Uncertainty Reduction Method Based on Distributed Computing Applied to Forest Fire Prediction*

Germán Bianchini[1], Ana Cortés[2], Tomàs Margalef[2] and Emilio Luque[2]

[1] Departmento de Ingeniería en Sistemas de Información, Universidad Tecnológica Nacional Facultad Regional Mendoza, M5502AJE (Mendoza) Argentina
[2] Departament d'Arquitectura de Computadors i Sistemes Operatius, Universitat Autònoma de Barcelona, 08193-Bellaterra (Barcelona) Espanya

Abstract. In general, the purpose of the Data-Driven Prediction methods is focused on finding the vector of parameters which better describes the real situation under consideration on a certain model. Therefore, it is expected that the same vector of parameters could be used to describe the immediate future. However, for those parameters that present a dynamic behavior (e.g. direction and speed wind in the case of forest fire models), the found values cannot be adequate for doing the prediction. We propose an alternative method developed in a new branch of Data-Driven Prediction which we called Multiple Overlapping Solution. This method combines Statistical concepts and Distributed Computing to obtain a high quality prediction. Each parameter is represented by a range of values with a particular cardinality for each one of them. A huge number of scenarios are generated and the forest fire propagation for each scenario is evaluated and considered. Then, all results are statistically aggregated to determine the burning probability of each area, which will be compared with the real state to adjust the method. This aggregation is used to predict the burned area in the next step using a pattern matching method, which provides the tendencies behavior for the next step.

1 Introduction

Fire Management includes different areas (prevention, detecting and monitoring, forest fire suppression, etc.) that could be enriched by the forest fire prediction since it provides an important and powerful tool which would improve the current methods and procedures. Numerous 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 [3, 4, 8, 9]. Such models require a set of input parameters (vector of parameters), including vegetation type, moisture contents, wind speed and direction and so on, and provide the evolution of the

* This work has been supported by the MEyC-Spain under contract TIN 2004-03388 and by the European Commission under contract EVG1-CT-2001-00043 SPREAD.

fire line in simulation steps. 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.

This paper concentrates on the idea that it is hardly probable to find an input parameters vector to be applied to the propagation model because, as has been observed, it is practically impossible to know the value of each parameter when a fire starts. Therefore, we offer an alternative for determining the fire behavior: a statistical method which uses the concept of multiple overlapping solution as core to find the pattern of fire behavior. The method takes into account thousands of possible situations rather than only one combination. Furthermore, in this work we describe two more data-driven methods that face the same problem but based on different principles: BBOF (Black-Box Optimization Framework), based on diverse algorithms (genetic algorithms, simulated annealing, taboo search) [1, 2] and GLUE (Generalized Likelihood Uncertainty Estimation), based on uncertainty prediction [5, 15].

The remaining of this paper is organized as follows: In Sect. 2, the main features of Data Driven methods are reported. BBOF and GLUE are described in the same section. In Sect. 3 we describe our method. A general description of the implementation of the methods is summarized in Sect. 4. Section 5 is dedicated to depict the experiments. We compare the results obtained after applying the three methods on five different real forest fires and, finally, Sect. 6 presents the obtained conclusions.

2 Forest Fire Methods Classification

Basically, there are two ways of applying a prediction method. The first, know as Classical Prediction, consists of using any existing fire simulator behavior to evaluate the position of the fire after a certain initial period of time. It is necessary to feed the simulator with all the required parameters (weather data, vegetation, terrain description, etc.). Next, the simulator is put into operation to predict the fire line after a certain time interval.

Classical Prediction shows certain limitations. The application of Classical Prediction commonly generates a certain degree of error. These errors not only come from the problems of the model (errors due to a 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 it also come from the implementation of the simulator. There are certain parameters that cannot be measured directly and, therefore, they must be estimated from indirect measures (for instance, wind speed and direction should be interpolated from several distant points).

The second option is composed by the Data-Driven Prediction Methods. Under this name we have grouped those methods that, looking for a solution to the problem manifested by classical methods, make use of optimization techniques in order to calibrate the input parameters vector. The optimization process objective is to find an ideal vector of parameters. If this vector feeds the simulator, the previous behavior would be described in the best form (i.e. the behavior that has been used to calibrate the set of parameters). Therefore, we would assume that the same vector of parameters could be used to describe the best possible form for the immediate future.

A diagram of Data-Driven Methods can be observed in Fig. 1. We can appreciate the main differences between Data-Driven Method and Classical method. In first place, Data-Driven Prediction Methods require a large ensemble of input parameters (different combinations that produce different scenarios). At the same time, this characteristic is related to the second difference: Data-Driven Prediction Methods need extra time to compute all these data. Finally, a very important feature is that Data-Driven Prediction Methods apply some kind of calibration or optimization (CS box) to find the most suitable vector of parameters.

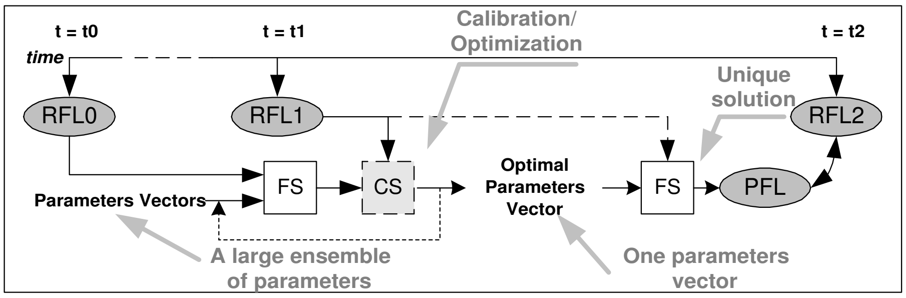

Fig. 1. Diagram of Data Driven Prediction Method of forest fire propagation (FS: Fire Simulator, CS: Calibration Stage, PFL: Predicted Fire Line, RFLX: Real Fire Line on time X)

Both kinds of methods have a similarity: they use only one vector of parameters to do the prediction. Therefore, we can say that Classical prediction methods and Data-Driven methods give only one solution. Within these methods, we will analyze two cases: BBOF and GLUE.

2.1 Black-Box Optimization Framework

The BBOF method is a framework [1]. It works in an iterative fashion, moving step-by-step from an initial set of guesses to a final value which is expected to be closer to the true (optimal vector of parameters) than the initial guesses. This approach also focuses on overcoming the input-parameter uncertainty problem by introducing the idea of applying an optimization scheme to calibrate the set of input parameters with the aim of finding an optimal set of input, thus improving the results provided by the fire-spread simulator.

A pre-set optimization technique is applied to generate a new set of guesses in each iteration, which should be better than the previous ones. This technique offers the possibility of applying diverse ideas: biological evolution, simulated annealing and taboo search. These algorithms work on populations of individuals instead of single solutions, which allows performing the search in a parallel approach.

In Fig. 2 is shown a graphical schema of the BBOF method.

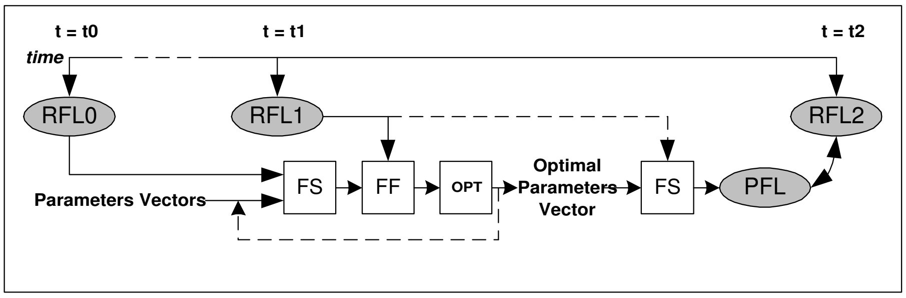

Fig. 2. Diagram of BBOF method (FS: Fire Simulator, CS: Calibration stage, FF: Fitness Function, OPT: Optimization stage, PFL: Predicted Fire Line, RFLX: Real Fire Line on time X)

The optimization method is associated with the specification of a mathematical objective function (called F) and a vector of parameters that should be tuned to optimize the objective function. This vector of parameters is referred to as Θ. Therefore, we can formulate an optimization problem as follows:

$$\text{Find } \Theta^* \text{ that optimizes } \substack{F(\Theta)\\ \Theta \in S} \quad . \tag{1}$$

where F represents the objective function. The optimization problem deals with the aim of defining a process to find a setting for the vector of parameters Θ (where Θ^* is a particular setting of Θ), which provides the best value for the objective function F. 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.

In Fig. 2 the vector Θ corresponds to the vectors of parameters to be optimized. The length of this vector will depend on the underlying simulator, but in general, given that the majority of fire simulators are based on the Rothermel model [17], in average there are ten o twelve components.

The fire simulator (FS box) and the fitness function (FF box) conform the objective function F. The OPT box will include the optimization strategy selected to solve the problem(evolutionary algorithms, simulated annealing, etc.). The goal of the optimization strategy consists in generating a new vector of parameters (Θ^*), which minimizes the underlying error prediction, taking into account the information provided by F. The optimization process is performed in an iterative way. This feedback loop will be repeated until either a 'good' solution is found or a predetermined number of iterations has been reached, 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 ($t2$ in Fig. 2). This process is repeated each time a new fire line feeds the process in order to readjust the prediction.

2.2 Generalized Likelihood Uncertainty Estimation

The GLUE method of Beven and Binley [5] is a Monte Carlo simulation based approach [14, 16] to model conditioning and uncertainty estimation. It is a framework for estimating predictive uncertainty of complex environmental models. Originally, this method was developed for being used with hydrological models, but it has subsequently been applied to a wide range of environmental systems [15].

GLUE rejects the idea that there is only one optimum vector of parameters in a model calibration. It considers that there are multiple vectors of parameters 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 parameters vector 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 variable predicted 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:

$$P(\hat{Z}_t < z) = \sum_{i=1}^{N} L\left[M(\Theta_i)|\hat{Z}_{t,i} < z\right] \quad . \tag{2}$$

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

$$L_p(M(\Theta_i)) = \frac{[L_0(M(\Theta_i) + L(M(\Theta_i)|Y)]^p}{C} \quad . \tag{3}$$

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

It was not possible to use the classical Bayes equation approach of multiplying these likelihoods (because when one vector of parameters would get a zero likelihood it would be zero forever), but the idea behind the procedure is exactly the same.

GLUE considers a high number of possible vectors of parameters which are stored in the PS box (Parameters Sets) where the likelihood of vectors of parameters are updated. To reduce the divergence between classical prediction and real-fire propagation, it is necessary 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 determine the degree of matching between the predicted fire line and the real fire line. In Fig. 3 are shown this features in a graphical schema.

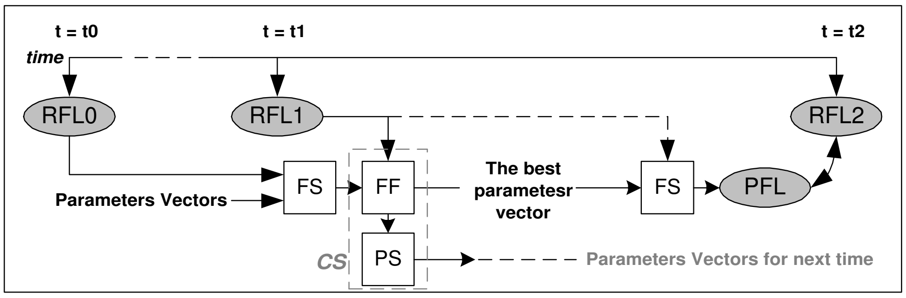

Fig. 3. Diagram of GLUE method (FS: Fire Simulator, CS: Calibration stage, FF: Fitness Function, PS: Parameters Sets, PFL: Predicted Fire Line, RFLX: Real Fire Line on time X)

3 Multiple Overlapping Solution

In this work we evaluate a new method which we called Statistical System for Forest Fire Management (S^2F^2M) [7]. Such a method is placed in a new branch of Data-Driven methods with Multiple Overlapping Solution. Although S^2F^2M belongs to the Data Driven type, it generates a prediction based on the totality of the proposed cases, rather than based on a single case, such as BBOF and GLUE.

We define the concept of scenario as a particular setting of the set of parameters. 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 a unique solution, it does not make distinction between good and bad cases. The interesting fact of this methodology is that every possible scenario contributes with its particular characteristics to find a better prediction to describe the behavior pattern. Finally, we also present a comparison among the two methods of Data-Driven prediction mentioned above (BBOF and GLUE) and our method, with the objective of showing the effectiveness of our proposal of Data Driven prediction with Multiple Overlapping Solution. This comparison will be done on four cases of real forest fires.

3.1 Statistical System for Forest Fire Management

The methodology of S^{2}F^{2}M 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 with 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. The 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 [11] (for example, wind conditions, moisture content and vegetation parameters). A scenario represents each particular situation that results from a set of values.

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

$$[\mathit{Inferior\_threshold\_i},\ \mathit{Superior\_threshold\_i}],\ \mathit{Increment\_i}\ . \tag{4}$$

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. The Parameter Domain Cardinality is calculated as follows:

$$C_i = \frac{((Superior\_threshold\_i - Inferior\_threshold\_i) + Increment\_i)}{Increment\_i}\ . \tag{5}$$

Finally, in (6) we show how by considering the cardinality of each parameter it is possible to calculate the total number of scenarios obtained from variations of all possible combinations.

$$\#Scenarios = \prod_{i=1}^{p} C_i\ . \tag{6}$$

where p is the number of parameters.

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 (P_{ign}) as:

$$P_{ign}(A) = n_A/n \ . \tag{7}$$

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 generates this probability in each cell is necessarily the same. Consequently, it is not possible to know the set of scenarios that generate a particular probability map.

Once we have obtained the output matrix, which includes all the probability maps, the next step consists in comparing the real fire against this matrix. The objective of such a comparison is to search for a particular value of P_{ign} whose associated probability map provides the best matching with the real fire propagation. In other words, we are interested in finding what we refer to as a Key Ignition number (K_{ign}). Therefore, the associated map of probability has to accomplish the condition expressed in (8).

$$\{x : P_{ign}(x) \geq K_{ign}/n \ \mid K_{ign} \in \mathbb{N}\} \ . \tag{8}$$

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.

A graphical schema of the method is presented in Fig. 4. The process of prediction needs a calibration stage just at the beginning (time period that goes from $t0$ to $t1$ in Fig. 4) 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 Fig. 4 for time $t2$. We can see that the output generated by SS box (Statistical System) is used for a double purpose. On the 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 constitutes the input of the 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 along the execution as the system is fed with new information about the fire situation.

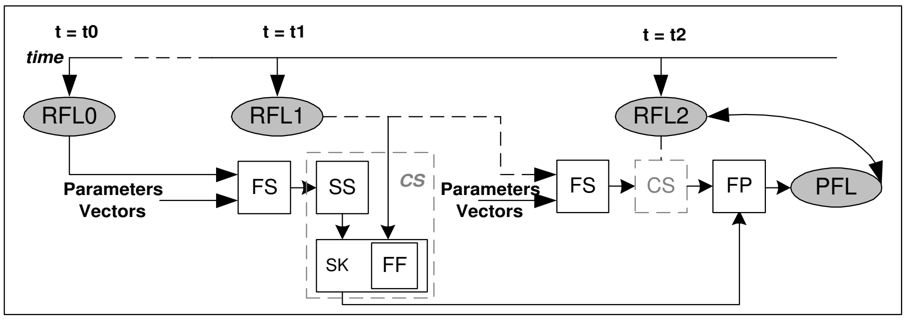

Fig. 4. Diagram of S^2F^2M method (FS: Fire Simulator, SS: Statistical System, SK: Search K_{ign}, FF: Fitness Function, CS: Calibration stage, FP: Fire Prediction, PFL: Predicted Fire Line, RFLX: Real Fire Line on time X)

4 Implementation Features

Although heuristic optimization techniques may reduce the time of search, we still could make the execution time faster by applying distributed computing concepts. As we explained in the previous sections, these kinds of techniques are very time consuming for different reasons: the complexity of the involved models, the huge number of scenarios considered, the real time restrictions, etc.

Distributed Systems have many advantages. We can mention some examples: high performance, high throughput, expandability and scalability, economies of scale, technology, between others. Furthermore, some applications are inherently distributed in nature, such as geographically diverse operations or suit multiple processors working in parallel such as inherently parallel problems. An example of this are the data driven methods mentioned in this work. Therefore, such methods have been parallelized in order to reduce the involved execution time.

The three methods described above have been implemented in operational systems that incorporate the same simulation kernel and apply a methodology to evaluate the fitness function. These systems have been developed on a PC Linux cluster using MPI [18, 13] as the message passing library.

In the following subsections, we comment the elements in common between the three systems, and we describe how we took advantage of parallel and distributed systems.

4.1 Simulation Kernel

BBOF, GLUE and S^2F^2M use as a simulation kernel a forest fire simulator (fireSim) based on the fireLib library [6].

fireLib is a C function library for predicting the spread rate and intensity of forest fires. It is derived directly from the BEHAVE fire behavior algorithms [3] for predicting fire spread in two dimensions, but is optimized for highly iterative applications such as cell-or wave-based fire growth simulation. In particular, this simulator uses a cell automata approach to evaluate fire spread. The terrain is divided into square cells and a neighborhood relationship is used to evaluate whether a cell will be burnt and to estimate the instant in which the cell will be reached by the fire.

As inputs, this simulator accepts maps of the terrain, vegetation characteristics, wind and the initial ignition map, and, as output, the simulator generates a map of the terrain in which each cell is labeled with its ignition time.

4.2 The Fitness Function

To evaluate and compare the systems' responses, we defined a fitness function. Since the three systems use an approach based on cells, the fitness function was specified as follows:

$$\text{Fitness} = \frac{(\#cells \bigcap - \#IgnitionCells)}{(\#cells \bigcup - \#IgnitionCells)} \ . \tag{9}$$

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 $\#IgnitionCells$ represent the number of burned cells before starting the simulation.

A fitness value equal to 1 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 the simulation did not coincide with reality at all.

4.3 Parallelism on the Methods

The simplest step from sequential methods into the parallel version is to recognize that many inner loops are only used to specify that the same operations are to be performed repeatedly on a set of disjoint values. These can be formulated as vector operations [10] where the same operation is performed on each element of a vector. This single vector operation is equivalent to writing a sequential loop that adds the corresponding elements.

The same operation can be carried out on multiple data items, that is, a vector operation, by a single control unit and replicated complete arithmetic units. It can also be done by organizing the parallel hardware into a SIMD scheme [12]. This way of work is called data parallelism and it has had a big impact on high-speed scientific computation.

In our case, every method has to make the same calculations a lot of times because they use a sequential simulator in a loop, giving as a result a very time consuming methodology.

Using multiple computational resources working in parallel to obtain the desired efficiency is a good solution. We choose Master-Worker architecture because it is suitable to achieve this aim. A main process (Master process) can calculate each combination of parameters and send them to a set of Workers. These

Workers carry out the simulation and return the partial result to the Master. This resulting map indicates which cells are burned and which are not. These explicit transfers of data are commonly done by the message passing operation.

In distributed computing, the interprocessor communication is made by I/O operations. As a consequence, one way of providing high-level support for distributed memory programming is to provide a communication subroutine package for an existing sequential language. In this case, the communication library MPI (Message Passing Interface) is a good option [18, 16].

MPI is defined as an extension to Fortran, C, and C++. We chose it because the simulator used as core (fireSim) has been developed in C, and furthermore, all methods presented and commented in this work have been developed in C and C++ also.

5 Comparative Experiments

An interesting and effective way of compare the three methods is to apply them on a set of real experiments in the field. These burns took place in Serra da Lousã (Gestosa, Portugal), at an altitude varying from 800 and 950 m above sea level. The set of burns were part of the SPREAD project [19]. In the Gestosa field experiments, the terrain was divided into dedicated plots in order to carry out different sorts of tests and measurements. We worked with four plots, which had the following characteristics:

1. *Experiment 1*: the plot was represented by means of a grid of 89 columns x 91 rows and the slope was 18°.
2. *Experiment 2*: the plot was represented by means of a grid of 75 columns x 126 rows and the slope was 21°.
3. *Experiment 3*: the plot was represented by means of a grid of 20 columns x 30 rows and the slope was 6°.
4. *Experiment 4*: the plot was represented by means of a grid of 20 columns x 30 rows and the slope was 6°.

In experiments 1 and 2 the cell size was 1 m^2, and in experiments 3 and 4 the cell size was 0.333 m^2. The remaining parameters such as wind conditions and moisture content were variable.

In order to gather as much information as possible about the fire-spread behavior, a camera recorded the complete evolution of the fires. The videos obtained were analyzed and several images were extracted every certain period of time. From the images, the corresponding fire contours were obtained and converted to cell format in order for methods to interpret them.

The results presented in this work, were obtained by executing the three systems on a cluster computer (32 processors) of homogenous PENTIUM IV (3.0 GHz, Fedora Core 4, Ethernet card Broadcom NetXreme Gigabit).

5.1 Experiment 1

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 [4].

After the application of each method, we obtained the fitness values shown on Table 1. The second row of the table means that if we are at time 4 and we make a prediction for minute 6, we will get a fitness equal to 0.5345 for S^2F^2M, 0.4188 for GLUE and 0.4513 for BBOF method.

Table 1. Comparative of found fitness in each method for experiment 1 (time is expressed in minutes)

Initial	Final		Fitness	
Time	Time	S^2F^2M	GLUE	BBOF
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

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. However, those values are still above GLUE fitness and above the Evolutionary case.

5.2 Experiment 2

This experiment is very different to the previous one. In this case the file of ranges exhibits some difference with the previous experiment because this one presents other characteristics: the wind speed is different and also the slope. Furthermore, this is an experiment started with only one ignition point (because of direction and speed of wind, fire grows in an elliptical way).

The minute 3 was chosen as the initial time $t0$. Since the step was defined to one minute, the first adjustment was made on minute 4, and, therefore, the first prediction was carried out on minute 5.

After the application of each method, we obtained the fitness values shown on the Table 2.

5.3 Experiment 3

This is a 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

Table 2. Comparative of found fitness in each method for experiment 2 (time is expressed in minutes)

Initial	Final		Fitness	
Time	Time	S^2F^2M	GLUE	BBOF
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

obtain a better terrain definition that allows us to study the spread fire in more detail. It is a case of linear ignition, started with pyrotechnic devices.

The duration of the burn in the experiment was 10 minutes. In this case, we applied the methods four times at time instances 4, 6, 8 and minute 10. 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. The results are summarized in Table 3. Once again we can easily conclude that S^2F^2M method provides better results than the GLUE scheme and BBOF method.

Table 3. Comparative of found fitness in each method for experiment 3 (time is expressed in minutes)

Initial	Final		Fitness	
Time	Time	S^2F^2M	GLUE	BBOF
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

5.4 Experiment 4

The last experiment, similar to the previous one, has a reduced plot size. For this reason, once again we defined a small cell size to maximize the number of cells in the grid. 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.

Table 4 lists the resultant fitness values after applying the methods to predict the fire behavior on this experiment. We can see that fitness values are very similar between the three methods. However, in general the statistical ap-

Table 4. Comparative of found fitness in each method for experiment 4 (time is expressed in minutes

Initial	Final		Fitness	
Time	Time	S^2F^2M	GLUE	BBOF
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

proach obtains better results. Also, it is positive that statistical predictions are increasing on each prediction step.

6 Conclusions

In this study, we have compared three methods which focus in the input parameter uncertainty reduction. The compared methods are included in a category of application that takes advantage of distributed computing. In particular, we are talking about a kind of distributed computing commonly used to obtain high performance computing: cluster computing.

The parallel version of the methods allowed us to reduce considerably the execution time (for instance, we reached a speed-up of 27.5 using 32 processors with S^2F^2M). Parallel applications and architectures are, thus, the natural way toward computational speed that can accompany the progress in the technology.

In general, with the S^2F^2M method we obtain a better prediction that using BBOF and GLUE. We found that the disadvantages of GLUE and BBOF methods are that the vector of parameters selected as the best predictor can differ with the real set of parameters. However, because S^2F^2M has a different basis, this kind of problem can never happen. This is a clear result of the benefits of Data Driven Prediction methods based on Multiple Overlapping Solution.

References

1. Abdalhaq B.: A methodology to enhance the Prediction of Forest Fire Propagation. Ph. D Thesis. Universitat Autnoma de Barcelona, Spain (2004)
2. Abdalhaq B., Corts A., Margalef T., Bianchini G., Luque E.: Between Classical and Ideal: Enhancing Wildland Fire Prediction Using Cluster Computing. Journal of Cluster Computing Special Issue on cluster computing in science and engineering. Vol 9, No. 3, pp. 329-343 (2006)
3. 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 (1986)
4. 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)

5. Beven K., Binley A.: The future of distributed models: model calibration and uncertainty prediction. Hydrological Processes 6:279-298 (1992)
6. Bevins C.D.: FireLib User Manual & Technical Reference. (1996) http://www.fire.org.
7. Bianchini G., Cortés A., Margalef T., Luque E.: S^2F^2M - Statistical System for Forest Fire Management. LNCS 3514, pp. 427-434 (2005)
8. 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)
9. Jorba J., Margalef T., Luque E., Campos da Silva J., 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)
10. Jordan H.F, Alaghband G.: Fundamentals of Parallel Processing. Prentice Hall (2003)
11. Montgomery D.C., Runger G.C.: Probabilidad y Estadstica aplicada a la Ingeniera. Limusa Wiley (2002)
12. Morrison R.S.: Cluster Computing: Architectures, Operating Systems, Parallel Processing & Programming Languages. GNU General Public Licence (2003)
13. Pacheco P.: Parallel Programming with MPI. Morgan Kaufmann Publisher (1997)
14. Pincus M.: A Monte Carlo Method for the Approximate Solution of Certain Types of Constrained Optimization Problems. Operations Research, 18. pp. 1225-1228 (1970)
15. Piñol P., Salvador R., Beven K.: Model Calibration and uncertainty prediction of fire spread. pp. 99- 111. ISBN 90-77017-72-0 (2002)
16. Quinn M.J.: Parallel Programming in C with MPI and Open Mp. First Edition. McGraw-Hill (2004)
17. Rothermel R. C.: A mathematical model for predecting fire spread in wildland fuels. USDA FS, Ogden TU, Res. Pap. INT-115 (1972)
18. Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J.: MPI: The complete reference. The MIT Press. Cambridge Massachusetts. London England (1996)
19. Viegas D.X. (Coordinator) et al. 2004.: Project Spread - Forest Fire Spread Prevention and Mitigation. Accessed on November 2008. http://www.algosystems.gr/spread/