Improved Prediction Methods for Wildfires Using High Performance Computing: A Comparison*

Germán Bianchini, Ana Cortés, Tomàs Margalef, and Emilio Luque

Departament d'Informàtica, E.T.S.E, Universitat Autònoma de Barcelona, 08193-Bellaterra (Barcelona) Spain

Abstract. Recently, dry and hot seasons have seriously increased the risk of forest fire in the Mediterranean area. Wildland simulators, used to predict fire behavior, can give erroneous forecasts due to lack of precision for certain dynamic input parameters. Developing methods to avoid such parameter problems can improve significantly the fire behavior prediction. In this paper, two methods are evaluated, involving statistical and uncertainty schemes. In each one, the number of simulations that must be carried out is enormous and it is necessary to apply high-performance computing techniques to make the methodology feasible. These techniques have been implemented in parallel schemes and tested in Linux cluster using MPI.

1 Introduction

Forest fires are a very serious hazard that, every year, cause significant damage around the world from the ecological, social, economic and human point of view [7]. These hazards are particularly dangerous when meteorological conditions are extreme with dry and hot seasons. An example of fire effects under severe conditions is summer 2003. That year, the temperatures in the Mediterranean area were extremely high and there was also a lack of precipitations. A relevant example of fire effects under these conditions was Portugal. In this country alone, 420,000 hectares were burned and, as a consequence, 20 people died.

To reduce the negative effects of fire it is crucial to improve current fire risk assessment methods. These methods base their recommendations on the results provided by a certain wildfire simulator. However, in most cases, the results provided by simulation tools do not match real propagation. Thus, such simulation tools are not wholly useful, since predictions are not reliable. One of the most common sources of deviation in fire simulation spread from that of real-fire propagation is imprecision in input simulator parameters. There are certain parameters that could be defined as static factor parameters such as the terrain slope and the vegetation type because they remain invariable through time. However,

* 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.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part I, LNCS 3991, pp. 539–546, 2006.

there are other parameters, which can be referred to as dynamic parameters, which typically are very difficult to determine as the fire progresses. Some examples of this kind of parameters are the moisture content in the vegetation and wind conditions [12].

One of the most common ways of approaching this problem, consists in searching for an optimal set of input parameters in order to obtain the best simulation results [4]. However, there are some authors that reject the idea that there is only one optimum parameter set in a simulator calibration. They consider that there are multiple parameter sets that may be acceptable in simulating the system under study [2, 11]. GLUE is one of the most relevant methods within this category. The main drawback of this method is caused by its random way of working. To overcome this difficulty, we propose the S^2F^2M method. S^2F^2M applies statistical analysis by simulating the fire propagation, considering a wide range of parameter combinations to determine different scenarios.

The GLUE and S^2F^2M methods are described in section 2 and 3 respectively. Since both strategies need to execute a large number of simulations, we have used the parallel scheme master-worker, implemented with an MPI [8] as a message-passing library and executed on a PC Linux cluster. The implementation details are described in section 4. Section 5 provides a comparison study between both strategies based on real field fires. And finally, the main conclusions are reported in section 6.

2 GLUE Methodology

The GLUE method of Beven and Binley [2] is a Monte Carlo 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 our case, we use a fuzzy measure of goodness of fit. 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 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 ($p \leq 1$):

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

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 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 to 1.

3 S^2F^2M Methodolgy

The methodology of S^2F^2M is based on statistics. 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 [10] (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 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 cell sets. In this manner we obtain a matrix with values representing the probability of each cell catching fire.

In this way, after calculation of the ignition probability (P_{ign}) for each cell, we can compare the real case against our matrix. Then, we find a key P_{ign} which defines an area similar to the real situation. So, we can use this value to predict in a next time the possible fire behavior.

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 [12].

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 (parameter domain cardinality), which is calculated as follows:

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

Finally, from each parameter's cardinality 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 Implementation

Both methods described above have been implemented in two operational systems that incorporate a simulation kernel and apply a methodology to evaluate the fitness function. This system has been developed on a PC Linux cluster using MPI as the message passing library.

4.1 The Simulator

The S^2F^2M and GLUE system use as a simulation core the wildland simulator proposed by Collin D. Bevins, which is based on the fireLib library [5]. **fireLib** is a library that encapsulates the BEHAVE fire behavior algorithm [1]. 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 at what time the fire will reach those cells.

As inputs, this simulator accepts maps of the terrain, vegetation characteristics, wind and the initial ignition map.

The output generated by the simulator consists of 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 both systems use an approximation based on cells, the fitness function was specified as follows:

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

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

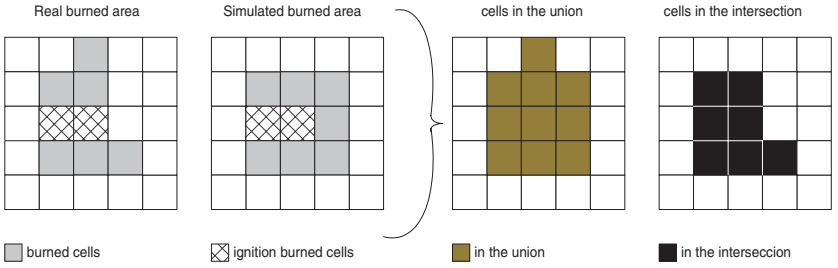

Fig. 1. Calculating the fitness for a 5 x 5 cell terrain

of the simulation results and the real situation, and $\#IgnitionCells$ represent the number of burned cells before starting the simulation.

Figure 1 shows an example of how to calculate this function for a terrain made up of 5x5 cells. In this case, the fitness function is $(7-2)/(10-2) = 0.7124$.

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 the simulation did not coincide with reality at all.

4.3 Parallelisation

GLUE and S^2F^2M have to make a large quantity of calculations because they use a sequential simulator as a kernel [5]. For this reason they need to make a simulation for each resulting combination of parameters ($\#Scenarios$), giving as a result a very time consuming simulation.

Using multiple computational resources working in parallel to obtain the desired efficiency is a solution. 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. GLUE implementation follows the same scheme as S^2F^2M. For more information about the S^2F^2M implementation, see [3].

5 Experimental Results

To compare the systems we used three experiments in the field. These burns took place in Serra da Lousã (Gestosa, Portugal (40°15'N, 8°10'O)) , at an altitude of between 800 and 950 m above sea level. The burns were part of the SPREAD project [9]. In the Gestosa field experiments [6], terrain was divided into dedicated plots in order to carry out different sorts of tests and measurements. We worked with plots 520, 533 and 534, which had the following characteristics:

Experiment 1 (Plot 520): the plot was represented by means of a grid of 89 columns x 109 rows and the slope was 18°.

Experiment 2 (Plot 533): the plot was represented by means of a grid of 95 columns x 123 rows and the slope was 21°.
Experiment 3 (Plot 534): the plot was represented by means of a grid of 75 columns x 126 rows and the slope was 19°.

In the three experiments each cell was 3.28083 x 3.28083 feet, and the other 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 fire. The video obtained was analyzed and several images were extracted every 2 minutes in the first and second experiment, and every 1 minute in the third. From the images the corresponding fire contours were obtained and converted to cell format in order for S^2F^2M and GLUE to interpret them.

Experiments were conducted on a cluster (16 processors) of homogenous Pentium 4, 1.8 Ghz (SUSE Linux 8.0) connected by 100 Mb/sec network.

5.1 Experiment 1

To make comparisons we fixed the initial time to 6 minutes (when it is possible to do the first prediction) and a limit value of 14 minutes. In figure 2 we can observe that the prediction proposed by S^2F^2M is always better or, in the worse case, equal to the GLUE predictions. The highest fitness value (0.7624) is reached at time 12. Also it is possible to observe that in time 6, 10 and 14, the fitness value becomes significantly lower. However, that value is still above GLUE fitness and above the best individual case of S^2F^2M. On the left figure it is possible to see the real propagation.

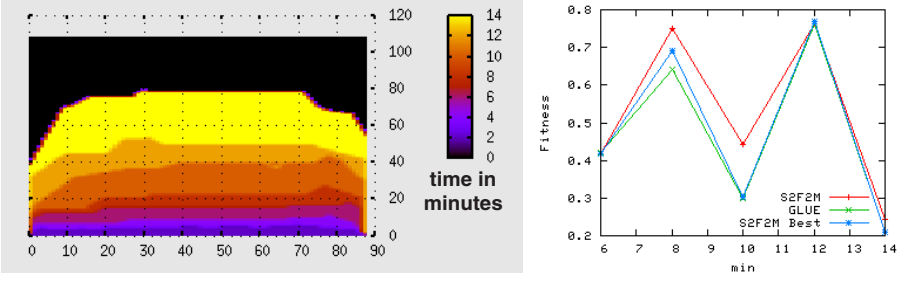

Fig. 2. Real spread for Plot 520 (axes in feet). Comparison between the methods.

5.2 Experiment 2

The second experiment had a similar duration to previous burning. Figure 3 shows the real propagation and the resulting fitness after applying the methods studied. This is an interesting case to analyze, because in it our method has a lower fitness than the others on three of four points. Such a situation has an explanation, which is related to the method. If we look at figure 4, in this simple

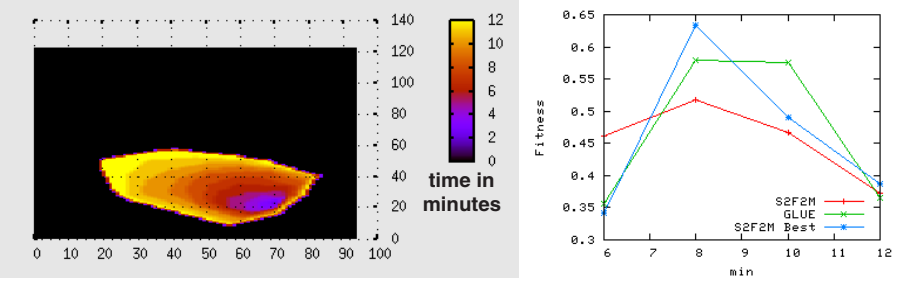

Fig. 3. Real spread for Plot 533 (axes in feet). Comparison between the methods.

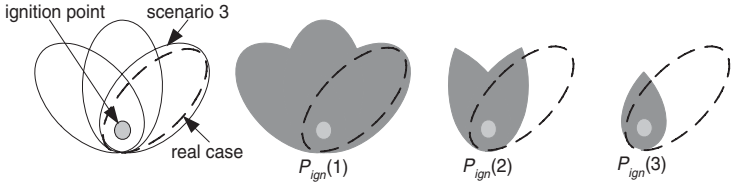

Fig. 4. Explanation

example we have only three scenarios. Scenario number 3 has a similar behavior to the real case, but when we choose $P_{ign}(1)$ or $P_{ign}(2)$ or $P_{ign}(3)$, we discover that fitness on each case is lower than those individual case.

5.3 Experiment 3

Finally, figure 5 shows the fitness function obtained on plot 534 (the shortest experiment). It is possible to identify clearly that the S^2F^2M, as in the first experiment, produces the best predictions.

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

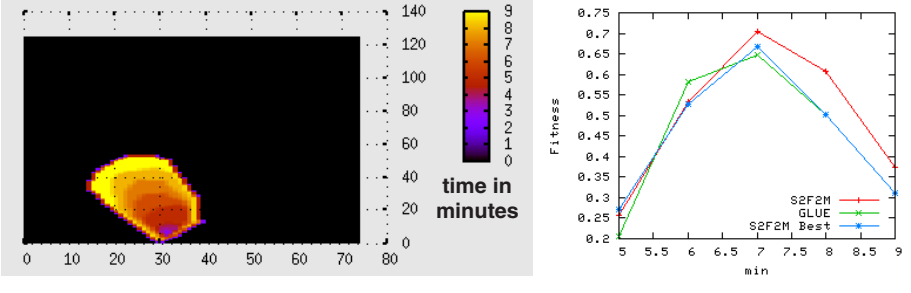

Fig. 5. Real spread for Plot 534 (axes in feet). Comparison between the methods

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 the wind speed value is high, when we use this set in a time $z + 1$ where the wind speed value changes brusquely, this new prediction will be distant from the real situation.

6 Conclusions

We have compared two methods which aim to avoid the problem caused by the lack of precision for dynamic input parameters. In general, with the S^2F^2M method we obtain a better prediction that using GLUE. In some case we saw that the S^2F^2M prediction was slightly worse, but importantly we know the reason why, and we can look for a solution to improve the method.

Because of the high number of simulations (in average 60,000 for each step) it was necessary to apply high-performance computing techniques to make the methodology feasible. We reduce considerably the execution time (we reached a speed-up of 14.43 using 16 processors with S2F2M and 13.94 with GLUE), and, for this reason, we think that parallel processing opens new possibilities for applying the methodology to real-time environments.

References

1. 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.
2. Beven K., Binley A. 1992. "The future of distributed models: model calibration and uncertainty prediction". Hydrological Processes 6:279-298.
3. Bianchini G., Cortés A., Margalef T., Luque E. "S^2F^2M - Statistical System for Forest Fire Management". LNCS 3514, pp. 427-434. 2005.
4. Abdalhaq B., Bianchini G., Cortés A., Margalef T., Luque E.: "Improving Wildland Fire Prediction on MPI Clusters". LNCS 2840, pp. 520-528, 2003.
5. Collins D. Bevins, "FireLib User Manual & Technical Reference", 1996. http://www.fire.org.
6. ADAI - CEIF (Center of Forest Fire Studies) http://www.adai.pt/ceif/Gestosa/
7. Morgan P., Hardy C., Swetnam T. W., Rollins M. G., Long D. G. 2001. Mapping fire regimes across time and space: Understanding coarse and fine-scale fire patterns. International Journal of Wildland Fire, Vol. 10: 329-342.
8. MPI: The Message Passing Interface Standard http://www-unix.mcs.anl.gov/mpi/
9. Project Spread, Forest Fire Spread Prevention and Mitigation http://www.adai.pt/spread/
10. Douglas C. Montgomery, George C. Runger, "Probabilidad y Estadística aplicada a la Ingeniería", Limusa Wiley 2002 ISBN: 968-18-5914-6
11. Piñol P., Salvador R., Beven K. "Model Calibration and uncertainty prediction of fire spread". 2002, pp. 99- 111. ISBN 90-77017-72-0
12. Rothermel R. C., "A mathematical model for predecting fire spread in wildland fuels", USDA FS, Ogden TU, Res. Pap. INT-115, 1972.