 3.2: GLUE: application structure

Some details about these stages are:

- **Parameter Reception**. The parameter set necessary is the following: *Parameters file*, for each parameter it contains information relative to the fire model, values of climatic factors, etc. The format is: *#Name Inferior_ Edge Superior_ Edge*; *Real Fire File*, it contains a matrix representing the evolution of the real fire. The cells have real values to represent the time in which each cell has been burned. If this has not happened, the value that appears is zero. This file is used to initialize the fire map, determining from which cells the fire will propagate; *Time interval to define the simulation*, the third and fourth value that system receives are the times between which the simulation is considered. These times have to be fixed in minutes.

- **Initialize**. The initialization stage consists of dynamic creation of matrixes which will contain the initial fire map. Note that in Parameter Sets Creation stage, the generation of each set is made in a random way, and these random sets will keep invariant for each simulation step.

- **Main Loop**. The main loop is in charge of calculating the fitness for each set of parameters. The obtained results are sent to a file for a posterior analysis. In order to have a fault tolerant system, every certain number of iterations the current state is recorded in a 'backup'. Consequently, if the system falls, execution would continue from the closer saved point recorded. Once all iterations have been concluded, it is possible to calculate the ignition probability to each cell.

- **Output**. GLUE method continuously generates output files containing information about parameters sets and their fitness and likelihood.

Figure 3.3 shows that it is possible to match a Master-Worker structure with the GLUE scheme. As we can observe, the Master process has a data reception stage which will be followed by an initialization stage for data structures. Once these initial stages have been carried out, the Master process will enter its main loop. In the main loop, the Master process distributes scenarios to the workers, waits for results, receives results and distributes more data to idle workers (if there are more parameters sets to simulate).

The Worker structure is complementary. It is necessary to add a data reception stage (to initialize terrain size, slope). Following this, it enters a loop to receive scenarios from the Master process to activate the simulation function for calculating fire spread.

3.2 Evolutionary Computation

The prediction method proposed by B. Abdalhaq in [3] and commented in this section is a step forward in respect of the classical methodology. This approach also focuses on overcoming the input-parameter uncertainty problem by introducing the idea of applying an optimization scheme to calibrate

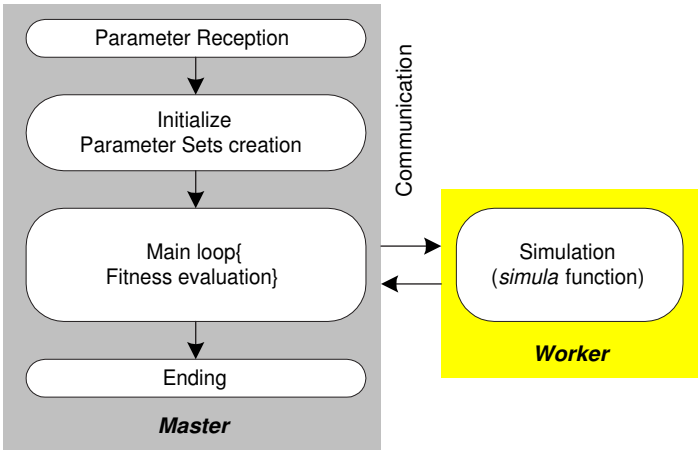

Figure 3.3: How Master-Worker is matched to the GLUE structure

the set of input parameter with the aim of finding an optimal set of inputs, thus improving the results provided by the fire-spread simulator. The aim of the optimization process is to find a set of input parameter to feed the simulator; this set should describe fire behavior in the past. Therefore, the idea is to expect that the same set of parameters could be used to improve the prediction of fire behavior in the near future by assuming that these parameters will remain valid for a period of time (*predictability limit*).

3.2.1 Method description

Formally, optimization is associated with the specification of a mathematical objective function (called L) and a collection of parameters that should be adjusted (tuned) to optimize the objective function. This set of parameters is represented by a vector referred to as θ^*. Consequently, we can formulate an optimization problem as follows:

$$\text{Find } \theta^* \text{ that solves } min_{\theta \in S} L(\theta)$$

where L represents some objective function to be minimized (or maximized). The optimization problem deals with the aim of defining a process to find a setting for the parameter vector θ, which provides the best value (minimum or maximum) for the objective function L. This search is carried out according to certain restrictions of the values that each parameter can

take. The whole range of possibilities that can be explored in obtaining the optimization goal is called the search space, which is referred to as S.

After accommodating the fire prediction elements to this optimization description, the resulting scheme is shown in figure 3.4.

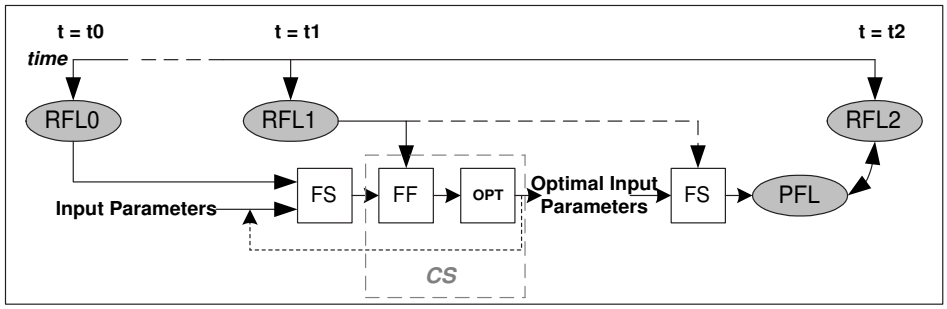

Figure 3.4: Evolutionary wildland fire prediction method

Now we will see the correspondences between the depicted boxes and the optimization components. First of all, the vector θ corresponds to the set of input parameters to be optimized. The length of this vector will depend on the underlying simulator, but in general, as the majority of fire simulators are based on the above-mentioned Rothermel model, it will not have less than 10 components.

Both, the fire simulator (FS box) and the fitness function (FF box), conform the objective function L together. Finally, the OPT box will include the optimization strategy selected to solve the problem. The goal of the optimization strategy consists in generating a new set of input parameters, which minimizes the underlying error prediction, taking into account the information provided by L (FS and FF boxes). The optimization process is performed in an iterative way, which is represented in figure 3.4 by the dotted arrow. This feedback loop will be repeated until either a "good" solution is found or a predetermined number of iterations has been reached. At that point, an "optimal" set of input parameters might be found, which will be used as the input set for the underlying fire simulator, in order to obtain the predicted position of the fire front (PLF) in the very near future (t_2 in figure 3.4). This process is repeated each time a new fire-line feeds the process in order to readjust the prediction.

3.2.2 Method implementation

A great improvement at the time of making implementation is to consider not only one guess at a time, but a wide set of guesses and, based on the results obtained for all of these, to automatically generate a new set of guesses and re-evaluate the objective function for them.

If this extended optimization procedure is applied to the evolutionary prediction scheme, we obtain the extended scheme depicted in figure 3.5.

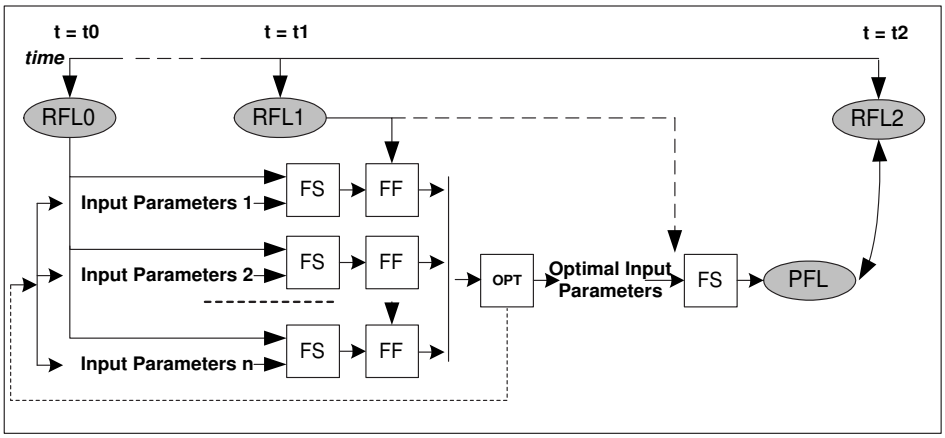

Figure 3.5: Parallel implementation of the evolutionary wildland fire prediction method

Having in mind the optimization goal, an optimization framework called BBOF (Black-Box Optimization Framework) [1] was developed, which consists of a set of C++ abstract-based classes that must be implemented in order to fit both the particular function being optimized and the specific optimization technique. Because these two elements are independent of each other, it is possible to experiment in a "plug&play" fashion with different complex objective functions and optimization techniques. BBOF works in an iterative way, where it moves step-by-step from an initial set of guesses to a final value that is expected to be closer to the optimal vector of parameters than the initial one. This goal is reached because, at each iteration of this process, a preset optimization technique is applied to generate a new set of guesses that should be better than the previous one. Schematically, this process proceeds with the following iterative steps:

1. Create a set of initial vectors randomly or distributed around some expected vector of parameters.

2. Evaluate the objective function for each vector.

3. Apply an optimization technique to move from the current set of vectors to a better set.

4. Repeat 2 and 3 until satisfying the exit conditions (number of iterations or accepted value).

This optimization scheme is adequate to be implemented into the Master-Worker [37, 71] programming paradigm working in an iterative scheme. An iterative Master-Worker application consists of two entities: a Master and multiple Workers. The Master process is responsible for decomposing the problem into small tasks (and distributing these tasks among worker processes), as well as for gathering the partial results in order to produce the total result of computation. The Master also runs the code of the optimization algorithm as it is shown in figure 3.6. The worker processes receive a message from the Master indicating the next task. Each worker processes the task, and sends the result to the Master. The Master process may carry out certain computation while tasks of a given batch are being completed. After that, a new batch of tasks is assigned to the Master, and this process is repeated several times until completion of the problem.

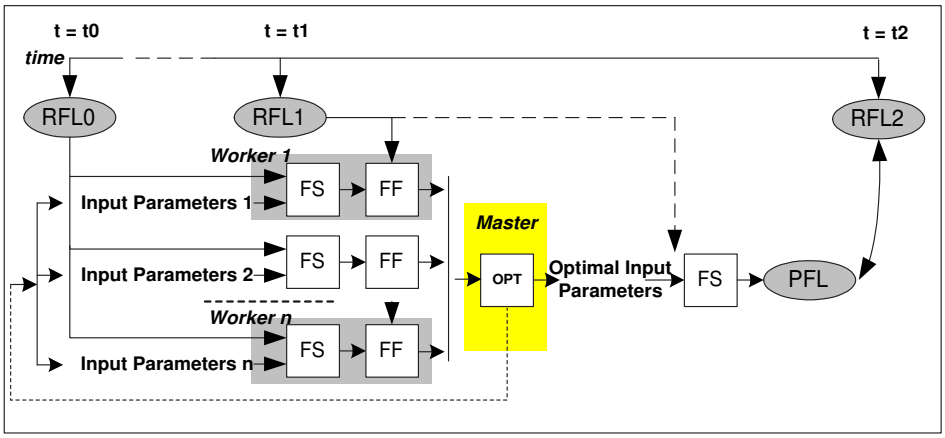

Figure 3.6: How the Master-Worker scheme is applied over BBOF elements

Subsequently, we comment the most relevant classes within BBOF [3].

- **BBOF_objective_function** this class corresponds to the objective function to be optimized. This is the only function that the user has to change if he or she wants to optimize a different objective function. It takes an individual, evaluates the objective function according to the parameters vector and modifies the individual to hold the objective function evaluation result. This class executes on the worker.

- **BBOF_guess** this class is identified with a vector of real numbers and the value of the objective function. This class represents the information related to the tasks to be executed by the workers.

- **BBOF_guess_set** is a set of BBOF_guess classes (vector guesses) and the current iteration number. One important method within this class is the method denoted by BBOF_optimization_technique, which is the optimization technique to be applied. This class keeps the best individual and it generates a report about the set characteristics and status (like average of the fitnesses and standard deviation).

It is possible to match each element of this Master-Worker scheme with the main components of the iterative optimization framework BBOF. Since evaluations of the objective function for each guess are independent from each other, they can be identified as the work done by the workers. The responsibility for gathering all the results from the different workers and for generating the next set of guesses by applying a given optimization technique will be centralized on the Master process.

This framework also was developed to be executed on a cluster system and to use MPI [51] for communication.

Chapter 4

S^2F^2M - Statistical System for Forest Fire Management

> "Todas las teorías son legítimas y ninguna tiene importancia. Lo que importa es lo que se hace con ellas."
>
> Jorge Luis Borges

This chapter focuses on the complete description of the proposed method S^2F^2M. The S^2F^2M characteristics and its implementation details are analyzed.

As we described in chapter 3, it is possible to minimize the input parameter problem by using techniques such as parameter optimization [2]. These techniques have the aim of determining, as precisely as possible, the values of the input parameters that provide a fire spread prediction as close as possible to the real fire propagation. The method described in this chapter also concentrates its effort on processing the input parameters in order to improve the fire propagation prediction. However, instead of searching for an unique 'good' solution, our goal was to develop a methodology based on statistical analysis to determine the most probable behavior of a forest fire. S^2F^2M (Statistical System for Forest Fire Management) [17] does not feed the simulation core with "known" single values, but rather carries out a set of simulations considering a range of possible values for the input parameters that are more uncertain.

Originally, this method was conceived as a statistical system to be useful as a part of a decision support system (DSS) [59]. However, during the initial development process and after the experimentation study published on [16], we glimpsed the possibility of applying the underlying concepts of the system as a prediction method.

4.1 Method description

The methodology of S^2F^2M is based on statistics. There are two possible ways of collecting data about an event. In an **observational study** the researcher only takes notes without interacting in the situation. Data are obtained as they appear. Another way is through **designed experiments**. In this kind of experiments it is possible to make deliberate changes in the controlled variables of a system or process. Results are observed and then it is possible to either make an inference or make a decision about variables that are responsible for changes. When there are a lot of significant factors involved (i.e. weather, wind speed, slope, etc.), the best strategy is to use a **factorial experiment**. A factorial experiment is one in which the factors vary at the same time [47] (for example, wind conditions, moisture content and vegetation parameters). A **scenario** represents each particular situation that results from a set of values.

For a given time interval, we want to know whether a portion of the terrain (called a cell) will be burnt or not. If n is the total number of scenarios and n_A is the number of scenarios in which the cell A was burned, we can calculate the **ignition probability** as:

$$P_{ign}(A) = n_A/n$$

The next step is to generalize this reasoning and apply it to some set of cells. As a consequence, we obtain a matrix with a value associated to each cell that represents the probability of each cell to be catch by the fire (P_{ign}) taking into account n scenarios. The set of cells whose P_{ign} value is bigger or equal to a certain particular value P_K, where $0 \leq P_K \leq 1$, conforms what we call the probability map with probability P_K.

It should be noticed that, although two different cells may have the same $P_{ign}()$ value, this does not mean that the set of parameters that generate this probability in each cell should be the same. Consequently, it is not possible to know the set of scenarios that generate a particular probability map.

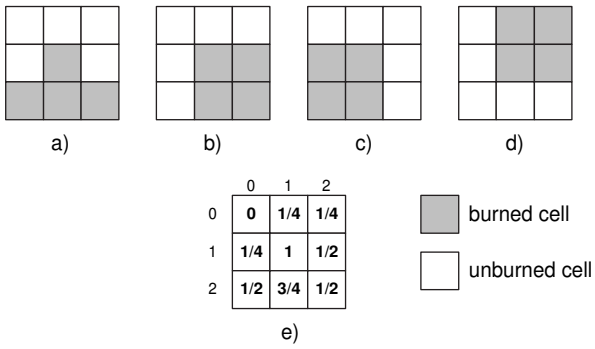

Figure 4.1: Example of cells probability

Figure 4.1 shows a simple case to exemplify this fact. For this simple example the number of scenarios is equal to 4 and the outputs produced by the simulator for each scenario are supposed to be the ones depicted in figures 4.1.a, 4.1.b, 4.1.c and 4.1.d respectively. The final map provided by S^2F^2M once $P_{ign}()$ has been evaluated for each cell would be the matrix shown in figure 4.1.e. As we can observe, there are three cells with $P_{ign}() = 1/2$, however, the $P_{ign}()$ for cell (2,0) has been obtained as an overlapping of results provided by scenarios a and c, whereas the $P_{ign}()$ of cell (2,2) is obtained from scenarios a and b. Therefore, since the number of cells is high in a real case and the number of scenarios is a huge number too, to determine the set of parameter (scenarios) that generate a particular $P_{ign}()$ value in an individual cell becomes an impracticable task.

Once we have obtained the output matrix, which includes all the probability maps, the next step consists on 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 on finding what we refer to as a Key Ignition number (K_{ign}), which accomplishes the following condition:

$$\{x : P_{ign}(x) >= K_{ign}/n \ \mid K_{ign} \in N\}$$

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.

We can clarify this concept graphically. See figure 4.2. In this example, the objective is finding an horizontal cut in the cone (for us, this value will be the K_{ign} value), hoping that the surface of that section will result similar to the real surface (in our case, this second surface corresponds to the real fire area).

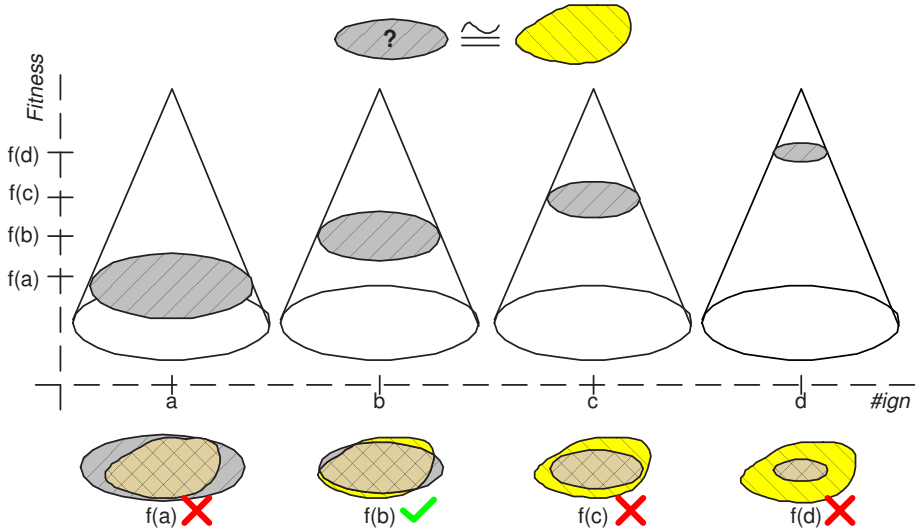

Figure 4.2: Graphic scheme of S^2F^2M method

For height a, the fitness is *f(a)*, for b is *f(b)*, and so on. In the inferior part of the figure we can see the superposition of sections. At least, between these four situations it is possible to identify that in b case the cut surface is very close to the real area. Then, we have found a K_{ign} (b in the example) which defines a similar area to the real situation. So we can use this value to predict in the next step the possible fire behavior; this means the cut with height b in the next cone will be the prediction.

A scheme of a whole prediction system is presented in figure 4.3. As we can observe, the prediction process needs a calibration stage just at the

beginning (time period that goes $t0$ to $t1$ in figure 4.3) to firstly obtain a K_{ign} value to start-up the prediction chain. Once this first K_{ign} has been obtained, both the prediction operation for time ti and the calibration stage for time $ti+1$ will be overlapped on time $ti+1$. This situation is the one depicted in figure 4.3 for time $t2$. As we can observe, the output generated by SS box (Statistic System) is used for a double purpose. On one hand, the probability maps are used as an input of the SK box (Search K_{ign}) to search for the current K_{ign}, which will be used at the next prediction time ($t3$). On the other hand, the SS box output enters a Fire Prediction box (FP), which will be in charge of generating the prediction map for time $t2$ taking into account the K_{ign} evaluated at $t1$. This process will be repeated in time as the system is 'fed' with new information about the fire situation.

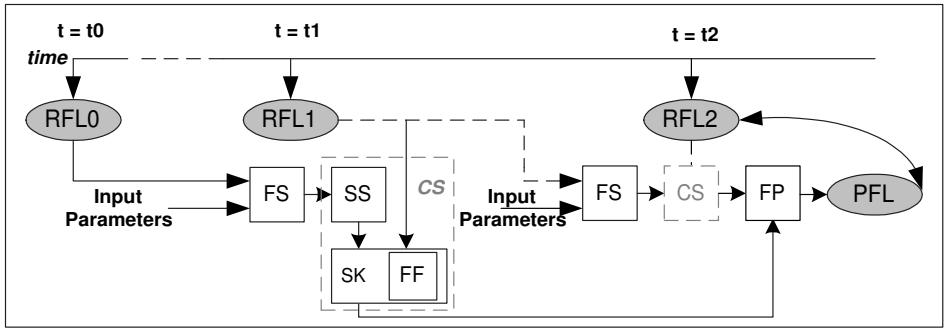

Figure 4.3: S^2F^2M implementation scheme

S^2F^2M uses a forest fire simulator as a black box which needs to be fed with different parameters in order to work. A particular setting of the set of parameters defines an individual scenario. These parameters correspond to the parameters proposed in the Rothermel model [61].

For each parameter we define a range and an increment value, which are used to move throughout the interval. For a given parameter i (which we will refer to as $Parameter\_i$) the associated interval and increment is expressed as:

$$[Inferior\_threshold\_i,\ Superior\_threshold\_i],\ Increment\_i$$

Then, for each parameter i, it is possible to obtain a number C_i, which expresses the parameter domain cardinality, i.e. how many different values

could take the parameter i according to its associated interval and increment. Parameter Domain Cardinality is calculated as follows:

$$C_i = \frac{((Superior\_threshold\_i - Inferior\_threshold\_i) + Increment\_i)}{Increment\_i}$$

Finally, from cardinality of each parameter it is possible to calculate the total number of scenarios obtained from variations of all possible combinations:

$$\#Scenarios = \prod_{i=1}^{p} C_i$$

where p is the number of parameters.

4.2 Method implementation

S^2F^2M system has been completely written in C language for some simple reasons: the code of the simulator fireSim is written in ANSI C and is necessary to include it within the S^2F^2M; the language is quite powerful and has good economy of expression.

The basic program structure can be seen in figure 4.4.

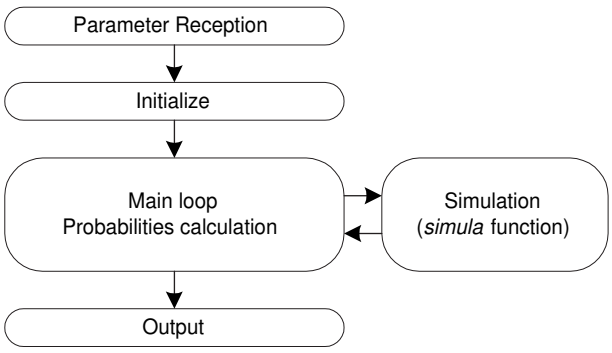

Figure 4.4: S^2F^2M: application structure

Following, we explain some details about these stages (see [16] for a comprehensive review of this theme).

- **Parameter Reception** The first part of the system is the data reception. The necessary parameters set is the following: *Parameters file*,

for each parameter it contains information relative to the fire model, values of climatic factors, etc. The format is: *#Name Inferior_ Edge Superior_ Edge Increase*; *Real Fire File*, it contains a matrix representing the evolution of the real fire. The cells have real values to represent the time in which each cell has been burned. If this has not happened, the value that appears is zero. This file is used to initialize the fire map, determining from which cells the fire will propagate; *Time interval to define the simulation*, the third and fourth value the system receives are the times between which the simulation is considered. These times have to be fixed in minutes; *Output File*, it is only necessary a name for this file. It will have an analogous format to the real fire, but instead of time, it will contain the number of times that each cell has been burned.

- **Initialization** The initialization stage consists on dynamic creation of the matrixes that will contain the initial fire map and the map that will be accumulating the number of scenarios in which each cell has been burned. The cardinal number is calculated as well for each parameter in this section of the program. This is necessary to know which value must have each parameter in each of the different scenarios.

- **Main Loop** Its value is calculated for each one of the parameters for the present number of scenario. Once all the values have been updated, it is possible to call to the `simula()` function. The function returns a matrix which will be added to a *Sumas* matrix. Over this matrix is computed the P_{ign} which will be used in the next step to compute the prediction (represented as fitness). To provide S^2F^2M with fault tolerance, each certain number of iterations is recorded in a file with the present state of the fire front (backup). If the system failed, the execution would continue from this point. Once all the iterations have been concluded, it is possible to calculate the ignition probability for each cell.

- **Simulation** Function `simula()` receives a set of parameters. The first consist of a structure containing the parameters values (wind, slope,

moisture, etc.), the second parameter corresponds to the ignition map and the following parameters are the map dimension and the cells size. With these data, the function effectuates the simulation and returns the resultant map.

- **Output** In this section of the program, to have a better representation of the output, the Gnuplot library has been used [36] to generate a graph from the collected data. An example is showed on figure 4.5

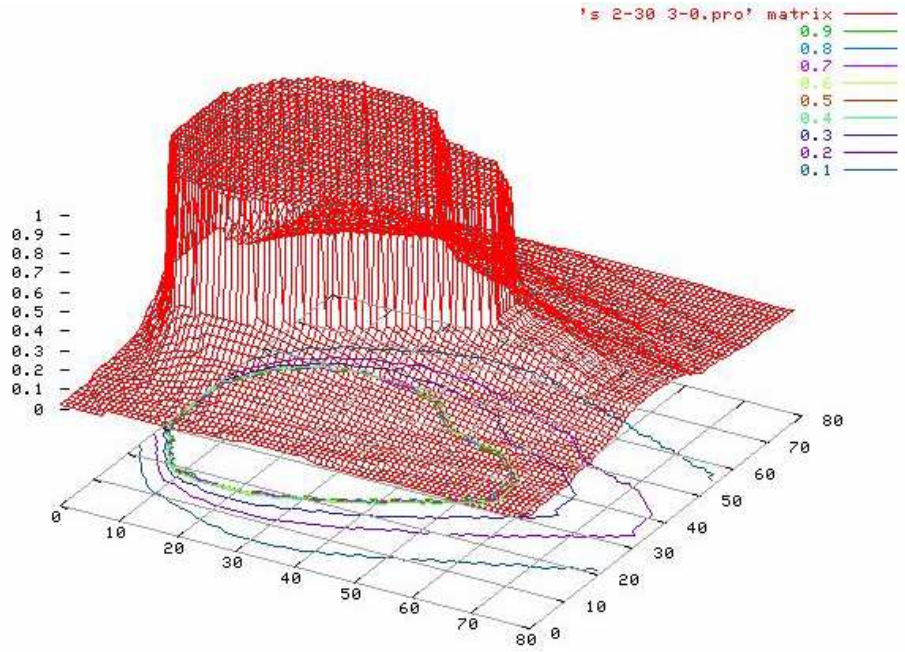

Figure 4.5: The system shows a view in three dimensions, in which X-Y axes indicate the cells division, and Z axis shows the ignition probability for each cell

To reduce the execution time we used multiple computational resources working in parallel to obtain better efficiency. Keeping in mind the nature of the problem that S^2F^2M tries to solve, we believe a Master-Worker architecture is suitable to achieve this aim, because a main processor can calculate each combination of parameters and send them to a set of Workers. These Workers carry out the simulation and return the map to the Master. This resulting map indicates which cells are burned and which are not.

4.3 Parallelizing S^2F^2M

The Master-Worker paradigm is based on real world organizations. It consists of two entities: a Master and multiple Workers. The Master is responsible for decomposing the problem into small tasks and distributing these tasks among a farm of Workers processes, as well as for gathering the partial results in order to produce the final result of computation. The Workers processes receive a message from the Master with the next task, and send the result to the Master. The Master process may carry out some computations while tasks of a given work are being completed.

In the figure 4.6 we see that it is possible to match the Master-Worker structure with the S^2F^2M scheme. The Master process has a data reception stage. After this there is an initialization stage for data structures. In the main loop, the Master process distributes scenarios to the Workers, waits for results, receives results and distributes more data to idle Workers (if there are more scenarios to simulate). Finally, it gives a graphical output.

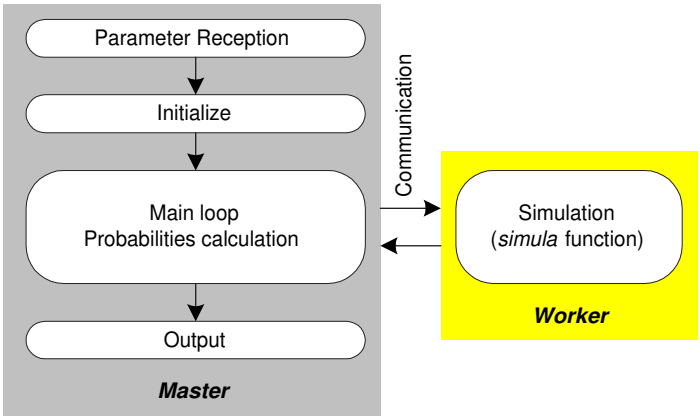

Figure 4.6: S^2F^2M structure related to a Master-Worker scheme

The Worker structure is complementary. It is necessary to add to it a data reception stage (to initialize terrain size, slope). After this, it enters a loop to receive scenarios from the Master process to activate the simulation function for calculating fire spread.

The method works using all possible solutions to find the K_{ign} number. The SS box needs the resulting map of each simulation. While each simula-

tion is independent from the others, we can evaluate them in parallel (figure 4.7). We can assign for each available processor a scenario to be simulated. Following, the Master process needs to collect these results. Once the SS box has added all results, the system does two things: FP box makes the prediction using the map resulting from SS box, and SK box calculates the K_{ign} for the next step. See [16] for more information.

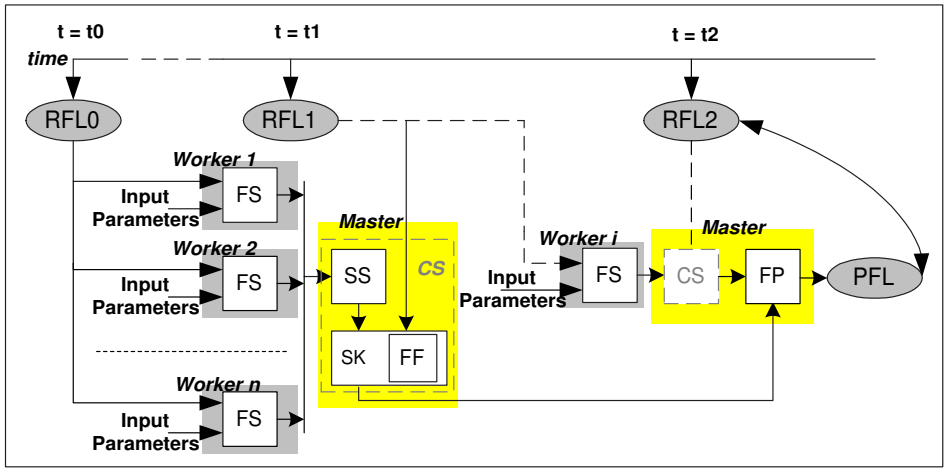

Figure 4.7: How Master-Worker is matched to the S^2F^2M structure

4.4 Tests on synthetic forest fires

Before beginning with tests on real cases, we thought that an important assay was to evaluate the S^2F^2M's capacity to find a Key Ignition Number (and, hence, a good prediction level) in a synthetic forest fire.

First of all, it is necessary to explain the concept of synthetic forest fire. A synthetic forest fire consists of an artificial forest fire obtained using a forest fire simulator and known input parameters. Thus, all in it is synthetic: the terrain dimension, the slope, the type of fuel, meteorological conditions and the beginning fire line. We feed the simulator with a particular set of values which have been set up in a manual way, and then we obtain as output a map that represents the spread of this synthetic fire after a certain period of time. Now, we use this map as input for S^2F^2M, and we define a range file that includes the original values for each parameter. This is a good manner to

test the S^2F^2M's behavior (although it can be used on different methods).

Following, we will see an example. Here, we show only one case with three different interpretations according to time and steps. The shape reached by this synthetic forest fire is showed on figure 4.8, and table 4.1 presents the initial values for each parameter.

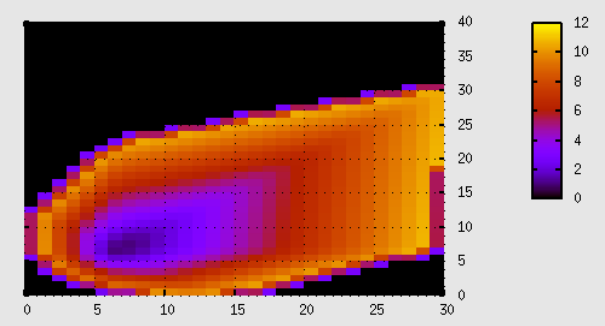

Figure 4.8: Synthetic plot behavior viewed in minutes

We made this case with similar characteristics to the real cases. The plot dimension are: 31 columns by 41 rows, with a cell of 1 m^2.

Parameter	Value
Model	2
Wind Speed	2.01
Wind Direction	60
Slope	7
Aspect	180
Moisture 1h	0.1
Moisture 10h	0.3
Moisture 100h	0.5
Moisture herb	0.1

Table 4.1: Parameter values for a synthetic forest fire

The parameters file is shown in table 4.2. It give us a number of scenarios equal to 56940.

We have divided the experimental study in three cases which have been done in the same synthetic forest fire. The difference between the experi-

Parameter	Initial	Final	Step
Model	1	13	1
WindSpd	0.67	10.28	0.67
WindDir	0.0	360.0	5.0
Slope	7.0	7.0	1.0
Aspect	180.0	180.0	1.0
M1	0.1	0.3	0.2
M10	0.1	0.5	0.2
M100	0.1	0.5	0.2
Mherb	0.1	0.1	1.0

Table 4.2: Parameters file for synthetic plot

ments consists of: *initial time*, *final time* and *step duration*. In tables 4.3, 4.4, 4.5, 4.6, 4.7 and 4.8, we show in pairs the fitness values reached on three experiments, but we do not only show the fitness as a prediction result; we also present the fitness found during the stage of search for the Key Ignition Number. Furthermore, we present the same results for the best individual scenario (i.e., among all scenarios analyzed, we look for the best scenario to check if there really is a set of parameters that fits with the forest fire in each interval of time).

Interval Time	Ignition Number	Found Fitness	Predicted Fitness
1-3	17545	0.5600	...
3-5	17144	0.6779	0.6694
5-7	20112	0.7078	0.6888
7-9	20395	0.7105	0.7066

Table 4.3: Fitness of first experiment on synthetic forest fire for S^2F^2M method

In order to understand better the information listed on the following tables, it is necessary to explain some things about the numbers they include. Let us analyze an individual row of any table. The first column represents the time interval where the simulation has been done (for example 1-3 means an analysis between minute 1 and 3 on table 4.3). The second column has a different meaning depending on which table we analyze. In tables 4.3, 4.5 and 4.7 that column is the Key Ignition Number (K_{ign}). On the one hand, as

we saw in section 4.1, the Key Ignition Number represents an addition which optimizes the fitness value for the current step. The other tables simply lists the Scenario Number that provides the best fitness.

Interval Time	Scenario Number	Found Fitness	Predicted Fitness
1-3	16602	1.0000	...
3-5	45266	0.9540	0.8965
5-7	2367	0.9621	0.9621
7-9	45266	0.9388	0.9388

Table 4.4: Fitness of first experiment on synthetic forest fire for the best scenario of S^2F^2M method

Interval Time	Ignition Number	Found Fitness	Predicted Fitness
2-4	16386	0.6641	...
4-6	17917	0.6724	0.6690
6-8	18206	0.7153	0.7050
0-10	20414	0.7538	0.7264

Table 4.5: Fitness of second experiment on synthetic forest fire for S^2F^2M method

Interval Time	Scenario Number	Found Fitness	Predicted Fitness
2-4	45266	0.9836	...
4-6	2366	0.9396	0.9017
6-8	45266	0.9610	0.9325
8-10	45266	0.9554	0.9554

Table 4.6: Fitness of second experiment on synthetic forest fire for the best scenario of S^2F^2M method

The third column represent the fitness found when the Key Ignition Number is applied to the current time (0.56 for the same example), or the fitness found for the scenario identified in the second column, if we consider the table of the best case (the fitness value for scenario number 16602 is equal to

1.0, that is to say the perfect fitness, on table 4.4). Finally, the last column exhibits the value of fitness corresponding to the application of the key Ignition found in the previous step, or the fitness value corresponding to the best scenario found in the previous step.

As we can see, on the first row this value does not appear because we need the first step to look for the Key Ignition Number. Therefore, in this step is not possible to do a prediction about the fire behavior. We can make these two activities in the following steps.

Interval Time	Ignition Number	Found Fitness	Predicted Fitness
4-5	15876	0.7507	...
5-6	20106	0.7029	0.7000
6-7	20289	0.7582	0.7341
7-8	16361	0.7055	0.7004

Table 4.7: Fitness of third experiment on synthetic forest fire for S^2F^2M method

Interval Time	Scenario Number	Found Fitness	Predicted Fitness
4-5	1952	0.7812	...
5-6	45656	0.8275	0.7088
6-7	1994	0.8243	0.7808
7-8	2746	0.8372	0.7976

Table 4.8: Fitness of third experiment on synthetic forest fire for the best scenario of S^2F^2M method

In general, we can observe that the best scenarios achieve very good fitness values, and hence, the S^2F^2M response is also good.

On one hand, seeing in detail the fitness values for S^2F^2M, we discover that in average we are obtaining fitness equal to 0.7027. On the other, the individual predictions done by single scenarios, are very high. In average they give us a fitness equal to 0.8749.

Chapter 5

Comparative Study on Real cases

> "[...] a me pare che quelle scienze siano vane e piene di errori, le quali non sono nate dall'esperienza, madre di ogni certezza, e che non terminano in nota esperienza, cioè che la loro origine o mezzo o fine non passa per nessuno de' cinque sensi."
>
> Leonardo Da Vinci

Once the descriptions and analysis of each method have been concluded, we effectuate diverse tests to evaluate and to compare these methods. Such tests were performed with real prescribed fires. The information about the burns was obtained in Portugal[1].

Along this chapter we will concentrate on describing the experiments related to fire on fields. Over them we will analyze the obtained results.

5.1 Platform description

The results presented in this chapter, were obtained by executing the S^2F^2M system on a cluster computer. The used cluster is composed by 32 PCs with the same basic hardware configuration, what allow us to consider the group as homogeneous.

[1] ADAI Laboratory (Association for the Development of Industrial Aerodynamics) - Coimbra, Portugal

The configuration was the following:

- *Processor PENTIUM IV 3.0 Ghz*
- *1 GB DDR-SDRAM 400 Mhz*
- *Ethernet card Broadcom NetXtreme Gigabit*

Furthermore, the operating system installed was Fedora Core 4. All the machines were configured to use NFS (Network File System) based on one server which has the same characteristics of the machines. The middleware used to make parallel programs was MPI [40, 41]. MPI is a library specification for message-passing. It is basically composed of a programming interface definition plus a binding collection for more extended languages in the parallel computer user community (C and FORTRAN).

5.2 Terrain description

The study area is located in Central Portugal (Gestosa, 40°15'N, 8°10'O), in a hillside of Serra de Lousã whose altitude is between 800 and 950 m above sea level. The burns were part of the SPREAD project [64] (Gestosa field experiments 2002 and 2004 [34]). The experiments of Gestosa field began in 1998 and have finished in December 2004, and aimed to collect experimental data to support the development of new concepts and models, and to validate existing methods or models in various fields of fire management. These experiments involved several research teams from the European Union and covered a very extensive characterization of variables related to the fire behavior.

To safeguard the safety of the burns and to carry out different sorts of tests and measurements, the terrain was divided into dedicated plots with regular shape and dimensions separated by firewalls to limit fire spread and to keep it inside the desired boundaries during each burn.

These experimental burning plots were established in Forest Service lands, in Gestosa forestry perimeter. Figure 5.1 presents a picture of the area.

The characteristics of the five plots used in our study (dimensions, slope and height) are showed in table 5.1.

Figure 5.1: Gestosa area photography

Plot	Width m	Length m	Slope
520	89	91	18°
533	95	123	21°
534	75	126	19°
751	20	30	6°
752	20	30	6°

Table 5.1: Plot characteristics for each experiment

In general, these experimental plots are located together, in the same vegetation mosaic, which consists in shrub with some isolated *Pinus pinaster* trees. Three arboreal species are dominant in the area: *Erica umbellata*, *Erica australis* and *Chamaespartium tridentatum*.

5.3 Experiment description

During the experiments, different plots have been burned, each one to study different aspects related to forest fires (effect of retardants, smoke production, analysis of behavior patterns, etc.). As we mentioned, we worked with five plots: 520, 533, 534, 751 and 752, which were represented as follows:

- *Plot 520*: by means of a grid of 89 columns x 109 rows (each cell was

$1m$ x $1m$).

- *Plot 533*: by means of a grid of 95 columns x 123 rows (each cell was $1m$ x $1m$).
- *Plot 534*: by means of a grid of 75 columns x 126 rows (each cell was $1m$ x $1m$).
- *Plot 751*: by means of a grid of 60 columns x 90 rows (each cell was $0.333m$ x $0.3333m$).
- *Plot 752*: by means of a grid of 60 columns x 90 rows (each cell was $0.333m$ x $0.3333m$).

In order to gather as much information as possible about the fire-spread behavior, a camera recorded the complete evolution of the fire (figure 5.2). The video obtained was analyzed and several images were extracted every certain quantity of minutes according to the experiment.

Figure 5.2: Ground and aerial cameras recorded the experiments

From those images, the corresponding fire contours were obtained and converted to cell format in order to be correctly interpreted by the program involved in the different prediction methods. In figure 5.3, we can observe one frame of the recorded field fire (figure 5.3.a). Figure 5.3.b shows the obtained fire front after processing the recorded image.

From picture on figure 5.3(b), which first must be corrected according to the perspective, we obtain a set of pairs representing coordinates. Af-

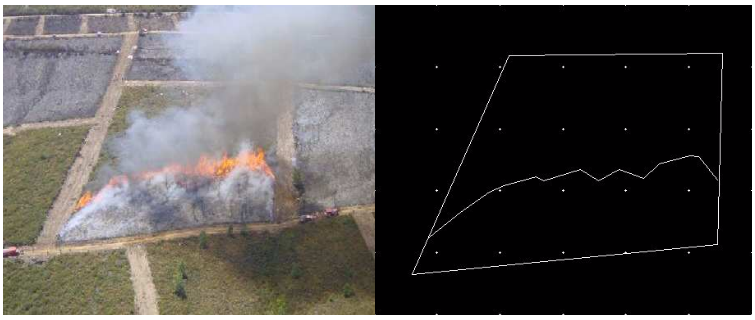

Figure 5.3: (a) Real fire during the burns (plot 520) (b) Fire line obtained from the image on minute 8

terwards, these pairs were processed for creating a matrix representing the burned cells. On figure 5.4 we can see a simple example to illustrate this transformation.

```
" 1", .403106497148371, .523973611081191
" 2", .30086444867948, 13.9068026572691
" 3", 9.0719095850867, 26.680575377737
" 4", 13.4357997465183, 32.3293352848914
" 5", 25.426459206945, 36.8424206669448
" 6", 28.537895613538, 34.7417603150042
" 7", 39.3868624187337, 39.0930578754336
" 8", 46.2945741017092, 33.1357010887857
" 9", 52.9533242027606, 38.2149280957073
" 10", 61.619179282736, 33.3658656028762
" 11", 69.7194488684484, 34.4576323020525
" 12", 78.5090958879127, 43.2714157108274
" 13", 81.285584035602, 41.8754432213699
" 14", 87.8836578502442, 30.201163198811
" 15", 88.2257621152451, 1.23252408601981
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 8 8 8 8 0 0 8 8 0 0 0 8 0 8 8 8 0
0 0 0 8 8 8 8 8 8 8 8 8 8 8 8 8 0 8 8 8 8 8 8 8
0 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
```

Figure 5.4: Conversion of pairs of fire line on a matrix which indicate the time of burn on each cell

5.4 Fitness Function

As we previously mentioned, it is necessary to define a criterion to compare the results of each method with reality. To evaluate the system's response we defined a fitness function. Since the simulator uses an approximation based on cells, the fitness function is defined as a quotient. The following equation shows the expresion:

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

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.

Figure 5.5 shows an example of how to calculate this function for a terrain made of 5x5 cells. In this case, the fitness function is $(7-2)/(10-2) = 0.625$.

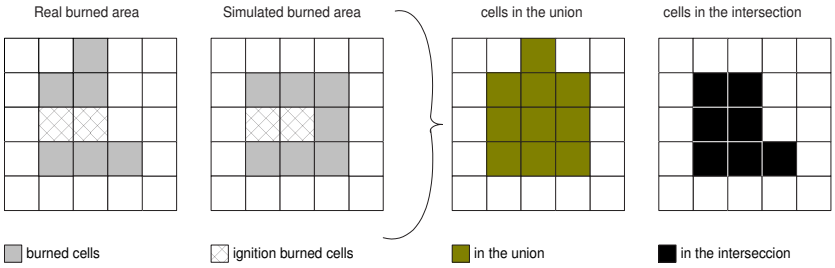

Figure 5.5: Calculating the fitness for a 5 x 5 cell terrain

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, because in this case our experiment did not coincide with reality at all.

5.4.1 Fitness function on GLUE

An important point to emphasize before analyzing the results, is a clarification about GLUE predictions. When we did the experiments using GLUE, we discovered a problem related to the method. According to the methodology explained in section 3.1, the GLUE idea is that a set of scenarios improve their likelihood while the steps are being simulated. In this manner, those good sets of parameter can distinguish themselves from others not so good. Therefore, after certain steps, we could have a population of good cases. However, after experimentation, we discovered that no case stands out about the others. The possible reason is that no scenario can maintain itself as better over all steps. This is straightforward to prove. If we see

the values obtained by each parameter in every scenario, we discover that sometimes they reach a very high fitness, and, we think, they will reach an interesting likelihood. However, normally those good scenarios do not obtain an acceptable fitness in the next step. Then, their likelihoods can not improve, and, finally, none can be chosen as a good scenario due to the fact that all have similar values.

Finally we do not use the concept of likelihood during experimentation, instead we only use fitness, and we choose the best set of parameters (scenario) to use in the next step.

The following values are an example of the previous explanation. They have been obtained from a simulation of each plot. In the tables it is possible to compare how working with fitness values instead of using the likelihood concept improves predictions.

	Observing Likelihood		Observing Fitness	
Time	Likelihood	Fitness	Likelihood	Fitness
6.0	0.000010	0.0098	0.000052	0.4188
8.0	0.000047	0.3236	0.000059	0.6907
10.0	0.000064	0.2559	0.000068	0.3014
12.0	0.000052	0.2300	0.000061	0.7675
14.0	0.000078	0.1889	0.000079	0.2093

Table 5.2: Plot 520. Comparison between values obtained by GLUE when it worked with likelihood and when it worked using fitness

	Observing Likelihood		Observing Fitness	
Time	Likelihood	Fitness	Likelihood	Fitness
6.0	0.000008	0.0000	0.000119	0.3554
8.0	0.000096	0.3071	0.000114	0.5788
10.0	0.000104	0.3672	0.000114	0.5748
12.0	0.000103	0.1899	0.000097	0.3646

Table 5.3: Plot 533. Comparison between values obtained by GLUE when it worked with likelihood and when it worked using fitness

Observing the tables (5.2, 5.3, 5.3, 5.4, 5.6), we can easily verify that it is not advisable to use the likelihood. As we have already explained, the likelihood value does not improve, and since plots have changing climatic conditions from one step to the following one, the set of parameters that

	Observing Likelihood		Observing Fitness	
Time	Likelihood	Fitness	Likelihood	Fitness
5.0	0.000010	0.0000	0.000124	0.2056
6.0	0.000098	0.2083	0.000158	0.5820
7.0	0.000175	0.6466	0.000175	0.6466
8.0	0.000171	0.4221	0.000167	0.5025
9.0	0.000136	0.1338	0.000149	0.3122

Table 5.4: Plot 534. Comparison between values obtained by GLUE when it worked with likelihood and when it worked using fitness

	Observing Likelihood		Observing Fitness	
Time	Likelihood	Fitness	Likelihood	Fitness
6.0	0.000007	0.0000	0.000041	0.6895
8.0	0.000045	0.5990	0.000050	0.7083
10.0	0.000051	0.5235	0.000057	0.6068

Table 5.5: Plot 751. Comparison between values obtained by GLUE when it worked with likelihood and when it worked using fitness

better adapts to the situation usually is very different in each step.

Furthermore, it is interesting to see that in many cases, the likelihood obtained for prediction is even higher when we use Fitness as criterion.

	Observing Likelihood		Observing Fitness	
Time	Likelihood	Fitness	Likelihood	Fitness
6.0	0.000008	0.0000	0.000041	0.5092
8.0	0.000037	0.3703	0.000041	0.4984
10.0	0.000043	0.6754	0.000047	0.4948

Table 5.6: Plot 752. Comparison between values obtained by GLUE when it worked with likelihood and when it worked using fitness

Finally, after doing an intensive experimentation over GLUE, repeating at least 15 times each experiment, we did not find any distribution around the sets of parameters. Therefore, we conclude that an operational GLUE should not work using the Likelihood pattern.

5.5 Experiments results

In this section, we will proceed to describe in detail each experiment and the result of the comparison. We present some important information con-

cerning the development of the experiments, namely the beginning and the end time of each plot burning, information with regard to some particular techniques to ignite the fire, and, obviously, the found fitness after applying each method.

During the total burning time for each experiment, we picked up several instants (t_0, t_1,... etc.). Then, we extracted a fire line from the recorded film at each of the selected instants (RFL0, RFL1,... etc.). We used the first fire line (RFL0) as the initial fire line. The duration of the simulation time was equal to the time period we chose between the initial fire line time and the next extracted fire line time ($\Delta t_1 = t_1 - t_0$). The fire line that the simulator creates when executed with these settings and a set of parameters is the fire line predicted using the classical method. To evaluate the quality of prediction, it is necessary to use the fire line extracted from the recorded film at the next instance and to apply the Fitness function. In this work, we do not evaluate the classical method because in previous works we had proved that data driven methods (GLUE, S^2F^2M and Evolutionary) are better than classical [1, 2, 4, 17, 19].

Basically, the steps to apply the prediction methods are the following: We used the first fire line as initial fire line (RFL0). We used the fire line for the next time (RFL1) as the reference fire line to optimize the parameters. The result of the optimization process is a set of parameters that minimizes the prediction error (in GLUE or Evolutionary) or a Key Ignition value (K_{ign}) for S^2F^2M. We substitute the expected parameters used in the classical way with the optimized parameters resulting from the optimization process, then as in the classical method, we run the simulator using the optimized parameters to predict the fire line at the next instance t_2 (in GLUE and Evolutionary). At that moment, we have a fire line that predicts the fire line at time t_2. Then, as in the classical prediction method, we use the Fitness function to estimate the prediction quality by comparing the estimated fire line with the real fire line.

In the case of S^2F^2M, we use the key Ignition value after calculating all scenarios in the next instance t_2 to show the behavior pattern. Also, the prediction quality is calculated using the Fitness function.

When the burns time has reached the time t_2, we repeat the same process,

but advancing the time by one. The initial fire line becomes the fire line at the time instance t_1, and the real fire line to be used as reference will be at the beginning of the time t_2 and the time of the predicted fire line will be t_3, and so on till the end of available time instants.

It is important to remark that these methods need more fire lines than the classical prediction time so we cannot apply it at the first time t_1. In the experiment, we want to compare three methods so, we cannot show some result at the first time (t_1) because we have no results to compare.

An important point to highlight before describing the results, is to mention that the showed values for GLUE and Evolutionary method are the average of ten executions. In the case of S^2F^2M this is not necessary because it gives a deterministic result.

5.5.1 Experiment 1: Plot 520

According to the known information about the experiment and the models of Rothermel, for some of the parameters certain ranges have been specified. A part of this information has been measured during the experiment, and the remainder has been taken from standard values used by BehavePlus. Therefore, ranges file was defined as we show on table 5.7:

Parameter	Initial	Final	Step
Model	1	13	1
WindSpd	0.67	10.289	0.67
WindDir	0.0	360.0	5.0
Slope	18.0	18.0	1.0
Aspect	180.0	180.0	1.0
M1	0.1	0.3	0.2
M10	0.1	0.5	0.2
M100	0.1	0.5	0.2
Mherb	0.1	0.1	1

Table 5.7: Parameters file for Plot 520

In order to be able to compare all predictions methods, we need to fix an initial time (t0) and a certain time step. These values have both been fixed to 2 minutes. Observing the real case (figure 5.6), we can see the burned area between these times. Notice that in this figure X-Y axes indicate the cells division (89 x 91). And, according to the colour of each cell, it is possible

to know the time of burn (black colour indicates not burned terrain). As we can observe, this is a case of line ignition at bottom using pyrotechnic devices.

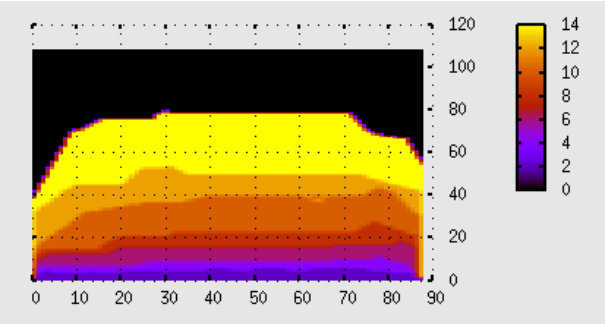

Figure 5.6: Real spread for plot 520, considering steps of 2 minutes

After the application of each method, we obtained the fitness values shown on table 5.8. The second row of the table means that if we are at time 4 minutes and we make a prediction for 6 minutes, we will get a fitness of 0.5345 for S^2F^2M, 0.4188 for GLUE and 0.4513 for Evolutionary method.

Initial Time	Final Time	Fitness		
		S^2F^2M	GLUE	Evolutionary
2.0	4.0	-	-	-
4.0	6.0	0.5345	0.4188	0.4513
6.0	8.0	0.7495	0.6907	0.6985
8.0	10.0	0.4413	0.3014	0.4173
10.0	12.0	0.7625	0.7623	0.6231
12.0	14.0	0.4354	0.2093	0.3956

Table 5.8: Comparative of found fitness in each method for plot 520

We can observe that the prediction proposed by S^2F^2M always overcomes the GLUE and Evolutionary methods. The highest fitness value (0.7625) is reached at time 12. In fact, it is possible to observe that in final time 10 and 14, the fitness value becomes significantly lower. Since this behavior has been observed in the majority of the analyzed cases, we provide a more detailed discussion about this fact later on this chapter (section 5.6). However, that value is still above GLUE fitness and above Evolutionary case.

5.5.2 Experiment 2: Plot 533

The second experiment has a similar duration to the previous burning, however, it is very different. In the present case the file of ranges exhibits some difference with the previous one because the experiment presents other characteristics: the wind speed is different and also the slope. Finally, the file of ranges was defined as we show on table 5.9.

Parameter	Initial	Final	Step
Model	1	13	1
WindSpd	12.97	26.619	0.8
WindDir	0.0	360.0	5.0
Slope	21.0	21.0	1.0
Aspect	180.0	180.0	1.0
M1	0.1	0.3	0.2
M10	0.1	0.5	0.2
M100	0.1	0.5	0.2
Mherb	0.1	0.1	1

Table 5.9: Parameters file for Plot 533

In figure 5.7 we can see the real area. It shows the fire spread at intervals of 2 minutes. The final time has been fixed at minute 12. It is interesting to see the wind effect. This is an experiment started with only one ignition point. Because of direction and speed of wind, fire grows in an elliptical way.

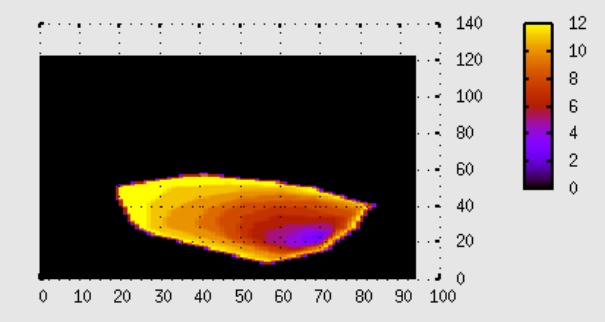

Figure 5.7: Real spread for plot 533, considering steps of 2 minutes

After the application of each method, we obtained the fitness values shown on the table 5.10.

This is an interesting case to analyze, because in it our method has a lower fitness than the others on two of four points. Such a situation has

Initial Time	Final Time	Fitness		
		S^2F^2M	GLUE	Evolutionary
2.0	4.0	-	-	-
4.0	6.0	0.4745	0.3554	0.4326
6.0	8.0	0.5208	0.5788	0.8292
8.0	10.0	0.4675	0.5748	0.6914
10.0	12.0	0.4501	0.3646	0.6654

Table 5.10: Comparative of found fitness in each method for plot 533

an explanation, which is related to the method. If we look at figure 5.8, in this simple example we have only three scenarios. Scenario number 3 has a behavior similar to the real case, but when we choose $P_{ign}(1)$ or $P_{ign}(2)$ or $P_{ign}(3)$, we discover that on each case fitness is lower than those individual cases.

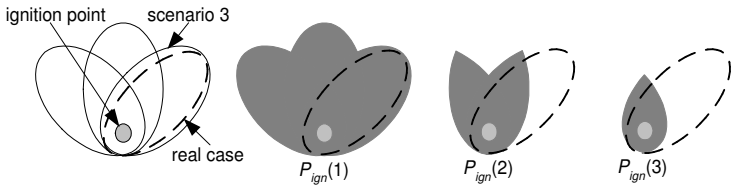

Figure 5.8: Explanation

On the other hand, in this experiment it is interesting to observe that, though the fitness reached by S^2F^2M is not very high, the value obtained by this function is really constant (around 0.5). This shows that although simulator can not correctly reproduce the fire behavior, it attempts to follow the real behavior (that, as we can see in figure 5.7, is quite constant).

Finally, with exception of the first step, in this experiment the best results are achieved by Evolutionary method.

5.5.3 Experiment 3: Plot 534

This experiment has similar characteristics to the previous one (the dimensions are analogous and both are started with a single ignition point), but its duration is inferior. The initial time t_0 was chosen as minute 3. Since the step was defined to one minute, the first adjustment was made on minute 4, and, therefore, the first prediction was effectuated on minute 5.

Parameter	Initial	Final	Step
Model	1	13	1
WindSpd	0.0	19.014	1.5
WindDir	0.0	360.0	5.0
Slope	19.0	19.0	1.0
Aspect	180.0	180.0	1.0
M1	0.1	0.3	0.2
M10	0.1	0.5	0.2
M100	0.1	0.5	0.2
Mherb	0.1	0.1	1

Table 5.11: Parameters file for Plot 534

The parameters file is listed in table 5.11

If we compare the parameters files of plots 534 and 533, it is possible to distinguish the difference between the wind speed range. This is a very important factor that could modify entirely the fire behavior.

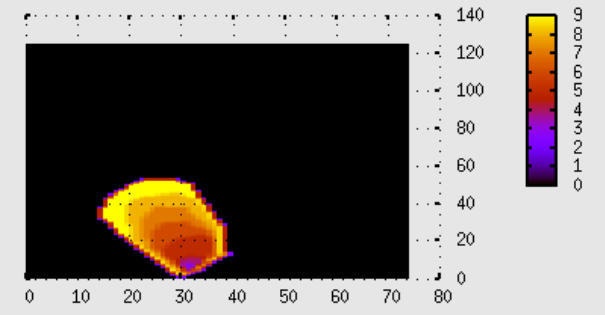

Figure 5.9: Real spread for plot 534, considering steps of 1 minute

Figure 5.9 shows that in this case the wind speed is inferior that in the previous one. However, the big problem is that the speed range is wider than in the previous case. This factor is a very inconstant parameter, and, although we consider the amplitude of variation of such parameter, this can affect the prediction quality. Table 5.12 presents the obtained fitness.

It is interesting to observe that the three methods have a similar behavior relating to the fitness reached: in all three cases the fitness function has an ascendant behavior until a certain point (final time 7 for GLUE and S^2F^2M, and final time 8 for Evolutionary). After this point, the fitness function declines (only Evolutionary method maintains certain level).

As we mentioned before, for a more detailed explanation about this fact,

Initial Time	Final Time	Fitness		
		S^2F^2M	GLUE	Evolutionary
3.0	4.0	-	-	-
4.0	5.0	0.2565	0.2056	0.2162
5.0	6.0	0.5336	0.5820	0.4811
6.0	7.0	0.7036	0.6466	0.6358
7.0	8.0	0.6074	0.5025	0.7728
8.0	9.0	0.4161	0.3122	0.7468

Table 5.12: Comparative of found fitness in each method for plot 534

we provide discussion about it later on this chapter (section 5.6).

5.5.4 Experiment 4: Plot 751

This is another short experiment. The reason is the reduced plot size. For this cause, the representation was 60 x 90 cells, using a smaller cell size. In this way, we can obtain a better definition. The duration of the burn in the experiment is 9 minutes. In this case, we applied the methods four times at time instances 4, 6, 8 and 10 minutes. The table 5.13 shows the parameters range. The value of slope is 6 degrees, therefore, the land inclination is not a determinant factor in the fire behavior, at least for this case.

Parameter	Initial	Final	Step
Model	1	13	1
WindSpd	0.0	12.3	0.67
WindDir	0.0	360.0	5.0
Slope	6.0	6.0	1.0
Aspect	180.0	180.0	1.0
M1	0.1	0.3	0.2
M10	0.1	0.5	0.2
M100	0.1	0.5	0.2
Mherb	0.1	0.1	1

Table 5.13: Parameters file for Plot 751

In figure 5.10 we can observe that the lines of the fire appear regularly, which reflects that the fire is quite continuous. Also, we can see that this is another case of linear ignition, started with pyrotechnic devices.

The results are summarized in table 5.14. Once again we can easily conclude that S^2F^2M method provides better results than the GLUE scheme and Evolutionary method.

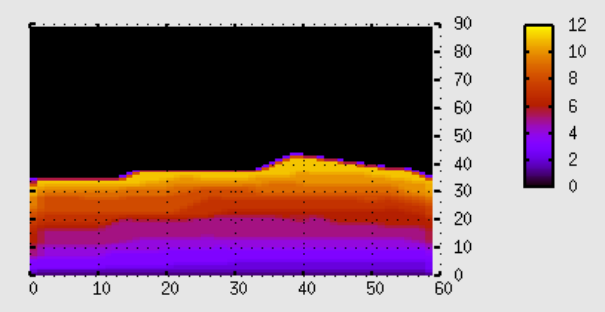

Figure 5.10: Real spread for plot 751, considering steps of 2 minutes

Initial Time	Final Time	Fitness		
		S^2F^2M	GLUE	Evolutionary
2.0	4.0	-	-	-
4.0	6.0	0.8819	0.6895	0.8230
6.0	8.0	0.7917	0.7083	0.7450
8.0	10.0	0.7073	0.6068	0.7068

Table 5.14: Comparative of found fitness in each method for plot 751

5.5.5 Experiment 5: Plot 752

The last experiment, as the previous, has a reduced plot size. For this reason, once again we define a small cell size to maximize the number of cells in the grid. We defined the cells dimensions equal to the Plot 751. The duration of this experiment was very short. A possible reason is the wind effect: with a high wind speed the ROS (ratio of spread) and the flame intensity can become very high. The combination of these factors produces a fast propagation, and, therefore, a more dangerous fire.

In this case, the plot was burned by linear ignition on left border. This, and the posterior advance to right size, can be observed on figure 5.11.

The table 5.15 shows the values of the parameters that have been used for the experiment.

Finally, table 5.16 lists the resultant fitness values after applying the methods to predict the fire behavior on this experiment.

In this last experiment, we discover that, though the fitness values are very similar between the three methods, in general we can see that the statistical method is better than others. Also, it is positive that statistical

Parameter	Initial	Final	Step
Model	1	13	1
WindSpd	2.13	5.81	0.21
WindDir	0.0	360.0	5.0
Slope	6.0	6.0	1.0
Aspect	180.0	180.0	1.0
M1	0.1	0.3	0.2
M10	0.1	0.5	0.2
M100	0.1	0.5	0.2
Mherb	0.1	0.1	1

Table 5.15: Parameters file for Plot 752

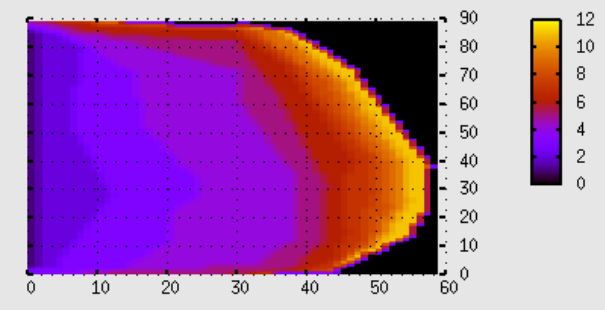

Figure 5.11: Real spread for plot 752, considering steps of 2 minutes

predictions are increasing on each step. Contrary to that, GLUE predictions have been declining on each point. On the other hand, the Evolutionary prediction has a behavior very similar to S^2F^2M.

Initial Time	Final Time	Fitness		
		S^2F^2M	GLUE	Evolutionary
2.0	4.0	-	-	-
4.0	6.0	0.4976	0.5092	0.4640
6.0	8.0	0.5257	0.4984	0.4539
8.0	10.0	0.6451	0.4948	0.4964

Table 5.16: Comparative of found fitness in each method for plot 752

5.6 Summarizing the experiments

The following figures (5.12, 5.13, 5.14, 5.15, 5.16) correspond to the tables (5.8, 5.10, 5.12, 5.14, 5.16), but they show graphically the fitness behavior on each method and for each experiment.

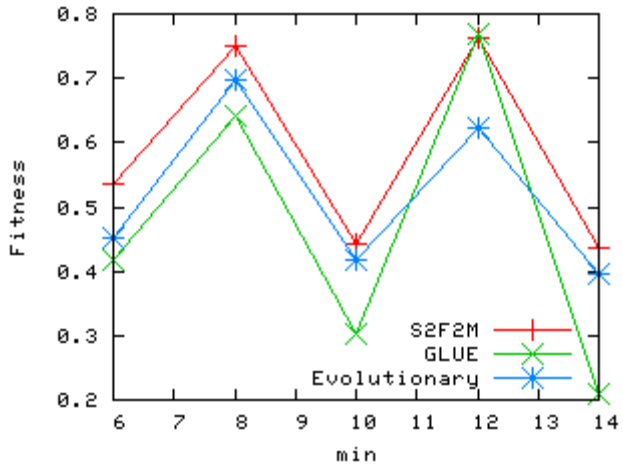

Figure 5.12: Plot 520 - Comparison between the methods results

In general, we saw that fitness function increases in certain intervals and decreases in others. The reason could be because of quick weather changes which are present during the burning. For example, if we find a parameter set that is suitable for a specific time z, and in this set of parameter the wind speed value is high, when we use this set of values in a time $z+1$ where the wind speed value changes suddenly, this new prediction will be distant from the real situation.

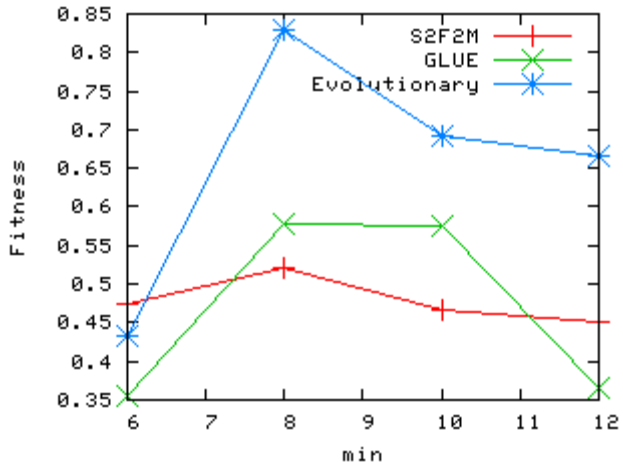

Figure 5.13: Plot 533 - Comparison between the methods results

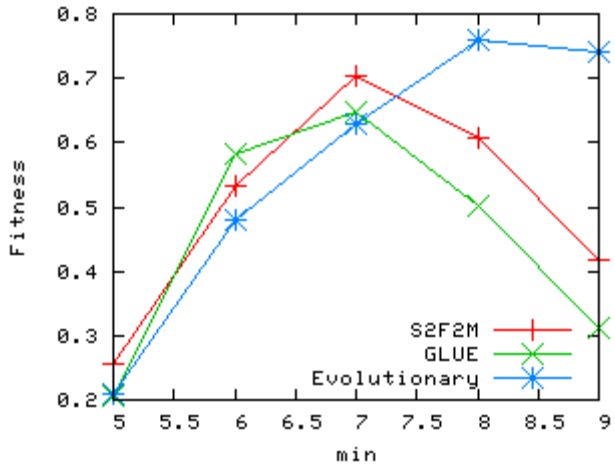

Figure 5.14: Plot 534 - Comparison between the methods results

Furthermore, there is always the possibility of not finding a good parameters set that explains the fire behavior. This is the case when the fitness is lower than previous step; we found between all cases analyzed a certain number of scenarios which in some aspects have a similar behavior to the real scenario, but in general the global case has an acceptable behavior. In the same way, we also have positive peaks where the fitness is really good, and, therefore, they can be useful as predictors.

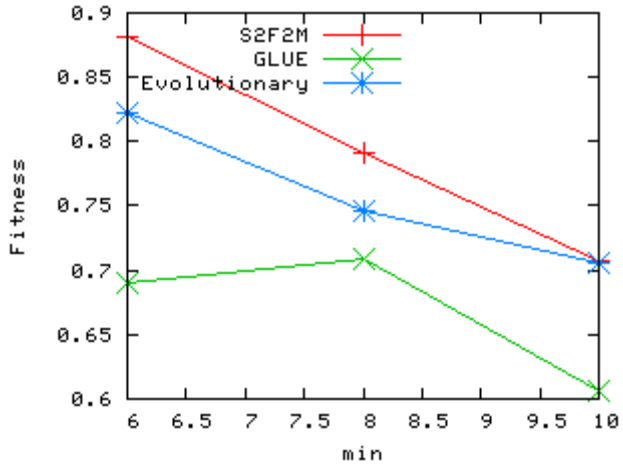

Figure 5.15: Plot 751 - Comparison between the methods results

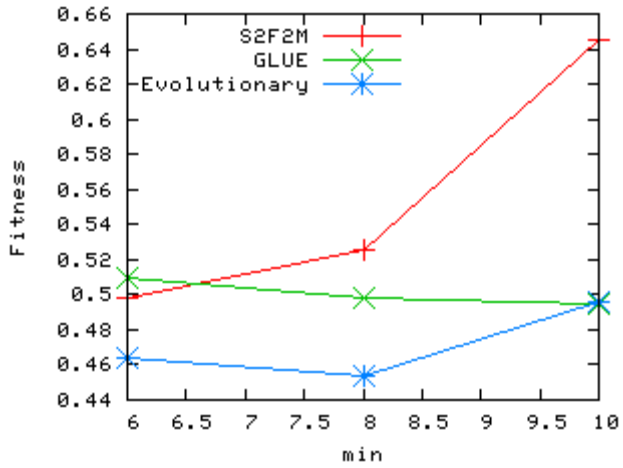

Figure 5.16: Plot 752 - Comparison between the methods results

5.7 Speed-up Test

As we mentioned on chapters 3 and 4, the three methods work under a parallel scheme to reduce the execution time. This has an explanation: each method needs to effectuate a large quantity of computation (in average, according to the experiments done, they have computed around 65,000 simulations on each step), which always translates into a substantial cost on time. Using the parallel processing it is possible to reduce the time spent in a drastic way.

In this section we will address how much the parallelism can reduce execution time, and how much it can scale by adding more processors. For this reason, we have analyzed speedup improvement for the methods as the number of processors increases. For this reason, the performance has been analyzed using the measure known as Speed-Up, which is derived from Amdahl law. Originally, this law was defined to show that vectorizing of a program can only affect that part of the program which lends itself to vectorizing. The Amdahl law can be written as:

$$S_{eff} = \frac{S_f}{S_f(1-f)+f}$$

where f is the fraction of the program that can be improved, S_f is the improvement factor on this fraction, and S_{eff} is the overall improvement

achieved. Obviously, for small f, $S_{eff} \approx 1$, whatever the value of S_f, i.e. insignificant overall gain is achieved.

As an alternative, we can see the formula only expressed in function of execution time [57]. Speedup is defined as the time it takes a program to execute in serial (with one processor) divided by the time it takes to execute in parallel (with many processors). The formula for speedup S is:

$$S = \frac{t(1)}{t(N)}$$

where $t(1)$ is the running time of the best available sequential algorithm and $t(N)$ is the running time of the parallel algorithm using N processors.

As we mentioned before, the proposed methods have been executed in a Linux cluster under MPI environment. The number of processors used were 1, 2, 4, 8, 16 and 32. Figure 5.17 shows the values obtained as an average of all experiments. In purple colour we can see the S^2F^2M speed-up, in brown the GLUE speed-up, in blue the Evolutionary speed-up and in cyan the linear case (the ideal case). From 4, GLUE method decreases lightly until 16. This behavior is even worse between 16 and 32 processors. Moreover, S^2F^2M method maintains the same level of performance from 1 to 32 processors, maintaining itself lightly under the linear case.

A different subject is the execution time. If we only consider the execution time, the method with better behavior is GLUE. At first, this appears like a S^2F^2M disadvantage, but when we consider the speed-up reached by each method, we discover the following: using 32 processors, this difference in time becomes less than a half. An example of this is the scenario number three (plot 534). When we are using one processor, the execution time for S^2F^2M is 1297.6 minutes (21.6 hours), and the GLUE time is 307 minutes (5.11 hours). Here we can see that the relationship is 1 to 4 approximately. Therefore, when we use 32 processors, the S^2F^2M time is only 44.8 minutes, and GLUE spends 23.9 minutes to finalize.

Also, there are even better situations, like for example experiment three. In this case, when we are working with one processor, the relationship is almost 1 to 3, but if we apply the method over 32 processors, the execution time reached is practically the same (5.9 minutes for S^2F^2M and 5.2 minutes for GLUE).

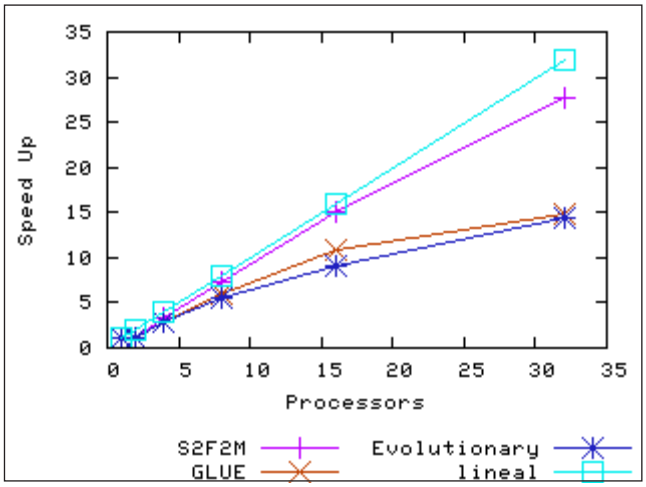

Figure 5.17: Speed-up for the methods for different number of processors

According to the measurements of time made on the experiments, the ratio between calculation time and communication time indicates that in average 98% of the total execution time is used for the calculation of the simulations on the side of workers. Therefore, sending and reception of data between master and workers are well overlapped within the total time of execution. Nevertheless, because data size handled between the master and each worker is not large (in average in our experiments the real land file has 44 Kbytes and the file of parameters has 200 bytes), the time dedicated to the data communication is little.

Related to this last topic, we can explain the low speed-up reached by GLUE when there is an important number of processors. GLUE method needs a smaller amount of operations (S^2F^2M works in a intensive way on matrixes), then the necessary time by each worker does not reach to overlap the communication, and for this reason, when the parallelism level is very high, the successive increase in the number of processors does not offer the speed-up awaited.

On the other hand, in different experiments where the land has greater dimensions, the time dedicated to the communication can be larger. However, this is a factor of relative importance since the tradeoff between time and level of detail of the cells (wide and long) can be modified according to the requirements and, therefore, a land with large dimensions can be represented

by means of a little matrix with large cells.

If we paid attention to the dimensions of each plot, we can see that plot 751 and 752 are smaller than the others. However, we represent them with matrixes not much smaller than other plots, since the cell size was 1/3 of others cells (0.333 vs. 1 m). The resulting land files in all the cases have similar sizes although the land that represent are different. This has direct relationship with the tradeoff between time and precision. If we want, we can obtain a result quickly, but sacrificing the quality of our prediction.

Chapter 6

Applying S^2F^2M on real situations

> "En realidad las cosas verdaderamente difíciles son todo lo que la gente cree poder hacer a cada momento."
>
> Julio Cortázar

As we previously commented, the S^2F^2M method was, at the beginning, only considered as an statistical method to be added to a DSS. The method, working in this manner, operates correctly (this was shown in [16]). For this reason, we evaluated our method within a grid environment, and, as a different test, our method was used as a tool to generate Risk Maps. Following we explain these topics.

6.1 Grid Computing

Sun Microsystems Ibérica along with the Universidad Politécnica de Valencia (UPV), the Parque Científico de Madrid (PCM), the Universidad Complutense de Madrid (UCM) and the Universitat Autònoma de Barcelona (UAB) made a conjunct work. This was performed within the framework of the Summer course called "High Performance Computing, Clusters and Grid Computing" [53] (El Escorial - Spain, July 2004). The practical part of this course consisted in testing the connection of the nodes of the five mentioned centers, with the purpose of verifying the efficiency reached when

Grid Computing technology is used. In figure 6.1 it is possible to visualize the scheme of this grid.

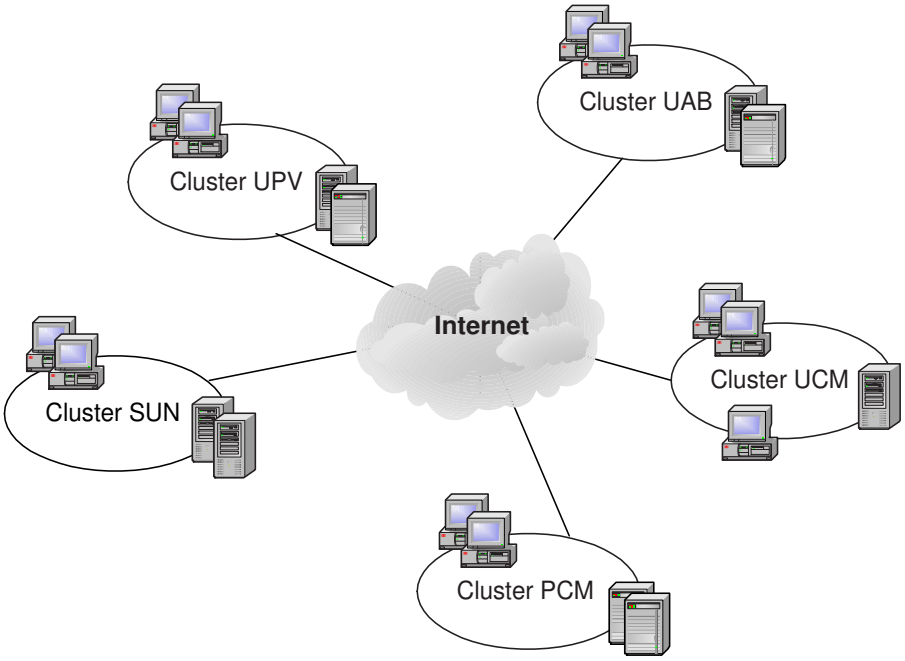

Figure 6.1: Scheme of grid interconnection

The initiative is framed within the **Proyecto de Portal de Ciencia Colaborativa** for the Spanish Researching Community. In this project a total of eight universities and three research centers work together. Foundation Telefónica and Sun Microsystems Ibérica [66] also participate.

In order to carry out this test, all the necessary infrastructure of hardware, software and communications, have been prepared on Linux and Solaris to start up the Grid, in which have been executed successfully applications like treatment of medical image, design of processors, and our method of fire prediction.

This experience allows the users to proceed to the gradual definition of the concrete programs of research, applications and processes that would benefit from this project, between which we can identify different areas: Physics, Mathematics, Sciences of the Life and other fields of Biomedical Computer Science. Also, the application of the additional resources of computation

would adapt to the necessities that the scientific community has in any other ambit, in Computer Science equipment performance, of applications and data treatment in general.

To do this experiment, we work with four plots: two real cases and two done in laboratory. We verified that when in our own cluster were launched other tasks which competed for CPU time the processes migrated to the other clusters. Finally, we saw that the results were correct and they did not take more time than in a normal situation.

The latest objective of the project is to create a platform that allows to use in an efficient form the resources of the different groups from investigation and Universities, multiplying the possibilities that the Grid technologies offer at the current time.

6.2 Risk Maps

As we mentioned before, we also used the method to generate risk maps. Most fire-risk mapping techniques evaluate the ignition danger based on meteorological conditions (temperature, humidity, rainfall, etc) and on human factors (negligence, arson, etc.), but do not take into account the propagation danger itself once a fire has been ignited. This feature is very important, because it can provide more complete information to determine the possible behaviour of a wildland fire and to determine those regions where an ignition is more dangerous by considering the possible rate of spread or flame intensity.

The fire behavior depends on static factors (for instance the slope of the terrain or the vegetation type in that particular region), but also on certain dynamic factors such as the wind conditions or the moisture content in the vegetation. It is not therefore possible to previously determine the present conditions when a fire begins. Finally, it is not possible to evaluate beforehand the effective rate of spread or flame intensity in a real situation.

Considering this uncertainty, our statistical method is suitable for determining the possible rate of spread and flame intensity. The method proposed takes the static parameters of the region under consideration (slope and fuel type) and applies statistical analysis by simulating the fire propagation, con-

sidering a wide range of parameter combinations to determine the average rate of spread and flame intensity in that particular region. This rate of spread and flame intensity represents the potential propagation danger for that region.

This methodology requires each region to be represented by an average slope and a dominant fuel type. The size of such cells cannot therefore be too large, so as to maintain a high degree of uniformity.

The methodology is promising, but it is not useful if only applied at a local scale: it must also be applied at a regional, national or even international scale (i.e. the Mediterranean region). Therefore, the whole region under consideration must be divided into a set of uniform areas, and the methodology must then be applied to all the areas within the region as a whole to provide a wildland-fire propagation danger map.

6.2.1 Rate of spread and flame intensity of a particular area

To determine the rate of spread and the flame intensity of an area, it is necessary to apply some form of statistical method, since we have a set of values corresponding to each scenario (for this reason, we can apply our system). One possibility would be to select the maximum value as being the value that represents the area under study. This approach would represent the most dangerous scenario, as the behaviour of that area and this approach is very restrictive, corresponding as it does to extreme situations that are not completely representative. The average of these results is more feasible than the value representing the area.

The output generated by the simulator consists of two maps of the terrain. In the first, each cell is labeled with its ignition time; in the second, each cell is labeled with its flame height. This information must be used to calculate the rate of spread and an average from all flame heights.

To calculate the rate of spread, the distance between the ignition point and each particular cell in the terrain is divided by the ignition time of that particular cell. This calculation is repeated for each cell in the terrain to determine the maximum value of the rate of spread. This maximum value will be used as the rate of spread for that particular situation (figure 6.2).

The flame intensity (height) for that terrain is calculated by evaluating the average flame intensity in the whole terrain.

To evaluate the rate of spread for a particular scenario (input parameter combination) in a given terrain, it is therefore necessary to simulate the fire propagation and then estimate the rate of spread and average flame height. This single calculation for a particular scenario is not very computer demanding and can be carried out on a single PC in a few seconds.

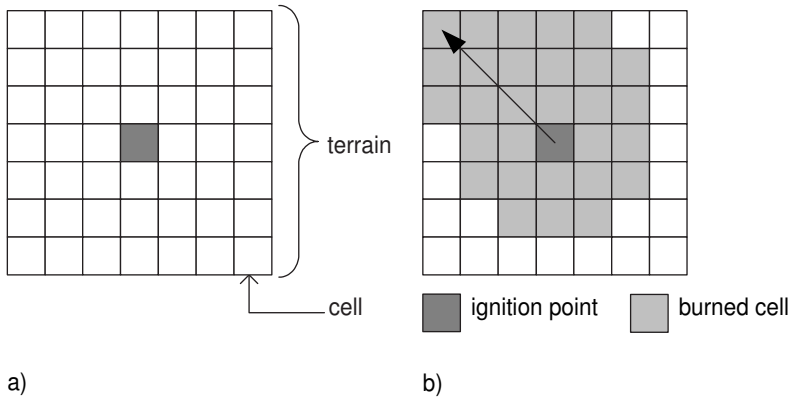

Figure 6.2: Methodology for calculating rate of spread. a) Ignition point in the middle; b) Search for maximal spread

However, it should be recalled that the amount of scenarios to be evaluated for each terrain (slope and fuel type) is very large, as is the number of different terrains. Therefore, the total computation time required to estimate all the rates of spread is extremely large. The use of parallel/distributed systems therefore appears as the only solution to make this approach feasible.

6.2.2 Experimental results

In order to obtain a global view of risk associated to fuel loads, terrain characteristics and wind flows, a global simulation analysis was performed. This analysis tried to obtain average values of rate of spread and flame length, considering different wind and topographic conditions for the estimated fuel maps of the whole EUMed area. This attempt should be considered as a general overview of average expected fire behaviour at global scale, in order to rank different danger levels according to the combination of fuel

and terrain spatial patterns.

To provide the propagation danger map, we created a set of prototype plots, considering all the fuel models from Rothermel classification and a certain slope percentage (from 0 to 100%, with a step of 5%). On figure 6.3 we can see an scheme of this prototypes generation.

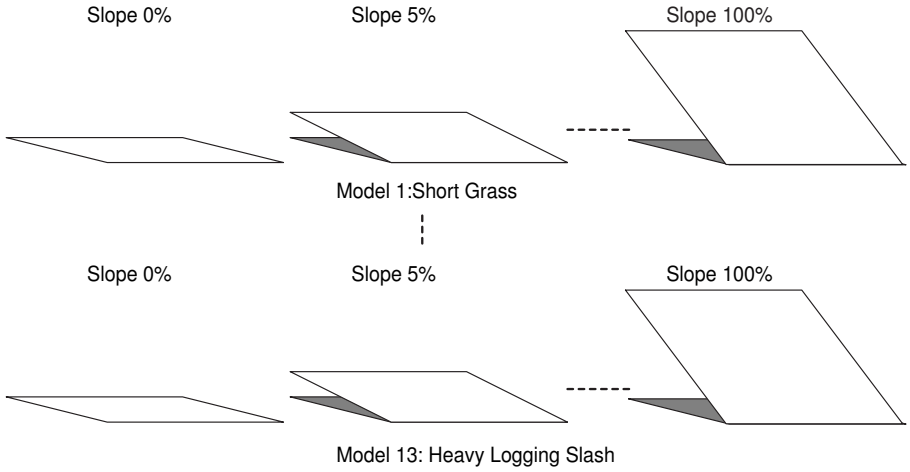

Figure 6.3: Synthetic Plot generation

The total number of plots was therefore 273. Each plot consisted of a grid of cells with 11 columns x 11 rows (each cell measured $100m^2$). The ignition point was located in the middle of plot. For each plot, many input parameter combinations were used to simulate the wildland fire behaviour and the average rate of spread and flame height were also calculated. The parameters considered for variation were: 1-hr dead fuel moisture, 10-hr dead fuel moisture, 100-hr dead fuel moisture, live herbaceous moisture. The ranges applied to these parameters and the precision considered were those established by Farsite Simulator [30]. These values are shown in table 6.1.

Parameter	Inferior Threshold	Superior Threshold	Increment
1-hr dead fuel moisture	0.03	0.12	0.01
10-hr dead fuel moisture	0.04	0.13	0.01
100-hr dead fuel moisture	0.05	0.14	0.01
Live herb. fuel moist.	0.7	1.7	0.3

Table 6.1: Parameter values

Considering these ranges and precision steps, the number of simulations per plot was 4300, and the total number of simulations was 1,173,900.

For each independent result (scenario), a value of flame height is obtained as the average among the flame height for each cell. Then, the value used is the average for all different resulting cases from combinations of moisture content. These values are shown in figure 6.4. It can be observed that each model has a well-defined height rank, with a minimum and a maximum. However, it is interesting to observe that, in certain cases (for example, models 3 or 4), this is not completely linear or incremental according to the slope.

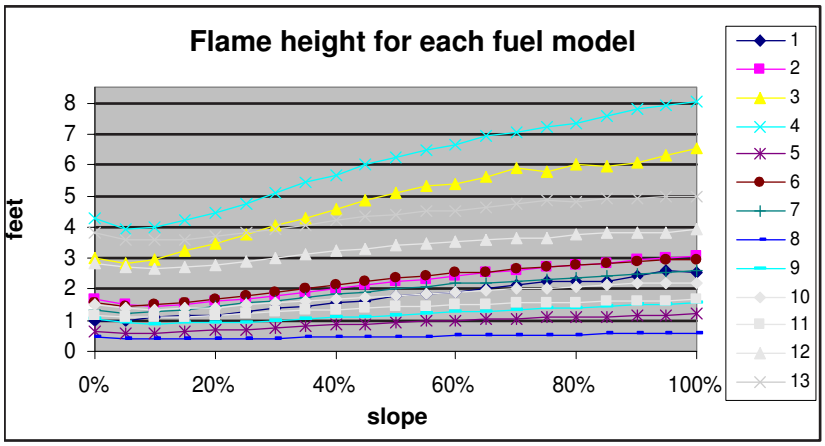

Figure 6.4: Flame height

On the other hand, each rate of spread found is averaged to calculate a representative value. These values are presented in figure 6.5. In this diagram, we can see that the more inclined the terrain, the faster the propagation. Therefore, the more dangerous the fire.

Observing the two figures at the same time, it is possible to conclude that flame height has no direct relation to rate of spread. For example, model 1 has a high rate of spread on a high slope, but its average flame height is not particularly great. This behaviour must be taken into account to avoid erroneous conclusions: in a wildland fire, this could be very dangerous.

Finally, these results can be applied to an European scale. To do so, S^2F^2M requires a European map divided into cells including the average

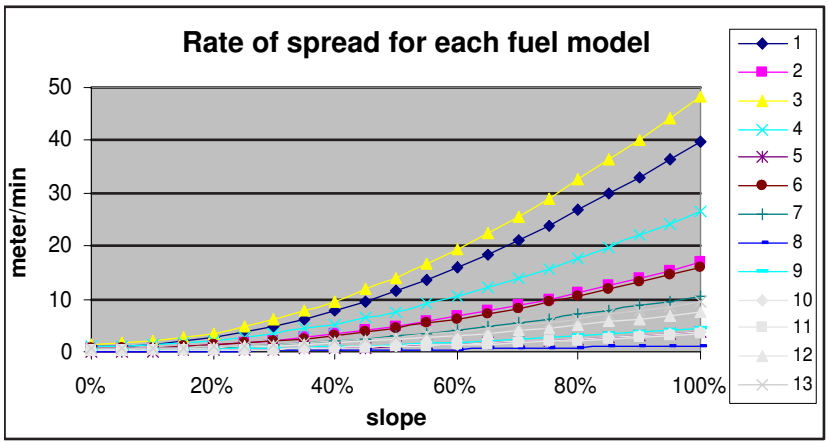

Figure 6.5: Rate of spread

slope and dominant fuel model for each cell. When the European maps are available, it is possible to elaborate two different maps: one for rate of spread-propagation danger and one for flame-height propagation danger. These maps are shown in figure 6.6 and figure 6.7.

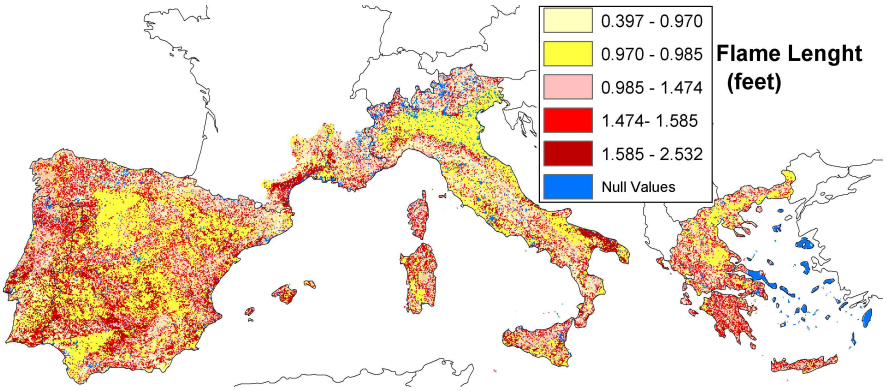

Figure 6.6: Flame-height propagation danger Map of the EUMed

For more information, the reader can see a more complete description about these results on [18], [20] and [27].

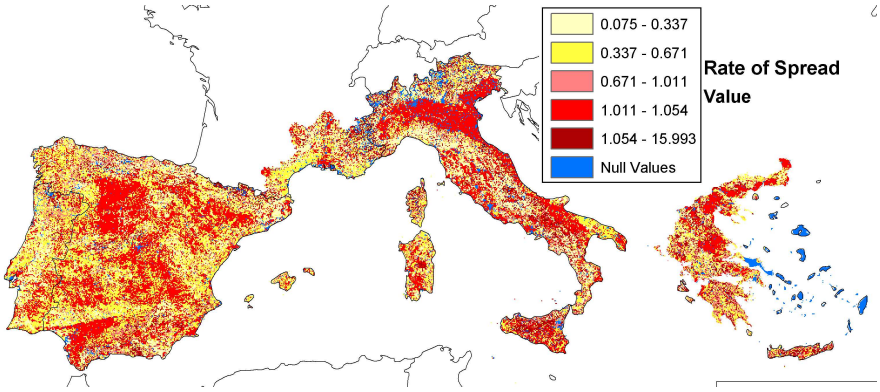

Figure 6.7: Rate of spread-propagation danger Map of the EUMed

Chapter 7

Method Refinements

> "La vida es tan corta y el oficio de vivir tan difícil, que cuando uno empieza a aprenderlo, ya hay que morirse."
>
> Ernesto Sabato

In this chapter, we want to describe some changes and optimizations applied to the S^2F^2M method. Although some of these changes did not report the expected improvement over the method, we believe it is important to explain and document these variations over the method. At a first moment these approaches appear like a good solution or improvement, but after doing the experimentation, we found the problem suffered by these techniques.

Having in mind the needed time to execute any of the described methods, in this chapter we describe some modifications included to reduce the number of iterations or simulations necessary to reach a "good" solution. Mainly, these techniques have the goal of finding a better solution than classical predictions, but on the other hand, in comparison they spend much more time. Then, we are also interested in reducing the time, maybe to reach real time.

We did three types of tests: on one hand, applying the sensitivity index proposed by B. Abdalhaq in [3] over the set of parameters; and, on the other hand, we tried to collect only the best scenarios and discard all those bad cases and, finally, we tried to reduce the parameter range using additional information.

7.1 Applying the Sensitivity Index

As we mentioned above, with the goal of finding a better prediction, we applied one of the results obtained by B. Abdalhaq: the sensitivity analysis. This analysis *"asses the impact on output of each input parameter and, consequently, it will allow us to determine which parameters are worth spending time on tuning and which are better to avoid spending such effort on"* [3].

At first, as it is possible to understand from the previous paragraph, the sensitivity analysis was proposed as a necessary way of accelerating the convergence of Evolutionary Method. This acceleration was reached by mean of three ways:

1. Fixing some parameters to their nominal values
2. Introducing a certain degree of knowledge
3. Sampling the search space

At that moment was proved that these techniques were useful to accelerate the optimization process and, therefore, they would be useful in the case of real-time constraint.

The way to calculate the sensitivity index was focused on the propagation speed, thus the wind had only one scalar value, which is the wind speed in the direction of fire propagation. To calculate the sensitivity index for each parameter, it was necessary to define a minimum and a maximum value for the parameter. These values were obtained from field and lab measurements. For all possible combinations of the other parameters, two simulations were needed, considering the minimum and maximum value of the parameter studied currently. The speed difference between both propagation simulations represents the effect of changing that particular parameter from its minimum to its maximum for some particular combination of the other parameters. If V_{ik} is the effect of varying factor i from its minimum to its maximum at case k, the total effect of parameter i is defined as the addition of the effect of each possible case:

$$V_i = \sum_{\nabla k} V_{ik}$$

where k is all possible cases.

Finally, V_i is the sensitivity index for the parameter i. Table 7.1 shows the values obtained for each parameter. For example, the most important parameter is load with a sensitivity equal to 0.77.

Parameter	Symbol	Index
Surface area-to-volume ratio	s	0.56
Low heat of combustion	H	0.13
Loading	l	0.77
Moisture content	M_f	0.61
Total silica content	St	0.03
Effective silica content	Se	0.16
Wind speed at mid flame height	U	0.71
Dead fuel extension moisture content	df	0.28

Table 7.1: Sensitivity index for each parameter

On the information presented in table 7.1, it is necessary to clarify that some values (for example the humidity content M_f) are considered differently from chapter 2. The reason is that this table was calculated to be used in a different simulator (ISS forest-fire simulator [42]), which uses a slightly different set of parameter. Then, to apply them to fireSim, we adapt them to our format. However, the problem is minor, because both simulators are based on the same scheme (Rothermel model [61]) and, although the number of parameters is not the same, in the background they represent the same behavior, because some parameters are dependent from others, and viceversa.

In our case, we applied this index with a different intention. We wanted to avoid wasting simulations on cases that do not contribute to the final results. That is to say, if a case doesn't propagate, it doesn't collaborate because in the sum of all cases it is a null case that only spends processor time. In this way, we use the index sensitivity to adjust the values of step in the parameter file we use to feed the S^2F^2M system. We respected the same number of scenarios, but the variations of parameters are different. That is to say, now the responsible for generating more o minus scenarios

are different from the proposed experiments of chapter 5. Table 7.2 presents the new values for each parameter after applying the index.

	Step				
Parameter	Plot520	Plot533	Plot534	Plot751	Plot752
Model	1	1	1	1	1
WindSpd	1.0687	4.549	4.75	1.23	0.368
WindDir	32.4	81.0	60.0	51.42	51.42
Slope	1.0	1.0	1.0	1.0	1.0
Aspect	1.0	180.0	1.0	1.0	1.0
M1	0.049	0.066	0.066	0.066	0.066
M10	0.39	0.133	0.199	0.133	0.133
M100	0.39	0.133	0.199	0.39	0.39
Mherb	0.39	0.133	0.199	0.39	0.39

Table 7.2: Parameters file for each plot using sensitivity analysis

Afterwards, we re-executed the simulations for each experiment, and found the results shown in tables 7.3, 7.4, and 7.5.

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.4214	0.5345
6.0	8.0	0.7058	0.7495
8.0	10.0	0.3333	0.4413
10.0	12.0	0.7604	0.7625
12.0	14.0	0.2437	0.4354

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.5034	0.4745
6.0	8.0	0.6110	0.5208
8.0	10.0	0.4861	0.4675
10.0	12.0	0.3923	0.4501

Table 7.3: Fitness for plot 520 and plot 533 using sensitivity analysis

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
3.0	4.0	-	-
4.0	5.0	0.1601	0.2565
5.0	6.0	0.3293	0.5336
6.0	7.0	0.4238	0.7036
7.0	8.0	0.4449	0.6074
8.0	9.0	0.4301	0.4161

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.8482	0.8819
6.0	8.0	0.7400	0.7917
8.0	10.0	0.7310	0.7073

Table 7.4: Fitness for plot 534 and plot 751 using sensitivity analysis

As we can see, in general terms, the information presented on tables demonstrates that when we use the sensibility analysis, the prediction done by S^2F^2M does not improve (the values shown in column *Fitness* S^2F^2M*

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.3914	0.4976
6.0	8.0	0.4540	0.5257
8.0	10.0	0.6038	0.6451

Table 7.5: Fitness for plot 752 using sensitivity analysis

are the original values described on chapter 5.). If we pay attention to each particular case, we can observe in Plot 533 the fitness value on each step is higher than in the original experiment. Therefore, at least for these five examples, only in one case the sensitivity index is useful, hence, in a 20%. However, this can have another explanation; it is strange that the only case improved is the same case that in the original experiment had a fitness very slow on each stage.

Maybe in this situation, using the values modified by index helped to guide to search for good results. However, in general we believe that to allow a wide range is better for the method behavior.

7.2 Considering only the best scenarios

Another type of test done using the intrinsic characteristics of S^2F^2M was to select only those scenarios that surpass a certain threshold-fitness.

The goals of this refinement were:

- reaching a better result (a prediction nearest to reality)
- accelerating the method (obtaining result in less time).

The first objective is similar to the goal explained in the previous section, but with some differences. In this case, after each simulation we evaluate the fitness value for each scenario. If the value reaches a previously defined threshold-fitness, we allow the scenario participate in the next step. In this way, we would be working only with a set of good scenarios, and, as the number of scenarios will decrease on each step, the simulation time should be inferior.

The reasoning was the following: when we are doing a simulation, we hope that the values for every parameter will be in certain domain; to wit, inside demarcated margins. Therefore, the prediction level should be improved step by step. Avoiding the cases that are distant from the real case, the error level added to the statistic result must tend to zero. In figure 7.1 we can see the idea.

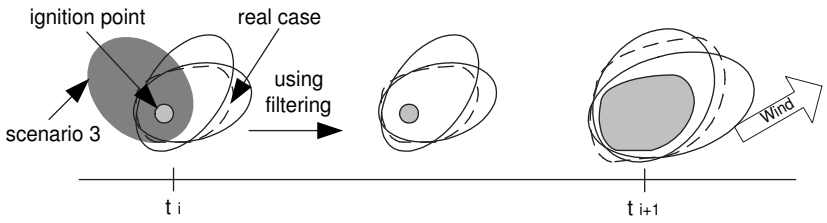

Figure 7.1: The method allows only the best cases and discards those that do not surpass the threshold

It is possible to observe that in this simple example (figure 7.1), *scenario 3* has a distant behavior from real case. Let us to suppose that it does not surpass the threshold-fitness. Then, *scenario 3* is avoided and it is not used in the next step (t_{i+1}). As we can see, the other scenarios have a similar behavior to the real case, and, therefore, the final prediction fitness will not be very different from reality. However, in figure 7.2 we can discover a problem that arises when meteorological conditions are too inconstant.

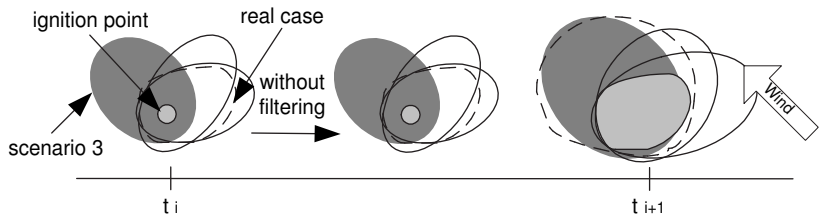

Figure 7.2: The method allows good and bad cases in order to contemplate different situations

In this example, we use again the statistical method without refinements. Here, we observe again the same scenarios than in the previous example. Nevertheless, now we permit *scenario 3* work in step t_{i+1}. Notice that al-

though *scenario 3* has a very different behavior from other cases, in this situation is very useful, because wind conditions have changed. Now, *scenario 3* coincides with real case, and thanks to it, the prediction in step t_{i+1} is better. If we cut *scenario 3* like in the example shown in figure 7.1, the final result would be worse (figure 7.3).

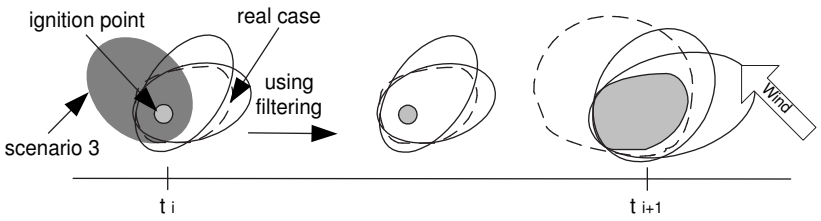

Figure 7.3: Possible problem when the method allows only the best cases

This is only a simple schematic example, anyhow, this was demonstrated doing the appropriated experimentation. The results can be observed on tables 7.6, 7.7 and 7.8 (*Fitness* $S^2F^2M^*$ means the original values).

Initial Time	Final Time	Fitness S^2F^2M	Fitness $S^2F^2M^*$
2.0	4.0	-	-
4.0	6.0	0.4394	0.5345
6.0	8.0	0.5362	0.7495
8.0	10.0	0.2804	0.4413
10.0	12.0	0.3493	0.7625
12.0	14.0	0.2263	0.4354

Initial Time	Final Time	Fitness S^2F^2M	Fitness $S^2F^2M^*$
2.0	4.0	-	-
4.0	6.0	0.2820	0.4745
6.0	8.0	0.2889	0.5208
8.0	10.0	0.3590	0.4675
10.0	12.0	0.3099	0.4501

Table 7.6: a) Fitness for plot 520 using a threshold-value equal to 0.8 b) Fitness for plot 533 using a threshold-value equal to 0.53

It is important to clarify that the values chosen as threshold are based on the possible values reached by scenarios on each plot. These values have been obtained during the experimentation stage explained on chapter 5 (it does not have sense to put like threshold a value that no panorama could reach, because we will lose all possible scenarios in the first step).

Moreover, it is important to know which is the percentage of "cutting", that is to say, which is the ratio between the number of scenarios discarded and the number of scenarios that will be used in the next step. In all studied cases, this cut has been found between the first and second step. In general,

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
3.0	4.0	-	-
4.0	5.0	0.2429	0.2565
5.0	6.0	0.4200	0.5336
6.0	7.0	0.5833	0.7036
7.0	8.0	0.6824	0.6074
8.0	9.0	0.5849	0.4161

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.8721	0.8819
6.0	8.0	0.7810	0.7917
8.0	10.0	0.7142	0.7073

Table 7.7: a) Fitness for plot 534 using a threshold-value equal to 0.57 b) Fitness for plot 751 using a threshold-value equal to 0.85

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.4589	0.4976
6.0	8.0	0.3083	0.5257
8.0	10.0	0.2705	0.6451

Table 7.8: Fitness for plot 752 using a threshold-value equal to 0.77

those scenarios, which have passed the first step, are capable to pass the next steps. According to the numbers chosen as threshold, the percentage of scenarios that surpass the threshold are the following:

- Plot 520: 1.1% (The first step starts analyzing 59,940 scenarios, and in the following steps remain 665)
- Plot 533: 0.32% (This experiment starts analyzing 68,328 scenarios, and in the following steps there are only 220)
- Plot 534: 2.9% (The first step of this case starts with 49,348 scenarios, and in the following steps we work with 1436 scenarios)
- Plot 751: 5.72% (This case starts analyzing 72,124 scenarios, and in the following steps we work with 4128 scenarios)
- Plot 752: 3.2% (The first step starts analyzing 68,328 scenarios, and in the following steps there were 2189)

After seeing these numbers, it is interesting to analyze their meaning. We have seen that the prediction level is not better than the level originally calculated. But, in some cases, the fitness reached is not very different from the

fitness computed when the method is applied without improvements (that is to say, without threshold). Then, if necessary, we should define a trade off between time and precision. As we can see, the number of resulting scenarios is substantially smaller than the number present in the first step according to the level of cutting. Therefore, the execution time can be reduced in a extraordinary way.

In average, the execution time can be represented according to the following expression:

$$time = t_1 + (N - 1) * (pr/100) * t_1$$

where, t_1 represents the time necessary to effectuate the simulations for step 1, N is the number of steps, and pr represents the percentage of scenarios that surpass the threshold in step 1 (in other words, the number of active scenarios in step 2 and following ones). According to the values seen in the five experiments, pr would be equal to 2.65, hence, the total time can be considered practically as t_1, because the consumed time by remaining steps is significantly inferior.

7.3 Range reduction

Finally, we test the effect of reducing the range of some wide range parameters on the S^2F^2M behavior. This experimentation arises as an initial step for further improvements of the method where certain degree of parameter knowledge would be take into account. In this case, the reasoning is very easy to understand: if we reduce the possible values of a parameter and limit it within a range near to the ideal value, then, we could obtain a simulation more similar to the real forest fire.

To prove this concept, instead of doing an adjust on the fly, we decide to guide the simulation. For this purpose, certain parameters that initially had a very wide range were chosen to reduce their variation range. The modified parameters were wind speed and wind direction. We used the information shown on table 7.9 to change the parameters file on each plot.

Obviously, after changing the parameters file, the number of scenarios becomes smaller, because we are discarding some values for the affected

parameter, and, therefore, we are reducing the number of combinations. Finally, similarly to the previous section, the simulation time is also reduced as a consequence of decreasing the number of scenarios.

In tables 7.10, 7.11 and 7.12 we can see the fitness values found for each plot.

Plot	Wind Speed Average (Milla/hour)	Wind Direction Average
520	4.608	176.47°
533	12.907	45.00°
534	11.855	56.00°
751	4.697	260.00°
752	3.802	285.00°

Table 7.9: Wind speed and wind direction average for each plot

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.4612	0.5345
6.0	8.0	0.4608	0.7495
8.0	10.0	0.2523	0.4413
10.0	12.0	0.3643	0.7625
12.0	14.0	0.1941	0.4354

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.1849	0.4745
6.0	8.0	0.2862	0.5208
8.0	10.0	0.2065	0.4675
10.0	12.0	0.2163	0.4501

Table 7.10: a) Fitness using reduced ranges for plot 520 b) Fitness using reduced ranges for plot 533

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M
3.0	4.0	-	-
4.0	5.0	0.0948	0.2565
5.0	6.0	0.0837	0.5336
6.0	7.0	0.1376	0.7036
7.0	8.0	0.2589	0.6074
8.0	9.0	0.1914	0.4161

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.7692	0.8819
6.0	8.0	0.7738	0.7917
8.0	10.0	0.6842	0.7073

Table 7.11: a) Fitness using reduced ranges for plot 534 b) Fitness using reduced ranges for plot 751

We can clearly verify that, in four of the five experiments, the fitness values obtained are evidently inferior to those found during the experimentation stage (chapter 5). Only in one case (experiment on plot 752) it is

Initial Time	Final Time	Fitness S^2F^2M	Fitness S^2F^2M*
2.0	4.0	-	-
4.0	6.0	0.6661	0.4976
6.0	8.0	0.5628	0.5257
8.0	10.0	0.5707	0.6451

Table 7.12: Fitness using reduced ranges for plot 752

possible to observe a pair of values that are higher than the original fitness. These correspond to times 4 and 6. On the other hand, we also found other results. An example is the experiment on plot 534. As we can appreciate, the fitness value is clearly low in all steps. The extremely low values are in time 4 (0.0948) and 5 (0.0837). These are unacceptable values for any prediction.

From the obtained results we can conclude that, since S^2F^2M has a inherently statistic core, it is not recommended to modify neither certain method characteristics or the input data, because a good result can be seriously damaged. Although the number of experiments is reduced, they are a good help to see what is the tendency of the behavior. Moreover, these experiments are useful to prove that, at least on S^2F^2M, guiding the simulation (to introduce knowledge, in other words) is not possible. Maybe introducing knowledge is feasible on the other methods, but in S^2F^2M this is a practice that does not report any improvements.

Chapter 8

Conclusions

> "We know very little, and yet it is astonishing that we know so much, and still more astonishing that so little knowledge can give us so much power."
>
> Bertrand Russell

This chapter presents the conclusions and results obtained from our work. The chapter also describes the possible open lines that can be undertaken in the future in order to continue research on forest fire prediction.

8.1 Conclusions and observations

In this work we have treated a very important problem that requires a precise prediction to minimize its effects: Forest fires.

The main purpose of existing works in the field of wildfire simulation is to create operational tools to help forest fire fighting in two ways: fire prevention and fire fighting. In both cases, we need to decide on the best way to act so as to minimize the losses. Therefore, good prediction tools are vital for making good decisions.

However, most of the existing wildland fire models are not able to predict exactly real fire spread behavior. This disagreement between real and simulated propagation arises for different reasons (model imprecision, limitations of numerical solutions, input data uncertainty, etc.)

In this work, we have proposed and developed a statistical method, called S^2F^2M (Statistical System for Forest Fire Management), that improves the

prediction of fire propagation by focusing on the imprecision of the input data. However, in this work, we do not only concentrate our effort on finding the best set of input parameters that provide the best fire prediction, but we offer a prediction by combining the effects of a large number of possible scenarios (set of parameters). Since the proposed method is very time consuming, we have used High Performance Computing as a tool to reduce its execution time.

We developed and coded a first version of our system as a statistical method which would be a DSS (Decision Support System) of low level (level 1). Let us clarify this version only considers homogeneous terrain (respect to vegetation, slope and climatic factors). It is undoubtedly true that working with a more detailed representation is better, because the simulated situation will be more resembling to reality. However, we think that to prove the concepts introduced by this thesis, it is not needful to establish an extremely real environment. Furthermore, reaching such situation carries a large number of design and implementation problems that do not concern the scope of this work.

We showed that S^2F^2M was useful testing with laboratory and field fires. The description, development and results of these tests can be found in:

> Bianchini G., "Sistema de Ayuda a la Decisión para la Gestión de Incendios Forestales". Experimental work for postgraduate program for Computer Architecture and Parallel Processing. Universitat Autònoma de Barcelona (Spain). July 2004.
>
> Bianchini G., Cortés A., Margalef T., Luque E., "S^2F^2M - Statistical System for Forest Fire Management". ICCS 2005 - International Conference on Computational Science, Emory University Atlanta, USA. LNCS 3514, pp. 427-434. 2005.
>
> Bianchini G., Cortés A., Margalef T., Luque E., "S^2F^2M - Sistema Estadístico para la Predicción de Incendios Forestales". CEDI 2005. pp. 623-629.

We have also used S^2F^2M as a tool to evaluate fire propagation danger on the Mediterranean region. The amount of simulations that must be carried

out is enormous and it is necessary to apply high-performance computing techniques to make the methodology feasible. The results show that S^2F^2M is a good tool to create risk maps. A summary of this experimental study can be found in:

> Bianchini G., Cortés A., Margalef T., Luque E., Chuvieco E., Camia A., "Wildland Fire Propagation Danger Maps Based on Factorial Experimentation". Information Technologies in Environmental Engineering (ITEE'2005). pp. 173-185. 2005
>
> Chuvieco E., Camia A., Bianchini G., Margalef T., Koutsias N., Martínez J., "Using remote sensing and GIS for global assessment of fire danger". XXII International Cartographic Conference (ICC2005).
>
> Bianchini G., Cortés A., Margalef T., Luque E., Chuvieco E., Camia A., "Wildland Fire Risk Maps using S2F2M". Journal of Computer Science & Technology (JCS&T) - Special Issue on Selected Papers from CACIC 2005. Vol. 5 No. 4. pp. 244-249. 2005.

We have compared two optimization techniques (GLUE and Evolutionary methods) whose goal consist on searching a unique set of input parameters to perform fire spread prediction, against our method. The obtained results allow us conclude that S^2F^2M is a powerful tool for solving this kind of problem, which can be considered a Grand Challenge Problem.

On the one hand, another important development of this work was GLUE implementation to compare it against our system. At the moment of considering GLUE as an interesting concept for contrasting our methodology, it had been implemented on a demonstrative software by means of Java Applet as a simple 1-D model [56]. Therefore, we decided to re-implement it in C language in a functional 2-D model.

During the experimentation stage, we discovered that a part of the GLUE methodology did not operate correctly. Finally, we opted for applying GLUE method without making use of the concept of likelihood, which we commented in Chapter 7.

On the other hand, we want to highlight that the Evolutionary Method has been developed within our team. It has been proven in previous works and it has also appeared in several papers [1, 2, 3, 4].

It is important to remark that we have compared the three methods using a set of five real cases. The first relevant results from these experiment, in addition to a description of the methodology, have been published in:

> Bianchini G., Cortés A., Margalef T., Luque E., "Improved prediction methods for Wildfires using High Performance Computing: A comparison". ICCS 2006 - International Conference on Computational Science, University of Reading, UK. LNCS 3991, pp. 539-546. 2006.
>
> Bianchini G., Denham M., Cortés A., Margalef T., Luque E., "Improving forest-fire prediction by applying a statistical approach". V International Conference on Forest Fire Research, Figueira da Foz, Coimbra, Portugal. *Paper in press.*

An important advantage of S^2F^2M in relation to the other methods is that, in the case of acting as a preventive method, it does not need feedback from real forest fire. As we explain above, it can work as a statistical method, and therefore its result is not a prediction of a particular forest fire, but it gives probability of ignition for certain terrains (for example, the case of risk maps). On the other hand, GLUE and Evolutionary method require this information to work.

Finally, comparing the three methods, we found that the disadvantages of GLUE and Evolutionary methods are that the set of parameters selected as best predictor can disagree with the real set of parameters —and generally it is thus—. However, this can never occur using S^2F^2M because it has a different basis, and precisely, it uses all combinations of parameters to compute the prediction.

8.2 Open lines

During the development of this thesis, we discovered a lot of topics that can be the goal of future work. Some of these open lines have a direct connection

with our work, and others are related but belong to other areas.

In the following paragraphs we highlight some possible lines to continue the work. The potential lines are presented in order according to its complexity level.

An immediate work to do, will be to make an analysis focused in the tradeoff time-precision. As we explained, the terrain is represented by a matrix. Then, the question is: What is it the ideal cell size? According to the cell size, the matrix has more or minus cells, and therefore the simulation time changes in a drastic way. If we are capable of calculating this relation, we would save valuable time and resources.

Currently, we consider the terrain as a matrix with homogeneous cells; that is to say, within each scenario, the characteristics of each cell are the same (vegetation, moisture, meteorological conditions, etc.). An improvement in the current model, would be to define heterogeneous terrains, where variables as slope, fuel or moisture can be different along the land. This change could offer a more realistic simulation.

A more ambitious goal is to add the wind field model to the simulator. Although speed and wind direction are two parameters that also vary within all the possible combinations of scenarios, the problem is that we are considering the wind like a homogeneous parameter in all the terrain. A interesting option would be to simultaneously consider a behavior wind model and make it vary with the rest of the parameters.

There are other aspects that deserve consideration, but they have strict relation with experimentation. In order to be able to validate these methods correctly and to carry out the pertinent comparisons, it is necessary to have abundant information of real cases. It is feasible to find information of forest fire, but generally this information is fragmented, and therefore not useful for our intention. Therefore, as an immediate continuation of this work, it would be advisable to collect data to continue experimenting and testing the system.

The study presented in this work deals with the analysis of parameter optimization and prediction on a postmortem system. However, it would be desirable to operate with the methods under the real time constraints. In order to do this, we needed to optimize the methods (as we did in a first

intent in chapter 6). But not only the method, it is also necessary to optimize the implementation to improve the data structure and communication.

Related with the last topic, we discover another big challenge: obtaining all necessary information in real time (terrain, vegetation, meteorological conditions, etc.). This is an important work that needs contributions from different fields, using different technologies. Furthermore, a necessary thing is to establish the mechanisms to integrate in a unique system such contributions.

It is necessary to tackle the problem of forest fires from many different ways. We trust our work is an important contribution from Computational Science to palliate this problem. We hope investigation continues on this area, then we will be able to see in an immediate future the results of such advances in practice in real systems.

Bibliography

[1] Abdalhaq B., Cortés A., Margalef T., Luque E., "Evolutionary Optimization Techniques on Computational Grids". International Conference on Computational Science. LNCS 2329. pp. 513-522. 2002.

[2] Abdalhaq B., Bianchini G., Cortés A., Margalef T., Luque E., "Improving Wildland Fire Prediction on MPI Clusters". PVM/MPI. pp. 520-528. 2003.

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

[4] Abdalhaq B., Bianchini G., Cortés A., Margalef T., Luque E., "Between Classical and Ideal: Enhancing Wild-land Fire Prediction Using Cluster Computing". Journal of Cluster Computing Special Issue on cluster computing in science and engineering. Vol 8, No. 4. 2004

[5] Albini F. A. "Estimating wildfire behavior and effects". Tech. Rep. INT-30. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station; pp. 91. 1976.

[6] Albini F. A. "A model for fire spread in wildland fuels by radiation". Combustion Science and Technology 42. pp. 229-258. 1985.

[7] Anderson H. E. "Predicting wind-driven wildland fire size and shape". Research Paper INT-305. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station. pp. 26. 1983.

[8] André J. C. S., "A theory on the propagation of surface forest fire fronts" PhD Dissertation (in portuguese), Universidade de Coimbra, Portugal. 1996.

[9] Andrews P. L. "BEHAVE: Fire Behavior prediction and modeling systems - Burn subsystem, part 1". General Technical Report INT-194. Odgen, UT, US Department of Agriculture, Forest Service, Intermountain Research Station. pp. 130. 1986.

[10] Andrews P. L., Bevins C. D., Seli R. C. "BehavePlus fire modeling system, version 2.0: User's Guide". Gen. Tech. Rep. RMRS-GTR-106WWW. Ogden, UT: Department of Agriculture, Forest Service, Rocky Mountain Research Station. pp. 132. 2003.

[11] Answers.com: Information from Grand Challenge Problem
http://www.answer.com
Accessed on May 2006.

[12] Beer T. "Bushfire rate of spread forecasting: deterministic and statistical approaches to fire modelling". Journal of Forecasting 10. pp. 301-307. 1991.

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

[14] Beven K. "Towards a coherent philosophy for environmental modelling". Proc. Roy. Soc. Lond. A 458, in press.

[15] Bevins C. D. , "FireLib User Manual & Technical Reference", 1996.
http://www.fire.org
Accessed on January 2004.

[16] Bianchini G., "Sistema de Ayuda a la Decisión para la Gestión de Incendios Forestales". Experimental work for postgraduate program for Computer Architecture and Parallel Processing. Universitat Autònoma de Barcelona (Spain). July 2004.

[17] Bianchini G., Cortés A., Margalef T., Luque E., "S^2F^2M - Statistical System for Forest Fire Management". LNCS 3514, pp. 427-434. 2005.

[18] Bianchini G., Cortés A., Margalef T., Luque E., Chuvieco E., Camia A., "Wildland Fire Propagation Danger Maps Based on Factorial Experimentation". Information Technologies in Environmental Engineering (ITEE'2005). Shaker Verlag. pp. 173-185. 2005.

[19] Bianchini G., Cortés A., Margalef T., Luque E., "Improved prediction methods for Wildfires using High Performance Computing: A comparison". ICCS 2006 - International Conference on Computational Science, University of Reading, UK. LNCS 3991, pp. 539-546. 2006.

[20] Bianchini G., Cortés A., Margalef T., Luque E., Chuvieco E., Camia A., "Wildland Fire Risk Maps using S^2F^2M". Journal of Computer Science & Technology (JCS&T) - Special Issue on Selected Papers from CACIC 2005. Vol. 5 No. 4. pp. 244-249. 2005.

[21] Binley A. M., Beven K. J., Calver A., Watts L. G., "Changing responses in hydrology: assessing the uncertainty in physically based model predictions". Water Resources Research, Vol. 27. pp. 1253-1261. 1991.

[22] Buyya R., "High Performance Cluster Computing: Programming and Applications" (v.2) Prentice Hall, 1999.

[23] Calogine D., Seró-Guillaume O., "Air flow model in a tree crown", Forest fire Research & Wildland Fire Safety. On CD-ROM, Millpress. 2002.

[24] Calver A., "Calibration, sensitivity and validation of a physically-based rainfall-runoff model". J. Hydrology, 103. pp. 103-115. 1998.

[25] Catchpole E. A., Catchpole W. R., "Modelling moisture damping for fire spread in a mixture of live and dead fuels". International Journal of Wildland Fire 1. pp. 101-106. 1991.

[26] Cheney N. P., Gould J. S., Catchpole W. R., "The influence of fuel, weather, and fire shape variables on fire spread in grasslands". International Journal of Wildland Fire 3. pp. 31-44. 1993.

[27] Chuvieco E., Camia A., Bianchini G., Margalef T., Koutsias N., Martínez J., "Using remote sensing and GIS for global assessment of fire danger". XXII International Cartographic Conference (ICC2005). 2005.

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

[29] ElPais.es International, "Una ola de incendios sacude el norte y centro de Portugal". 5 de Agosto de 2005.
http://www.elpais.es/articulo/20050805elpepuint_11/Tes/elpporint/
Accessed on December 2005.

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

[31] FIRESTATION: ADAI Products
http://www.adai.pt/products/firestation/
Accessed on June 2004.

[32] FIRE.ORG - Public Domain Software for the Wildland fire Community
http://www.fire.org
Accessed on May 2006.

[33] Frandsend W. H. "Fire spread through porous fuels through the conservation of energy". Combustion and Flame 16. pp. 9-16. 1971.

[34] GESTOSA: ADAI - CEIF (Center of Forest Fire Studies)
http://www.adai.pt/ceif/Gestosa/
Accessed on January 2005.

[35] Gillon D., Valette J. C., Moro C., "Foliage moisture content and spectral characteristics using near infrared reflectance spectroscopy (NIRS) ". Forest fire Research & Wildland Fire Safety. On CD-ROM, Millpress. 2002.

[36] Gnuplot homepage, http://www.gnuplot.info/index.html
Accessed on May 2006.

[37] Grama A., Gupta A., Karypis G., Kumar V., "Introduction to Parallel Computing. Second Edition." Pearson Addison Wesley. 2003.

[38] Greenpeace España, "Incendios". 23 de Agosto de 2004.
http://www.greenpeace.org/espana/campaigns/bosques/incendios
Accessed on January 2006.

[39] Green D. G., Gill A. M., Noble I. R. "Shapes of similar fires in discrete fuels". Ecological Modelling 20. pp. 21-32. 1983

[40] Groop W. D, Lusk E., "User's Guide for mpich, a Portable Implementation of MPI", Mathematics and Computer Science Division, Argonne National Laboratory, 1996.

[41] Groop W. D, Lusk E., Doss N., Skjellum A., "A high-performance, portable implementation of the MPI message passing interface standard", Parallel Computing, volume 22-6, pp.789-828, September 1996.

[42] Jorba J., Margalef T., Luque E., J. Campos da Silva, Viegas D. X., "Parallel Approach to the Simulation of Forest Fire Propagation". Proc. 13 International Symposium "Informatik fur den Umweltshutz" der Gesellshaft Fur Informatik (GI). pp. 68-81. 1999.

[43] Kernighan B. W., Ritchie D., "El Lenguaje de Programación C", Prentice Hall, 1991.

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

[45] McAlpine R. S., "Testing the effect of fuel consumption on fire spread rate". International Journal of Wildland Fire 5. pp. 143-152. 1995.

[46] McArthur A. G., "Weather and grassland fire behavior". Australian Forestry and Timber Bureau Leaflet 100, Camberra, AFTB. 1966.

[47] Montgomery D. C., Runger G. C., "Probabilidad y Estadística aplicada a la Ingeniería", Limusa Wiley, 2002.

[48] Morgan P., Hardy C., Swetnam T.W., Rollins M. G., Long D. G. Mapping fire regimes across time and space: Understanding coarse and fine-scale fire patterns. International Journal of Wildland Fire, Vol. 10. pp. 329-342. 2001.

[49] Morrison R. S., "Cluster Computing: Architectures, Operating Systems, Parallel Processing and Programming Languages", 2002.
http://www.ace.ual.es/~jaberme/docsppal/cluster/
Accessed on May 2006.

[50] Morvan D., Tauleigne V., Dupuy J. L., "Wind effects on wildfire propagation through a Mediterranean shrub". Forest fire Research & Wildland Fire Safety. On CD-ROM, Millpress. 2002.

[51] MPI: The Message Passing Interface Standard.
http://www-unix.mcs.anl.gov/mpi/
Accessed on May 2006.

[52] Ortigosa P. M., García I., Jelasity M., "Reliability and Performance of UEGO, clustering-based global optimizer", Journal of Global Optimization, 19(3). pp. 265-289. 2001.

[53] Parque Científico de Madrid, http://www.fpcm.es/amT3_2004.htm
Accessed on April 2006.

[54] Perry G. L. W., "Current approaches to modelling the spread of wildland fire: a review". Progress in Physical Geography 22. pp. 22-245. 1998.

[55] Pincus M., "A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems", Operations Research, 18. pp. 1225-1228. 1970.

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

[57] Quinn M. J., "Parallel Programming in C with MPI and Open Mp". First Edition. McGraw-Hill, 2004.

[58] Riaño D., Meier E., Allgöwer B., Chuvieco E., "Generation of vegetation height, vegetation cover and crown bulk density from airborne laser scanning data". Forest fire Research & Wildland Fire Safety. On CD-ROM, Millpress. 2002.

[59] Ríos Insua S., Bielza Loyola C., Mateos Caballero A., "Fundamento de los Sistemas de Ayuda a la decisión". RaMa, 2002.

[60] Reinhardt E. D., Keane R. E., Brown J. K., "First Order Fire Effects Model: FOFEM 4.0, User's Guide". General Technical Report INT-GTR- 344. 1997.

[61] Rothermel R. C., "A mathematical model for predicting fire spread in wildland fuels", USDA FS, Ogden TU, Res. Pap. INT-115, 1972.

[62] Ruiz A. D., Maseda C. M., Lourido C., "Possibilities of dead fine fuels moisture prediction in Pinus pinaster Ait. stands as 'Cordal de Ferreiros' (Lugo, North-western of Spain)". Forest Fire Research & Wildland Fire Safety. On CD-ROM, Millpress. 2002.

[63] Salvador R., Piñol J., Tarantola S., Pla E., "Global sensitivity analysis and scale effects of a fire propagation model used over Mediterranean shrublands". Ecological Modelling 136. pp. 175-189. 2001.

[64] SPREAD Project, Forest Fire Spread Prevention and Mitigation
http://www.adai.pt/spread/
Accessed on May 2004.

[65] Stocks B. J., Lawson B. D., Alexander M. E., van Wagner C. E., McAlpine R. S., Lynham T. J., Dubé D. E., "The Canadian forest fire danger rating system: and overview". Forestry Chronicle 65. pp. 450-457. 1989.

[66] SUN Microsystems, "Noticias SUN España, SUN Microsystems", 8 de Julio de 2004
http://es.sun.com/infospain/noticias/2004/julio/040708.html
Accessed on April 2006.

[67] Tapia Richard A., Lanius C., Mc Zeal C. M., Parks Teresa A., "Computational Science: Tools for a Changing World". Rice University.
http://ceee.rice.edu/Books/CS/index.html
Accessed on May 2006.

[68] Top 500 Supercomputer Sites
http://www.top500.org/
Accessed on November 2005.

[69] Viegas D. X., Ribeiro L. M., Matos L., Palheiro P., Pita L. P., Alfonso C., "Slope and wind effects on fire spread". Forest Fire Research & Wildland Fire Safety. On CD-ROM, Millpress. 2002.

[70] Weber R. O., "Thermal theory for determining the burning velocity of a laminar flame using the inflection point in the temperature profile". Combustion Science Technology 64. pp. 135-139. 1989.

[71] Wilkinson B., Allen M., "Parallel Programming - Techniques and Applications Using Networked Workstations and Parallel Computers. Second Edition". Pearson Prentice Hall. 2005.

[72] Wilson R.A., "Reexamination of Rothermel's fire spread equations in no-wind and no-slope conditions" Jr. 1990. Res. Pap. INT-434. Ogden, UT: U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station; pp. 13. 1990.

[73] Xanthopoulos G., Manasi M., "A practical methodology for the development of shrub fuel models for fire behavior prediction". Forest Fire Research & Wildland Fire Safety. On CD-ROM, Millpress. 2002.

[74] 20Minutos.es, "La sequía recrudece ...", 18 de Julio de 2005
http://www.20minutos.es/noticia/39307/0/SEQUIA/EUROPA/
Accessed on April 2006.


¡SE ME APROBATARON LOS CHORI!