Journal of Computational Science 25 (2018) 76–88

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Using efficient parallelization in Graphic Processing Units to parameterize stochastic fire propagation models

Mónica Denham [a,b,*,1], Karina Laneri [a,c,*,1]

[a] Consejo Nacional de Investigaciones Cientícas y Técnicas (CONICET), Argentina
[b] Laboratorio de Procesamiento de Señales Aplicadas y Computación de Alto Rendimiento, Sede Andina, Universidad Nacional de Río Negro, Argentina
[c] Física Estadística e Interdisciplinaria, Centro Atómico Bariloche, Río Negro, Argentina

ARTICLE INFO

Article history:
Received 4 August 2017
Received in revised form
22 November 2017
Accepted 13 February 2018
Available online 14 February 2018

Keywords:
Forest fire model
Forest fire spread simulations
GPU

ABSTRACT

Wildfires are a major concern in Argentinian northwestern Patagonia and in many ecosystems and human societies around the world. We developed an efficient cellular automata model in Graphic Processing Units (GPUs) to simulate fire propagation. The graphical advantages of GPUs were exploited by overlapping wind direction, as well as vegetation, slope, and aspect maps, taking into account relevant landscape characteristics for fire propagation. Stochastic propagation was performed with a probability model that depends on aspect, slope, wind direction and vegetation type. Implementing a Genetic Algorithm search strategy we show, using simulated fires, that we recover the five parameter values that characterize fire propagation. The efficiency of the fire simulation procedure allowed us to also estimate the fire ignition point when it is unknown as well as its associated uncertainty, making this approach suitable for the analysis of fire spread based on maps of burnt areas without knowing the point of origin of the fires or how they spread.

1. Software availability

Name of software: CUDAFires
Developers: Karina Laneri, Mónica Denham
Contact address: John O'Connor 181, San Carlos de Bariloche, Río Negro, Argentina.
Email: karinalaneri@gmail.com, mdenham@unrn.edu.ar
Availability and Online Documentation: All the source code is available with a simple request to the authors.
Year first available: 2016
Hardware required: CUDA devices with compute capability 1.1 and above.
Software required: CUDA SDK
Programming language: C/CUDA
Program size: 3.0 MB.

2. Introduction

Wildfire affects many ecosystems and human societies around the world. Global environmental change is apparently producing changes in fire regimes, underscoring the importance of understanding how fire spreads in different ecosystems [1]. Topography, wind velocity and direction, fuel properties, air and fuel moisture, fire history and topographic aspect are some of the variables that can influence how fire will propagate at a specific site [2–5]. In the Argentine Patagonia, forest and shrubs have high fuel loads (100 tons/ha or more of rotting wood, fallen trees, etc.). The relative humidity of this coarse material achieves the critical level of 20% during dry seasons and weather conditions maximize the fire risks during summer months.

Several models and simulators have been developed, i.e. BEHAVE, FARSITE, fireLib, PROMETHEUS [6,7] which implement related models of fire spread based on the physics of fire behavior [8–10]. However these forest fire simulators use environmental variables as inputs which are usually difficult to quantify for many specific sites like Patagonia. Probabilistic models use instead, a fire propagation probability from cell to cell that depends on potentially relevant covariates, e.g. humidity, wind, slope. Therefore, the advantage of probabilistic models in our region relies in the fact that fuel models, that are still not available for the Patagonia region, are not necessary to define fire propagation. For instance, wind speed and direction are hard to measure in a fire context because fire itself produce strong wind gusts. Fuel and air humidity change because of fire. Fuel type is usually considered as the average of typical vegetation of the area, while weather and topography are generally interpolated values that can be calculated for example using WindNinja [2,5,11,4].

* Corresponding authors at: Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina.
E-mail addresses: mdenham@unrn.edu.ar (M. Denham), karinalaneri@gmail.com (K. Laneri).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.jocs.2018.02.007
1877-7503/

The cellular automaton (CA) approach was efficiently used by Russo et al. [12] to run thousands of simulations to build hazard maps that have been shown to be robust and efficient in predicting fire spreading behavior in several cases. Some stochastic spatial propagation models where fire contagion occurs from cell to cell in a lattice were successfully used to reproduce fire data of semi-arid mountain systems in USA as well as of Northern Patagonia Andean region [13,14]. In these models, the propagation probability from cell to cell is calculated from several covariates that represent the environment in which propagation takes place [13], instead of using a function based on the physical principles of heat propagation [8]. The associated parameters of the model can vary in a range of acceptable values. One possible method for the exploration of parameter space is the Genetic Algorithm (GA) [15], which has proven to be useful for Individual Based Model (IBM) calibration [2,5].

As a benchmark, the model used in [13] was run with SELES [16] and the performance of one hundred thousand simulations of 9 fires with known ignition point fitted using Approximated Bayesian Computation (ABC), took about 10 days on a PC, running on Intel(R) i7-4770K CPU. This method requires a large number of simulations to explore the parameter space, for which high performance computing can be specially useful. Often, millions of simulations need to be compared with a data set of reference, using a given measure of similarity between simulated and observed data (or fitness in the GA sense). This stage of the process can become a bottleneck of the application.

In this work, we develop a high performance application, parallelized using General Purpose Graphical Processing Units (GPGPU) architectures and CUDA programming model for the type of fire propagation models presented in [13]. We propose a Genetic Algorithm (GA) in order to improve the search on parameter space. This methodology was previously tested [5,2,4] for forest fire parameter estimation and was proved to be very effective to accelerate convergence to the best set of parameters. Alternatively, a more simple brute force method based on Monte Carlo sampling (MC) can be used to find the best set of parameters. Albeit computationally more simple, this method is less efficient in terms of computational time, as we will show later.

Recent papers [17–19] present forest fire model subroutines implemented in parallel using GPUs (for example, BEHAVE model included in fireLib simulator). These works have shown that using GPUs, accelerations ranging from 64× to 229× were reached and therefore a real-time simulator can become possible, which would be a vast improvement on the current state of the art. In our work we included these improvements and in addition we developed and implemented a fitting methodology to determine fire propagation parameters.On top of the runtime improvement, interactive simulations on GPUs are actually used as operational prediction tools for fire propagation [20].

The work we present here has two main goals: on the one hand the development of a high performance modular open-source application to accelerate fire propagation prediction and scale-up fire propagation applications. On the other hand, the accurate estimation of fire propagation parameter values from maps of burnt areas.

An important limitation of the approach presented by Morales et al. [13] is that the original ignition point has to be known in order to perform the estimation of fire propagation parameters. Very recent works have developed alternative ways to determine the fire ignition point from only the final fire perimeter [21]. With an efficient simulation procedure like the one presented in this work, we are able to recover all five parameters that characterize our fire propagation using only the information of the final burnt area corresponding to a single fire. However as expected, there is a limited number of propagation parameters that can be accurately determined from the information of only one fire scar and considering the ignition point as an additional parameter may generate overfitting. Besides these limitations that will be discussed, the feasibility to determine the ignition point and the associated errors, constitutes a valuable information when mapped fires have unknown starting points.

Even though we do not take into account the fire front advance, our application was built with the potential to fit fuel consumption times as the fire spreads, allowing to test in the future different hypothesis on the inflammability of different vegetation types.

3. Related work

In a recent paper [13] the parameterization of an stochastic Cellular Automata Model (CA) is done using SELES (Spatially Explicit Landscape Event Simulator) [16]. SELES simulates fire spread over land surface performing millions of simulations, allowing users to concentrate on the model features and not on the cellular automaton itself. The huge amount of simulations needed is the bottle neck of the application. The parallelization of that work is our main objective, developing a high performance application in CUDA-C to be used instead of SELES.

The Genetic Algorithm (GA) methodology was previously used [5,2] for forest fire parameter estimation and was proved to be very effective to accelerate convergence. It was applied to fires of Portugal and used in combination with fireLib.

Very recent papers [17,18] describe the implementation of three fire spread models both sequential and parallel. Accelerations ranging from 64× to 229× were reached. Their results show the potential for a forest fire simulator implemented using the GPU as underling architecture.

Russo et al. [12] implemented a computational approach to build hazard maps on the basis of a CA model which has been shown to be robust and efficient in predicting fire spreading behavior in several cases.

Arca et al [22,23] addressed a challenging optimization problem using GPGPU modifying the type and amount of fuel to reduce fire spread and damages, showing that GPUs are convenient in order to accelerate an optimization problem in a large search space.

In [24] a Monte Carlo method is used to find burn probability maps (BPMs) overlapping thousand of CA simulations that are carried out on GPUs under different weather scenarios and ignition locations. This work presents an CA parallelization similar to our work.

In [19], CUDA-C is used to implement fireLib subroutines. Simulated fires are used as reference so the exact solution is known allowing to analyze system errors. Their approach is similar to ours, an CA is mapped on the GPU where several threads perform the calculus over different data cells in a SIMD (Single Instruction Multiple Data) fashion. One thread per cell is created and each thread calculates if it is reached by fire or not, depending on the states or its neighboring cells and its own state.

Some authors [25,26] claim that the raster approach (including the CA model) is more efficient than the vectorial approach. On the one hand, vector implementation treats the fire perimeter as a closed curve, discretized through a number of points. Each point spread the fire based on its local conditions (fuel, weather, slope, etc.) according to the fire propagation model. Then, the new perimeter is formed by the union of the outer shape of all individual fires. The new fire line is discretized and expanded in the same form. On the other hand the raster approach spread fire over a mesh of contiguous cells that can be inactive (burnt or not burning) or active (burning). Each cell has its own state (topography, fuel, weather, etc) and the fire spreads from a cell to its neighbours based on a set of rules. In work [25] an CA is proposed where

typical angular fire shapes are mitigated with the modification and improvement of ROS equations: five correction factors are added in order to improve Rothermel model. The problem of finding the optimum values for these correction factors that minimize an objective function is a non trivial optimization problem.

4. Methods

4.1. Forest fire spread simulation

In our model, fire spreads to neighboring cells according to the following probability:

$$p = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 I_f + \beta_2 \psi + \beta_3 \omega + \beta_4 \sigma))} \quad (1)$$

where p is the propagation probability that is a function of fuel type (forest or shrubland), aspect, relative wind direction and slope of the target cell. This formula is inspired on the one used in [13], but with the numerator set equal to one to accomplish with a probability requirement to be between 0 and 1.

In Eq. (1), I_f is a vegetation indicator variable equal to 1 for cells occupied by forest and 0 otherwise, so that β_0 is the baseline fire-propagation probability for shrubland cells and β_1 measures the difference between shrubland and forest. For instance, if fire is less likely to propagate in forest than in shrublands, then β_1 should be less than zero.

Aspect has an important effect in the studied area due to its effects on fuel moisture: sites facing towards the NW have the driest conditions and sites facing SE are the moistest. Thus for landscape cells with a slope greater than 5 degrees, the relative aspect ψ is calculated as: $\psi = \cos(\theta - 315^\circ)$. Then the relative aspect takes values of 1 for NW facing sites and -1 for those facing SE.

As in [13], wind direction for each cell was derived from the average dominant wind direction during the fire season and modified from topography using WindNinja [27]. Then, for each of the 8 cells neighboring an ignited cell $\omega = \cos(\phi_w - \phi_c)$ was calculated where ϕ_w is the wind direction in the target cell and ϕ_c corresponds to the angle between the ignited cell and the neighboring cell. Then, ω takes values of 1 if the landscape cell is downwind from the ignited cell and -1 if it is upwind from it.

Slope is calculated based on the digital elevation map (DEM) and is ignored whenever the target landscape cell is at a lower elevation or when slope is zero. The normalized slope ($\sigma = slope/100$) is used when the target cell is higher in elevation than the burning cell.

Therefore, parameters β_2, β_3 and β_4 modify the propagation probabilities according to aspect (ψ), wind direction (ω), and slope (σ) respectively, being the upper limit for this logistic function equal to 1. The β_i values are the same for all cells since they are characteristics of the whole fire propagation process, being the coefficients $(\omega, I_f, \psi, \sigma)$ the ones that change from cell to cell according to the environment.

The model is defined on a two-dimensional lattice of length L that represents the landscape, with sites in four possible states, burnt and active, burnt and inactive, burnable and impossible to burn. A burnt cell gets inactive after a given time that depends on the vegetation type of the cell. We set both times for forest and shrubland equal to one time step of simulation because these were the conditions used in [13] and allow us to compare results. However, as this is a strong assumption, we also tested results for several fuel consumption times for forest and shrubland. We show that we are also able to identify the set of input parameters. In Fig. C.13 of Appendix C we show the particular case of 7 burning time steps for forest and 2 for shrubland, but for other times combination we obtain similar results.

To test our method, we generated a synthetic (simulated) fire with a set of known parameters extracted from [13], that from now on, we will refer to as the "reference fire".

Fire ignition and propagation were set according to the following rules:

- For the reference fire, the ignition point was set fixed at an arbitrary site. However, for simulations we considered two cases: fire starting from the same point than the reference fire or fire starting from a random ignition point within the reference fire burnt area.
- All cells are checked in parallel and a given target cell is burnt according to the probability $1-(1-p_i)^8$ with $i = 1, 2, \ldots, 8$ the number of neighbors cells and p defined in Eq. (1).
- Once the fire propagation stops by itself, we compute the fitness according to Eq. (2) (Section 4.2). We got sure that the fire does not reach the edges of the lattice.
- We ranked all simulations according to their fitnesses. We selected the ensemble of parameters with fitness smaller than a cutoff value. The cutoff was chosen as the value from which the histogram is stationary.

For model calibration, we took as the reference a fire produced with a set of parameters extracted from [13]. We then produced one million simulations with different sets of parameters that were compared to the reference in order to recover the known set of parameters.

Fire propagation occurs on the substrate shown in Fig. 1 that corresponds to the Northern Patagonia Andean region at 40°–41°30′S latitude. Wind direction is highly constant during the fire season with 78% of day coming from NW or WNW. Terrain is mountainous, with valleys and steep slopes formed by glacial activity. Average annual precipitation varies approximately from 800 mm to 3000 mm (eastern limit and western areas respectively) and approximately 60% of the total precipitation falls during the winter season (from May to August). Spring and summer months are typically dry [28].

Raster maps for slope, aspect, vegetation and wind direction are used as inputs. Maps metadata allow them to be geographically referenced in the physical real space. In our example, maps were 801×801 cells, with 30 m $\times$ 30 m cell resolution. Fig. 1 shows some of these maps.

4.2. Model fitting

A Genetic Algorithm (GA) was used to search for combinations of input parameter values that generate final burnt areas similar to that of the reference fire. According to the GA methodology, populations are formed by individuals (or combinations of β_i values from Eq. (1)) that are the propagation parameters of the fire simulator. As an alternative more simple methodology, we implemented a brute force Monte Carlo algorithm, explained in Appendix B, that allowed us to have a benchmark to compare simulation times between the Genetic Algorithm and a very simple brute force Monte Carlo approach.

A measure of similarity between simulation maps and reference map is needed to select the best ensemble of simulations that determine parameters and associated uncertainty.

Two fitness functions were used, both of them based on the difference between simulated and reference fire maps (Eq. (2) and the other one is presented in [13]). The second fitness function is presented in detail in [13] and was used to compare our results against the ones obtained by Morales et al. using SELES. The results using these two fitness functions were very similar giving the same estimated parameters and simulation accuracy. Moreover, the fitness function proposed by Morales et al. takes into account the areas of

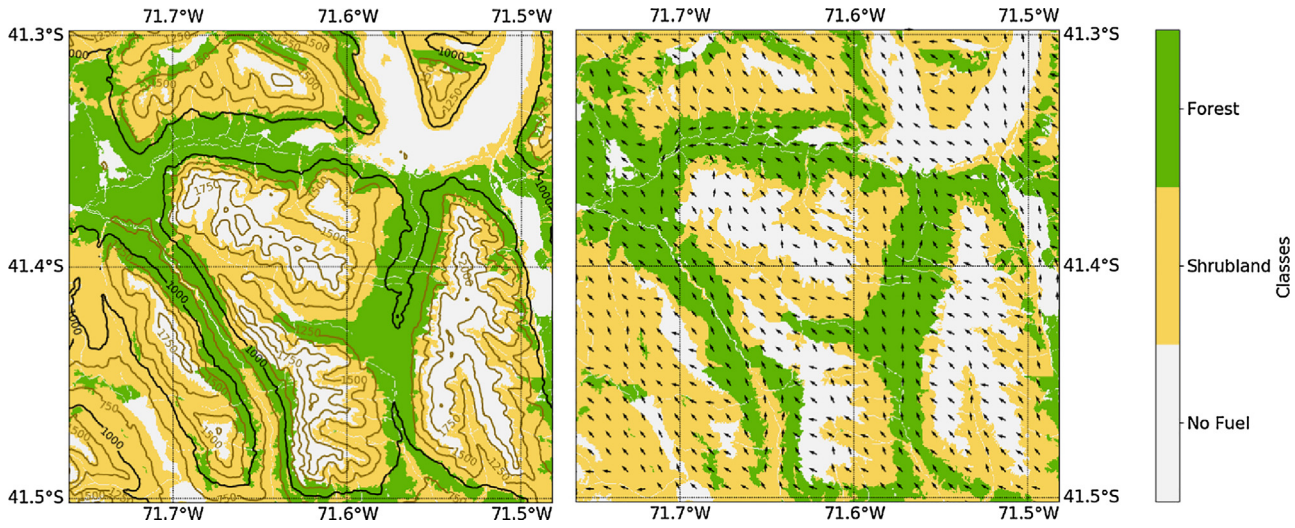


Fig. 1. Landscape layers where the fire propagation takes place. Level curves shows terrain elevation and wind speed was considered as the average wind in the region for fires seasons. No fuel cells indicates lakes, roads and firewalls.

different fuel types being difficult to generalize to more fuel types, adding complexity to the algorithm. Therefore, we will show in this work the results using Eq. (2).

The "reference fire" of our synthetic experiment was compared with simulations using the fitness function of Eq. (2). Given that our fire propagation is stochastic, the same set of parameters and ignition point will give rise to different final fire burnt areas, for different random seeds.

The fitness function (Eq. (2)) implemented and used in [5] denotes the discrepancy between real and simulated fires, counting burnt overlapped cells. This function is 0 when simulation and real maps are perfectly overlapped and does not discriminate cells by fuel type.

$$\delta = \frac{(A \cup A^*) - (A \cap A^*)}{A} \tag{2}$$

In Eq. (2), A denotes cells of reference map (real fire map) and A^* denotes cells of simulated fire map. The $\cup$ symbol means the amount of cells in the union of burnt cells (burnt cell in simulated map or burnt cell in the reference map), $\cap$ is the total of burnt cells in both maps (the intersection of the reference and simulated map). The numerator of Eq. (2) is the difference between real and simulated map. The denominator is for normalizing that difference with respect to the size of real fire area and accounts for the amount of burnt cells in the real fire map. Simulations were ranked according to their fitness and the corresponding parameter values were used to estimate parameters and associated uncertainties. With the best ranked simulations we plotted the histograms of β_i values that allowed to determine the uncertainty associated with the fitting methodology. Simulations were parallelized in order to take advantage of actual hardware architectures like graphic cards. Next section presents the CUDA C parallel implementation of our application.

4.3. CUDA C implementation

Our parallel application was developed in CUDA C for General Purpose Graphical Processing Units (GPGPUs). These graphic cards are a massively parallel platform that offer high computational power [16,29–31].

Our approach is based on the parallelization of simulation steps: a thread per matrix cell is launched in order to compute if a (target) cell will be ignited or not. Each thread calculates Eq. (1) for each of its neighbouring cells and accumulates the product $(1-p_i)^8$ which is the probability of no ignition of the target cell. Therefore, the ignition probability of the target cell is calculated as $1-(1-p_i)^8$. A random number in the range [0,1] is calculated and compared with the ignition probability determining if fire spreads to the target cell. Therefore, the state of all cells is updated at the same time, avoiding sequential cell by cell computation and in this way, reducing total runtime.

For each simulated map, when fire stops the current simulation is terminated and next simulation starts. This is done with a flag used in GPU but tested in CPU, since the cellular automata is driven and executed on GPU. This data movement from GPU to CPU produces an overhead that was not significant in this case because most simulations stopped at early iterations. Then, this reduction of simulation steps overcomes the communication overhead.

As we are dealing with stochastic simulations, even if the set of parameters for all the simulations is exactly the same as the reference parameters, we obtain different final fire perimeters for different random seeds. The fitness difference for that case is 0.05, which we took as a baseline fitness below which we have the effect of stochasticity. In other words, we cannot obtain a fitness value below 0.05, unless we use the same random number generator seed and equal set of parameters.

For histograms of fitted propagation parameters with fixed ignition point, we took the cutoff value of 0.3, while for the case of variable ignition point the cutoff value used to plot the histograms was 1.5. The cutoff values were chosen as the value from which the histogram shape was stationary.

We had used a counter based random number generator: the Random123 library. This library is a collection of counter-based random number generators (CBRNGs) for CPUs (C and C++) and GPUs (CUDA and OpenCL). From this library, Philox generator was used. It allows to generate a very large number of random numbers concurrently, and guarantees that the random numbers are uncorrelated and uniform distributed between 0 and 1.

5. Results and discussion

Parameters of the fire of reference are listed in Table 3 and were chosen so that the fire front never reach the borders of the lattice. Propagation stops because of the environment and because cells have a turn off time that depends on vegetation fuel type. We present the results for a turn off time of 1 step of simulation for

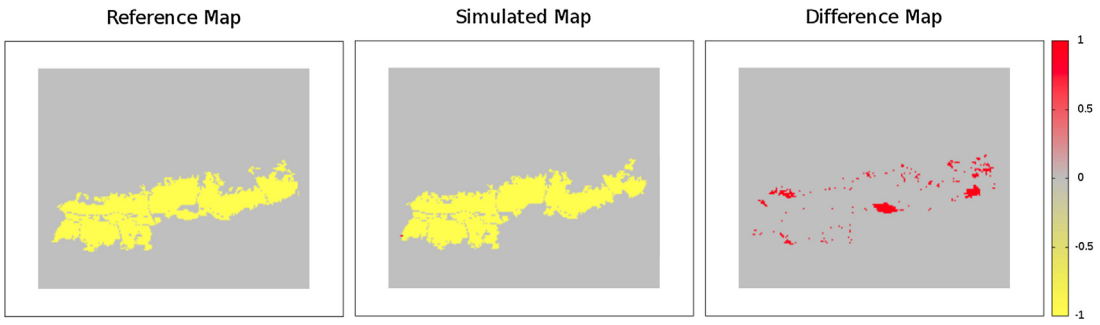


Fig. 2. Difference (right panel) between reference fire (left panel) and the best simulation (center panel) with all the fires starting from the same ignition point ($x = 400$; $y = 250$). Fitness was 0.15 (calculated with Eq. (2)).

Table 1
Initial distribution features of all β_i input parameters of the model.

Initial distribution	Mean	Standard deviation	Range
Gauss	0	5	$(-\infty, \infty)$
Uniform	0	17.3	$[-30, 30]$

Table 2
Parameters of the reference fire and after fit histogram values. The β parameters are defined in Eq. (1). The ignition point coordinates were fixed at the values of the reference fire. The average parameter values were calculated from 62,732 (initial Gaussian) and 70,516 (initial normal) simulations that were below the fitness cutoff value of 0.3.

Parameter	Reference value	Initial uniform mean (sd)	Initial normal mean (sd)
β_0	−5	−8.7 (2)	−4.9 (2)
β_1	−10	−22 (6)	−16.2 (6)
β_2	5	−5.1 (7)	0.4 (7)
β_3	18.1	27 (2)	21.3 (6)
β_4	−5	−9.2 (10)	−5.3 (3)

both forest and shrubland (in Appendix C we show the case of turn off equal to 7 steps for forest and 2 steps for shrubland).

Parameters β_i corresponding to coefficients for forest, wind, slope and aspect were randomly sampled from a uniform initial distribution and from a Gaussian distribution. In order to provide clarity through this section, only results using an initial Gaussian distribution are shown and results using an initial uniform distribution can be seen in Appendix C. We show in Table 1 the initial distribution parameters.

As an example, Fig. 2 shows the difference map between the best of all simulations and the reference map after fitting and estimating the best parameter values. Both simulations started from the same ignition point.

After performing one million simulations starting from the same (known) ignition point, sampling parameter values from an initial Gaussian distribution, we obtained histograms of values for each of the parameters using Genetic Algorithm (Fig. 3 and Table 2). Similar results are obtained when sampling from a uniform initial distribution (Fig. C.11 in Appendix C) showing the independence of the methodology on the initial distribution.

It is a remarkable result to recover the all five parameters of fire propagation, by only comparing between the final fire burnt area of reference and simulations. For instance, we are dealing with a 5 dimensional search space that implies a very large amount of simulations. The Genetic Algorithm is a search strategy that might not work for some situations, e.g. the search may get stuck in a relative minimum of the fitness surface or it might not work if the surface is very complex with lots of minimums. Also if there are interactions between parameters, a fitness distribution with more than one minimum can be obtained, which indicates that more than one set of parameters could give place to the same observed fire perimeter.

Table 3
Parameters of the reference fire and histogram of fitted parameter values using GA. The β_i parameters are defined in Eq. (1) and (x, y) is the ignition point coordinates. The average parameter values were calculated from 134,253 (initial Gaussian distribution) and 141,249 (initial normal distribution) simulations that were below the fitness cutoff value of 1.5.

Parameter	Reference value	Initial uniform mean (sd)	Initial normal mean (sd)
β_0	−5	−9 (2)	−2.3 (5)
β_1	−10	−20 (6)	−13.3 (6)
β_2	5	−2 (19)	3.3 (10.5)
β_3	18.1	19 (14)	3.7 (10)
β_4	−5	−4 (15)	−2.7 (11)
x	400	331 (95)	328 (87)
y	250	224 (33)	217 (34)

As already mentioned, it could be of interest to also determine the ignition point. To that aim we considered the ignition point as an additional parameter, performing one million simulations starting from different ignition points picked at random from the burnt area of the reference fire. As can be seen, histograms after fitting, contain the true parameter values (Table 3). For the wind parameter (β_3 in Table 3) it is seen that a uniform initial distribution leads to more accurate results (Fig. C.12 in Appendix C). However, for the rest of parameters a more informative initial distribution seems to be a more appropriate choice (Fig. 4). In other words, the wind parameter (for initial Gaussian distribution) or aspect and slope parameters (when an initial uniform is used), loose identifiability when the coordinates of the ignition point are considered as an additional parameter (see Figs. 4 and C.12).

Regarding the efficiency of Genetic Algorithm (GA) as compared with a simple Monte Carlo brute force method (MC), we implemented an alternative and more basic fitting procedure (Appendix B), that we can take as a benchmark for time comparison. The beta histograms obtained with MC are shown in Figs. B.8 and B.9. According to our results, the averaged speed of GA is half the one of a simple MC brute force approach, as can be seen in Fig. B.10, where we show the histograms of times for each simulation performed with both methods. This is because the ensemble of simulations using GA remains closer to the "real simulation" (specially for the last evolution of the algorithm) while simulations with random sampled parameters are more heterogeneous and therefore have more variability in simulation time.

5.1. Scalability analysis

When applications with high computation requirements are designed and programmed, is usual to develop them taking into account hardware features, in order to achieve an efficient use of all resources in that architecture (high performance applications). But

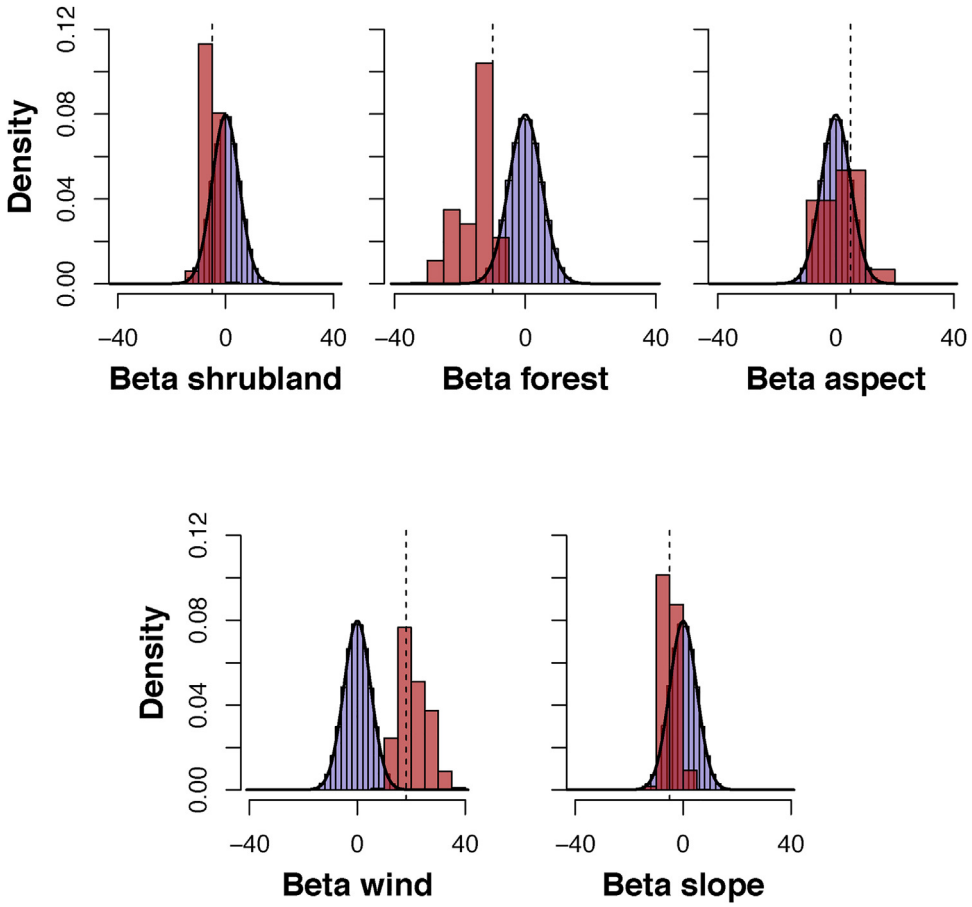


Fig. 3. Histograms for β_i values after fitting the model using the Genetic Algorithm (red bars). Dotted line indicates the true value of the corresponding parameter that was used to generate the fire of reference. One million simulations were performed with all fires starting from the same ignition point ($x = 400$; $y = 250$) and with propagation parameters taken from an initial Gaussian distribution (mean = 0, sd = 5; solid black line and light blue bars). Burning times for forest and shrubland were set to 1 time step. Fitness was calculated with Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

constant advances in technology require a software that remains useful and efficient when computer capabilities increase.

CUDA model and GPU hardware offer high scalable platforms, meaning that CUDA applications may remain efficient through different graphic cards. One of the reasons is that GPU schedulers deliver workload through Streaming Multiprocessors, looking for best occupancy of cores and special units within a multiprocessor.

We therefore executed our application in different GPUs platforms in order to study its behavior and scalability. Table 4 shows runtimes of our application when different graphic cards are used. In this test the overall lattice size remains constant (map size 801×801 and 1 million simulations) as the GPU platform changes.

Four different platforms were used in order to evaluate application scalability. Runtimes show us that our application is scalable. When better graphic cards are used, application runtimes are shorter. It is an important feature that no modification is applied over the computer code, it is just compiled and launched through these different architectures.

We can see that best runtimes were achieved using the GeForce GTX Titan. This graphic card offers the best memory bandwidth (fourth column in Table 4). When application profiling was used in order to evaluate application performance, we could see that main kernels are memory bounded, that means that memory accesses (for read or write operations) are the limiters of the application.

Topography, wind, fuel, fire propagation maps are allocated in GPU global memory. This memory is the most expensive memory in the GPU for an application because it has the highest latency, is not cached and the bandwidth is low when not aligned and not coalesced accesses are carried out.

In order to analyze application performance, two profilers were used: NVIDIA Visual Profiler (nvvp) and NVIDIA Profiler (nvprof) [29]. Using these profiling tools we could see that read from and write to global memory are very inefficient operations. Using `gld_efficiency` and `gst_efficiency` metrics we could see that very low percentages of bandwidth use were achieved. This fact means that most of the memory transaction had to be replayed and each access is handle by more than one memory transaction (because access is not coalesced nor aligned).

For this reason, the use of constant memory and shared memory was analyzed. Both of them are cached memories, then, if read and write operations are studied in order to fit some access pattern, access operation may not have latency penalties. But current graphic cards have very small shared and constant memories that are not enough for our application, where several raster maps are necessary to perform simulations.

Another scalability analysis is the study of the behavior of application runtime when lattice size increases. With this objective, more tests were done where fire landscape sizes were consistently varied (fire map, topography map, wind map, etc.). Results are shown in Fig. 5, where runtime (Y axis) is specified for different map sizes (X axis).

All tests were performed using the Tesla K40c architecture due to memory requirements. This architecture has 12 GB of total on board memory while the remaining graphic cards available have 6 GB, which was not sufficient to perform all the tests.

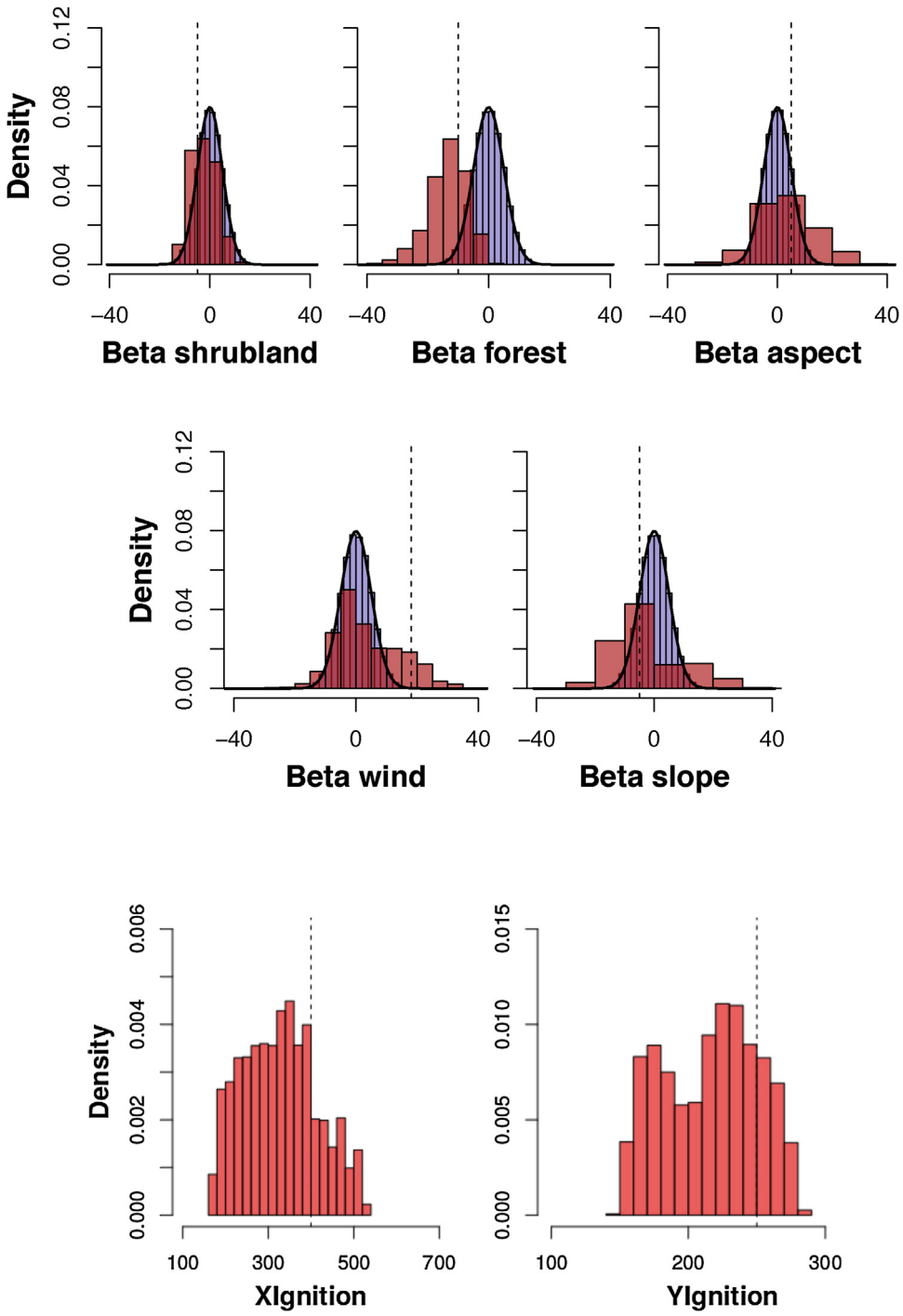


Fig. 4. Histograms for β_i values and ignition point, after fitting the model using the Genetic Algorithm (red bars). Dotted lines indicate the true value of the corresponding parameter that was used to generate the fire of reference. One million simulations were performed with parameters taken from an initial Gaussian distribution (mean = 0, sd = 5; solid black line, light blue bars). Burning times for forest and shrubland were set to 1 time step. Fitness was calculated with Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

Table 4
Additional tests changing hardware architecture. Map size is 801 × 801 and runtime corresponds to 1×10^6 simulations, for Genetic Algorithm (GA) and simple brute force Monte Carlo (MC).

	Name	CUDA cores	Processing power[a]	Memory (GB/s)[b]	Runtime GA (hs)	Runtime MC (hs)
1	Tesla C2070 (2.0)	448	1030	144	23.39	78.74
2	Tesla K40c (3.5)	2880	4291–5040	288	18.29	61.60
3	GeForce GTX 780 (3.5)	2304	3977	288	15.39	48.86
4	GeForce GTX Titan (3.5)	2880	5121	336	12.98	47.50

[a] Single precision GFLOPs peak.
[b] Memory bandwidth.

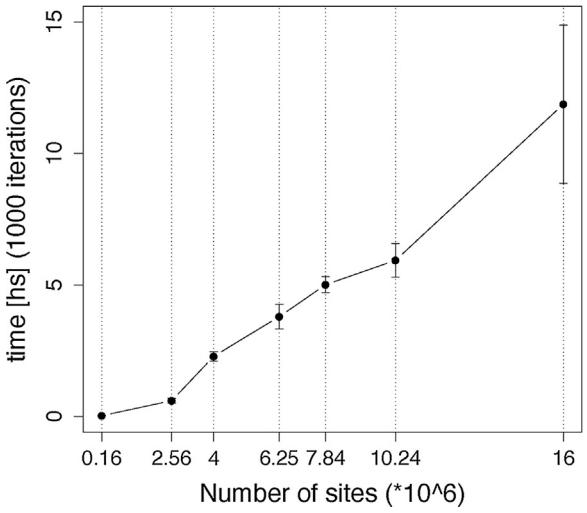


Fig. 5. Scalability study. Different runtimes for different configuration of fire maps for 10 thousand simulations using the Genetic Algorithm. Vertical lines indicate the total amount of cells (rows × columns).

Fig. 5 shows that runtime increases in a linear way when map sizes increase. Even though application maps are squared data structures, and data increases in a quadratic form too, runtimes have a linear growth. This is an important advantage of this parallel solution as compared to the frequently used sequential cellular automata. An important result is achieved when 1600×1600 and 3200×3200 cell maps are used for which a reduction in time is observed because maps sizes are multiple of 32. When GPUs are used, warps of 32 threads are executed in a SIMD way with better GPU cores occupancy as well as more efficient global memory access. Therefore, it is very important to make a correct design of the application when graphic cards are used in order to obtain high performance applications. Using the sequential solution developed on SELES as in [13] it takes near 10 days to perform 9×10^5 simulations. With our parallel implementation it takes 13 h to perform 1 million of simulations (GTX Titan Black, GK110 architecture).

6. Conclusions and open lines

We have developed an application for forest fire simulation and fitting to accomplish with two main goals: the estimation of parameters involved in fire propagation and the efficient implementation of a parallel modular open source computing tool.

It is of importance to be able to determine stochastic fire propagation parameters and ignition points once a fire had occur. Having that information it is possible to identify which are the more sensitive parameters on the propagation of one or more fires of interest and eventually predict fire propagation under different scenarios. Our parallel cellular automata is an spatial stochastic model for fire propagation mounted on several layers that describe topography, fuel type, wind speed and direction, vegetation aspect and slope, that can be easily adapted to include other layers, as well as to implement different rules for fire propagation.

We show in this work that Genetic Algorithm allows to recover propagation parameter values that generated the reference fire. We proved that departing from an initial shifted distribution we can also recover the known parameters by efficiently surfing in the multidimensional parameter space. We obtained well-defined stationary histograms after fitting one million simulations, which makes parallel GPU computing a valuable tool specially when large lattices are used.

The reduction of our application runtime is very significant when compared with previously used serial applications and the scalability analysis allows to determine the time needed to run the propagation on any desired lattice size. Depending on the GPGPU technology used, our first parallel version takes between 13 and 23 h to perform and fit one million of simulations to a given fire of reference using the Genetic Algorithm.

Using CUDA profilers we could see that our application is a memory bounded application, then, the study of memory access patterns and optimization opportunities are open lines. In a near future we will study the application memory use in order to reduce memory latencies as well as to improve memory bandwidth usage.

With this tool is also possible to estimate the fire starting point, allowing the analysis of mapped fires with unknown starting points. However, some of the propagation parameters loose identifiability when the coordinates of the ignition point are considered as an additional parameter. This could be probably improved either by fitting several fires scars instead of only one (assuming they were generated with the same set of propagation parameters) considering ignition points as nuisance parameters, or by using an ensemble of several fitness functions to rank simulations.

The next step will be to use our methodology to fit real fire maps to estimate propagation parameters and eventually fire ignition points, as well as to test other functional dependencies of fire propagation probability. Additionally it would be of interest to fit times of fuel consumption, a feature that will be also added to our application.

Given that this tool is inherently parallel it could be eventually used for fire propagation prediction for which computing times should be the minimum possible simulation times. These results have brought us closer to the real time requirements for forest fire spread prediction which is not our actual main goal but that is a desirable property of any application and should be explored in the near future.

Taking advantage of graphic cards capabilities we are now developing an interface for the visualization of the fire progress in simulation time to be able to interact with the simulation for management purposes. The advantages of GPU's are not restricted to spatial fire propagation but also to a broad range of ecological models that could be studied inspired on a similar program structure, allowing to simulate and fit several models at several scales, specially for large spatial extensions that were difficult to explore with previously used technologies.

Acknowledgements

M. Denham and K. Laneri are members of CONICET. M. Denham and K. Laneri are part of the project TIN2014-53234-C2-1-R of Ministerio de Ciencia e Innovación (MICINN-Spain). We thank David Alonso, Alejandro Kolton and Andrés Solarte for fruitful discussion.

Appendix A. Parallel Genetic Algorithm

The search is performed on a large parameter space: combinations of 5 parameters are evaluated. When the initial ignition point is unknown, the application deals with 7 parameters (X and Y values are added).

In order to accelerate convergence, a Genetic Algorithm (GA) was implemented. This algorithm is based on three main operators: selection, crossover and mutation. The goal of these operations is to generate populations with individual characteristics with good fitness [15]. Selection is based on: individuals that are well suited to its environment can survive and inherit its genes to their offspring. The selection function is based on giving more chances of being chosen to individuals which are well adapted to its environment. Therefore, good features (genes) are inherited by children (individual of next population). CUDA implementation of selection function

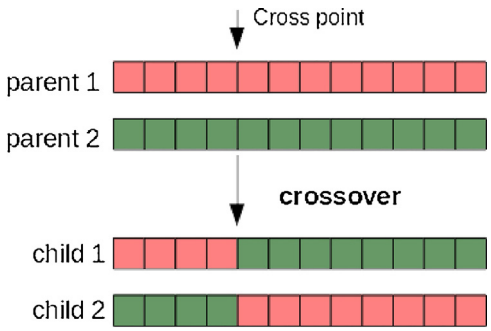


Fig. A.6. One-point crossover of genes.

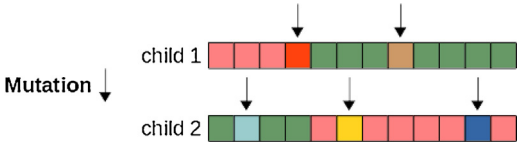


Fig. A.7. Mutation changes makes a tiny (random) gene change.

includes the elitism function: every evolution includes best individuals of previous citizen. Elitism warranties that good individuals are always selected and its features are included in future generations. Tournament selection was parallelized and implemented using CUDA: a thread for each new individual is launched and each thread chose randomly 2 individuals. Tournament selection involves running tournaments between these two individuals (or "chromosomes"). The winner of each tournament (the one with the best fitness) is selected for crossover.

Crossover operator is used in order to mix parents features and to form new individuals. Half of population size threads are launched and each of them choose randomly a cross point. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children. This technique is called one-point crossover, due to just one cross-point is used (Fig. A.6).

Mutation operator is used to maintain population diversity. Using a small probability, each of the individual gene can be mutated. This operation avoid that search process falls down in a local maximum or minimum. Mutation operation is executed by a thread for each population member. For each individual gene, a random number is compared with the mutation probability. We had implemented a dynamic mutation: mutation probability decreases as GA advances. Earlier generation mutation allows to search through more distant zones of the whole search space. Then, as population evolves, it is less likely to have a gene mutation (Fig. A.7).

Appendix B. Parallel brute force Monte Carlo

The very simple methodology starts by randomly sampling a large number of parameter values from a biologically reasonable distribution (assumed uniform in this case). Simulations are per-

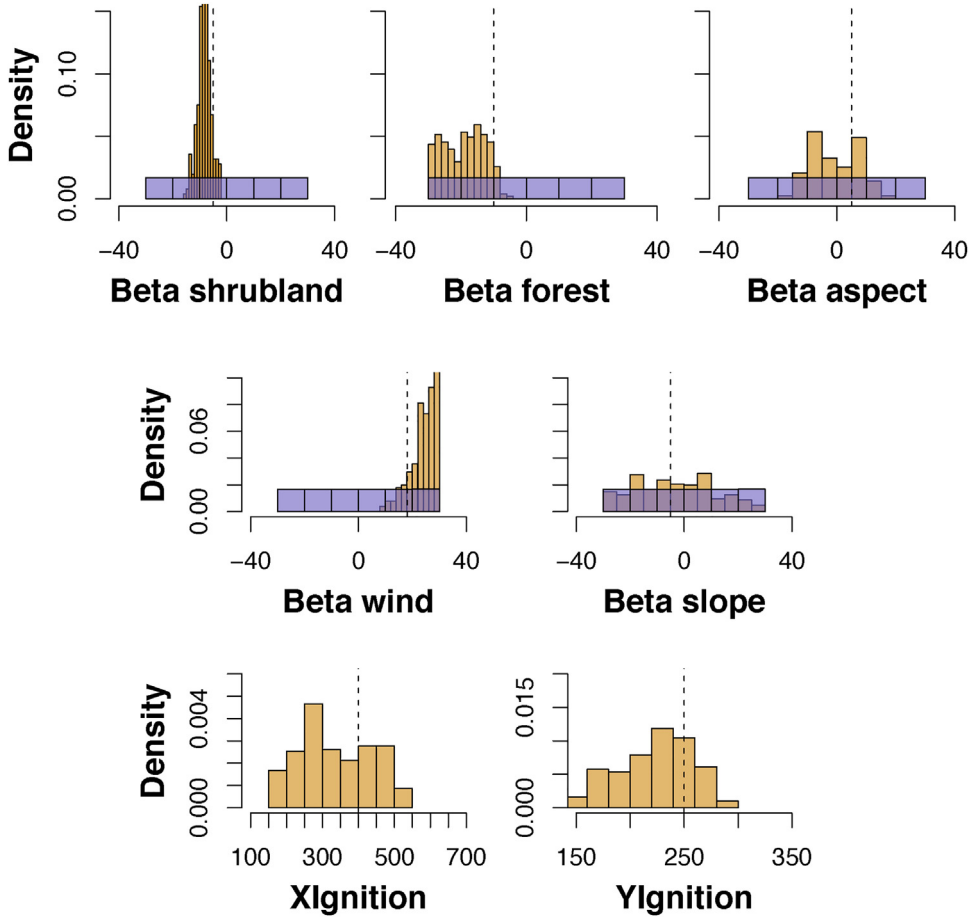


Fig. B.8. Histograms for β_i values after fitting the model using brute force Monte Carlo (yellow bars). Dotted lines indicate the true value of the corresponding parameter that was used to generate the fire of reference. One million simulations were performed with all fires starting from the same ignition point ($x = 400$; $y = 250$) and with propagation parameters taken from an initial Uniform distribution (mean = 0, sd = 5; light blue). Fitness was calculated with Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

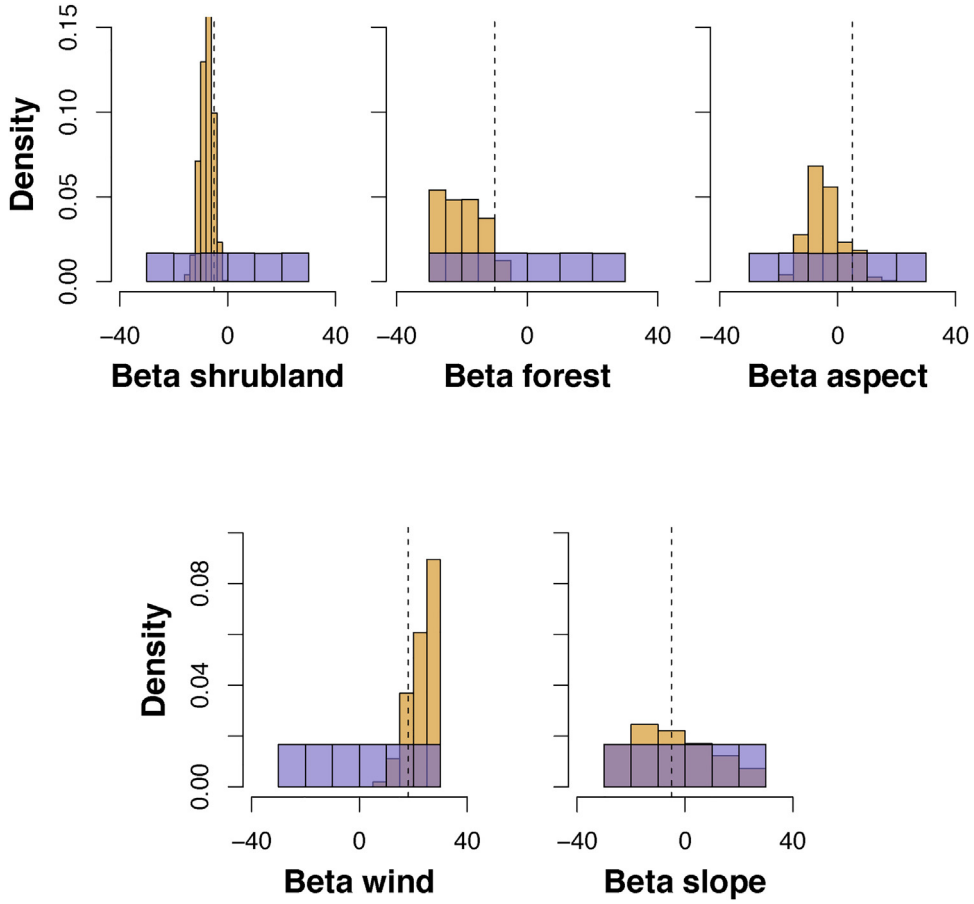


Fig. B.9. Histograms for β_i values after fitting the model using brute force Monte Carlo (yellow bars). Dotted lines indicate the true value of the corresponding parameter that was used to generate the fire of reference. One million simulations were performed with all fires starting from the same ignition point ($x = 400$; $y = 250$) and with propagation parameters taken from an initial Uniform distribution (mean = 0, sd = 5; light blue).Fitness was calculated with Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

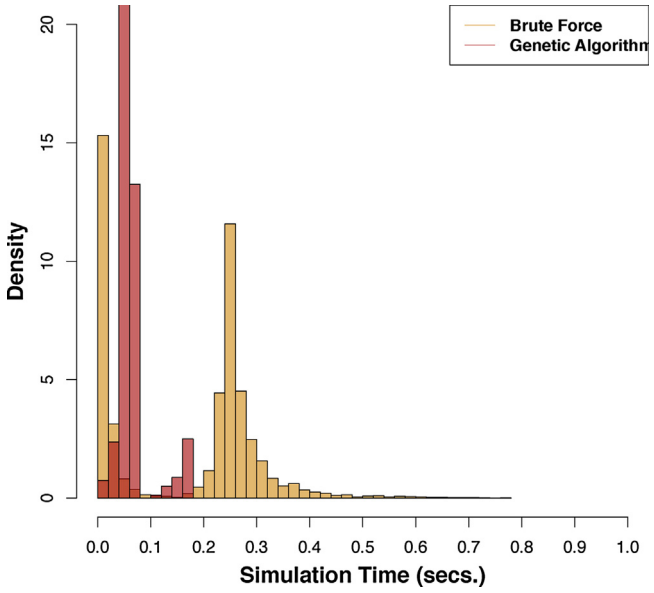


Fig. B.10. Histograms for simulation times using Genetic Algorithm (red) and brute force Monte Carlo (yellow) obtained using a GeForce GXT Titan. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

formed with the complete set of sampled parameters, and are compared with the reference according to the distance function of Eq. (2) ordered according to their fitness. With this methodology no intelligent search is done in the parameter space. Results obtained with this methodology are shown in Figs. B.8 and B.9. A time comparison between simulations performed with GA and MC

is shown in Fig. B.10, where we can see that the averaged speed of GA is half the one of a simple MC brute force approach. This is because the ensemble of simulations using GA remains closer to the "real simulation" (specially for the last evolution of the algorithm) while simulations with random sampled parameters are more heterogeneous and therefore have more variability in simulation time.

Appendix C. More results

This appendix present results using uniform initial distributions for input parameters. Each histogram shows initial distribution and histograms built with fitted parameter values.

After performing one million simulations with parameter values sampled from a uniform distribution, we obtained histograms for each of the parameters as shown in Figs. C.12 and C.11. As can be seen in both figures, histograms of fitted parameters contain the true parameter values (Table 3). The best set of parameter values was defined as the one corresponding to the minimum fitness. The associated error was computed as the standard deviation of the selected ensemble of parameters. The cutoff fitness value for the selected ensemble was set as the minimum needed to obtain a stationary histogram shape for all the parameters. It is interesting to notice that when the ignition point is considered as an additional parameter (Fig. C.12) some parameters become less identifiable (i.e. β_{slope} and β_{aspect}). However when the ignition point is fixed in the known values we obtain, after the fitting procedure, well defined histograms around the (known) values used for the reference fire.

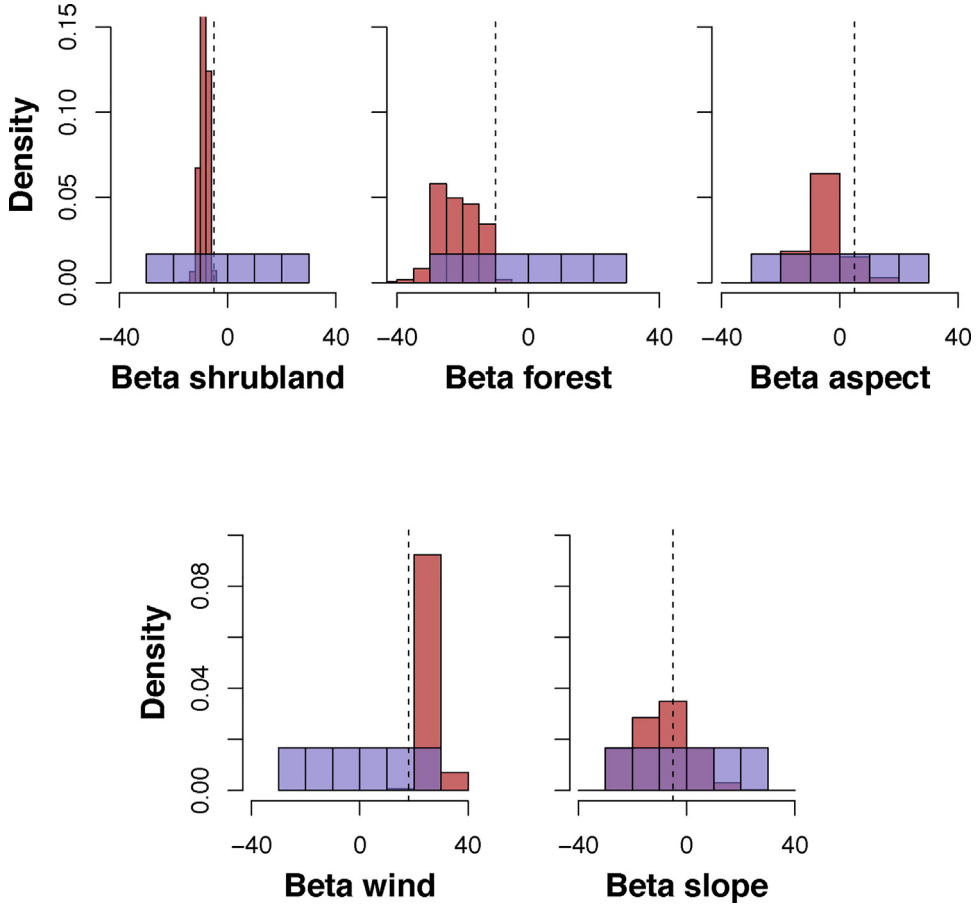


Fig. C.11. Histograms for β_i values after fitting the model using the Genetic Algorithm. Dotted lines indicate the true value of the corresponding parameter that was used to generate the fire of reference. One million of simulations were performed with parameters taken from initial uniform distributions between −30 and 30 (transparent light blue). All the fires started from the same ignition point ($x = 400$; $y = 250$). Fitness was calculated with Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

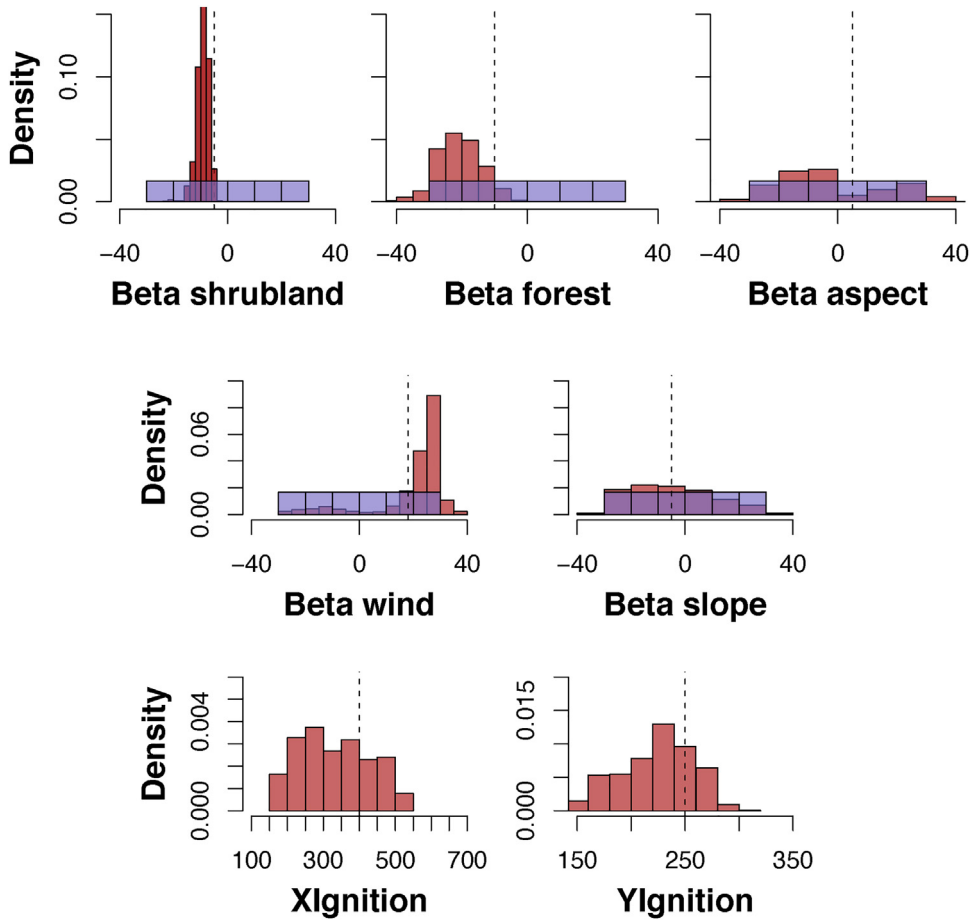


Fig. C.12. Histograms for β_i values and ignition point after fitting the model. Dotted lines indicate the true value of the corresponding parameter that was used to generate the fire of reference. One million of simulations were performed with parameters taken from initial uniform distributions between −30 and 30 (transparent light blue) and fire starting from a point inside the burnt area of the fire of reference. Fitness was calculated with Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

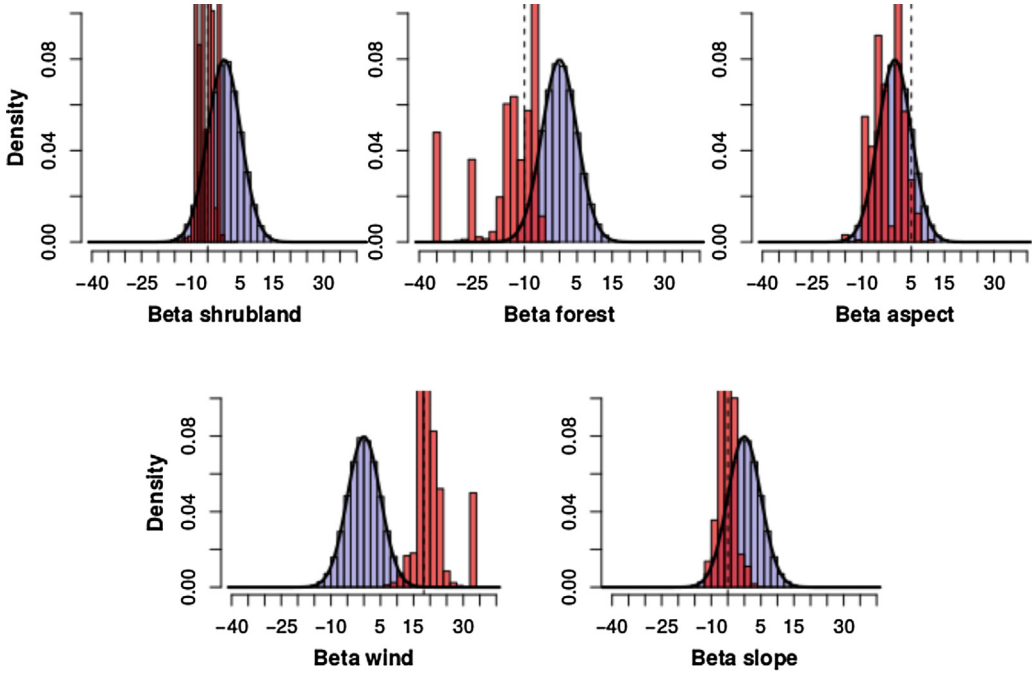


Fig. C.13. Histograms for β_i values after fitting the model using the Genetic Algorithm (red bars). Dotted lines indicate the true value of the corresponding parameter that was used to generate the fire of reference. One million simulations were performed with all fires starting from the same ignition point ($x = 400$; $y = 250$) and with propagation parameters taken from an initial Gaussian distribution (mean = 0, sd = 5; solid black line and light blue bars). Burning times for forest and shrubland were set to 7 and 2 time steps respectively. Fitness was calculated with Eq. (2). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

References

[1] S.L. Stephens, J.K. Agee, P.Z. Fulé, M.P. North, W.H. Romme, T.W. Swetnam, M.G. Turner, Managing forests and fire in changing climates, Science 342 (6154) (2013) 41–42, http://dx.doi.org/10.1126/science.1240294, arXiv:http://science.sciencemag.org/content/342/6154/41.full.pdf, http://science.sciencemag.org/content/342/6154/41.

[2] B. Abdalhaq, A methodology to enhance the prediction of forest fire propagation (Ph.D. thesis), Universidad Autonoma de Barcelona, 2004.

[3] G. Bianchini, M. Denham, A. Cortés, T. Margalef, E. Luque, Wildland fire growth prediction method based on multiple overlapping solution, J. Comput. Sci. 1 (4) (2010) 229–237, http://dx.doi.org/10.1016/j.jocs.2010.07.005 http://www.sciencedirect.com/science/article/pii/S1877750310000463.

[4] M. Denham, K. Wendt, G. Bianchini, A. Cortés, T. Margalef, Dynamic data-driven genetic algorithm for forest fire spread prediction, J. Comput. Sci. 3 (5) (2012) 398–404, http://dx.doi.org/10.1016/j.jocs.2012.06.002, Advanced Computing Solutions for Health Care and Medicine. http://www.sciencedirect.com/science/article/pii/S1877750312000658.

[5] M. Denham, Predicción de la evolución de los incendios forestales guiada dinámicamente por los datos (Ph.D. thesis), Universidad Autónoma de Barcelona, 2009.

[6] M.A. Finney, Farsite: fire area simulator – model development and evaluation, Tech. rep., USDA Forest Service, 2004.

[7] C.D. Bevins, fireLib user manual and technical reference, Systems for Environmental Management, 1996.

[8] R.C. Rothermel, A mathematical model for predicting fire spread in wildland fuels, Tech. rep., USDA Forest Services, 1972.

[9] C.V. Wagner, Effect of slope on fire spread rate, Tech. Rep. Vol 33 number 1, Fisheries and Environment Canada, Forestry Service, 1977.

[10] C.E.V. Wagner, Effects of slope on fire spreading downhill, Tech. rep., Petawawa Natural Forestry Institute, 1987.

[11] G. Sanjuan, C. Brun, T. Margalef, A. Cortés, Determining map partitioning to minimize wind field uncertainty in forest fire propagation prediction, J. Comput. Sci. 14 (2016) 28–37, http://dx.doi.org/10.1016/j.jocs.2016.01.006, The Route to Exascale: Novel Mathematical Methods, Scalable Algorithms and Computational Science Skills. http://www.sciencedirect.com/science/article/pii/S1877750316300060.

[12] L. Russo, P. Russo, D. Vakalis, C. Siettos, Detecting weak points of wildland fire spread: a cellular automata model risk assessment simulation approach, Chem. Eng. Trans. 36 (2014) 253–258, http://dx.doi.org/10.3303/CET1436043.

[13] J.M. Morales, M. Mermoz, J.H. Gowda, T. Kitzberger, A stochastic fire spread model for north Patagonia based on fire occurrence maps, Ecol. Model. 300 (0) (2015) 73–80, http://dx.doi.org/10.1016/j.ecolmodel.2015.01.004 http://www.sciencedirect.com/science/article/pii/S0304380015000162.

[14] M.M.D. Kennedy, Using a stochastic model and cross-scale analysis to evaluate controls on historical low-severity fire regimes, Landsc. Ecol. 25 (2010) 1561–1573.

[15] J. Koza, Genetic Programming. On the Programming of Computers by Means of Natural Selection, 1st ed., MIT Press, Massachusetts Institute of Technology Cambridge, MA, USA, 1992.

[16] R. Farber, CUDA Application Design and Development, 1st ed., Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2011.

[17] J. Smith, L. Barford, S.M.F.C.H. Dascalu Jr., High parallel implementation of forest fire propagation models on the GPU, Proceedings of the 2016 International Conference on High Performance Computing and Simulations (HPCS 2016) (2016).

[18] J.E. Smith, vFireLib: A Forest Fire Simulation Library Implemented on the GPU (Ph.D. thesis), University of Nevada, Reno, 2016, May.

[19] F. Sousa, R. dos Reis, J. Pereira, Simulation of surface fire fronts using firelib and GPUs, Environ. Model. Softw. 38 (2012) 167–177, http://dx.doi.org/10.1016/j.envsoft.2012.06.006 http://www.sciencedirect.com/science/article/pii/S1364815212001867.

[20] C. Miller, J. Hilton, A. Sullivan, M. Prakash, SPARK – A Bushfire Spread Prediction Tool, Springer International Publishing, Cham, 2015, pp. 262–271, http://dx.doi.org/10.1007/978-3-319-15994-2_26.

[21] S. Monedero, J. Ramirez, D. Molina-Terrén, A. Cardil, Simulating wildfires backwards in time from the final fire perimeter in point-functional fire models, Environ. Model. Softw. 92 (2017) 163–168, http://dx.doi.org/10.1016/j.envsoft.2017.02.023 http://www.sciencedirect.com/science/article/pii/S1364815216306697.

[22] B. Arca, T. Ghisu, W. Spataro, G.A. Trunfio, GPU-accelerated optimization of fuel treatments for mitigating wildfire hazard, International Conference on Computational Science, ICCS 2013. Procedia Computer Science 18 (2013) 966–975.

[23] B. Arca, T. Ghisu, G.A. Trunfio, GPU-accelerated multi-objective optimization of fuel treatments for mitigating wildfire hazard, J. Comput. Sci. 11 (Suppl. C) (2015) 258–268, http://dx.doi.org/10.1016/j.jocs.2015.08.009 http://www.sciencedirect.com/science/article/pii/S1877750315300120.

[24] S.D. Gregorio, G. Filippone, W. Spataro, G.A. Trunfio, Accelerating wildfire susceptibility mapping through GPGPU, J. Parallel Distrib. Comput. 73 (8) (2013) 1183–1194, http://dx.doi.org/10.1016/j.jpdc.2013.03.014 http://www.sciencedirect.com/science/article/pii/S0743731513000580.

[25] T. Ghisu, B. Arca, G. Pellizzaro, P. Duce, An improved cellular automata for wildfire spread, Procedia Comput. Sci. 51 (Suppl. C) (2015) 2287–2296, http://dx.doi.org/10.1016/j.procs.2015.05.388, International Conference On Computational Science, ICCS 2015. http://www.sciencedirect.com/science/article/pii/S1877050915011965.

[26] T. Ghisu, B. Arca, G. Pellizzaro, P. Duce, A level-set algorithm for simulating wildfire spread, CMES Comput. Model. Eng. Sci. 102 (1) (2014) 83–102, http://www.techscience.com/doi/10.3970/cmes.2014.102.083.pdf.

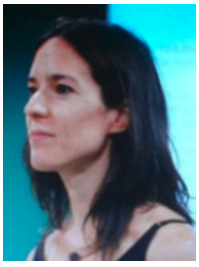
[27] J. Forthofer, Windninja 2.0.1.

[28] J. Paruelo, A. Beltran, E. Jobbagy, O. Sala, R. Golluscio, The climate of Patagonia: general patterns and controls on biotic processes, Ecol. Austral 8 (2) (1998) 85–101.

[29] T.M. John Cheng, M. Grossman, Professional CUDA C Programming, Wrox, 2014.

[30] D.B. Kirk, W.-m.W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, 1st ed., Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2010.

[31] S. Cook, CUDA Programming. A Developer's Guide to Parallel Computing with GPUs, Morgan Kaufmann Publishers Inc, 2013.



Monica Denham graduated as System Analyst in 2003 and obtained her degree in computer science in 2005 both from National University of La Plata, Argentina. In 2007 she received her Computer Science researcher degree, and Ph.D. in computer science in 2009 form "Universidad Autónoma de Barcelona", Spain. Her research is focused in parallel computing and high performance computing. At the moment her work involves radar signal processing and prediction systems applications.

Karina Laneri obtained her Ph.D. Physics degree at National University of La Plata, Argentina in 2003. She studied population dynamics by designing dynamical models, individual based models and complex networks to understand the key mechanisms underlying ecological processes, combining approaches from physics, biology, mathematics and computer science. After some interdisciplinary postdocs abroad, she is currently a researcher of CONICET at the Statistical and Interdisciplinary Physics Group at Bariloche Argentina.