Chapter 4

Optimization Techniques: Comparative Study.

"-dijo Sancho- ¿No le dije yo a vuestra merced que mirase bien lo que hacía,..."

Miguel de Cervantes

Once the optimization framework has been satisfactory implemented including some optimizing techniques, we tested them for the particular case of forest fire propagation in order to find the points of strength and weaknesses of these algorithms and, furthermore, to realize their ability to afford this real problem. In this chapter, we describe the performed experiments to tune these algorithms and to try to answer the question "Which method is most suitable to solve this problem?" Some discussion of the obtained results is also provided.

4.1 Platform Description

We used a cluster of 21 PCs with Celeron processor 433 MHz. Each had 32 MB RAM connected with Fast Ether Net 100 Mb. Furthermore, the operating system installed was Linux SuSe. All the machines were configured to use NFS (Network File System) based on one server which has the same characteristics of the other machines. The middle-ware used to make parallel programs was MPI[51, 50]. MPI is a library specification for message-passing, proposed as a

standard by a broad based committee of vendors and implementors.

4.2 Experiment Description

The strategy we used to perform our experiments was to start from very simple cases in order to obtain a good knowledge of the problem behavior. The next step consists of developing such simple cases toward more complex situations until we reach a real case. Since the wind has been shown to have a great effect on the shape and size of the fire front, we started our set of experiments optimizing the wind parameters and the rest of input parameters were kept fixed. Once we had our system mature, we expanded the optimization to all the parameters of the simulator.

In order to simplify the initial experiments, we deal with an easy imaginary terrain description. The terrain we used was a completely plain square terrain, whose dimensions were 6750mx5150m and which also contains homogeneous vegetation in the whole area.

To optimize the simulator using our framework, we need to compare the results obtained during the optimization process against a real fire line. However, to have access to real data about fires such as the initial shape of the fire line, fire line propagation, wind parameter and so on, in the time of the experiment was virtually impossible. For this reason, we created our own real fire line using the same simulator we wanted to tune (ISS). This real fire line was obtained by fixing all the input parameters needed by the ISS simulator as has been described above, and using as a wind vector (wind speed and wind direction) two values by us set a priori. Once the real fire line was obtained, we supposed that we did not know the real parameters' values that leads to the synthetic real fire line. Afterward, we used the BBOF system to find the two parameters related to wind.

As we mentioned in chapter 1, one potential source of simulation errors may reside in the model simulated itself because of its impossibility of exactly reflecting the real behavior of the fire spread. Therefore, this initial way of proceeding - where the objective fire line is created in a synthetic way- have the advantage of avoiding the possible errors in the simulator, because we have the certainty that there exist some parameters' values that if applied to ISS leads to the synthetic real fire line we have.

The aim of the following experiments was to investigate the ability of the

proposed Optimization Framework to tune the hidden parameters but not to test the simulation modules. However, the proposed BBOF system can also be used to test the goodness of the model itself. For that purpose, we should follow the same scheme mentioned above for finding some lost parameter and, afterward, analyze whether the obtained vector of parameters can be found in the nature or not. If the obtained values are impossible to be matched within the reality it can be considered that there is some error in the model that keeps track of those parameters.

In the following section, we shall describe the procedure to obtain the synthetic real fire lines to be used in the set of experiments reported in this chapter.

4.2.1 Creating the Synthetic Real Fire Line

As mentioned in the previous section, wind direction and wind speed have been chosen as the tested parameters in our experimental study. Since we have included ISS the possibility to deal with wind fields, we have divided our experiments into two groups: depending on whether the real fire line has been obtained using the same wind speed (W_s) and wind direction (W_d) for all the terrain or a variable wind field. In the first case we talk about a fire line obtained with a homogeneous wind field, whereas in the second case, we refer to it as heterogeneous wind field. Within the heterogeneous group, two different wind fields have been used to generate a real fire line: smooth wind field and rough wind field, in order to test different wind conditions. Subsequently, we describe how each fire line has been obtained using the different wind fields just described.

4.2.2 Homogeneous Wind Field

We used the simulator with homogeneous wind field (i.e. one wind speed and wind direction for all simulation time and all the terrain) as is shown in figure 4.1. Wind speed and wind direction were set up to 15 km/h and 180º respectively. We have chosen these values because the fire line outcoming from this speed is not so large to exit from the terrain we have and not so small that the effects of changing will be slight.

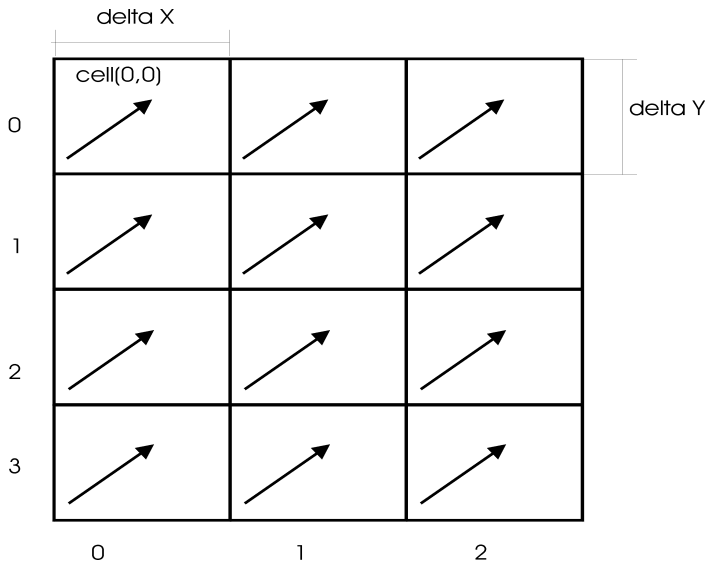

Figure 4.1: homogeneous wind field

4.2.3 Smooth Heterogeneous Wind Field

We have created a realistic wind field by changing the wind vector from one terrain cell to its neighbor's cells. The wind vector changes across time and the two resulting space dimensions (X, Y). The terrain of the simulation have been divided virtually into 10x10 cells and we assigned for each cell a different wind value. Each wind value differs from its neighbors by an arbitrary factor (sp). Furthermore, we have divided the time of the simulation which was 45 minutes into 3 time steps, each of 15 minutes. The vector assigned to each cell changes by another arbitrary factor (tim). Taking into account these two variation factors, the formulas used to generated the wind speed and wind direction for a given cell are the following:

$$Ws(x, y, s) = ws - ws \times sp \times x + ws \times sp \times y + ws \times sp \times tim \times s$$

Where:

x is an integer representing the horizontal location of the cell,

y is an integer which represents the vertical location of the cell,

s is integer number representing the order of the time step (1,2 or 3 in this case),

$Ws(x, y, s)$ is the wind speed for cell (x,y) in time step number s,

ws is the seed for the wind speed.

The wind direction is calculated in the same way:

$$Wd(x, y, s) = wd + wd \times sp \times x - wd \times tim \times s$$

Where :

Wd is the wind direction for the cell (x, y, z),

wd is the seed for the wind direction.

Applying the formulas, the wind speed increases by factor (tim) by time and by factor (sp) by y diminution and decreases by (sp) through the x diminution. And wind direction increases by time and both directions by factors (tim) and (sp), respectively. In order to create a smooth wind field, we used wind speed ws equal to 10 and a wind direction wd equal to 45, furthermore, sp and tim were set to 0.01 each. The wind field created by these parameters was not very abrupt and, therefore, it could be considered as smooth. Figure 4.2 illustrates a small heterogenous wind field for a 3x4 cells terrain.

4.2.4 Rough Heterogeneous Wind Field:

We also created a rough wind field in the same way as the smooth heterogeneous wind field was created but, in this case, we use sp equal to 0.02 and tim equal to 0.1. This wind field description provides a wind that varies more than the smooth wind field. This case reflects when the wind changes quickly, and in this text will be called rough heterogeneous wind field (see figure 4.3).

4.3 The Objective Function

In our case, the objective function should consider two different aspects: the suitableness of the solution and its reliability. The suitableness deals with the degree of matching between the fire-line obtained for a particular parameters vector after executing the ISS simulator for a time period equal to T, and the real

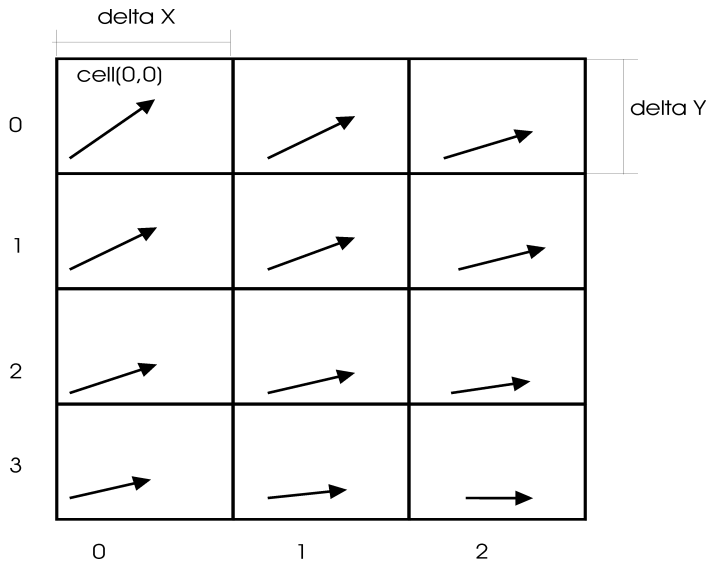

Figure 4.2: Smooth heterogeneous wind field

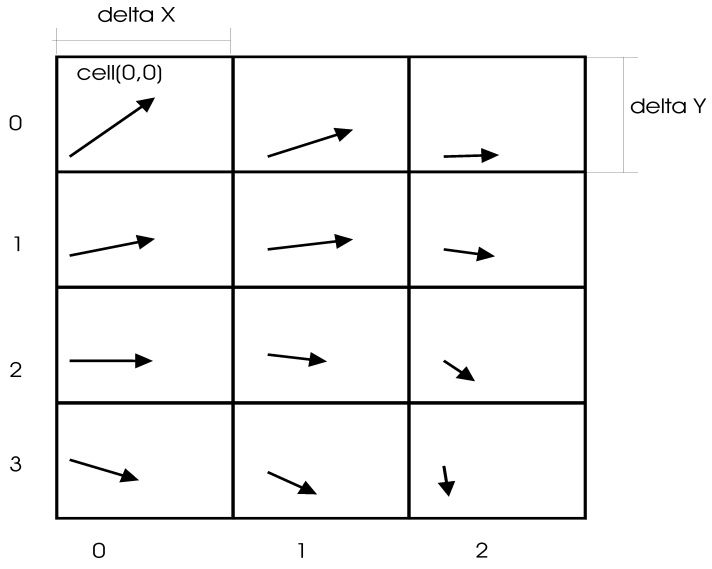

Figure 4.3: Rough heterogeneous wind field

fire-line after expired this period T. Reliability considers physical restrictions in order to avoid solutions that provide impossible real situations. We will try to optimize both objectives at the same time.

For suitability we have chosen the Hausdorff distance [89]. Hausdorff distance is used in the literature of shape recognition to measure the degree of concordance of two shapes.

In order to deal with the reliability of the objective function, spectacular wind changes were considered less likely to occur than light wind speed and wind direction variations. For this purpose, we include a factor P that measures such restriction. The P factor is evaluated as the addition of vector difference between consecutive wind vectors provided by a particular solution. Formally, the P factor is defined as follows:

$$P = \sum_{n=1}^{m-1} |\overrightarrow{w_{n+1}} - \overrightarrow{w_n}|$$

More precisely, for each parameter vector we need to evaluate the Hausdorff distance and the P factor. In order to consider both aspects in the prediction error and control their influence on this, we introduce a parameter called α which determined the weight of each aspect in the objective function (L). Finally, the L is evaluated as

$$L = \alpha H + (1 - \alpha)P$$

The smaller the prediction error, the better the corresponding solution.

4.3.1 Preliminary Analysis of the Objective Function

As we have previously mentioned, we are interested in finding the optimal parameters that make the result of the simulator match the real fire line. Since the Hausdorff distance HD will tend to zero as the objective results improve, our aim is to find the global minimum of this objective function, that means that Hausdorff distance between the simulated fire line and the real fire line is minimum.

At this point, many questions about the shape of the objective function occur to us like: does our objective function have local minimums or only one global minimum? Is the function smooth all over the space? Is there more than one global minimum of the function? In order to answer all these questions, we tried to sketch some cross sections of the simplest objective function. For this purpose,

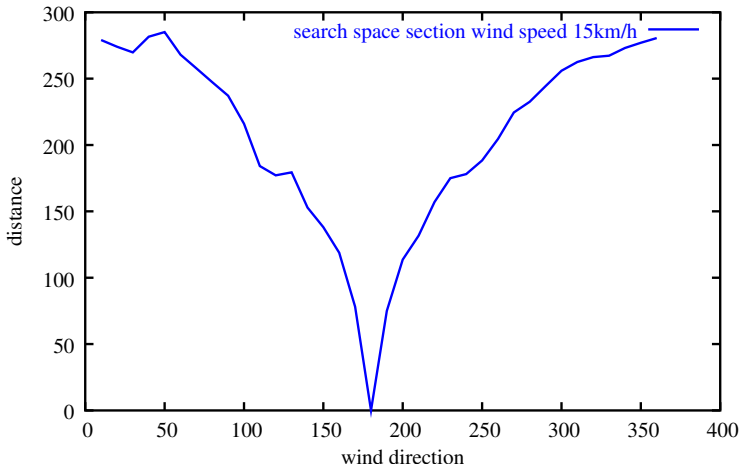

Figure 4.4: Objective function cross section at optimal wind speed (15 km/h) using a homogeneous wind field to generate the real fire line

we deal with the fire line created in the simplest way using the homogeneous wind field. In this case our search space have only two dimensions, wind speed and wind direction, which provides us an easy and affordable objective function (the Hausdorff distance).

In order to easily analyze the search space of our objective function, we have fixed one dimension at the true value and we have drawing the other one within its valid range. Therefore fixing the wind speed on the optimal value (15 Km/h) and changing the direction, we will have a cross section of the function in the optimal wind speed. Figure 4.4 shows that the function has one global minimum and several local minimums and it is somewhat symmetric around the global minimum, having a "V" shape in general.

For sake of completeness, we also sketched the objective function at wind speed equal to 20 km/h and 5 Km/h, which are respectively depicted in figures 4.5 and 4.6. The graph shown in figure 4.5 denotes a strange behavior of the function: it has many local minimums and the global minimum of this cross section (which is also a local minimum) is shifted from the optimal minimum a little.

Furthermore, when we observe the shape of the search space depicted in figure 4.6, we notice that the behavior of the function is very strange. The reason for such a behavior is because the direction does not affect the fire-line very much when the speed of the wind is small and it has no effect at the speed

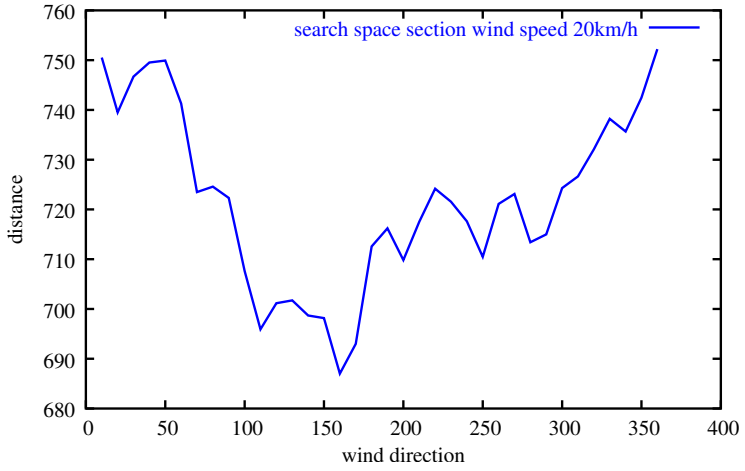

Figure 4.5: Objective function cross section at wind speed=20 Km/h using a homogeneous wind field to generate the real fire line

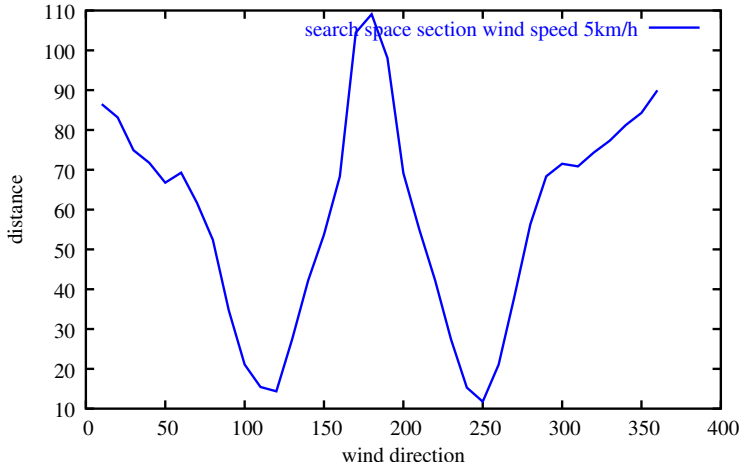

Figure 4.6: Objective function cross section at wind speed=5 Km/h using a homogeneous wind field to generate the real fire line

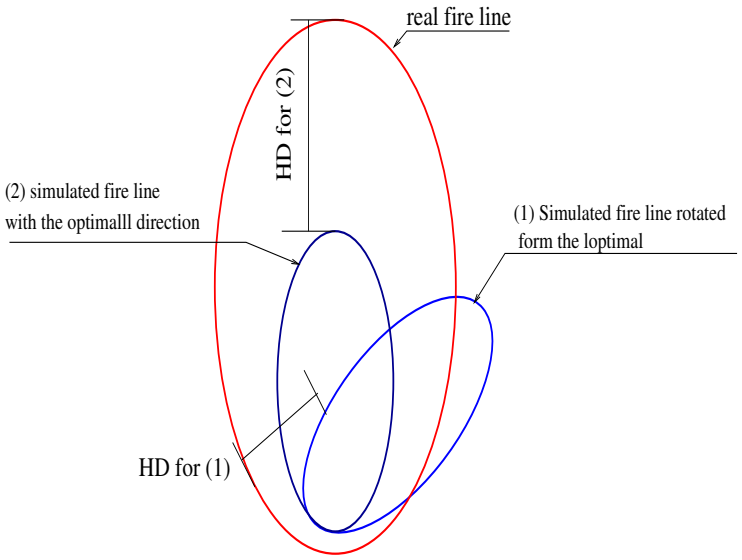

Figure 4.7: Two simulated fire lines with speed smaller than real.

of 0 km/h.

It seems strange that in the optimal direction we have the maximum (worst) distance. This can be explained as the following: in the optimal direction the curve is well centered inside the real fire line when calculating the Hausdorff distance, we get the distance behavior as show in the figure 4.7, which is greater than the distance of the rotated simulated fire line. In other words, when rotating the graph, in this case, their points get closer to the points of the real graph, which is larger.

Once we have analyzed the behavior of the objective functions taking into account its relation to the wind direction, we perform the same analysis but now considering what happens to the Hausdorff distance when fixing wd to 180^o (the optimal value) and varying the wind speed. The obtained graph is shown in figure 4.8.

We can observe that the curve is smooth but not very symmetric. It has a more linear form when the speed is greater than optimal and the slope changes when the speed is less than optimal. More precisely, it starts with a big slope, then the slope decreases by the decreasing of the speed.

The function we have is not continuous because we are calculating the distance of the farthest two points of the two fire lines. By changing the orientation

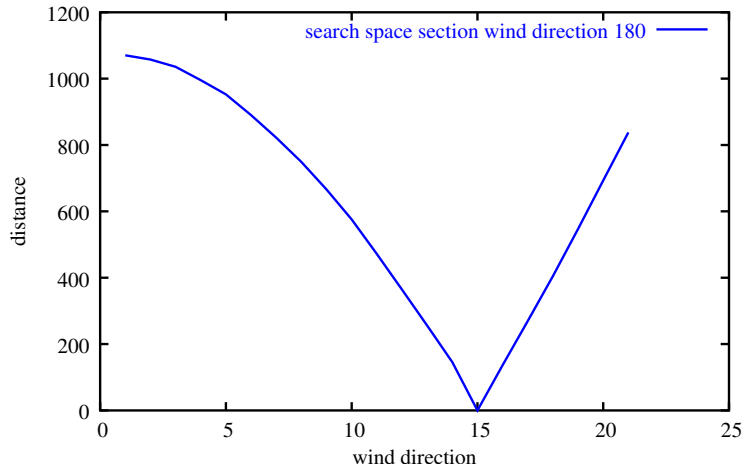

Figure 4.8: Objective function cross section when wind direction is fixed at the optimal value (180°)

or the size of the graph the function may return the distance between other two points.

After this study of the objective function, we could imagine a surface that has a dominating global minimum and several local minimums. If we look at the surface as a whole, ignoring the local minimum, it may have a bowl shape, which can be approximated to a parabolic function. This conclusion is useful to create the approximation model for the analytical optimization approach.

Subsequently, we describe the performed experiment considering the different wind field created and bearing in mind the "assumption" that our objective function could have a parabolic aspect.

4.4 Homogeneous wind Field

We started our experiments with a simple case: homogeneous wind field (i.e. one wind speed and one wind direction). Therefore, we had only two parameters to tune. Since we do not consider any wind variation (temporally or spatially) throughout all the experimental study reported in this section, the objective function to be evaluated can be reduced to evaluate the Hausdorff distance without considering the physical restrictions. Bear in mind the full description of the objective function that has been introduced, the complete objective function

is described as follows:

$$L = \alpha H + (1 - \alpha)P$$

In this case, factor α is considered to be 1 and, therefore, the objective function will be reduced as follows:

$$L = H$$

where H represents the Hausdorff distance.

In the following section, we will discuss Genetic Algorithm (GA), Taboo Algorithm (TA), Analytical and Random algorithms for the simple case. Since GA and TA have themselves certain parameters that should be tuned in order to obtain their best performance, we attempted to tune GA and TA parameters for our particular case. Afterwords, we compare the performance of the four optimizations techniques with respect to the number of evaluations of the objective function and the goodness of the best value obtained.

4.4.1 Tuning Genetic Algorithm

As has been described in section 2.6.3, the Genetic Algorithm has many parameters to be tuned and they all affect the performance of the algorithm. These algorithm parameters are: generation size, mutation probability, crossover probability, number of crossover points, scaling factor and, in our implementation, we have added the mutation distance which is the real number that is added to the mutated parameter. To start the process of tuning, we fixed certain algorithm parameters from the preliminary experiment and we extensively studied others.

Fixed Parameters:

Mutation: This involves two parameters *mutation probability* and *mutation distance*. We used a large probability of mutation 40% and a large distance to generate genetic diversity, because we use real number representation. If we look at the binary representation, we see that the probability is for each bit which will be more for each real number because the real number is more than one bit. In the binary representation, mutation is reversing mutated bit from 1 to 0 or from 0 to 1, which means a big number in the higher significant bits. (16, 32, 64 ... etc).

Scaling factor	No scaling	1.1	1.2	1.3
Hausdorff Distance	11	135	85	96

Table 4.1: The effect of scaling factor on the convergence of GA with generation size equal to 20 after 200 evaluations.

Crossover: This also involves 2 parameters the *crossover probability* and *crossover number of crossover points*. As recommended in the literature, we have chosen a very high crossover probability (99%) to force the algorithm to explore more space every generation, however, this setting can be tuned in later search. The second parameter is the crossover number of points; In this case, we have chosen the simplest one which was one crossover point.

Tuned Parameters:

Generation Size and Scaling: were subject to the tuning process, which is subsequently described.

Since each experiment starts from a random set of initial guesses of the parameters, we performed 5 different experiments for each particular setting of the generation size and scaling factor. The five results were averaged in order to obtain a reliable value. From the literature, we have extracted that the best scaling factor is around 1.2 [27], therefore, we have tried three scaling factors around this recommended value 1.1, 1.2 and 1.3. Furthermore, and in order to observe what happens if this factor was not considered, we also checked by not scaling. The generation size chosen for doing these experiments was 20.

Figure 4.9 shows the evolution of the best distance (objective function) as the number of executions of the objective function increases (each generation executes the objective function as many times as indicates the generation. Which was equal to 20). The distance after 200 evaluations is summarized in table 4.1.

If we only consider the effect of the scaling factor when this is non-negligible, we observe that the best results are obtained when the scaling factor is set to 1.2. However, in this case, without scaling the system converges to the optimal of the objective function faster. This fact is due to the kind of crossover we use, which uses the average of the parameters and the mutation. This seems to be sufficient to create genetic diversity and there is no need to use scaling. In order to evaluate the effect of scaling to different generation sizes, we also analyzed the evaluation of the optimization process for a generation size equal

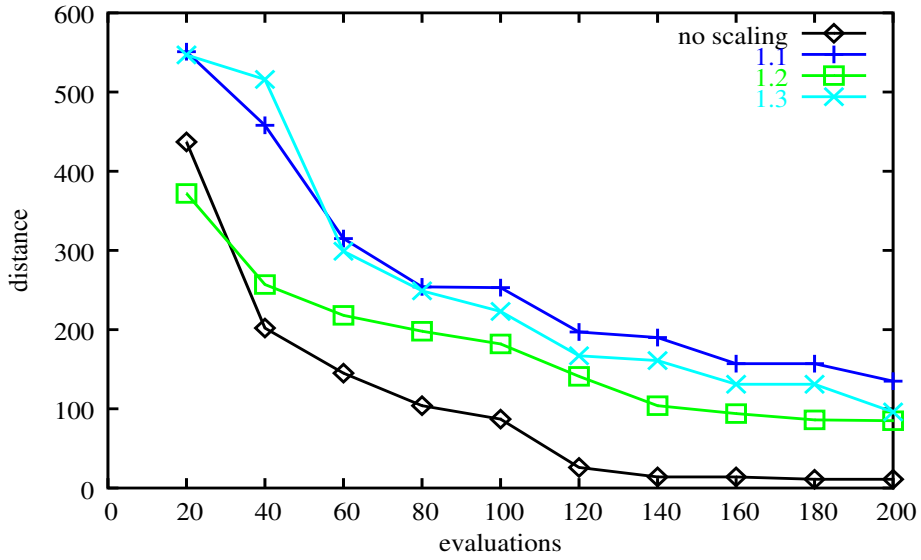

Figure 4.9: The effect of the scaling factor on the convergence of GA using a generation size equal to 20

to 10. Since the best scaling factors for the previous experiments were 1.2 and no-scaling, we performed the new experiments only considering the two values.

Figure 4.10 shows the Hausdorff distance as the number of generations increase for the above commented set of GA's parameter. We can observe that, without scaling, the algorithm converges faster at the beginning of the search process, but this becomes slower after certain number of iterations. The HD distance at the end of the optimization process was 98m when no-scaling is used and 50m when we applied a 1.2 scaling factor.

Once we have performed the two previous experiments, we compare the best result for each one between themselves. In the case of generation size 20, the best distance was obtained when no scaling factor was used, whereas with generation size 10 was with scaling factor equal to 1.2. Figure 4.11 shows this comparison. For comparison purpose, we look at the number of evaluations of the objective function, which is shown in the x-dimension, whereas the y-dimension represents the objective function. Since our aim is to minimize the number of objective function evaluations, the best result will be the one that minimizes this factor (bear in mind that the number of function evaluations is obtained by multiplying the generation size by the number of generations).

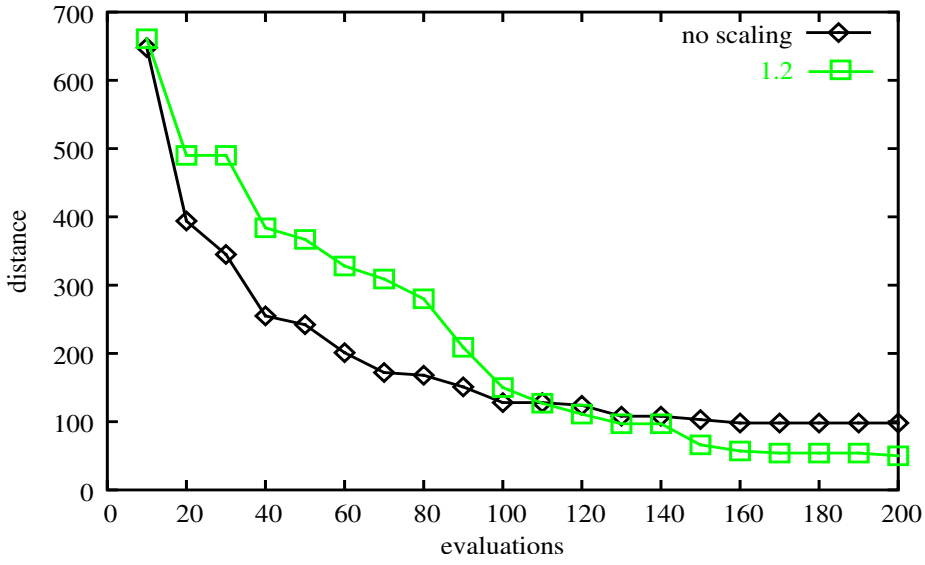

Figure 4.10: The effect of the scaling factor on the convergence of GA using a generation size equal to 10

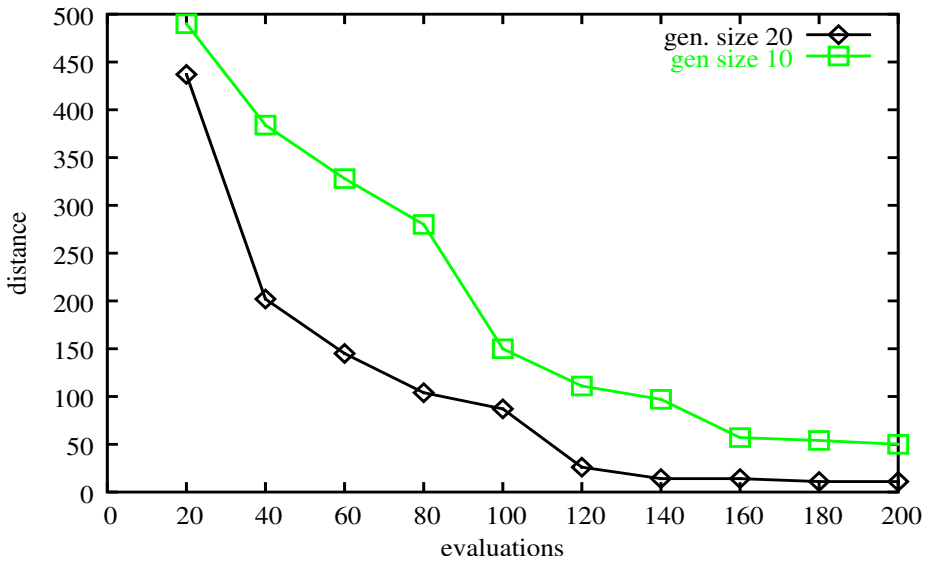

Figure 4.11: Comparison between generation size 10 and 20 when executing GA as optimization techniques

In figure 4.11, we can observe that using a generation size of 20 without scaling converges faster all over the test iterations. We can conclude then that the generation size of 20 and the mutation we used are sufficient to maintain the genetic diversity and there is no need to use scaling.

4.4.2 Tuning Taboo Algorithm

We used the Taboo algorithm described in section 2.6.1. Taboo algorithm has fewer parameters than GA, therefore, it is easier to tune. The parameters that should be tuned are the following:

ϕ: *frequency* participation in the sorting of the evaluated vectors, which has an effect on choosing the vector for the next move.

taboo tenure: The number of iterations that a given *move* is taboo have been chosen for obvious reasons. *Taboo tenure* was equal to 1 and *contra move tenure* equals to 1 as well. We have chosen the taboo tenure in such a way that lets no more than the half of the moves be marked as taboo in each iteration due to short-term taboo memory. If many moves are marked as taboo, the algorithm will choose moving directions that are very bad, while using a very small number of tenures makes almost all the moves available and the algorithm chooses the minimum of the neighbors every time falling in local minimums very easily.

The algorithm needs to start from a point in the search space. Since we consider that we are totally ignorant at the beginning of the process, we can start from more than one random choice. Then, we can continue with the normal algorithm. However, in this case, we start from one random point because the number of evaluations of the objective function is not so large as spend some of them randomly.

On average, we have observed that the taboo algorithm is able to converge to zero (i.e. finds the optimal solution) in 55 iterations. Since during each iteration, the TA evaluates 4 point in the search space, which means that 220 evaluations of cost function are needed to reach the optimal.

Furthermore, we tried to see the effect of frequency on the taboo algorithm. For this purpose, we execute the algorithm without using frequency and with a weight of 10% ($\phi = 0.9$).

Figure 4.12 shows the convergence of taboo search with ϕ equal to 1.0 and shows the same algorithm changing ϕ to 0.9. The results obtained for a value of

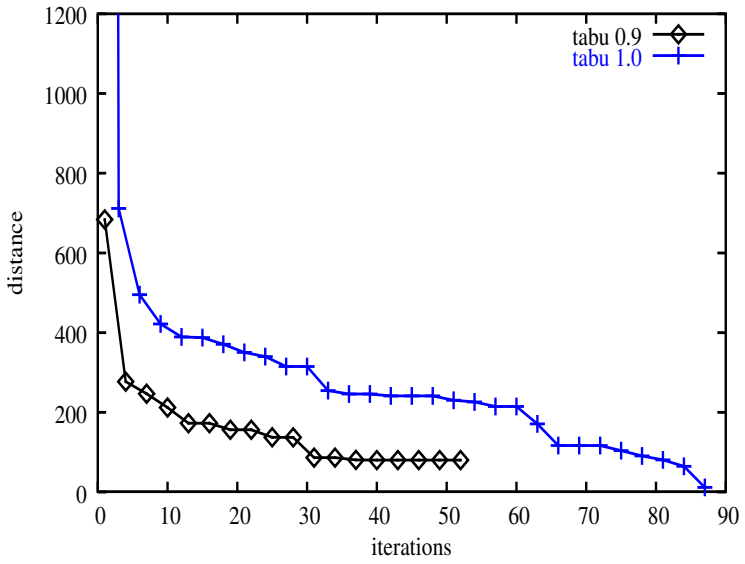

Figure 4.12: Taboo Algorithm evolution with $\phi = 1.0$ and 0.9

	$\phi = 1$	$\phi = 0.9$	$\phi = 0.8$
ITERATIONS	62	54,6	Do not converge
NUMBER OF EVALUATIONS	248	218,4	Do not converge

Table 4.2: Taboo Algorithm performance changing ϕ.

ϕ equal to 0.8 are not depicted because these experiments do not converge to a stable value. Using frequency has a direct effect on diversification. This result means that some degree of diversification is good but too much is very bad.

The optimization process was executed until the optimal distance (0m) was reached (notice that in the case of GA this distance has not been reached). The same process has been performed 4 times and the average for these 4 runs of the algorithm is provided in Table 4.2

From table 4.2 we conclude that there is a difference between the results provided by the different frequency used (in the cases that the process converges), the best result, on average, is obtained when using ϕ equals to 0.9. This result is expected because using the frequency helps in exploring the search space. Using ϕ less than 0.9 will make the algorithm explore more space that is not necessary to be explored. We have experimentally observed that the algorithm did not converges to zero using ϕ equal to 0.8 within acceptable number of iterations.

Degree	χ^2	Wind speed	Wind direction	Distance
2	8.097e+06	7.5	251	621
3	4.174e+06	9.9	264	443
4	3.983e+06	6.9	43.9	477
5	3.307e+06	5.4	40	654
6	4.381e+07	2.4	40	845
7	3.837e+07	0	125	888
8	6.689e+07	0.3	46	887

Table 4.3: The results of analytical algorithm for different degrees

4.4.3 Analytical Search

As described in section 3.3.2.1, one of the implemented optimization techniques in our optimization framework was the analytical approach. From the simple analysis of the behavior of the objective function performed in section 4.3, we assumed that a parabolic approximation was a justifiable choice in our case. However, as we will see throughout the following experimented results, this choice has been shown not to fit well to our problem. Furthermore, we state what was widely mentioned in the literature: analytical optimization is not recommended when the objective function is not calculus friendly [27].

We ran the analytical algorithm changing the degree of the fitted function from 2 to 8. The set of vectors was generated randomly as in the other optimization scenarios. In table 4.3 the results for different runs changing the degree of the approximated function are shown. We can see that χ^2 increases until reaching the degree of 5 and it decreases again for bigger degrees. The best distance was at degree 3, which is 443 meters, however this is a very poor result.

Using this analytical method in any degree, we did not reach an acceptable result. We can see in table 4.3 that the best HD is 443m, which is even worse than the random search, as we will show later on this chapter. We also tried to use mesh (i.e. divide the search space into equal parts and then take the corner vectors of the division) and a set of vectors that was the last generation of the genetic algorithm but the results were the same. The values we get are far from optimal, even farther than the random search. The reason for these bad results can stem from the rare shape of the objective function always preceding a deviation from the global minimum that means that the objective function cannot be approximated to the mathematical model we have proposed.

Algorithm	Genetic	Taboo	Mathematics	Random
Distance	11	0	443	147,25
Evaluations	200	220	200	200

Table 4.4: Comparison between all the algorithms for homogeneous wind field

4.4.4 Random Search

The main purpose of experimenting random search is to compare with other techniques. To be fair in the comparison, we used the same number of objective function evaluations used in the other techniques. When using the random algorithm for the homogeneous wind field we ran the program 4 times for 200 wind vectors randomly created, keeping the minimum of each run. We obtained the average of minimums as 147,25m of the distance between the real and the simulated fire line. These results clearly differ from the results obtained when GA and TA are applied.

4.4.5 Comparison Study

In this section, we summarize the results obtained for all tested optimization techniques in order to deal with the following question: "What is the best algorithm to optimize the objective function we have under more complex wind field conditions?"

In table 4.4 we illustrate together the results from all the algorithms we have tested in order to easily compare them. In particular for the GA and TA techniques, the results included in table 4.4 are the mean results of the tuned algorithm.

As we can observe, TA manages to reach the optimal number (distance 0) in 220 evaluations, on average. However, the GA reaches a distance, on average, equal to 11m and it goes gradually to the optimal. The Analytical algorithm suffers from deviation from the optimal due to the model and the rare shape of the objective function.

Comparing the Genetic and Taboo algorithms with respect to the Random algorithm is a way to obtain the improvement provided by those algorithms. In other words, the utilization of some optimization technique, which improves the goodness of the guesses during the optimization process, provides a significant improvement in the results.

From table 4.4 we can extract that, on the one hand, Genetic and Taboo

are both candidate algorithms to solve optimization problems of the type we are dealing with. On the other hand, in the case of the analytical approach, we need a better model, which is very complex to determine and, furthermore, which will be problem specific.

Since the previous experiments consider only two parameters and in order to make sure of the abilities of the algorithms, we need to test them against harder problems. In the following section, we will use a more complex real fire line by using the heterogeneous wind field to obtain it. Consequently, we experimentally check how GA and TA react under more complex problems. In this case, since the wind characterization involves more that only one wind vector the number of parameters to be optimized increases.

4.5 Smooth Heterogeneous Wind Field

In order to deal with more realistic synthetic fire lines, we used the heterogeneous wind filed described in section 4.2.3, to obtain them. This wind field produces a fire line that cannot be exactly reproduced using only two parameters. It can be approximated to some precision using 2/ 4 or more parameters. So we used 4 parameters in this set of experiments to test the ability of the algorithm to reach the optimal distance. These 4 parameters have been obtained as follows: we have divided the total time into two time steps. The first time interval lasted 30 minutes and the second time interval was 15 minutes long. Since the terrain was considered as only one cell, our vector of parameters include wind speed and wind direction for the above two time intervals. Consequently we had four parameters.

At this point, we should bear in mind the full description of the objective function that was introduced in section 4.4 that is as follows:

$$L = \alpha H + (1 - \alpha)P$$

As mentioned before, there are two aspects that our objective function takes into account: the suitability (H) and the reliability (P). The first one deals with the degree of matching between the obtained fire line and the synthetic real fire line. This aspect has been considered for the experimental study reported in the case of homogeneous wind field. The second aspect, which manages the physical restrictions, has not been included in the objective function because all the experiments only deal with one wind vector (This means that the factor α

was set to 1 for those experiments). In this section, since we are dealing with a heterogeneous wind field, we may consider the physical restrictions that are represented by the factor P in the objective function. Therefore, in this case, the value of α should be different to 1 in order to consider both objective function aspects. Consequently, we have another parameter to be tuned for both TA and GA schemes, which is the physical penalty that should be applicable, when we have more than one wind vector. The physical penalty, as previously described, is the difference between all wind vectors. In other words, we added another objective to the process, which is to minimize changes in the wind field.

As for the homogeneous wind field study, we first needed to properly tune GA and TA in order to maximally exploit their abilities under the new problem conditions. In the subsequent sections, we report the experiments performed to tune both strategies by analyzing the evolution of the Hausdorff distance throughout 20,000 evolutions of the objective function.

4.5.1 Tuning Genetic Algorithm

As in the homogeneous wind field experimental study, we fixed certain algorithm parameters and changed others as follows:

- *Mutation probability*: This was set to 40% or, in other words, there is 40% probability of adding or subtracting a real number from a parameter.
- *Generation size*: This takes the values 25, 50, 100,200,400 and 800.
- *Scaling constant*: (c) we tested the scaling factor equal to 1.2 and without scaling.
- *Physical penalty*: the α factor was set to two values: 0.5 and 0, in order to analyze what happens if we consider this restriction or if it was dismissed.

Furthermore, due to the existence of random factors, we execute 4 tests for each setting of algorithm parameters and we then calculate the average of the obtained results. The results of the experiments are shown in the figure 4.13.

The graph shows the result of changing the generation size in various scenarios of using and not using penalty and scaling factors. Each curve corresponds to a certain scenario, and the horizontal axes shows various generation sizes. Each point of a given curve is the mean value of the final situation. From the graph we observe that the algorithm converges to the optimal faster in the two

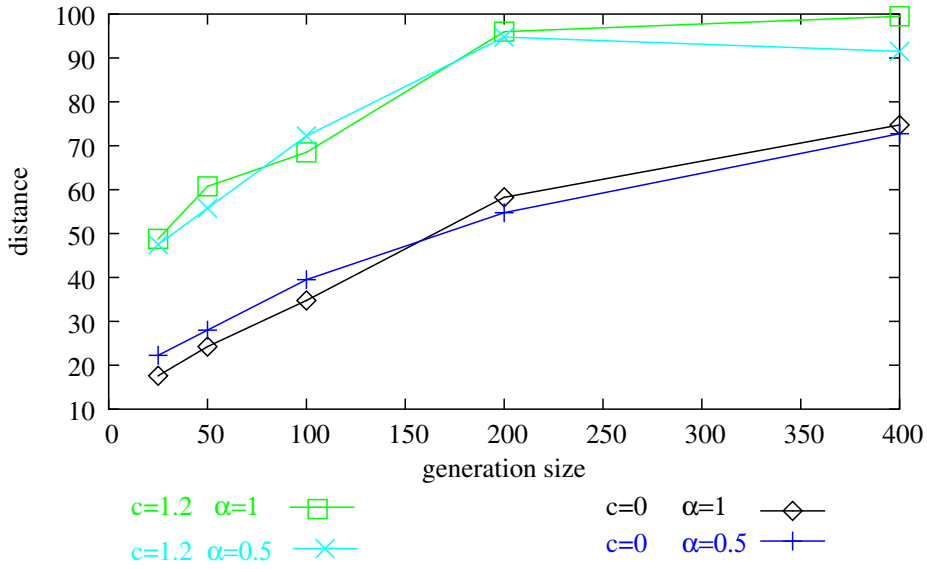

Figure 4.13: GA best distance changing generation size, scaling constant (c) and physical penalty (α).

scenarios that do not use scaling. Here the scaling has a negative effect because the scaling and mutation both work in the same way to explore more space that produces the exploration of unnecessarily evaluations.

The physical penalty has an irrelevant effect in this case. Since we are using two wind vectors, the evaluation of the difference between them, which is at the beginning of the iterations, is small compared to the Hausdorff distance. Therefore, it has a small effect on the selection process. In particular, the Hausdorff distance varies between 600 to 200 and the physical penalty ranges between 30 to 10. We suggest normalizing both Hausdorff distance and physical penalty before getting their weighted average.

Also we can observe a tendency to smaller generation sizes. The optimization process reaches the same performance as the random search in very large generation sizes. We can observe in figure 4.13 when the generation size is 200 or greater with scaling and in the case of not using scaling it is accelerated toward the random search as the size of the generation increases. It is better to have a higher number of iterations using a small generation size than to perform fewer number of iterations with a larger generation size. The main reason for such behavior is that large generation sizes have redundant information about the

search space, therefore, they explores search space unnecessarily. Notice that if we use one generation of size equal to 20,000 individuals, we are practically making a random search. Accordingly, the recommended generation size here is no more than 25 individuals (guesses of the parameter vector).

These results are not very far from the results we have obtained using the simple experiments. Recalling that in the simple experiment, when a homogeneous wind field has been considered, we conclude that a generation size equal to 20 without scaling was a good election. This value is very close to the generation size recommended in this case, which is no more that 25 individuals per generation and without scaling. Furthermore, we have also observed that for tuning only three parameters of the genetic algorithm, we made many executions, which take a long period of time on several non-dedicated machines. Therefore, we can conclude that the difficulty of the GA resides in tuning its own parameters, because its performance hardly depends on them. GA has a high number of parameters and we need to execute a great number of experiments to tune them all. We have observed that non-tuned GA behaves like a Random search, and GA, which has been properly tuned, gives excellent results.

As mentioned in the case of homogeneous wind field, TA also involve the process of tuning its own parameter in order to obtain the highest performance. In the following section, we deal with this tuning process for the Taboo Algorithm. All the experiments reported have been performed using the same fire line and the same number of parameters as in the case of GA.

4.5.2 Tunning Taboo Algorithm

As with the genetic algorithm, we used the fire line that has been created using the heterogeneous wind field where the parameter vector involves 2 time steps and one terrain cell, i.e., we deal with 4 real number parameters.

From preliminary studies, it seems that the algorithm reaches a good optimal at the precision of speed equal to 0.05 km/h and direction equal to 0.5 degree, and then it did not show any more obvious improvement. So, the *move step* schedule was shown in table 4.5 where the first column indicates the different distance values that have been used as threshold in order to change the speed move and direction move parameter of the TA. The second and third columns show the new values for the corresponding parameters when the threshold distance has been reached.

Furthermore, *tenure time* was set at 1 (taboo time $\tau = 1$) and the *taboo*

Best distance	Speed move	Direction moves
>100	1.0	10
<100	1.0	5.0
<50	0.5	2.0
<20	0.1	1
<10	0.05	0.5

Table 4.5: Move distance schedule

contra move time was initialized to 3. We have 4 parameters and 8 moves and with these settings we will have a maximum of 4 moves marked as taboo from the moves available due to the short-term taboo. The Taboo list size is equal to 1000 individuals. The list size has to be large because its function is to keep track of the old points that have been visited to prevent the entrance in non-productive loops.

In order to find the effect of the algorithm parameters, we have changed ϕ from 1 to 0.9, as we did in the case of homogeneous wind field. Furthermore, the α parameter (physical penalty factor) has been set 1 and 0.5. Figure 4.14 shows the effect of these parameters on the average number of iterations that the algorithm spends to find the optimal parameter (note: the numbers in the graph are the average of 4 experiments).

We can see that with ϕ equal to 0.9 and a value of alpha (α equal to 1) for some frequency factor and no physical penalty, the number of iterations is the minimum. The maximum number of iterations is achieved when ϕ has a value of 1 and α is set to 0.5 or no frequency is present but a physical penalty is included. Therefore, TA exhibits a different behavior with respect to GA when considering the physical penalty. Whereas this factor does not affect the results provided by GA, in the case of TA one can observe that the presence of physical penalty appears to have a slightly negative effect. With respect to the ϕ factor, we can conclude that the physical penalty done not help in accelerating the convergence, but may helps in giving better quality vectors.

As a conclusion from this set of experiments, we obtain that Taboo Algorithm does not need too much tuning to be productive.

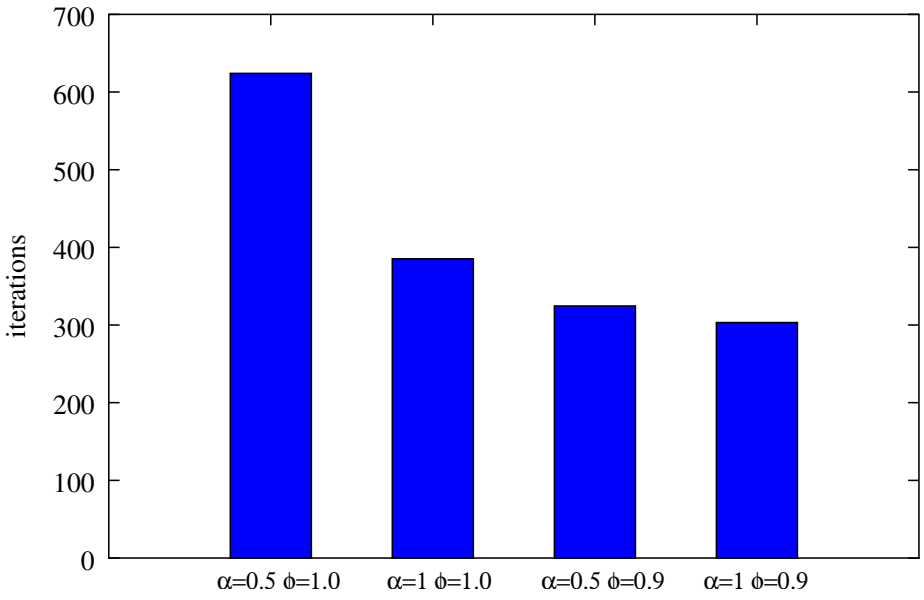

Figure 4.14: Taboo algorithm average number of iterations to reach the optimal solution with respect to physical penalty (α) and frequency (ϕ).

4.5.3 Comparison of Optimization Techniques Using Heterogeneous Wind Fields

In this section, we compare the two optimizations techniques studied in the previous sections (GA and TA) when a parameter vector with four parameters $({w_d}^1, {w_s}^1, {w_d}^2, {w_s}^2)$ is used. For this purpose, we simply consider the best distance obtained throughout all the experiments performed which is shown in table 4.6. Furthermore, in table 4.6 we include the best distance obtained in the case of Random search in order to denote the improvement provided by GA and TA. Table 4.6 also reports the number of evaluations of the objective function that have been needed to achieve such distances. The maximum number of evaluations has been set a priori and it is equal to 20,000 evaluations. From the analysis of the table, we can easily conclude that TA converges faster than GA and, of course, faster than the Random search.

The advantage that TA has over GA may come from the implementation of the move. TA in this implementation does not search the entire search space. Instead it has a limited subset of the search space that GA searches in. TA searches do not have floating-point representation of the parameters as their

Algorithm	Distance	N° of evaluations
Random	97,5	20'000
Genetic	17,6	20'000
Taboo	8,95	2596

Table 4.6: Comparison between genetic, taboo and random search using smooth heterogeneous wind field

domain but GA do. In addition TA uses an adaptive move distance which makes convergence even better.

The following section experiments the effect of adding more parameters and complicates the fire line produces on the number of evaluations of the objective function when GA and TA are used.

4.5.4 Algorithm Scalability

In this section, we aim to analyze the robustness of the algorithms and the effect of assigning more parameters to be tuned on the quality of the solution. For this purpose, we have used the same fire line as in the previous sections (smooth heterogeneous wind field).

In table 4.7, we summarize the mean values obtained for all the experiments for the different sizes of the parameter vector we have used (2, 4 and 6). The first two columns show the results of the experiment with the simple fire line and the other columns show the results of the experiment with the smooth heterogeneous wind field for 4 and 6 parameters, respectively.

If we analyze what happens in the case of Genetic Algorithm, we can observe that the number of evaluations increased from 200 to 20,000, which means an increment of 100 times in the number of evaluations. However, in the case of Taboo Algorithm, the number of evaluations has increased from 218 to 2'596 which implies an approximate increment of 12 times. Furthermore, when we change the number of parameters from 4 to 6 parameters using the same real fire line and the same number of evaluations of objective function, we do not reach better distance, on the contrary, the obtained distance increases.

Using 4 parameters, we reach a smaller distance in the same number of iterations, which means that when the number of parameters increases, the number of iterations needed to find the optimal solution increases as well. Therefore, we suggest using smaller number of parameters to find the optimal and, subse-

	Simple			Heterogeneous		
Param.	2		4		6	
	Iter.	distance	Iter.	distance	Iter.	distance
Genetic	200	11	20'000	18.8	20'000	84.75
Taboo	218	0	2'596	8.95	15'648	48.43

Table 4.7: Comparison between TA and GA with respect to the number of parameters

quently, apply the optimal vector to create extended vectors with more parameters. Afterward, these enlarged vectors can be tuned if we need more accurate vectors. In other words, it is not recommended to start from an ignorant state using the full size of the vector we want to tune. It seems to be better to initiate the optimization process using a few rough parameters and, when we are in the neighborhood of the optimal solution, we can enlarge the size of the vector in order to reach more accurate solutions. We can consider the values optioned in the first stage as the starting values of the parameters when the vector is enlarged. For instance, in the case of using 2 time steps where the first step is 30 minutes and the second is 15 minutes, these could be used as the first stage of the optimization process. Afterward, once we have found the optimal values for the two time steps process, we could then divide the first step into two steps of 15 minutes each and consider the number we have optioned for the time period of 30 minutes as the same for the two new smaller steps. Subsequently, we could proceed with the optimization process starting with this new vector.

After these results, some new questions arise: "What happens when the real fire line becomes more complicated and the wind field changes more? What is the effect of the real fire line on the performance of the algorithms? " The following section tries to answer these questions.

4.6 Rough Heterogeneous Wind Field

The experiments reported in this section have been performed for the tuned algorithms (GA and TA) and considering a different real fire line. We have changed the fire line that has been created by using a smooth wind field for one that has been created using a rough wind field with a space factor (sp) equal to 0.02 and time factor (tim) 0.1 (see creating real fire line section 4.2.1).

Table 4.8 shows the results, on average, obtained for both experiments

Wind Field	Smooth		Rough	
	Distance	Evaluations	Distance	Evaluations
Genetic	18,8	20'000	47,75	20'000
Taboo	8,95	2'596	25	1'830

Table 4.8: Comparison between TA and GA with respect to changing a real fire line using different wind fields and 4 parameters.

(smooth and rough wind field) by maintaining the same settings of the algorithms.

We can consider the wind as fixed for more time and larger terrain cell in the case of a smooth wind field than in the case of a rough wind field. Therefore, we need to divide the time and/or space more to reach the same distance as we did with smoother wind field. We expect that using a rough wind field gives an objective function that has a large distance global minimum, in other words, the absolutely optimum vector with 4 parameters has an objective function shifted from zero with more distance than the one calculated against smooth wind field.

The distance using Genetic Algorithm increased from 18.8 to 47.75 and in the case of Taboo from 8.95 to 25 which is almost the same ratio of increase or the same shift, due to the shift of the objective function.

We notice that the number of iterations of taboo is smaller. This result can be explained because we collect the average of 4 tests and the starting point is random which means that chance can influence the results. In order to eliminate this factor, we need to make more tests.

4.7 Comparing Modern Heuristic Techniques

In this section, we extend the optimization process to include all the input parameters required by the ISS simulator. As in the previous experiments, a synthetic real fire line was obtained. Once this synthetic fire line had been obtained, it was dismissed and was only used as a comparative member to calculate the objective function. Table 4.9 summarizes the input parameters that are required by the simulator and optimized in this study.

The Genetic algorithm, which was tuned in the previous sections, was used in this experiment. The Simulated Annealing algorithm have two important parameters that can be guessed easily,T_0 and T_n . Simulated Annealing does not need extensive tuning and it was not introduced earlier in the study. These

Param. Name	Meaning	Units	Typical value	Min. value	Max. value
w_0	Fuel loading	Kg/m2	–	0.3	4
β	Compactness factor	–	–	0.005	0.12
σ	Fuel particle area-to-volume ratio	1/m	–	315	11500
St	Fuel particle total mineral content	–	0.0555	0.0001	0.09
Se	Fuel particle total mineral contents except silica	–	0.01	0.0001	0.07
Mx	Moisture of Extinction	–	–	0.1	0.4
M_f	Humidity contents	–	–	0	Mx
h	Fuel particle low heat content	MJ/kg	20.3	19.5	24.0
U	wind speed	Km/h	5	0	20

Table 4.9: Typical and bounding values for ISS input parameters

two parameters ware set to 1000 and 10, respectively. From our experience with Taboo algorithm, the taboo moves need to be no more than half of the moves available in each iteration. The algorithm do not perform well when the taboo tenure is more than one. Therefore taboo tenure was set to 1 and taboo tenure contra move was set to 5, so that we have 6 moves marked as taboo out of 20. The frequency factor ϕ was set to 0.9, as the tuning experiment shows that this was the optimal.

In this study, the optimization process has been iterated 1000 times and, furthermore, since the initial set of guesses for all strategies was obtained in a random way, the global optimization process was performed 10 times. Therefore, all the values reported in this study correspond to the means values of the corresponding 10 different experiments conducted. In figure 4.15, we show the evolution of the Hausdorff distance while optimization is iterating. We can observe that SA,TS and GA exhibit a similar behavior, whereas the pure random has a different convergence speed pattern. If we analyzed each individual optimization technique more carefully, we would also observe that TS converges faster at the beginning, but, after 200 iterations, provides almost the same results as SA. However, after 300 iterations, both exhibit a particularly tight behavior.

We can also observe that GA exhibits a slight convergence speed at the

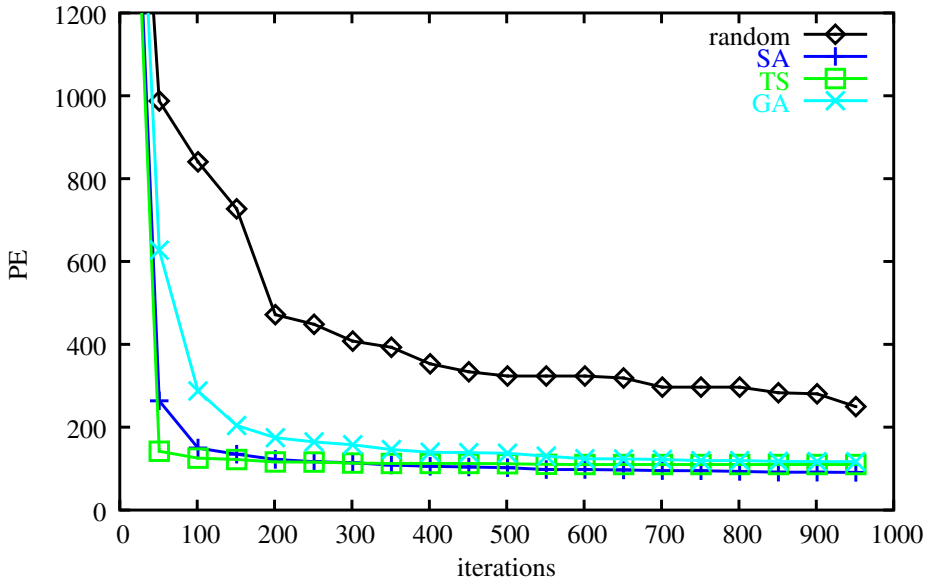

Figure 4.15: Hausdorff distance throughout the optimization process

iteration nº	Random	SA	TS	GA
100	840.5	149.7	125.3	287.5
500	323.4	102.3	111.8	137.2
1000	249.1	91.0	109.9	115.1

Table 4.10: Hausdorff distance (m) for RS, SA, TS and GA optimization algorithms.

beginning of the optimization process compared to TS and SA, which may stem from the difficulty of tuning the quantity of parameters involved. However, this feature is overcome as the iterations progress. Table 4.10 shows, in greater detail, the exact Hausdorff distance, on average, at iteration numbers 100, 500 and 1000. As we can observe, the Taboo Search practically freezes after almost 500 iterations and shows a slow development after 100 iterations. The value of the distance at iteration 100 was 125.3 m, and at iteration 527, the value was 109.9m, which is the same as at the end of the optimization process (iteration number 1000).

To summarize, The experiments were carried out using different fire scenarios. The results show that Simulated Annealing, Taboo Search and Genetic

Algorithm provide similar patterns. At the beginning of the process, the Taboo search algorithm provides better configuration, but, after a certain number of iterations, results were improved by Simulated Annealing.

4.8 Chapter Conclusion

The implemented optimization techniques described in section 3.3, have been tuned and tested to optimize real life objective function that consists of forest fire simulator and a prediction error. We started with a simplified problem by tuning two parameters for a simple synthetic fire line and we then complicated the problem to reach a full simulator input parameters. Classical optimization techniques get out from the beginning of the race. However, modern heuristic evolutionary techniques shows robustness against the changing and complication of the problem. Some techniques show small advantages in certain cases. But in general, all the modern heuristic evolutionary techniques (genetic, taboo and simulated annealing) shows the ability to solve the problem.

Chapter 5

Accelerating Optimization Convergence

"han de poder poco sus malas artes
contra la bondad de mi espada"

Miguel de Cervantes

As mentioned in chapter 3, parallelizing the method will reduce the time needed to execute every iteration. In this chapter, we will develop methods to reduce the number of iterations necessary to reach a "good" solution. In particular, this technique depends on reducing the search space. Since the number of parameters is quite large, the resulting search space becomes enormous, consequently, to assess the whole search space is not feasible. In order to reduce the underlying search space, we propose applying a sensitivity analysis to the input parameters. The sensitivity analysis will 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.

5.1 Reducing the Search Space

To reduce the search space we propose the following three techniques: reducing the problem dimensionality: searching in reduced ranges and sampling the

search space. All three techniques needs a sensitivity index to reduce the search space in a rational way. In the following we shall briefly describe how each one of these techniques work, and, then, we will concentrate on how the sensitivity analysis has been carried out inorder to determine the sensitivity index of each input parameter.

5.1.1 Fixing Some Parameters to their Nominal Values

Fixing some parameters to their nominal values will reduce the dimensionality of the search space which will reduce the search space. Recalling the example of section 2.2.2 where we need to optimize 10 real parameters on a machine that guarantees 7 decimal digits of accuracy, if we reduce the number of parameters from 10 to 9 the search space will be reduced from 1E70 to 1E63 and, consequently, reducing the number of parameters to 8, 7 and 6 parameters will reduce the search space to 1E56, 1E49 and 1E42 possible combinations, respectively.

Choosing the parameters to be fixed is not an arbitrary process. If we fix some parameters that have significant effect on the result to bad values, will deviate the optimization process from the good optimums. But fixing the parameters with less significant effect will not affect finding good solutions and at the same time will reduce the search space.

5.1.2 Introducing a Certain Degree of Knowledge

In order to be more effective in tuning the most sensitive parameters, we also propose introducing a certain degree of knowledge during the optimization process. This knowledge will consist of limiting the range of the tuned parameters around an estimated value (which may be the real measurement if any, bearing in mind that all measurement instruments have a degree of error) for those parameters.

5.1.3 Sampling the Search Space

The other possible way to accelerate the convergence of the optimization is sampling the search space. What we mean by sampling is to select some discrete values from the range of possible values of the parameter to be used by the optimizer. The values that will be examined cannot be any value within the range of possible values of the parameter that permits the floating-point representation but will be a discrete subset of them. Once again, with the same

example of 10 parameters and machine of 7 decimal digits, instead of using all the parameters with the full machine accuracy (of 7 digits), we reduce some of them to 4 digits - let us imagenimagine a situation where 3 parameters have the accuracy guarantee of 4 decimal digits and the rest will remain with 7 decimal digits of accuracy- the original search space of 1E70 will be reduced to 1E61. It is expected that sampling the parameters may accelerate the convergence of the optimizing technique but, at the same time, we may lose the optimal solution. A good balance between sampling and arriving at a reasonable minimum can be reached by decreasing the sampling frequency of the parameters that has little effect on the result, at the same time using a rather high sampling frequency of the parameters that have little effect on the result, and by using a rather high sampling frequency for the parameters with higher effects. The parameters that have a greater sensitivity index need to have more sampling frequency because by definition a little change in the parameter with great sensitivity index has a big effect on the result, so sampling frequency needs to be correlated with the sensitivity index. The distribution of the points across the axes of the parameters of more sensitivity need to be more dense than the distribution of points in the axes of parameters with less sensitivity.

In essence, the three above-proposed techniques to accelerate the convergence of the optimization process, are based on knowledge about the sensitivity of each involved input parameters. Therefore, in the following section, we will outline different strategies to deal with the sensitivity analysis.

5.2 Sensitivity Analysis

Sensitivity Analysis (SA) classically aims to ascertain how a model/simulator depends upon the information fed into it (input model/simulator parameters). The objective of any sensitivity analysis is to identify the most important factor among all inputs, which will be defined as the input that, if determined (i.e. fixed to its true although unknown value), would lead to the greatest reduction in the variance of the model/simulator output. Likewise, we can define the second most important factor, and so on, until all factors are ranked in order of importance [99].

Sensitivity methods can be classified as: mathematical, statistical and graphical. In the following, we briefly describe each of these methods.

Mathematical Mathematical sensitivity methods assess the sensitivity of a model output to the range of variation for an input. These methods typically involve the output evaluation for a few values of an input, representing the possible range of the input. Such methods do not address variance in the output due to the variance in the input, but they can assess the impact on the output of range of variation in input values. In some cases, mathematical methods can be helpful in screening the most important inputs. These methods also can be used for verification and validation, as well as in identifying inputs that require further data acquisition or research [38].

Statistical Statistical methods involve running simulations in which inputs are assigned probability distributions and processing the effect of variance in inputs on the output distribution. Depending upon the method, one or more inputs are varied at a time. Statistical methods allow us to identify the effect of interactions among multiple inputs [61].

Graphical Graphical methods give representation of sensitivity in the form of graphs, charts, or surfaces. Generally, graphical methods are used to give a visual indication of how an output is affected by variation in inputs. Graphical methods can be used as a screening method before further analysis of a model or to represent complex dependencies between inputs and outputs. Graphical methods can be used to complement the results of mathematical and statistical methods for better representation.

5.2.1 Sensitivity Analysis on the Enhanced Prediction Approach.

A suitable method in our case is a method based on nominal range sensitivity analysis, which is also known as local sensitivity analysis or threshold analysis [99]. This method is applicable to deterministic models. It is usually not used for probabilistic analysis. One use of nominal sensitivity analysis is as a screening analysis to identify the most important input to propagate through a model in a probabilistic framework. Nominal range sensitivity can be used to prioritize data collection needs.

Basic nominal sensitivity analysis evaluates the effect on the model output exerted by individually varying only one of the model inputs across its entire range of possible values, while holding all other inputs at their nominal or base-

case values. The difference in the model output due to the change in the input variable is referred to as the sensitivity or swing weight of the model to that particular input variable, in that given case.

However, there may be interdependencies among the parameters. Therefore, the effect of one parameter may depend on the values of the fixed parameters. The model can be very sensitive for one parameter when the other parameters have certain values, and not so sensitive in other values. To have a global view of the sensitivity of one parameter we need to examine its sensitivity in various representative cases. We define the "case" here as a certain values setting of all the parameters except the parameter we are analyzing. For example if we are analyzing the wind, we define several cases in which the fire can occur and the model sensitivity to the wind might change. Consider, for instance, the case of very large surface-to-volume ration (e.g. straw) and the compliment case of low surface-to-volume ration (e.g. woody fuel) in these two cases, we observe different wind sensitivity. The wind has greater effect in the first case. So we cannot depend on one case to calculate the sensitivity of a parameter. Therefore, we create all the possible cases and calculate the nominal sensitivity for all the cases.

All the possible cases in the fire propagation models can be defined by combining all the combination of minimum and maximum values of the parameters. Suppose we have 10 parameters and we want to calculate the sensitivity of the first parameter, we need to construct all the possible cases then calculate the nominal sensitivity of the first parameter for all the cases. Setting the rest of the parameters at their minimum is considered as one case, setting all the parameters at their minimum except one, we set it to its maximum, is considered as another case, and so on for all the parameters. Following this approach we will have 2^9 different cases. The nominal sensitivity of the first parameter needs to be calculated in 2^{n-1} cases where n is the number of the parameters of the model.

In the particular case of fire propagation, the number of parameters is quite high and the number of combinations that must be evaluated in order to reach the sensitivity index is enormous.

5.3 Experimental Study

The aim of the experiments in this section consist of investigating the ways of reducing the time necessary to execute the enhanced prediction method described in chapter 2. As mentioned in the beginning of this chapter, we can accelerate the optimization convergence by reducing the search space. In this section, we focus on the experimental study developed to reduce the search space. We will start with calculating the sensitivity index needed by the tree proposed methods.

5.3.1 Calculating the Sensitivity Index

The sensitivity of the parameters, in our case, depends on the fire propagation model used in the core of the objective function. For a generic study, we studied the effect of the parameters of the model in one dimension. In particular, we focused on the propagation speed, thus the wind had only one scalar value, which is the speed of the wind in the direction of the fire propagation. To calculate the sensitivity index for each parameter, it is necessary to define a minimum and maximum value for the parameter. These values are typically obtained from field and lab measurements. For all the possible combinations of the other parameters, therefore, two simulations are executed considering the minimum and the maximum value of the parameter currently studied. The speed difference between both propagation simulations represents the effect of changing that particular parameter from its minimum to its maximum for that particular combination of the other parameters. Let V_{ik} be the effect of varying factor i from its minimum to its maximum (difference of the speed of the minimum and the speed of the maximum) at case k.

The total effect of parameter i is defined as the addition of the effect of each possible case:

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

where k is all the possible cases (combinations of input factors).
Thus, V_i will be our index of sensitivity for the parameter i. This index not only reflects the effect of the parameter, but also the effect of its range. Higher parameter ranges mean greater uncertainty in the measurement of that parameter.

parameter	Min	Max	Index
W_0	0,1	4	0,77
β	0,01	0,11	0,86
σ	315	11500	0,56
S_t	0,001	0,08	0,03
S_e	0,0001	0,07	0,16
M_x	0,1	0,4	0,28
M_f	0	Mx	0,61
h	18571429	22000000	0,13
U	0	15	0,71

Table 5.1: Ranges used to calculate the sensitivity index.

Parameter	Index
β	0,86
W_0	0,77
U	0,71
M_f	0,61
σ	0,56
M_x	0,28
S_e	0,16
h	0,13
S_t	0,03

Table 5.2: Parameters ordered by sensitivity index

As we have previously commented, for simulation purposes, we used the ISS forest-fire simulator, which incorporates the Rothermel fire spread model [96]. Therefore, the Rothermel model input parameters will conform the vector to be optimized in our case. Table 5.1 outlines each one of these parameters and their corresponding minimum and maximum values according to [7], also showing the calculated index. Using the value of the index, we can classify the input parameters by their sensitivity, as shown in table 5.2. This table shows that the two most important parameters are the load parameters (W_0,β); the third is wind (U), followed by humidity (M_f). The parameters with weakest effect are metal content (S_t, S_e) and heating content (h). This result concords with the results obtained by [98], which also uses the Rothermel set of equations as a forest fire propagation model.

Since sensitivity analysis implies a high number of simulations, we have

also used the master/worker programming paradigm to evaluate all sensitivity indexes. The master prepares the set of parameters that need to be evaluated and keeps the resulting values; then it calculates the index. The worker executes the simulator and calculates the corresponding spread speed. After that, it sends back the result to the master. After the master prepares the set of parameters that need to be evaluated. The evaluation process can be executed in parallel without any need for communication, except sending the results.

5.3.2 Reducing Problem Dimensionality

Considering the definition of the sensitivity index, if we were able to find the real value of the parameters with greatest importance, we would minimize the divergence of the simulator from the reality. Therefore, it is crucial to calibrate the parameters that have greater sensitivity index, while we do not know their real values. Likewise, we can say that calibrating the parameters that have little effect on the results will not significantly improve the simulator results, and this will consume processing time. The impression is therefore created that it is not worth tuning the parameters with small sensitivity index. If we have fewer parameters to be optimized, the process will converge faster and, at the same time, fixing certain unimportant parameters to a given value with a reasonable error will not deviate the optimization process too far from the global minimum.

However, it is evident that if we need more accuracy, we then need to tune all the parameters, regardless of their importance.

This experiment is designed to observe the effect of fixing the parameters that have a small sensitivity index on the convergence of the optimization process. The real values plus 10% of their full range is used as estimated values for the parameters to be fixed. This percentage of variation allows us to experiment the effect of having an error of 10% in the estimation of the parameter. Table 5.3 shows the real value of the less sensitive parameters, and their corresponding estimated values, when applying this estimation error (10%).

Figure 6.3 shows the convergence of the optimizing process by reducing the number of optimized parameters. Each curve differs from the other by omitting one parameter each time, i.e., the curve labeled '10 parameters' shows the convergence of the tuning process when all parameters are considered. The curve labeled '9 parameters' plots the convergence evolution when tuning all the parameters except that with a smaller sensitivity index ('S_t' in this case), and so on.

Parameter	Real value	Estimated value
S_t	0.04	0.04799
h	18971429	19314270
S_e	0.02	0.0269
M_x	0.3	0.33

Table 5.3: The real and estimated values of the fixed parameters.

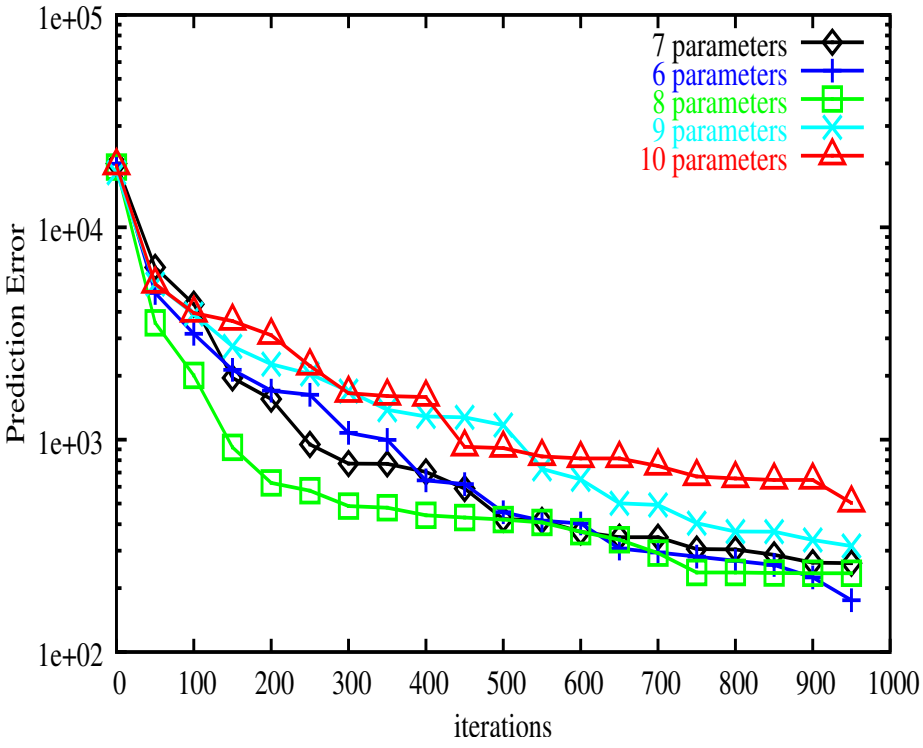

Figure 5.1: Optimization Convergence Changing the Number of Parameters

As we can observe, we can clearly distinguish two different phases: one phase from the beginning of the optimization process up to iteration 500, approximately, and a second phase, which goes from iteration 500 until the end. During the first phase, independently of the number of parameters optimized, we clearly observe fast convergence and, in particular, the case where two parameters are fixed (the curve labeled '8 parameters') has a superior optimization performance. In contrast, in the second phase, the convergence speed for all cases seems to stabilize.

However, since GAs may have a warm-up phase (i.e. starting iterations before the convergence), we applied statistical hypothesis testing [107] to the results in order to asses whether or not the observed behavior can be considered statistically different. We can observe in figure 5.2 that at iteration 500, in general, for each case (fixing a certain number of parameters) the corresponding mean value lies inside the confidence intervals of the other cases. That means that there is no statistical evidence that the differences are relevant between the means due to fixing some parameters at that iteration. However, figure 5.3 shows that the means have a trend to diverge one from the other at iteration 1000. In the same way we have applied statistical hypotheses testing for iterations 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. We found that there is no statistical difference between the means before iteration 500; consequently, it is irrelevant to discus the behavior of the curves during the first phase of the optimization process.

However, at iteration 1000, the results show a statistical difference between optimizing all parameters as opposed to fixing 1, 2, 3 and 4 parameters (figure 5.4) because the mean value for the case of '10 parameters' clearly lays out of the other confidence intervals. Furthermore, we can observe that, as the number of fixed parameters increases, a statistical difference also appears (i.e., 8 and 9 versus 6 and 7). The mean values of the objective function (prediction error) at the end of the optimization process (iteration 1000) is shown in figure 5.3. As we can see, the objective function for the case of '6 parameters' is one third of the mean value obtained for the case of '10 parameters'. Therefore, since the statistical study has shown that the convergence improvement for the case of '6 parameters' is not a matter of chance, we can conclude that reducing the search space is a good policy to speedup the optimization convergence.

Table 5.4 shows the search space reduction for cases plotted in figure 5.3. As we can observe, the search space reached a considerable size reduction from the '10 parameters' case to that of '6 parameters' case. This reduction is directly

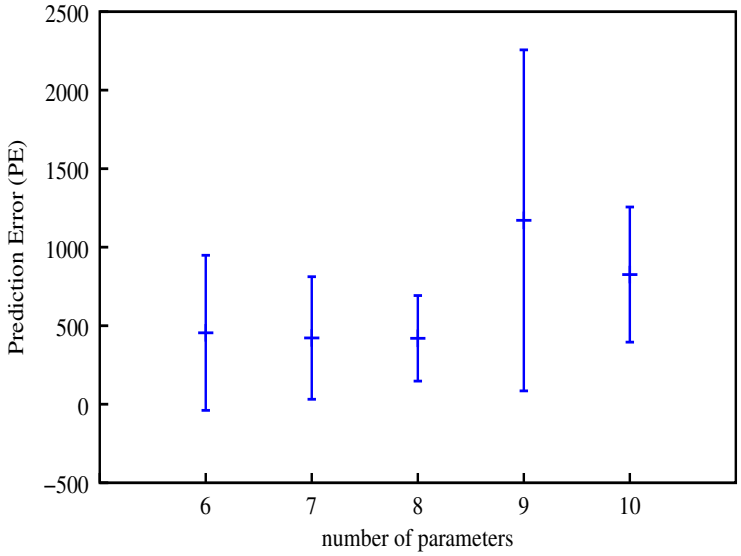

Figure 5.2: 95% confidence intervals at iteration 500.

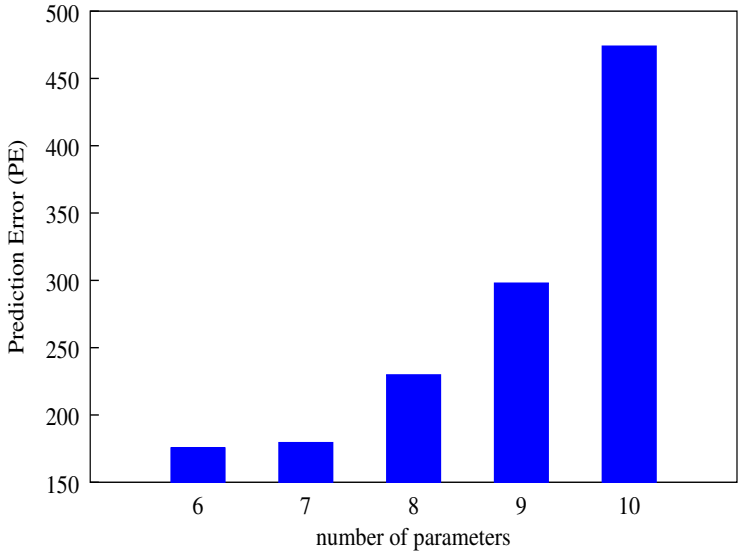

Figure 5.3: mean value of the prediction error (PE) at iteration 1000

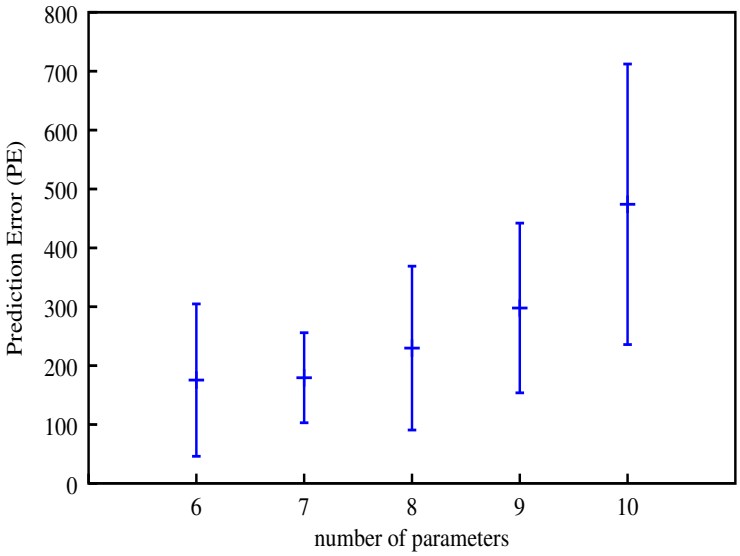

Figure 5.4: 95% confidence intervals at iteration 1000.

N° parameters	Search space size
10	1.0E+77
9	1.0E+69
8	1.0E+61
7	1.0E+53
6	1.0E+45

Table 5.4: Search space size for different number of parameters.

related to the improvement obtained in the objective function.

However, we should bear in mind that, these results are obtained using an error of estimation equal to 10%. If the error is greater, the practice of fixing the value of the parameters to estimated values will not be good. This method therefore assumes a "good" estimation of the real parameter value.

5.3.3 The Effect of Limiting the Parameters Ranges.

Once we have observed that fixing 4 parameters to a certain estimated value provides a considerable improvement in optimization convergence, we focus on this case to introduce a certain degree of knowledge of the optimized parameters in order to further improve such convergence. We assume that we have some

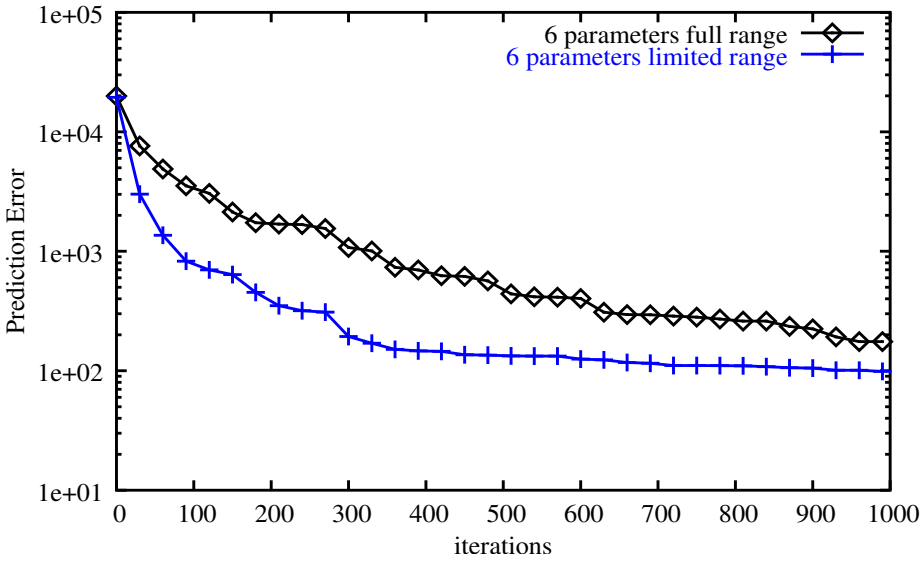

Figure 5.5: Optimization convergence comparison using both the full and limited ranges.

knowledge about the limits within which a parameter can vary, therefore it is not necessary to search within its full possible range. For the purpose of this experiment, we limited the range of the parameter to 15% above and below its "known value", so as to simulate the expected range. Figure 5.5 shows the optimization convergence when optimizing 6 parameters using either their full range or a limited searching range. As we can observe, cutting the range of the parameters significantly accelerates optimization convergence. Although from the figure it seems that, at iteration 1000, both situations provide similar results, the limited range at the end of the optimization process provides an objective function (prediction error) equal to 98.71, on average, whereas the final value is 175.47, using the full range.

5.3.4 Sampling the Search Space.

In section 5.3.1, we ranked input parameters according to their sensitivity index. The parameters that have a greater sensitivity index need to have high sampling frequency since, by definition, a small change in a certain parameter with a great sensitivity index has a large effect on the result. Thus, the sampling frequency of all parameters that are going to be tested, needs to be correlated

with their sensitivity index. The distribution of the points across the axes of the parameters of greater sensitivity need to be denser than the distribution of points in the axes of parameters having less sensitivity.

To establish a relation between the sensitivity index and the sampling frequency of the parameter, we did the following: we want to associate the parameter with the greater sensitivity index to the highest sampling frequency that we want to use, and associate the parameter with lowest sensitivity to the lowest sampling frequency that we want to use. The relation between the sampling frequency and the sensitivity index is therefore as follows:

$$F_i = F_{min} + \frac{F_{max} - F_{min}}{I_{max} - I_{min}} I_i$$

where :

F_i is the number of selected points (sampling frequency) in the feasible range of parameter i.

I_i sensitivity index of i

I_{min}, I_{max} minimum and maximum index of all the parameters.

F_{min}, F_{max} minimum and maximum sampling frequency.

In this experiment, the value selected as F_{min} is 1 and as a maximum sampling frequency (F_{max}), we tested several cases. The obtained values is used as a power of 10, so that the lowest sampling frequency selects 10 values from the range of feasible values, and the maximum will be 1E+6 and 1E+4. Table 5.5 demonstrates the effect of applying the formula in different parameters and different sampling frequencies.

Since we are dealing with a well determined search problem, by using the value of that table, we can easily obtain the size of the whole search space. Using the sampling technique with the sensitivity index reduces the search space and, at the same time, uses a large sampling frequency for the parameters with a large sensitivity index. The resulting search space for the float representation was calculated for this specific problem as 1e+77, however the search space when F_{max}equals to $1e+4$ is $1e+28$ and for F_{max}equals to $1e+6$ is $1e+39$.

In order to evaluate improvement in the optimization convergence using the proposed Sampling Input Parameters technique, we repeated the same optimization process using sampling frequency of 1e+4 , 1e+6 and using floating

parameter	$F_{max} = 1e+6$	$F_{max} = 1e+4$
W_0	1e+6	1e+4
β	1e+6	1e+4
σ	1e+4	1e+3
S_t	1e+1	1e+1
S_e	1e+2	1e+2
M_x	1e+3	1e+2
M_f	1e+5	1e+3
h	1e+2	1e+1
U	1e+5	1e+4

Table 5.5: The result of applying the (SIP) formula on different parameters

point representation. Figure 5.6 plots the optimization process convergence using the normal floating point representation for the optimized parameters and using the sampling technique representation with frequency 1E+6 and 1E+4. We can observe that actually before iteration 500 all the curves have almost the same convergence pattern, but after iteration 500, we can observe that the curve that represents the floating point representation convergence slower than the other two curves. At iteration 1000, we can clearly observe that sampling with frequency of 1E+4 reaches better solutions and converges faster (figure 5.7). We can see that a clear trend for faster convergence by decreasing sampling frequency. We can conclude that using less sampling frequency helps in accelerating the optimization process. This is because the technique reduces the search space, which makes it easier for the optimizer to reach minimum, as there are fewer combinations to examine. At the same time, the points that participate in the process are distributed throughout the search space. However, using such a technique has the risk of missing the global minimum. We cannot guarantee that the global minimum is one of the values selected. Even when using a computer representation of real values, we do not guarantee the existence of the mathematical real minimum, as the computers' representation of real values is limited to a certain precision. Furthermore, the use of sampling is more likely to deviate from the global minimum.

5.4 Chapter Conclusions

In this chapter we have described three ways to accelerate the convergence of the optimization process. 1) Fixing some parameters to their nominal values

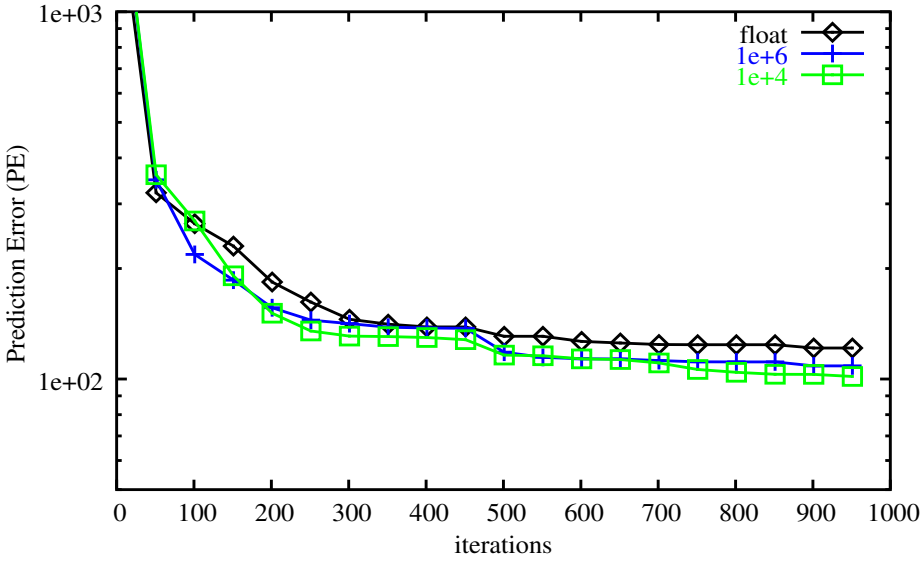

Figure 5.6: The convergence of the optimization using various representations (float and sampling with 1E+6 , 1E+4)

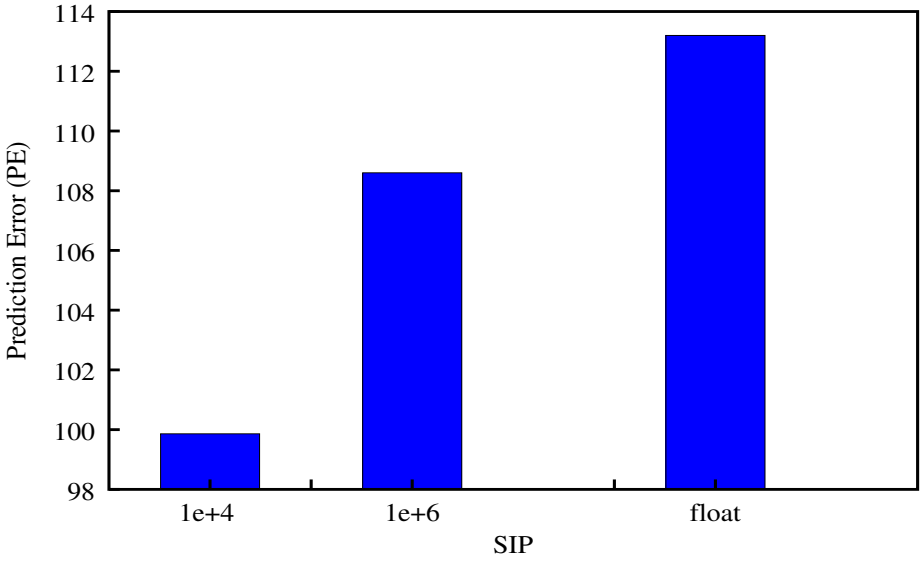

Figure 5.7: Prediction Error after 1000 iterations with float and sampling frequency of 1E+6 and 1E+4.

2) Introducing a certain degree of knowledge 3) Sampling the search space. All these techniques depends on calculating a sensitivity index to rationally reduce the search space. We have shown in the experimental part in this chapter that the proposed techniques are useful to accelerate the optimization process and, therefore they will be useful in the case of real-time constraint.

Chapter 6

Applying the Methodology on Real Cases

"-Dios lo haga como puede -respondió Sancho Panza"

Miguel de Cervantes

In this chapter, we will apply the methodology in real life cases. We illustrate some experiments using fire lines extracted from real laboratory fires, wich allow us to compare the classical and enhanced prediction methods. We will start the chapter with a discussion of the experiments' platform and methodology, followed by the analysis of each individual case.

6.1 Experiment Platform

The proposed fire-prediction model has been studied using a purposely-built experimental device (picture 6.1). It is composed of a burn table of 3x3 m^2 that can be inclined at any desired angle (slope) and by a group of fans that can produce a horizontal flow above the table with an arbitrary velocity. In order to gather as much information as possible about fire-spread behavior, an infrared camera recorded the complete evolution of the fire. Subsequently, the obtained video was analyzed and several images were extracted, from which the corresponding fire contours were obtained.

Figure 6.1: Experiment table (ADAI, Portugal)

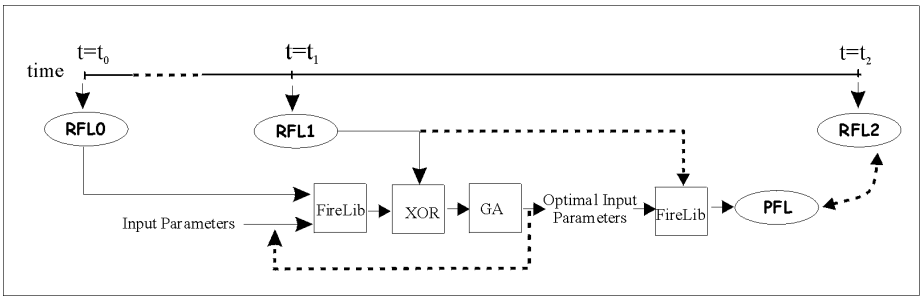

Figure 6.2: Enhanced wild-land fire prediction method

Subsequently, we will briefly describe the FS (fire simulator), input parameters and PE (prediction error) component of the enhanced-prediction method depicted in figure 6.2 for our particular case. The Optimization strategy used in this framework was a Genetic Algorithm (GA).

6.1.1 Fire Simulator Used to Make the Prediction.

For wild-land fire simulation purpose, we have used the wild-land simulator proposed by Collin D. Bevins, which is based on the Fire-Lib library. Fire-Lib is a library that encapsulates the BEHAVE fire behavior algorithm. 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 is to be burnt and at what time the fire will reach the burnt cells. As inputs, this simulator accepts the maps of the terrain, the vegetation characteristics, the wind and the initial ignition map. The output generated by the simulator consists of the map of the terrain where each cells is tagged with its ignition time. In our experiment, we divided the terrain (burn table) into 40x40 cells, and 80x80 cells.

As previously commented, any fire-spread simulator needs to be fed with certain input parameters. Since our experimental fire was carried out under determined conditions (laboratory conditions), we have fairly good estimation values for the input parameters. In the next section, we will describe how we estimate the parameters.

parameter	value	unit
fuel bed depth	1	feet
dead fuel extinction moisture contents	0.12	lb water / lb fuel
load	0.034	lb fuel / sq ft bed
area-to-volume	3500	sq ft fuel / cu ft fuel
density	32	
total silica content	0.05555	lb silica / lb fuel
effective silica content	0.01	lb silica / lb fuel
low heat of combustion	8000	BTU/lb fuel

Table 6.1: model 1 in Rothermel classification

6.1.2 Input Parameters Estimation

The conducted experiments have been carried out using two types of fuel: straw and maritime pine ("pinus pinaster"). Both fuels are considered to have the characteristics of model number 1 in the Rothermel models classifications, which have the values described in the table 6.1. We have used these values as input values for the simulator to simulate the classical prediction method.

Sensitivity studies show that the Rothermel model has from weak to very weak sensitivity against density, total silica content, effective silica content and low heat of combustion. For this reason, we conducted the following experiment to determine the effect of not including these parameters on the convergence speed and to decide whether to include these parameters or not.

We have used the fire line that has been obtained from a table experiment as one to be compared with the simulator output. The fuel used was maritime pine ("pinus pinaster") and the table dimensions was 3x3 m^2. There was no artificial wind and the slope of the table was 30^o degrees. The real fire line is the contour of the fire at the time 1.5 minutes. The optimization process was repeated 10 times for each case and the average of these repetitions is plotted in the curve.

Figure 6.3 shows the convergence of the optimizing process by reducing the number of optimized parameters. The curve labeled 13 parameters represent the optimizing of all the parameters in the table 6.2 and the other (labeled 10 parameters) represents the convergence of the optimization process when fixing the parameters that appears in the table 6.3 to their nominal values. The same experiment was repeated with the replacement of the real fire line to that obtained in the same way described above, but with the time of burning from

Parameter	Unit
Loading	lb fuel / sq ft bed
1-hr dead fuel moisture	lb water / lb fuel
10-hr dead fuel moisture	lb water / lb fuel
100-hr dead fuel moisture	lb water / lb fuel
Live herbaceous fuel moisture	lb water / lb fuel
Surface area-to-volume ratio	sq ft fuel / cu ft fuel
Fuel bed depth	feet
dead fuel extension moisture content	lb water / lb fuel
Low heat of combustion (h)	BTU / lb fuel
Total silica content (St)	lb silica / lb fuel
Effective silica content(Se)	lb silica / lb fuel
Wind speed(U)	feet / min
Direction of wind heading	degrees clockwise from north

Table 6.2: Simulator input parameters

Parameter	Nominal value
St	0.0555 lb silica/lb fuel
Se	0.01 lb silica/lb fuel
h	8000 BTU/lb

Table 6.3: Estimated values of the fixed parameters.

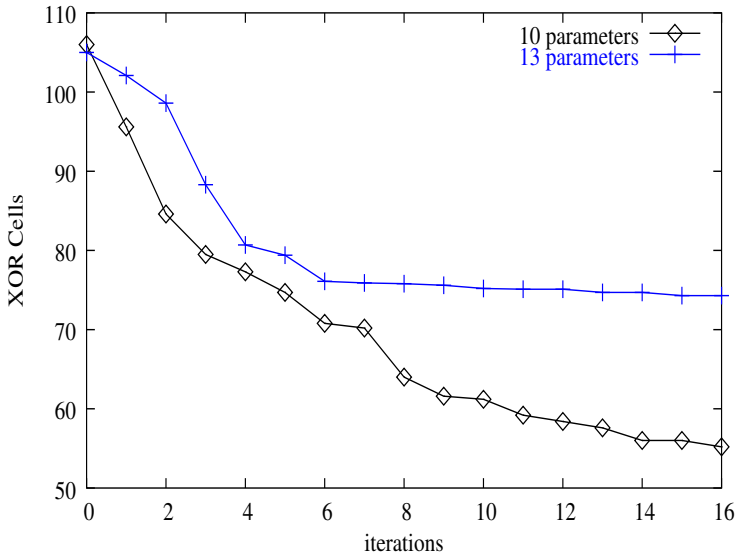

Figure 6.3: Optimization Convergence of all the parameters v.s. fixing some for time of 1.5 minutes

the initial ignition time of 2 minutes and the result is shown in the figure 6.4.

The two figures (6.3 and 6.4) clearly show the tendency of faster optimization convergence when fixing the parameters with low sensitivity index. This behavior is due, on the one hand, to the reduction of the search space when optimizing fewer parameters, and, on the other hand, because the possible error of the fixed value does not have great influence on the result because the fixed parameters have a low sensitivity index. We can conclude that we can exclude the parameters in table 6.3 from the optimization process and use their nominal values and only use the parameters listed in table 6.4 for optimizing.

Finally, humidity values needed by the simulator resulted from measurements just before the fire experiment. The wind speed and direction are calculated from the fans' rotation velocity.

6.1.3 The Prediction Error

As mentioned in section 2, in order to measure the goodness of the predicted fire line, we need to define a prediction error. Taking into account the particular implementation of the simulator used in this experiment, the terrain is treated as a square matrix of terrain cells. We define the prediction error as the number

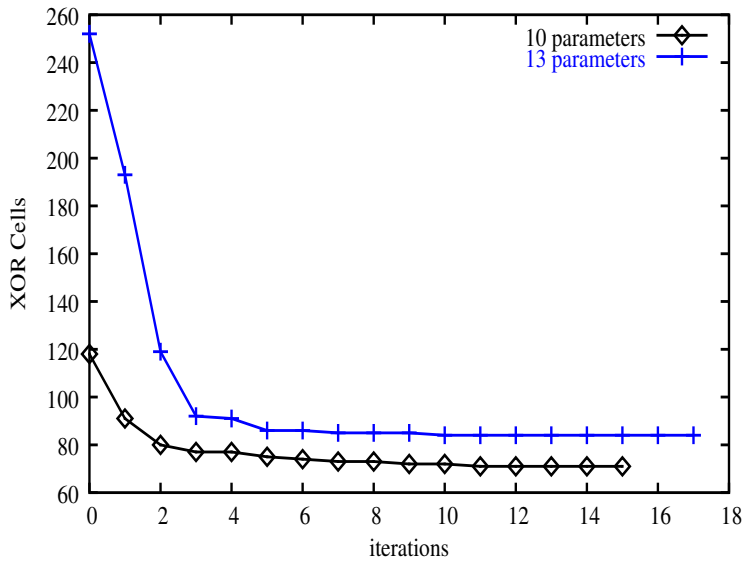

Figure 6.4: Optimization Convergence all the parameters v.s. fixing some for the time of 2 minutes.

parameter	unit
fuel bed depth	feet
dead fuel extinction moisture contents	lb water / lb fuel
load	lb fuel / sq ft bed
area-to-volume	sq ft fuel / cu ft fuel
wind speed	mil/h
wind direction	degree
Moisture 1 hour	lb water / lb fuel
Moisture 10 hours	lb water / lb fuel
Moisture 100 hours	lb water / lb fuel
Live herbaceous fuel moisture	lb water / lb fuel

Table 6.4: Tuned parameters

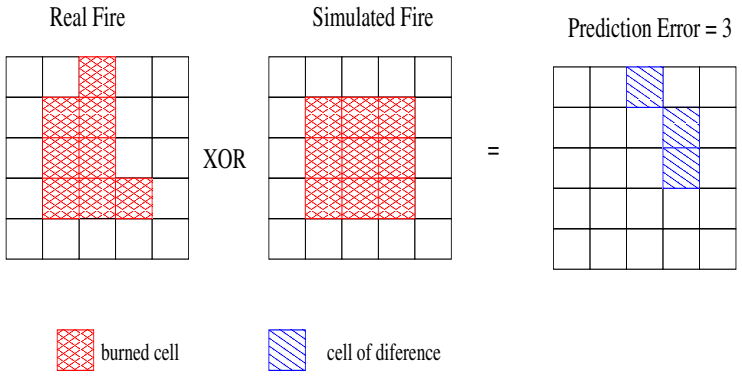

Figure 6.5: Evaluation of the XOR function as a fitness function for a 5x5 cell terrain.

of cells that are burned in the simulation but are not burned in the real fire map, and vice versa. This expression is known as the XOR function. Figure 6.5 shows an example of how this prediction error is evaluated for a 5x5 cell terrain. The obtained value will be referred to as the prediction error that, for this example, is 3.

6.1.4 Speedup Test of the Methodology

As mentioned in chapter 3, we use parallel processing to reduce the execution time of the optimization techniques. We gain parallelism by distributing the objective function evaluation over a farm of workers. In this section we will experiment how much this structure can reduce execution time, and how much it can scale by adding more machines.

As has been shown, in order to provide useful fire spread prediction, it is necessary to work under real-time constrains, which must be accomplished for the proposed enhanced prediction method. For this reason, we have analyzed speedup improvement for the prediction process as the number of processors increases.

The proposed method has been executed on a Linux cluster under an MPI environment. The number of processors used were 1, 2, 4, 8 and 16. Figure 6.6 shows the evolution of the speedup for this particular example. From 4 processors (1 master and 4 workers) we clearly observe a speedup improvement, which continues as the number of processors increases. Figure 6.6 illustrates that by using more machines, we can carry out more iterations in a predetermined

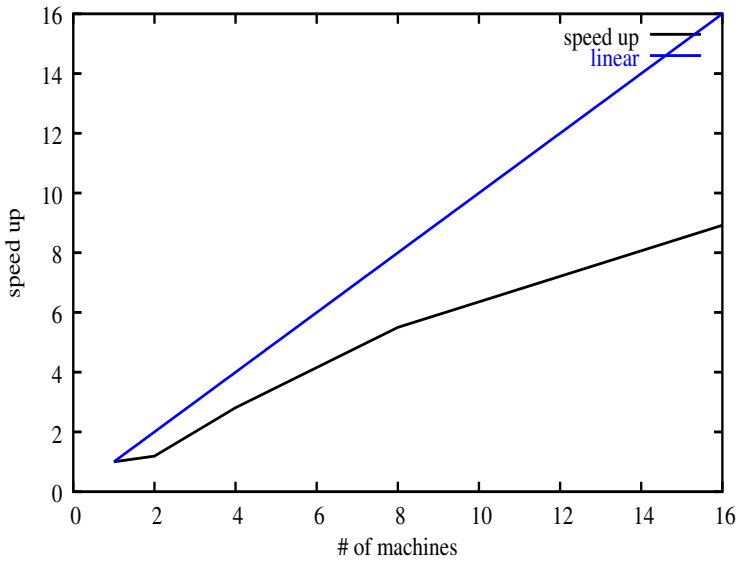

Figure 6.6: speedup for the enhanced prediction method for different number of processors

limitation of time and a better quality of prediction can be obtain. Therefore, if we want to predict fire evolution faster than real fire spread, the use of parallel processing becomes crucial.

6.2 Experiments Methodology

The experiments were designed to test the difference between the classical prediction method and the enhanced prediction method. For every time period we therefore applied both classical and enhanced prediction methods.

During the total burning time for each experiment, we picked up several time instants (t_0, t_1, t_2...etc.). We then extracted a fire line from the recorded film at each of the selected instants (RFL0, RFL1, RFL2...etc.). We used the first fire line (RFL0) as the initial fire line. As input to the simulator we used the expected parameters mentioned in section 6.1.2. 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 ($\triangle t_1 = t_1 - t_0$). The fire line that the simulator creates when executed with these settings is the predicted fire line using the classical method (CPFL1). We then compare the predicted fire line

using the classical method with the fire line extracted from the recorded film at the next instance using the XOR function to estimate the prediction error $PE_1 = XOR(RFL_1, CPFL_1)$.

After that, we apply the enhance prediction method as follows: 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. 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. At that moment, we have a fire line that predicts the fire line at t the time t_2 (EPFL2). Then, as in the classical prediction method, we use XOR to estimate the prediction error by comparing the estimated fire line with the real fire line $PE_2 = XOR(RFL_2, EPFL_2)$.

When the burn time of the experiment was long enough, we repeated 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.

We start our optimization process with a set of guesses. This set of guesses contains the expected parameters. At the next optimizations of the same burning experiment we use the final generation from previous optimization process that includes the optimized set of parameters.

Note that the enhanced prediction method needs more fire lines than the classical prediction time so we cannot apply it at the first time t_1, but the classical prediction method can be applied. In the experiment, we want to compare the two methods so we do not apply the classical at the first time because we have no enhanced prediction method to compare it with.

Despite the duration of this laboratory experiment, its post-mortem analysis allows us to validate the behavior of our prediction method, as we will show in the following sections.

Figure 6.7: Ignition point of table experiment

6.3 Case 1: 35º Slope, no Wind and Maritime Pine Fuel

In this experimental study, the laboratory environment was set up as follows: table inclination was set to 35º grades; there was no artificial wind and the fuel bed consisted of maritime pine ("pinus pinaster"). figure 6.7 shows ignition of a table experiment. Figures 6.8a and 6.8b show the recorded images for the ignition fire and fire spread after 1 minute, respectively. It should be taken into account that the slope increases from the bottom of the images toward the top. Fire time was about 1m 30" from the moment of ignition.

The parameters that can vary from one experiment to another with the same kind of fuel are described in table 6.5, with their corresponding values. There was no artificial wind, so the wind velocity and direction are set to zero. The humidity was measured and load was set by the investigators.

We used the prediction scheme described in figure 1.7 for two different situations. In the first case, the initial time t_0 was chosen as 30", consequently, the fire line at that time was RFL1. For this initial situation, we predicted the new situation of the fire front at time 1 minute by executing the fire simulator in

(a) (b)

Figure 6.8: case 1: Image recorded as an ignition fire (a), and fire spread after 1 minute (b)

parameter	value	unit
wind speed	0	mil/h
wind direction	0	degree
Moisture 1 hour	0.1536	lb water / lb fuel
Moisture 10 hours	0.1536	lb water / lb fuel
Moisture 100 hours	0.1536	lb water / lb fuel
Moisture of herpes	0.1536	lb water / lb fuel
fuel depth	0.25	feet
fuel load	0.134	lb fuel / sq ft bed

Table 6.5: case 1: Input parameter values measured in the laboratory fire.

initial time	prediction time	total burned cells	prediction error	error %
30"	1 min	180	95	52%
1 min	1 min 30"	401	101	25%

Table 6.6: case 1: Prediction error for the classical prediction method applied in the laboratory fire.

an isolated way. The second prediction was performed by considering 1 minute as the initial time and the predicted fire line was obtained for time instant 1m 30". Table 6.6 summarizes the obtained results. If we consider the prediction errors in terms of percentage according to the total number of cells burnt, we have 52% and 25% for the first and second prediction, respectively. Bearing in mind the dimension of the fire (3x3 m^2), this percentage of error is considered high.

The values in table 6.6 show that the classical prediction method does not provide accurate predictions, despite the study case being well known due to its laboratory nature. What would happen in a real fire situation where there are several types of uncertainty? We therefore applied the enhanced prediction method to the same lab fire in order to compare the obtained prediction with that from the earlier experiment. As commented, we used the Genetic Algorithm as an optimization strategy, which will be iterated 1000 times. The input parameters to be tuned are those shown in table 6.4, and the optimization process will work under the assumption that there is no previous knowledge of the input parameters. The complete enhanced prediction method, depicted in figure 2.2, was applied twice, once at 30 seconds and again at 60 seconds. The first optimization process provided an "optimal" set of input parameters, which were used to predict the new fire line situation at 60 seconds. This prediction was obtained by executing the fire spread simulator once, feeding it with the real fire line at 30 seconds and with the "optimal" set of parameters obtained for that time. Subsequently, we continue the process of optimization to predict the fire line at 1 minute 30 seconds. We used the optimized parameters at 30 seconds as the initial generation, repeating the same process using the real-fire line at 1 minute as reference. Optimized parameters were used to predict the fire line at 1minute 30 seconds. The result obtained in terms of improvement in prediction quality are shown in figure 6.9. This figure plots both the enhanced

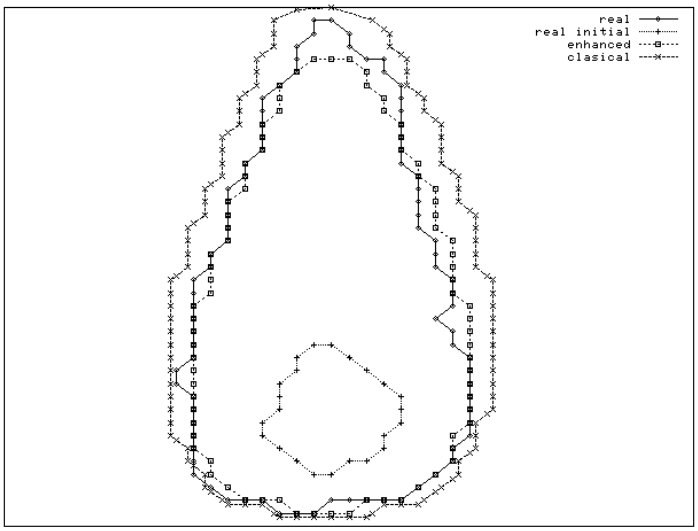

Figure 6.9: case 1: Predicted fire lines for the laboratory fire at 90" applying the classical and the enhanced prediction methods.

and classical predicted fire line versus the real fire line. As we can observe from both the plotted fire lines and the obtained prediction errors, the proposed prediction scheme outperforms the results obtained applying the classical scheme. In particular, the prediction errors obtained, in percentage, are 40% and 11% for both predictions, respectively. This means a reduction of 20% of the error in the first case with respect to the classical approach, and a reduction of more than 50% of the error for the prediction at 90 seconds.

We can therefore conclude that the enhanced prediction method provides better results than the classical prediction scheme in this case. In particular, we observe that the accumulation effect of the optimization method (1000 iterations at 30 seconds plus an additional 1000 iterations at 60 seconds) provides better prediction quality. We should thus be able to iterate the process as much as possible under real-time constraints in order to guarantee good prediction quality. For this reason, we apply parallel processing to accelerate the optimization process.

initial time	prediction time	classical prediction error	enhanced prediction error
30"	60"	95	72
60"	90"	101	45

Table 6.7: case1 :Predicted error for the laboratory fire at 60" and 90" applying the classical and the enhanced prediction methods

parameter	value	unit
wind speed	0	mil/h
wind direction	0	degree
Moisture 1 hour	0.12	lb water / lb fuel
Moisture 10 hours	0.12	lb water / lb fuel
Moisture 100 hours	0.12	lb water / lb fuel
Moisture of herbs	0.12	lb water / lb fuel
fuel depth	0.25	feet
fuel load	0.0614	lb fuel / sq ft bed

Table 6.8: Input parameter values measured in the laboratory fire

6.4 Case 2: 30º Slope no Wind and Straw Fuel.

In this experiment, we repeated the same steps as in case 1, but at this time we had different fuel and table slope. The fuel was straw and the table slope was 30º. The moisture was measured before the burn and the result is shown in table 6.8. We have loaded 0.3 kg/m2 of straw on the table, which is equivalent to 0.0614 pounds per squared feet.

Once again we applied the classical prediction method by feeding the simulator with the stated parameters values and calculate the prediction error using the resulting simulated burned area and the extracted burned area from the experiment infrared images. The duration of the burning in this experiment is more than the previous one, which allows us to make further prediction steps. In this case, we applied the method three times at time instances 60, 90 and 120 seconds. The results of classical and enhanced prediction methods are summarized in table 6.9. The first row of the table means that if we are at time instance 60 seconds and make a classical prediction for 90 seconds, we will get a prediction error of 133 cells, and if we use the enhanced prediction we can make the prediction with 97 cells of error.

In this case the classical prediction method (third column) also shows poor

initial time (seconds)	prediction time (seconds)	classical prediction error (cells)	enhanced prediction error (cells)	Total burned cells
60	90	133	97	539 at "90
90	120	136	83	913 at "120
120	150	130	90	1465 at "150

Table 6.9: case 2: comparison between classical and enhanced prediction methods

prediction accuracy. Comparing the enhanced prediction with the classical, the numbers show clear enhancement of prediction quality. For instance, predicting for 90 seconds, the classical prediction error was 24% of the total burned area and the enhanced was 17%, at 120 seconds the classical prediction error was 14% and enhanced was 9%, at 150 9% and 6 %, respectively. This result shows that the enhanced error is between 60% to 70% of the classical error. Or we have from 30% to 40% of enhancement.

We can therefore conclude that the enhanced prediction method provides better results than the classical prediction scheme in this case. However, we cannot observe, as in the previous experiment, accumulative enhancement in the prediction quality.

6.5 Case 3: No Slope, Variable Wind and Straw Fuel.

The methodology assumes that the changes in the input parameters' values in the short term is not so radical. Therefore, if we know the best parameters for the current time, we expect to have good prediction using the same parameters. The wind is one of the parameters which is not static at all. However, in this experiment we have changed the wind speed substantially during the experiment in order to change the situation drastically. This experiment was set to identify how the proposed methodology reacts when certain environment parameters drastically change. We started the experiment by setting the fans at the speed of 2.297 mph. Then, after 51 seconds, we switched the fans speed to 4.594 mph. Table 6.10 shows the expected values of input parameters for this experiment.

The fire duration was 80 seconds, which is a very short time. So, we have applied the method only for one step. Table 6.11 shows the prediction errors

parameter	value	unit
wind speed	4.059	mil/h
wind direction	90	degree
Moisture 1 hour	0.12	lb water / lb fuel
Moisture 10 hours	0.12	lb water / lb fuel
Moisture 100 hours	0.12	lb water / lb fuel
Moisture of herbs	0.12	lb water / lb fuel
fuel depth	0.25	feet
fuel load	0.0614	lb fuel / sq ft bed

Table 6.10: Input parameter values measured at the laboratory fire

initial time (seconds)	prediction time (seconds)	classical prediction error (cells)	enhanced prediction error (cells)	Total burned cells
60	80	184	161	1122

Table 6.11: case 3: comparison between classical and enhanced prediction methods

of applying the classical and enhanced prediction methods. We optimized the parameters for 60 seconds, then we provided the simulator with the parameters resulting from the optimization to predict the burned area at 80 seconds. The resulting XOR between the real fire line and the simulated was 161 cells (or 14% of the total burned area), while when we feed the simulator with the input parameters that had been estimated as described in subsection 6.1.2, the prediction error was 184 cells (or 16% of the burned area).

Even in this case where the wind changes drastically, we can conclude that the enhanced prediction method provides better results than the classical prediction scheme. However this case is of very short duration. In the next section we will describe another similar experiment but of greater duration so we can make more prediction steps to have better view of the method behavior.

6.6 Case 4: No Slope, Variable Wind and Maritime Pine Fuel.

As in the previous experiment, we changed fan speed from 2.297 mph to 4.594 during the experiment. The only variation in this experiment was the fuel: in-

parameter	value	unit
wind speed	2.297/4.594	mil/h
wind direction	0	degree
Moister 1 hour	0.16	lb water / lb fuel
Moisture 10 hours	0.16	lb water / lb fuel
Moister 100 hours	0.16	lb water / lb fuel
Moisture of herpes	0.16	lb water / lb fuel
fuel depth	0.25	feet
fuel load	0.134	lb fuel / sq ft bed

Table 6.12: Input parameter values measured at the laboratory fire

stead of using straw, we used maritime pine. The characteristics of maritime pine are different to straw. It has more humidity and the load is different. Maritime pine burns slower than straw, so we will have more time to make additional prediction steps so we can understand the behavior of the methodology in more details.

Table 6.12 shows the values of the parameters that have been measure and set for the experiment.

After we have filmed the experiment with infrared camera and extracted the fire lines every 15 seconds, we used the method to predict the behavior of the fire. Table 6.13 summarizes the results of the classical and enhanced prediction. At the first step, the enhanced prediction makes very good enhancement over the classical prediction. The method adjusts the simulator inputs to the given fire perimeter at time equal to 0.75 minutes and when using these values the simulator can predict the behavior of the fire at time 1.25 minutes with good prediction error. But when adjusting the parameters for the fire perimeter at time 1.25, it fails to predict the behavior of the fire at time 1.75 because the change of the wind speed has its effect on the fire at the time of prediction. The wind speed value that fits well at 1.25 does not make since at 1.75. This expected effect of changing the parameters drastically in short time. If we look at figure 6.10 we can distinguish between the different fire behavior at the beginning, where the fire lines are close together (the time of extracting different fire lines is constant) which means that the fire is slow and then suddenly the fire consumes large areas meaning that the speed of the fire propagation has been increased. The method once again adjusts the values of the input parameters to adapt to the change in the next step. Here we have to bear in mind two things: first we are working on a small scale of time, second the classical prediction here

initial time (m)	prediction time (m)	classical prediction error (cells)	enhanced prediction error (cells)	Total burned cells
0.75	1.25	38	28	224
1.25	1.75	81	123	518
1.75	2.25	117	50	965
2.25	2.75	123	67	1563

Table 6.13: case 4: comparison between classical and enhanced prediction methods

benefits from a knowledge of the changes in the wind speed. In real fire, the wind will not be changed by anybody, instead it will be predicted with uncertainty.

We can conclude that the method requires two step executions when the environment changes very roughly in order to react to the new environmental states. In figure 6.10, we can observe that the first lines of the fire are close together, then the last lines are sparse, which reflects that the fire starts slowly; then, when we change the wind speed, the fire increases its speed. Optimizing the parameters for the first part will give us a slow wind; then using this wind will not be good for predicting the second part. When using the new fire line, optimization will reflect the speed of the fire by increasing wind speed, which will adjust the prediction.

We can also conclude that the method needs complementary tools of prediction such as wind and humidity models to be more efficient. For example if we have a prediction that the wind will increase, we can increase the wind speed that we have as result of optimization before making the enhanced prediction. Another possibility is to use the history of the parameters and extract the trends from them, then, to add the trend on the optimized parameters. Or time series analysis prediction methods can be used, this depends on the nature of the parameter; for example wind parameters can depend on weather prediction, humidity can use some other models to predict it. This method is therefore not mutually exclusive with other methods: it can be used side by side and in interaction with others.

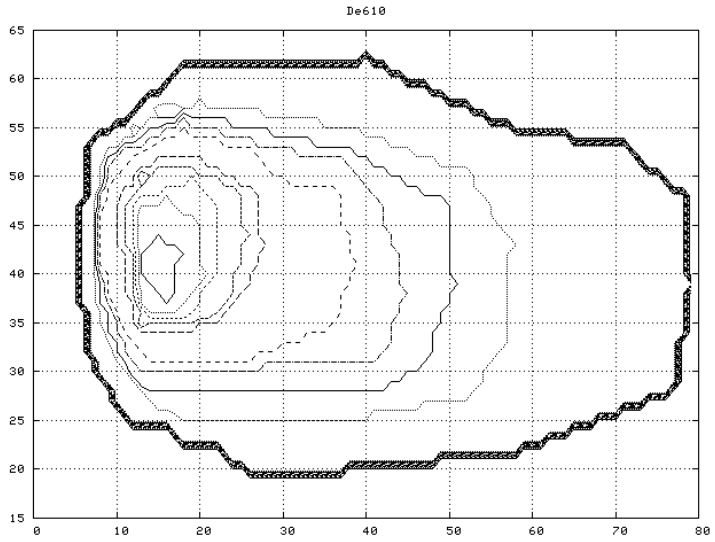

Figure 6.10: extracted fire lines of the experiment case 4

6.7 Case 5: 30° Slope, no Wind and Variable Fuel.

This case has been especially designed to experiment the methodology when the fuel parameters changes. Different from all other table experiment that have been experimented, in this one we used two types of fuels. The lower half of the burning table was covered with "pine pinuster" and the upper half with straw. This experiment has several aspects of difficulties in comparison with the previous cases. The first aspect is the sudden change of the fuel parameters. In the previous experiment, only one parameter changes (wind speed), In this experiment, when the fuel changes several important parameters change, such as the humidity, load ...etc.

The second difficulty is for the simulator. The fire makes strange shapes at the boundary of two fuels, one burns faster than the other. The simulator used needs to reflect the behavior of the fire in a reasonable way.

Using two fuels means that we need to feed the simulator with two sets of fuel parameters, so, when optimizing, we need to optimize more parameters (in this case 18 parameters) which makes the search space exponentially bigger than the search spaces in the previous cases. This increase in search space makes it

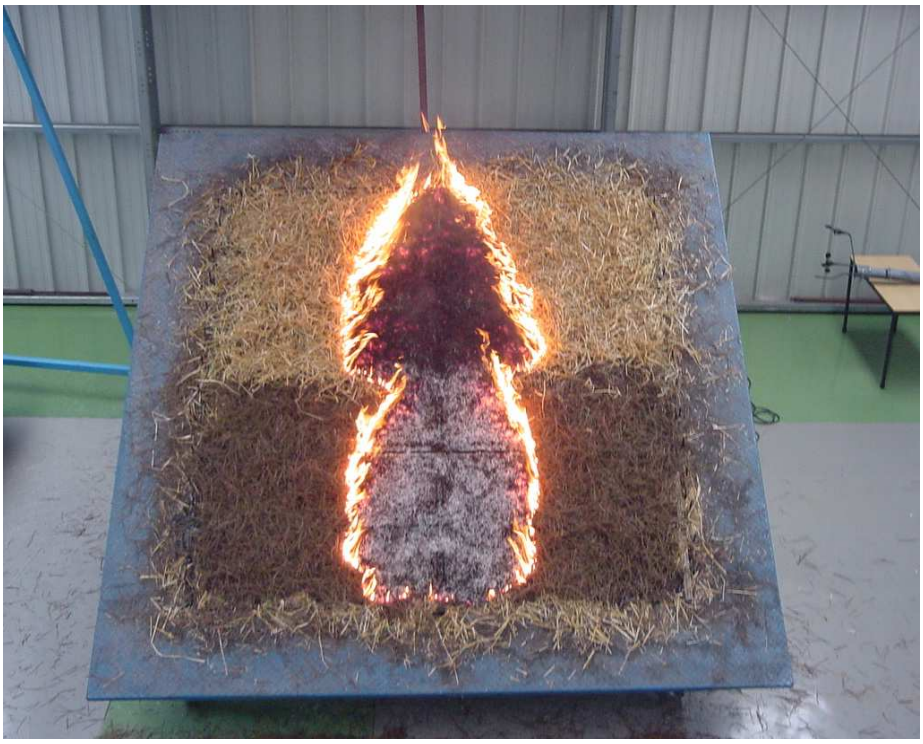

Figure 6.11: Table experiment using two types of fuel

harder for the optimization algorithm to converge to fitter solutions.

With all the difficulties mentioned in this case, the method still has good performance as in the previous case. Enhanced prediction is better than the classical, except in the point of drastic change (table 6.14). The table shows that the enhanced prediction method outperforms the classical prediction method except in the step where the second fuel starts to burn, where the classical outperforms the enhanced, but the method immediately readjust the parameters and in the next step it has a better performance than the classical. From these experiments, we can conclude that the method is robust and can enhance prediction.

6.8 Chapter Conclusions

In this chapter, we have applied the enhanced prediction methodology on some real experimental fires and we have compared it with the classical prediction

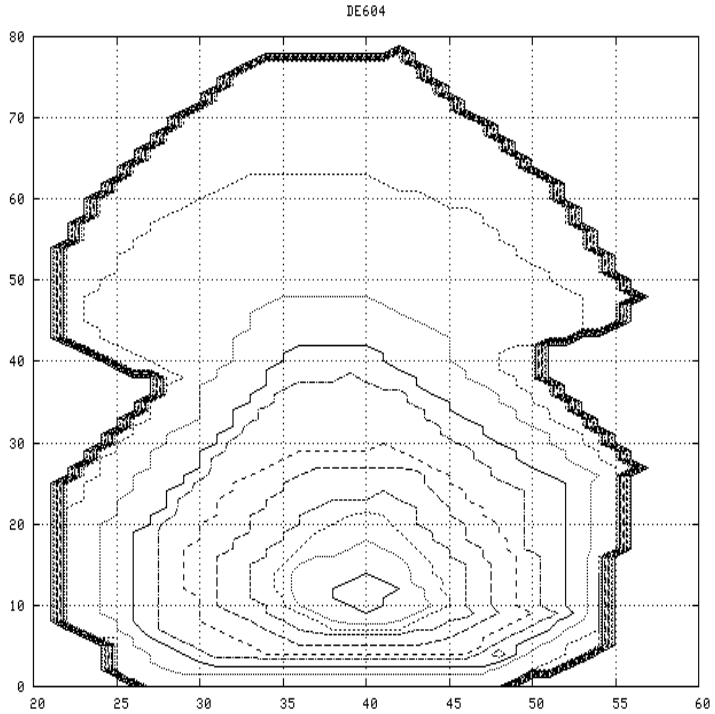

Figure 6.12: extracted fire lines of the experiment case 5

initial time (m)	prediction time (m)	classical prediction error (cells)	enhanced prediction error (cells)	Total burned cells
1.75	2.25	58 / 21%	16 / 6%	264
2.25	2.75	71 / 18%	17 / 4%	379
2.75	3.25	51 / 8%	92 / 14%	620
3.25	3.75	133 / 17%	96 / 12%	742
3.75	4.25	260 / 25%	138 / 13%	1009

Table 6.14: case 5: comparison between classical and enhanced prediction methods

method. The experimental cases show that the classical prediction method does not provide accurate predictions, despite the study cases being well known due to their laboratory nature. As we can observe from the obtained prediction errors in all cases, the proposed prediction scheme outperforms the results obtained when applying the classical scheme. We can therefore conclude that the enhanced prediction method provides better results than the classical prediction scheme in these cases. On the whole, the enhanced method has 72% of the classical error and, in the best case, it was 24% of the classical error.

Analyzing those cases where the environment changes drastically, we can conclude that the method needs a two step execution when the environment changes very roughly in order to adjust to the new environmental states. We can also conclude that the method needs complementary tools of prediction such as wind and humidity models to be more efficient.

The method overcomes many difficulties of the forest fire prediction problem but in some extreme cases it is not so good and still needs to be associated with other methods in order to forecast and model most particularly the wind and humidity.

Chapter 7

Conclusion and Future Work

> "podemos, hermano Sancho Panza,
> meter las manos hasta los codos en
> esto que llaman aventuras."
>
> *Miguel de Cervantes*

This chapter presents the conclusions obtained from this thesis and the current work and work plan to be undertaken in the future in order to continue research on forest fire prediction.

7.1 Conclusions and Global Observations

Forest fire is a very important problem that requires precise prediction to minimize its effects. There are several propagation models in the literature, but most of these require a great number of parameters that are usually difficult to measure or estimate. In this work, we have proposed a methodology that enhances the prediction of the fire propagation.

The amount of parameters and range of possible values mean that the search space required to optimize the parameters is extremely large. Therefore, optimization techniques supported by high performance computing is a significant tool with which to allow the propagation models to become truly operational. Using optimization techniques on distributed computing systems, it is possible to guide the optimizing process by reducing the search space and to test many possibilities in a short period of time (far faster than real time).

To implement the proposed methodology, we have developed a meta-tool for optimizing black-boxes. We call our tool BBOF (Black-Boxes Optimization Framework). We show that BBOF is useful for the problem in hand. The description of BBOF and its application can be found in:

> Baker Abdalhaq, "Cluster-Oriented Framework for Optimisation Problems" experimental work for postgraduate program for Computer Architecture and Parallel Processing, Universitat Autònoma de Barcelona, Barcelona (Spain), September 2001.

> B. Abdalhaq. A. Cortés, T. Margalef. and E. Luque, "Evolutionary Optimization Techniques on Computational Grids", Computational Science: International Conference: proceedings/ICCS 2002, Amsterdam, The Netherlands. PP. 513-522, April 2002, LNCS 2329.

> B. Abdalhaq. A. Cortés, T. Margalef. and E. Luque, "Optimization of Fire Propagation Model Inputs: A Grand Challenge Application on Meta-computers", 8th International Euro-par Conference Paderborn, Germany. 447-451, August 2002, LNCS 2400.

We have implemented several optimization techniques (Analytical optimization, Random search, Genetic Algorithms, Taboo Search and Simulated Annealing) and we have concluded that heuristic techniques are powerful tools for solving this kind of problem, which can be considered as a Grand Challenge Application (GCA).

The Analytical approach does not work if we have no clear information about the objective function. We needs considerable research to reach a satisfactory model, which will be problem specific. If we change some parameters in the problem, the shape of the objective function will change and we need to search for another model. For example, if we change the wind field or the terrain, there is no guarantee of having the same shape of the objective function and, therefore, a previous objective function cannot fit well for the new characteristics of the problem.

The genetic algorithm needs a lot of tuning itself because there are several parameters that must be tuned that affects the performance. There are some

parameters without extensive tuning left for further work, which may lead to better performance.

The Taboo Algorithm is a powerful algorithm, easy to program. It does not have many parameters to tune.

Four optimization techniques (Random Search, Simulated Annealing, Taboo Search and Genetic Algorithm) have been tested on a PC cluster with 21 PCs. The experiments were carried out using different fire scenarios. The results show that Simulated Annealing, Taboo Search and Genetic Algorithm provide similar results. A summary of this experimental study can be found in :

> B. Abdalhaq. A. Cortés, T. Margalef. and E. Luque, "Optimization of parameters in forest fire propagation models" Forest Fire Research & Wild-land Fire Safety 2002 abstract P. 114 full paper in CD. ICFFR02.

Since optimizing is a time-demanding task, we have proposed a global sensitivity analysis to accelerate optimization convergence. This technique reduces the search space screened by fixing the less sensitive parameters to an estimated value and by focusing optimization on the most sensitive parameters. We have also reduced the range of each optimized parameter by introducing some degree of knowledge for each of these. This was considered by limiting the variation of these parameters around a known value (field measurement). Sampling the search space is another way of reducing it. We extend the sampling to be related to the sensitivity of the parameters in order to make more effort on the sensitive parameters and save effort in optimizing non-sensitive parameters. These techniques were carried out on a Linux cluster composed of 21 PC's. We used a master/worker programming paradigm, where the master and worker processes communicate with each other using MPI. The results show that these accelerating strategies help in reducing optimization time. Some results of this experiment have been published in:

> B. Abdalhaq. A. Cortés, T. Margalef. and E. Luque, "Accelerating optimization of input parameters in wild-land fire simulation.", Fifth International Conference on parallel processing and applied mathematics, Czestochowa, Poland, LNCS 3019, 7-10 September 2003.

B. Abdalhaq. G. Bianchini. A. Cortés, T. Margalef. and E. Luque, "Improving convergence speed of optimization of input parameters in wild-land fire simulation.", congress: XIV Jornadas de Paralelismo. 15-17 September 2003, Leganés-Madrid, pp. 337, 15-17 September 2003.

We have experimented the methodology on a set of table experiments that have been designed especially to challenge the method and mimic real fire situations. The experiments show that the method is robust and has good performance in all of these cases.

The most relevant results from this experiments in addition to a description of the methodology have been published in:

B. Abdalhaq. G. Bianchini. A. Cortés, T. Margalef. and E. Luque, "Improved Wild-land Fire Prediction on MPI Clusters", 10th European PVM/MPI User Group Meeting Venice,Italy. pp. 520-528, September / October 2003., LNCS 2840.

B. Abdalhaq. G. Bianchini. A. Cortés, T. Margalef. and E. Luque, "Accelerating Wildland Fire Prediction on Cluster Systems", Congress: International Conference on Computational Science, reference:Part II PP. 225-232 (volume number LNCS 3037), June 6-9, (2004).

B. Abdalhaq. G. Bianchini. A. Cortés, T. Margalef. and E. Luque "Between Classical and Ideal: Enhancing Wildland Fire Prediction Using Cluster Computing", Jornal of Cluster Computing Special Issue on cluster computing in science and engineering. (2004).

7.2 Current and Future Work

From the experience obtained throughout the development of this work, as explained in this thesis, new ideas have emerged, some of which are practically concluded, while others are still being worked on. We now outline all current and future lines of work.

The implementation of the genetic algorithm that we have carried out has some parameters that have not been intensively studied; therefore, an immediate work that could be performed constitutes studying how the tuning of these

parameters affects the results. In particular, for the case of mutation probability, which used to be constant all the time, we propose that it can vary as the process continues in order to improve results. More precisely, it can be changed dynamically while getting closer to the solution. It would start with a large probability to insure exploring search space and, afterwords when the solution gets closer to the optimal, we would use a smaller mutation distance to concentrate on elite solutions. This technique is borrowed from Taboo and Simulated Annealing.

Other types of mutation can be introduced and experimented, for example, a mutation that uses normal description and expected values as the mean.

Crossover is carried out using two selected individuals. We can add restriction on the two individuals participating in the crossover process to increase the speed of conversion. First we divide individuals into three groups: smaller than the real fire line, greater than the real fire line and equal to real fire line. Then, we do not allow the crossover between the individuals that are in the same group. Bearing in mined that the crossover includes averaging, when we choose one individual that yields fire line greater than the real fire line with one yields fire line that is smaller than the real fire line it is most probably that the product of the crossover is better individual.

It is worthwhile discovering more optimization methods, especially modern heuristic methods, because it is clear from this study that these methods are suitable for solving black-box optimization problem such as the one we have evaluated. Also hybrid methods that combine more than one optimization method and work in two phases (globally and local) can be potentially powerful methods.

All the study reported in this work deals with the analysis of parameter optimization on a post-mortem system. However, we have experimented on several methods to accelerate the convergence that can be useful in the case of using the method under the real time constraints. It would still be interesting to study the method using a real fire in a real forest to experiment in what conditions the method meets the real-time constraint, and when it fails to do so.

We have experimented on what we call a linear sampling technique, where the relation between the sampling frequency and the sensitivity index is linear. Other types of relations are possible and may have better performance.

Integrating the methodology in a complete decision-support system that contains an on-line real fire line acquisition and supports the decisions of the firefighters in the real emergency will be a very interesting line of investigation.

It would be interesting to discuss the applicability of the method in other similar broadcasting problems such as floods.

The idea of using the feedback from the fire itself can be used to create a new generation of fire simulators that are intelligent and self-adapted. The simulator can learn from the fire itself and tries to guess the basic rate of spread R_0, which is the speed of spread without slope and wind depending on the characteristics of the fuel. The wind can also be guessed and terrain is the only input of the model. In this way, we will only have R_0 and U as parameters to be optimized.

As mentioned in the introduction, the main purpose of all work in the field of forest-fire simulation is to create operational tools to help the 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 loss. Good prediction tools are vital to making good decisions. In this work, we provide a tool to provide good predictions. But we can still make additional support for the decision process. For example, we can create a tool that chooses the best evacuation plan for the houses in the forest. We also can make a tool that allocates firemen in the forest, taking into account the safety factor as well as the probability of the fire reaching such locations. A wide range of such tools can be created based on a prediction system.

Bibliography

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

[2] Abdalhaq B., Cortés A., Margalef T., Luque E., "Optimization of Fire Propagation Model Inputs: A Grand Challenge Application on Metacomputers". LNCS 2400,pp. 447-451. (2002).

[3] Adenso-Díaz B., Laguna M. "Fine-tuning of Algorithms using Fractional Experimental Designs and Local Search" http://leeds.colorado.edu/faculty/laguna/articles/finetune.html, technical report (2002).

[4] Akl S., Bruda S. "Improving A Solution's Quality Through Parallel Processing." The Journal of Supercomputing 19 (2) pp. 221 - 233(2001).

[5] Akl S., Bruda S., "Parallel Real-time numberical computation: Beyond Speedup III" Technical report No 99-424, Queens University, August 10, (1999).

[6] André J.C.S., D.X. Viegas, "A Strategy to Model the Average Fireline Movement of a light-to-medium Intensity Sirface forest Fire", Proc. of the 2nd International Conference on Forest fire Research, pp. 221-242, Coimbra, Portugal, (1994).

[7] André, J.C.S., "A Thery on the propagation of surface fire fronts", PhD Dissertation (in protugues), Universidade de Coimbra, Portugal, (1996).

[8] Andrews P.L. "BEHAVE: fire behavior prediction and modeling systems" Burn subsytem, part1. General Technical Report INT-194. Odgen, UT,

US Department of Agriculture, Forest Service, Intermountain Research Station; 130 pp (1986).

[9] Arnold D., Agrawal S., Blackford S., Dongarra J., Miller M., Sagi K., Shi Z., Vadhiyar S. "Users' Gide to NetSovle V1.4:((http//icll.cs.utk.edu/netsolve/),university of Tennessee (2001).

[10] Arsham H. "System Simulation: The Shortest Path from Learning to Applications", Http://ubmail.ubalt.edu/~harsham/simulation/sim.htm

[11] Asselmeyer T., Ebeling W., "Analytic Investigation of Random Search Strategies for Parameter Optimization", http://arxiv.org/abs/adap-org/9508003, 10 August (1995).

[12] Baeck, T., Hammel, U., Schwefel, H., "Evolutionary Computation: Comments on the History and Current State" IEE Transactions on Evolutionary Computation, Volume 1, number 1, pages 3-17, April 1997.

[13] Bachmann A., Allgöwer B.,"Error Propagation in Wildfire Behaviour Modelling" Proceeding Accuracy 2000, Amsterdam , Jully (2000).

[14] Bäk T. "Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms", Oxford University Press Oxford, UK, ISBN:0-19-509971-0 (1996)

[15] Bäck T, Rudolph G., Schwefel H.-P. "Evolutionary Programming and Evolution Strategies: Similarities and Differences", pp. 11-22 in D.B. Fogel and W. Atmar (eds.), Proceedings of the 2nd Annual Conference on Evolutionary Programming, La Jolla, Evolutionary Programming Society (1993).

[16] Blesa M., Petit j., Xhafa F. "Computación en Internet: Librería MALLBA para problemas de optimización" Ciencia y Tecnología, volume II, pp. 9-12. Tibidabo Ediciones, Barcelona, ISBN: 84-8033-145-3 (2001).

[17] Bennett K., Blue J. "An Extreme Point Tabu Search Method for Data Mining" Math report no 228, Dep. Mathematical Scinces Rensselaer Polytechnic Institute (1996).

[18] Beven K., Binley A. "The Future of Distributed Models: Models Calibration and Uncertainty Prediction" Hydrological Process, Vol. 6, pp. 279-298 (1992).

[19] Blue J. A. "A hybrid of Tabu search and local descent algorithms with applications in artificial intelligence" Thesis, Rensselaer Polytechnic Institute, NewYourk, May (1998).

[20] Erik Bølviken and Ragnar Winther, Databehandling og anvendt matematikk, in "IT neste TI,Informasjonsteknologi de neste ti ar" (in Norwegian), Petter Gottschalk (editor), DND og Gyldendal 1993.

[21] Bruda, Akl S., "Real-time computation: A formal definition and its applications", technical report no, 2000-435, Queens univeristy, February 8, (2000).

[22] Buyya A. Coley. "An Introduction to Genetic Algorithms for Scintists and Engineers", World Scientific, (1999).

[23] Buyya R., "High Performance Cluster Computing: programming and Applications" volume 2, Prentice Hall PTR (1999).

[24] Calégari P. "Parallelization of population-based evolutionary algorithms for combinatorial optimization problems" Thesis no 2046, École Polytechnique Fédérale de Lausanne (1999).

[25] Calogine D., Séro-Guillaume O.,"Air flow model in a tree crown", Forest fire Research & Wildland Fire Safty, " abst. p 115 full paper on CD-ROM ISBN 9077017720,Millpress (2002).

[26] Chetehouna K., DeGiovanni. A., Margerit J. , Séro-Guillaume O. Technical Report, INFLAME Project, October (1999).

[27] Coley D. A.: "An Introduction to Genetic Algorithms for Scientists and Engineers", World Scientific, (1999).

[28] Coello C. "An empirical study of evolutionary techniques for multiobjective optimization in engineering design", Thesis Tulane University" (1996).

[29] Collin D. Bevins , "FireLib User Manual & Technical Reference", 1996, www.fire.org

[30] Crainic T. G., Toullouse M., Gendreau M., "Toward a Taxonomy of Parallel Tabu Search Heuristics",INFORMS Journal on Computing vol. 9, no 1, Winter (1997).

[31] Computational Science Education Project, "E-book" (electronic book), Available through Mosaic or similar programs at "http://csep1.phy.ornl.gov/csep.html".

[32] De Jong, K.A., "Genetic Algorithms are NOT function Optimisers", In whitleg, L.D.,(ed), Foundation of genetic Algorithms 2, Morgan Kaufmann, (1993).

[33] De Berg M.,van kreveld M., Overmars M., Schwarzkopf O., "Computational Geometry" second edition, Springer,2000.

[34] Efstratiadis A., Koutsoyiannis D. "An evolutionary annealing-simplex algorithm for global optimization of water resource systems" Fifth international conference on hydroinformatics, Cradiff,UK july (2002).

[35] Finney, M.A. "FARSITE: Fire Area Simulator– Model Development and Evaluation." USDA For. Serv. Res. Pap. RMRS-RP-4. (1998).

[36] Fishwick P. "Computer Simulation: The art and scince of digital world construction", september 21, http://www.cis.ufl.edu/~fishwick/introsim/paper.html (1995).

[37] Foster I. "Designing and building Parallel Programs." Addison Wesley, 1996, available at http://mcs.anl.gov/dbpp `http://www.mcs.anl.gov/dbpp`

[38] Frey H. C., Patil S. R."Identification and review of sensitivity analysis methods" NCSU/USDA workshop on Sensitivity analysis Methods june 11-12 2001, North Carolina State University. (2001).

[39] Galassi M., Davies J., Theiler J., Gough B., Priedhorsky R., Jungman G., Booth M., Rossi F., "GNU Scientific Library -Reference Manual" ,Edition 0.7, for gsl Version 0.7, 26 October 2000.

[40] Gillon D.,Valette J.C. , Moro C."Foliage moisture content and spectral characteristics using near infrared reflectance spectroscopy (NIRS)" Forest fire Research & Wildland Fire Safty, Portugal, " abst. p 127 full paper on CD-ROM ISBN 9077017720,Millpress (2002).

[41] Globus A., Langhirt E., Levny M., Ramamurthy R., Solomon M. Traugott S. "JavaGenes and Condor:Cycle-Scavenging Genetic Algorithms"

`Http://www.nas.nasa.gov/~glolbus/papers/JavaGrande2000/javaGrandePaper.html`.

[42] Glover, F., "Future paths for integer Programing and Links to Artificial Intelligence", Computers and Operations Research, 5:533-549, 1986.

[43] Glover f., Laguna M. "Tabu Search, in Modern Heuristics Techniques for Combinatorial Problems", C.R. Reves, editor, John Wiley & Sons, Inc., (1993).

[44] Glover f., Laguna M. " Tabu Search", Kluwer Acadimic Publisher, Boston, (1997).

[45] Gene Golub and James M. Ortega. "Scientific Computing, An Introduction with Parallel Computing" Academic Press (1993).

[46] Goux J., Linderoth J., Yoder M. "Metacomputing and the Master-Worker Paradigm", September 1999. http://www.cs.wisc.edu/condor/mw/.

[47] Goux J., Kulkarni S., Linderoth J., Yoder M. "Master-Worker: An Enabling Framework for Applications on the Computational Grid", Cluster Computing 4 , pp.63 -70.(2001)

[48] Gray P., Hart W., Painton L., Philips C., Trahan M., Wagner J."A Survey of Globall Optimization Methods", Sandia National Laboratories Albuquerque, NM 87185, `Http://www.cs.sandia.gov/opt/survey/` ,(1997).

[49] Crosetto M., Tarantola S. "Uncertainty and sensitivity analysis: tools for GIS-based model implementation" International Journal of Geographical Information Science 15 (5):415 - 437(2001)

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

[51] Gropp W., 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, Septempre, (1996)

[52] Hamby D.M., "A comparison of sensitivity analysis techniques" Health Physics 68: 195-204. (1995)

[53] Hanslen P.B, "Model Programing for computational Science: A Programming Methodololgy for Multicomputiers." Concurrency: Practice and Experience, Vol. 5(5). pp.407-423, (1993)

[54] Hargrove W.W., Gardner R.H., Turner M.G., Romme W.H., Despain D.G. "Simulating fire patterns in heterogeneous landscapes", Ecological Modelling 135 pp. 243-263.(2000)

[55] Herrera F., Lozano M., Verdegay J.L. " Applying Genetic Algorithm in Fuzzy Optimization Problems" Fuzzy Systems & A.I. Reports and Letters 3,1 39-52,Spain (1994).

[56] Holland, J., "Adaptation in Natural and Artificial systems", Universtity of Michigan Press. (1975).

[57] Jason D. Lohn and Silvano P.- colombano, "Automated Analog Circuit Synthesis Using a Linear Representation " Second Internationall Conference on Evolvable Systems: From Biology to Hardware, Springer-Verlag, 23-25 September 1998.

[58] Jorba J., Margalef T., Luque E., J. Campos da Silva Andre, D. X Viegas "Parallel Approah to the Simulation Of Forest Fire Propagation". Proc. 13 Internationales Symposium "Informatik fur den Umweltshutz" der Gesellshaft Fur Informatik (GI). Magdeburg (1999) pp. 69-81

[59] Jiang Xiao, Zbigniew Michalexicz, Lixin Zhang, and Krysztof Trojanowski, "Adaptve Evolutionary Planner/Navigator for Mobile Robots", IEEE Transactions on Evolutionary Computation, Volume 1, nuper 1, p.p. 18-28, April 1997.

[60] Keane A.J. "A Brief Comparison of some evolutionary optimization methods" pp.255 - 272 in Modern Heuristic Search Methods, ed. V. Rayward-Smith, I. Osman, C. Reeves and G. D. Smith,J. Wiley (ISBN5 96280 471 0) (1996).

[61] Kleijnen J. P.C. "Sensitivity Analysis and Related Analyses: a Survey of Statistical Techniques" International Symposium Theory and application lof Sensitivity Analysis Output in computer simulation, 25-27 Septemper 1995, Belgirate, Italy (1995).

[62] Kliewer G., Tschöke S. "A general parallel simulated annealing library (parSA) and its applications in industry", PAREO'98: First meeting of the PAREO working group on Parallel Processing in Operations Research, Versailles, France, July8 -10,(1998).

[63] Knight I., Coleman J. "A Fire Perimeter Expansion Algorithm Based on Huygens' Wavelet Propagation" Wildland Fire 3(2):73-84,(1993).

[64] Kohn M., "Use of sensitivity analysis to asses reliability of mathematical models" white paper laboratory of computational biology and risk analysis, http://www.ce.ncsu.edu/risk/abstracts/kohn.html.

[65] Laguna M. "A Gide to Implement Tabu Search" Investigación Operativa vol.4 no. 1 abril 1994.

[66] Laguna M., Martí R., Campos V. "Intensification and Diversifcation with Elite Tabu Search Solutions for the Linear Ordering Problem", Computers and Operations Research, 26, pp 1217-1230 (1998).

[67] Larnini M., "The complete physical model", University of provece (Marseille) CNRS-UMR 6595, Portugal 1998.

[68] Lopes,A.M.G., Viegas, D.X. & Cruz, M.G. "FireStation - An Integrated System for the Simulation of Fire Spread and Wind Flow Over Complex Topography Spreading", III International Conference on Forest Fire Research, Vol. B.40, pp. 741-754. (1998). [6] Martine-Millán and Saura, 1998.

[69] Lorenc A. C., "Atmospheric Data Assimilation", Forcasting research divition scientoific paper no 34.Forcasting research Meteorological Office, England, May 1995

[70] Luis Merino and Aníbal Ollero, "Computer Vision Techniques for Fire Monitoring using Aerial Images", Annual Conference of the IEEE Industrial Electronics Society, CD ROM ISBN: 0-7803-7475-4, (2002).

[71] Madsen H.,Kristensen M. "A multi-objective calibration framework for parameter estimation in the MIKE SHE integration hydrollogical modelling system." ModelCARE 2002, preceeding of the 4th international conference on Calibration and reliability in Groundwater Modelling, Prage, Czech repoblic,PP270-273 (2002).

[72] Madsen H., Butts M.B., Khu S.T., Liong S.Y. "Data assimilation in rainfall-runoff forecasting" Hydroinformatics 2000 4th International Conference on hydroinformatics, Cedar Rapid, Iowa, USA, pp. 23-27 july (2000).

[73] Madsen H. "Automatic Calibration and Uncertainty Assessment in Rainfalll-runoff modelling" 2000 Joint Conference on Water Resources Engineering and Water Resources Planning & Management, Hyatt Regency Minneapolis, USA July30- August 2, (2000).

[74] Martinez-Millán, J. and Saura, S. "CARDIN 3.2: Forest Fires Spread and Fighting Simulation System", III International Conference on Forest Fire Research. (1998).

[75] Mendes A., França, Moscato P. "NP-Opt: and optimization framework for NP problems"proceeding of IV SIMPOI/POMS Guarujá/SP- Brazil August 11-14, (2001).

[76] Merino L., Ollero A. "Computer vision techniques for fire monitoring using aerial images" IECON´02,CD ROM ISBN: 0-7803-7475-45-8 Nov (2002).

[77] Merino L.,Ollero A, "Forest Fire Perception using Aerial Images in the COMETS project", 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.11-21, (2002).

[78] Merino L., Gómez-Rodríguez F., Arrue B., Ollero A., "Aerial monitoring and measurement of forest fires" proceeding of the aerosense, Spie 16th Annual International Symposium, Vol. 4713. Enhanced and Synthetic Vision, pp.95-105 ISBN: 0-8194-4504-5,April (2002).

[79] Michalewicz, Z. Fogel, D. "How to Solve it: Modern Heuristics", Springer-Verlag Berlin Heidenberg 2000.

[80] Morvan D., Tauleigne V., Dupuy J.L, "Wind effects on wildfire propagation through a Mediterranean shrub", Forest fire Research & Wildland Fire Safty, " abst. p 116 full paper on CD-ROM ISBN 9077017720,Millpress (2002).

[81] Murta A., "A General Polygon Clipping Library" Version 2.31, Advanced Interfaces Group, Departament of Computer Scince, University of Manchester, Manchester M13 9PL,UK, http//www.cs.man.ac.uk/aig/staff/alan/software/gpc.html.

[82] Lasse Natvig, "Why Computational Science and Engineering Should be of Interest to Computer Scientist", Norsk Informatikkonferanse 1994, Molde 15 November (1994).

[83] Pelikan M., Goldberg D. E., Lobo F. "A survey of Optimization by Building and Using Probabilistic Models" IlliGAL Report No. 99108, Illinois Genetic Algorithm Laboratory. September 1999.

[84] Pincus M., "A monte Carlo Method for the Aprroximate Solution of Certain Types of Constrained Optimization Problems", Operations Research, 18, 1225-1228, (1970).

[85] Piñol , J. , Salvador, R., Beven K., " Model calibration and Uncertenity prediction of fire spred" Forest Fire Research & Wildland Fire Saftety, Viegas(ed), Millpress, Roterdam, 2002.

[86] Press W., Teukolsky S., Vetterling W., Flannery B., "Numerical recieps in C The art of Scinetific Computing", second edition, Campridge University press, (1992).

[87] Rajkumar Buyya, "High performance cluster computing, volume 2, Programming and applications", Prentice-Hall (1999).

[88] Ratto M., Tarantola S., Saltelli A., Callies U."model reduction techniques for time series normalisation" Estimation of human impact in the presence of natural fluctuations EC Fifth framework program SCA project IST-1999-11313, Deliverable 17, December (2000).

[89] Reiher E, Said F., Li Y. and Suen C.Y., "Map Symbol Recognition Using Directed Housdorff Distance and a Neural Network Classifier" proceedings of International Congress of Photogrammetry and Remote Sensing", Vol, XXXI, Part B3, Vienna, pp. 680-685 (1996).

[90] Resende M., Ribeiro C. "Greedy Randomized Adaptive Search procedures" AT&T labs Research Technical Report, To appear in State-ofart Handbook in Metaheuristics, F. Glover, G. Kochnberger, eds. Kluwer Academic Publishers, sebtember 13, (2001).

[91] 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 Safty, " abst. p 122 full paper on CD-ROM ISBN 9077017720, Millpress (2002).

[92] Rice John R., Academic Programs in Computational Science and Engineering, Technical report 93-042, Computer Science Dept., Purdue University, 1993.

[93] Richards G. D., Gryce R. W., " acomputer Algorithm for simulating the Spread of Wildland Fire Perimeters for Hetrogeneous Fuel and Meterological Conditions", Wildeland Fire 5(2): 73-79, USA (1995).

[94] Richards G. D. "A General Mathematical Framework for Modeling Two-Dimensional Wildland Fire Spread" Wildland Fire 5(2):63-72, (1995).

[95] Robinson A. R., Lermusiaux F.J. "Overview of data assimilation" Harvard report in Physical/interdisciplinary Ocean Scince number 62 august (2000).

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

[97] Ruiz A.D., Maseda C.M., lourido C., "Possibllities of dead fine fuels moisture prediction in Pinus pinuster Ait. stands as "Cordal de Ferreiros" (Lugo, North-western of Spain)", Forest fire Research & Wildland Fire Safty, " abst. p 126 full paper on CD-ROM ISBN 9077017720,Millpress (2002).

[98] Salvador, R., Piñol, P, Tarantola, S. And Pla, E. "Global Sensitivity Analysis and Scale Effects of a Fire Propagation Model used Over Mediterranean Shrub lands". Elsevier, Ecological Modelling 136 pp. 175-189, (2001).

[99] Satelli, A., K. Chan, M. Scott, Editors. "Sensitivity analysis". John Wiley & Sons publishers, Probability and Statistics series. (2000).

[100] Sen S. "Stochastic Programming: Computational Issues and Chalenges" Encyclopedia of OR/MS, S. GAss and C Harris (eds.)

[101] Shang Y. "global search methods for solving nonlinear optimization proplems" Thesis Urbana Illinois (1997).

[102] D. E.. Stevenson, Science, Computational Science and Computer Science: At a Crossroads,Aug. 1993, 16 pages. (To appear in CACM)

[103] Strousturp B., " El Lenguaje de Programación C++", Person Education, S.A., Mardir, (2002).

[104] Tanev I., Uzumi T., Ono K. "Scalable architecture for parallel distributed implementation of genetic programming on network of workstations" Jornal of systems architecture 47 PP 557-572, (2001).

[105] Vatti, B.R., "A Generic Solution to Polygon Clipping", Communications of the ACM, 35(7), July 1992, pp.56-63.

[106] Viegas D.X, Ribeiro L.M, Matos L., Palheiro P., Pita L.P., Afonso C., "Slope and wind effects on fire spread", Forest fire Research & Wildland Fire Safty, " abst. p 117 full paper on CD-ROM ISBN 9077017720,Millpress (2002).

[107] Wadsworth, Harrison M. "Handbook of statistical methods for engineers and scientists", McGraw.Hill, Inc. (1990).

[108] Wangwongwiroj N., Schlütter, Mark O. " Application of an automatic calibration scheme for Urban rainfall -runoff models in mouse" 4th DHI software conference ,(2001).

[109] Wright S. J. "Solving Optimization Problems on Computational Grids" November 12, http://www.cs.wisc.edu/~swright/papers/ (2000).

[110] Xanthopoulos G. , Manasi M. , "A practical methodology for the development of shrub fuel models for fire behavior predection", Forest fire Research & Wildland Fire Safty, " abst. p 124 full paper on CD-ROM ISBN 9077017720,Millpress (2002).

[111] Yi Shang "Global search Methods for solving nonlinear optimization problems", Phd Thesis University of Illinois as Urbana-Champaign, 1997.

[112] Zeigler B. P. "Multifacetted Modrlling and Discrete Event Simulation" Academic press. London (1984).

[113] Zeigler B. P., Praehofer H., Kim Tag Gon "Theoty of Modeling and Simulation" second edition, Acadimic Press (2000).

Index