Dynamic Tuning of a Forest Fire Prediction Parallel Method

Paola Caymes-Scutari[1,2](✉), María Laura Tardivo[1,3], Germán Bianchini[1], and Miguel Méndez-Garabetti[1]

[1] LICPAD, UTN-FRM, Mendoza, Argentina
pcaymesscutari@frm.utn.edu.ar

[2] CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Buenos Aires, Argentina

[3] Departamento de Computación, Universidad Nacional de Río Cuarto, Río Cuarto, Argentina

http://www.frm.utn.edu.ar/licpad/

Abstract. Different parameters feed mathematical and/or empirical models. However, the uncertainty (or lack of precision) present in such parameters usually impacts in the quality of the output/recommendation of prediction models. Fortunately, there exist uncertainty reduction methods which enable the obtention of more accurate solutions. One of such methods is ESSIM-DE (Evolutionary Statistical System with Island Model and Differential Evolution), a general purpose method for prediction and uncertainty reduction. ESSIM-DE has been used for the forest fireline prediction, and it is based on statistical analysis, parallel computing, and differential evolution. In this work, we enrich ESSIM-DE with an automatic and dynamic tuning strategy, to adapt the generational parameter of the evolutionary process in order to avoid premature convergence and/or stagnation, and to improve the general performance of the predictive tool. We describe the metrics, the tuning points and actions, and we show the results for different controlled fires.

Keywords: Dynamic tuning · Differential Evolution · Fire prediction · Parallel computing

1 Introduction

Year after year, forest fires constitute a great threat in different regions of the world, especially in summer, where high temperatures and prolonged drought provide an environment favorable to the development of these phenomena. In 2019, several serious fires occurred worldwide, including the case of the Amazon area with a loss of 2.5 million hectares of rainforest, and the case of Australia with a loss of 6 million hectares, as shown in Fig. 1. Due to the serious consequences they produce, it is important to have tools that allow predicting the behavior of fire to collaborate in fire fighting and fire prevention plans.

Fig. 1. Left: Forest fire in the Amazon area [14]. Right: Forest fire in Australia [15].

The prediction of a forest fire consists in determining how the fire will spread on the terrain in an instant of future time. Figure 2 illustrates both the general view of the simulation process and the particular case of the forest fire simulation for prediction. Generally, the prediction methods implement models that describe the behavior of the fire, and use as input data a set of variables representing those factors that condition the propagation. Some of the variables that are worth mentioning are: the wind speed and direction, the slope of the terrain, the type of combustible material, the humidity of the material, etc.

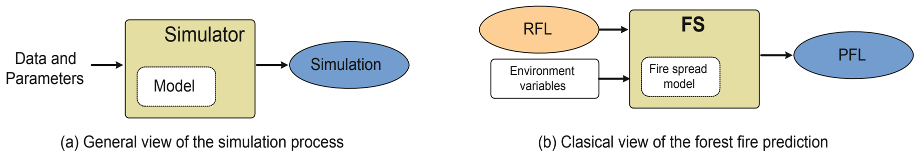
Data and Parameters
Simulator
Model
Simulation
(a) General view of the simulation process
RFL
FS
Environment variables
Fire spread model
PFL
(b) Clasical view of the forest fire prediction

Fig. 2. General view of simulation and prediction process. **RFL**: Real Fire Line. **PFL**: Predicted Fire Line. **FS**: Fire Simulator.

Unfortunately, in general it is not possible to have the exact values for these factors, due to two main reasons: on the one hand, it is not feasible to provide all forest land with sensors and measuring instruments. On the other hand, the value of some of the parameters can vary dynamically throughout the development of the fire itself. This lack of information and/or precision regarding the values of the parameters is called **uncertainty**. There are also other sources of uncertainty, such as the limitations for mathematically modeling all the details of the system under consideration, the computer representation of the information (which for example causes truncation), the implementation of the simulator, etc. Every uncertainty source could cause drastic consequences if the output of the model provides a solution certainly different from reality. All of them are detrimental to the accuracy of the simulator and the quality of the prediction. In [11] we introduced a series of methods belonging to the class of so-called Data

Driven Methods with Multiple Overlapped Solutions (DDM-MOS [1]). In this article, we provide some details about them, as can be appreciated in Fig. 3.

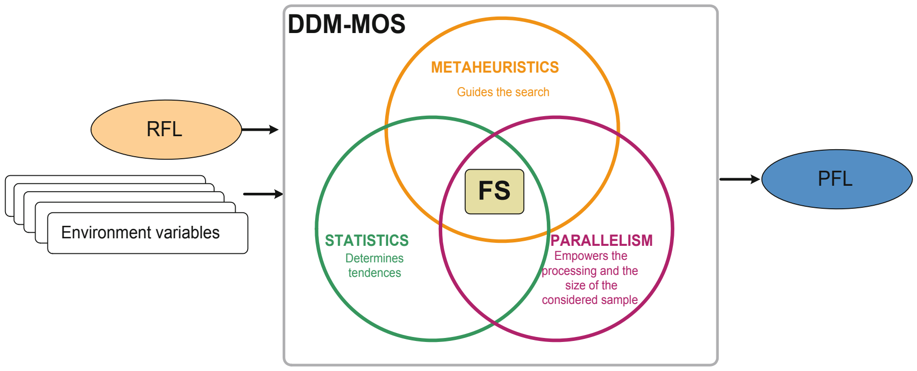

Fig. 3. Subjects involved by Data Driven Methods with Multiple Overlapped Solutions (DDM-MOS). **RFL**: Real Fire Line. **PFL**: Predicted Fire Line. **FS**: Fire Simulator.

On the one hand, the DDM-MOS take the initial state of the phenomenon as input parameter, in this case the real fire line (RFL). On the other hand, unlike what is observed in Fig. 2(b) in which the simulator takes as input a single combination of input parameters, the DDM-MOS take into account a certain number of combinations of values for the input parameters, valid within the possible search space. Each combination is called a *scenario* or *individual*, and is considered a possible solution to the problem. Since several individuals are considered at the same time, DDM-MOS are considered multiple solution methods. The set of individuals is called a *population*. As can be seen in Fig. 3, the operation of the simulator is enriched by the interrelation of Metaheuristics, Statistics and Parallelism. The metaheuristic allows to orientate the search within the extensive search space. Statistics allow to determine the tendency of the behavior of the individuals under consideration, by aggregating the results obtained for each of them; and this is why the DDM-MOS are considered as multiple overlapping solutions methods. Finally, the parallelism provides the necessary power to face the large amount of computation required by so many simulations in parallel and the consequent treatment of the results to yield a single prediction (PFL). The level of parallelism of DDM-MOS is also determined by the amount of populations considered: some DDM-MOS manage a unique population, whilst other operate on several populations in parallel, obeying to the Island Model [9]. In summary, these methods obtain predictions of the fireline based on the aggregation of multiple solutions, and focus on reducing the negative impact caused by uncertainty.

In particular, we have developed the ESSIM-DE method (Evolutionary Statistical System with Islands Model and Differential Evolution), a DDM-MOS which involves Differential Evolution as metaheuristic [8], and increases the level of parallelism by considering the Master/Worker pattern [5] and the Island Model [9] with several islands each responsible for the evolution of an independent population. In order to improve the performance of ESSIM-DE, we have defined and incorporated into the method a dynamic and automatic tuning strategy, which was presented in [11] and is extended in this work. Given that dynamic tuning is the core of this work, in the following we provide the general background necessary to understand this new version of the method: **ESSIM-DE(ldr)**. In general, the Tuning process [2,6] allows to calibrate, improve, adjust, or modify any critical aspect, bottleneck or factor that limits application performance. It consists of incorporating the phases of instrumentation, monitoring, analysis and tuning, during the execution of the program, as Fig. 4 illustrates.

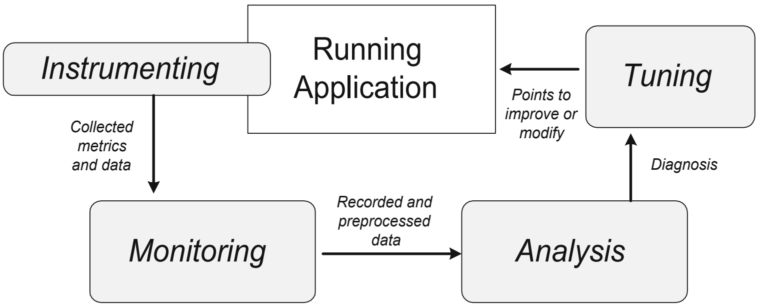
Instrumenting
Running Application
Tuning
Points to improve or modify
Collected metrics and data
Diagnosis
Recorded and preprocessed data
Monitoring
Analysis

Fig. 4. The tuning process and its phases operating on the improvement of the application

In the Instrumentation phase, the code of the application is augmented or annotated with sentences or record certain metrics of interest. These metrics, in general, correspond to some model of the behavior of the application, and define specific knowledge about the behavior of it. This knowledge can respond to mathematical performance models, fuzzy logic, ad hoc heuristics, among others. Along the Monitoring phase the metrics are detected and recorded, for later analysis and tuning. The Analysis phase is focused on the processing and analysis of the recorded metrics. The type of expert knowledge considered (mathematical performance models, fuzzy logic, ad hoc heuristics, etc.) defines how the analysis process is carried out to determine the tuning actions necessary to improve the performance of the application. Subsequently, in the Tuning phase the defined actions are applied in order to improve the critical aspects. In this work, all these phases are performed with no user intervention, at runtime. This is why the proposed tuning strategy is classified as automatic and dynamic. In particular, in this paper we present a performance model to dynamically and automatically tune one of the key stages of ESSIM-DE: the Optimization stage, which involves the Differential Evolution algorithm (more details are provided in the next section). This metaheuristic is a population optimizer (based on multiple solutions) and consists of an iterative process in which a population of individuals evolves. Each individual represents a possible solution to the problem.

Stagnation and/or premature convergence towards a local optimum constitute possible performance problems associated with Differential Evolution [4]. This paper establishes a criterion by which it is possible to tune the number of generations by which populations have to evolve in order to detect in advance a tendency to stagnation and/or premature convergence. The performance model uses statistical information from the trend of population dispersion, monitored in a distributed manner. The objective is to prevent the performance of the ESSIM-DE method from being affected by stagnation or premature convergence, both in the quality of the predictions and in the response time. Another objective is that the tuning can be carried out independently of the case of burning considered and the particular characteristics of the execution. We call ESSIM-DE(ldr) this version of the method, given that it is capable of tuning the **limit** of evolutionary generations for a given prediction step, according to the analysis of the tendency of the **distributions**; what is more, it is capable of **restart** the populations along the successive prediction steps. In [11], we presented a general view of ESSIM-DE(ldr) and all the related basis of it. In this article, we are extending some key concepts (such as uncertainty, DDM-MOS, differential evolution, architecture and operation of ESSIM-DE), we provide some complementary graphics, and we include two new experimental cases of study that allow for confirming the effectiveness of ESSIM-DE(ldr). Section 2 provides more details about the architecture and operation of ESSIM-DE. In Sect. 3 we explain the performance problem addressed by ESSIM-DE(ldr). In Sect. 4 we show the results obtained for five study cases. Finally, in Sect. 5 we present the main conclusions of this work.

2 Forest Fires Prediction with ESSIM-DE

To make the prediction of the fire front, ESSIM-DE divides the total development of the fire into different discrete time slots, called *simulation steps*. For the simulation step i, ESSIM-DE operates in the manner explained in Fig. 3. It takes as input the real fire line at instant $i-1$, and also takes a sample of candidate individuals which manages in several parallel populations. After processing them through Differential Evolution, parallelism and statistics, ESSIM-DE yields the corresponding prediction for instant i. The *Differential Evolution* (or DE) algorithm [8,13] is a population-based optimizer that uses multi-dimensional vectors to represent candidate solutions, also called population of individuals. Each dimension encodes a variable of the problem to be optimized and is represented with a value belonging to real numbers. The population evolves in different generations or iterations, in which the mutation, crossing and selection operators are applied to all of the individuals in the population. The mutation operator disturbs each individual in the current population, and for each one generates a new individual, called a "mutated vector". To do this, vector differences are applied between each individual considered and other randomly selected individuals. Subsequently, each mutated vector is submitted together with the original individual to the crossing operator, generating a new vector, called "trial vector". Finally, the selection stage determines the best candidate between the original individual and the trial vector. The one that has the best value regarding the function to be optimized will be the one that will survive the next generation. Below we present more details about the architecture and operation of ESSIM-DE. We also explain how the quality of an individual and a prediction is evaluated. In the next section we focus on the tuning component that allows to improve the performance of ESSIM-DE, constituting ESSIM-DE(ldr).

Architecturally, ESSIM-DE has a double hierarchy of processes that are organized in parallel islands and collaborate through migration, under the supervision of a Monitor process. Figure 5 presents how ESSIM-DE organizes its processes in a double hierarchy model, basing on the Master/Worker pattern [5] and the Island Model [9]. At a lower hierarchy level, each island manages a population of individuals, that is, different scenarios that constitute subsets of the total search space. On each island operates a main process called Master (green box), which is responsible for generating the initial population and applying the evolutionary operators of DE to improve the scenarios. In turn, each island has a group of Workers processes (light-blue "W" boxes) that are responsible for accelerating the prediction process, thanks to the parallel simulation of the different individuals or burning scenarios. In a higher hierarchy, a Monitor process (yellow box) coordinates the interaction between the Master processes of each island. These communicate with each other to exchange individuals from their islands through the Migration operator [9], a particular operator related to the evolutionary island models. The migration process promotes the global exploration of the search space represented by the populations of the different islands.

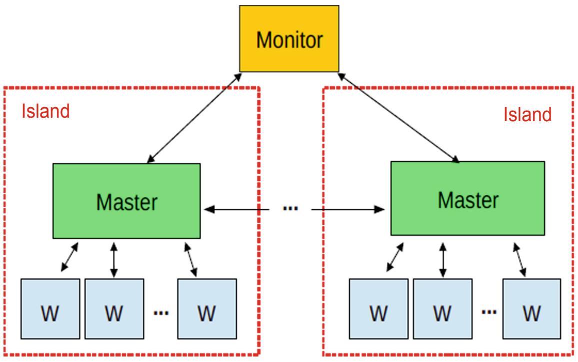

Fig. 5. Architecture of ESSIM-DE: double hierarchy of processes. **W**: Worker. (Color figure online)

Taking into account the ESSIM-DE architecture and the operation of the DE algorithm, we can now concentrate on the mode of operation of ESSIM-DE. The general operational scheme of ESSIM-DE is described in Fig. 6. For each prediction step four main stages operate: Optimization stage (OS), Statistical stage (SS), Calibration stage (CS) and Prediction stage (PS). Because ESSIM-DE uses a hierarchical process scheme, these four stages are subdivided to be cooperatively performed by the different entities of the process hierarchy: Master (M), Workers (W) and Monitor (Mon).

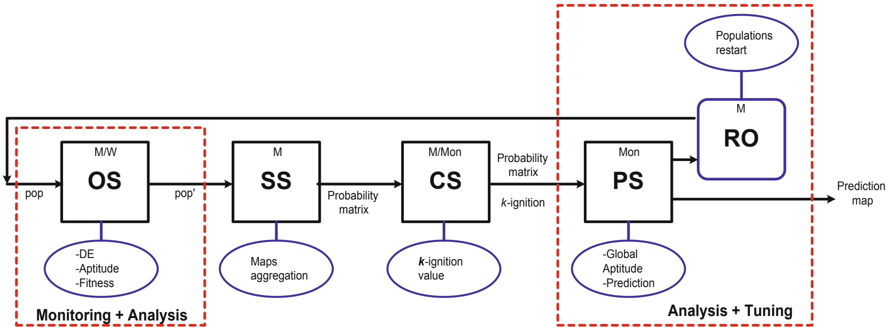

Fig. 6. General operation of ESSIM-DE. **OS**: Optimization stage. **SS**: Statistical stage. **CS**: Calibration stage. **PS**: Prediction stage. **RO**: Restart Operator. **DE**: Differential Evolution. **M**: Master. **W**: Worker. **Mon**: Monitor. Dotted line: tuning process in ESSIM-DE(ldr). (Color figure online)

In Fig. 6 can be seen the four stages involved in the method, of which we will focus on the **Optimization Stage** (**OS**), which is carried out between the Workers (W) and the Master (M) processes of each island. This stage allows a population of individuals to evolve based on the Differential Evolution algorithm. Each individual represents a combination of values for variables that determine the progress of the fire (the wind speed, direction and slope of the land, type of combustible material, humidity of the living combustible material, etc.). The Master process initializes the population (*pop*), applies the DE mutation and crossing operators to generate new candidate individuals, and distributes the individuals among the Workers. These use the current state of the fire, together with an individual to perform the simulation. Subsequently, each Worker evaluates the aptitude of the prediction obtained, weighing the accuracy of the simulation by comparing the cells burned in the real fire and the cells reached by the fire on the simulated map. The evaluation of the prediction quality is quantified based on a **fitness function**, which obeys to the Jaccard index [7], where the division between A and B is performed, being A the set of cells on the real map without the subset of cells burned before starting the predictive process, and being B the set of cells on the simulated map, without the subset of cells burned before starting the prediction (the subset of cells burned before starting the predictive process are not considered to avoid biased results). Therefore, the fitness value can be considered the percentage of coincidence between the map obtained from the simulation and the real map, and will be in the range $[0, 1]$: a value equal to 1 will represent a perfect prediction, and a value equal to zero would indicate the maximum error. Therefore, aptitude represents the percentage of coincidence between both maps. Subsequently, the following stages are carried out: Statistics (**SS**), Calibration (**CS**) and Prediction (**PS**), which collect the information and results obtained by all the islands in the **OS** stage, and based on this, they allow to obtain the global prediction of the state of the fire. As the Prediction stage finishes and the following fire step is going to start, ESSIM-DE also involves the Restart Operator (RO). This operator generates a new population (new search space). For more detail on the operation of the ESSIM-DE stages it is possible to consult [10].

3 Dynamic Tuning: Performance Model

In this section we concentrate on the performance model defined to enhance ESSIM-DE, constituting ESSIM-DE(ldr). For each prediction step, the Master process of each island determines when the evolution of its population finishes to continue with the next stages of the prediction process. In the OS stage of ESSIM-DE, the termination condition of the evolutionary cycle consists in determining a certain maximum level or limit of iterations. This condition has a double influence on the optimization process, since it limits the amount of evolutionary generations through which each population will evolve, and consequently determines the execution time of the evolutionary process. In part, these were the reasons why it was proposed to define the dynamic tuning process applied to the limit of evolutionary iterations.

In the definition of the tuning process we have considered two possible problems associated with the evolution process carried out by DE: premature convergence and stagnation. *Premature convergence* is the situation in which the population converges to a local optimum, due to the loss of diversity; for its part, *stagnation* is the situation in which the optimizer is not able to generate new solutions better than its predecessors, even when the population has a certain diversity [4]. The problem of population stagnation depends on the effective movements of the DE optimizer. When a new individual is generated, there is a movement in the search space. This movement is considered effective if the new generated individual has a better aptitude compared to his predecessor [13]. Among all the possible movements that are made in the population, some are effective, while others are not, and therefore the latter involve a computational effort in vain. To address the treatment of these problems, it was proposed to quantify two different metrics for the population:

– **Effective movements (EM metric):** quantifies the number of individuals that have been improved after an evolutionary cycle (better fitness value than that of their predecessor).
– **Population diversity (IQR metric):** quantifies the dispersion of the population (variability of the population distribution). It is computed based on the Interquartile Range [3] of the fitness values of individuals.

These metrics are proposed after an experimental analysis in which they were monitored, recorded and studied for different cases of controlled burning. In a first approximation, a fixed threshold value was used for each metric. However, analyzing the results presented in Fig. 7, it was possible to note that the graphic of the IQR metric is variable from case to case, throughout the iterations, even when the same seeds are involved in the different fire cases. This is because the distribution of the values of fitness is influenced by multiple factors (method convergence speed, mutation factor, crossing probability, migration rate, map dimensions, among others). For this reason, the proposed performance model uses the information of the dispersion of the population, considering the successive IQR values obtained throughout the evolutionary generations. The purpose is –according to the current burning case considered– to detect states with some tendency to stagnation and/or premature convergence, making use of recent information about fitness distributions.

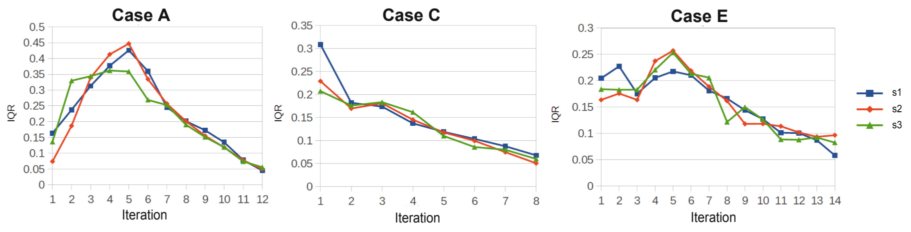

Fig. 7. The distribution of IQR is dependent of the study case. The same three seeds were used for the three illustrated cases: *s1*, *s2* and *s3*.

To achieve this, it was proposed: (*i*) to record the minimum IQR value obtained; (*ii*) to update such minimum IQR value throughout the iterations; and (*iii*) to compare it with respect to the IQR value obtained in the current iteration. This idea is exemplified in Fig. 8.

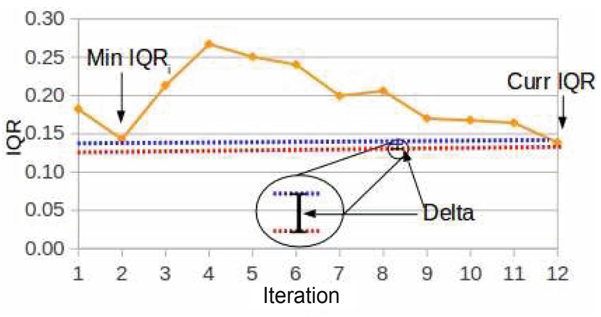

Fig. 8. Example: how is *Delta* determined.

The x axis represents the evolutionary generation (iteration) whilst the y axis represents the registered IQR value. The graph shows the minimum IQR recorded (*MinIQR*) and the IQR of the current iteration (*CurrIQR*). If the difference between both of them –called *Delta*– is very small, it means that the population has achieved a very similar distribution (in terms of fitness values) to that achieved in any of the previous iterations. Defining *Delta = (MinIQR − CurrIQR)*, the tuning criterion is defined by the Eq. 1 in which it is verified that the tendency of IQR is decreasing (*Delta* $>= 0$) and Delta is bounded by a certain small *thresholdDELTA* value.

$$Delta \leq thresholdDELTA \quad \wedge \quad Delta \geq 0 \quad \wedge ME \leq thresholdME \qquad (1)$$

For its part, the value of the ME metric is also considered, required to analyze whether the optimizer still has effective movements to be made on the search space, bounded to a certain threshold value *thresholdME*. In ESSIM-DE, the Eq. 1 is computed in each iteration throughout the OS, according to the current population distribution, for each burning map and particular conditions of execution. The proposal aims to improve response times with respect to the method without tuning, since the new condition used as a cut-off criterion is a specific property of each population, which avoids unnecessary cycles. The tuning process is included in Fig. 6 (see the boxes in red dotted lines). The aptitude values are recorded along the Monitoring phase and the metrics are computed in the Master Analysis phase. When analyzing each iteration, each Master sends the metrics of its island to the Monitor (Mon) process, who makes an aggregation of the values received in its Analysis phase, and determines if there is any island with a tendency to stagnation and/or premature convergence. If some island evaluated as true the Eq. 1, the Monitor decides: (a) to stop the evolution of all the islands; (b) to make the prediction by aggregating the information of all of them; and (c) to resume the next prediction step.

4 Experiments

In [11] the proposal was validated based on experimentation with three cases of controlled burning belonging to the SPREAD project [12] (see Table 1, cases A, B and C). In this article we extend such experimentation, including two additional cases also from SPREAD project (see Table 1, cases D and E). For each case of burning, the experiment was performed to compare two different scenarios:

– Fire prediction with ESSIM-DE (with no tuning process).
– Prediction of the fire with ESSIM-DE(ldr) (ESSIM-DE empowered by dynamic tuning).

It should be remembered that **ldr** is the acronym with which we call the method with the process of dynamic tuning of the iteration limit and with the population restart operator. Each experiment was executed 10 times with different seeds. We present the average of the results obtained along the ten executions. The island model was configured with 5 islands, of 7 Workers each. The migration process involves 20% of the individuals in the population, and it is carried out in each iteration. The size of each population was defined as 200 individuals. For ESSIM-DE and ESSIM-DE(ldr) the same configuration of the evolutionary parameters was used: crossover probability 0.3, mutation factor 0.9, and binomial crossover. The value used as the *thresholdME* was set at 20%. The value established for the *thresholdDELTA* parameter was 10^{-3}.

Table 1. Study cases: dimension, slope, start time, end time, and increase.

Case	Width (m)	High (m)	Slope (degrees)	St. T. (min)	Incr. (min)	End T. (min)
A	89	109	21	2	2	14
B	60	90	6	2	2	10
C	89	91	21	2.5	2.5	12.5
D	95	123	21	2	2	12
E	75	126	19	3	1	9

The results obtained are presented in Fig. 9, 10, 11, 12 and 13. Two graphics are included in each figure. The graph (a) represents the fitness averages for each prediction step, obtained by ESSIM-DE(ldr), and compared with respect to the ESSIM-DE with no tuning. Graph (b) shows the distribution of fitness values obtained for each prediction step and method, which allows analyzing the dispersion of the results and the robustness of the method in terms of obtaining different solutions under different executions. Table 2 shows the average runtime values obtained.

Figure 9 shows the results obtained for the case of burning A of Table 1. The fire starts in minute 2 and lasts for 14 min, constituting a calibration step and five prediction steps (calibration: minute 2 to 4; prediction steps: minutes 4 to 6 first step, minutes 6 to 8 second step, minutes 8 to 10 third step, minutes 10 to 12 fourth step, and minutes 12 to 14 fifth step). In general, it can be seen from graph (a) that ESSIM-DE obtains low quality of prediction, especially in the prediction step 3. Figure 9(b) shows how step 1 and step 5 present a wide variability of the results obtained with ESSIM-DE, whilst for ESSIM-DE(ldr) there is less distribution in all prediction steps, which indicates robustness in terms of obtaining solutions under different executions. In general, ESSIM-DE(ldr) gets better performance, with average of fitness greater than 0.7 in all the prediction steps, and with a reduction of the execution time of approximately 38%. This reduction in time is associated with the ability of ESSIM-DE(ldr) to prematurely detect the tendency to stagnation and/or premature convergence, avoiding unnecessary cycles, and therefore, achieving less response time. Execution time reductions are very relevant in the context of prediction methods, allowing decisions to be made in advance of the fire. It is important to note that the reduction of time not only depends on the speed with which the stagnation and premature convergence is detected, but also depends on the characteristics of the burning case and the

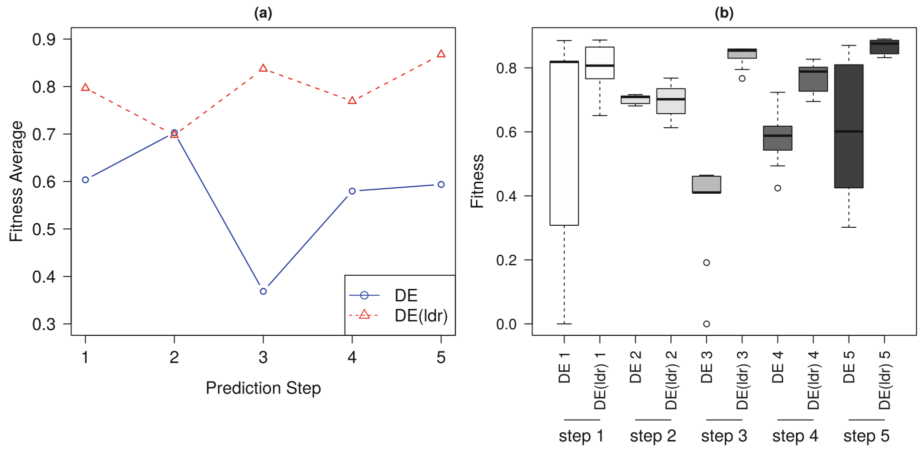

Fig. 9. Case A. (a) *fitness* average for ESSIM-DE and ESSIM-DE(ldr). (b) Distribution of the fitness values obtained in the successive prediction steps.

size of the considered map, which influences the overall behavior of the system in the different stages.

Table 2. Average execution times and percentage reduction.

Case	ESSIM-DE	ESSIM-DE(ldr)	Time reduction
A	3540	1292	63.5%
B	1696	446	73.7%
C	2332	1438	38.3%
D	3582	1265	64.7%
E	2406	702	70.8%

Figure 10 shows the results obtained for the case of burning B of Table 1. The fire consists of three prediction steps (minutes 4–6, 6–8, 8–10). It can be seen from the Fig. 10(a) that ESSIM-DE has a decreasing trend in the quality of the predictions, with a low percentage of coincidence in the third prediction step, less than 60%. Although in the first prediction step can be observed good quality (higher than 0.8) and low distribution (see the results obtained in Fig. 10(b), step 1 for ESSIM-DE), ESSIM-DE(ldr) improves all prediction steps, with fitness averages greater than 0.85 in all steps. In addition, in this case ESSIM-DE(ldr) obtains a gain in runtime, with a significant reduction of approximately 73%.

Figure 11 shows the results obtained for the case of burning C of Table 1. The fire consists of three prediction steps. It can be seen from the Fig. 11(a) that ESSIM-DE(ldr) obtains a better fitness average in the first and third prediction step, while ESSIM-DE obtains a quality close to 0.9 in the second prediction step.

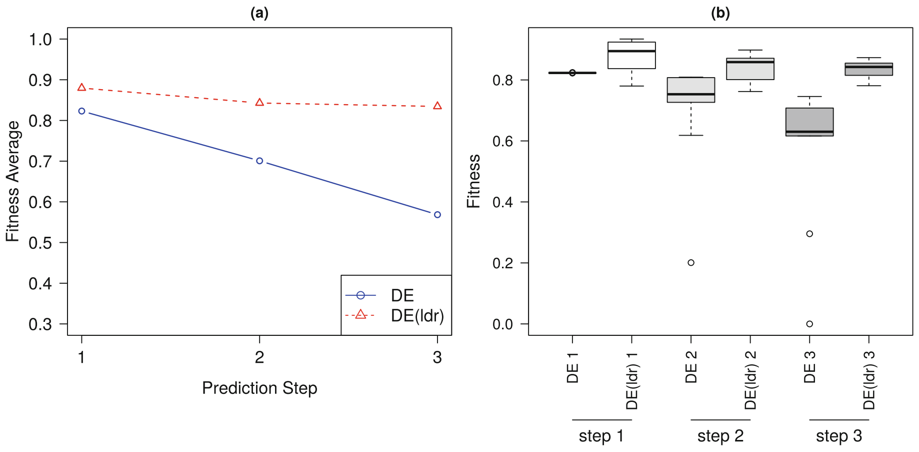

Fig. 10. Case B. (a) *fitness* average for ESSIM-DE and ESSIM-DE(ldr). (b) Distribution of the fitness values obtained in the successive prediction steps.

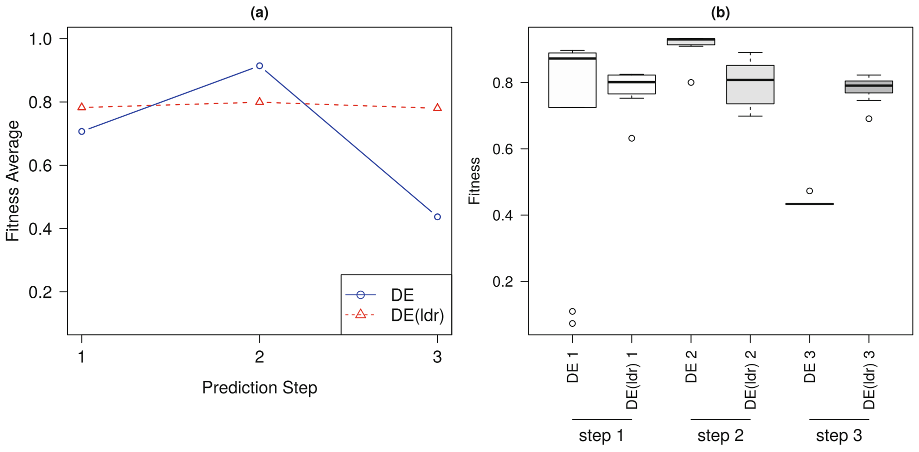

Fig. 11. Case C. (a) *fitness* average for ESSIM-DE and ESSIM-DE(ldr). (b) Distribution of the fitness values obtained in the successive prediction steps.

It is important to highlight in this experiment that ESSIM-DE(ldr) significantly improves the quality of the prediction obtained in the third prediction step with respect to ESSIM-DE, achieving a percentage of coincidence with the real fire close to 80% of coincidence (fitness close to 0.8). In addition, it can be seen from the results in Table 2 that ESSIM-DE(ldr) reduces the execution times approximately by 63% compared to ESSIM-DE.

Figure 12 shows the results obtained for the case of burning D of Table 1. The fire consists of four prediction steps (minutes 4–6, 6–8, 8–10, 10–12). It can be seen from the Fig. 12(a) that ESSIM-DE present good results. However,

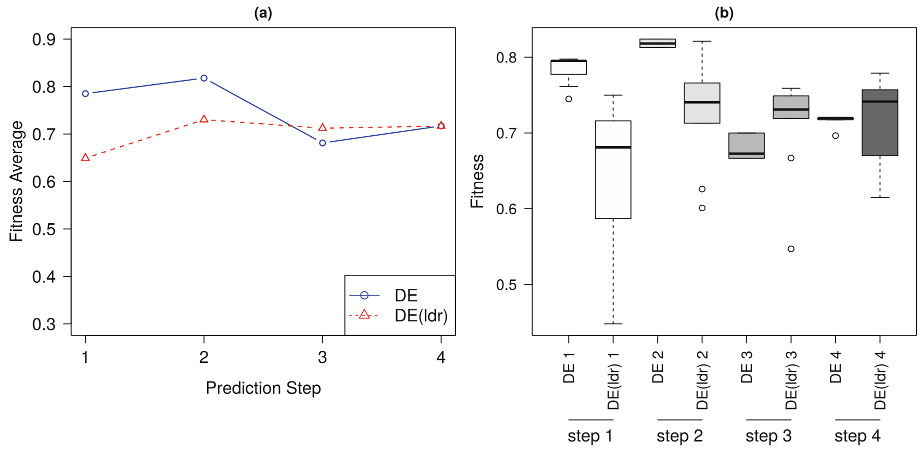

Fig. 12. Case D. (a) *fitness* average for ESSIM-DE and ESSIM-DE(ldr). (b) Distribution of the fitness values obtained in the successive prediction steps.

it has a decreasing trend in the quality of the predictions, showing a loss in the percentage of coincidence in the third prediction step, less than 70%. Even when ESSIM-DE(ldr) overtakes ESSIM-DE only in the third prediction step, it is possible to note that the quality of ESSIM-DE(ldr) is maintained around a 70% of coincidence with the real fire situation along the four prediction steps, and from Table 2 it is possible observe that ESSIM-DE(ldr) obtains a gain in runtime, with a significant reduction of approximately 65%.

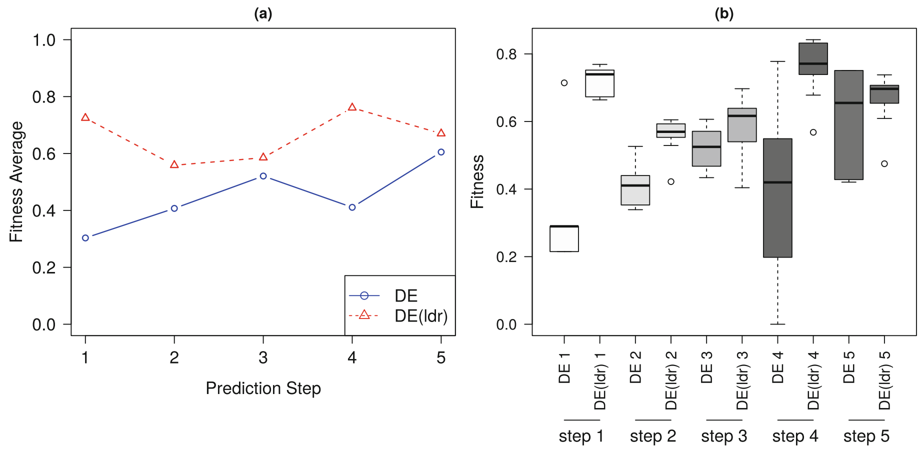

Fig. 13. Case E. (a) *fitness* average for ESSIM-DE and ESSIM-DE(ldr). (b) Distribution of the fitness values obtained in the successive prediction steps.

Figure 13 shows the results obtained for the case of burning E of Table 1. The fire consists of five prediction steps. It can be seen from the Fig. 12(a) that ESSIM-DE(ldr) improves the average fitness in every prediction step, whilst ESSIM-DE obtains a quality at most close to 0.6 in the last prediction step. It is important to highlight in this experiment that ESSIM-DE(ldr) significantly improves the quality of the prediction obtained in the first and fourth prediction steps with respect to ESSIM-DE, achieving a percentage of coincidence with the real fire over 70% of coincidence, in contrast to the 30% or 40% reached by ESSIM-DE without tuning. Figure 13(b) shows that in this case, the fitness distributions for ESSIM-DE(ldr) do not present major variability, confirming the robustness of the method. In addition, it is important to note that ESSIM-DE(ldr) reduces the execution times approximately by 70% compared to ESSIM-DE, as shows Table 2.

In general, considering all studied cases, the quality of the predictions obtained by ESSIM-DE(ldr) was improved regarding ESSIM-DE (the method without tuning). With respect to response times, ESSIM-DE(ldr) obtained the results with a significant reduction in the execution time: between 38% and 73% savings with respect to ESSIM-DE. Both improvements are associated with the early detection of stagnation and premature convergence, avoiding unnecessary cycles to the Evolution Differential optimizer, and obtaining better individuals that contribute to the overall solution. It is important to highlight that the use of a mathematical/statistical model as the basis of expert knowledge for the dynamic tuning process allows for the reduction of the time taken in decision making, unlike other strategies with higher computational cost, such as search, iterative processes, logic diffuse, neural networks, approximations, among others. This is because the analysis and tuning decisions are based only on the evaluation of mathematical expressions. In the context of prediction methods such as ESSIM-DE, these time reductions are essential to obtain short-term predictions.

5 Conclusions

In this paper, we proposed a performance model to dynamically tune the ESSIM-DE method, a general method for the uncertainty reduction in the prediction of spread phenomena. We proposed to incorporate the tuning process in order to improve the ESSIM-DE performance both in terms of quality of the obtained predictions, and the response time. The defined performance model uses the information of the dispersion of the population in the succession of values obtained throughout the evolutionary generations, in order to detect tendency to stagnation and/or premature convergence of the population. The state and distribution of the search space (populations) is computed at two different levels: (*i*) in a distributed manner, given that each island analyzes the current distribution of its own population, for each burning map and particular conditions of execution, and (*ii*) in a centralized manner as the Monitor process analyzes the search space in a global view. The results obtained have shown that the proposal improves both the quality and the response times with respect to the method without tuning, since the new condition used as a cut-off criterion is a specific property of each population, which avoids unnecessary cycles once certain level of convergence has been detected. As future work, we propose to tune the threshold values of the metrics defined in the performance model, and consider other parameters of the method that may be potentially tunable, such as the population size, or the parameters related to the parallel/distributed model.

References

1. Bianchini, G., et al.: Wildland fire growth prediction method based on multiple overlapping solution. J. Comput. Sci. **1**(4), 229–237 (2010)
2. Caymes-Scutari, P., Bianchini, G., Sikora, A., Margalef, T.: Environment for automatic development and tuning of parallel applications. In: International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, pp. 743–750 (2016)
3. Healey, J.: The Essentials of Statistics: A Tool for Social Research. Thomson/Wadswort, Belmont (2007). ISBN 9780495009757
4. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm, In: Ošmera, P. (ed.) Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing, pp. 76–83. Brno, Czech Republic (2000). http://www.lut.fi/jlampine/MEND2000.ps
5. Mattson, T., et al.: Patterns for Parallel Programming. Addison-Wesley, Boston (2004)
6. Naono, K., Teranishi, K., Cavazos, J., Suda, R. (eds.): Software Automatic Tuning: From Concepts to State-of-the-Art Results. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-6935-4
7. Real, R., Vargas, J.M.: The probabilistic basis of Jacard's index of similarity. Syst. Biol. **45**(3), 380–385 (1996)
8. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. NCS. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-31306-0
9. Talbi, E.: Metaheuristics: From Design to Implementation. Wiley, Reading (2009)
10. Tardivo, M.L., Caymes-Scutari, P., Méndez-Garabetti, M., Bianchini, G.: Optimization for an uncertainty reduction method applied to forest fires spread prediction. In: De Giusti, A.E. (ed.) CACIC 2017. CCIS, vol. 790, pp. 13–23. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75214-3_2
11. Tardivo, L., Caymes-Scutari, P., Bianchini, G., Mendez-Garabetti, M.: Sintonización Dinámica del Método Paralelo de Predicción de Incendios Forestales ESSIM-DE. In: Proceedings XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019), pp. 115–124. UniRio Editora, Río Cuarto (2019). ISBN 978-987-688-377-1
12. Viegas, D.X.: Project Spread Forest Fire Spread Prevention and Mitigation (2004). https://cordis.europa.eu/project/rcn/60354/factsheet/fr. Accessed Sept 2019
13. Yang, M., Li, C., Cai, Z., Guan, J.: Differential evolution with auto-enhanced population diversity. IEEE Trans. Cybern. **45**, 302–315 (2015)
14. El País. https://elpais.com/elpais/2019/09/11/album/1568221457_486259.html#foto_gal_1
15. Radio Gráfica. https://radiografica.org.ar/2020/01/08/australia-la-ciencia-predijo-los-incendios-y-la-politica-lo-ignoro/