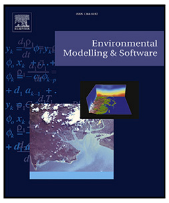
Contents lists available at ScienceDirect

Environmental Modelling and Software

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



Position Paper

Cloud-based urgent computing for forest fire spread prediction☆

Edigley Fraga [a,*], Ana Cortés [a], Tomàs Margalef [a], Porfidio Hernández [a], Carlos Carrillo [b]

[a] *Universitat Autònoma de Barcelona, 08193 Bellaterra, Barcelona, Spain*
[b] *MITIGA SOLUTIONS S.L., Carrer Ramon Turro 169, 08005, Barcelona, Spain*

ARTICLE INFO

Keywords:
Data uncertainty
Forest fires
Genetic algorithm
Data-driven calibration
Urgent computing
Cloud computing

ABSTRACT

Forest fires cause every year damages to biodiversity, atmosphere, and economy activities. Forest fire simulation have improved significantly, but input data describing fire scenarios are subject to high levels of uncertainty. In this work the two-stage prediction scheme is used to adjust unknown parameters. This scheme relies on an input data calibration phase, which is carried over following a genetic algorithm strategy. The calibrated inputs are then pipelined into the actual prediction phase. This two-stage prediction scheme is leveraged by the cloud computing paradigm, which enables high level of parallelism on demand, elasticity, scalability and low-cost. In this paper, all the models designed to properly allocate cloud resources to the two-stage scheme in a performance-efficient and cost-effective way are described. This Cloud-based Urgent Computing (*CuCo*) architecture has been tested using, as study case, an extreme wildland fire that took place in California in 2018 (Camp Fire).

Software and data availability

- The cloud-based two-stage prediction platform (implemented in the Java programming language) is available at:
  https://github.com/edigley/two-stage-prediction.
  Software requirements: Ubuntu 22.04.2 LTS, Java SE 8+, Docker, Kubernetes.
- All the data (GIS themes, shapefiles, configuration files, etc.) needed for the *Camp Fire* (together with other scenarios) can be found at:
  https://github.com/edigley/fire-scenarios.
  Size of archive: campfire/landscape/campfire.lcp: 15 MB

1. Introduction

Fire is a natural and sometimes essential element of many ecosystems. They are beneficial for forest renewal, to help control insect and disease damage, and to reduce the buildup of fuel and thus future fire intensity. Even large wildfires are part of a defined disturbance regime (San-Miguel-Ayanz et al., 2013). Nevertheless, frequent and large-scale fires have negative impacts on the quality of the air and water, biodiversity, soil, and landscape aesthetics. Not to mention that they can cause economic damage and loss of human lives in populated areas. For that reason, the challenge from both a prevention and a suppression point of view is to anticipate and reduce the spread potential of large wildfires and the succeeding risk to human lives, property, and land use systems (Tyndall, 2023).

Forest fire prevention strategies for detection and suppression have improved significantly through the years, both due to technological innovations and the adoption of various skills and methods. Nowadays, wildfire researchers use technologies that integrate data on weather prediction, topography, fuel modeling, and other factors to predict how fires spread (Zacharakis and Tsihrintzis, 2023; Bakhshaii and Johnson, 2019). Forest fire prediction, prevention, and management measures have become increasingly important over the decades. Systems for wildfire prediction represent an essential asset to back up forest fire monitoring and extinction. They are also applied to predict forest fire risks, and to help in fire-control planning and resource allocation (Veronica Casartelli, 2023).

Notwithstanding the significant technological advances over the past decades, this kind of natural hazard is still difficult to be modeled and to be accurately simulated, for that reason, one can find different simulations tools devoted to that purpose Wilfire Analyst (WFA) (Ramirez et al., 2011; Monedero et al., 2019), FARSITE (Finney, 1998), QUIC (Linn et al., 2020), PhyFire (Asensio et al., 2023) among others.

☆ This research has been supported by Ministerio de Ciencia e Innovación MCIN AEI/10.13039/501100011033 under contract PID2020-113614RB-C21 and by the Catalan government under grant 2021-SGR-574.

* Corresponding author.
E-mail addresses: edigley@gmail.com (E. Fraga), ana.cortes@uab.cat (A. Cortés), tomas.margalef@uab.cat (T. Margalef), porfidio.hernandez@uab.cat (P. Hernández), carles.carrillo@mitigasolutions.com (C. Carrillo).

https://doi.org/10.1016/j.envsoft.2024.106057
Received 15 June 2023; Received in revised form 19 March 2024; Accepted 24 April 2024
Available online 27 April 2024

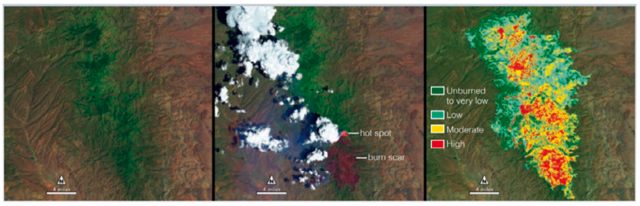


Fig. 1. Burned Area Emergency Response Maps. Credit: USDA Forest Service.

A critical point when dealing with forest fire spread simulations is data uncertainty. A fire spread simulation needs a plethora of dynamic and static input data. Dynamic data includes weather inputs like air temperature and humidity, wind speed and direction, and fuel conditions like live and dead initial fuel moisture. The static or semi-static data include spatial information regarding the elevation, slope, aspect, and fuel types. Furthermore, the simulation process requires to start up from an initial state of the forest fire (single ignition point or shape) to be able to predict the behavior of the event in a near future. Therefore, fire perimeter acquisition is also a key factor when dealing with this simulations frameworks, not only for initializing purposes, but also for post event tools validation. Unfortunately, event ignition points and intermediate perimeters are subject to high levels of uncertainty at the moment of a disaster occurrence. Besides field observations, moderate resolution satellite images are used to estimate fire perimeters, especially those from *Terra and Aqua* satellites (NASA, 1999).

MODIS (or Moderate Resolution Imaging Spectroradiometer) is a key instrument aboard *Terra and Aqua* satellites. Their images are taken at a resolution from 250 m to 1 Km, and the satellites pass through a particular area twice a day. Satellites from Landsat (NASA/USGS, 1972) and Copernicus (ESA, 1998) programs have optimal ground resolution and spectral bands to efficiently track land use and to document land change due to wildfire and other natural and human-caused changes. Landsat has a finer resolution of 15 m and 30 m, but unfortunately its combined (Landsat 8 + Landsat 9) revisit time for data collection is eight days, too much longer for disaster response. Using publicly available data from these global projects is the way-to-go option, but, with proprietary access to other satellite constellations, more coverage and higher resolution, intra-daily revisit can also be an option.

Observing surface temperatures from space can be difficult, however, as atmospheric moisture and air temperature can skew the signals. Despite of cloud cover, every surface emits thermal infrared radiation, or heat. By detecting radiation in two thermal wavelengths, satellite's instruments can measure the temperature of an observed area. Fire managers also analyze images from the shortwave-infrared band, which reflects strongly from exposed ground, to identify scorched areas, as well as the thermal observations to locate the perimeters of active fires. Since the thermal bands can penetrate smoke that might otherwise obscure the view, the resulting images can be used to estimate the fire perimeters of interest.

When a fire is detected, the first perimeter describing the burned area is computed by monitoring agencies. This perimeter, with a high degree of accuracy, is then made available in their systems, like in the European Forest Fire Information System (EFFIS) and the Fire Information for Resource Management System (FIRMS) in the United States and Canada. For example, Fig. 1 shows the location of the 2013 Silver Fire in New Mexico, USA. The image on the left represent the status "before" the fire event, acquired in May 28, 2013. The middle image shows the location of the fire (bright red dot) and burn scar (dark red) on June 13, 2013, "during" event, while the fire was still growing. The image on the right is an example of a Burned Area Emergency Response, showing areas with high (red), moderate (yellow) and low (green) severity burns.

Consequently, one can state that input data describing the current scenario of a given event is subject to high levels of uncertainties that represent a serious drawback for the correctness of the prediction (Thompson and Calkin, 2011; Benali et al., 2016; K.C. et al., 2021).

To deal with this issue, the scientific community developed a handful of input data calibration methods reported in Jain et al. (2020). This review paper highlights that Genetic Algorithm scheme is the Machine Learning approach that has been widely used in the forest fire research community for studying and enhancing the prediction of the area burned by a forest fire (Pereira et al., 2022). An implementation based on *Genetic Algorithm (GA)* has been successfully adopted as a calibration technique in the two-stage prediction method used in Cencerrado et al. (2014). The two-stage methodology adjusts the input parameters for a given forest fire spread prediction in a calibration stage implemented using GA. This calibration stage uses the recent past data to calibrate certain input simulator values, in order to later use these values for predicting the near future evolution of the fire. This approach requires the iterative execution of a widely set of simulations having each different input parameters configuration. This scheme increases the overall response time of the simulation framework in a real-time operational environment, therefore, one of the challenges that arises consists of finding the best trade off between the time incurred in the calibration process and the accuracy of the obtained results by keeping the whole process within hard-deadline constraints. This characteristic turns the wildfire spread prediction process into a *hard-deadline-driven* task. For instance, a wildfire simulation that could accurately predict the perimeter of a wildfire a couple of hours ahead can help firefighters to put firebreaks at the most effective place to stop the fire propagation. Then, for the task of fire suppression, an accurate prediction that comes up late is useless.

These characteristics represent an urgent computing system: *"simulation results needed by relevant authorities in making timely and informed decisions to mitigate financial losses, manage affected areas and reduce casualties"* (Leong and Kranzlmüller, 2015). The three urgent computing requirements to be met are:

1. The computation operates under a strict deadline ("*late results are useless*");
2. The beginning of the event is unpredictable;
3. The computation requires significant resource usage.

Any viable solution must be *deadline-driven*, *on-demand provisioned*, and *scalable* to fulfill these requirements. To deal with them, High-Performance Computing (HPC) community used to rely on dedicated high-end clusters, supercomputers, or distributed computing platforms (Denham et al., 2022). As an enabling technology, cloud computing allows new strategies to cope with the urgent computing challenge, as it offers on-demand provisioning, immediate scalability, and abundant provision of resources.

In addition, cloud-based solutions allow access to these features for the price of a few dollars per hour. Such a characteristic is a fit for forest fire spread prediction systems due to (1) the seasonality of the wildfire occurrences, and (2) because forest fire prevention services (fire brigades) usually cannot afford the *Total Cost of Ownership* (*TCO*) to keep an infrastructure idle until eventually needed by an urgent computation. In this work, we devise and evaluate an adequate solution to this problem, defining a performance-efficient and cost-effective cloud-based solution (*CuCo*) built upon a proven methodology for forest fire spread prediction. The proof of concept of the *CuCo* architecture is described in Fraga et al. (2021). In that work, the authors described the *CuCo's* architecture at a module level by defining what each module should do and the interactions among them. The modules devoted to the two-stage prediction scheme incorporate an adaptive evaluation technique based on a periodic monitoring of the spread prediction error for the calibration phase, avoiding the waste of computing time running undoubtedly unfit individuals. The insight of these modules are described in Fraga et al. (2020, 2022). However, the design of the models within the *CuCo*'s modules associated to the resource allocation in the cloud environment in an cost-effective way, were not previously described. In the present work, an overview of the just defined modules is done, and a deeply description of the proposed models for the modules devoted to optimize resource allocation in the cloud is performed, where an elastic and scalable cloud-based solution platform implemented through coarse-grain parallel processing using a work queue has been exploited. In consonance with the hard-deadline-driven nature of fire extinction activities, the proposed strategies improve the convergence of the genetic algorithm and decrease the response time for the time-critical calibration stage being a solid step forward to bring an efficient and cost-effective easy to deploy cloud-version of this prediction framework.

Finally, the complete framework has been applied to a real complex forest fire scenario that took place in California (USA) in 2018.

The remainder of this document is organized as follows. Related work is discussed in Section 2. The cloud-based architecture to deal with data uncertainty problem including the models for the cost-effective deployment of the simulations framework is presented in Section 3. The experimental results are discussed in Section 4 and, finally, some concluding remarks are given in Section 5.

2. Related work

As it has been stated in the previous section, this work describes a novel cloud-based architecture for forest fire spread prediction that has been designed to be *deadline-driven*, *on-demand provisioned* and *scalable*. This framework is able to calibrate uncertain input parameters using the two-stage prediction scheme based on *GA*. The framework can be deployed in a easy way by automatically select and allocate those resources that better fits the *cost/time* constraints defined by the users. There have been several efforts in the literature to develop easy-to-use forest fire spread frameworks, however, the majority has been focused on defining user-friendly API that facilitates the usability of complex simulators. Although some of them have web oriented graphical user interface, the simulation tool will be finally running in a dedicate high-end on premise server.

Arca et al. (2019) presented a web-based wildfire simulator for operational applications that can assist the incident command teams in charge of tactical wildfire suppression. The simulator consists of a graphical user interface, a model devoted to the downscaling of wind fields, and a module that provides the wildfire propagation. The whole solution is a client–server application, with the heavy computational work executed in parallel on a dedicated server. Oliveira et al. (2023) developed a fire-spread prediction system tailored for the Brazilian Cerrado. Their system allows automatically upload of hotspots and satellite data to calculate maps of fuel load and moisture, and probability of burning for simulating fire spread. Results are available on an interactive web-platform, used as a tool for fire prevention and suppression. It is executed on a parallel platform that uses execution threads boosted by task-stealing algorithms, running on a dedicated high-end on-premise server.

Wildfire Analyst (WFA) is a software application that allows real-time analysis of wildfire, simulating the spread of wildfires using Rothermel's model among others (Ramirez et al., 2011). It is integrated with GIS tools, allowing to change parameters to better reflect actual conditions. Although being originally a desktop application, it has been updated and it is also offered as a web and mobile, being designed to be used at the operations center, or directly on scene (Monedero et al., 2019). It provides a comprehensive set of outputs and tools, also including a data assimilation technique which tunes the simulations results to the actual observed fire behavior.

Kalabokidis et al. (2014) implemented a cloud application composed of wildfire risk and spread simulation service. End users access

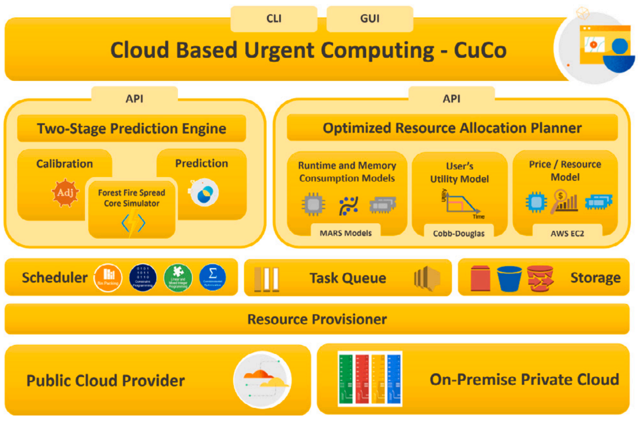


Fig. 2. High-level architecture of the cloud-based two-stage fire spread prediction solution.

the application in a software as a service delivery model, being charged for their consumed processing time during the actual wildfire simulation period. The application presents the flexibility to scale up or down the number of computing nodes needed for the requested processing depending on the number of simultaneous users.

Altintas et al. (2015) conducted the comprehensive WIFIRE Firemap project, a dynamic data-driven system to predict wildfire progress through data analysis and map visualizations. They used *FARSITE* as one of the models that are coupled with a wind simulator. Compute-intensive tasks run in parallel on distributed computing environments.

Miller et al. (2015) presented an integrated software system for forest fire spread prediction, which uses a user-defined algebraic spread rate to model fire propagation. The software model is run based on a modular workflow-based software environment. Garg et al. (2018) proposed a scalable cloud-based bushfire prediction framework, which allows forecasting of the probability of fire occurrences. The solutions allows the selection of different bushfire models for specific regions and scheduling users' requests within their specified deadlines.

Considering the evaluated applications and solutions, although being a step toward the spread of adoption of simulation techniques to local fire agencies, they do not address the issues related to input data uncertainties in an agnostic approach, neither was developed to be applied in urgent computing firefighting scenarios. Both characteristics, diminishing data uncertainty and easy-to-deploy cloud-based urgent computing, are the objective of our proposed solution.

3. Methods

This section details the modules of the **C**loud-based **u**rgent **Co**mputing (dubbed *CuCo*) architecture, which are mainly grouped into two principal components: the*Two-Stage Prediction Engine* and the *Optimized Resource Allocation Planner*. The scheme of this architecture is depicted in Fig. 2. The *Two-Stage Prediction Engine* is responsible for both the *calibration* and the *prediction* stages. The *Optimized Resource Allocation Planner* component is capable of minimizing cost while maintaining a prediction deadline. As it has been previously mentioned, the design of the modules of the first component has been previously published in Fraga et al. (2020, 2022), meanwhile the design of the models involved in the second component (*Optimized Resource Allocation Planner*) is the core contribution of this work. In addition to these two principal components, *CuCo* includes secondary components such as: *Scheduler*, *Task Queue*, *Storage* and *Resource Provisioner* components. These components are later on described in this work.

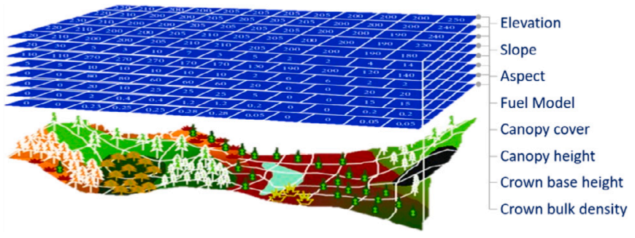


Fig. 3. FARSITE Landscape file (LCP).

3.1. *Forest fire spread simulator*

The core element of the *CuCo*'s framework is the underlying forest fire spread simulator. As it is has been stated above, there exist several forest fire spread simulator oriented to reproduce the behavior of a wildfire. The proposed *CuCo* architecture has been design to be simulator independent, so, it could be deployed using different simulation tools. The forest fire spread simulator used in this work is *FARSITE*. *FARSITE* (Finney, 1998) is a well-known fire growth modeling system that uses spatial information on topography and fuels along with weather and wind inputs. *FARSITE* requires static data, which is introduced using a single file, called landscape fir (or LCP for short), composed by different layers of information each related to one type of this static input data. Fig. 3 shows the composition of this multi-layer file.

FARSITE makes it possible to compute wildfire growth and behavior for long periods under heterogeneous conditions of terrain, fuels, and weather. It also incorporates existing models for surface fire, crown fire, spotting, post-frontal combustion, and fire acceleration into a two-dimensional fire growth model. However, in this work, we do not consider spot fire models. Any FARSITE simulation generates a sequence of fire perimeters representing the growth of a fire under a given input condition. For that purpose, it incorporates, among others, the simple but effective Rothermel's surface fire spread behavior model (Rothermel, 1972) along with Huygens's Principle of wave propagation. *FARSITE* is part of *FlamMap* solution,[1] a fire analysis desktop application.

As wind is one of the most influential environmental factors affecting wildland fire behavior, *FARSITE* can be coupled with wind models for better prediction results. *WindNinja* (Forthofer et al., 2009) is a microscale wind model developed for use in wildland fire applications. It computes spatially varying wind fields, generating high resolution wind prediction in complex terrain. It is specifically designed to simulate the effect of terrain on wind flow, and it can use information from standard weather forecasts to help determine the future wind inputs.

A wildfire simulation, whether coupled to a wind field model or not, is a process inherently complex. Henceforth, a long execution time for a single simulation is not atypical, especially for large wildfires. Although most recently there are some independent work improving *FARSITE* computation time with more sophisticated strategies (Yoo and Song, 2023), in this work, we use *FARSITE* in its original version.

3.2. *Two-stage prediction engine*

Usually, to predict forest fire behavior, a simulator takes the initial state of the fire front perimeter (P_0) along with other parameters as input. As output, the simulator returns the fire front spread prediction for a later instant in time ($\hat{P}_1$). After comparing the simulation result with the actual advanced fire front (P_1), the predicted fire line tends to differ from the actual one. Besides the natural phenomena

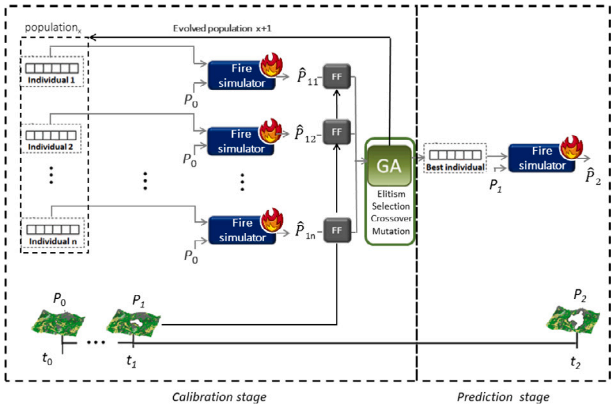


Fig. 4. Basic scheme of the two-stage fire spread prediction method.

modeling complexity uncertainty, the reason for this mismatch is that the *prediction* in the classic scheme is based solely on a single set of input parameters, affected by data uncertainty. To overcome this drawback, a simulator-independent data-driven prediction method was proposed to calibrate model input parameters (Abdalhaq et al., 2005). Due to the inclusion of a *Calibration Stage*, the set of input parameters is adjusted before every prediction step. Thus, the solution comes from reversing the problem, coming up with a parameter configuration such that the fire simulator would produce predictions that match the actual fire behavior. After detecting the simulator input that best reproduces the observed fire propagation, the same set of parameters is used to describe the conditions for the next prediction ($\hat{P}_2$), assuming that meteorological circumstances remain constant during the next prediction interval.

We leverage this forest fire prediction method in the *Two-Stage Prediction Engine* component of the *CuCo*'s architecture, which includes two modules associated each to one of the stages of this approach: *calibration* and *prediction*.

The calibration uses data from the recent past behavior of the fire to adjust the dynamic input data values and alleviates the data uncertainty problem. The prediction then applies the adjusted data to estimate more accurate future fire spreads. The calibration stage is orchestrated by an iterative data-driven compute-intensive Genetic Algorithm (GA) that runs *FARSITE* as the *Core Simulator*. More precisely, this stage starts from an initially random population of individuals. One individual corresponds to a certain configuration of the dynamic input parameter required to run the simulator. Each one of these input parameters is referred as *gene* in the context of GA. In detail, the input parameters calibrated in this stage are: dead and live fuel moisture, wind speed and wind direction, air temperature and humidity, vegetation type and the adjustment factor. As it is shown in Fig. 4, all individuals can be executed in parallel and, once all individuals have finished, they are ranked according to a fitness function (*FF* box in Fig. 4) that determines the goodness of the simulation result. To determine the fitness of an individual, we use the *goodness-of-fit* function, that compares the predicted and actual fire spreads (This fitness function is later on introduce in this paper). Then, using the ranked population, a new evolved population (set of individuals) is generated using the so called *genetic operators*: *elitism*, *selection*, *crossover*, and *mutation* (*GA* box in Fig. 4). This process is repeated a predetermined number of times (GA generations) what could result in a huge number of simulation. In the end, the best individual is selected as the one to be used for prediction purposes (*Prediction stage*). As a data-driven prediction scheme, to enhance the quality of the predictions, the two-stage method is applied continuously, as it is shown in Fig. 5, where a new step of the whole methodology is performed using the subsequent observed real fire perimeter (P_3 in Fig. 5).

[1] https://www.firelab.org/project/flammap

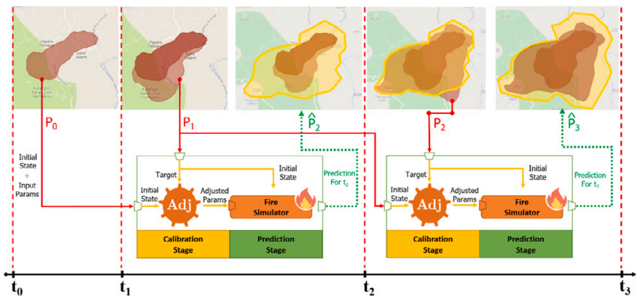


Fig. 5. Two-stage fire spread prediction method through time.

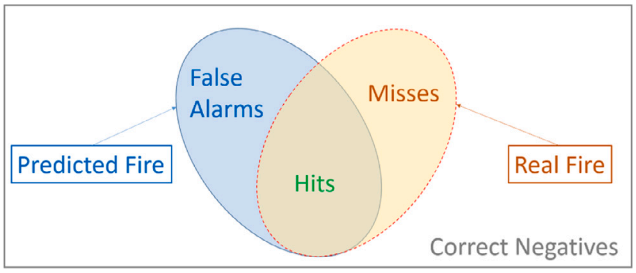


Fig. 6. Different categories present in forecast verification.

GA are very suitable for dealing with the problem at hand, as they avoid local minimums, and the convergence of the algorithm is quite independent of the initial solution. In addition, with regards to each generation, they are inherently parallelizable, which allows the fully exploitation of on-demand, scalable, and elastic cloud computing resources. Nonetheless, there is the sequential part of the GA, where it is necessary to wait for the end of the slowest simulation of the current generation in order to apply the GA operators and only then to generate a new evolved population. In Section 3.2.2 we shall describe how we deal with this issue.

3.2.1. *How good is a fire spread prediction?*

There are different categories present in forecast verification, as illustrated in Fig. 6. The areas around the simulation map that have not been burned by the actual fire nor by the simulated fire are considered **Correct Negatives**. **Hits** are areas that have burned in both fires. **Misses** are the areas that have burned only in the actual fire, whereas **False Alarms** are areas that have burned only in the simulated fire.

We use a *Goodness-of-Fit* (*GoF* for short) metric as fitness function to guide the evolution of the genetic algorithm in the calibration stage. It was inspired in a method that unambiguously shows the degree of spatial concordance between two or more categorical maps (Hargrove et al., 2006). Eq. (1) shows how it is calculated.

$$GoF = \frac{Hits}{Hits + Misses} \times \frac{Hits}{Hits + FalseAlarms} \tag{1}$$

Fig. 7 generalizes the idea of the goodness of fit function in terms of polygons (map) comparison. The *GoF* metric is based on two values: (1) the proportion of the intersecting area to the total area from Map 2, and (2) the proportion of the intersecting area to the total area of Map 1. The first term gives the proportion of "*insideness*" between Map 2 and Map 1. The second term weights this degree of fit by the fractional share of the Map 1 area that gets intersected. Notice that there is a direct relationship between the *GoF* for map comparison and the one proposed for fire spread prediction, where $C = Hits$, $B = Misses$, and $A = FalseAlarms$.

Fig. 8 depicts the *GoF* measure as the degree of spatial overlap between two polygons increases. When spatial overlap gets maximized, the goodness of fit is high, and *GoF* suggests an identity between the map categories. On the other hand, when there is little "*insideness*", the goodness of fit is low, and identity is unlikely. An ideal *GoF* model will be especially responsive to incremental increases at the high overlap. This extra sensitivity will discriminate *excellent* from *good* fit, while distinguishing both of them from poor fits. This latter characteristic is vital when selecting a *GoF* as a fitness function for an evolutionary algorithm.

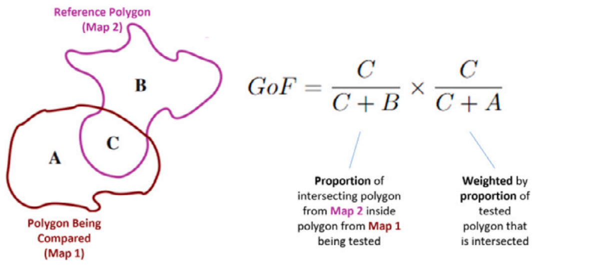

Reference Polygon (Map 2)
B
A
C
Polygon Being Compared (Map 1)
GoF = C/(C+B) × C/(C+A)
Proportion of intersecting polygon from Map 2 inside polygon from Map 1 being tested
Weighted by proportion of tested polygon that is intersected

Fig. 7. Goodness-of-Fit function for generic map comparison (Hargrove et al., 2006).

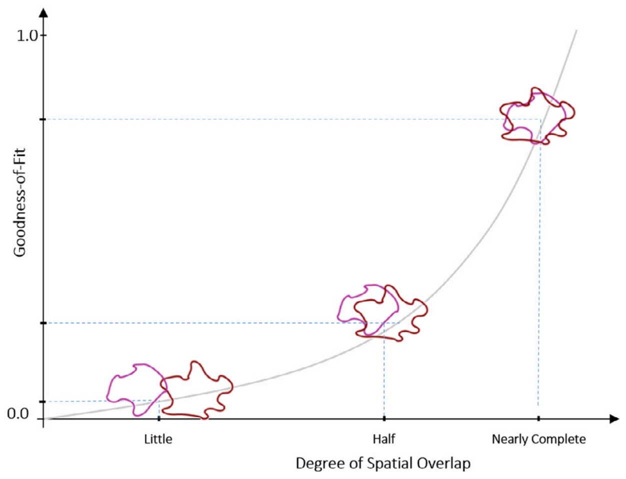


Fig. 8. Goodness-of-Fit measure as the degree of spatial overlap between two polygons increases (Hargrove et al., 2006).

3.2.2. *Early adaptive-evaluation strategy*

In order to overcome the slow time-to-result characteristic of the genetic algorithm, we adopt an ***adaptive evaluation*** technique (Fraga et al., 2020). In the past, the strategy used to keep the overall running time controlled was to simply discard individuals based on a deadline previously defined to avoid delaying an entire generation due to longer individuals. Or, even worse, previously filtering out those individuals whose execution time is estimated to be longer than a preset value. The problem that arise when discarding individuals is its impact on the population diversity, a crucial characteristic to the GA's ability to continue fruitful exploration of the search space.

The *adaptive evaluation* strategy overcome such a problem by keeping track of the behavior of each individual simulation at each GA generation. That is, once the simulation of a given individual starts, at the same time a monitor agent is launched attached to it, therefore, for example, for each GA generation with 100 individual population, there will exist 200 processes running, 100 forest fire spread simulations plus 100 monitor agents. Since we are dealing with a strict deadline constraint, the duration of one generation of the GA in the calibration stage is limited by a deadline previously defined to avoid delaying one entire generation of the GA due to a few longer individuals. This deadline is referred as *GAiterationTime* in Fig. 9, where the interaction between the simulation process and the monitor agent is depicted.

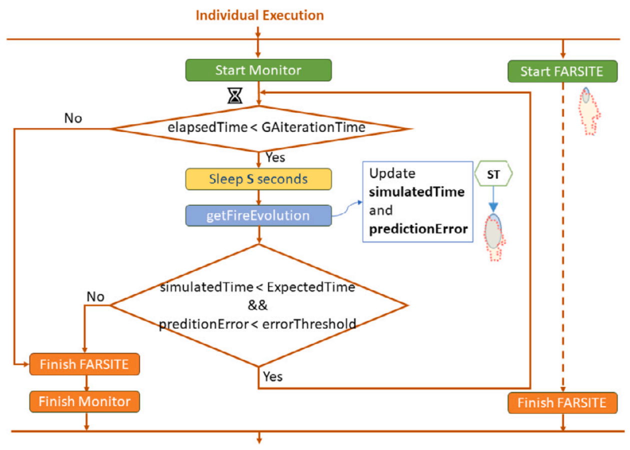

Individual Execution
Start Monitor
Start FARSITE
No
elapsedTime < GAiterationTime
Yes
Sleep S seconds
getFireEvolution
Update simulatedTime and predictionError
ST
No
simulatedTime < ExpectedTime && preditionError < errorThreshold
Yes
Finish FARSITE
Finish Monitor
Finish FARSITE

Fig. 9. Termination scenarios.

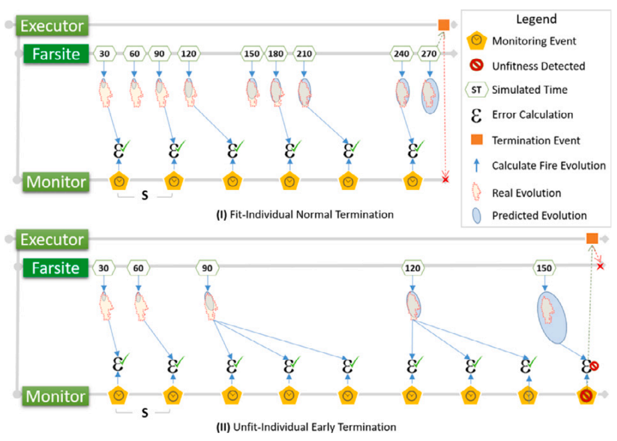


Fig. 10. Individual termination strategies.

The objective of the monitor agent is to periodically check the simulated time (*simulatedTime*) spent by the monitored individual and its *predictionError*, which is evaluated according to $\epsilon = 1 - GoF$. The execution time (clock time) of the monitor agent corresponds to the *elapsedTime* value in Fig. 9. This time is set to zero for each individual and it is increased in steps of S seconds, therefore, the monitor agent performs its checks each S seconds. Therefore, when the *elapsedTime* reaches the *GAiterationTime*, that means that the corresponding individual simulation cannot finish within the predefined deadline for the GA generation, the monitor agent then will finish the corresponding FARSITE simulation.

Furthermore, since the calibration stage uses past information data, the real spread time of the fire perimeter used to evaluate the *predictionError* is known. This time corresponds to the *ExpectedTime* value in Fig. 9. Then, in those cases that the error associated to the monitored individual overpasses a predefined *errorThreshold* and the individual simulation is still properly running, the monitor agent will also finish the corresponding FARSITE simulation. This situation corresponds to an *early termination* case. The reasoning behind the early termination is to avoid wasting computing time running individuals that are doomed to unfitness. If along its execution the monitor agent detects that the prediction deviates too much from the actual fire spread, it is considered safe to early terminate the individual.

The rest of the situation are individuals that terminates the simulation within the generation deadline and their errors are kept within certain limits. These cases corresponds to *normal termination* cases and correspond to the right straight dotted line in Fig. 9.

Fig. 10 depicts two individuals being executed and monitored according to the monitor flow defined in Fig. 9. The one described in Fig. 10 (I) terminates normally whereas the other described in Fig. 10 (II) is early terminated by the monitoring agent due to its unfitness based on its ongoing prediction error.

To keep the early terminated individuals in the population, we use a weighted version of the evaluation function. The formula shown in Eq. (2) is a weighted version in which *ExpectedTime* is the total time to be simulated whereas *simulatedTime* is the total time simulated until a normal or an early termination takes place.

$$fitness = \frac{ExpectedTime}{simulatedTime} \times GoF \tag{2}$$

In Fig. 10(I), in the scenario of a normal termination, for a total simulated time equals to 270 min (*ExpectedTime*), we can see that the FARSITE simulation reaches the 270 min (*simualtedTime*) just before the *Termination Event*, which is determined by the *GAiterationTime*. Therefore, the simulated and expected times are equals, thus, the fitness function is equal to the goodness of fitness metric. On the other hand, in the early termination scenario shown in Fig. 10(II), the penalty is directly proportional to the simulated time left to a normal termination, i.e., $fitness = \frac{270}{150} \times GoF = 1.8 \times GoF$. That is, at a *simualtedTime* equal to 150 min, the *predictionError* exceeds the *errorThreshold*, therefore, the monitor agent finishes the corresponding FARSITE simulation, which has only advanced the simulation process until simulation time equal to 150 min. The idea is to be able to compare unfinished individuals results against the actual burned area, considering how much has been simulated until the moment of the early termination: the sooner the termination, the greater should be the penalty. Therefore the main advantage of such strategy is to avoid wasting too much computing time running indubitably unfit individuals. Nevertheless, thanks to the weighted evaluation function, those individuals can still be evaluated and be unlikely selected to the following generations, ensuring more diversity to the GA evolution process.

3.3. Optimized resource allocation planner

The *Optimized Resource Allocation Planner* is the component responsible for defining the resources where the calibration and the prediction will be executed. As a black box, it simply returns the list of available machines. But, to its fullest extent, it is a domain-specific module, crafted to statistically model the forest fire core simulator based on the specified computational resources properties, and determine the user preferences through strict deadlines or utility functions. Together, these information are used to acquire the computing resources that best fit the user needs. Its three sub-components (illustrated in Fig. 2) are the following: *Runtime and Memory Consumption Models*, *Price/Resource Model* and *User's Utility Model*. Subsequently, each of these elements is described in more detail.

3.3.1. Runtime and memory consumption models

In order to estimate the execution time of a fire spread simulation, it is necessary to perform a large set of executions of the underlying simulator with different input parameters configurations on the target architecture. The parameters of each individual simulations, together with the runtime and memory consumption are recorded and, afterwards, they are used to analyze its behavior. These steps are made offline, which means that they must be carried out before the fire occurrence. Typically, this offline study will be done for those areas with a high wildfire risk. As forest fires show seasonal behavior and an occurrence pattern that affects the same zones every year, several indexes have been developed to determine the fire risk and fire spread potential. Besides past occurrences, these indexes take into account the underlying vegetation and the meteorological conditions on a given area. The indexes most used are the Fire Weather Index (FWI, 2008) and the Haines Index (Haines, 1988). FWI is a meteorologically based

Table 1
Input parameters distributions.

Input	Distribution	μ; σ	Min; Max
Vegetation model	Uniform	–	1; 13
Wind speed	Normal	70; 10.0	–
Wind direction	Normal	45; 5.0	–
Dead fuel moisture	Uniform	–	2; 15
Live fuel moisture	Uniform	–	50; 100
Adjustment factor	Uniform	–	0.0; 2.0

index used worldwide to estimate fire danger that accounts for the effects of fuel moisture and wind on fire behavior and spread. Haines uses the meteorological conditions as an indicator of the potential risk of wildland fires, considering the stability and moisture content of the lower atmosphere. In this way, fire analysts can run the executions and prepare these models in the moment prior to the fire occurrence.

Our methodology is comprised of 3 steps: **1.** prepare the training database; **2.** trace the information regarding runtime and memory consumption; **3.** build the regression models.

- **Training database**: Currently, we work with training databases composed of 10,000 up to almost 40,000 different scenarios. Each one represents a random individual, an input configuration for the target scenario. For the use case presented in this study, the distribution of each input parameter corresponds to the ones specified in Table 1. The vegetation models correspond to the 13 standard Northern Forest Fire Laboratory (NFFL) fuel models (Anderson, 1982).
- **Determination of execution time and memory consumption**: For each execution we trace the *Runtime* and the maximum *Resident Set Size* (the portion of memory occupied by a process held in main memory) metrics. Fig. 11 depicts the multivariate relation between runtime and the most important individual parameters. As it can be seen, after calculating the *Pearson Coefficient*, we conclude that wind speed and humidity are the two parameters that most impact on the runtime. The scattering plot confirms visually such importance. Similar relations happen considering memory consumption.
- **Building the regression models**: We evaluated 5 multiple regression techniques to build a cross-validated regression model representing the relation between the parameters and the response metrics. The models are: *Multiple Linear Regression*, *Principal Component Regression*, *Partial Least Square*, *Regularized Regression Elastic Net* and *Multivariate Adaptive Regression Splines (MARS)*. From the built models we choose the best one based on the standard deviation of the residuals measure, i.e. the prediction errors, also called *Root Mean Square Error* (*RMSE*).
  Table 2 shows a summary of the prediction performance for each model.

Based on the prediction performance, we have chosen MARS as the best model to characterize *FARSITE* runtime based on input parameters. Considering *median* and *mean* columns, we can see that it shows results with the smaller errors. MARS is a form of non-parametric regression analysis technique and it is an extension of linear models that automatically models non-linearities and interactions between predictor variables (Friedman, 1991). Equivalent results apply to the memory consumption analysis. Fig. 12 shows the graphical representation of the two built models. From the two graphics we can observe that there is a region where runtime and memory consumption grow significantly. Considering an informed two-stage prediction scheduler, individuals located in such region should be allocated in the most efficient available resources in order to decrease the overall calibration and the final fire spread prediction time.

Once this methodology is followed, only a last step remains: the application of the resulting model with the scenario describing the ongoing fire, in order to assess in advance the execution time its simulation will produce. This action supposes a negligible cost, in terms of time overhead (on the order of a few seconds). Accurate *FARSITE* runtime and memory consumption models are used to predict performance of actual GA individuals in the specified cloud resources and, then, determine the resulting monetary cost.

3.3.2. Price and resource models

In this work, we consider an *Infrastructure-as-a-Service* (*IaaS*) cloud system, in which a number of data centers deliver on-demand storage and compute capacities over the Internet. These computational resources are provided in the form of an abstract unit of compute and storage called *Virtual Machine* (VM for short), object storage, or remote file systems volumes. In a cloud, VMs are offered in several types, each of which has distinct characteristics such as different numbers of CPUs, amount of memory, and network bandwidth capacity. We have chosen Amazon Web Services (AWS) as the public cloud provider. AWS is a comprehensive and broadly adopted cloud platform, offering fully featured services from data centers globally. We then use the resources characteristics, including their pay-as-you-go prices, to characterize the *Price/Resource Model*, against which we build the *Runtime and Memory Consumption Models*. Assessing the requirements of the forest fire prediction model and selecting the appropriate instance family is the starting point for the application performance testing. It is crucial to identify how the model needs compare to different instance families (e.g. if it is compute-bound, memory-bound, network-bound, etc.), and to size the workload to identify the appropriate instance size. Related to **Resource Model**, Amazon Web Services (AWS) Elastic Compute Cloud (EC2) provides a variety of instance types, each one providing different combinations of CPU, memory, disk, and networking. As of May, 2023, there are six families optimized for several types of applications and, for each one, ten different instance types. Table 3 presents the families, the number of instances available, the number of CPUs, and the range of on-demand cost per hour for each one of them.

Furthermore, the **Cost Models** must be taken into account, as with everything, there are costs associated with the cloud usage. With the many economic benefits of the cloud, this may often seem negligible for simple workloads. For applications with highly parallel workloads, the cost may easily surpass a thousand of dollars in a couple of hours if the instance provision is not made in a diligent way.

Table 4 presents the different cost models used to pay for Amazon EC2 instances. Although the *Saving Plans* might be an interesting model to high-demanding fire analysts, we focus on the *On-Demand* and *Spot Instances* models to perform the characterizations.

3.3.3. User's utility model

Once the models are built, they can be actioned upon the utility functions to capture user's satisfaction over prediction time and monetary cost of the overall fire spread prediction and, therefore, define the list of machines that will be provisioned to run the prediction engine. Utility functions are commonly used to communicate the value of work and other quality of service aspects such as its timely completion (Walsh et al., 2004). A utility function commonly used is the simplified version of the *Cobb–Douglas* production function (Holmes Thomas and Calkin David, 2013).

We consider the *FARSITE* characterization described in Section 3.3.1 to define the most cost-effective AWS EC2 instance types to be used as the underlying virtual infrastructure. As a result, the *Compute Optimized* family listed in Table 5 has been chosen. The *Optimized Resource Allocation Planner* is fed with these prices and resource characteristics and, based on the *Core Runtime and Memory Consumption Models* (see Fig. 2), generates as output the *average cost per individual* per *calibration time*. In Fig. 13 we can see the result for configuration spanning a potential cluster with $1, 2, 4, 8, 16$, and 32 nodes. Estimations are grouped per potential cluster size, as instances are immediately terminated when there is no workload to be processed.

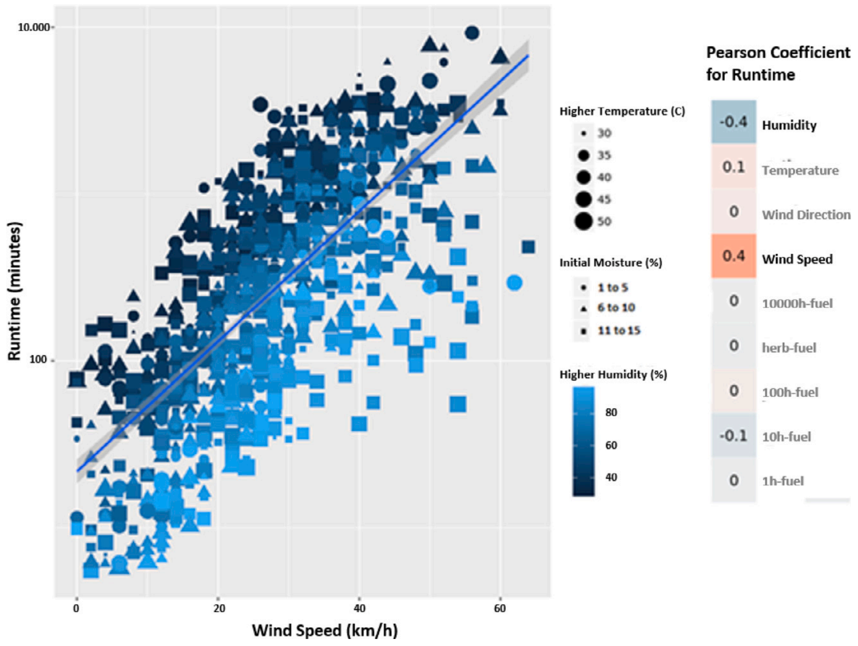


Fig. 11. Multivariate analysis of the runtime for *FARSITE* executions.

Table 2
Standard deviation of the prediction errors - Root Mean Square Error (RMSE).

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
Multiple Linear Regression	1022.458	1267.519	2215.918	2591.679	3984.889	4889.377
Principal Component Regression (PCR)	989.443	1263.651	2234.553	2590.982	3992.400	4867.281
Partial Least Square (PLS)	1018.913	1266.828	2216.746	2590.798	3984.444	4887.721
Regularized Regression Elastic Net	760.384	1030.713	2202.121	2533.858	4049.466	4978.295
Multivariate Adaptive Regression Splines (MARS)	746.422	1112.445	**1670.995**	**2249.241**	3430.974	4624.153

Last step is to decide which configuration will be used based on the user's preference. We model the user's preference as a utility function, a mathematical function that ranks alternatives according to their utility to an individual. As it has been previously mentioned, a commonly used utility function is the simplified version of the Cobb–Douglas production function $u(x_1, x_2) = x_1^{b_1} x_2^{b_2}$, where b_1 and b_2 are positive numbers describing the preferences of the consumer. Applying it for $b_1 = \frac{1}{2}$ and $b_2 = \frac{1}{2}$, i.e. the user is equally concerned with price and runtime, the resulting ranking of preference is shown in Table 6. There are $6 \times 6 = 36$ configurations in total (*the number of instance types* $\times$ *the number of options for cluster size*). Considering the results, the configuration that gives more utility to user preferences is the one with 4 nodes of type c5.2xlarge. For different parameters of the utility function, a new ranking is provided, but it is up to the user, i.e. the *wildfire analyst*, to inform his or her preferences. The *Resource Provisioner* module can provision the corresponding virtual resources and then trigger the *Scheduler* to perform the scheduling task.l

3.4. Secondary components

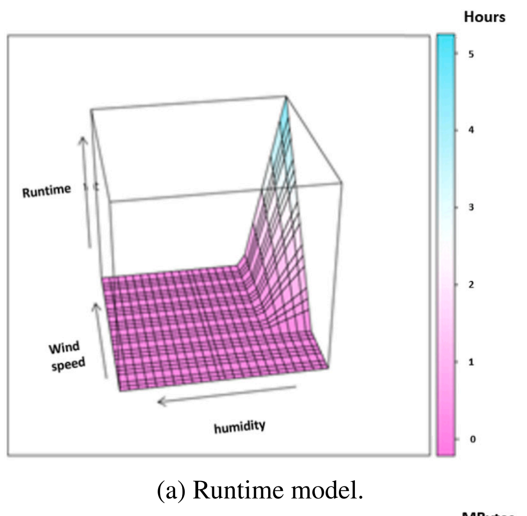
Finally, to properly deploy the complete *CuCo* architecture, several considerations must be taken into account related to other components required. Regarding the **Storage**, as an urgent computing solution, it is necessary to have on-demand access to the input data. It is needed a persistent repository consisting of all the static data for the areas with a greater probability of forest fire occurrence (those with high fire danger indices). For these static data, as a data lake, the solution leverage object storage services that offers industry-leading scalability, data availability, security, and performance. Furthermore, a **Task Queue** module is required to integrate the different components providing an *Application Programming Interface* (API) to create queues as well as send, receive, and delete messages. Messages are used to represent simulation tasks, and to queue states to trigger orchestration or scheduling activities. The **Scheduler** is in charge of assigning resources to perform computing tasks and it determines, which resources are valid placements for each task according to their constraints. Finally, the **Resource Provisioner** is responsible for providing the virtual resources in which the actual simulation will be run. After understanding the memory consumption and runtime of the core simulator, a suitable memory or compute-intensive pool of instances can then be provisioned.

4. Results and discussion

In this section, the proof-of-concept implementation of the high-level architecture of the cloud-based solution is presented and discussed. Later, the solution is validated against a case study of a challenging and destructive forest fire scenario.

4.1. Implementation details

The implementation uses well-established cloud tools, being possible to be easily deployed on a public cloud or on-premise infrastructure. It also relies on scalable object-storage service to allow the efficient parallel retrieval and storage of input and output data. The *Resource Provisioner* is linked to the *AWS EC2* infrastructure, *FARSITE* is the *Core Simulator* and it has been isolated as a containerized application running on *Docker* runtime engine.



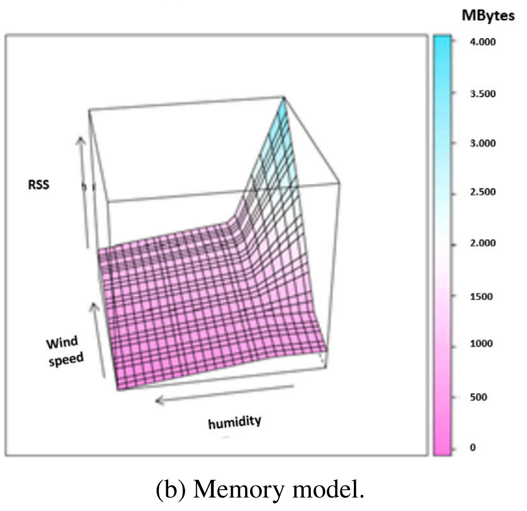


Fig. 12. MARS runtime and memory models for *FARSITE* executions.

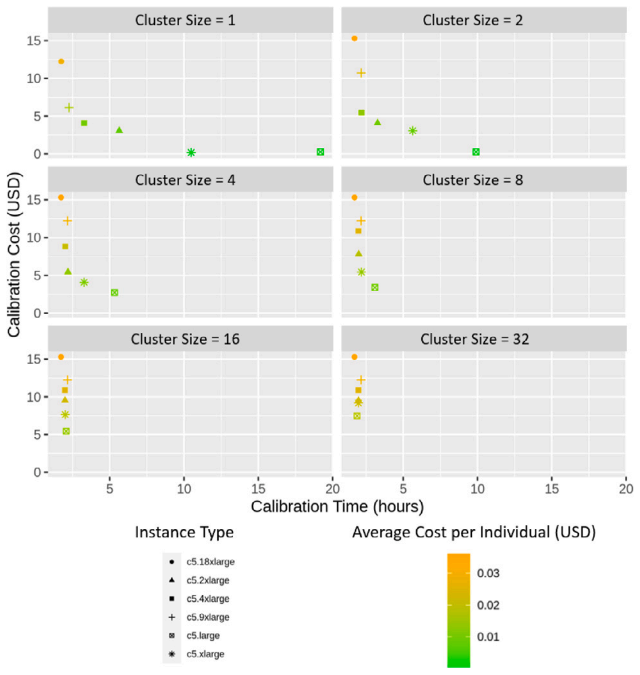


Fig. 13. Estimations for cloud cost for the calibration phase.

Table 3
AWS EC2 families of instances.

Family	Recommendation	#	vCPU	$
General purpose	Variety of diverse workloads. Balance of compute, memory, and networking resources.	114	1 to 192	0.005 to 9.677
Compute optimized	Compute bound applications that benefit from high performance processors.	85	1 to 192	0.042 to 9.158
Memory optimized	Workloads that process large data sets in memory.	96	1 to 128	0.056 to 34.68
Accelerated computing	Use hardware accelerators, or co-processors, to perform functions more efficiently than is possible in software running on CPUs.	14	4 to 96	0.274 to 9.389
Storage optimized	Workloads that require high, sequential read and write access to very large data sets on local storage.	27	2 to 128	0.172 to 12.109
HPC optimized	Built to offer the best price performance for running HPC workloads at scale.	–	–	–

Table 4
AWS EC2 cost models.

Model	Description
On-demand	pay for compute capacity by the hour or the second depending on which instances are run. No longer-term commitments or upfront payments are needed.
Spot instances	request spare EC2 computing capacity for up to 90% off the On-Demand price.
Savings plans	offers low prices on EC2 usage, in exchange for a commitment to a consistent amount of usage (measured in $/h) for a one- or three-year term.
Dedicated host	a physical EC2 dedicated server.

Table 5
AWS EC2 compute instances family.

Model	#vCPU	Memory (GiB)	Cost per Hour
c5.large	2	4	$ 0.085
c5. × large	4	8	$ 0.17
c5.2 × large	8	16	$ 0.34
c5.4 × large	16	32	$ 0.68
c5.9 × large	36	72	$ 1.53
c5.18 × large	72	144	$ 3.06

The stream of generations from the *Genetic Algorithm* is responsible for generating all the compute demands that will be fulfilled by the orchestrating component (*Kubernetes*). For each generation, a *Kubernetes* Job performs a coarse parallel processing using a work queue. The initial population is randomly generated and, for each one of its individuals, a task definition is created and pushed to the task queue.

The *Kubernetes* Job starts one *FARSITE* pod for each task in order to obtain maximum parallelism. Each of them takes one task from the

Table 6
Resulting ranking of preference to run the calibration.

Rank	Utility	#Nodes	Instance Type
1	0.6545	4	c5.2 × large
2	0.5598	2	c5. × large
3	0.5418	1	c5. × large
4	0.5044	16	c5.large
5	0.4316	2	c5.2 × large
...	...	...	...
35	0.1327	2	c5.large
36	0.0916	1	c5.large

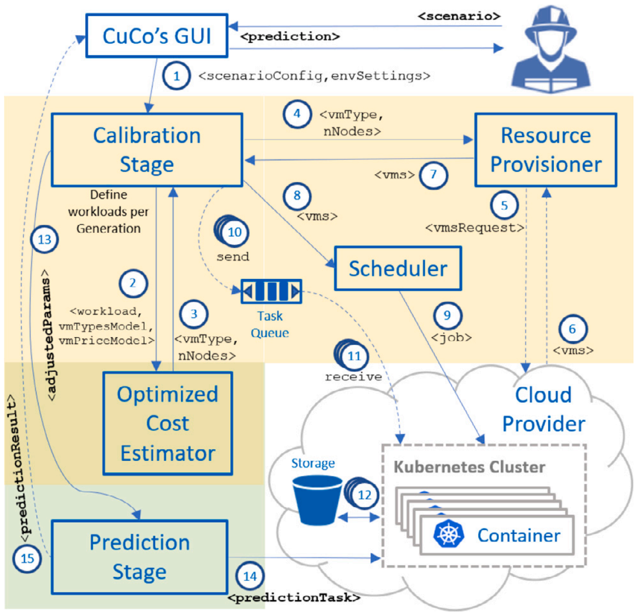


Fig. 14. Interaction flow between the main components of the cloud-based two-stage fire spread prediction.

message queue, processes it, and gets terminated. Once the current generation is entirely evaluated, a new one can be computed, and the message queue can be filled up again. One task represents a scenario describing parameters for every single *FARSITE* execution, taking the role of an individual of the *Genetic Algorithm*. Its message must include the input bucket and the input directory name. Likewise, the output bucket needs to be informed, accompanied by the name of the directory to which the output files are pushed.

A *Docker* template has been prepared to build a containerized version of *FARSITE*. In its first execution, the container attempts to consume a message from the task queue. In case there is at least one task, it proceeds to run the scenario that the task describes and, in the end, push the results to the output directory. At the end of the execution, the message gets deleted from the queue. Once there is no message to be consumed, the pod gets terminated.

Once the queue is empty, the *Genetic Algorithm* can take back the control of the execution and, if the evolution is not complete yet, applies the genetic operators (elitism, selection, crossover, and mutation) resulting in a new generation. The queue is filled up again and *Kubernetes* takes the control in order to deploy, start, and at the end terminate all the pods used to run the individuals.

The process is repeated until the pre-defined number of generations is reached. In the end, the parameters and files from the best individuals are made available in the output bucket and they can then be used in the *Prediction Stage*. Fig. 14 illustrates the interaction flow between the main CuCo's components for both the *Calibration* and *Prediction* stages. The wildfire analyst informs the scenario's configuration **(1)**, with all the static data available in a storage service. For each generation,

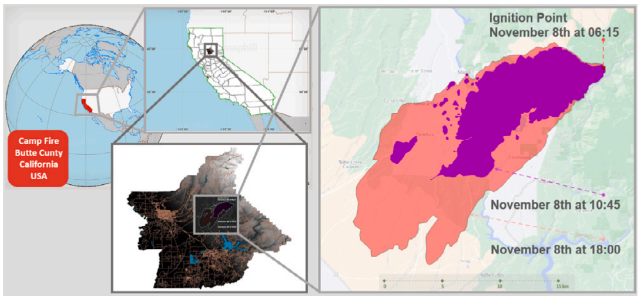

Camp Fire Butte Cunty California USA
Ignition Point November 8th at 06:15
November 8th at 10:45
November 8th at 18:00

Fig. 15. Camp Fire location (Butte County, California - USA), and its initial perimeters at 06:15, 10:45, and 18:00.

the *Optimized Resource Allocation Planner* gets consulted **(2)**, and an allocation plan is defined **(3)**. The list of resources to be acquired is then informed to **(4)** and provisioned by the *Resource Provisioner*. It is responsible for requesting **(5)**, receiving **(6)**, and passing them to the *Two-Stage Prediction Engine* **(7)**. The Scheduler **(8)** receives the list of resources used to start a *Kubernetes* Job **(9)**. The *queue* gets fed with the tasks **(10)** used to define each *FARSITE* scenario that later will be consumed **(11)** by the pods running in the *Kubernetes* cluster. Results are sent to the output bucket **(12)** until the end of the *Calibration Stage*. Once the calibration finishes, the adjusted parameters get passed on to the *Prediction Stage* **(13)** that runs the actual prediction on a high-performance container **(14)**. In the end, the prediction result **(15)** is made available to the wildfire analyst.

4.2. Experimental study

The enhancements to the two-stage prediction framework presented in this work have been validated against fire scenarios in the Mediterranean region at Spain and Greece (Fraga et al., 2020, 2021, 2022). These scenarios have been thoroughly explored, studied, and compared in previous works by our research team (Artés et al., 2013). In this work, we have set up an experimental study to validate the *CuCo* platform against the *Camp Fire*, a deadliest and destructive wildfire that occurred in California - USA in the year 2018.

The West part of the USA is one of the regions most affected by forest fires during high-risk seasons. In particular, the state of California is frequently affected by wildfires. In relation to the two-stage prediction methodology, *Camp Fire's* calibration is particularly challenging due to the unusual weather conditions that cause the fire to spread faster than expected. It is a resource-demanding scenario, represents an urge for timely predictions, and gives much less time to act.

4.2.1. Camp Fire's scenario

The *Camp Fire* was, at that time, the most destructive wildfire in California's history. The fire started on Thursday, November 8, 2018 at 6:15 a.m., in Northern California's Butte County (see Fig. 15). Ignited by a faulty electric transmission line, the fire originated above several communities and an east wind drove the fire downhill through developed areas. Sustained winds of 40 to 48 km/h, with gusts ranging from 65 to 80 km/h, drove rapid fire spread. Drought was a major factor for the fire intensity and spread. Relative humidity across the area was generally below 20 percent. Exhibiting extreme fire spread, fireline intensity, and spotting behaviors through rural communities, an urban firestorm also formed in a foothill town.

The fire advanced nearly 24 km in the first 12 h, burning over 22,000 hectares. This means that the average rate of spread of the fire was, approximately, 2 km/h but, as it will later on explained, the rate of spread was more than twice this velocity during the first hour of the fire evolution. Only after the arrival of the first winter rainstorm of the season, the fire reached 100 percent containment after seventeen days on November 25 (Maranghides et al., 2021).

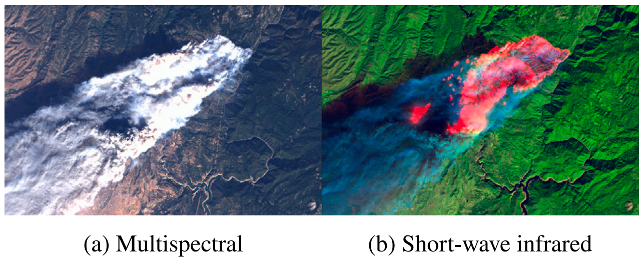


Fig. 16. Landsat 8 multispectral and infrared satellite images captured at 10:45 a.m., four hours after fire ignition.

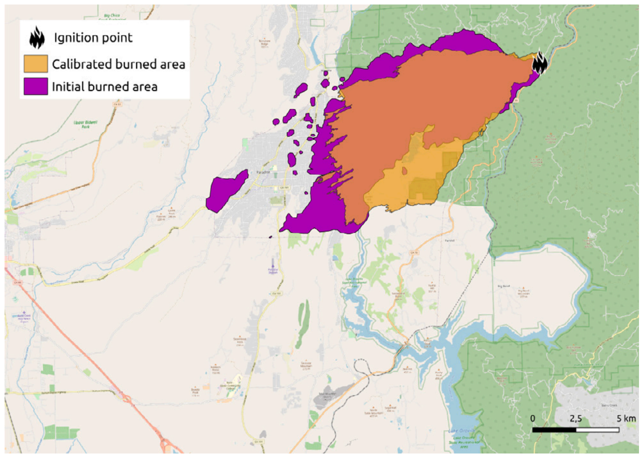


Fig. 17. Result of calibration stage from 6:15 to 10:45. Calibration perimeter (P_1') is shown overlaid in mustard while the actual fire spread is shown in purple (P_1). Calibration shape in lighter shade represents *False Alarms*, while darker one represents *Hits*.

In the following sections, we analyze each iteration of the whole prediction scheme separately, in order to better understand the results obtained when applying the two-stage strategy. Although the fire spread over 17 days in total, we consider only the perimeters in the propagation occurred in the first 12 h, which was the most critical ones. The propagation results have been compared using the function stated in Eq. (1), which evaluates the goodness-of-fit of a fire spread prediction.

4.2.2. Calibration window

The time window selected for the *Calibration* goes from 6:15 a.m. to 10:45 a.m., covering from the ignition point until the first complete perimeter extracted from a satellite image. That is, the *calibration* stage covers the initial 4,5 h of the fire event, while the corresponding *prediction* stage covers the following 7,25 h (from 10:45 to 18:00).

Fig. 16 shows Landsat 8 captures at about 10:45 a.m. PST on Thursday, Nov. 8, just around four hours after fire ignition. In its corresponding short-wave infrared band, the images are processed to show green vegetation in the near infrared (*NIR*), and dead vegetation and the fire in red colors using shortwave infrared (*SWIR*), with the burn front clearly visible. It shows the full extent of the actively burning area, and the red patches are fires that leap in front of the primary burn front (spot fires). The fire was growing from 6:25 to 7:25 at a rate of approximately 5 km per hour, decreasing its rate of spread to approximately 3 km/h during the rest of the calibration window (Maranghides et al., 2021).

We use actual values of elevation, slope and aspect of the terrain, fuels (vegetation types) and canopy cover. The genetic algorithm gets configured to evolve for 10 generations, each one with a population size set to 64 individuals. The probabilities of crossover and mutation used in the executions are 0.8 and 0.1, respectively. Those values are used empirically and used to present good calibration results (Artés et al., 2017). Each individual gets simulated, and the resulting fire perimeter is compared to the actual fire spread using the weighted *GoF* function, as described in Section 3.2.1.

First of all, we need to be sure that there are enough available pods/cores to run all individuals in parallel and then avoid queuing *FARSITE* tasks (see Section 3). In the present case, we are not concerned about what instances get actually chosen to host the pods, as long as we can obtain the maximum level of parallelism at the same performance level. Although we could have by-passed the *Optimized Resource Allocation Planner* informing directly the cluster configuration of preference, we obtained the same result setting the parameters of the utility function (see Section 3.3.3). Setting $b_1 = \frac{9}{10}$ and $b_2 = \frac{1}{10}$, i.e. the user is much more concerned with the runtime than with the cost. In this case, the chosen instance is again the *c5.2xlarge*, but this time for the configuration with 8 nodes.

The results obtained for this calibration stage are shown in Fig. 17. As it can be observed, although not capturing the spotting behavior of the fire, it resembles the main trend of the actual fire shape, resulting in a somewhat good calibration. The parameters corresponding to this best individual can then be immediately fed into the *prediction* stage.

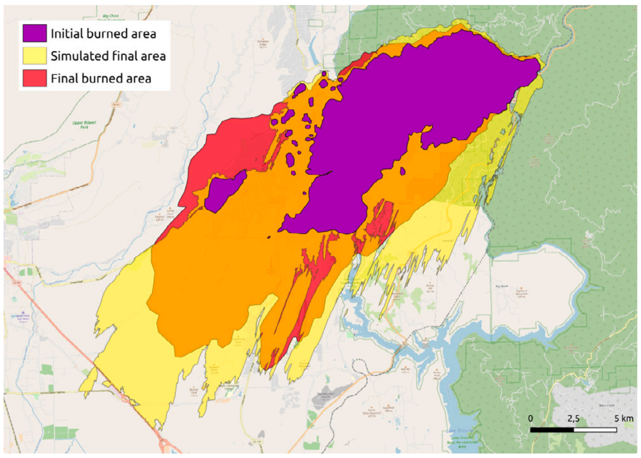


Fig. 18. Prediction results from 10:45 to 18:00. Prediction shape (P_2') is shown in yellow (*False Alarms*) and orange (*Hits*). The actual fire spreads are shown in purple for perimeter (P_1) while perimeter (P_2) is shown in red and orange, to account for the *Hits* in the prediction.

4.2.3. Prediction window

Prediction stage goes from 10:45 to 18:00 and uses the adjusted input data resulting from the *calibration* stage. This simulation is challenging as its reference fire affects both rural communities, urban constructions, and forest fuel. Wildfires in these conditions do not spread exactly as modeled in the fire propagation reference patterns. Besides, the expansion of the high number of spot fires increases the prediction time.

Notwithstanding these caveats, the prediction managed to capture the flanks of the actual fire spread, thanks to the incorporation of wind field model. Fig. 18 shows the resulting prediction perimeter, contrasted with the actual fire spread at the same time. The resulting prediction overestimates the fire spread, but it clearly describes the three main spread direction of the fire. Overestimation of the fire spread is not really a big concern, as in the real case the wildfire does not spread freely due to human intervention.

Curiously, an important fire front on the north-west part was not detected. This was probably due to that part being the one affected by an urban firestorm in the foothill town, difficult to modeling as it

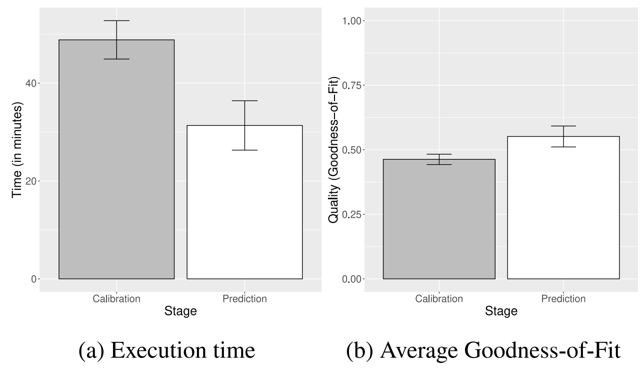


Fig. 19. Execution time and quality of prediction for calibration and prediction stages of the Camp Fire scenario, running on a virtual cluster comprised of 8 instances of type *c5.2xlarge*.

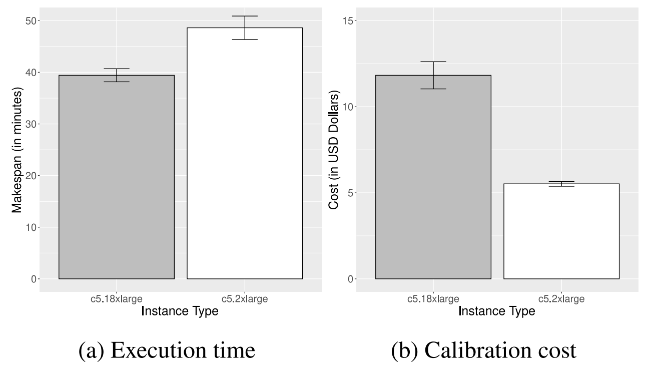


Fig. 21. Comparison of calibration execution time and cost between clusters of two selected instance types: *c5.18xlarge*, offering more performance, and *c5.2xlarge*, most cost-effective.

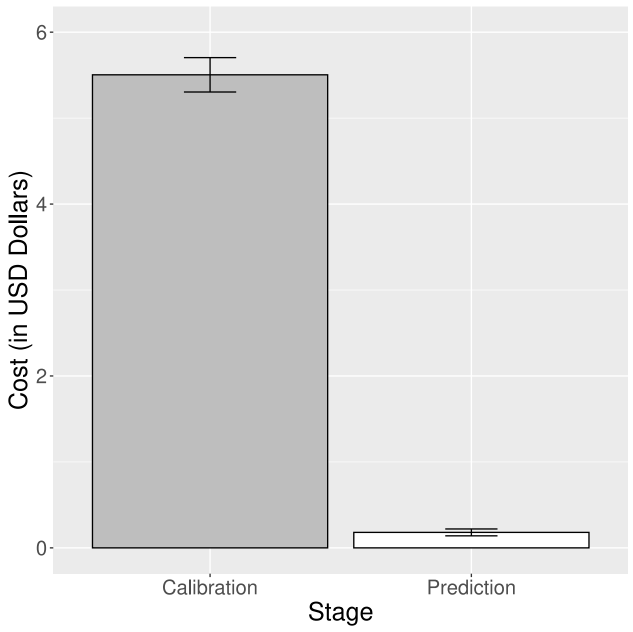


Fig. 20. Cloud cost for calibration and prediction stages.

propagated mainly by new spot fires over a *Wildland–Urban interface* (WUI).

The execution of the two-stage prediction scheme has been repeated 10 times and each data point in Figs. 19(a) and 19(b) shows the average of these runs. The error bar represents the corresponding standard deviation. Fig. 19(a) shows the execution time for the calibration and prediction stages, while Fig. 19(b) shows the average *goodness-of-fit* (i.e. prediction quality, see Eq. (1)) also for both stages.

Regarding the execution time, the most important is the timely-response. As the real event occurs in a time-window of 7 h and 15 min, the prediction must be delivered in a fraction of that time. The two-stage prediction takes around 1 h and 30 min (55 min from calibration and 35 min from the prediction stage), around 20% of the prediction window.

As we can see in Fig. 20, the prediction cost is just a minor fraction of the overall two-stage cost. Due to the high-parallel and compute-intensive nature of the genetic algorithm calibration, its execution consumes 95% of the two-stage prediction budget. Although the prediction stage execution time is around 60% of the calibration stage, we can see that when it comes to the cost, its comparison with the calibration is fairly negligible.

Although we have engineered the solution to be cost-effective, i.e. able to balance between cost and time-to-response, in an operational event, the fire analyst may want to allocate the most powerful resource, no matter how much it cost. In this case, the *Optimized Resource Allocation Planner* can be by-passed and the *Resource Provisioner* allocates the cluster configuration informed by the analyst.

For example, if the fire analyst believes that having more virtual machines the workload can be evenly spread between them, resulting in an improved performance, he or she can then choose a configuration with 4 instances of type *c5.18xlarge*, with 288 *vCPUs* in total ($4 \times 72 = 288$). We would then have the results showed in Fig. 21, where we can surprisingly see indeed a significant reduction in the time needed for the calibration stage. This result cannot be explained by considering the number of cores (or *vCPUs*) alone, as a virtual cluster with 8 instances of type *c5.2xlarge* has exactly 64 *vCPUs* ($8 \times 8 = 64$), sufficient to run all the individual of a given generation in parallel. The reason must be the underlying resources, as the cloud provider usually allocated the most powerful instances in high-end physical machines. On the other hand, such calibration cost double in comparison with the cluster suggested by the *Optimized Resource Allocation Planner*.

In the end, it is a decision justifiable in the presence of the trade-off between cost and performance while facing a natural hazard. In the day-to-day activities, involving planning wildfire prevention and suppression, where thousands of wildfire simulations need to be run, the decision needs to be well informed and cost-effective.

5. Conclusion

Forest fire is a significant natural hazard that every year cause important damages in areas with high fire risk indexes. Wildfire models and their implementation in simulators can provide estimations of fire behavior, but their results are affected by high level of data uncertainty. Besides, input parameters are difficult to know or even to estimate so there is a need of strategies to minimize this uncertainty and to provide better predictions. When applied to a real case scenario in production, there is also an extra key factor that challenge the forecast process: the response time. In such situation, late results are useless.

In this work, we present a cloud-based solution for the data uncertainty problem in forest fire spread prediction. We build up our work on an established two-stage fire spread prediction model. Our solution uses a goodness-of-fit function for the genetic algorithm calibration phase, improving the genetic algorithm convergence and decreasing the response time for the calibration stage. We use an adaptive evaluation technique to evaluate the individuals, based on a periodic monitoring of their fitness, avoiding the waste of computing time running undoubtedly unfit individuals. This strategy is able to, at the same time, simplify implementation and improve response time of the critical calibration stage of the two-stage prediction framework.

We have also set up an experimental study to validate the platform against the *Camp Fire*, a deadliest and destructive wildfire that occurred in California - USA in the year 2018. The results showed that the final prediction has successfully captured the main fire growth trend for the first 12 h of fire, using the first 4 h of data to calibrate and then predict the fire spread for the next 8 h.

It is part of our ongoing work to evaluate the cloud platform and an strict deadline policy using other real fire spread scenarios. We also plan to follow a dynamic approach to terminate the genetic algorithm evolution, being able to save time that can be used in the prediction phase. Another non-excluding option is to evaluate the effect of increasing the population size in favor of a decrease in the number of generations.

CRediT authorship contribution statement

Edigley Fraga: Investigation, Writing – original draft. **Ana Cortés:** Funding acquisition, Supervision, Writing – original draft, Writing – review & editing. **Tomàs Margalef:** Funding acquisition, Supervision, Writing – review & editing. **Porfidio Hernández:** Supervision, Writing – original draft. **Carlos Carrillo:** Investigation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data has been shared in corresponding author github repository.

References

Abdalhaq, B., Cortés, A., Margalef, T., Luque, E., 2005. Enhancing wildland fire prediction on cluster systems applying evolutionary optimization techniques. Future Gener. Comput. Syst. 21 (1), 61–67.

Altintas, I., Block, J., de Callafon, R., Crawl, D., Cowart, C., Gupta, A., Nguyen, M., Braun, H.-W., Schulze, J., Gollner, M., Trouve, A., Smarr, L., 2015. Towards an integrated cyberinfrastructure for scalable data-driven monitoring, dynamic prediction and resilience of wildfires. Procedia Comput. Sci. 51, 1633–1642, proc. of the International Conference On Computational Science, ICCS 2015.

Anderson, H.E., 1982. Aids to Determining Fuel Models for Estimating Fire Behavior. Technical Report, U.S. Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station, http://dx.doi.org/10.2737/int-gtr-122.

Arca, B., Ghisu, T., Casula, M., Salis, M., Duce, P., 2019. A web-based wildfire simulator for operational applications. Int. J. Wildland Fire 28 (2), 99–112. http://dx.doi.org/10.1071/WF18078.

Artés, T., Cencerrado, A., Cortes, A., Margalef, T., 2013. Relieving the effects of uncertainty in forest fire spread prediction by hybrid MPI-OpenMP parallel strategies. In: Proc. of the International Conference on Computational Science. ICCS 2013, Vol. 13, (2), pp. 2277–2287.

Artés, T., Cencerrado, A., Cortés, A., Margalef, T., 2017. Time aware genetic algorithm for forest fire propagation prediction: exploiting multi-core platforms. Concurr. Comput.: Pract. Exper. 29 (9), e3837, e3837 cpe.3837.

Asensio, M., Cascón, J., Laiz, P., Prieto-Herráez, D., 2023. Validating the effect of fuel moisture content by a multivalued operator in a simplified physical fire spread model. Environ. Model. Softw. 164, 105710. http://dx.doi.org/10.1016/j.envsoft.2023.105710, URL: https://www.sciencedirect.com/science/article/pii/S1364815223000968.

Bakhshaii, A., Johnson, E., 2019. A review of a new generation of wildfire–atmosphere modeling. Can. J. Forest Res. 49 (6), 565–574. http://dx.doi.org/10.1139/cjfr-2018-0138.

Benali, A., Ervilha, A.R., Sá, A.C., Fernandes, P.M., Pinto, R.M., Trigo, R.M., Pereira, J.M., 2016. Deciphering the impact of uncertainty on the accuracy of large wildfire spread simulations. Sci. Total Environ. 569–570, 73–85. http://dx.doi.org/10.1016/j.scitotenv.2016.06.112, URL: https://www.sciencedirect.com/science/article/pii/S0048969716312852.

Cencerrado, A., Cortés, A., Margalef, T., 2014. Response time assessment in forest fire spread simulation: An integrated methodology for efficient exploitation of available prediction time. Environ. Model. Softw. 54, 153–164.

Denham, M.M., Waidelich, S., Laneri, K., 2022. Visualization and modeling of forest fire propagation in patagonia. Environ. Model. Softw. 158, 105526. http://dx.doi.org/10.1016/j.envsoft.2022.105526, URL: https://www.sciencedirect.com/science/article/pii/S1364815222002262.

ESA, 1998. Copernicus program - Europe's eyes on earth. https://www.copernicus.eu/en.

Finney, M.A., 1998. FARSITE: Fire area simulator-model. Development and evaluation. USDA Forest Service Research Paper, RMRS-RP-4.

Forthofer, J., Shannon, K., Butler, B., 2009. Simulating diurnally driven slope winds with WindNinja. In: Proceedings of 8th Eighth Symposium on Fire and Forest Meteorology. American Meteorological Society, Kalispell, MT. Boston, MA.

Fraga, E., Cortés, A., Cencerrado, A., Hernández, P., Margalef, T., 2020. Early adaptive evaluation scheme for data-driven calibration in forest fire spread prediction. In: Computational Science – ICCS 2020. Springer International Publishing, Cham, pp. 17–30.

Fraga, E., Cortés, A., Margalef, T., Hernández, P., 2021. Cloud-based urgent computing for forest fire spread prediction under data uncertainties. In: 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics. HiPC, pp. 430–435. http://dx.doi.org/10.1109/HiPC53243.2021.00061.

Fraga, E., Cortés, A., Margalef, T., Hernández, P., 2022. Efficient cloud-based calibration of input data for forest fire spread prediction. In: 2022 IEEE 18th International Conference on E-Science. E-Science, pp. 128–136. http://dx.doi.org/10.1109/eScience55777.2022.00027.

Friedman, J.H., 1991. Multivariate adaptive regression splines. Ann. Statist. 19 (1), 1–67. http://dx.doi.org/10.1214/aos/1176347963.

FWI, 2008. Fire weather index system. https://www.nwcg.gov/publications/pms437/cffdrs/fire-weather-index-system.

Garg, S., Aryal, J., Wang, H., Shah, T., Kecskemeti, G., Ranjan, R., 2018. Cloud computing based bushfire prediction for cyberphysical emergency applications. Future Gener. Comput. Syst. 79 (P1), 354–363. http://dx.doi.org/10.1016/j.future.2017.02.009.

Haines, D., 1988. A lower atmospheric severity index for wildland fire. Natl. Weather Digest 23–27.

Hargrove, W.W., Forrest, M.H., Hessburg, P.F., 2006. Mapcurves: a quantitative method for comparing categorical maps. J. Geogr. Syst. 1–22.

Holmes Thomas, P., Calkin David, E., 2013. Econometric analysis of fire suppression production functions for large wildland fires. Int. J. Wildland Fire 22 (2), 246–255. http://dx.doi.org/10.1071/WF11098, URL: https://www.publish.csiro.au/paper/WF11098.

Jain, P., Coogan, S.C., Subramanian, S.G., Crowley, M., Taylor, S., Flannigan, M.D., 2020. A review of machine learning applications in wildfire science and management. Environ. Rev. 28 (4), 478–505. http://dx.doi.org/10.1139/er-2020-0019.

Kalabokidis, K., Athanasis, N., Vasilakos, C., Palaiologou, P., 2014. Porting of a wildfire risk and fire spread application into a cloud computing environment. Int. J. Geogr. Inf. Sci. 28 (3), 541–552.

K.C., U., Aryal, J., Garg, S., Hilton, J., 2021. Global sensitivity analysis for uncertainty quantification in fire spread models. Environ. Model. Softw. 143, 105110. http://dx.doi.org/10.1016/j.envsoft.2021.105110, URL: https://www.sciencedirect.com/science/article/pii/S1364815221001535.

Leong, S.H., Kranzlmüller, D., 2015. Towards a general definition of urgent computing. Procedia Comput. Sci. 51, 2337–2346, proc. of the International Conference on Computational Science, ICCS 2015.

Linn, R., Goodrick, S., Brambilla, S., Brown, M., Middleton, R., O'Brien, J., Hiers, J., 2020. QUIC-fire: A fast-running simulation tool for prescribed fire planning. Environ. Model. Softw. 125, 104616. http://dx.doi.org/10.1016/j.envsoft.2019.104616, URL: https://www.sciencedirect.com/science/article/pii/S1364815219307388.

Maranghides, A., Link, E., Mell, W., Hawks, S., Wilson, M., Brewer, W., Brown, C., Vihnaneck, B., Walton, W.D., 2021. A Case Study of the Camp Fire – Fire Progression Timeline. National Institute of Standards and Technology, URL: https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.2135.pdf. U.S. Department of Commerce.

Miller, C., Hilton, J., Sullivan, A., Prakash, M., 2015. SPARK – A bushfire spread prediction tool. In: Denzer, R., Argent, R.M., Schimak, G., Hřebíček, J. (Eds.), Environmental Software Systems. Infrastructures, Services and Applications. Springer International Publishing, Cham, pp. 262–271.

Monedero, S., Ramirez, J., Cardil, A., 2019. Predicting fire spread and behaviour on the fireline. Wildfire analyst pocket: A mobile app for wildland fire prediction. Ecol. Model. 392, 103–107. http://dx.doi.org/10.1016/j.ecolmodel.2018.11.016, URL: https://www.sciencedirect.com/science/article/pii/S0304380018304046.

NASA, 1999. MODIS - moderate resolution imaging spectroradiometer. https://modis.gsfc.nasa.gov/about/.

NASA/USGS, 1972. Landsat program. https://landsat.gsfc.nasa.gov/.

Oliveira, U., Soares-Filho, B., Rodrigues, H., Figueira, D., Gomes, L., Leles, W., Berlinck, C., Morelli, F., Bustamante, M., Ometto, J., Miranda, H., 2023. A near real-time web-system for predicting fire spread across the Cerrado biome. Sci. Rep. 13 (1), http://dx.doi.org/10.1038/s41598-023-30560-9.

Pereira, J., Mendes, J., Júnior, J.S.S., Viegas, C., Paulo, J.R., 2022. A review of genetic algorithm approaches for wildfire spread prediction calibration. Mathematics 10 (3), http://dx.doi.org/10.3390/math10030300, URL: https://www.mdpi.com/2227-7390/10/3/300.

Ramirez, J., Monedero, S., Buckley, D., 2011. New approaches in fire simulations analysis with wildfire analyst. In: 5th International Wildland Fore Conference.

Rothermel, R., 1972. A Mathematical Model for Predicting Fire Spread in Wildland Fuels. USDA Forest Service Research Paper INT, Intermountain Forest & Range Experiment Station, U.S. Dept. of Agriculture.

San-Miguel-Ayanz, J., Moreno, J.M., Camia, A., 2013. Analysis of large fires in European Mediterranean landscapes: Lessons learned and perspectives. Forest Ecol. Manag. 294, 11–22, The Mega-fire reality.

Thompson, M.P., Calkin, D.E., 2011. Uncertainty and risk in wildland fire management: A review. J. Environ. Manag. 92 (8), 1895–1909.

Tyndall, J., 2023. Taming wildfires in the context of climate change-policy highlights. Wildfire Policy Highlights. URL: https://www.oecd.org/climate-change/wildfires/policy-highlights-taming-wildfires-in-the-context-of-climate-change.pdf. Organisation for Economic Co-operation and Development.

Veronica Casartelli, J.M., 2023. Peer review programme for disaster risk management: Wildfire peer review assessment framework (Wildfire PRAF). Wildfire PRAF. URL: https://civil-protection-humanitarian-aid.ec.europa.eu/system/files/2023-06/Wildfire_PRAF_V2.pdf. Union Civil Protection Mechanism.

Walsh, W.E., Tesauro, G., Kephart, J.O., Das, R., 2004. Utility functions in autonomic systems. In: International Conference on Autonomic Computing, 2004. Proceedings. pp. 70–77. http://dx.doi.org/10.1109/ICAC.2004.1301349.

Yoo, S., Song, J., 2023. Rapid prediction of wildfire spread using ensemble Kalman filter and polyline simplification. Environ. Model. Softw. 160, 105610. http://dx.doi.org/10.1016/j.envsoft.2022.105610, URL: https://www.sciencedirect.com/science/article/pii/S1364815222003103.

Zacharakis, I., Tsihrintzis, V.A., 2023. Integrated wildfire danger models and factors: A review. Sci. Total Environ. 899, 165704. http://dx.doi.org/10.1016/j.scitotenv.2023.165704, URL: https://www.sciencedirect.com/science/article/pii/S0048969723043279.