Universitat Autònoma de Barcelona

Re-Engineering Forest Fire Spread Simulator to Consider Urgent and Green Computing

Carles Carrillo Jordan

Universitat Autònoma
de Barcelona

Escola d'Enginyeria

Computer Architecture & Operating Systems Department

Re-Engineering Forest Fire Spread Simulator to Consider Urgent and Green Computing

PhD thesis by Carles Carrillo Jordan, belonging to the research line of High Performance Computing for the degree of Philosophiae Doctor in Computer Science, by the Universitat Autònoma de Barcelona , under the supervision of Dr. Tomàs Margalef Burrull and Dr. Antonio Espinosa Morales.

Barcelona (Spain), March, 2022

Computer Architecture & Operating Systems

Re-Engineering Forest Fire Spread Simulator to Consider Urgent and Green Computing

Thesis submitted by **Carlos Carrillo Jordan** in fulfillment of the requirement for the PhD degree from Universitat Autònoma de Barcelona. This doctoral thesis belongs to the High Performance Computing doctoral program and presented to the Computer Architecture and Operating Systems Department at the Escola d'Enginyeria of Universitat Autònoma de Barcelona, under the supervision of Dr. Tomàs Margalef Burrull and Dr.Antonio Espinosa Morales.

Advisors signature

MARGALE F BURRULL TOMAS MANUEL -

ANTONIO MIGUEL ESPINOSA MORALES

Author signature

CARLOS CARRILL O JORDAN -

This research has been supported by MINECO-Spain under contract TIN2017-84553-C2-1-R and by the Catalan government under grant 2017-SGR-313.

Acknowledgements

Sin duda alguna, está es la parte que más me aterroriza de esta tesis. Me aterroriza parecer demasiado sensiblero o no poder expresar con palabras toda la gratitud que siento hacia las personas que me han acompañado durante esta aventura. Así que ahí va:

Esta tesis no se podría haber llevado a cabo sin mi familia. Padres, hermano, tíos, tías y primos. Aunque no entiendan muy bien lo que hago, siempre han sido un apoyo y un ejemplo de lucha y superación. Gracias por estar siempre allí.

No podría dejar de expresar mi más honda gratitud a mis dos directores, Tomàs Margalef y Toni Espinosa, sin su guía y apoyo esta tesis no pasaría de ser un conjunto de ideas sin mucha coherencia. También debo mencionar a Anna Cortés, sin ella esta experiencia nunca podría haber sido posible. Muchas gracias a todos por creer en mí y darme la oportunidad de poder realizar mi sueño. Gracias por vuestra paciencia y comprensión. Gracias por tener la puerta siempre abierta para solucionar dudas o inquietudes. Realmente me siento afortunado de haber compartido esta investigación con vosotros. Gracias a todos los miembros del *GFIRE Group*, pasados y presentes. Sin su trabajo y sus aportaciones nunca habría llegado tan lejos.

Gracias a Dani y Gemma, mis solucionadores de problemas, sin cuya ayuda esta tesis nunca habría visto la luz.

Gracias a Eduardo, Ania, Juan Carlos y todos los miembros del Departamento de Arquitectura de Computadores y Sistemas Operativos. Gracias por vuestro buen humor, vuestras bromas y el buen ambiente de trabajo. Con vosotros siempre me he sentido como uno más. Gracias por resolver mis dudas, por estúpida o pequeña que fuera.

A mis compañeros de viaje, Lidia, Jordi, Dani, Betzabet y Felipe. Sin vosotros este momento nunca habría sido posible. Gracias por vuestra ayuda y soporte, pero sobre todo gracias por todos los momentos que hemos compartido. Buenos y malos. Una vez me dijeron, para poder finalizar un doctorado, lo más importante es tener compañeros en los que apoyarse. Gracias a vosotros he tenido amigos en los que apoyarme.

Finalmente, quiero hacer una mención especial para mi mujer y mejor amiga. Ella que me ha soportado en los momentos más oscuros, con la que he celebrado esas pequeñas victorias que hacen de la vida una experiencia maravillosa. Muchas gracias por compartir este viaje conmigo, gracias por animarme a perseguir un sueño, gracias por apoyarme incondicionalmente y sobre todo hacerme el mejor regalo que alguien pude desear, mis dos pequeñas joyas, Biel y Marc, cuyas sonrisas endulzaban los momentos más amargos. Gracias por todo, no sé qué me deparará el futuro, pero a vuestro lado solo puede ser algo maravilloso.

Abstract

Forest fire spread simulators have been proven to be handy tools when fighting against wildfire disasters. Due to the necessity of achieving realistic predictions of the fire behaviour in near-real-time, the time spent performing one fire spread simulation must be limited. When providing forest fire spread predictions, there are two main considerations: the accuracy of the prediction and the computational time. In the context of forest fires, part of the forecast error comes from the uncertainty in the input data. To reduce the impact of this input-data uncertainty, different strategies have been developed. One of these strategies consists of introducing a new stage where the input parameters are adjusted according to the actual evolution of the fire. In order to optimize this adjusting stage, a Genetic Algorithm (GA) is used. This calibration strategy is computationally intensive and time-consuming. Considering the urgency in the forest fire spread prediction, it is necessary to maintain a balance between accuracy and time needed to calibrate the input parameters. During this thesis, we follow three different strategies to improve the forest fire spread prediction quality and reduce the execution time.

The first strategy consists of implementing the mixed-precision methodology into the forest fire spread simulator. Most scientific codes have over-engineered the numerical precision required to obtain reliable results. Therefore, there exists the possibility to get substantial speedups from using a more appropriate choice of precision. We propose to use a mixed-precision approach to accelerate the simulation of each individual without sacrificing the accuracy of the prediction. Because the GA is an iterative methodology, the more iterations we can do, the

better the solution we find. However, the time spent performing the forecasted fire behaviour must be served before the real fire evolution; this implies that the number of individuals per generation of the GA is determined by time response. If the time execution to simulate an individual is reduced, we can increase the number of generations and the number of individuals per generation, so the fire behaviour prediction quality will improve. Our work has concluded that using the mixed-precision approach can speed up the whole evolutionary prediction system.

The second strategy consists of applying a fine-grained parallelization to increase the accuracy of the Forest Fires Spread simulator without excessive increase the execution time. In order to effectively implement this fine grain parallelization, we exploit the computational capabilities of the GPUs (Graphics Processing Units). Most studies done up to now with GPUs use forest fire spread simulators based on cellular automata (CA) because they are easier to parallelize, but these kinds of simulators suffer from precision lack. An alternative approach to the CA propagation strategy is the Elliptical Wave Propagation (EWP) scheme. The EWP approach has been proven to provide more accurate results than CA, but it incurs higher execution times. Furthermore, the complexity of the EWP approach results in a challenging problem to parallelize. Our proposal consists of implementing parallelization on GPUs to reduce the execution time required to simulate the evolution of a forest fire, also allowing to improve the accuracy of the results. In order to properly analyze the speed up obtained when using the proposed parallel EWP scheme, we compare our GPU implementation against and OpenMP parallelization using real fire. The obtained results highlighted that the proposed parallel EWP scheme reduces the execution time spent in running a forest fire spread simulation and allows to deliver the results in higher resolution what would come of more accurate results when applying this parallel scheme in real scenarios.

The last strategy encourages the use of new platforms to help collect real-time data in the same area where the fire is taking place, which can profoundly reduce the input data uncertainty. However, if the site where the fire is taking place has low connectivity, the data cannot be shipped, so it is the paramount importance to dispose of a platform to perform the forest spread simulation *in situ*. The embedded system resulting would be more efficient to predict the near future behaviour of the fire near Real-time, close to where the fire is burning. The main objective consists of applying our EWP GPU implementation with a low consumption GPU to achieve the execution time requirements without losing accuracy in the simulations of the fire spread for being used in real scenarios. To accomplish this objective, we presented a quantitative analysis of the execution of a forest fire spread simulator in a embedded system with a low consumption GPU and compared its performance against a desktop GPU. Moreover, because embedded systems have different power configurations, the different energy modes are tested using a real wildfire. Results highlighted that the utilization of the embedded system allows performing the fire forecast *in situ*, with a high resolution in near-real-time. Thus, the results of this study can be extrapolated to another kind of computational model, taking into account their specific particularities.

Keywords

Forest Fire, Elliptical Wave Propagation, High Performance Computing, Mixed-Precision, Graphics Processing Units, Optimization, Embedded Systems

Resum

Els simuladors de propagació d'incendis forestals han demostrat ser unes eines realment útils en la lluita contra els incendis forestals. A causa de la necessitat d'aconseguir prediccions realistes de la propagació del foc en temps gairebé real, el temps per a simular la seva evolució és limitat. A l'hora de realitzar la predicció de la futura evolució de l'incendi forestal, hi ha dues consideracions principals: la precisió de la predicció i el temps de còmput. En el camp dels incendis forestals, gran part de l'error en la predicció és degut a la incertesa de les dades d'entrada. Per a reduir l'impacte d'aquesta incertesa, s'han desenvolupat diferents estratègies. Una d'aquestes estratègies consisteix a introduir una nova etapa intermèdia en la qual els paràmetres d'entrada del simulador s'ajusten en funció de l'evolució real de l'incendi. Per a optimitzar aquesta etapa d'ajust s'utilitza un Algorisme Genètic (AG). Aquesta estratègia de calibratge és molt intensiva des del punt de vista computacional i requereix molt temps. Tenint en compte la urgència en la predicció de la propagació d'incendis forestals, és necessari mantenir un equilibri entre la precisió i el temps necessari per a calibrar els paràmetres d'entrada. En aquesta tesi, seguim tres estratègies diferents per a millorar la qualitat de la predicció de la propagació d'incendis forestals i reduir el temps d'execució.

La primera estratègia consisteix a implementar la metodologia de precisió mixta al simulador de propagació d'incendis forestals. La majoria dels codis científics han exagerat la precisió numèrica necessària per a obtenir resultats fiables. Per tant, existeix la possibilitat d'obtenir augments de velocitat substancials si s'empra una elecció més adequada de la precisió d'algunes variables. Proposem usar un enfocament de

precisió mixta per a accelerar la simulació de cada individu sense sacrificar precisió en la predicció. Atès que el AG és una metodologia iterativa, quantes més iteracions puguem fer, millor serà la solució que trobem. No obstant això, la predicció de la propagació de l'incendi ha de ser obtinguda abans de l'evolució del foc real; això implica que el nombre d'individus per generació del AG està determinat pel temps de resposta. Si es redueix el temps d'execució per a simular un individu, podrem augmentar el nombre de generacions i el nombre d'individus per generació, per la qual cosa la qualitat de la predicció obtinguda millorarà. El nostre treball ha demostrat que l'ús de l'enfocament de precisió mixta pot accelerar tot el sistema de predicció evolutiva.

La segona estratègia consisteix a aplicar una paral·lelització de gra fi (fine-grain parallelization) per a incrementar la precisió del Simulador d'incendis forestals sense incrementar de forma excessiva el temps d'execució. Per a aplicar eficaçment aquesta paral·lelització de gra fi, s'aprofiten les capacitats computacionals de les GPU (Unitats de Processament Gràfic). La majoria dels estudis realitzats amb GPUs fins al moment utilitzen simuladors de propagació d'incendis forestals basats en Autòmats Cel·lulars (AC) perquè són més senzills de paralelizar, però aquest tipus de simuladors tenen una baixa precisió. Un enfocament alternatiu a l'estratègia de propagació CA és l'esquema basat en la Propagació d'Ona El·líptica (EWP). S'ha demostrat que l'enfocament EWP proporciona resultats més precisos que CA, però requereix temps d'execució majors. A més, la complexitat de l'enfocament EWP fa que la seva paral·lelització sigui un problema complex. La nostra proposta consisteix a implementar una paral·lelització en GPUs per a reduir el temps d'execució necessari per a simular l'evolució d'un incendi forestal que, al seu torn, permeti millorar la precisió dels resultats. Per a poder analitzar adequadament la millora del rendiment obtingut en utilitzar l'esquema EWP paral·lel proposat, comparem la nostra implementació en GPU amb una paral·lelització en OpenMP utilitzant un incendi real. Els resul-

tats obtinguts posen de manifest que l'esquema EWP paral·lel proposat redueix el temps d'execució emprat en l'execució de la simulació de la propagació d'un incendi forestal i permet obtenir els resultats amb major resolució, la qual cosa es tradueix en resultats més precisos en aplicar aquest esquema paral·lel en escenaris reals.

L'última estratègia fomenta l'ús de noves plataformes per a ajudar a mesurar els paràmetres de l'incendi en temps real en la mateixa zona on s'està produint, la qual cosa pot reduir considerablement la incertesa de les dades d'entrada. No obstant això, si el lloc on es produeix l'incendi té una baixa connectivitat, aquestes dades no poden ser enviades, per la qual cosa és de summa importància disposar d'una plataforma per a realitzar la simulació de propagació forestal *in situ*. El sistema integrat resultant seria més eficient per a predir el comportament futur del foc en temps real, prop d'on el foc està cremant. L'objectiu principal consisteix a aplicar la nostra implementació de la GPU EWP amb una GPU de baix consum per a aconseguir els requisits de temps d'execució sense perdre precisió en les simulacions de la propagació del foc per a ser utilitzades en escenaris reals. Per a aconseguir aquest objectiu, presentem una anàlisi quantitativa de l'execució d'un simulador de propagació d'incendis forestals en un sistema integrat amb una GPU de baix consum i comparem el seu rendiment amb el d'una GPU d'escriptori. A més, atès que els sistemes embeguts tenen diferents configuracions de potència, es proven les diferents maneres d'energia usant un incendi forestal real. Els resultats posen de manifest que la utilització del sistema embegut permet realitzar la previsió d'incendis *in situ*, amb una alta resolució en temps gairebé real. Els resultats obtinguts d'aquest estudi poden extrapolar-se a una altra mena de models computacionals, tenint en compte les seves particularitats específiques.

Paraules Clau

Incendi forestal, Propagació d'Onda El·líptica, Computació d'Altes Prestacions, Precisió Mixta, GPU, Optimització, Sistemes Integrats

Resumen

Los simuladores de propagación de incendios forestales han demostrado ser unas herramientas realmente útiles en la lucha contra los incendios forestales. Debido a la necesidad de conseguir predicciones realistas de la propagación del fuego en tiempo casi real, el tiempo para simular su evolución es limitado. A la hora de realizar la predicción de la futura evolución del incendio forestal, hay dos consideraciones principales: la precisión de la predicción y el tiempo de cómputo. En el campo de los incendios forestales, gran parte del error en la predicción es debido a la incertidumbre de los datos de entrada. Para reducir el impacto de esta incertidumbre, se han desarrollado diferentes estrategias. Una de estas estrategias consiste en introducir una nueva etapa intermedia en la que los parámetros de entrada del simulador se ajustan en función de la evolución real del incendio. Para optimizar esta etapa de ajuste se utiliza un Algoritmo Genético (AG). Esta estrategia de calibración es muy intensiva desde el punto de vista computacional y requiere mucho tiempo. Teniendo en cuenta la urgencia en la predicción de la propagación de incendios forestales, es necesario mantener un equilibrio entre la precisión y el tiempo necesario para calibrar los parámetros de entrada. En esta tesis, seguimos tres estrategias diferentes para mejorar la calidad de la predicción de la propagación de incendios forestales y reducir el tiempo de ejecución.

La primera estrategia consiste en implementar la metodología de precisión mixta al simulador de propagación de incendios forestales. La mayoría de los códigos científicos han exagerado la precisión numérica necesaria para obtener resultados fiables. Por lo tanto, existe la posibilidad de obtener aumentos de velocidad sustanciales si se emplea

una elección más adecuada de la precisión de algunas variables. Proponemos usar un enfoque de precisión mixta para acelerar la simulación de cada individuo sin sacrificar precisión en la predicción. Dado que el AG es una metodología iterativa, cuantas más iteraciones podamos hacer, mejor será la solución que encontremos. Sin embargo, la predicción de la propagación del incendio tiene que ser obtenida antes de la evolución del fuego real; esto implica que el número de individuos por generación del AG está determinado por el tiempo de respuesta. Si se reduce el tiempo de ejecución para simular un individuo, podremos aumentar el número de generaciones y el número de individuos por generación, por lo que la calidad de la predicción obtenida mejorará. Nuestro trabajo ha concluido que el uso del enfoque de precisión mixta puede acelerar todo el sistema de predicción evolutiva.

La segunda estrategia consiste en aplicar una paralelización de grano fino (fine-grain parallelization) para aumentar la precisión del simulador de propagación de incendios forestales sin aumentar excesivamente el tiempo de ejecución. Para aplicar eficazmente esta paralelización de grano fino, se aprovechan las capacidades computacionales de las GPU (Unidades de Procesamiento Gráfico). La mayoría de los estudios realizados hasta el momento con GPUs utilizan simuladores de propagación de incendios forestales basados en Autómatas Celulares (AC) porque son más sencillos de paralelizar, pero este tipo de simuladores adolecen de una baja precisión. Un enfoque alternativo a la estrategia de propagación CA es el esquema basado en la Propagación de Onda Elíptica (EWP). Se ha demostrado que el enfoque EWP proporciona resultados más precisos que CA, pero requiere tiempos de ejecución mayores. Además, la complejidad del enfoque EWP hace que su paralelización sea un problema complejo. Nuestra propuesta consiste en implementar una paralelización en GPUs para reducir el tiempo de ejecución necesario para simular la evolución de un incendio forestal que, a su vez, permita mejorar la precisión de los resultados. Para poder analizar adecuadamente la mejora del rendimiento obtenido

al utilizar el esquema EWP paralelo propuesto, comparamos nuestra implementación en la GPU con una paralelización en OpenMP utilizando un incendio real. Los resultados obtenidos ponen de manifiesto que el esquema EWP paralelo propuesto reduce el tiempo de ejecución empleado en la ejecución de la simulación de la propagación de un incendio forestal y permite obtener los resultados con mayor resolución, lo que se traduce en resultados más precisos al aplicar este esquema paralelo en escenarios reales.

La última estrategia fomenta el uso de nuevas plataformas para ayudar a recoger parámetros del incendio en tiempo real en la misma zona donde se está produciendo, lo que puede reducir considerablemente la incertidumbre de los datos de entrada. Sin embargo, si el lugar donde se produce el incendio tiene una baja conectividad, estos datos no pueden ser enviados, por lo que es de suma importancia disponer de una plataforma para realizar la simulación de propagación forestal *in situ*. El sistema integrado resultante sería más eficiente para predecir el comportamiento futuro del fuego en tiempo real, cerca de donde el fuego está ardiendo. El objetivo principal consiste en aplicar nuestra implementación de la GPU EWP con una GPU de bajo consumo para conseguir los requisitos de tiempo de ejecución sin perder precisión en las simulaciones de la propagación del fuego para ser utilizadas en escenarios reales. Para lograr este objetivo, presentamos un análisis cuantitativo de la ejecución de un simulador de propagación de incendios forestales en un sistema integrado con una GPU de bajo consumo y comparamos su rendimiento con el de una GPU de escritorio. Además, dado que los sistemas embebidos tienen diferentes configuraciones de potencia, se prueban los distintos modos de energía usando un incendio forestal real. Los resultados ponen de manifiesto que la utilización del sistema embebido permite realizar la previsión de incendios *in situ*, con una alta resolución en tiempo casi real. Los resultados obtenidos de este estudio pueden extrapolarse a otro tipo de modelos computa-

cionales, teniendo en cuenta sus particularidades específicas.

Palabras Clave

Incendio forestal, Propagación de Onda Elíptica, Computación de Altas Prestaciones, Precisión Mixta, GPU, Optimización, Sistemas Integrados

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

1.1 Forest Fires

Climate change is arguably the most significant challenge for modern society and one of the utmost international concerns emerging as a critical problem to humanity. The rise in average global temperatures has led to higher spring and summer temperatures and, significantly, an earlier onset of forest fire season. These warmer climatic conditions result in a substantial increase in the frequency and intensity of forest fires around the world. Across continents, uncontrollable and destructive forest fires are becoming an expected part of the annual calendar. The 2020 wildfire season was a record-setting one for the United States of America, where $52,113$ forest fires burned 8.8 million of acres or 3.6 millions of hectares, [44]. In Australia, in 2019-2020, forest fire season scorched 18.21 million hectares, and 27 people lost their lives to the wildfires.

Figure 1.1 shows the number of fires between the 1st of June and the 1st of July of 2021. As we can observe, forest fires are a global problem that affects all the countries of the globe. Besides, forest fires are responsible for far greater air pollution than industrial emissions and produce a combination of particles, carbon monoxide, and other pollutants that can be hazardous to the health of all life on the planet. Approximately $6,735$ megatons of CO2 have been released into the

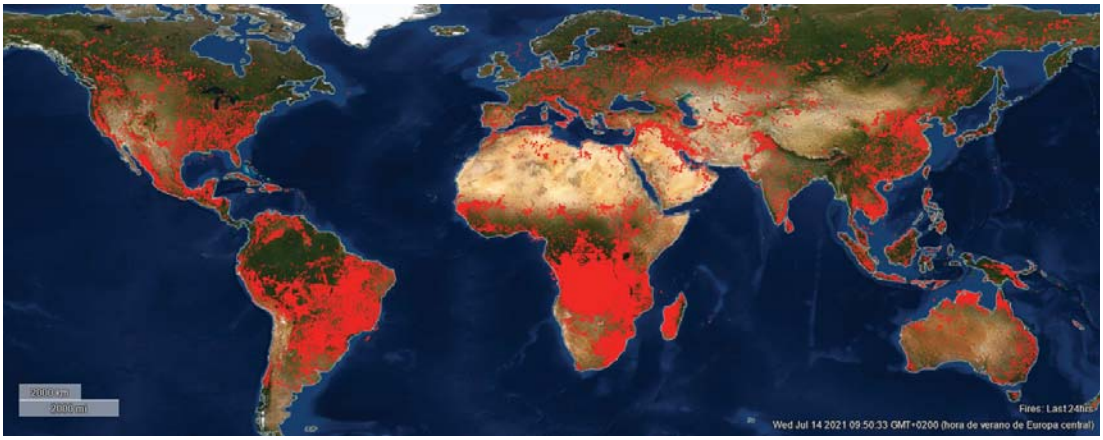


Figure 1.1: Number of Forest Fires around the world between 1st of June and 1st of July of 2021, [31]

atmosphere from wildfires between 1 January and 30 November 2019. Many of them have made headlines across the globe, including the Amazon fires, Indonesian fires, the Arctic wildfires, and the Australian bushfires [15].

The need to anticipate potential fire behaviour and its resultant impacts has led to the development of the field of fire modelling. In the last decades, several physical and mathematical models have been developed to provide reliable forecasting of fire behaviour, trying to optimize firefighting resource management during outbreaks. Simulators implementing forest fire spread models require several input parameters to describe the characteristics of the environment where the fire is taking place in order to evaluate its future propagation. Furthermore, there are severe difficulties in gathering precise values of certain parameters at the right places where the fire is taking place, often because the hazard itself distorts the measurements.

Currently, exists several fire propagation simulators, based on some physical or mathematical models, whose main objective is to predict the fire evolution, [48], [55]. In order to evaluate future fire evolution, these simulators need specific input data to define the characteristics of the environment where the fire is taking place. Describing the forest fire scenario is a complex task because it involves data coming from very different sources such as satellites, weather stations, databases,

and it also requires data computed by complementary models. The information required to define a forest fire scenario accurately is the following:

- *Topographic data*: Topographic data provides information about the elevation, slope, and aspect of the terrain. The elevation map is a regularly spaced grid of elevation points that discretizes a continuous surface, taking into account measurements at specific terrain points. The slope is the angle of incline on a hillside. Finally, the aspect corresponds to the direction in which a slope faces and relates to the degree of solar exposure.

- *Vegetation*: The vegetation map, or fuel map, displays the vegetation heterogeneity of the area. Each vegetation or fuel type has its own parameters, such as moisture content, flammability, fuel-bed load and density, and heat content. These kinds of maps consider a limited group of standard fuels representing the great diversity of possible fuels and parameters, [4]. Another type of map regarding vegetation is called canopy cover map. This map shows the percentage of tree crowns present in a terrain division.

- *Meteorological data*: The meteorological variables are a primary factor in real-time forest fire spread forecast. These parameters have a direct impact on the fire spread direction and intensity of the wildfire. Only those variables, which have more relevance in fire propagation, are considered. Temperature and humidity influence fire intensity, but the most critical variable factor to consider is wind.

- *Fire perimeter*: The perimeter of the fire can be obtained from aeronautical or satellite images; however, in most cases is provided by firefighters working in the field.

1.2 Forest Fire Spread Simulators

In the last decades, several fire spread models have been implemented in different computing simulators with the purpose of helping the management and decision-making procedures. The forest fire spread simulators may be split into three big

categories according to the method used to propagate the front of the fire [50], [55].

- **Cellular Automata**: These systems approach fire propagation through some set of rules across a uniform grid. While fast and straightforward to implement, they lack precision compared to other approaches. One of these fire spread simulators are *Behave-Plus* [5].

- **Elliptical wave propagation**: This kind of approach follows the idea of elliptical wave propagation more strictly; that is, after some time step, the fire shape can be updated by generating ellipses along with the previous fire shape and determining the new outline. Simulation is done in a continuous space. FARSITE is one of the simulators that use this approach, [29].

- **Level Set Method**: In this method, the burned region is defined as a level set function $\psi = \psi(\overrightarrow{x}, t)$ whose values are:

$$\psi(\overrightarrow{x}, t) = \begin{cases} \psi(\overrightarrow{x}, t) \leq 0 & \text{Burned}n \text{ Area} \\ \psi(\overrightarrow{x}, t) = 0 & \text{Fire}n \text{ Front} \\ \psi(\overrightarrow{x}, t) \geq 0 & \text{Unburned}n \text{ Area} \end{cases} \tag{1.1}$$

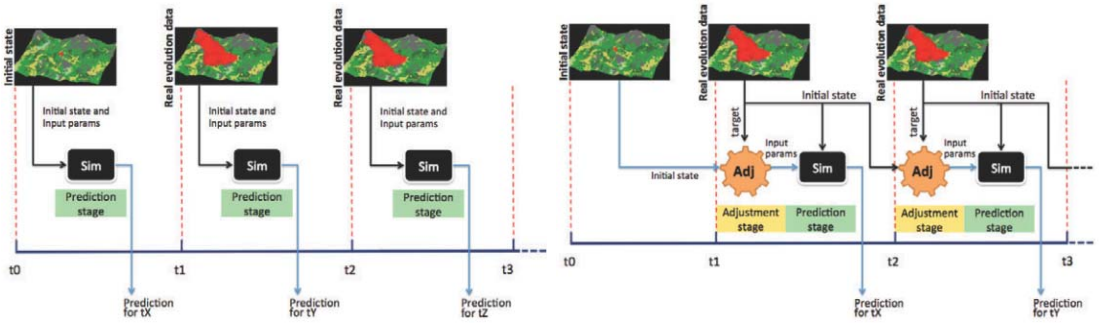
  and it evolves through the partial differential

$$\frac{\partial \psi}{\partial t} = R \cdot \nabla \psi(\overrightarrow{x}, t) \tag{1.2}$$

  where R is the rate of spread [28]. Equation 1.2 can be solved numerically given initial and boundary values for ψ, as well as a model, to compute the rate of speed. An advantage of this method is that the behaviour of fire fronts arises naturally from the underlying mathematics and thus does not require the special handling that is needed when an Elliptical wave propagation method is used. WRF-SFIRE [42] and Spark [53] use this approach.

1.3 Two-stage methodology

The traditional way of predicting forest fire behaviour takes a particular state of the fire front as input, as well as the input parameters that are given for this time instant, as seen in Figure 1.2(a). The simulator then returns the fire evolution prediction for a succeeding time instant. This classic prediction scheme produces forecasts that differ to a greater or lesser extent from the real fire evolution.



(a) Classical Prediction Strategy.

(b) Two-Stage Dynamic Data Driven Forest Fire Prediction Strategy.

Figure 1.2: Prediction Methods

The quality results of these simulators depends not only on the propagation equations describing the behaviour of the fire but also on the input data required to initialize the model. Unfortunately, it is impossible to obtain the data that populates these models without error. This data uncertainty is due to the difficulty in gathering precise values at the right places where the catastrophe is taking place or because the hazard itself distorts the measurements. So, in many cases, the unique alternative consists of working with interpolated, outdated, or even absolutely unknown values. Obviously, this fact results in a lack of accuracy and quality on the provided predictions. In order to minimize the uncertainty in all this input data and improve the accuracy of the delivered predictions, a *Two-Stage Dynamic Data Driven Forest Fire Prediction Methodology* was developed [26]. The dynamic data driven approaches for forest fire spread prediction seek to drive the model forecast using control variables. That is, to enhance basic forest fire spread simulations with the knowledge obtained from a calibration/adjustment stage [11] [21]. In this Calibration stage, the obtained simulations are evaluated

in order to weigh them according to some fitness/error function, which determines the similitude of a given simulation to the observed real fire propagation. Once the best scored simulation is selected, the configuration of the control variables associated with that winner is applied for prediction purposes in the near future, see Figure 1.2(b). In order to optimize the adjusting stage, a Genetic Algorithm (GA) is used.

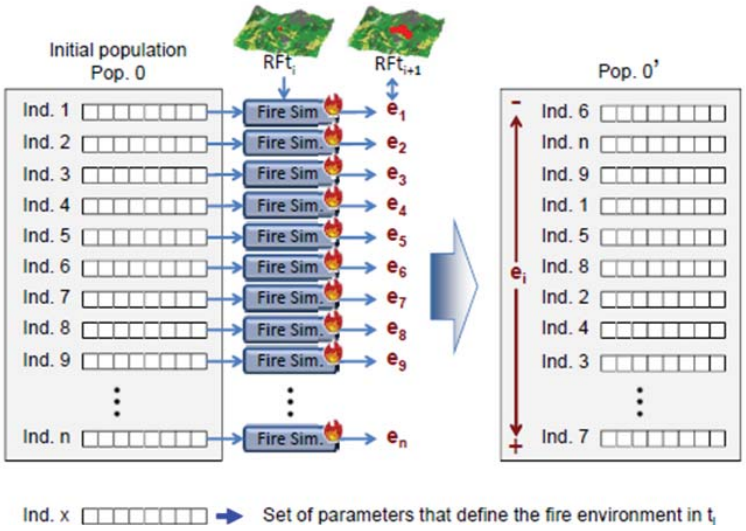
1.3.1 Calibration Stage: Genetic Algorithm (GA)

The most important phase within the Two-stage prediction strategy is the calibration process. For the particular case of forest fire spread prediction, Genetic Algorithms have been demonstrated to provide good adjustment results [22]. Genetic Algorithms (GA) are a heuristic optimisation method that can be used to find exact or near-optimal solutions to certain search problems. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

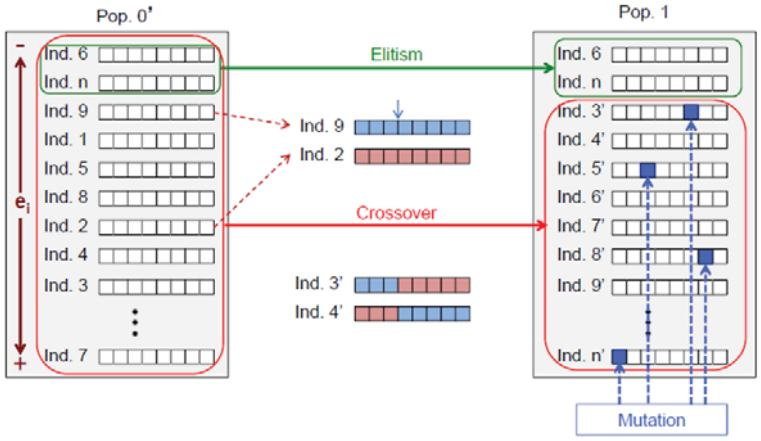
The GA begins with a set of p individuals $x = [x_1, x_2, ..., x_p]$ which is called a Population. Each individual of the population is characterized by a fire simulator input parameter set (scenario) made up of d parameter values, for example, humidity, temperature, cloud cover, wind speed, and wind direction, [57]. GA will evolve through several iterations to find better input parameters that best reproduce the real fire propagation. Each scenario is used to simulate the behaviour of the fire during one time interval, $t_i - t_{i+1}$, see Figure 1.3(a). Each individual is simulated, and the resulting forest fire spread is compared to the real observed fire spread to compute each individual's error. Then, according to the quality of the simulations evaluated in the Calibration stage, the individuals are ranked, and the next generation is obtained applying the genetic operations over the individuals of the population, see Figure 1.3(b). These operations are:

- *Elitism*: With this operation, the best j individuals are used directly in the next generation.
- *Crossover*: The crossover consists of the exchange of the characteristics of

two individuals by swapping parts. The crossover operation is in charge of mixing good blocks that are into different parents and who potentially can give the children a good score.



(a) Genetic Algorithm applied to forest fire simulations.



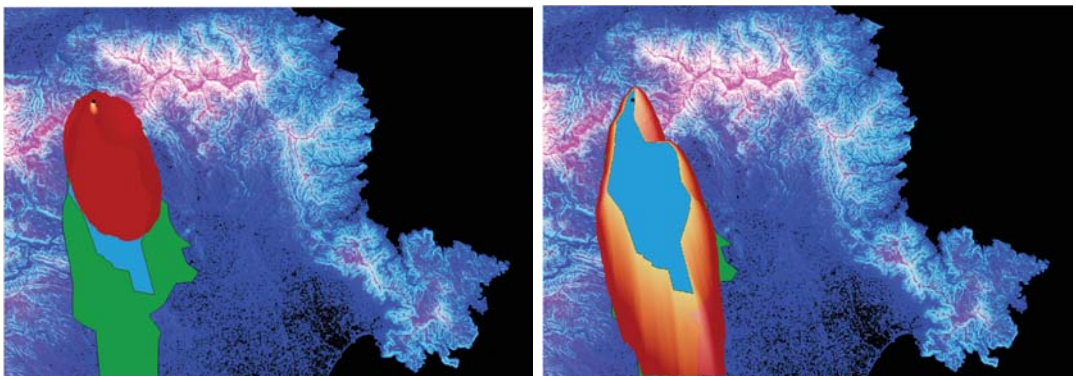
(b) Functionality of the Genetic Algorithm to calibrate the input variables.

Figure 1.3: Genetic Algoritm scheme.

- *Mutation*: The mutation involves varying a characteristic of the child in order to obtain individuals with different properties to their parents. The

mutation is carried out by each of the parts of the new individuals: Some genes are selected at random, and their value is changed randomly. This makes it possible to evaluate individuals with very different characteristics.

The Genetic Algorithm is repeated over n generations obtaining an improved population. The individual that provides the best adjustment among the simulated and real propagation after all iterations is introduced in the simulator with the real front at time t_{i+1} to provide a prediction at time t_{i+2}. The adjustment quality depends on the number of individuals that constitute the population and the number of iterations. The results highlight that the utilisation of the *Two-Stage Dynamic Data Driven Forest Fire Prediction Strategy* with a GA improve significantly the reliability of the prediction of the Forest Fires behaviour, see Figure 1.4. Nevertheless, there is a penalty in computing time as more iterations are needed to find a good individual. Nevertheless, simulation results are helpful in a prediction context. However, any evolution prediction of an ongoing hazard must be delivered as fast as possible, not to be outdated.



(a) Forest Fire Spread prediction when the Classical Prediction Strategy is used.

(b) Forest Fire Spread prediction when the Two-Stage Dynamic Data Driven Forest Fire Prediction Strategy is used.

Figure 1.4: Forest Fire Spread Prediction obtained with the Classic and the Two-Stage Strategies. The black point is the Ignition point of the Wildfire. The blue color represents the perimeter used in the Calibration stage, and the green perimeter represents the final burned area that we want to predict.

This dynamic data-driven forest fire spread prediction system has been designed to be simulator independent, so any forest fire simulator could easily be included

in a plug&play fashion. In particular, the forest fire spread simulator used in this work has been FARSITE ([29]), which is introduced in section 2.

1.3.2 Error Simulation Calculation

The evaluation of the quality of the simulations is a crucial point in the Genetic Algorithm because the measured error drives the ranking of the individuals. In previous studies, we have demonstrated that the prediction of the forest fire behaviour depends directly on the error function utilized to rang the individuals of the GA, [16], [17] and [18]. To compute the error of each individual, the landscape where the fires are taking place is represented as a grid of cells, and each cell will have assigned a state according to its belonging to either a real burnt area or to a simulated burnt area. There are four possible states for a given cell: cells that were burnt in both fires, the real and the simulated fire, *(Hits)*, cells burnt in the simulated fire but not in the reality *(False Alarms)*, cells burnt in reality but not in the simulated fire *(Misses)* and cells that were not burnt in any case *(Correct negatives)* [27] [32]. These four possibilities are used to construct a 2×2 contingency table, as shown in Figure 1.5. A perfect simulation system would have data only on the main diagonal.

		Observed: yes	Observed: no
Predicted	yes	Hits	False alarms
	no	Misses	Correct negatives

Figure 1.5: Standard structure of a contingency table.

In the particular problem of forest fire spread simulation, the correct negatives are ignored since the area of the map to be simulated may vary independently of the fire perimeter, so they may distort the measurement of the error. To facilitate data processing, the elements of the contingency table are expressed in the context of the difference between sets. The description of the conversion of these metrics is shown in Table 1.1.

Table 1.1: Elements of the contingency table expressed in the context of difference between sets.

RealCell	$Hits + Misses$	Cells burnt in the real fire
SimCell	$Hits + FA$	Cells burnt in the simulated fire
UCell	$Hits + Misses + FA$	Union of cells burnt in both fires
ICell	$Hits$	Cells burnt in real and simulated fires

There are several metrics that compare real and simulated values, and each one weights the events involved differently, depending on the nature of the problem [10]. During this research, the difference between the real and the simulated fire is computed using Equation 1.3.

$$\in= \frac{Misses + FA}{Hits + Misses} \tag{1.3}$$

The same error function can be expressed in terms of the difference between sets where the initial fire is considered a point and, therefore, it can be removed from the equation. The obtained formula is also shown in Equation 1.4.

$$\in= \frac{UCell - ICell}{RealCell} \tag{1.4}$$

The resulting error is used to rank the individuals in a list, from minor error to higher error. Then, a new generation of individuals is created by applying the aforementioned genetic operations.

1.4 Objectives

In the area of emergency management, the concepts of urgency and accuracy are closely related. Fire fighting decisions depend on reliable predictions that must allow time for planning and applying actions in the field. For that reason, during this thesis, we focus on three different objectives:

1. The *Two-Stage Dynamic Data Driven Forest Fire Prediction Methodology* is

an expensive process in terms of execution time and computing power. For that reason, the first objective is to reduce the execution time of the GA in the calibration stage. To accomplish this objective, we implement the *Mixed-Precision Methodology*.

2. The prediction of the simulation has to be as accurate as possible. For that reason, the second objective of our research is to increase the accuracy of the Forest Fires Spread simulator without excessive execution time penalties. To achieve this goal, we implement a *Fine Grained Parallelization*.

3. The last strategy has the objective to reduce the uncertainty of the input parameters and, at the same time, provides a powerful tool to the firefighters. This strategy consists of performing the simulation/prediction of the forest fire spread simulation near real-time and near the fire's location. To reach this strategy useful, evaluate the benefits of applying an *Edge Computing* solution.

Chapter 2

Fire Area Simulator (FARSITE)

To carry out this study, we focus on the *Fire Area Simulator* (FARSITE) forest fire simulator [29], which has extensively been tested on real fires and produces successful results. FARSITE has been used for several years for forest fire evolution simulation, and it has become widely used for operational prediction of fire growth of active wildfires as well as for modelling in support of planning for potential fires.

FARSITE is a Forest Fire spread simulator based on the Propagation of Elliptical Wave (EPW), see Section 1.2. This kind of simulator has been proven to provide more accurate results than the Forest Fire Simulators based on the Cellular Automata and faster than the Forest Fires simulators based on the Level Set Method. The main problem with the Forest Fire Spread simulators based on the EWP is that they cannot distinguish the burned areas from the unburned. The simulation process (see Figure 2.1) shows the nested logic of the model structure. The total length of the simulation is broken down into time steps. In each time step, the fire evolution in FARSITE can be divided into two main processes: the *Fire Front Propagation*, and the *Fire Front Reconstruction*.

Figure 2.2 displays the impact of the execution time distribution of FARSITE. Previous studies found that for fire with large propagation, the point expansion represents only around of the 7% of the total execution time, while the fire front reconstruction is around the 58%, [6]. In the following sections, both algorithms

```
For each timestep (date and time specified)
{   FIRE FRONT PROPAGATION
        For each point (X, Y)
        {       Get the fire environment (fuels, weather, topography)
                Calculate fuel moistures from initial conditions
                Calculate vertex orientation angle
                Surface fire Calculations
                {       Compute forward equilibrium spread rate
                        Vector wind and slope
                        Compute elliptical dimensions using resultant wind-slope vector
                        Compute spread rate R_t by accelerating fire over timestep
                        Compute average spread rate of fire over timestep
                        Compute spread differentials
                        Slope transformation
                }
        }
    FIRE FRONT RECONSTRUCTION
        For each point (X_i, Y_i) and point (X_i+1, Y_i+1)
        {       For each point (X_j, Y_j) and point (X_j+1, Y_j+1)
                {
                        Correct crossovers
                }
        }
        Compute area and perimeter
}
```

Figure 2.1: Fire growth process control used in FARSITE, [30].

are explained in deep detail.

2.1 Fire Front Propagation

As we said, FARSITE is based on the *Elliptical Wave Propagation*. In this approach, the perimeter of the fire is divided into a series of points, [40]. To obtain the evolution of the fire front, an ellipse is generated for each point. The local characteristics at each point determine the shape of the ellipses. In this way, the new perimeter is obtained by joining the obtained ellipses (see Figure 2.3).

To compute the propagation of each point perimeter, the model uses the orthogonal spread rate differentials ($m \cdot min^{-1}$):

$$X_t = \frac{a^2 \cdot \cos\theta(x_s \sin\theta + y_s \cos\theta) - b^2 \cdot \sin\theta(x_s \cos\theta - y_s \sin\theta)}{(b^2(x_s \cos\theta + y_s \sin\theta)^2 - a^2(x_s \sin\theta - y_s \cos\theta)^2)^{1/2}} + c\sin\theta \qquad (2.1)$$

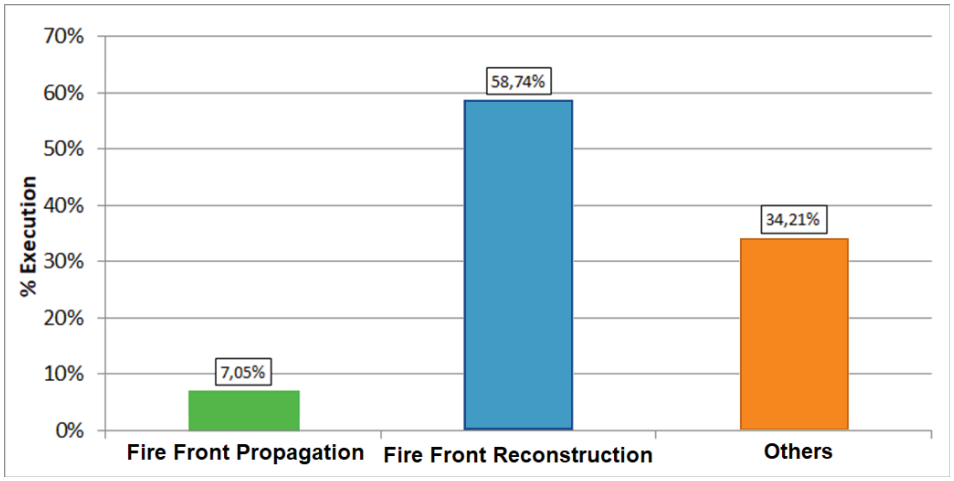


Figure 2.2: Percentage of the execution time invested into perform a forest fire spread simulation.

$$Y_t = \frac{-a^2 \cdot \sin\theta(x_s \sin\theta + y_s \cos\theta) - b^2 \cdot \cos\theta(x_s \cos\theta - y_s \sin\theta)}{(b^2(x_s \cos\theta + y_s \sin\theta)^2 - a^2(x_s \sin\theta - y_s \cos\theta)^2)^{1/2}} + c\cos\theta \tag{2.2}$$

where x_s and y_s are the orientation of the point on the fire front in terms of component differential, in meters. θ is the direction of the maximum spread rate, in radians, and a, b and c determine the shape of the ellipse from the local conditions of that point, see Figure 2.4.

The angle differentials x_s and y_s determine the direction normal to the fire front for point (x_i, y_i) in a plane parallel to the ground surface. They are transformed from their original horizontal values by adding or subtracting a slope correction D_i (m) depending on the aspect ω_i (radians) of the i^{th} point:

$$x_s = (x_{i-1} - x_{i+1}) \pm D_i \sin\omega_i \tag{2.3}$$

$$y_s = (y_{i-1} - y_{i+1}) \pm D_i \cos\omega_i \tag{2.4}$$

where

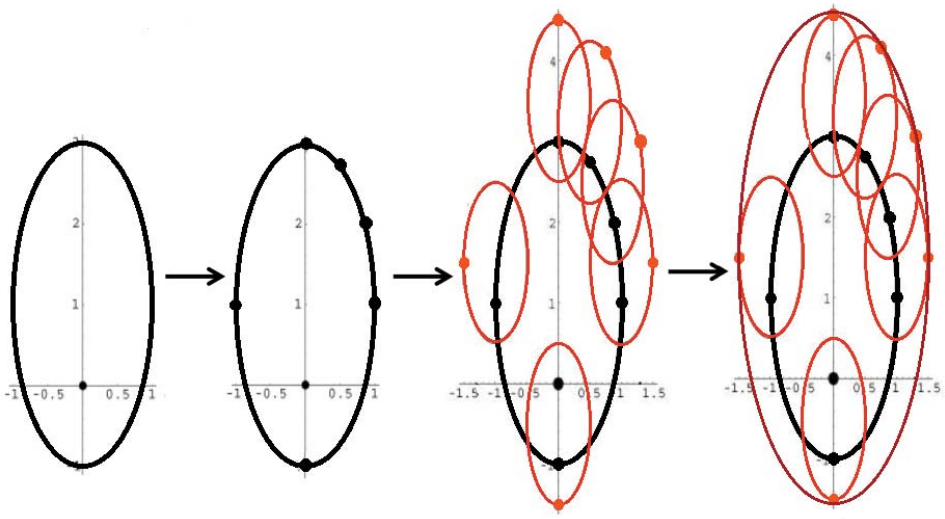


Figure 2.3: Elliptical wave propagation from t_1 to t_2.

$$D_i = \left[(x_{i-1} - x_{i+1})^2 + (y_{i-1} - y_{i+1})^2\right]^{\frac{1}{2}} \cdot \cos\delta_i(1 - \cos\phi_i) \tag{2.5}$$

ϕ_i is the local slope, ω_i is the aspect, and δ_i represents the difference between the aspect and the orientation of the perimeter segment.

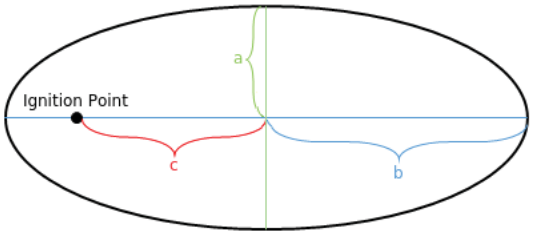


Figure 2.4: Dimensions of elliptical wavelets used in computing fire growth with equations 2.1 and 2.2. Dimension a corresponds to 1/2 the minor axis (lateral from the center), b identifies 1/2 the major axis (forward from the center), and c is the distance forward of the ignition point to the center.

The parameters of the ellipses axes can be computed in the following way:

$$a = 0.5 \cdot \frac{R + R/HB}{LB} \tag{2.6}$$

$$b = \frac{R + R/HB}{2} \tag{2.7}$$

$$c = b - \frac{R}{HB} \tag{2.8}$$

where R symbolizes the rate of spread, LB and HB are the lengths to breath ratio and the head to back ratio respectively expressed by equations 2.9 and 2.10:

$$LB = 0.936 \cdot e^{(0.2566 \cdot U)} + 0.461 \cdot e^{(-0.1548 \cdot U)} - 0.397 \tag{2.9}$$

$$HB = \frac{LB + (LB^2 - 1)^{1/2}}{LB - (LB^2 - 1)^{1/2}} \tag{2.10}$$

U is the mid flame wind speed ($m \cdot s^{-1}$). In this model, the propagation of each point of the perimeter is done in a continuous space and independently from other points. The rate of spread, which determines the dimensions of each ellipse, is calculated using Rothermel's fire spread model.

2.1.1 Rothermel's Fire Spread Model

Fire modelling is used to calculate fire behaviour characteristics and subsequent fire effects. Rothermel's semi-empirical forest fire spread model [51] is the foundation of most forest fire simulators. The fire spread model is based on the conservation of energy principle to a unit volume of fuel ahead of a steadily advancing fire in a homogeneous fuel bed. The fuel-reaction zone is viewed as fixed, and the unit volume moves at a constant depth toward the interface. The unit volume ignites at the interface. The rate of spread is then a ratio between the heat flux received from the source and the heat required for ignition by the potential fuel. Rothermel's model is formulated in the following way:

$$R = \frac{I_R \xi \cdot (\overrightarrow{n} + \overrightarrow{\phi}_w + \overrightarrow{\phi}_s)}{\rho_b \epsilon Q_{ig}} \tag{2.11}$$

where R represents the Rate of Spread (ROS) in a particular point ($\frac{M}{s}$), I_R is the reaction intensity, ξ is the propagation flux ratio, ρ_b is the ovendry bulk density, ϵ is the effective heating number, Q_{ig} is the heat of pre-ignition, $\overrightarrow{n}$ is the normal direction to the fire perimeter on that particular point, $\overrightarrow{\phi}_w$ is the wind factor and $\overrightarrow{\phi}_s$ the slope factor. Except $\overrightarrow{\phi}_w$ and $\overrightarrow{\phi}_s$, all the remaining factors only depend on the particular vegetation of the cell. Equation 2.11 can be rewritten as shown below:

$$R = R_0 \cdot (\overrightarrow{n} + \overrightarrow{\phi}_w + \overrightarrow{\phi}_s) \tag{2.12}$$

where R_0 represents the rate of spread in a particular point with no wind and no slope.

Slope factor $\overrightarrow{\phi}_s$ is based on an evaluation of experimental data, and can be formulated in the following way:

$$\overrightarrow{\phi}_s = 5.275 \dot{\beta}^{-0.3} \cdot \tan \phi^2 \tag{2.13}$$

where β is the packing ratio of the fuel bed, which is the ratio of the fuel oven-dry bulk density(ρ_b) and the fuel oven-dry particle density (ρ_p), ϕ represents the slope of the terrain in radians.

Wind and slope change the no-wind no-slope propagating flux ($I_R \xi$) by exposing the potential fuel to additional convective and radiant heat. The vertical flux is more significant during wind-driven and upslope fires because it was thought that flame tilt over the potential fuel would increase radiation and, more significantly, cause direct flame contact and increased convective heat transfer to the potential fuel.

Wind factor $\overrightarrow{\phi}_w$ was developed from wind tunnel data and wildfire data, and

is expressed as follows:

$$\overrightarrow{\phi}_w = C \cdot (3.281 \cdot U)^B \left(\frac{\beta}{\beta_{op}}\right)^{-E} \tag{2.14}$$

where U is the middle flame wind speed. The others factors depends on the fuel bed:

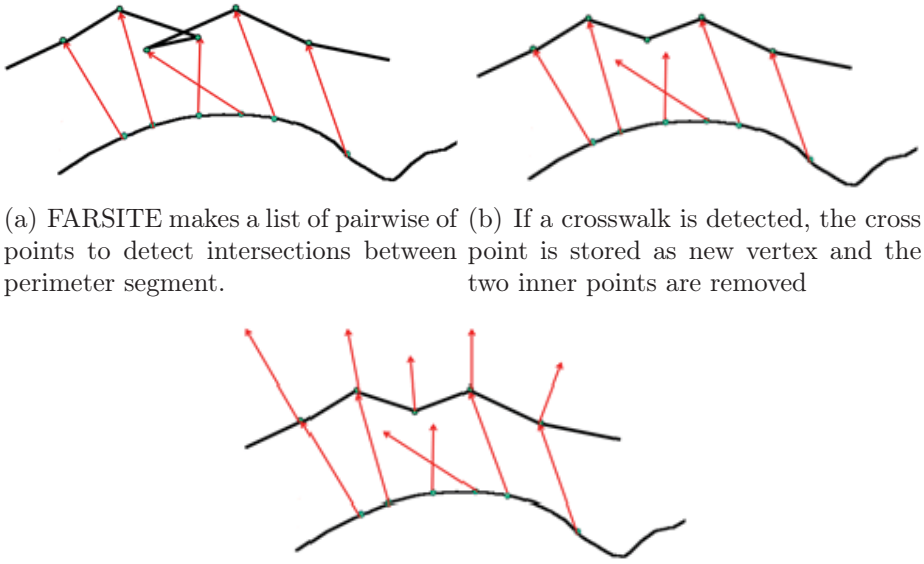
$$\beta_{op} = 3.348 \cdot \sigma^{-0.8189} \tag{2.15}$$
$$C = 7.47exp(-0.133 \cdot \sigma^{0.55}) \tag{2.16}$$
$$B = 0.02526 \cdot \sigma^{0.54} \tag{2.17}$$
$$E = 0.715exp(-3.59 \cdot 10^{-4} \cdot \sigma) \tag{2.18}$$

where σ is the surface-area-to-volume ratio ($m^{-1)}$). The higher its value, the faster a particle responds to changes in environmental conditions, such as temperature or moisture. Higher values are also correlated to shorter fuel ignition times and faster fire spread rates.

The precision of forest fire spread simulators based on the Elliptical Wave Propagation is directly related to the number of points of the perimeter of the fire. The higher the number of points, the more sensitive to small-scale variations. The execution time depends on the number of points to expand; hence, high-resolution simulations have long execution times. Hence, in this two-step methodology, very high-resolution simulations may be of limited use due to their extended execution time. However, the propagation of each point is entirely independent of the propagation of other points. For that reason, this algorithm is the best starting point for apply parallelization techniques.

2.2 Fire Front Reconstruction

One of the main drawbacks of forest fire simulators based on the EWP is that they do not intrinsically distinguish burned from unburned areas. If allowed to continue without detecting burned terrain, the fire front will form artificial complex loops and knots. These intersections of the fire front must be removed to preserve the meaningful portions of the fire front.



(a) FARSITE makes a list of pairwise of points to detect intersections between perimeter segment.

(b) If a crosswalk is detected, the cross point is stored as new vertex and the two inner points are removed

(c) The new point is used to perform the fire perimeter expansion in the next time step .

Figure 2.5: Fire front reconstruction algorithm.

The algorithm used in FARSITE to reconstruct the perimeter of the fire requires three preliminary steps, [30]:

1. The vertices of the fire perimeter must be ordered beginning from a vertex on the outside edge of the fire polygon; this is guaranteed by employing one of the polygon vertices that defines the farthest extent of the polygon along a given horizontal axis.

2. A list of pairwise points comparisons is made to identify intersections between

each perimeter segment and another perimeter segment of a given burned area. If an intersection is found, its pairs of points are stored in the order found and designated by the order number of the first vertex on the crossing spans, as seen in figure 2.5.

3. The intersection list is reviewed to identify multiple intersections within a given segment. Multiple intersections on a segment are reordered to appear in the sequence in which they would be encountered starting from the first vertex.

If intersections are detected during the preliminary steps, the algorithm corrects the fire polygon by following the outer edge from the first vertex on the polygon. For an outward burning fire, it proceeds with each perimeter segment (pair of vertices) until an intersection with another segment is encountered. Vertices between intersections are stored separately to form what will be the corrected fire polygon. When the process finds one of the intersections identified above, it decides first the rotation direction produced by the intersection with the new perimeter segment and second the local shape of the fire front (convex or concave). These criteria are used to determine the next vertex to be processed (either in the existing direction around the fire polygon or in the reverse direction), see Table 2.1.

Table 2.1: Criteria used in FARSITE to determine the next vertex to be processed.

	Clockwise	Counter-Clockwise	Linear
Convex	reverse direction	reverse direction	existing direction
Concave	reverse direction	existing direction	existing direction

The intersection point is stored as the next vertex of the new fire polygon. The process continues until the algorithm arrives at the start and determines the vertices that now define the outermost fire perimeter. This process is repeated until there are no remaining unprocessed crosses on the original fire perimeter polygon. The process of fire front reconstruction is expensive in time and computing power.

2.3 FARSITE Accuracy Control

In FARSITE there are three different parameters that have a direct impact on the accuracy of the simulations. They are used to control the spatial and temporal resolution of the calculation and, therefore, on the execution time, [30], see Figure 2.6:

- **Time Step:** The time step is the maximum amount of time that the conditions at a given point are assumed constant so that the position of the fire front can be projected. The positions of all fires will be projected over this time step so that possible mergers can be computed. The time step is really of secondary importance compared with the spatial resolution of the calculations (as determined by the perimeter and distance resolutions below). FARSITE dynamically adjusts the *Time Step* to achieve a specified level of spatial detail determined by the *Distance Resolution*. The *Time Step* may also be reduced to ensure the use of all the time-specific wind data if the time until the next wind observation is less than that required to achieve the distance resolution.

- **Perimeter Resolution:** The perimeter resolution determines the maximum distance between points used to define the fire perimeter. It is a resolution of a fire front in the direction tangential to the fire perimeter at each point. The perimeter resolution controls the detail of the fire front, both in curvature and in the ability of a fire perimeter to respond to heterogeneous situations occurring at a fine scale. A low **Perimeter Resolution** is necessary to make a spreading fire sensitive to small-scale variations in spatial variables (e.g. fuels or topography varying on the scale of a few cells) as well as temporal changes in wind directions.

- **Distance Resolution:** The distance resolution is the maximum projected spread distance from any perimeter point. This distance cannot be exceeded in a time step before new fuels, weather, and topography data are used to compute the spread rate. It is the resolution in the radial or spreading direction for each point. The distance resolution depends somewhat on the

perimeter resolution because it cannot exceed that distance because of the potential for complex crosswalks between points. However, the distance resolution can be set shorter than the perimeter resolution. This would give a greater radial than the perimeter resolution.

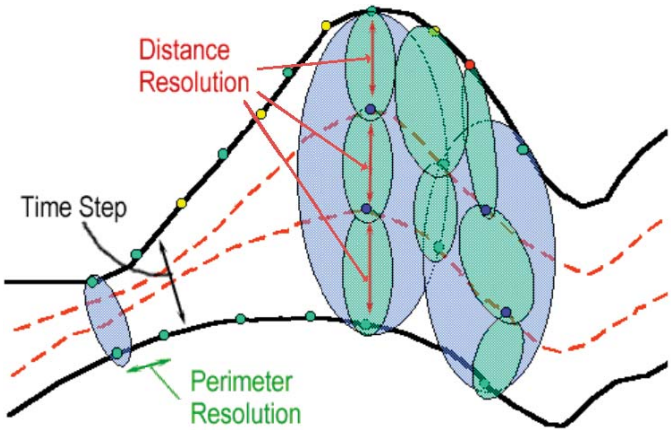


Figure 2.6: Graphic representation of time step (black), Perimeter Resolution (green) and Distance Resolution (red), [30].

The following procedure illustrates how FARSITE uses the *Time Step* and distance resolution to control a fire simulation. For a given *Time Step*:

1. Determine the fastest spreading point on the fire front and calculate the amount of time required to spread the distance resolution (including acceleration from its current state). This will be the new *sub-time step* required to achieve the distance resolution if it is less than the original *Time Step* or the time interval to the next wind observation.

2. Set the dynamic time step to this *sub-time step*.

3. Calculate the fire growth for the sub-*sub-time step*.

4. Calculate the time remaining in the original time step by subtracting the *sub-time step*.

5. Repeat steps 1 through 4 until the original *Time Step* is exhausted.

At the end of *sub-time step* each fire perimeter is re-discretized to check the

Perimeter Resolution. New points must be inserted at the mid-span of a perimeter segment if the distance between points is greater than the perimeter resolution.

These variables are defined in the Settings input file by the user at the beginning of the simulation, and they have a direct impact on the accuracy of the forest fire spread simulations. Nonetheless, the increment of the accuracy of the simulations has, as a consequence, an increment of the time invested into performing the simulation.

Chapter 3

Mixed Precision Methodology

The *Two-Stage Dynamic Data Driven Methodology* is an expensive process in terms of resources and execution time. For that reason, the first objective is to implement a strategy to reduce its requirements.

For historical reasons, most scientific codes, including forest fire spread simulators, have overestimated the needed numerical precision of a model leading to a situation where simulators use more precision than required without considering whether this precision is really needed.

In many cases, not all variables in a program need this double-precision. This over-engineering of the numerical precision code often leads to a situation where models are using more resources than required without a clear view of the actual needs of the program. The impact in the application execution of this implementation decision is far from trivial. On modern computer architectures, the performance of 32-bit operations is often at least twice as fast as the performance of 64-bit operations. By using a more appropriate choice of 32-bit and 64-bit floating-point arithmetic, the performance of some scientific applications was significantly enhanced without affecting the accuracy of the results, [38], [13], [43]. For this reason, it is adapting the computational models to use the numerical precision that is really required could pay back in terms of performance improvements. One research field that has gained momentum and can improve the performance

of a model is the use of mixed-precision. The main principle is to review programs from the observation that, in many cases, a single-precision solution of a problem can be refined to the degree where double-precision accuracy is achieved. Different studies have demonstrated the potential benefits that mixed-precision approaches can provide to many different kinds of scientific codes since it is possible to achieve substantial speed-ups for both compute and memory-bound algorithms requiring little code effort and acceptable impact on functionality, [9], [49]. This new way of managing floating-point precision that outstrips double-precision performance motivated us to study if using the mixed-precision models allows reducing the execution time of the forest fire spread simulations without losing accuracy. Therefore, we shall subsequently introduce the validation process used to determine that the obtained results applying mixed-precision guarantee the forest fire prediction quality. Later on, the methodology used to determine the suitable set of variables to be moved to single-precision is described.

3.1 Validation Process

As we have just mentioned, we need to compare the proposed mixed-precision model against a reference implementation (double-precision scheme) to ensure not to modify excessively the quality of the results. For that purpose, the forest fire area is divided into cells, and each cell will have a meaning according to the values depicted in figure 3.1. Considering these definitions, then, a *contingency table* can be constructed, as shown in Figure 1.5 where a perfect match would have data only on the main diagonal. Focusing in figure 3.1, the cells around the map that have been burnt by neither the reference simulation nor the mixed-precision simulation are considered *Correct Negatives* (CN). Those cells that have been burnt in both simulated fires are called *Hits*. The cells that are only burnt in the Reference simulation and are not burnt in the mixed-precision simulation are called *Misses*. Finally, in the opposite case, the cells burnt in the mixed-precision simulation, but the Reference simulation does not reach them are called *False Alarms* (FA).

To verify the proposed mixed-precision implementation, we used three different validation metrics, [10]. The first metric used is the *Bias score or frequency Bias*

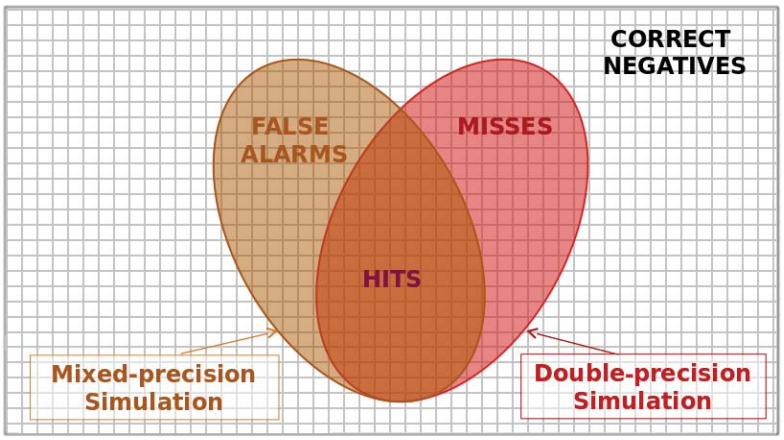


Figure 3.1: Events involved in metrics related to model verification.

Table 3.1: Elements of model verification expressed in the context of difference between sets.

DPCell	$Hits + Misses$	Cells burnt in the reference simulation
MPCell	$Hits + FA$	Cells burnt by the mixed-precision
UCell	$Hits + Misses + FA$	Union of cells burnt in both simulations
ICell	$Hits$	Cells burnt by both implementations

(BIAS). The *BIAS* represents the ratio of the number of correct forecasts over the number of correct observations. Equation 3.1 describes this validation function. This metric represents the normalized symmetric difference between the real and simulated maps.

$$BIAS = \frac{Hits + FA}{Hits + Misses} \tag{3.1}$$

The same *BIAS* equation can be expressed in terms of the difference between sets, see Table 3.1. The Equation 3.2 shows the obtained formula.

$$BIAS = \frac{MPCell}{DPCell} \tag{3.2}$$

The *BIAS* indicates whether the simulation has tendency to underestimate ($BIAS < 0$) or overestimate ($BIAS > 0$). The perfect has a *BIAS* value equal to

1.

The second validation metric used is the *False Alarm Rate* (FAR). The *FAR* measures the proportion of the wrong events forecast (see Equation 3.3).

$$FAR = \frac{FA}{(Hits + FA)} \tag{3.3}$$

Equation 3.4 shows the *FAR* validation function in terms of difference between cell sets.

$$FAR = \frac{(MPCell - ICell)}{(MPCell)} \tag{3.4}$$

The *FAR* is sensitive to *False Alarms*, but ignores the misses. A perfect comparison has a *FAR* value equal to 0.

The last metric used to validate our implementation is the *Probability of Detection of hits rate* (POD). *POD* takes into consideration the observed and positively estimated events. Thus, it represents the probability of an event being detected. The Equation 3.5 shows the *POD* validation function in terms of events and difference between cell sets.

$$POD = \frac{Hits}{Hits + Misses}$$

$$BIAS = \frac{ICell}{DPCell} \tag{3.5}$$

The *POD* is sensitive to hits but ignores the false alarms. The ideal value of the *POD* is 1.

Once we have described how we shall validate the obtained results when applying the mixed-precision approach, we introduced the proposed methodology to determine the set of variables to apply the precision reduction.

3.2 Mixed-precision Arithmetic

Our objective is to evaluate the sensitivity of the simulation quality and execution time when changing the variables of FARSITE that can use single-precision. In this way, we can estimate how the use of mixed-precision affects the *Two-stage Methodology* and, thus, the general prediction of the forest fire behaviour. As we will see, in our particular case, the performance benefice depends on the wildfire scenario considered. Depending on the local condition where the fire took place, the reduction of the execution time and the computational cost can vary more or less. However, in all situations, we obtain some performance improvement.

First of all, we perform a representative simulation without modifying the precision of any variable. The obtained simulation will be utilized as a reference against which we will validate any mixed-precision approach taken later on.

The next step is to define an accuracy threshold. As we saw, the validation functions used to verify the mixed-precision implementation are the *BIAS*, see Equation 3.1, *FAR*, see Equation 3.3, and *POD*, see Equation 3.5. Table 3.2 displays the thresholds used for the different metrics when the mixed-precision simulations are compared against the Reference one. If the computed error complies with all three thresholds simultaneously, the variable can be defined in single-precision. If the error does not satisfy any of the thresholds, the variable has to keep its double-precision.

Table 3.2: Threshold for the different validation metrics used to compare the Reference and mixed-precision simulations.

$1.05 > \text{BIAS} > 0.95$
$\text{FAR} < 0.05$
$\text{POD} > 0.95$

An individual accuracy test was done for each variable to analyze the global impact of the mixed-precision implementation. For each test, the precision of a single variable was reduced, and a new fire spread simulation was done. The obtained simulation was compared with the reference, and a new BIAS, *FAR* and *POD* were calculated. If the new values do not exceed the thresholds, the precision

reduction for this variable was accepted. If the error values were beyond the thresholds limits, the variable maintained its original double-precision definition.

Then, we repeated this process with all variables in the simulator. That is, a second variable was selected, its precision was reduced, and the validation process started again. Finally, when all variables were tested, the new forest fire spread simulator based on a mixed-precision model was obtained.

It is of mandatory importance to indicate that in this work, the fire front propagation and the fire front reconstruction algorithms are evaluated separately to have more detailed information on the performance and quality of both algorithms as they represent the two most relevant parts of the application. We will see that the performance of each algorithm has a direct impact on the final performance of the whole simulator.

At the end of the whole process, 266 different variables were tested, 221 for the fire front propagation algorithm and 45 for the fire front reconstruction algorithm. We found that nearly the 74% variables of the fire front propagation algorithm could safely use single-precision. In this case, we observed that the single-precision variables that have the highest impact in the fire propagation are those variables involved in the calculation of the rate of spread in a particular point, (R_0). For the fire front reconstruction algorithm, only 15% of the variables could use single-precision without compromising the accuracy of the prediction. In this case, the highest impact arises from those variables used to compute the crossovers.

This variable accuracy analysis is costly in time since to determine the necessary accuracy for each variable; a complete fire propagation simulation has to be done. Then, the validation values have to be computed. Some studies are working on different methods to automatically explore the precision needed for real variables without compromising the accuracy of the results, [33], [56]. Thus, we expect that this variable precision analysis will be done in a faster way in the future. Anyhow, its approach's most remarkable characteristic is that the work invested in finding the variables that can use a lower precision and the code's modifications only have to be done once, but the work done will be applied in all future executions.

For that reason, when we have to run thousands of simulations, although the earned performance benefit from a single execution is not prominent; therefore, the performance benefit obtained will be noticeable in the long run.

This study is a starting point for implementing the mixed-precision approach on Forest Fire simulators that work with accelerators and low consumption systems like GPUs and FPGAs. Some studies have demonstrated the the use of accelerators allows to reduce the execution time without penalizing the accuracy of the simulations,[52], [45], [25], [19], [35]. Considering GPU platforms, the number of floating operations per second (FLOPS) in single-precision is around 32 times better than in double-precision. Therefore, the use of 32-bits floating points representation implies that a theoretical speed up of 32 is possible. Consequently, the utilization of the mixed-precision in GPUs could lead to a significant execution time reduction.

3.3 Experimental Study and Results

In order to see how the mixed-precision affects the propagation of the fire, two different studies were carried out. First of all, we validate the methodology described in the previous section 3.2 by analyzing the mixed-precision effect on a classical prediction scheme, like described in Figure 1.2(a).

In the second study, we measured the effect of the mixed-precision implementation using the *Two-stage Methodology*, see Figure 1.2(b), to reduce the input uncertainty. In this study, we also want to analyze the possible limitations of the proposed mixed-precision methodology. As we will see, the actual impact in the simulation performance will depend on the characteristics of the fire. We will use two different fire scenarios to describe the performance variations in real cases. We have selected as study cases two different events belonging to the database of EFFIS (*European Forest Fire Information System*) [23].

In this research, a population size of 128 individuals was used, and the number of iterations (generations) was set to 10. All calculations reported here were performed using CPU Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz, with six

cores. All the individuals of a single generation of the Genetic Algorithm are simulated in parallel. For this analysis, we assess the accuracy of the output when the mixed-precision is utilized compared to the reference simulation.

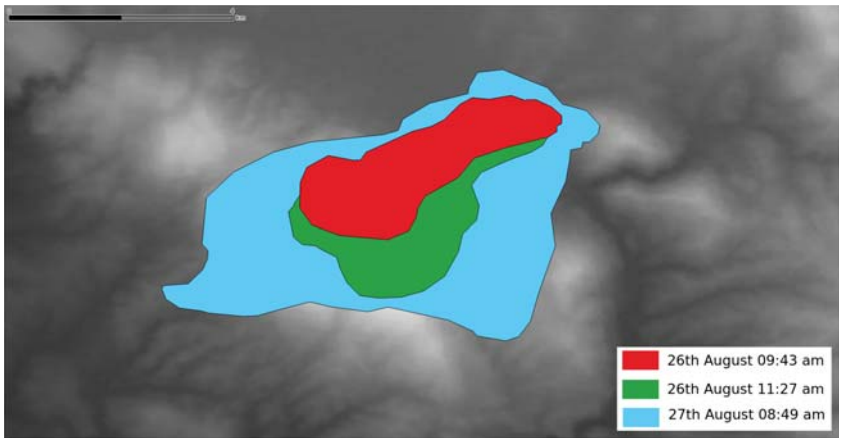


Figure 3.2: Digital Elevation map of Arkadia fire area and the three different perimeters of the forest fire. The *Perimeter 1-red* was used as the initial perimeter (ignition Perimeter), *Perimeter 2-green* was used in the calibration stage and the perimeter to be predicted is *Perimeter 3-blue*, [23].

3.3.1 Fire Front Reconstruction Oriented Fire Study: Arkadia Case

As first study case, we have retrieved the information of a past fire that took place in Greece during the summer season of 2011 in the region of Arkadia. The forest fire began on August 26th, and the total burnt area was $1,761$ ha. In Figure 3.2, we can see the fire perimeters at three different time instants: t_0 (August 26th at 09:43am), t_1 (August 26th at 11:27am) and t_2 (August 27th at 08:49am). The grid size of this fire is $100m$ x $100m$. The reason for choosing this fire is because characteristics of the terrain benefit a slow propagation of the fire front, and the final spread is relatively modest. So the time invested in propagating the perimeter points is paltry. Besides, the complexity of the terrain implies that as the fire progresses, it could form a large number of complex loops and knots to be solved. Therefore, the weight of the Fire Front Reconstruction is much higher than the point expansion algorithm.

Considering the whole execution, this case represents the worst condition case for our methodology.

Methodology Validation

First of all, we apply the classical prediction scheme for simulating fire behaviour. In this part, we perform the simulation of the burned area using a *Perimeter resolutions*, see Section 2, of 100 meters. The main objective of this section is to validate the methodology described above, analyzing the impact of the mixed-precision on the fire front evolution and the performance benefit.

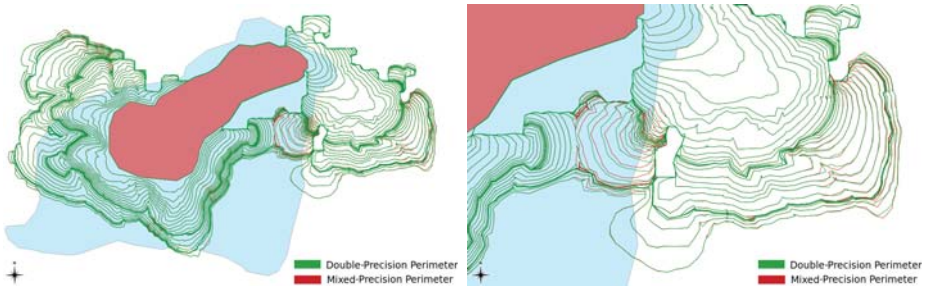
Table 3.3 displays the comparison of the real fire front evolution against the propagation of the simulated fire when the double-precision and the mixed-precision implementations are employed. When considering the different validation metrics, *BIAS*, *FAR*, *POD*, the values are very close for both implementations. Therefore, we can confirm that the two implementations reproduce real fire behaviour similarly. Nonetheless, the computed *ERROR* is slightly higher for the mixed-precision implementation.

Table 3.3: Results of the comparison of both simulations (double and mixed-precision) with the final perimeter of the real fire. *RealCell* represents the cells burnt in the real fire and *SimCell* represents the cells burnt by the simulation.

	double-precision	mixed-precision
RealCell	1,305	1,305
SimCell	1,571	1,566
UCell	2,136	2,136
ICell	740	735
BIAS	1.203	1.200
FAR	0.529	0.531
POD	0.567	0.563
ERROR	1.069	1.074

Figure 3.3 shows the obtained fire propagation when mixed-precision (*red*) and Reference (*green*) are utilized. We see that both simulations reproduce the behaviour of the real fire alike, see Figure 3.3(a).

Our objective is to validate the mixed-precision implementation; therefore, we



(a) Forest fire spread simulation.

(b) Detail of the difference between both obtained simulations.

Figure 3.3: Simulation of the Arkadia's fire. The initial perimeter is drawn in *red*. In b, the *dark red* line indicates the fire propagation when mixed-precision is used, the *dark green* line indicates the fire propagation when we use Reference. The *light blue* represents the final perimeter of the fire. The *Perimeter Resolution* is fixed to 100 meters.

are more interested in comparing the simulation when mixed-precision is used with the reference simulation. We observed that the difference between both simulations is not very significant, see Figure 3.3(b). For short propagation times, the difference between both simulations is negligible. However, this difference increases for higher propagation times. Nonetheless, the maximum difference between both perimeters is around a few hundred of meters.

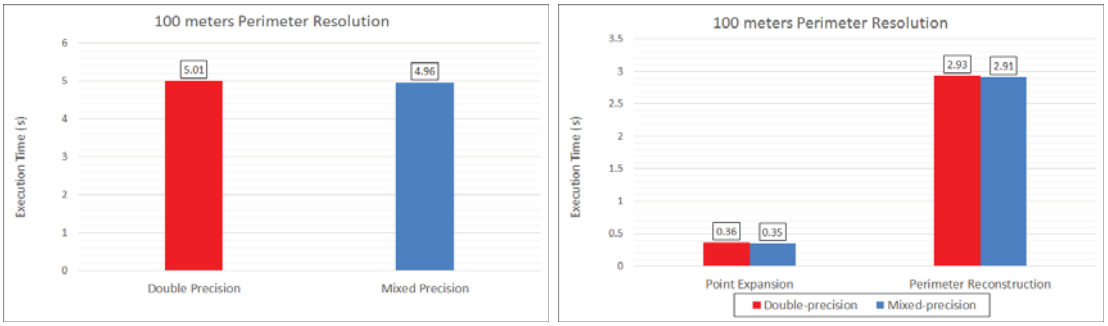
Due to the input data uncertainty, when a forest fire is simulated, it is impossible to reproduce the exact behaviour of the real fire. As we are more interested in the evolution tendency of the fire front, a maximum difference of hundreds of meters between fire perimeters is usually acceptable.

Table 3.4 shows the comparison between both simulations in terms of the difference between sets. We see that the number of cells burnt by both simulations is very close. In this case the number of *FA* is 8 cells and the number of *Misses* is 7 cells. These values are hardly significant in front of the $1,565$ cells burned by the reference simulation. When the validation values are computed, we obtain a *BIAS* and a *POD* very close to one and the *FAR* is equal to 0.04. The obtained results suggest that the use of mixed-precision does not excessively modify the simulation of the fire evolution.

Table 3.4: Obtained results of the comparison between the Reference simulation of the evolution of the fire and when using mixed-precision with a *Perimeter Resolution* equal to 100 meters.

DPCell	1,565
MPCell	1,566
UCell	1,573
ICell	1,558
BIAS	1.001
FAR	0.004
POD	0.995

Figure 3.4 summarizes the execution time for the reference simulation and for the mixed-precision implementation. In Figure 3.4(a), we describe the execution time measured for the different scenarios of the reference fire. In this case, the performance benefit is not very high when the mixed-precision is used. In this scenario, we obtain a speed up of 1.01. Figure 3.4(b) show the execution time of point expansion and Fire Front Reconstruction algorithms. It is easy to see that the fire front reconstruction algorithm takes more time than the point expansion algorithm. We can see that the execution time for the two implementations is very similar. The use of mixed-precision reduces the execution time of the point expansion into 0.01 seconds.



(a) Execution time of the two different implementations for all the simulation process.

(b) Execution time of the point expansion and fire front reconstruction algorithms.

Figure 3.4: Performance of two different implementations, the double-precision reference (*red column*), the mixed-precision (*blue column*) with a *Perimeter Resolution* equal to 100 meters.

As shown, the use of the mixed-precision methodology does not compromise the simulation of fire propagation; the quality of the simulation is maintained.

The performance impact detected in propagation is much better, but this impact is much more remarkable in the point expansion algorithm than in the fire front reconstruction.

Two-Stage Methodology

After validating our methodology and seeing that the mixed prediction utilization does not compromise the simulation's correctness, we now want to analyze the precision sensibility of the Two-Stage methodology to calibrate the fire's input parameters. The *Perimeter resolution* is equal to 100 meters in all cases. Our objective is to evaluate whether using the mixed-precision methodology can improve the Two-Stage Methodology's performance while keeping the same prediction accuracy.

As we said, the *Two-stage Methodology* consists of two different operations: the calibration stage, where the best individuals are selected, and the prediction stage, where the best individuals are used to perform the prediction of the fire behaviour. We present and compare two implementations: with double and mixed-precision using the same individuals in both cases. In a second analysis, to evaluate in detail the impact of the mixed-precision usage, we select the 5 individuals with the lowest average error to perform 5 different fire predictions.

To select the best individuals, the place where the fire is taking place is divided into cells, see Section 1.3.1. then we use a contingency table, see Figure 1.5, and the error Equation 1.4. The individuals with the lowest computed error are selected as the best individuals. Figure 3.5 reports the evolution of the error of the simulation per each generation during the calibration stage. As we can observe, the evolution of the computed error reveals an improvement of the reproduction of the real wildfire, generation after generation. Besides, we cannote that the computed error for both implementations is very close. These values confirm that the reference simulation and the obtained when the mixed-precision is used reproduce the behaviour of the real fire similarly. We detected that the difference

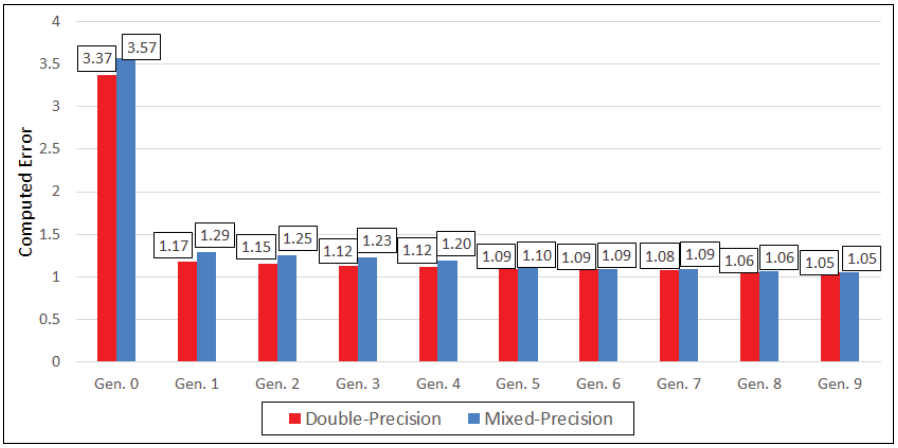


Figure 3.5: Error Simulation computed using Equation 1.4 when double-precision (red) and mixed-precision (blue) are used.

between the output fire perimeters is too small, a few meters; for that reason, it is not possible to distinguish the difference between the fire propagation of both implementations in the calibration stage. In this scenario, the five best individuals tend to underestimate the real fire perimeter.

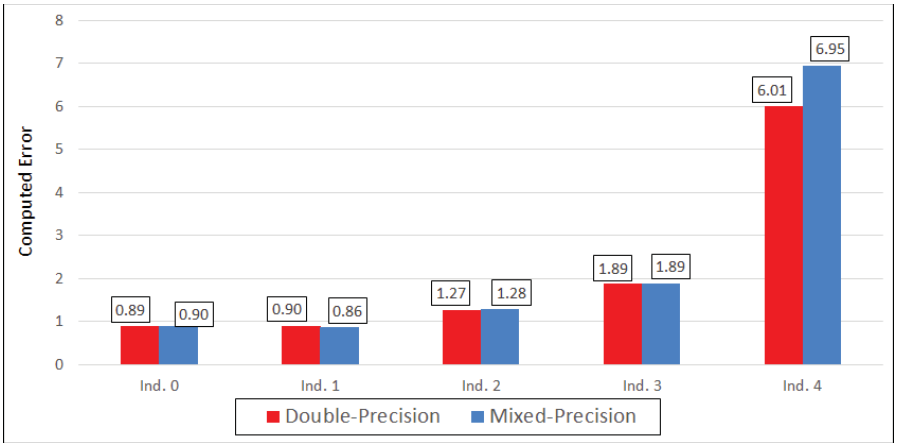


Figure 3.6: Computed error using Equation 1.4 for the prediction of the Arkadia's fire when double-precision (red column) and mixed-precision (blue column) are used.

After the calibration stage, we select the same five individuals for both double and mixed implementations to evaluate the impact of the mixed-precision implementation of affecting the calibration algorithm. Figure 3.6 plots the error for the

different fire spread predictions when the five best individuals are used in both implementations. We can see that the calculated error is very similar in all cases for both implementations. In the first and the fourth prediction, the difference of the computed error is minimal. The highest difference, 0.93, is obtained when the fifth individual is used. This individual tends to overestimate most of the propagation of the fire front. In Figure 3.7, we can see the difference of the fire front between the reference simulation (green) and the mixed-precision (red) of the fifth individual. We observe that the difference between both implementations is more evident as the spread of the fire is extended. This is because, for short distances, the propagation difference between both implementations is negligible. Though, as the propagation of the fire increases, this difference is accumulative; therefore, for the most extended fire fronts, the difference between both perimeters is more significant. However, when a forest fire is simulated, it is impossible to reproduce the exact behaviour of the real fire, which is why we are more interested in the general fire evolution tendency. In this fire evaluation context, a difference of a few meters between fire perimeters is usually acceptable.

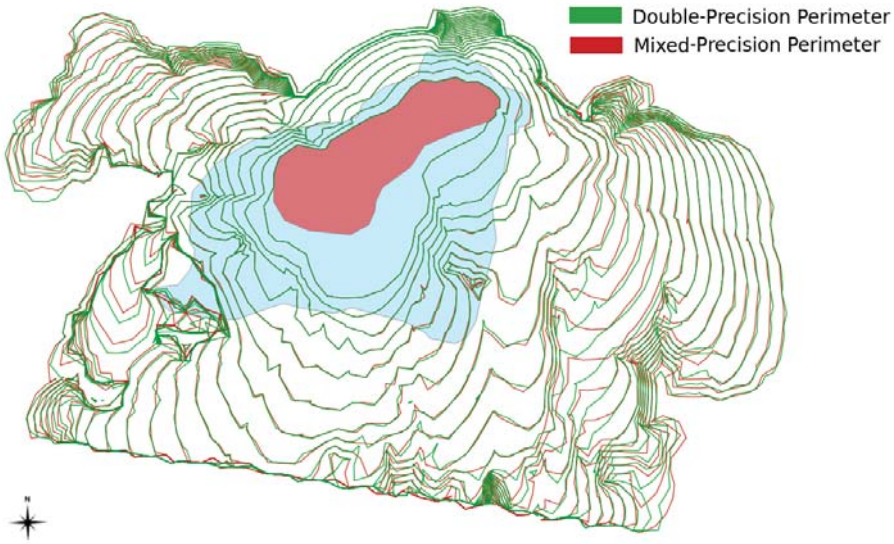


Figure 3.7: Obtained simulations of the Arkadia's fire after prediction stage when the fifth individual is used. The initial perimeter is drawn in *green area*. The *dark red* line indicates the fire propagation when mixed-precision is used, the*dark green* line indicates the fire propagation when we use double-precision. The *light blue* area represents the final perimeter of the fire.

Table 3.5: Obtained results of the comparison between the double-precision prediction of the fire spread and when mixed-precision using the *Two-stage Methodology*.

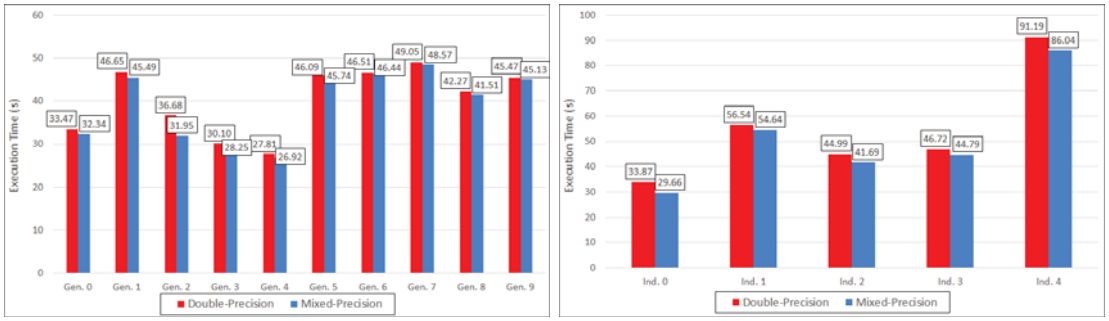
	1st Pred.	2nd Pred.	3th Pred.	4th Pred.	5th Pred.
DPCell	5,907	6,042	5,850	5,563	7,055
MPCell	5,935	6,076	5,862	5,578	7,086
UCell	5,961	6,100	5,912	5,607	7,146
ICell	5,881	6,018	5,800	5,534	6,995
BIAS	1.004	1.005	1.002	1.002	1.004
FAR	0.009	0.009	0.010	0.008	0.012
POD	0.996	0.996	0.991	0.995	0.991

Table 3.5 shows the comparison between simulations in terms of the difference between sets. We see that the number of cells burnt by both simulations is very close. For all individuals the difference between *DPCell* and *MPCell* is less than 35 cells. The computed error of the best individual is 0.9 for both implementations, while the error for the classic prediction scheme is higher than 1.06, even we use a higher *Perimeter resolution*. In this case, the computed validation values for the best individual are $BIAS = 1.004$, $FAR = 0.009$ and $POD = 0.996$. A BIAS value higher than one means that the mixed-precision implementation burns more area than the reference simulation.

These results mean that the difference between both simulations is shallow. When a wildfire is simulated, the most basic information is related to the evolution tendency of fire behaviour, which is the most relevant information that the firefighter can use to tackle these hazards efficiently. For this reason, we can confirm that the mixed-precision implementation does not excessively influence the quality of the prediction when the *Two-Stage Methodology* is used.

It is time to see the impact on the reduction of the execution time. Figure 3.8 displays the execution time invested in achieving a single prediction using the *Two-stage Methodology* in a forest fire spread. In Figure 3.8(a) we detail the execution time per generation in the Calibration Stage. Here we have to notice that the execution of all individuals is done in parallel, so the slowest individual determines the execution time of the generation. It shows that the impact of mixed-precision is not very significant in terms of execution time. The best improvement is obtained

in the third generation, where we compute a speed up of 1.14. Figure 3.8(b) indicates the execution time to simulate each individual of the Prediction Stage. As in the calibration stage, the repercussion of the mixed-precision methodology shows a modest performance improvement in terms of execution time. However, we can see that the mixed-precision methodology reduces the execution time of the last individual, which represents the longest prediction, in 5.15 seconds.



(a) Maximum execution time invested to simulate a generation of the Calibration Stage.

(b) Execution time spent to perform the prediction of the best individuals.

Figure 3.8: Execution time comparison of the *Two-stage Methodology* for two different implementations, reference and mixed-precision in Arkadia's fire.

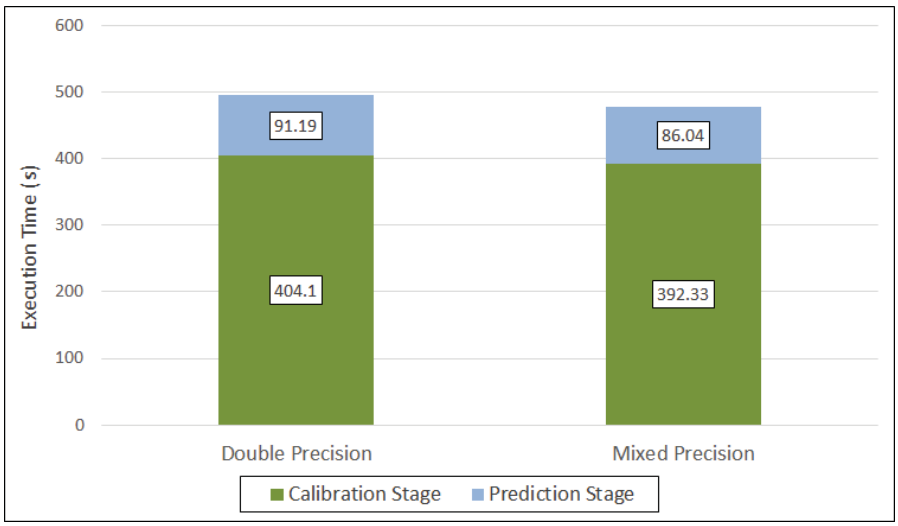


Figure 3.9: Execution time of the *Two-stage Methodology* for two different implementations, reference and mixed-precision in Arkadia's fire.

As we said at the beginning of this section, the simulation of each individ-

ual of a generation is done in parallel. Therefore if all individuals are executed simultaneously, the time invested in applying the *Two-stage Methodology* is determined by the slowest individual of each generation and the slowest individual of the prediction stage. Figure 3.9 compares the execution time invested in the *calibration stage* (lower green) and the *prediction stage* (upper blue) for the the two different implementations: double and mixed-precision. The time invested in the calibration stage is the sum of the execution time of the slowest individual of each generation. The execution time reduction is 16.92 seconds, 11.77 seconds for the Calibration Stage, and 5.15 for the Prediction Stage. The speeds up are 1.03 for Calibration Stage and 1.06 for the Prediction Stage, respectively. The speed up of the whole process is 1.04. In this case, the use of mixed-precision produces a modest execution time reduction of the *Two-stage Methodology*.

Table 3.6 compares the error of the classic scheme and the *Two-stage Methodology*. In an emergency, the utilization of the *Two-stage Methodology*, can only produce a single prediction, the prediction for the best individual after the calibration stage. For that reason, table 3.6 uses the computed error of the best individual (the first of the Figure 3.8(b)).

If we compare the performance of both implementations, see Figures 3.4(a) and 3.9, we observe that the *Two-stage Methodology* is much slower than the classic scheme; however, its prediction improvement justifies this execution time increment.

Table 3.6: Comparison of the computational error, see Equation 1.4, for the Classic Scheme and the *Two-stage Methodology*.

	Double-precision	Mixed-precision
Classic Scheme	1.069	1.074
Two-Stage Methodology	0.896	0.904

As we saw in the classic scheme, the quality of the forest fire spread simulation is not affected by the utilization of the mixed-precision strategy. Nevertheless, the reduction of computational cost and execution time is relatively modest. The collected data reveals that the scenarios, like the Arkadia's fire, where the characteristics of the terrain benefit a slow evolution of the fire spread and to form a

large number of complex loops and knots to be solved, the fire front reconstruction algorithm has a higher weight than the point expansion algorithm. Therefore, the performance improvement is meager. This is due to the fire front reconstruction algorithm representing around the 60%, see Figure 2.2 of the execution time; therefore, it is a significant bottleneck that limits improvement of the point expansion performance. However, when we analyzed which variables can be expressed in single-precision, we observed that only 15% of the variables used to reconstruct the perimeter of the fire could use single-precision without compromising the simulation of the fire spread.

3.3.2 Point Expansion Oriented Fire Study. Nuñomoral Case

The second study case took place in 2009 in the region of Nuñomoral, Spain. The forest fire began on July 25th, and the total burnt area was $3,314$ha. In Figure 3.10, we show the fire perimeters at three different time instants: t_0 (July 26th at 11:27am), t_1 (July 27th at 10:32am) and t_2 (July 28th at 11:15am). As in the first scenario, the grid size of this fire is $100m$ X $100m$. The main reason to pick up this fire is that the characteristics of the place where the fire took place are very different from the Arkadia's fire. In this case, the nature of the terrain fostered the fast propagation of the fire spread. The burned area is wider than in the Arkadia's fire, so we have a larger number of perimeter points. Then, the time invested in point expansion is more relevant than in the previous study case. Also, the complexity of the terrain is lower than in the previous scenario, which reduces the possibility that the fire front forms complex loops and knots. That combination of these factors: velocity and extension, produces an increment of the total weight of the point expansion in the global computation; therefore, the benefits of mixed-precision could produce a significant performance improvement.

Unlike the previous case, we only present the obtained result when the *Two-stage Methodology* is applied, see Figure 1.2(b), as the procedure used to validate the methodology is the same as the presented in the previous section, 3.3.1. As in Arkadia's fire, the obtained results show that the quality of the forest fire spread

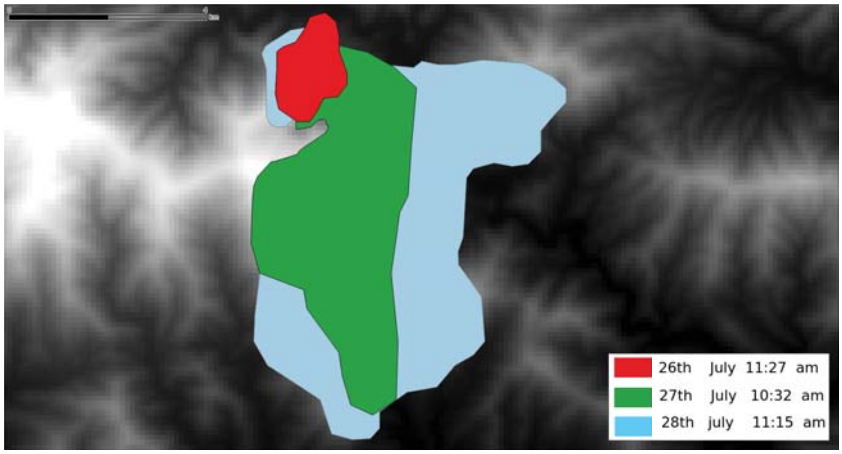


Figure 3.10: Digital Elevation map of Nuñomoral fire area and the three different perimeters of the forest fire. The *Perimeter 1 - red* was used as initial perimeter (ignition Perimeter), *Perimeter 2 - green* was used in the calibration stage and the perimeter to be predicted is *Perimeter 3 - blue*, [23].

simulation is not compromised when the mixed-precision is used.

Two-Stage Methodology

In this section, we apply the *Two-stage Methodology* to calibrate the input parameters of the fire. As in the case of Arkadia's fire, to effectively compare the double and mixed-precision implementations, the same individuals are used in both cases. The *Perimeter resolution* is fixed to 100 meters. After the calibration stage, the 5 individuals with the lowest error value are used to perform fire predictions. The error of each simulation is calculated using Equation 1.4.

In Figure 3.11, we can see the evolution of the computed error of the simulation through the generations of the calibration stage. As we can see, the progression of the computed error reveals that the simulated perimeter of the fire looks more and more like the real fire perimeter. Moreover, we can see that the error is lower for the double-precision simulation than the mixed-precision. Nevertheless, at the end of the calibration process, the error of the tenth generation is very close for both implementations. We observed that the two simulated fire fronts are very similar.

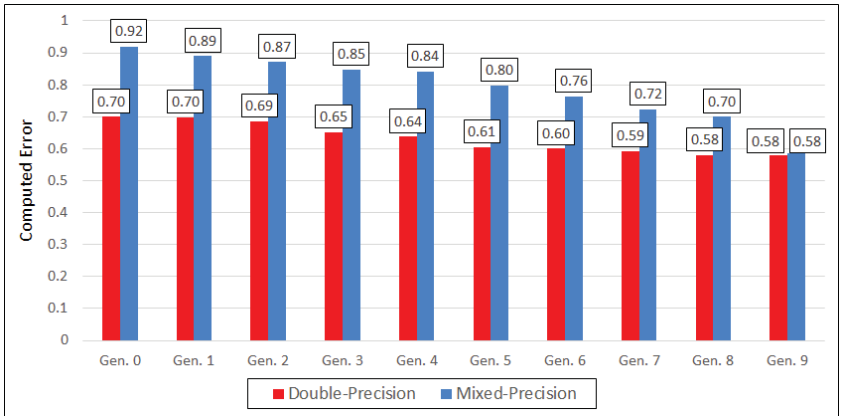


Figure 3.11: Error Simulation computed using Equation 1.4 when double (red) and mixed-precision (blue) are used.

At the end of the whole calibration process, the same best individuals are selected for the double and mixed-precision implementations. For this analysis, we can assume that the mixed-precision does not alter the selection of the individuals of the calibration process. Figure 3.12 reveals the computed error of the predictions when the same five best individuals are used in both implementations. We see that, in all predictions, the error for both implementations is very close. In the example, we will use the third individual, as the prediction produces the lowest error.

The maximum error difference is obtained with the second individual in this scenario. Figure 3.13 details the difference of the fire evolution between double (green) and mixed-precision (red) when the second individual is used. We can see that the difference between the perimeter evolution of both implementations increases as the extension of the fire widens. As we said, the reduction of the precision produces a slight difference when Rothermel's model is applied, see 2.11. This difference is accumulative and increases with the velocity of the propagation of the fire front. We notice that in some parts of the maps, the perimeter of the fire spreads very fast.

Table 3.7 compares the five fire predictions when the two implementations are used in terms of the difference between sets. It shows that the number of cells burnt by both simulations is very close. In this case, the computed validation values for the best individual are $BIAS$= 1.012, FAR= 0.018 and POD= 0.993. A

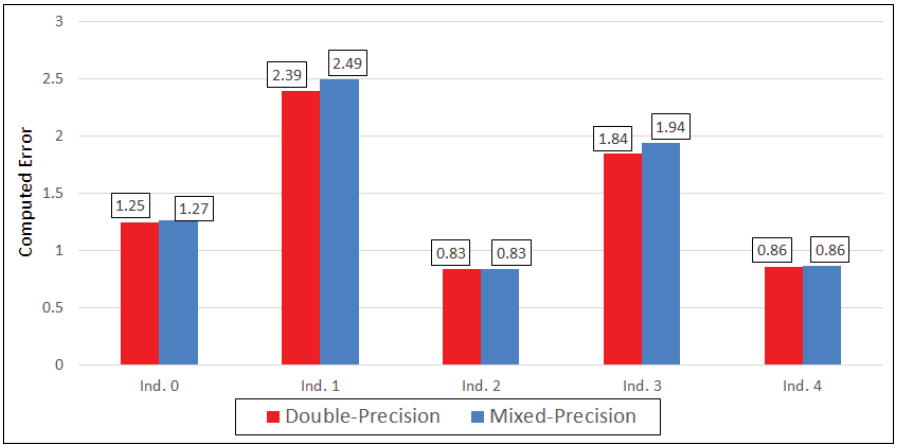


Figure 3.12: Computed error using Equation 1.4 for the prediction of the Nuño-moral's fire when double-precision (red) and mixed-precision (blue) are used.

Table 3.7: Obtained results of the comparison between the double-precision prediction of the fire spread and when mixed-precision using the *Two-stage Methodology*.

	1st Pred.	2nd Pred.	3th Pred.	4th Pred.	5th Pred.
DPCell	2,569	3,217	1,666	3,585	1,798
MPCell	2,599	3,224	1,666	3,579	1,802
UCell	2,616	3,332	1,666	3,648	1,807
ICell	2,552	3,121	1,666	3,516	1,793
BIAS	1.012	1.002	1.000	0.998	1.002
FAR	0.018	0.032	0.000	0.017	0.005
POD	0.993	0.970	1.000	0.980	0.997

BIAS value higher than one means that the mixed-precision implementation burns more area than the reference simulation. We can see that the mixed-precision implementation produces the same prediction of the fire behaviour when the third individual is used. Its validation values are $BIAS$= 1.000, FAR= 0.000 and POD= 1.000. In this case, the propagation of the fire front is not very extended. The computed validation values confirm that the difference between both simulations is low.

A consequence of the mixed-precision implementation is that, in some circumstances, the fire perimeter shape is simplified due to its lower accuracy. This outcome has an indirect repercussion in the execution time. When the shape is

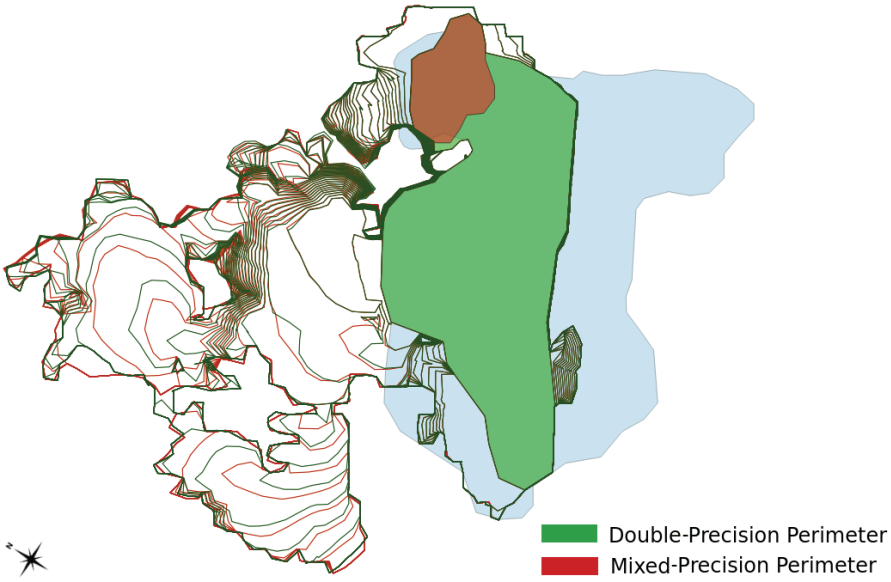
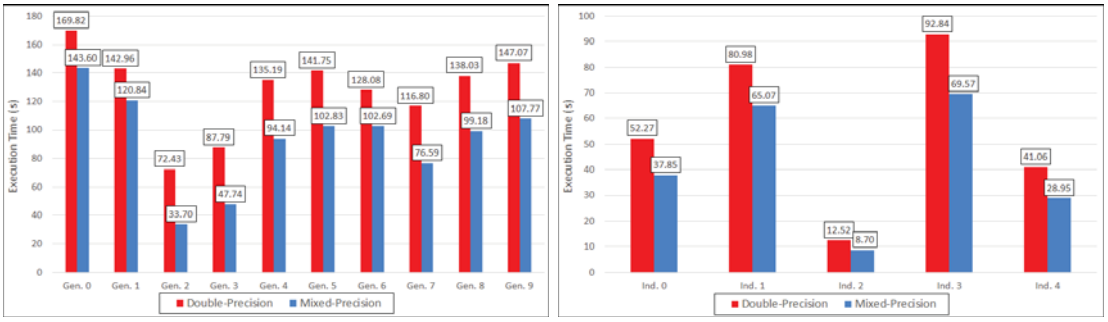


Figure 3.13: Obtained simulations of the Nuñomoral's fire after prediction stage when the second best individual is used. The initial perimeter is drawn in *green* area. The *dark red* line indicates the fire propagation when mixed-precision is used, the *dark green* line indicates the fire propagation when we use double-precision. The *light blue* area represents the final perimeter of the fire.

smoother, see Figure 3.13, we need a fewer number of points to represent the fire front; therefore, the time consumed by the point expansion and the fire front reconstruction algorithms decreases, which generates an execution performance improvement. However, the computed error increases in these cases but within the acceptable parameter range of values. As we mentioned above, when a wildfire is simulated, the most basic information is related to the evolution tendency of fire behaviour, which is the most relevant information that the firefighter can use to tackle these hazards efficiently. Therefore, we confirm that the mixed-precision implementation does not significantly reduce the prediction quality when the *Two-Stage Methodology* is applied.

In Figure 3.14 we illustrate the execution time when the *Two-stage Methodology* is used to predict the forest fire behaviour. Figure 3.14(a) displays the execution time per generation in the calibration stage. As we commented, the execution time of a generation is determined by the slowest individual. This figure shows a



(a) Maximum execution time invested to simulate a generation of the Calibration Stage.

(b) Execution time spent to perform the prediction of the best individuals.

Figure 3.14: Execution time of the *Two-stage Methodology* for two different implementations, the reference and the mixed-precision in Nuñomoral's fire.

considerable reduction in the execution time when the mixed-precision is employed. The highest performance improvement is obtained in the third generation, where the execution time is reduced a 53.5% when the mixed-precision is applied, which represents a speed up of 2.15. The worst performance improvement is obtained in the first and second generation, where the computed speed up is around 1.18 in both cases. Figure 3.14(b) illustrates the execution time of each individual from the prediction stage. As in the calibration stage, the usage of mixed-precision shows a significant reduction of the execution time. The third best individual obtains the maximum speed up, 1.44. If we focus only on the best individual, the speed up calculated is 1.38, representing a substantial performance improvement. Therefore, we can confirm that the use of mixed-precision methodology reduces the execution time.

Figure 3.15 details the execution time invested in completing a forest fire spread prediction when the *Two-Stage Methodology* is applied. The green column represents the execution time of the *calibration Stage* and the blue column the *prediction stage* for the two different implementations: double and mixed-precision. We can see that, in this fire scenario, the reduction of the execution time is notable, 374.12 seconds, 350.85 seconds for the Calibration Stage, and 23.27 for the Prediction Stage, which represents an improvement around the 27.3% of the execution time, 27, 4% for the Calibration Stage and 25.1% for the Prediction Stage. The

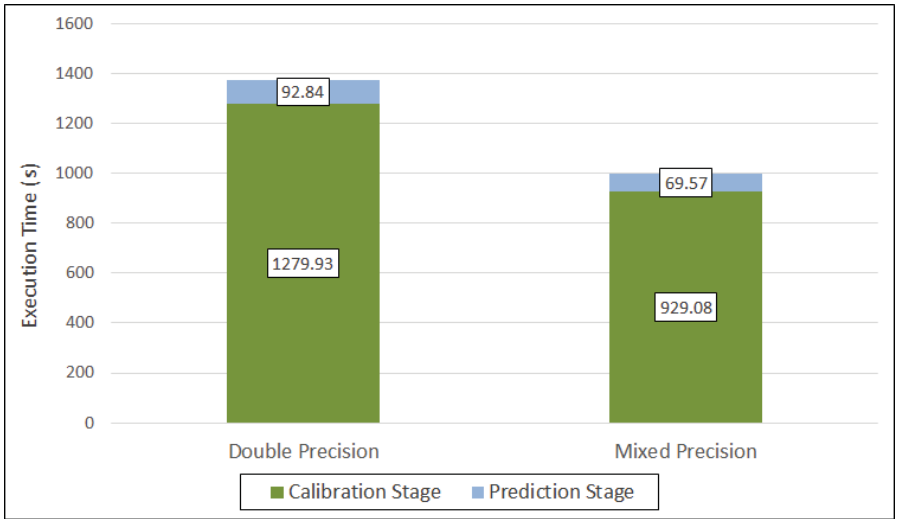


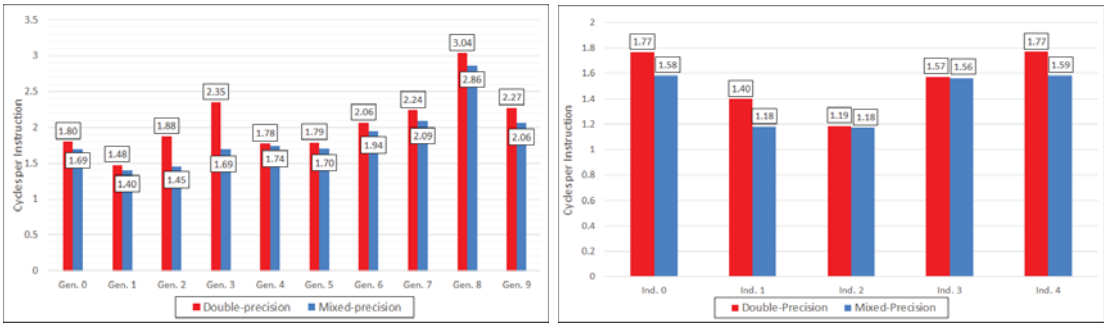
Figure 3.15: Execution time of the *Two-stage Methodology* for the different scenarios: double and mixed-precision in Nuñomoral's fire.

computed speeds up are 1.40 for Calibration Stage and 1.33 for the Prediction Stage, respectively. The speed up of the whole *Two-stage Methodology* is 1.37. If the performance of both scenarios is compared, 3.8, 3.9 and 3.14, 3.15, we note that for the case of Nuñomoral's fire the execution time reduction is significant.

In order to better understand the computational effect of the implementation of the mixed-precision in the *Two-Stage Methodology*, we used a profiling tool, *Linux prof*, to get the basic performance metrics of the experiments and compare the results of the execution. We will use CPU Cycles Per machine code Instruction (CPI) as a metric for comparing the computational cost of both implementations. The CPI is the ratio between the number of CPU cycles over the number of instructions executed. It reflects the average number of CPU cycles needed to complete an instruction. Thus, CPI is an indication of how much latency is in the system and can be a valuable measure of how an application is performing. A low CPI indicates that a few instructions are needed to execute an instruction. However, if the CPI is high, the execution of a single instruction requires a large number of CPU cycles, which indicates that the application has poor performance. Because CPI is affected by either change in the number of CPU cycles that an application takes or changes in the number of instructions executed, it is best used

for comparison when only one part of the ratio changes. The goal is to lower the CPI in certain parts of the code as well as the whole application. In our particular case, we want to understand the impact of mixed precision in the complexity of calculating specific functionality. CPI helps to determine the reduction of the processing complexity in those parts of the code where the mixed precision has been implemented, as in the point expansion and fire front reconstruction algorithms.

Figure 3.16(a) displays the average CPI for each generation in the calibration stage. It shows that the usage of mixed-precision implementation improves the average CPI in all generations. As we can see, the highest CPI reduction is found in the fourth generation, with a reduction of around 28%. This substantial computational cost reduction is due to a high number of individuals with an extended perimeter. In these individuals, the weight of the point expansion is considerable, so we have many potential instructions that can be simplified with mixed precision.



(a) Average CPI per generation in the Calibration Stage.

(b) CPI per individual of the Prediction Stage

Figure 3.16: CPI of the *Two-stage Methodology* for two different implementations, reference and mixed-precision in Nuñomoral's fire.

This notable difference is a consequence of the *explosion points*. An *explosion point* is an area of the terrain with the worst possible combination of the factors that drive the fire evolution. These factors are the slope, aspect, fuel, and wind. When these four factors are combined in a specific adversarial way, the acceleration of the fire front increases exponentially when the fire arrives at this area.

As we saw, there are a few differences between the fire propagation of both implementations. In the majority of the cases, this difference is negligible. Nonethe-

less, when double precision is used, the fire perimeter tends to spread slightly further than when mixed precision is used. We found that, in some individuals, this small difference in fire propagation causes the perimeter of the reference simulation to reach an *explosion point*, whereas simulations using mixed precision do not burn this zone, which leads to the difference between both burned areas could be considerable. Therefore, double precision simulation considers a larger number of perimeter points in these specific cases and induces a higher computational cost than equivalent mixed-precision simulation.

An important point is how the different fire spread of these individuals affects the burned area at the end of the prediction stage. We found that the final burned area is not influenced by these individuals, as the error function derives the GA, see Section 1.3.2, these expensive to simulate individuals are discarded for their high error rates.

Another consideration that justifies this computational cost reduction is due to the *Distance Resolution*, see Figure 2.6. When the fire front arrives at one *explosion point*, the perimeter points will propagate forward a long distance. As explained above, the Distance Resolution limits the maximum expansion of a single perimeter point in a single time iteration. For that reason, the propagation of these perimeter points is divided into a set of shorter propagation sub-steps, which implies a greater consumption of resources by the perimeter expansion algorithm. The propagation is much simpler when mixed-precision is applied as it needs fewer steps. If we evaluate the whole calibration stage, the obtained CPI is 1.91 for the reference simulation and 1.67 for the mixed-precision implementation, representing a reduction of 12.60% of the computational cost. So, we can conclude that the utilization of the mixed-precision produces a notable computational consumption reduction maintaining the quality of the simulation.

In Figure 3.16(b), we can observe the CPI for each individual of the prediction stage. It shows that, as in the calibration stage, the utilization of the mixed-precision provides a reasonable diminution of the computational cost. In this scenario, the highest CPI reduction when the simulation utilizes the second-best individual, in which the computational cost decreases from 1.40 CPI in the refer-

ence simulation to 1.18 when the mixed-precision implementation is used. This represents a reduction of the 15.5% of the computational cost. In a real forest fire emergency, the velocity to provide a prediction of the future fire behavior is very crucial; for that reason, only the best individual is used to predict the fire spread. If we focus on the best individual, we see a reduction of the 10.5% of the CPI. This computational cost reduction contributes to performance improvement when the mixed-precision approach is employed.

In conclusion, when we consider those fires with faster and more extensive fire front propagation, "Point Expansion Oriented" fires, more time is invested in the point expansion stage. Hence, the opportunity of performance improvement of mixed precision methodology is higher. In those particular cases, the land's characteristics where the fire take place generate a fast and large perimeter spread. Besides, if the complexity of the terrain is low, the likelihood of the fire front creating complex loops and knots are relatively small. These characteristics of the terrain may facilitate the apparition of explosion points, where the mixed-precision approach reduces the computational cost of the point expansion algorithm substantially. For these reasons, in these kinds of scenarios, the mixed-precision implementation can achieve notable performance improvements without compromising the quality of the simulation fire spread.

3.3.3 Mixed-precision impact comparison

This study presents two types of scenarios instanced by Arkadia's and Nuñomoral's fires. The purpose of using different fires is to show the sensitivity of the mixed-precision impact on the local conditions of the scenario where the fire occurred. Depending on local conditions, we can classify the wildfires as *Fire Front Reconstruction Oriented*, as Arkadia, or *Point Expansion Oriented*, as Nuñomoral. However, this is not a permanent classification. Not all forest fires belong to just one type. Some fires could begin to be more Fire Front Reconstruction Oriented, and as the fire front evolves, become more Point Expansion Oriented. Also, a fire could be more Point Expansion Oriented in some zones of the fire front and Fire Front Reconstruction Oriented on others.

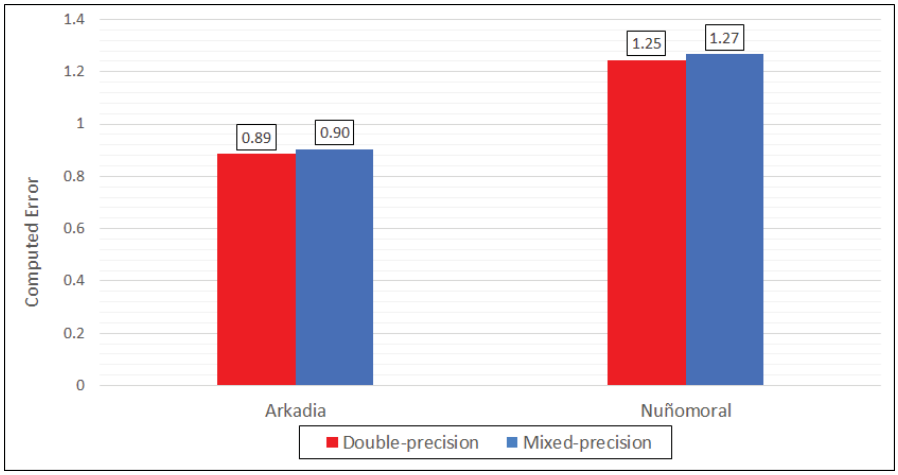


Figure 3.17: Quality comparison of the two different fire scenarios, Arkadia and Nuñomoral when the mixed-precision is used.

Our results highlight that the quality of simulations is not substantially modified by utilizing the mixed-precision in any case. However, the reduction of the computational cost and the execution time depends on the study case. Figure 3.17 displays the computed error of the prediction using the best individual after the calibration stage of the two types of fires studied, Arkadia and the Nuñomoral when the *Two-stage* methodology is applied. We can detect that the utilization of mixed precision does not significantly increase the error of the simulations. As we see, in the case of Arkadia's fire, the increase of the error is only 0.01. In the case of Nuñomoral's fire, this raise is around 0.02, representing an increment of the 1.12% and 1.60% respectively of the computed error. Most important is to highlight that the utilization of the mixed-precision does not negatively affect the quality of the simulations. As we commented, the relevant information when we perform a forest fire spread simulation is to predict the general forest fire propagation. In all scenarios, the utilization of the mixed-precision does not alter the forest fire's general behaviour simulated.

In order to effectively compare the computation cost and the performance impact of mixed-precision in both scenarios studied, we introduced the metric seconds per point ($Time/Point$), see Figure 3.18. This metric reflects the average time needed to propagate a single point from the initial perimeter to the final perime-

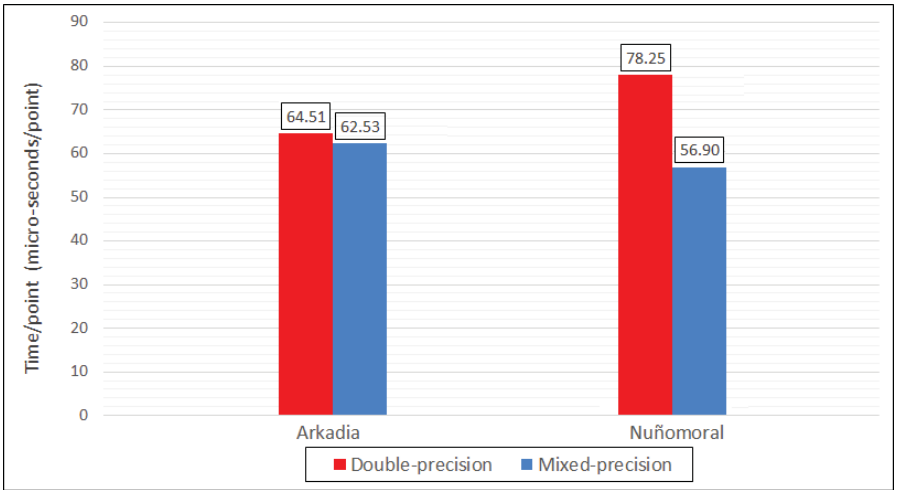


Figure 3.18: *Time/point* comparison on the two different fire scenarios, Arkadia and Nuñomoral, between the reference and the mixed-precision implementations.

ter; the higher it is, the more time is spent to spread one perimeter point. Figure 3.18 illustrates the average *seconds/point*. As we expect, in the case of Arkadia's fire, less time is invested in propagating one perimeter point. In this kind of fire, the impact of mixed-precision is moderated. However, in this scenario, the gain is only 3% of the time needed to spread a single point. If we focus on the Nuñomoral's fire, we see a significant reduction in the time required to propagate a single point when using mixed-precision. In this case, the decline of the *seconds/point* is around 24.5%. So, we confirm that the usage of the mixed-precision could produce an execution time reduction. We can express the performance improvement in terms of speed up. Figure 3.19 displays the comparison of the speed up of the two scenarios. On the one hand, it reveals that in Nuñomoral's case, applying the mixed-precision methodology allows improving the performance. The computed speed up is 1.37 when the Two-stage scheme is used. On the other hand, in the case of Arkadia's fire, the obtained speed up is close to 1, which indicates that, in this scenario, the implementation of the mixed-precision methodology provides a poor performance improvement.

As we saw, the repercussion of the mixed-precision strategy is different in the two scenarios studied. There exist significant differences between the characteristics of both fires. In the case of Arkadia, as a Fire Front Reconstruction Oriented

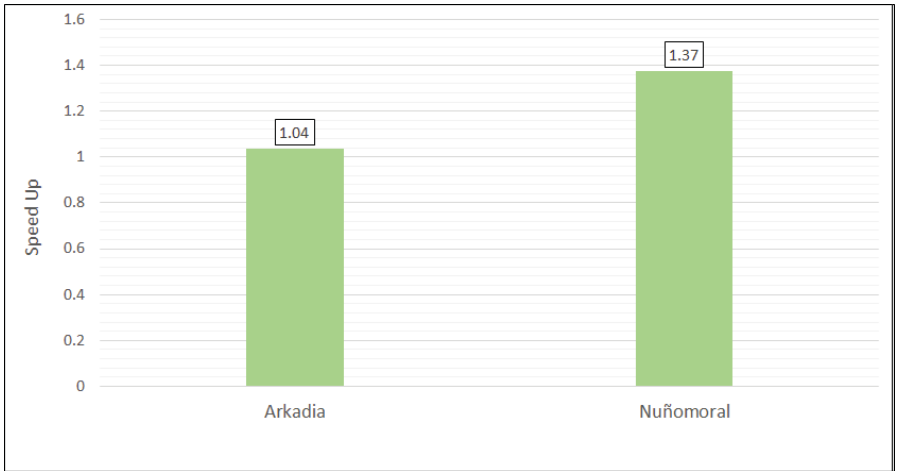


Figure 3.19: Speed up comparison of the two different fire scenarios, Arkadia and Nuñomoral when the mixed-precision approach is used.

fire, the peculiarity of the zone where the fire took place helps to the relatively slow velocity of the fire propagation. Moreover, the complexity of the region is conducive to forming complex loops and knots. For that reason, the weight of the point expansion is meager in front of the fire front reconstruction, which consumes the majority of the execution time to perform a simulation. On the other side, in the Nuñomoral's scenario, a Point Expansion Oriented fire, the fire front propagation is quite fast and the extension of the burned area very high, also the complexity of the place where the fire burned is lower than in Arkadia's fire. This low complexity helps to reduce the number of knots and complex loops. These land characteristics reduce the weight of the fire front reconstruction and increase the proportion of the execution time invested in the point expansion algorithm. In terms of quality, the utilization of the mixed-precision does not compromise the quality of the fire predictions. We are interested in predicting the general evolution tendency of fire because it is the most useful information that fire firefighters can get. In both cases, both implementation, the double and the mixed-precision predict the same fire propagation tendency. In all scenarios, the computed error for both implementations is very close.

The Point Expansion Oriented fires, Nuñomoral's, tends to have a more considerable performance improvement than in the case of Fire Front Reconstruction

Oriented fires, like Arkadia's case. This improvement variation could be caused due to the higher terrain complexity, like in the Arkadia's fire. In the fire front reconstruction algorithm, all the pairs of perimeter points are checked against all the other pairs of points to find loops and knots. This operation is repeated until all the crosswalks are solved. When the simulation is done in complex terrain, the number of loops and knots is higher. Therefore the checking process of the points has to be repeated more times, which implies that the weight of the fire front reconstruction algorithm raises. The fire front reconstruction algorithm complexity is $O(n^4)$, where n represents the number of perimeter points. As we commented, only the 15% of the variables invested in the fire front reconstruction can use single-precision; for this reason, the performance benefit is penalized. Thus, the performance improvement of the mixed-precision approach will be more substantial when the region where the fire takes place facilitates that fire becomes more Point Expansion Oriented.

Chapter 4

Fine-Grained Paralelization

One critical aspect when dealing with simulations related to wildfires taking place in complex terrains is the necessity of using high-resolution data to obtain reliable simulations. The use of finer resolution makes the spreading fire sensitive to small-scale variations in spatial variables (e.g., fuels or topography) and temporal changes in the weather conditions. Figure 4.1 compare the simulation of the forest fire spread when two different accuracy are used, *Perimeter Resolution* = 5 meters and *Perimeter Resolution* = 100 meters. We can see a notable difference between both *Perimeter Resolution* values. In addition, the utilization of low *Perimeter Resolution* implies an increment of the number of points where the fire front splits. This perimeter points increment will be allowed to substitute the currently Fire Front Reconstruction algorithm for an envelope algorithm, like α-shape, with low errors and reducing the execution time invested in the Fire Front Reconstruction. Nonetheless, when the precision of the simulation rises, the computational complexity of the model increases, and, therefore, its effectiveness in real-time prediction is reduced. For that reason, it is necessary to develop new methods to tackle this complexity to provide accurate fire spread simulations in a reasonable time. Several strategies have been improved the performance of wildfire spread simulators without altering the precision of the results. On the one hand, some studies apply multi-core architectures to accelerate forest fire simulations [20][12][7]. The main idea in these approaches consists of determining the number

of cores to allocate for a given simulation to accomplish particular time constrain. The main concern about these methods is their dependency on access to computing clusters in real-time, which operationally limits their use. Furthermore, these approaches are impractical for embedded applications that would be more appropriate for running the simulations on the field next to where the disaster is taking place. On the other hand, different works carried out to apply computational accelerators, like Graphic Processor Units (GPUs) systems, to the simulation of forest fire behaviour in order to accelerate these simulations [52][45][25][34][35]. These studies demonstrate that the computational capacity of GPUs can improve the simulator's performance in terms of execution time without losing the accuracy of the results for a given data resolution. However, most of these works focus on simulators based on CA, whose primary constraint is the low accuracy of their prediction results. Simulators based on the EWP scheme provide better predictions than those based on CA in terms of quality. Nonetheless, the execution time incurred in running one simulation based on EWP is higher.

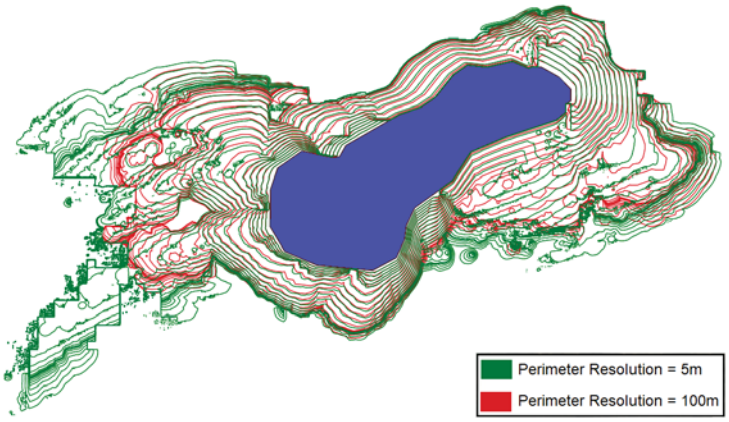


Figure 4.1: Comparative of the final fire perimeters simulated *Perimeter Resolution* equal to 100 meters (*Red*) and *Perimeter Resolution* equal to 5 meters (*Green*).

The main objective of this study is to present a novel Fine-Grained Parallelization implemented in GPU to increase the accuracy of the Forest Fires Spread simulator FARSITE without excessive execution time penalties. During our research, we focus on the parallelization of the Perimeter Propagation and the Fire Front Reconstruction algorithms, which are the most time execution expensive, see Figure 2.2. In order to program the GPU, the Compute Unified Device Architecture

(CUDA) [46] is used. The GPU activation is obtained in CUDA by writing device functions in C language called kernels. When a kernel is invoked, the CPU sends the data from the host memory to the GPU global memory and invokes the parallel computation on the GPU. Our GPU fire spread simulator has been compared against a CPU parallel implementation to analyze the performance improvement. To employ the maximum capabilities of the multi-core CPU, the point propagation algorithm was implemented using OpenMP (*Open Multi-Processing*) [24], which is a set of compiler directives, library routines, and environment variables. The following section describes the implementation of the Fire Front Propagation and the Fire Front Reconstruction parallelizations in detail.

4.1 Perimeter Propagation

The execution time invested into performing a simulation increases when the number of points increases in the forest fire spread simulators based on the EWP. So, simulations with high resolutions provide long execution times, which limits their use in real situations. In the forest fire spread simulators based on the EWP, the evolution of each point only depends on the local condition of that point; therefore, each point propagation is entirely independent of the other points' evolution. This characteristic suggests that substantial gains in performance could be obtained through parallel processing of each point expansion. CPUs are latency-oriented systems that use architecture features like large caches or branch prediction techniques to deal with data dependencies. GPUs, by contrast, are throughput-oriented systems that use massive parallelism to hide latency. Modern GPUs are multiprocessors with highly efficient hardware-coded multi-threading support. The key capability of a GPU unit is thus to execute thousands of threads running the same function concurrently on different data. Hence, GPUs are good target platforms for processing problems with large amounts of data. To effectively use a GPU, programmers need to deal with different challenges like memory management, load balance of the execution threads, data races, and data transfer cost. The data flow between the CPU (*Host*) and the GPU (*Device*) means that part of the program's execution time must be dedicated to copying data between CPU

and GPU detached memory systems. In previous work, it has been demonstrated that for high-resolution simulations, a parallel implementation of Fire Front Propagation in GPU has better performance than a CPU parallel implementation, [19].

4.1.1 OpenMP parallel Optimization

To explode the capabilities of the multi-core CPU, the point propagation algorithm was implemented using OpenMP. The workflow of the fire front evolution simulation process is shown in Figure 4.2. The OpenMP parallelization used in this research is based on the OpenMP implementation described in [8].

As can be observed, the fire spread is done using two nested loops. The outer loop is temporal. That is, it controls the lifespan of fire propagation. The simulated propagation time is broken down into *Time Step*. The *Time Step* is defined in the Settings input file by the user. However, in FARSITE, the spread of each perimeter point is done in a continuous space. For that reason, for a given *Time Step*, in some cases, the local characteristics of a single point produce such long propagation that crosses some places where the local conditions could be very different and affect the evolution of that point. To avoid this issue, the *Distance Resolution* is used, [30]. The *Distance Resolution* is the maximum projected spread distance from any perimeter point. This distance cannot be exceeded in a time step before new fuels, weather, and topography data are used to compute the new spread rate.

When the *Perimeter Resolution* is defined, the perimeter of the fire is split in a set of points, each of them with their coordinates X_i, Y_i. The CPU parallel implementation divides the front of the fire into zones with the same number of perimeter points. The number of threads that the CPU can support; determines the number of zones in which the fire front is partitioned. Figure 4.3 shows how the perimeter of the fire is partitioned to capitalize the multi-core architecture of the CPU. The assignation of the area is done in the same order as the threads. For example, all the points into zone 0 are propagated by thread 0, and the zone n is spread in the thread n.

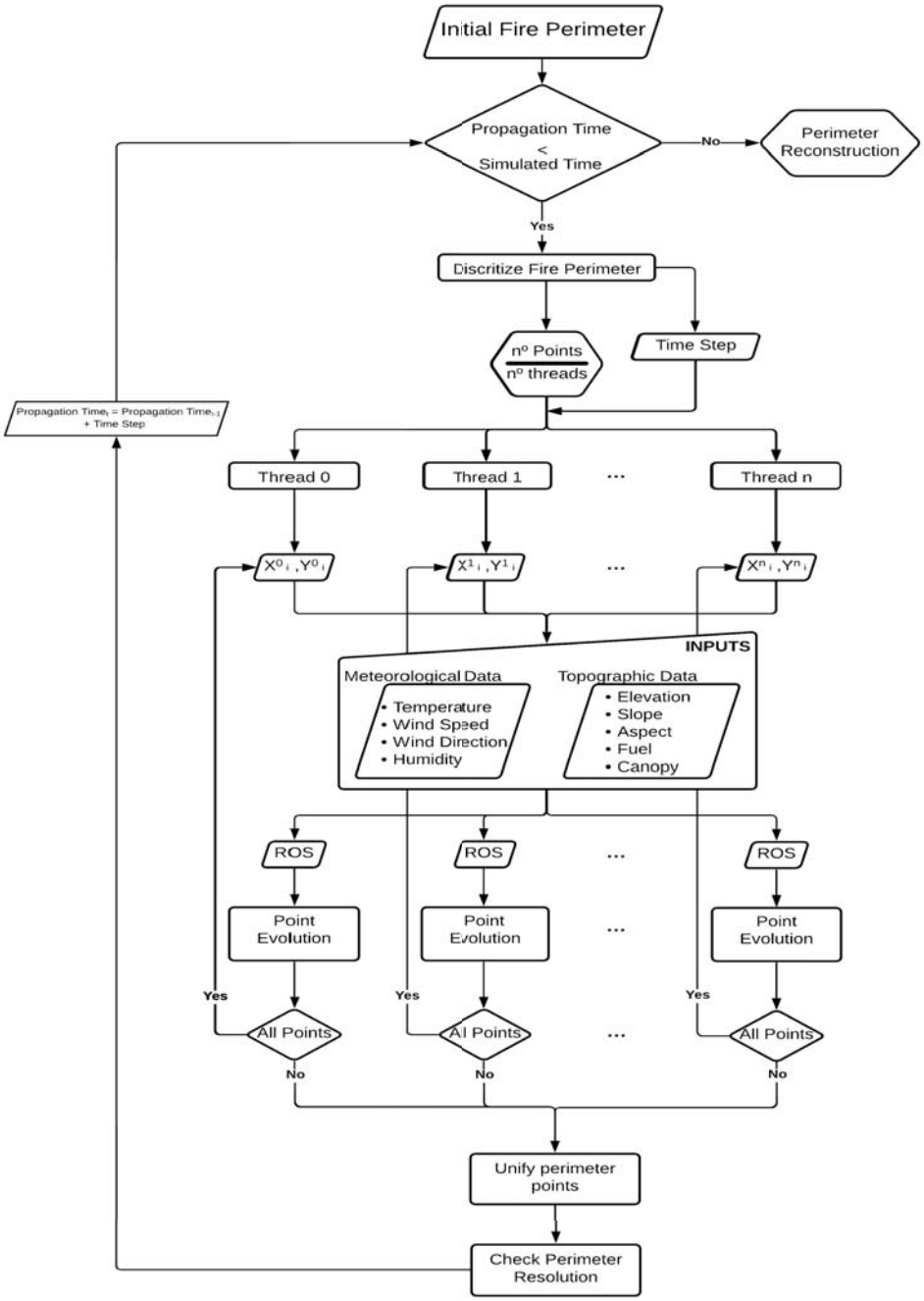


Figure 4.2: Perimeter propagation based on the Elliptical wave propagation implemented in OpenMP.

Next, for each zone or thread, a pointer of structures is created. These struc-

tures are used to store the information needed to propagate the points. The evolution of the perimeter points in the same perimeter partitioning is done in serial, so the information of a new point to evolve overwrites the information of the previous one. When the spatial loop starts, the simulator checks the coordinates X_i and Y_i, reed topography and the meteorological information of that point and writes the information into the corresponding structure.

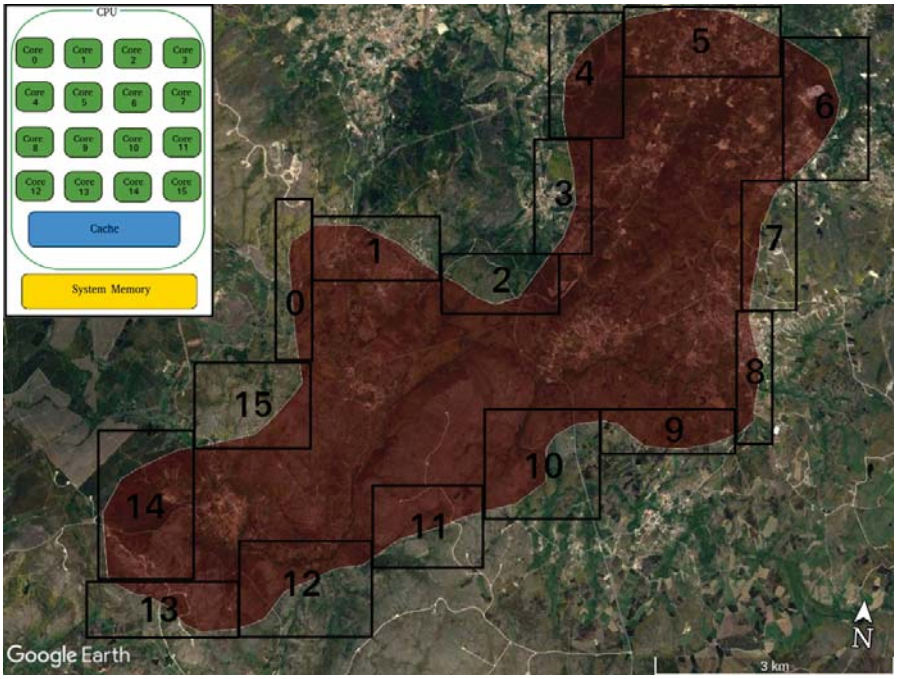


Figure 4.3: Fire front partitioning. The number threads executed by the CPU determine the number of divisions.

To expand a single point, two functional parts are coupled together to calculate the next fire front evolution. These functions run parallel for every area, but they run in serial for the same zone points. The first part computes the *ROS* for the point X_i, Y_i using the Rothermel Model without taking into account the wind and the slope of the terrain. This represents the *ROS* of that point on flat terrain with no wind. Then, the corrections of the wind and the slope effects are applied to the ROS. The second part uses the obtained *ROS* to calculate the shape of a propagation ellipse of the point. Finally the new coordinates X_i', Y_i' are stored in a vector. This procedure is repeated for all the points in the partition.

At the final of the spatial loop, there is a synchronized directive. This directive aims to ensure that all threads have been able to complete the propagation of all points in their perimeter zone. Finally, the new coordinates are stored in a global vector. The thread id determines the order where the area's information is stored, so the first points stored are the points of the thread 0 and so successively. When all the points have been written in the global vector, the *Perimeter Resolution* is checked. If the distance between points is higher than the *Perimeter Resolution*, new points must be inserted in the mid-span of a perimeter segment if it is required. At the end of the time loop, the new fire perimeter is reconstructed, and the operation is repeated for the next time step.

A critical aspect of dealing with in the event of a forest fire is the response time of emergency systems and the ability to act in the most efficient way to avoid significant damages. That implies making critical decisions as quickly as possible with the only support of existing information. In this context, the time invested in performing an informative simulation has to be as short as possible.

Then, the use of highly accurate, small *Perimeter Resolution* simulations provides long and unpractical execution times, which limits their use in real situations. In this work, a novel EWP algorithm is presented. Our proposal consists of exploiting the computational power of the GPU computational resources to perform a forest fire spread evolution with very high resolution without increasing the time invested in the point evolution process.

4.1.2 GPU parallel Optimization

Figure 4.4 displays the algorithm flowchart of GPU Perimeter Propagation implementation. In this implementation, only the temporal loop is needed. As we saw, the temporal loop controls the lifespan of fire propagation. The number of time iterations is related with the *Time Step* and the *Distance Resolution*, see Section 4.1.1. The temporal evolution of the fire perimeter must be done in serial because the position of the point X_i, Y_i at time $t + 1$ depends on its position at time t. The discretization of the fire front is done in the CPU. Then, the perimeter points (X_i,Y_i) are copied from the CPU memory (*Host*) to the GPU memory

(*Device*). A significant difference from the OpenMP implementation is how the fire variables (topology, vegetation, and meteorology) are passed to the Rothermel model to expand the points. While in the OpenMP implementation, a single structure to store the topology and meteorological information is created, in the GPU implementation, this structure is created for each perimeter point.

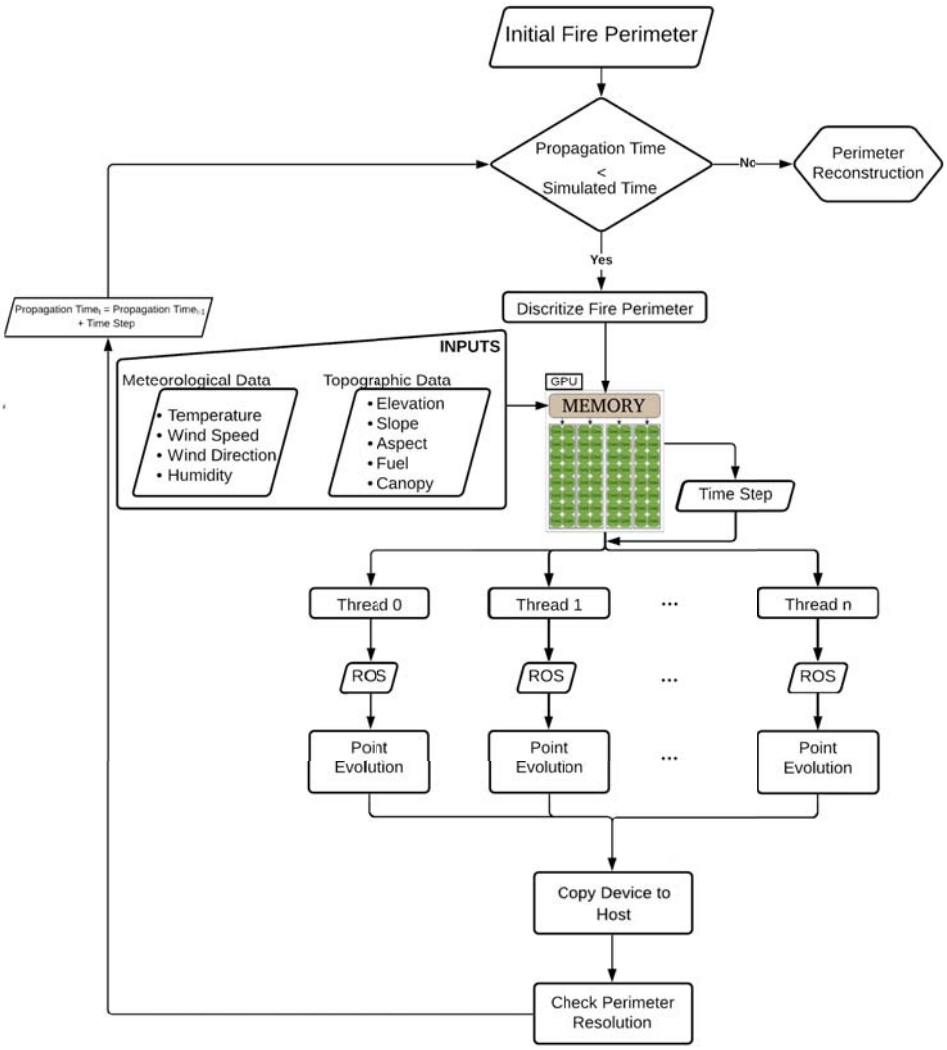


Figure 4.4: Elliptical wave propagation implemented on GPU (EWPG).

In our fine-grained parallel strategy, a single thread is assigned to compute a single perimeter point propagation. As we saw, in the forest fire spread simulations based on the elliptical wave propagation, the evolution of a perimeter point

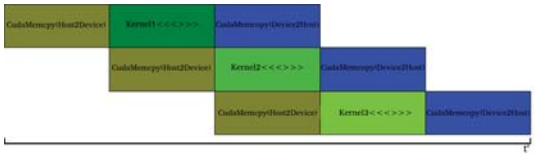
is independent of the propagation of the other; therefore, threads can be processed independently (i.e., in any order), having a unique identifier assigned by the CUDA framework. Here lies one of the main differences from the previous implementation: the sequential loop, overall points, is replaced by an implicit loop, where all elements can be calculated simultaneously.

In order to minimize the communication between the *Host* and the *Device*, the *Time Step* is computed in the GPU. As we said, the *Time Step* depends on the *Distance Resolution*; so, the *Distance Resolution* of each point is checked by its assigned thread. If the evolved distance is higher than the *Distance Resolution*, a new *Time Step* is calculated for each point. Finally, this new *Time Step* is used to compute the propagation of the perimeter point.

At the final of the temporal loop, there is an implicit synchronization to guarantee that all perimeter points are propagated. Then the computed coordinate X'_i, Y'_i are stored in a vector. Finally the vector is copied from the *Device* to the *Host*. The *Perimeter Resolution* is checked in the CPU, and, if it is necessary, new points are introduced in the fire front. Finally, the current fire perimeter is reconstructed, and the operation is repeated for the next *Time Step* until the simulation is finished.

One of the main drawbacks of this implementation is the large amount of data necessary to compute the propagation of a single point. To optimize the time invested in the data transfer from the *Host* to the *Device* we use the *CUDA Streams*. In the earliest versions of CUDA, all the kernels had to be executed sequentially. At the end of each Kernel, there is an implicit synchronization of the threads; so, the execution time is related to the number of kernels that our application executed. A *CUDA Stream* is a sequence of operations that execute on the *Device* in the order in which the host code uses them. While operations within a stream are guaranteed to execute in the prescribed order, operations in different streams can be interleaved, and, when possible, they can even run concurrently, [54]. *CUDA Streams* allows us to overlap the data transfer and the execution of the kernels. Figure 4.5 schematizes the how *CUDA Streams*.

(a) Execution of GPU implementation without *CUDA Streams*.



(b) Execution of GPU implementation when the *CUDA Streams* are used.

Figure 4.5: Difference between the CUDA execution without and with *Streams*. It can be appreciate that overlapping the data transfer with the Kernel execution could reduce the execution time notably.

The employment of the *CUDA streams* allows the concurrence of different kernel execution. In our case, we defined three different *CUDA streams*. At the beginning of the execution, the three streams optimize the data transfer between the *Host* and *Device*. FARSITE uses different structures to compute the evolution of the fire perimeter. This strategy lets us minimize the time invested in the copy data, for example, characteristics of the fuels or the weather data and features of the fire front. Next, the estimation of the *Time step*, the reading of topographical and meteorological data from the input files, is done at the same time. Because the kernel which computes the propagation of the perimeter points depends on the previous kernels, there is an explicit thread synchronization before it starts. As a result of this dependence, the execution of this kernel cannot be overlap with any other kernel. Finally, the new coordinates and features of the fire front are copied from the *Device* to *Host* exploiting the three different *CUDA streams*. When all the information is into the *Host*, the reconstruction of the fire front starts.

4.2 Fire Front Reconstruction

As we said, the main drawback of the forest fire spread simulators based on the EWP, like FARSITE, is that they do not intrinsically distinguish burned from unburned areas. For this reason, it is necessary an algorithm to rebuild the fire front of the fire after each time step. This process of Fire Front Reconstruction is

expensive in time and computing power because it checks all the possible points pairwise against the rest possible pairs of points. The complexity of the Fire Front Reconstruction algorithm is $O(n^4)$ where n represents the number of perimeter points.

4.2.1 CPU parallel Optimization

The Fire Front Reconstruction algorithm needs fewer memory requirements than the Fire Front Propagation algorithm. However, it is much more computationally expensive. Figure 4.6 displays the workflow of the fire front reconstruction process.

As we saw, the Fire Front Reconstruction algorithm test all the pairwise points against all the possible pairs of points. Two nested spacial loops are needed. As in the case of the Fire Front Propagation, see Section 4.1.1, when the implementation is running with OpenMP, the perimeter of the fire is divided into different areas, see Figure 4.3. The number of zones is determined by the number of threads used to execute the Fire Front Reconstruction algorithm.

Each thread tests the pairwise point of its zone against the pairs of points of the whole perimeter. The first pair of points, X_1, Y_1 and X_2, Y_2 are tested against the first different point pairwise of the fire perimeter, X_3', Y_3' and X_4', Y_4'. If an intersection is found, the coordinates X_1^{cross}, Y_1^{cross} of the new points are stored in a pointer array. FARSITE creates a pointer array for each thread or zone. If there is no intersection, the pair X_1, Y_1 and X_2, Y_2 are tested against the next different point pairwise X_3', Y_3' and X_5', Y_5'. When all the possible pairs points are checked, the next pairwise point, X_1, Y_1 and X_3, Y_3 is tested. This process is repeated until all the possible combinations are checked. At the end of the outer loop, all the intersections are unified in a single pointer array. The thread id determines the order of coordinates; the thread 0 writes first the coordinates of the intersections that it found, then the thread 1, etc. Finally, the new coordinates are introduced in the fire perimeter, and the points are reordered so that they appear in the sequence in which they would be encountered, starting from the first vertex.

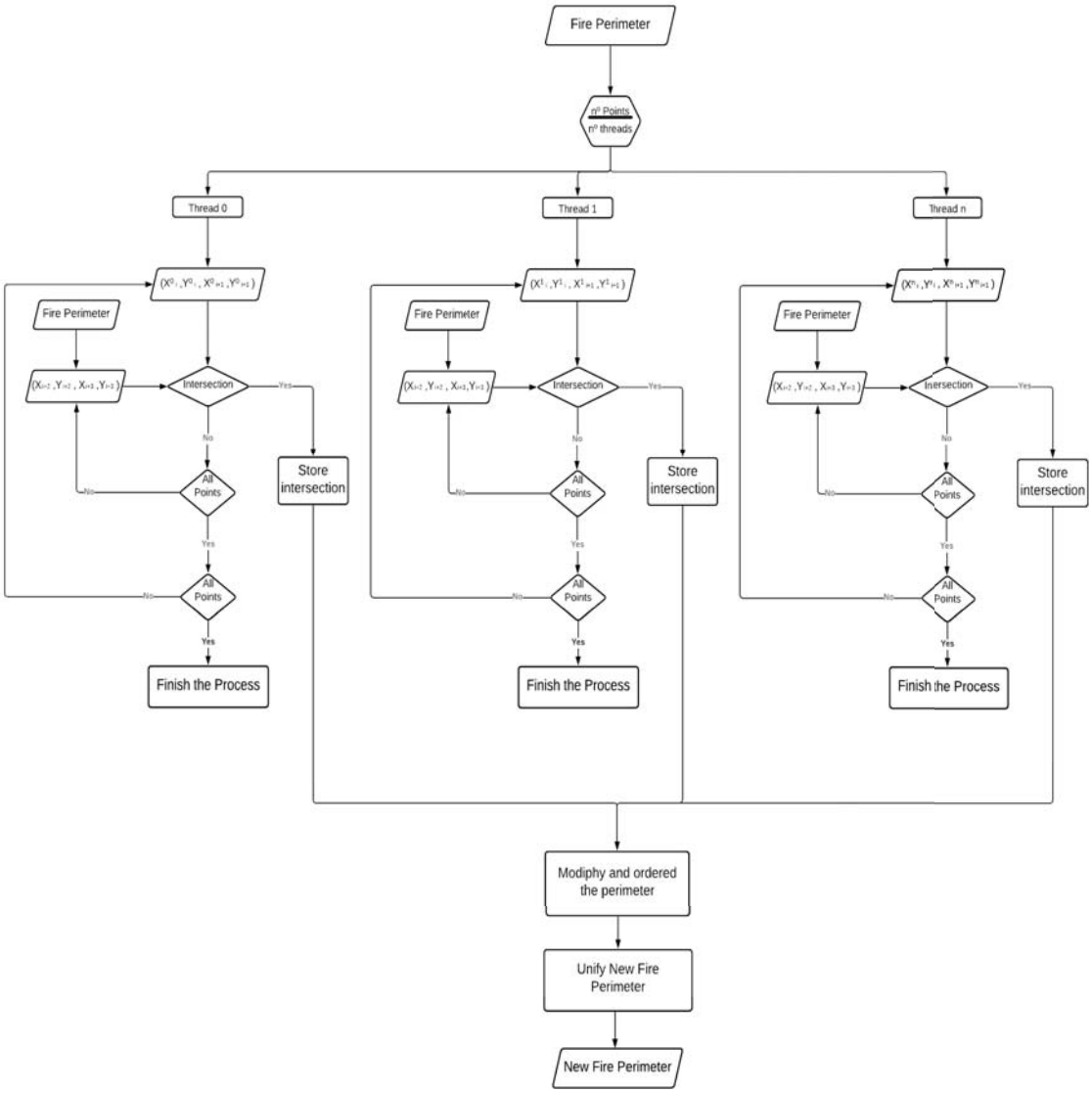


Figure 4.6: Fire Front Reconstruction Algorithm implemented in OpenMP.

4.2.2 GPU parallel Optimization

In order to reduce the execution time invested in the Fire Front Reconstruction, the original algorithm has been modified, implementing a fine-grain parallel implementation for exploiting the computational power of the GPUs. The workflow of the Fire Front Reconstruction process is illustrated in Figure 4.7. Because in the Fire Front Reconstruction, the memory requirements are fewer than in the Point

Propagation algorithm; consequently, the time invested in the data transfer from the *Host* to *Device* is shorter. However, it is more computationally expensive and consumes more execution time.

In the GPU fine-grain parallel implementation of the Fire Front Reconstruction algorithm, a thread is responsible for checking only a particular pairwise point, X_1, Y_1 and X_2, Y_2, against all other possible pairs of points, X_i', Y_i' and X_j', Y_j', from the fire front. Therefore, in the GPU implementation, a single spatial loop is needed. Because the GPU is throughput-oriented, thousands of pairs can be tested simultaneously. In this case, a single pointer vector is used to store the coordinates, X_1^{cross}, Y_1^{cross}, of the intersection points. As in the CPU implementations, The position in which the coordinates are written is controlled with a single variable. When an intersection is found, the value of this variable is incremented by 1. In the CPU implementations, this value is increased in serial. Therefore the pointer used to store the intersection coordinates is filling sequentially. However, the GPU implementation could have a competition between threads to read and write this variable, race conditions. To avoid this conflict between threads, we employ an atomic operation, [46]. This kind of operation is used to avoid race conditions. When an intersection is found, the atomic operation increases the value of the variable by 1. Then, this variable determines the position in which position the coordinates and the fire characteristics of the point (ROS, Flame Intensity, and Reaction Intensity) are stored. This is possible because the order in which the intersections are stored does not affect the final shape of the fire perimeter. If there is no intersection, the thread continues checking the next pairwise. There is an explicit thread synchronization at the end of the whole process to verify that all the possible pairs of points have been processed. Then the intersection pointer is copied from the *Device* to *Host*. Finally, the intersection coordinates and arrangements are entered into the fire perimeter and sorted in the CPU. As we can observe, the order in where the intersections are found does not affect the shape of the fire at the end of the process. The fine-grain GPU implementation reduce the complexity of the Fire Front Reconstruction from $O_{serial}(n^4)$ to $O_{GPU}(n^3)$.

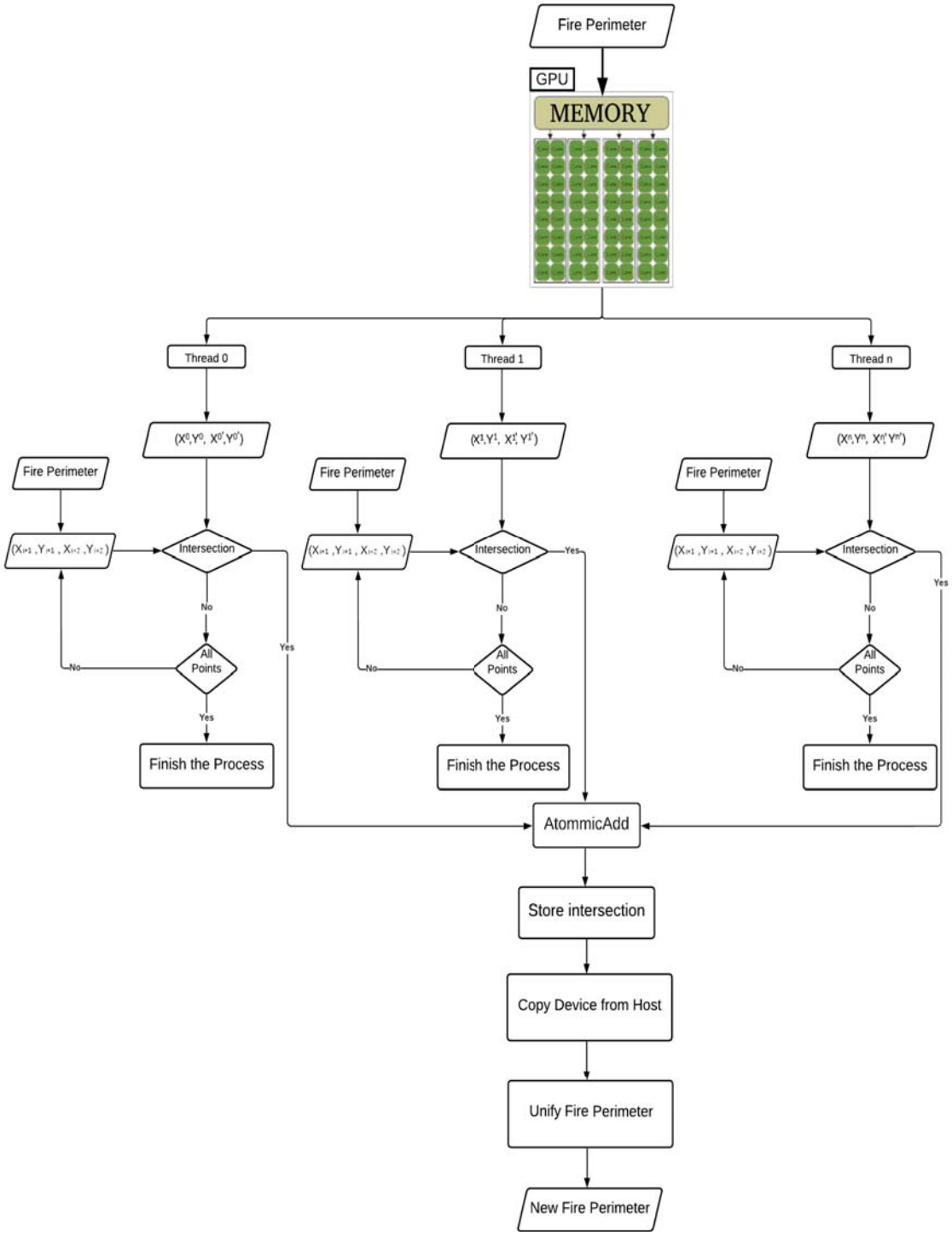


Figure 4.7: Fire Front Reconstruction Algorithm implemented in GPU.

4.3 Experimental Study and Results

All calculations described here were performed using a server system with a CPU and a single GPU card. Our execution platform has an Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz with eight cores and a GeForce RTX 2080 Ti with 4352 CUDA cores.

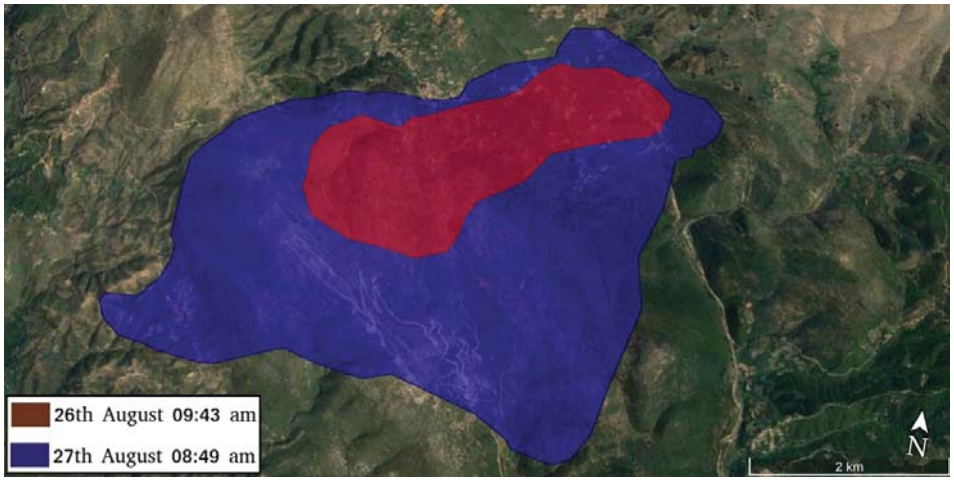


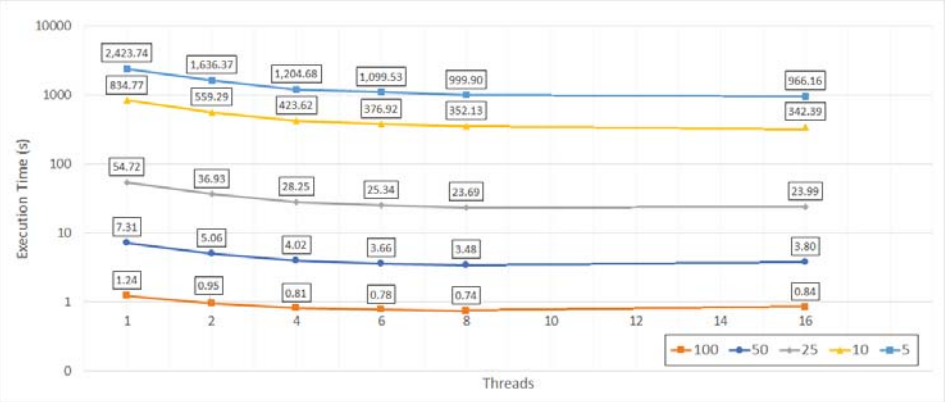
Figure 4.8: Digital Elevation map of Arkadia fire area. The *Red Perimeter* was used as initial perimeter (ignition Perimeter). The perimeter to be predicted is *Blue Perimeter*, [23].

As a study case, we selected a past fire that took place in Greece during the summer season of 2011 in the region of Arkadia fire belonging to the database of EFFIS (*European Forest Fire Information System*) [23]. The forest fire began on the 26th of August, and the total burnt area was 1.761 ha. In Figure 4.8, it can be seen the fire perimeters at three different time instants: t_0 (August 26th at 09:43am), t_1 (August 26th at 11:27am) and t_2 (August 27th at 08:49am).

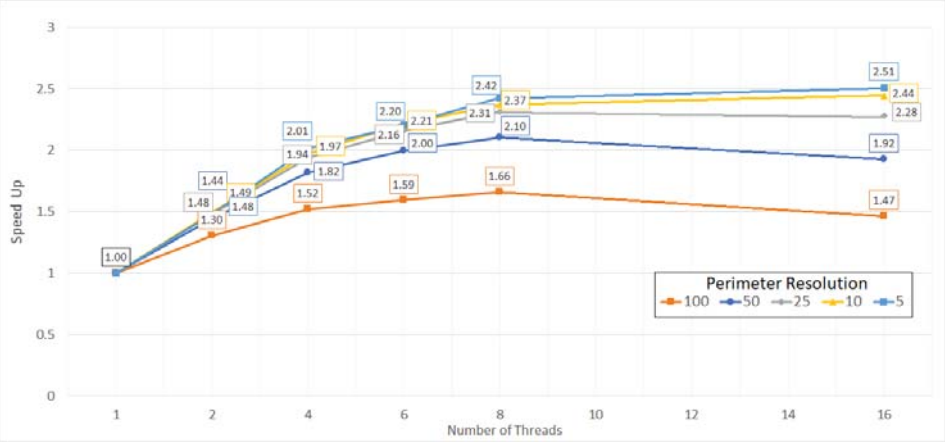
4.3.1 OpenMP Scalability of FARSITE

When we work with parallel implementations, one important aspect is determining the adequate amount of resources invested in executing an application. If the application shows some kind of dependency, performance will not improve if more resources are provided. As is well known, above certain resources employed, the

performance improvement is not significantly. For that reason, as a first step, we study the scalability of FARSITE when the Perimeter propagation and the Fire Front Reconstruction are implemented in OpenMP. The CPU used in this study has eight cores with multi-threading, which means we can execute FARSITE with 16 threads. Figure 4.9 shows the scalability of FARSITE with different *Perimeter Resolution*.



(a) Execution time of the Arkadia forest fire with different *Perimeter Resolution* and different number of threads. The Y-axis is in a log scale.



(b) OpenMP speed up of the Arkadia forest fire with different *Perimeter Resolution* and different number of threads.

Figure 4.9: Scalability of FARSITE with five different *Perimeter Resolution*.

Figure 4.9(a) displays the execution time for five different *Perimeter Resolu-*

tion. The Y-axis is on a log scale. As we can appreciate, the *Perimeter Resolution* has a direct impact on the time invested in performing a forest fire spread simulation. As we said, as low *Perimeter Resolution*, greater is the number of perimeter points; therefore, the execution time increases. It can be observed that for all *Perimeter Resolution*, the lower execution time is reached when we execute FARSITE with 8 threads. In Figure 4.9(b) we can see the speed up of the simulation for different threads and different *Perimeter Resolution*. It demonstrates that, over some point, the increment of resources does not significantly improve the performance of the FARSITE. During this study, we focus our effort to paralyze the Fire Front Propagation and the Fire Front Reconstruction algorithms, which represent around the 67% of the execution time for big fires, see Figure 2.2. If the number of perimeter points increases, this percentage could be higher.

It can be observed that, for all *Perimeter Resolution*, above 8 threads, the increment of threads does not significantly reduce the execution time. When the perimeter resolution is $100m$ and $50m$, the maximum performance improvement is obtained with 8 threads. For 10 and 5 meters, the maximum speed up is reached with 16 threads. However, the speed up difference between 8 and 16 threads is between 0.07 and 0.09 of speed up, which does not justify the increment of threads. In the following sections, our Fine-Grained Parallelization implementation in GPU against the implementation in OpenMP with 8 threads is compared.

4.3.2 Forest Fire Spread Simulator Performance Analysis

An essential aspect, when we compare the implementations of the forest fire spread simulator in GPU and OpenMP, is the final burned area obtained at the end of the simulation. Figure 4.10 shows the final fire perimeter obtained for both implementations. We can observe a small difference in both simulations.

We can observe that the OpenMP implementation tends to propagate a few more in the forest fire front. This slight difference is due to the different numerical methods implemented in GPU and OpenMP, [39]. However, a forest fire is a very complex system subject to a high number of variables. For that reason, we are more interested in simulating the general behaviour of forest fire. Therefore,

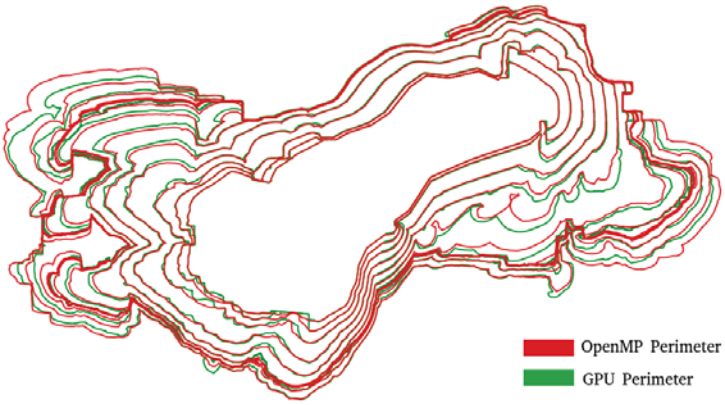
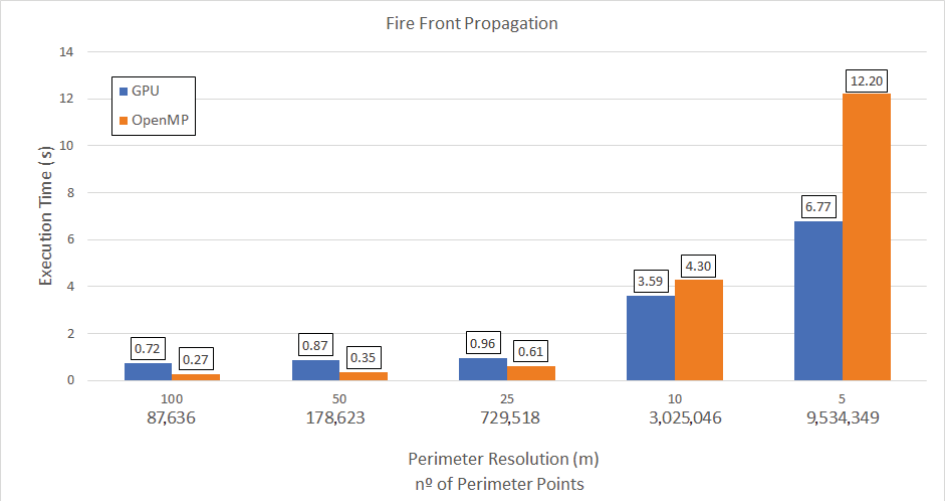


Figure 4.10: Comparative of the final fire perimeters simulated for the implementation of FARSITE in OpenMP (*Red*) and in GPU (*Green*).

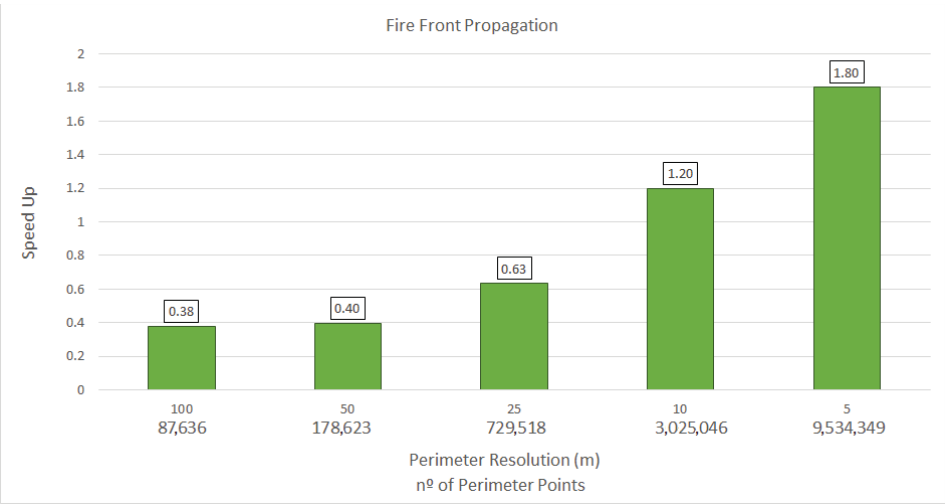
we can conclude that the difference between GPU and OpenMP implementations is unimportant. The main objective of this research is to accelerate the high-accuracy forest fire spread simulator FARSITE without excessively increasing the execution time. We wonder whether it is possible to carry out faster executions while keeping accurate simulations using GPUs accelerators. For that reason, we are going to compare the efficiency of the GPU and OpenMP applications proposed and compare the throughput of both implementations. We are interested in the evolution of the throughput in a simulation modifying the *Perimeter Resolution* and, therefore, the number of points to expand. For that, we use three different parameters: the *Execution Time*, the *Speed UP* and the processed number of *Points/Second* which is the number of points propagated per time unit.

Fire Front Propagation Performance Analysis

Because FARSITE has become broadly used for operational prediction of active wildfires, in this section, we focus on analyzing only the performance of the Fire Front Propagation algorithm. As we saw, the propagation of the fire front represents around the 7% execution time, see Figure 2.2. Nonetheless, because the propagation of a point is entirely independent of the propagation of the others, its parallelization is relatively easy. Figure 4.11 displays the performance of the Fire Front Propagation algorithm into the GPU and the OpenMP implementations.



(a) Execution time of Perimeter Propagation algorithm with different *Perimeter Resolution* into the GPU (*Blue*) and the OpenMP (*Orange*) implementations.



(b) Speed up of The GPU over the OpenMP implementation the Perimeter Propagation Algorithm with different *Perimeter Resolution*.

Figure 4.11: Perimeter Propagation Performance with five different *Perimeter Resolution*. Below the *Perimeter Resolution* the number of total perimeter point processed in the GPU are shown.

Figure 4.11(a) shows the execution time invested in both implementations to perform the evolution of the fire front in the whole simulation. To better understand the computational work for each *Perimeter Resolution*. In order to un-

derstand the order of magnitude of the computational work, the total number of perimeter points processed in the GPU are shown under the *Perimeter Resolution*. We have to bear in mind that in the OpenMP implementation tends to propagate a few more the fire front, see Figure 4.10, so the number of perimeter points to be processed is higher, but this difference is not relevant, around thousands of points. As we can see, for high *Perimeter Resolution* (100, 50, and 25 meters), a low number of perimeter points, the OpenMP implementation invests less time into propagating the perimeter of the fire than the GPU implementation. For *Perimeter Resolution* above 10 meters, the execution time of the forest fire front propagation algorithm in GPU is higher than the time invested in the OpenMP implementation. All GPU has a start-up cost caused by its initialization and the data transfer time from CPU memory to GPU memory. The computational work has to compensate for the start-up time. In our particular case, it means that under a certain number of perimeter points, this start-up time is not compensated, [14]. For example, when the *Perimeter Resolution* is equal to 25 meters, the number of perimeter points, $729,518$ points, is not enough to compensate for the start-up and the copy time. For that reason, in this case, the OpenMP implementation is the most efficient. This tendency is confirmed for lower *Perimeter Resolution*. For *Perimeter Resolution* equal to 10 meters, the number of perimeter points, $3,025,046$ points, is large enough so that the computational work can compensate for the start-up and the copy time. For that reason, the OpenMP implementation is the less efficient and, therefore, the slowest one over this number of points. When the *Perimeter Resolution* is 5 meters, $9,534,349$ perimeter points, the GPU implementation allows a reduction of 44.5% of the execution time, which implies that the efficiency of the GPU implementation increases while the distance between perimeter points is reduced.

Figure 4.11(b) displays the computed speed up when the propagation of the fire front is done in the GPU. As it can be expect, when the *Perimeter Resolution* is 100, 50 or 25 meters ($87,636$, $178,623$ and $729,518$ perimeter points respectively) the speed up is under 1. This indicates that, for these *Perimeter Resolution* values, the OpenMP parallelization is the faster one. In the case of the Fire Front Propagation, when the *Perimeter Resolution* is equal to 5 meters, $9,534,349$

perimeter points, the maximum speed up, 1.80, is obtained. As it can be seen, the speed up increases as the number of perimeter points rises, or *Perimeter Resolution* decreases. This is due to the GPUs have evolved on the line of throughput-oriented processor architecture. This is due to the fact that GPUs have evolved on the line of throughput-oriented processor architecture.

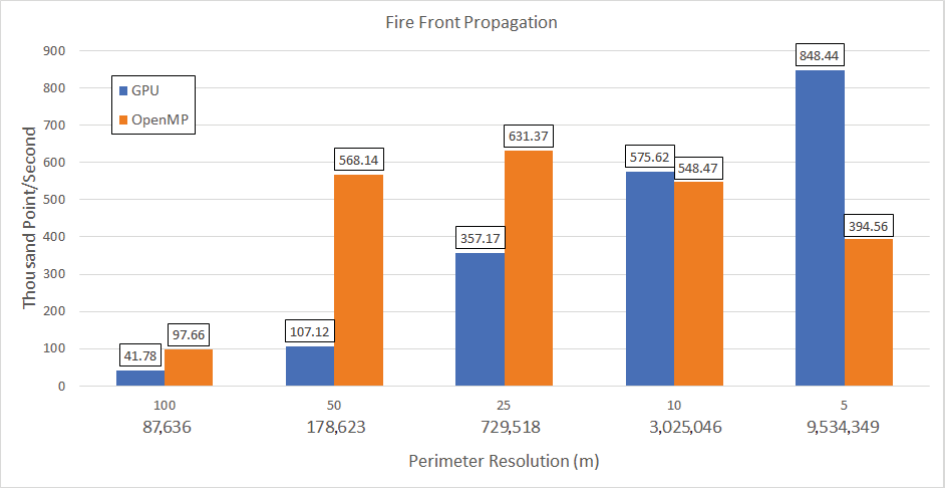


Figure 4.12: Points per second of the Perimeter Propagation algorithm depending on the *Perimeter Resolution* for the GPU (*Blue*) and OpenMP (*Orange*) implementations. Below the *Perimeter Resolution* the number of total perimeter point processed are shown.

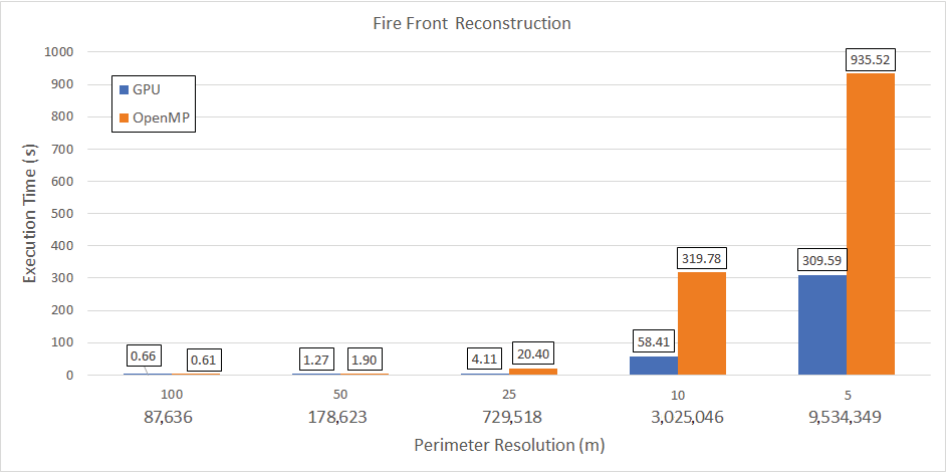
The OpenMP implementation tends to propagate a few more in the front of the forest fire. Consequently, the OpenMP parallelization has to propagate a few more perimeter points, penalizing its execution time. In order to compare the performance of both implementations, the number of perimeter points per second is used to normalize the computational work. It represents the number of points that evolved in a time unit. Figure 4.12 presents the number of thousand points per second for GPU and OpenMP Fire Front Propagation implementations. For *Perimeter Resolution* equal to 100, 50 and 25 meters, 87.636, 178, 623 and 729, 518 perimeter points, the number of points processed by the OpenMP implementation is higher than the number of points evolved by the GPU implementation. But, when *Perimeter Resolution* is 50 and 25 meters, the OpenMP implementation expands much more perimeter points per second than the GPU implementation. As

we saw, the OpenMP implementation tends to spread a few more the perimeter of the fire, see Figure 4.10, than GPU implementation. For that reason, the number of the *points/Second* is bigger in the OpenMP implementation. The number of perimeter points of the OpenMP implementation, with a *Perimeter Resolution* equal to 5 meters, is $10,082,192$ points, $547,843$ more than the GPU implementation, and decreases as the *Perimeter Resolution* diminish. When the *Perimeter Resolution* is 10 meters, both implementations evolve a comparable number of points per second, but the GPU implementation is the fastest. Nonetheless, for *Perimeter Resolution* of 5 meters, the GPU propagates more than the double number of points per second than the OpenMP implementation. These results highlight that implementing a fine-grain parallelization in the GPU has a better performance than the OpenMP implementation for high accuracy simulations.

Fire Front Reconstruction Performance Analysis

In the following section, the performance of the Fire Front Reconstruction algorithm is analyzed separately. In FARSITE, the Fire Front Reconstruction algorithm represents around 60% of execution time, see Figure 2.2. This is the part of the code with the highest computational cost of the whole process. Figure 4.13 summarizes the performance of the Fire Front Reconstruction algorithm parallelizations implemented in the GPU and OpenMP. In Figure 4.13(a) the execution time invested in reconstructing the front of the fire for both parallel implementations is presented. At first sight, we see that the Fire Front Reconstruction has a higher computational cost than the Perimeter Propagation algorithm. It is possible to observe that when we work with low resolution, *Perimeter Resolution* equal to 100, the OpenMP implementation is faster than the GPU implementation. This is because with this *Perimeter Resolution* the computational work is not enough to exploit the computational power of the GPU, and the star-up and copy times are not compensated. However, the time invested by both applications is less than a second. For lower values of *Perimeter Resolution*, the GPU parallelization is more efficient than the OpenMP implementation. For simulations with high accuracy, *Perimeter Resolution* equal to 5 meters, the execution time invested to the OpenMP implementation is $935,52$ seconds while the GPU consumes 309.59

seconds, more than 10 minutes faster than the OpenMP parallelization.



(a) Execution time of Fire Front Reconstruction algorithm with different *Perimeter Resolution* into the GPU (*Blue*) and the OpenMP (*Orange*) implementations.

(b) peed Up of The GPU over the OpenMP implementation the Fire Front Reconstruction Algorithm with different *Perimeter Resolution*.

Figure 4.13: Fire Front Reconstruction Performance with five different *Perimeter Resolution*. Below the *Perimeter Resolution* the number of total perimeter point processed in the GPU are shown.

Figure 4.13(b) displays the speed up when the Reconstruction algorithm is executed in the GPU. We can see that for the lowest *Perimeter Resolution*, 5 meters,

the computed speed up is 3.02. However, this is not the maximum speed up computed. When the *Perimeter Resolution* is equal to 25 and 10 meters, the speed up is higher, 4.96 and 5.58 respectively. This outcome is because the proposed GPU parallelization reaches maximum bandwidth, so for *Perimeter Resolution* of 5 meters, we skip from a computational-bound problem to a memory-bound problem has been reached. For this reason, the speed up decreases for high accuracy simulations, more than $9,500,000$ perimeter points. Nonetheless, the GPU is faster than the OpenMP parallelization.

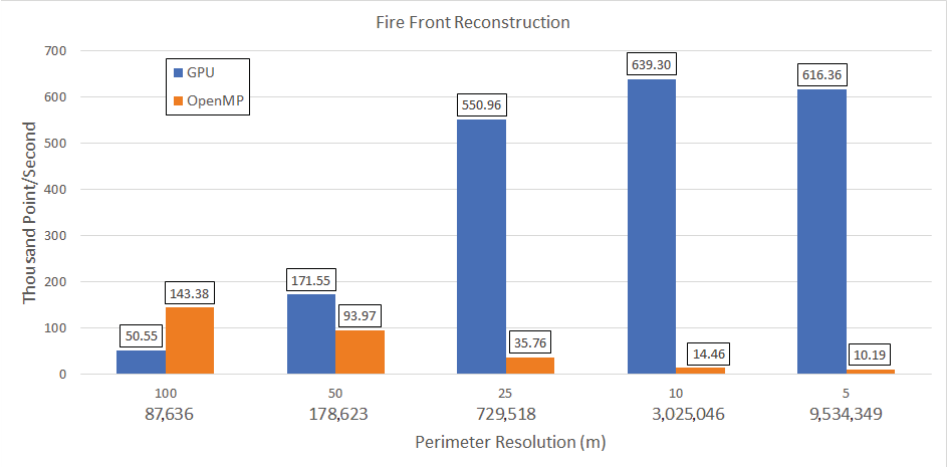


Figure 4.14: Points per second of the Fire Front Reconstruction algorithm depending on the *Perimeter Resolution* for the GPU (*Blue*) and OpenMP (*Orange*) implementations.

As in the previous section, to compare the performance of both implementations, the computational work has been normalized. The number of perimeter points per second computed is compared. Figure 4.14 shows the number of thousand points per second processed by the Fire Front Reconstruction algorithm for both parallelizations. When the *Perimeter Resolution* is equal to 100, $\sim 87,636$ perimeter points, the OpenMP parallelization processes more points per second than the GPU implementation. This means that the OpenMP implementation is most efficient. However, for lower *Perimeter Resolution*, or a higher number of perimeter points, the GPU implementation treats more points per second. When the *Perimeter Resolution* is 5 meters, the difference between both parallelizations is not significant; the GPU treats $77,6$ than the OpenMP application. Neverthe-

less, the difference is notably when the number of perimeter points raises. This is a clear indicator that the GPU parallelization is the most efficient when we work with high simulation resolution, low *Perimeter Resolution*. In this case, the maximum of points per second, 639.30 *points/sec* is reached at *Perimeter Resolution* of 10 meters, or $3,025,024$ perimeter points. In Addition, we can see which *Perimeter Resolution* equal to 5 meters, the number of *point/second* does not increase; on the contrary, it is lower. This is due because above $\sim 3,000,000$ perimeter points, the GPU is saturated, and the execution time invested into the copy data between the *Device* and *Host* augment as the number of points increases. Therefore, in the case of *Perimeter Resolution* under 10 meters, the execution of the perimeter reconstruction algorithm is dominated by memory transfers, and global performance is not improved. The obtained result highlight that, for low *Perimeter Resolution*, the execution of the Fire Front Reconstruction algorithm is more efficient in the GPU than in the OpenMP implementation.

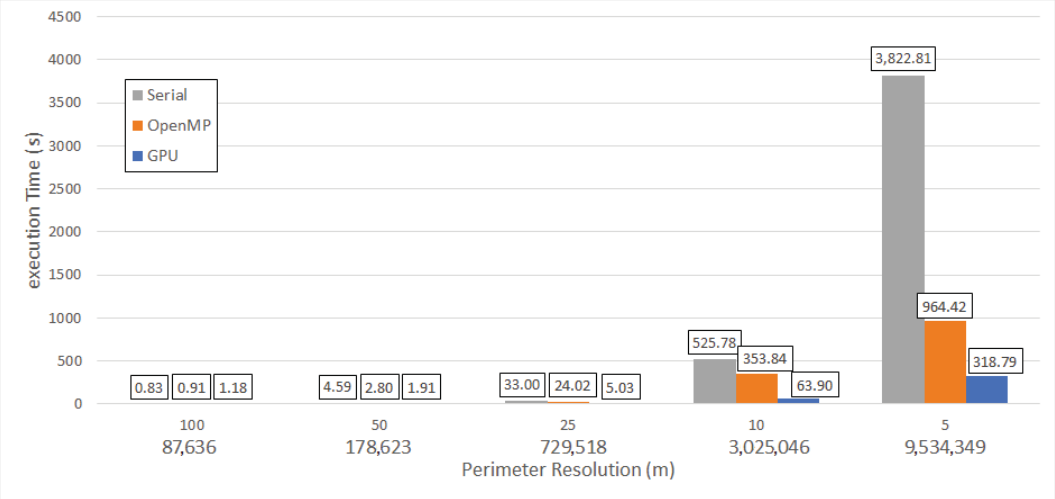


Figure 4.15: Execution time of the Arkadia's fire simulation for FARSITE in Serial (*Grey*), in GPU (*Blue*) and in OpenMP (*Orange*) for different *Perimeter Resolution*. Below the *Perimeter Resolution* the number of total perimeter point processed are shown.

FARSITE Performance Analysis

In order to estimate the impact of both parallelizations, the performance of a whole FARSITE simulation is analyzed. Figure 4.15 displays the execution time when FARSITE is executed in serial, in GPU, and in OpenMP. We can see that for *Perimeter Resolution* equal to 100 meters, 87,636 perimeter points; the GPU implementation is the one that consumes more time, around 0.3 seconds slower than the OpenMP application. In this case, the faster implementation is the Serial. For lower *Perimeter Resolution* values, both parallel implementations are faster than the serial simulation. Furthermore, in all these cases, GPU parallelization is the implementation with the best performance. For high accuracy simulations, *Perimeter Resolution* equal to 5 meters, or 9,534,349 perimeter points, the serial implementation spends more than an hour to perform the forest fire spread simulation. In the case of the OpenMP implementation, this time is reduced to around 16 minutes, 964.42 seconds. However, as we said, the most efficient application is GPU parallelization. With this implementation, the execution time is reduced to approximately 5 minutes, 318.79 seconds, representing a reduction of the 67% of the execution time of the OpenMP execution, and a reduction of 91% of the Serial simulation. Table 4.1 summarizes the speed up of parallel both implementations depending on the *Perimeter Resolution*. As we can observe, as the *Perimeter Resolution* decreases or the number of perimeter points is raised, the parallel applications become more efficient. The maximum speed up is obtained with a *Perimeter Resolution* equal to 5 meters, with a value of 11.99 for the GPU implementation and 3.96 for the OpenMP implementation. As we can observe, the GPU parallelization allows a notable execution time reduction, indicating that it is the implementation with the best performance.

Finally, the performance of both parallel applications is compared. Figure 4.16 shows the speed up of the GPU implementation over OpenMP. As we saw, for simulations with low accuracy, *Perimeter Resolution* equal to 100 meters, the OpenMP implementation is faster than the GPU one. However, this is the only situation in which the OpenMP parallelization has the best performance as the *Perimeter Resolution* is reduced, the speed up of the GPU simulation increases. Nonetheless, we can see that the maximum speed up is reached for *Perimeter*

Table 4.1: Speed up computed for the GPU and OpenMP implementation with different *Perimeter Resolution*.

Perimeter Resolution	OpenMP	GPU
100	0.91	0.70
50	1.21	2.40
25	1.37	6.56
10	1.49	8.23
5	3.96	11.99

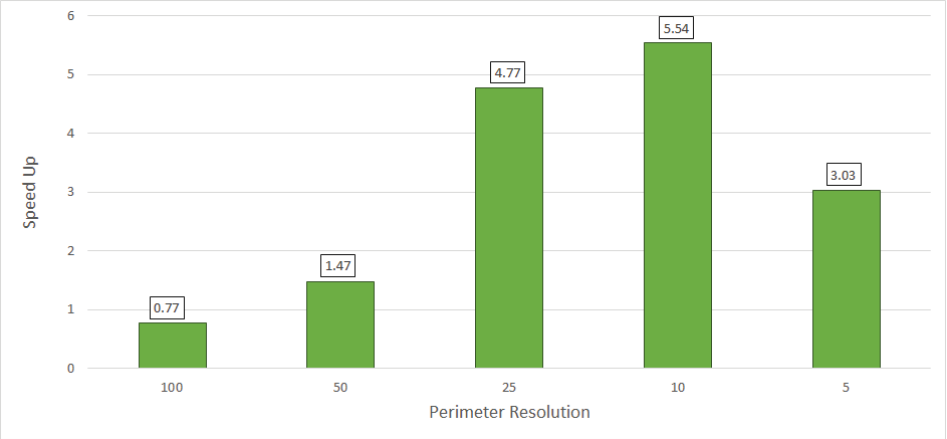


Figure 4.16: Computed speed up when FARSITE is executed in the GPU and OpenMP.

Resolution equal to 10 meters, 5.54. For *Perimeter Resolution* under this value, 5 meters, the speed up recede to 3.03. This is because when the *Perimeter Resolution* correspond to 5 meters, the number of perimeter points, more than 9.5 million points, saturates the bandwidth of the GPU device. As a consequence, the weight of the copy data between the GPU and CPU memory increases, penalizing the execution time for low *Perimeter Resolution*.

In order to understand the reason because the speed up decreases for *Perimeter Resolution* below 10 meters, we analyze the execution time invested in the copy data between memories of the GPU parallelization of the forest fire spread simulator using the NVIDIA visual profiling, [47]. The obtained results reveal that the time invested in the copy data from the GPU memory to the CPU memory increases as the *Perimeter Resolution* diminishes. Figure 4.17 exposes the execution

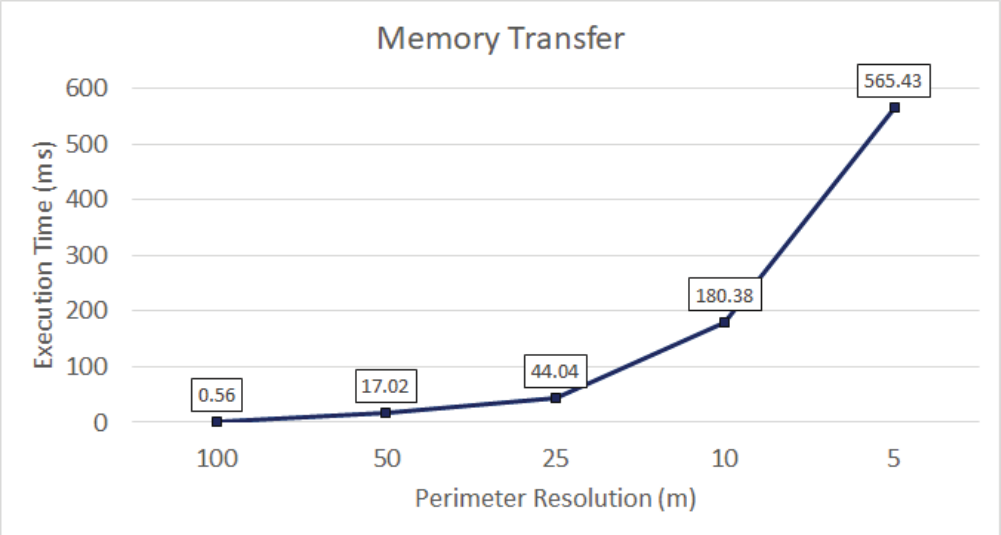


Figure 4.17: Execution time invested in the data transfer between memories in the GPU parallelization for different *Perimeter Resolution*.

time consumed by the movement of data between memories in milliseconds. As it can be seen, this copy time increases exponentially when the *Perimeter Resolution* is reduced. On the one hand, for *Perimeter Resolution* equal to 100 meters, the copy time is 0.56 milliseconds. On the other hand, the time invested in the data transfer is 564.43 milliseconds for a *Perimeter Resolution* equal to 5. As we saw, when the *Perimeter Resolution* decreases, the number of perimeter points increases significantly, therefore the time invested in the copy of the perimeter coordinates from the GPU memory to the Host memory and the time invested in copying the intersection vector from Host to Device increases notably.

In Figure 4.18 are shown the transfer data work normalized. We compute the copy time invested in each perimeter point, *Points/millisecond*. In this case, the measurement of the *Perimeter Resolution* equal to 100 meters has been discarded because its short execution time distorts the results. As it is shown for *Perimeter Resolution* of 25, 10, and 5 meters, the number of perimeter points per milliseconds does not rise significantly; even the number of perimeter points is more than three times higher. The obtained results highlight that for a *Perimeter resolution* equal of 25 meters, the number of perimeter points is high enough to saturate the bandwidth of the GPU. Therefore, over this number of perimeter points copied

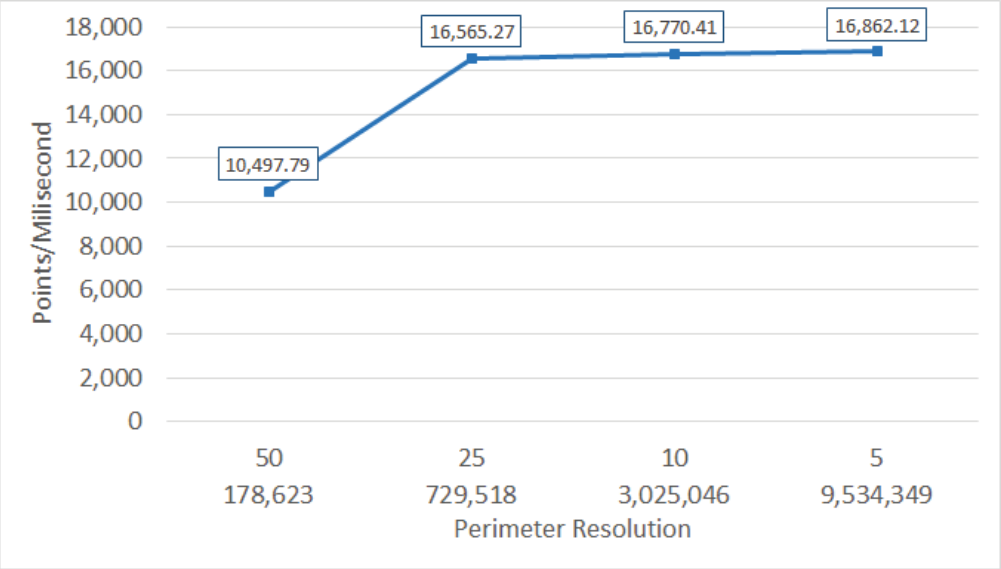


Figure 4.18: Number of points per millisecond transferred between memories in the GPU parallelization for different *Perimeter Resolution*.

does not increase. As a consequence, the execution time of the simulation is penalized. However, the GPU implementation is the most efficient execution when we perform simulations with low *Perimeter Resolution*.

In the future, we will deeply analyze the bottleneck of our GPU parallelization to overcome the current restrictions. The best way to overcome these restrictions is to utilize an alternative Fire Front reconstruction algorithm with low complexity.

Chapter 5

Edge Computing

The uncertainty of the input data required by wildfire simulators could be a relevant drawback to accurately predicting the forest fire evolution in a short period. As we saw, different strategies have been used to reduce this uncertainty. However, these studies are based on adjusting the input data better to reproduce the behaviour of the real wildfire. Nevertheless, the acquisition of the input data is not improved. In the last decades, many experiments demonstrate that the application of Unmanned Aerial Vehicles (UAVs) to collect real-time data, such as meteorology, vegetation status, or fire evolution, in the same area where the fire is taking place can significantly reduces this input data uncertainty, [2], [1], [41]. The problem arises when we assume that the location where the fire is taking place has low connectivity. In that case, the data cannot be sent, so it is of paramount importance to have a platform to perform the forest fire spread simulation *in situ* using the gathered information. The new forecasting platforms for fighting natural disasters such as extreme forest fire events are swarms of UVAs with low energy consumption computational resources on board, including a wide range of embedded sensors to measure almost everything in real-time. The resulting embedded system could exploit the edge computing paradigm, distributing the forecasting system into mobiles elements with computing capacities that couldprovide a more realistic view of the input data parametrs and dynamic conditions of the situation. In addition, we saw that when the fire takes place in complex terrain, high reso-

lution, low *Perimeter Resolution*, is necessary to obtain reliable simulations. The use of small *Perimeter Resolution* makes the spreading fire sensitive to small-scale variations in spatial variables. At this point, we have to distinguish between two different types of forest fire spread simulations: *Mid-Term Simulation* and the *Short-term Simulation*. In the *Mid-Term Simulation* the forest fire spread simulator simulates 24 hours or more of the fire evolution. To accomplish the time response requirements, a few hours, the utilization of High-Performance Computing becomes imperative. On the other side, the *Short-term Simulation* consists of simulating less than 5 hours of fire spread. The firefighters can use this information of the evolution of the near future fire propagation to optimize the resources and strategy promptly. In this sense, the new parallelization of GPU-FARSITE presented in this work, see Chapter 4, allows the possibility to exploit the maximum capabilities of the low consumption embedded systems. Embedded systems are wholly integrated systems on a module, including CPU, GPU, memory, power management, and high-speed interfaces. They are the ideal device for industrial automation, healthcare, transportation, aerospace, and defense. The fast-growing capabilities of embedded systems make them the ideal tool to edge devices. In order to overcome the energy restriction, the embedded systems with low consumption GPUs are the response of some manufacturers to the edge computing requirements of the new century, which principal are three: it has to work in a low connectivity environment, has low execution times, and be very energy efficient. In the particular case of forest fire, we examine edge computing as a tool to explore different fire scenarios and h parameters, like wind change, to take decisions in the field, like concentrate fire prevention in predicted areas. The objective is to advance the data calculations of complex models, like wildfire, in not so powerful close-to-emergency distributed environments.

As we highlight, the new GPU-FARSITE parallelization significantly reduces the execution time needed to simulate the forest fire evolution with high resolution. We will see that the utilization of embedded systems combined with our GPU-FARSITE simulation permits the near real-time time simulation of the *Short-Term Simulation*.

Figure 5.1: Nvidia Jetson AGX Xavier developer kit.

In this study, we focus on analyzing the capability of low consumption embedded systems to see if they can reach the requirements of execution time and accuracy in the simulations of the fire spread for being used in real scenarios. During this work, we analys the possibilities of the low consumption GPU NVIDIA Jetson AGX Xavier developer kit, (*Xavier*), see Figure 5.1. It is specially designed as an AI computer for autonomous machines, like robots, drones, and other autonomous machines, delivering the performance of a GPU workstation in an embedded module under 30W, [36]. In order to achieve this objective, the GPU-FARSITE execution performance in the *Xavier* platform is analysed. As preliminary study, the efficiency of the *Xavier* is compared against a desktop GPU, a GeForce RTX 2080 Ti *(GeForce 2080)*. Table 5.1 gathers the platform's hardware specs as declared by the manufacturer. It shows that *Xavier* has 8.5 times less computational capacity and 8.3 less energy needs than a desktop system. For that reason, we want to study if it is possible to use such a system as our base resource for an edge computing solution.

The Jetson AGX Xavier has the possibility to work with different *Energy Modes* defined by a set of parameters that effectively characterizes the performance for a given power envelope. *Xavier* GPU covers a wide range of performance and power requirements, [37]. Balancing the performance and power requirements is essential to obtaining optimal performance in a particular case. To study the optimal configuration of the *Xavier*, we took advantage of its flexibility to modify

Table 5.1: Hardware specifications of the experimentation platforms.

	Jetson AGX Xavier	GeForce RTX 2080 Ti
CPU	ARM v8.2 64-bit	Intel(R) Core(TM) i9-9900K
CPU Cores	8	8
GPU Architecture	Volta	Turing
CUDA Cores	512	4,352
Frequency (Mhz)	670	1,545
Bandwitdh (GB/s)	137	616
TDP (W)	15 or 30	250

the energy Mode of the *Xavier* and compare their performance for our particular problem. To characterize the optimal balance between the energy consumption and the computational power, a real fire has been used to compare our GPU-FARSITE application with different power Modes.

Mode Name	EDP	10W	15W	30W	30W	30W	30W
	MAXN	MODE_10W	MODE_15W	MODE_30W_ALL	MODE_30W_6CORE	MODE_30W_4CORE	MODE_30W_2CORE
Power Budget	n/a	10W	15W	30W	30W	30W	30W
Mode ID	0	1	2	3	4	5	6
Number of Online CPUs	8	2	4	8	6	4	2
CPU Maximal Frequency (MHz)	2265.6	1200	1200	1200	1450	1780	2100
GPU TPC	4	2	4	4	4	4	4
CPU Maximal Frequency (MHz)	1377	520	670	900	900	900	900
DLA Cores	2	2	2	2	2	2	2
DLA Maximal Frequency (MHz)	1395.2	550	750	1050	1050	1050	1050
Vision Accelerator (VA) cores	2	0	1	1	1	1	1
VA Maximal Frequency (MHz)	1088	0	550	760	760	760	760
Memory	2133	1066	1333	1600	1600	1600	1600

Figure 5.2: *Jetson AGX Xavier* power configuration Mode characteristics. The green colour indicates the default power Mode characteristics, [37].

For studying the performance and the energy requirements of each GPU and the power Modes of *Xavier*, we use three different parameters:

- *Execution Time*: It represents the total time to perform a simulation, in s.

- *Points per second*: Which represents the number of processed points per second, in $\#Points/s$.

- *Energy Capability*: It is the number of computed points per work unit, (see Equation 5.1). It represents the number of points that can be propagated per work unit. The higher it is, the more energetically efficient it is.

$$E.C. = \left(\frac{Number\ Points}{Time \cdot TDP} \right) \cdot Execution\ Time \tag{5.1}$$

  where the TDP is the *Thermal Design Power*, which is defined by the manufacturers.

5.1 Experimental Study and Results

In order to analyse the viability of the utilization of the low consumption embedded systems in a real forest fire, we analyse the performance of the GPU-FARSITE parallelization, see Chapter 4 in two different scenarios. First of all, as a first approximation of the problem, a synthetic fire was used. In this particular case, it has been considered a flat terrain, with homogeneous vegetation and constant wind speed and wind direction during the whole simulation. Under these conditions, the crosswalks between points are minimised. In this experiment we compare the performance of the *Xavier* default Mode (Mode 2), see Figure 5.2, against a desktop GPU *GeForce 2080*.

As we observed, the *Xavier* is an embedded system with the capacity to work with a wide range of energy configurations. Because of this adaptability capacity and to taking account all the possible computational capabilities and the energy modes characteristics, in the second experiment, a real fire is simulated to find the optimal balancing between energy consumption and computational power. Therefore, the performance of different energy Modes of the *Xavier* is tested and

analysed.

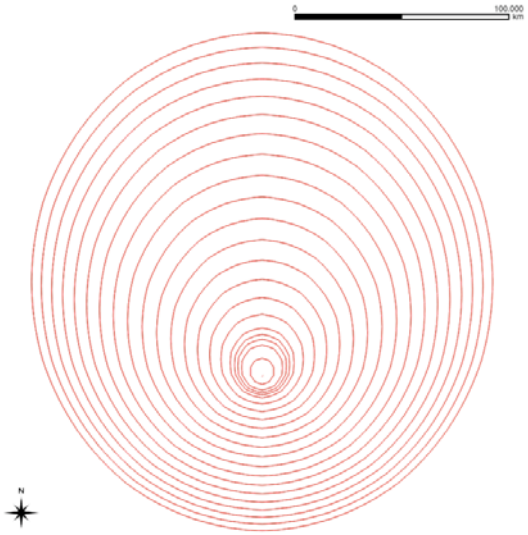


Figure 5.3: Evolution of the synthetic fire front on flat terrain, with homogeneous vegetation and constant wind speed and wind direction. A single ignition point is used.

5.1.1 Preliminary Study (Synthetic fire)

To analyze under which condition the *Xavier* can accomplish with the computational requirements and the time execution limitation, we compare the performance of its low consumption GPU and the *GeForce 2080* with various simulations and evaluate a list of evolution times. The tested propagation times were *5 hours*, *10 hours*, *24 hours*, and *48 hours*. The *Perimeter Resolution* was fixed to 100, 50, 25, 10 and 5 meters. Finally, we also carried out 5 repetitions for each particular simulation. The times presented are obtained from the average of the five times obtained of each simulation with a maximum standard deviation under $\sigma = 0.05$. We used a synthetic fire in flat terrain, with homogeneous fuel and a constant wind speed and wind direction as a study case. The evolution of the fire front is shown in Figure 5.3.

Table 5.2 summarises the obtained execution time for the different simulations performed in both GPUs. As we can see, in most of the situations, the

Table 5.2: Execution time invested for both GPU to perform the simulations with *Perimeter Resolution* of 100, 50, 25, 10 and 5 meters for four different propagation time, 5, 10, 24 and 48 hours.

Perimeter	5 hours		10 hours		24 hours		48 hours	
Resolution	*Xavier*	*GeForce*	*Xavier*	*GeForce*	*Xavier*	*GeForce*	*Xavier*	*GeForce*
100	0.528	0.826	0.629	0.854	1.062	0.941	1.883	1.066
50	0.531	0.843	0.649	0.856	1.229	0.959	1.265	1.205
25	0.546	0.827	0.729	0.860	1.736	1.062	23.564	2.555
10	0.749	0.881	1.608	1.027	8.766	1.936	80.021	12.860
5	1.645	1.050	5.600	1.572	42.938	4.927	121.389	29.392

GeForce 2080 performs the simulation faster. As we said, the *GeForce 2080* has 8.5 times more computational resources available than the *Xavier* see Table 5.1, consequently, the expected result should show that the *GeForce 2080* is around 8.5 time faster than the *Xavier*. For a *Perimeter Resolution* of 10 and 5 meters and a fire propagation of 48 and 24 hours, the *GeForce 2080* is 9.22 and 8.71 times, respectively faster than the *Xavier*. However, for all the other scenario, the *Xavier* is not 8.5 time slower than *GeForce 2080*. This is a crucial point because, in some situations, the fastest device is the *Xavier*. This happens when the execution time is under one second. For example, when the *Perimeter Resolution* is equal to 100 meters, and the fire propagation equal to 5 and 10 hours, the *Xavier* is around 0.3 seconds faster than the *GeForce 2080*. Nevertheless, *GeForce 2080* consumes less execution time for the *Mid-Term Simulation*.

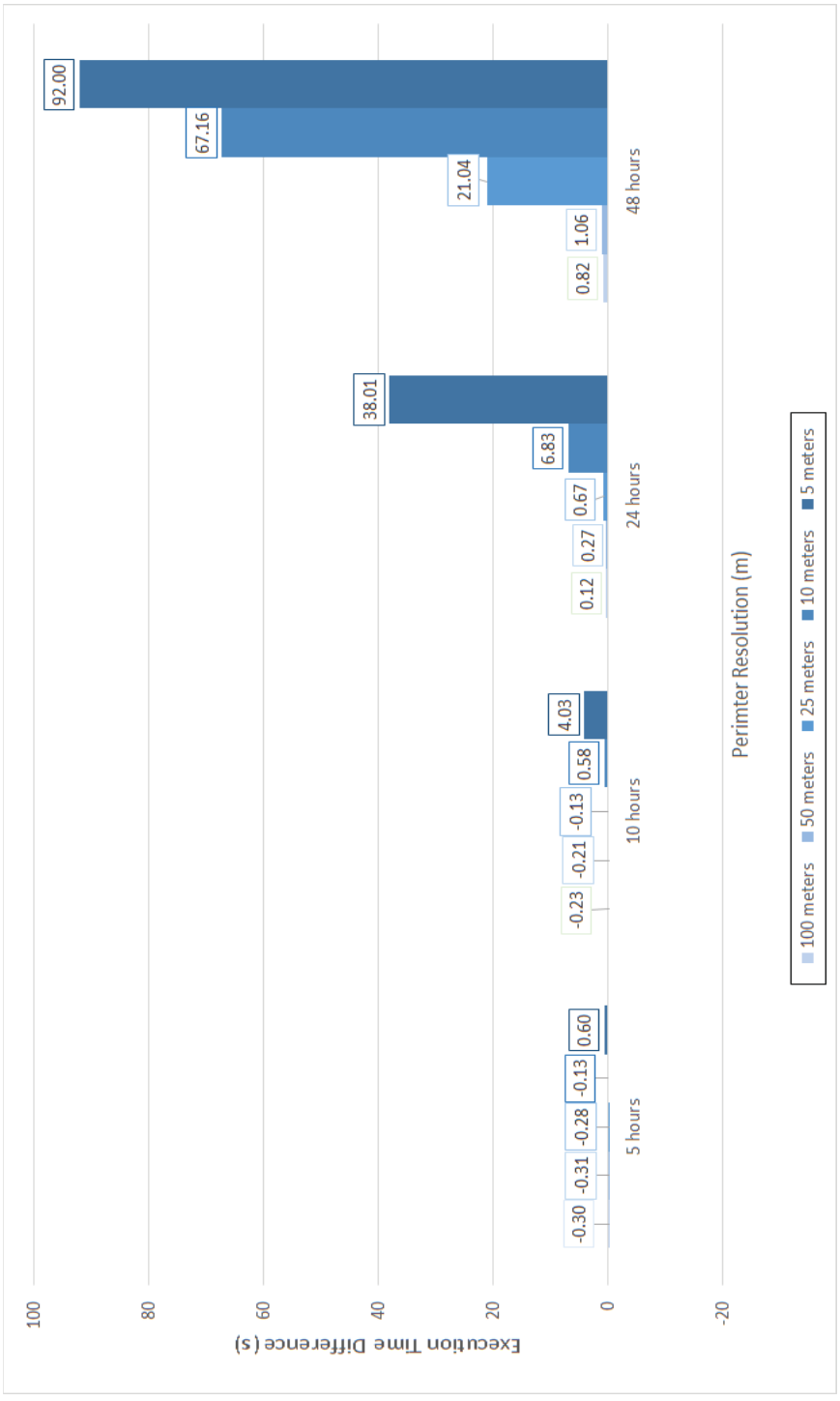


Figure 5.4: *Execution time* difference depending on the different *Perimeter Resolution* for the two different GPU platforms.

Figure 5.4 displays the difference of execution time for both GPUs per different *Perimeter Resolution* and propagation times. For for the *Short-Term Simulation*, 5 hours, *Perimeter Resolution* equal to 5 meters, the *Xavier* is only 0.6 seconds slower than the *GeForce 2080*. The execution time difference increases as the fire propagation and the *Perimeter Resolution* increases. The maximum difference between both executions is reached with 48 hours of fire evolution and *Perimeter Resolution* of 5 meters. However, we can see that exists some situations in which the device with the better performance is the *Xavier*, although the *GeForce 2080* has more computational resources. This is because, for short propagation time or low accuracy, the computational work does not compensate the initialization time. In previous works, we observed that the initialization time of the *GeForce 2080* is around 0.8 seconds, which means that only to start up the GPU, you need this time before starting to process any data. For that reason, in the case where the whole simulation takes less than a second, the *Xavier* is the faster device. This overhead is less relevant as the number of propagation hours grows, or the *Perimeter resolution* decreases. However, when we work with high simulation accuracy, *Perimeter Resolution* equal to 5 meters, the *GeForce 2080* is the faster device.

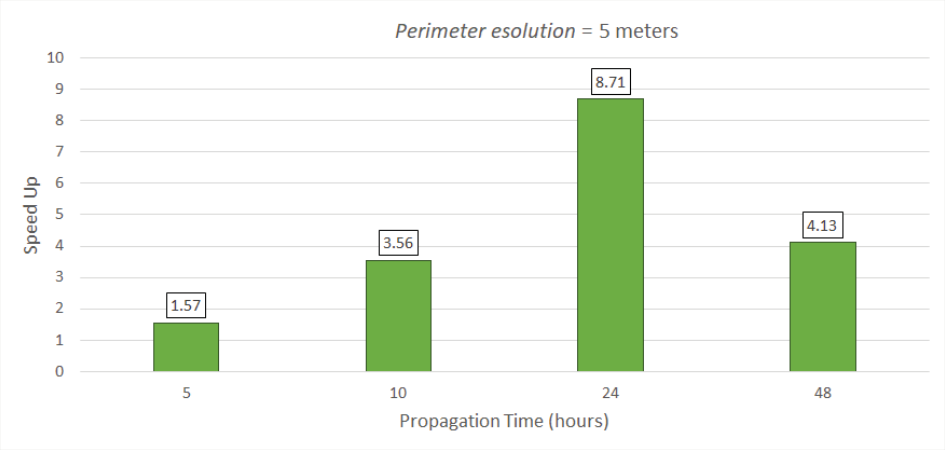


Figure 5.5: Speed up of the *GeForce 2080* Ti in front of the *Xavier* when the *Perimeter Resolution* is 5 meters for different propagation time.

Because we are interested in performing the simulation of the forest fire spread with high accuracy, for that reason, we analyse in deep detail the performance

of both devices when the *Perimeter Resolution* is equal to 5 meters. Figure 5.5 represents the speed up of the *GeForce 2080* in front of the *Xavier* for the different propagation times. As we repeated, the *GeForce 2080* has 8.5 times more computational power than the *Xavier*, the expected speed up should be around 8.5. Nonetheless, for propagation times equal to 10 and 48 hours, the speed up is 3.56 and 4.13 respectively, much lower than the expected speed up. The only situation where the speed up is around the expected result is when the propagation time is 24 hours, 8.7. The worst speed up value is obtained for *Short-Term Simulation*, 5 hours of fire propagation where the speed up is reduced significantly, only 1.5.

Table 5.3: Thousands of perimeter points per seconds processed for *Xavier* and *GeForce 2080* GPUs for a *Perimeter Resolution* equal to 5 meters.

Propagation Time	*Xavier* (*K points/Second*)	*GeForce 2080* (*K points/second*)
5 hour	41.365	64.792
10 hours	45.010	160.343
24 hours	31.519	274.690
48 hours	11.575	47.807

In order to normalize the execution time invested in the propagation of a single point, we compute the number of perimeter points processed per second. Table 5.3 summarises the computed values. First of all, we see that the *GeForce 2080* processes more perimeter points per second than the *Xavier*. As we can see, the *Xavier* reaches its maximum number of points per second when we simulate 10 hours of fire evolution, but the *GeForce 2080* reaches its maximum number when we simulate 24 hours of fire propagation. However, when we are in the *Short-Term Simulation* zone, 5 hours or less, the difference of the number of perimeter points processed per second by both GPUs is less than $25,000$ points. Considering that we are interested in a device that can collect the necessary input parameters and performs the simulations of the wildfire spread *in situ*, we are interested in evaluating the energy efficiency of the platform. To measure the energy efficiency of each GPU, we computed the *Energy Capability*, see Equation 5.1.

Figure 5.6 shows the *Energy Capability* of both GPU. *Energy Capability* represents the number of propagated points per work unit. The larger it is, the more

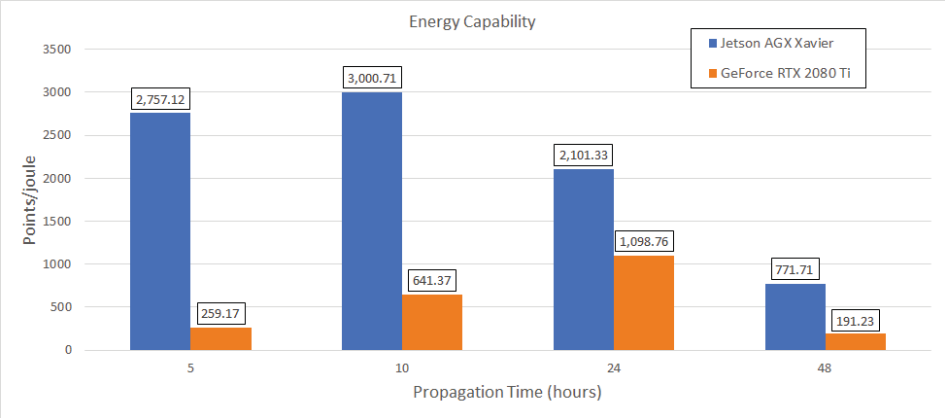


Figure 5.6: *Energy Capability* of the forest fire spread simulations for both GPUs with a *Perimeter Resolution* equal to 5 meters, depending on the propagation time.

energetically efficient it will be. When comparing the nominal energetic efficiencies (Figure 5.6, obtained from the performances and Table 5.1) one notes that the *Xavier* stands out among *GeForce 2080* in terms of the number of $point/joule$. As we can see, the *Energy Capability* of the *Xavier* is higher than the *Energy Capability* of the *GeForce 2080*, specially when we are in the *Short-Term Simulation* region. In this case *Energy Capability*, for the *GeForce 2080*, is between 259.17 $points/Joule$, while the *Xavier* obtains a value of $2,757.12$ $points/Joule$. These values indicate that the propagation of a single point consumes around 10 times more energy in the*GeForce 2080* than in the *Xavier*. However, we need a parameter to compare the energy efficiency of both devices and help us choose in which situation the increment of the energy consumption is not compensated by the reduction of the execution time.

$$Green\ Up = \frac{Energy\ Capability_{Jetson}}{Energy\ Capability_{GeForce2080}} \tag{5.2}$$

To compare the energy efficiency of the two GPUs, we define the *Green Up*. The term *Green Up* is defined as the ratio of the *Energy Capability* of the *Xavier* over the *Energy Capability* of the *GeForce 2080*, see Equation 5.2. *Green Up* is analogous to *Speed Up* as it reflects how better is the simulation in terms of energy

consumption. If the *Green Up* is higher than the speed up, it means that the increment of the energy consumption is not compensated with the acceleration of the simulation. On the contrary, if the speed up is higher than the *Green Up*, it indicates that the increment of the energy consumption is compensated by the reduction of the execution time.

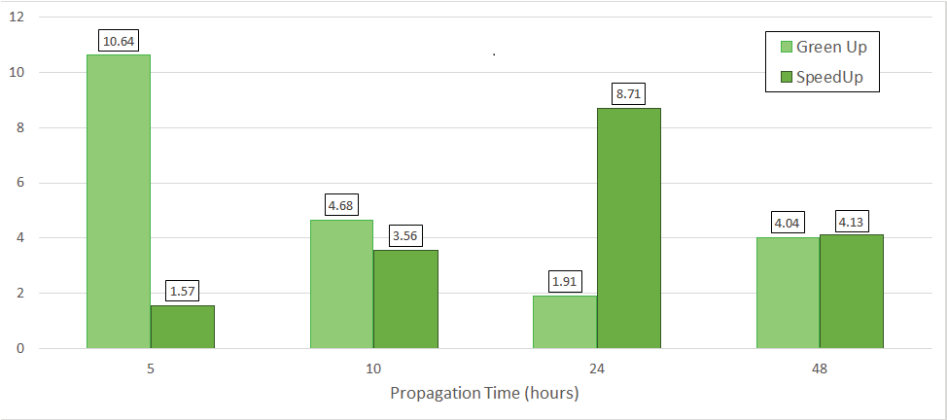


Figure 5.7: Speed up (dark green column) of the *GeForce 2080* and *Green Up* (light green column) of the *Xavier* for different propagation time. The term *Green Up* is defined as the ratio of the *Energy Capability* of the *Xavier* over the *Energy Capability* of the *GeForce 2080*, see Equation 5.2.

Figure 5.7 displays the speed up (dark green column) of the *GeForce 2080* and *Green Up* (light green column) of the *Xavier* for the four different propagation times. On one hand, the *GreenUp* shows that the *Xavier* platform is between 2 and 10 more energy efficient that the *GeForce 2080*. On the other hand, the speed up varies from 1.57 to 8.71, which indicates that the *GeForce 2080* has a better performance than the *Xavier*. We observe a balance between the speed up and *Green Up* that determines when it is expedient the use one GPU or the other. When we compare the obtained values of the *Green Up* with the speed up, we can identify a clear use case. It can be seen, than for 5 hours (*Short-Term Simulation*) of fire evolution, the *Green Up* of the *Xavier* is 10.64, while the speed up of the *GeForce 2080* is 1.57. It means that, in this case, the acceleration of the *GeForce 2080* does not compensate the increment of the energetic consumption. As we expect, in all cases, the *Xavier* platform has the best energy efficiency; nevertheless, when we simulate 24 hours of fire propagation, the speed up of the *GeForce*

2080, 8.71, extensively compensates the *Green Up*, 1.91, of the *Xavier*, therefore, in this situation, the increment of the energy consumption is clear compensated by the speed up of the *GeForce 2080*. It can be observed that when we simulated 48 hours of fire evolution, the *speed up* and the *Green Up* are very close in this situation.

Analysing the balance between the speed up and *Green Up* we can select the best scenario to be used in edge computing. The results show a clear case in which the *Xavier* takes advantage in front of the *GeForce 2080*. When the propagation time is equal to 5 hour, the *Xavier* is 10 time more efficient than the *GeForce 2080*, but the speed up is only 1.57. For that reason, we can define this situation as the *Edge Scenario*. This is the best candidate to be simulated with edge computing. Therefore, this scenario is studied in deep detail for different *Perimeter Resolution*.

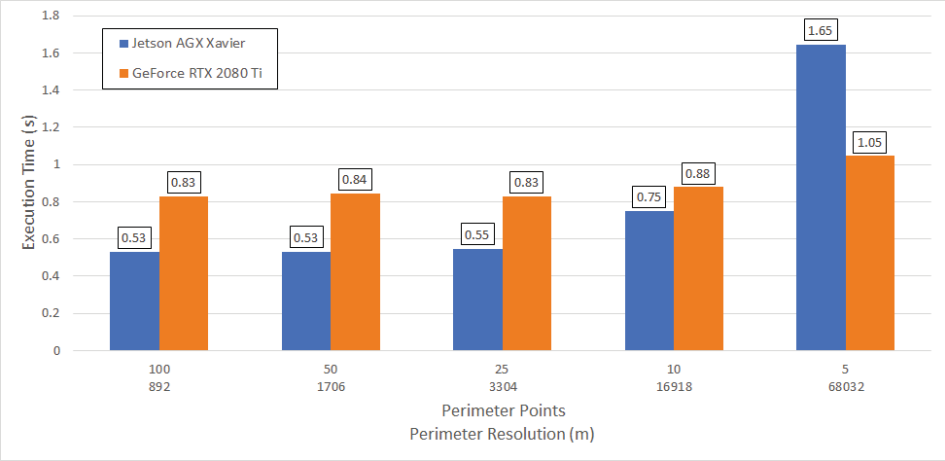


Figure 5.8: Execution time depending of the *Perimeter Resolution*in the *Edge Scenario* for *Xavier* (blue) and *GeForce 2080* (orange).

In figure 5.8, the execution time consumed in the *Edge Scenario* for different *Perimeter Resolution* is shown. We can observe that for *Perimeter Resolution* above 5 meters, or less than $68,032$ perimeter points, the *Xavier* is faster than the *GeForce 2080*. As we said, the regular desktop GPUs, like *GeForce 2080*, spend around $0,8$ seconds in the start-up. For that reason, when the number of perimeter points is not big enough, the start-up time cannot be compensated by the computational work. For that reason, in these cases, the *Xavier* consumes less

execution time. When the *Perimeter Resolution* is 5 meters the *GeForce 2080* is faster than the *Xavier*. In this case, the number of perimeter points is sufficient so that the computational work can compensate the start-up time.

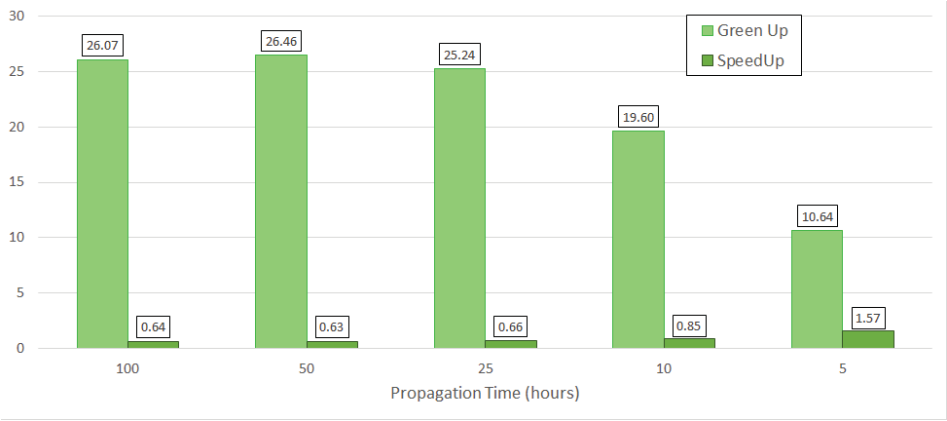


Figure 5.9: Speed up (dark green column) of the *GeForce 2080* and *Green Up* (light green column) of the *Xavier* for different *Perimeter Resolution* in the *Edge Scenario*.

As we did above, we evaluate the balance between the *Green Up* of the *Xavier* and the speed up of the *GeForce 2080* for different *Perimeter Resolution* in the *Edge Scenario*. When we work with high *Perimeter Resolution*, the *Xavier* is faster than the *GeForce 2080*, as a result, the speed up is under 1. In addition, because its execution time is less than the execution time needed by the *GeForce 2080*, the computed *Green UP* is between 19 and 26, which means that the *Xavier* is 20 time more efficient than the *GeForce 2080*.

Analysing these results, we can clearly see that, although the *GeForce 2080* has 8.5 time more computational resources, in most scenarios, the *Xavier* is not 8.5 slower than the *GeForce 2080*. In addition, the power efficiency of the *Xavier* makes it the perfect platform to apply edge computing to the simulation of the forest fire spread. Therefore, the utilization of the *Xavier* can help to assess the short-term risk of a forest fire *in sito* where the fire is burning, favoring a rapid response to possible fire variations and optimizing firefighting resources.

The previous results obtained, *Mode 15W* or Mode 2, (see Figure 5.2), was used, which is the *default Mode*. Nonetheless, one of the most interesting aspects

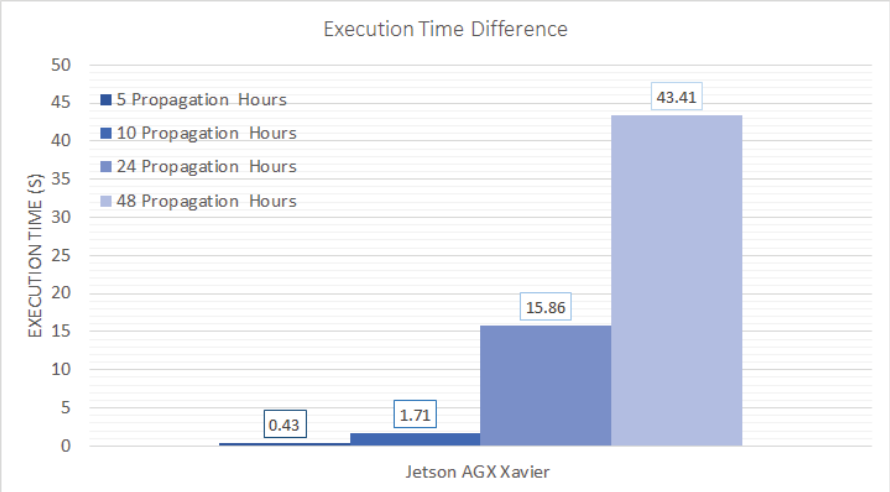


Figure 5.10: Maximum *execution time* difference between *Xavier* in Mode 0 (TDP=30W) and Mode 2 (TDP=15W) for the different fire evolution times.

of the *Jetson ARG Xavier* is the capacity to work in different energy Modes and configurations. We are interested in studying the optimal balance between performance and energy consumption. Therefore, to see if the reduction of the execution time compensates the increment of the energy consumption, we compare the previous results obtained with Mode 2 (TDP=15W) against the *Mode EDP*, or Mode 0, which work with a TDP of 30W, (see Figure 5.2). Figure 5.10 displays the maximum execution time difference between the *Xavier* in Mode 2 and the *Xavier* in Mode 0. We observed that when the simulation is performed in Mode 0, the simulation is faster than when Mode 2 is used. However, we are interested in studying the optimal balance between the execution time and the *Energy Capability* in the particular case of forest fire simulation. To allow us to analyze this equilibrium between performance and energy consumption, we compare the speed up of Mode 0 and the *Green Up* of Mode 2. The ideal case would be one in which the speed up is equal or greater than the *Green Up*, which means that the reduction in the execution time compensates the increment in energy consumption.

Figure 5.11 illustrates the maximum speed up (dark green column) of the *Xavier* in Mode 0, and the obtained *Green Up* (light green column) for the *Xavier* in Mode 2. As displayed, the *Green Up* decreases when the propagation time

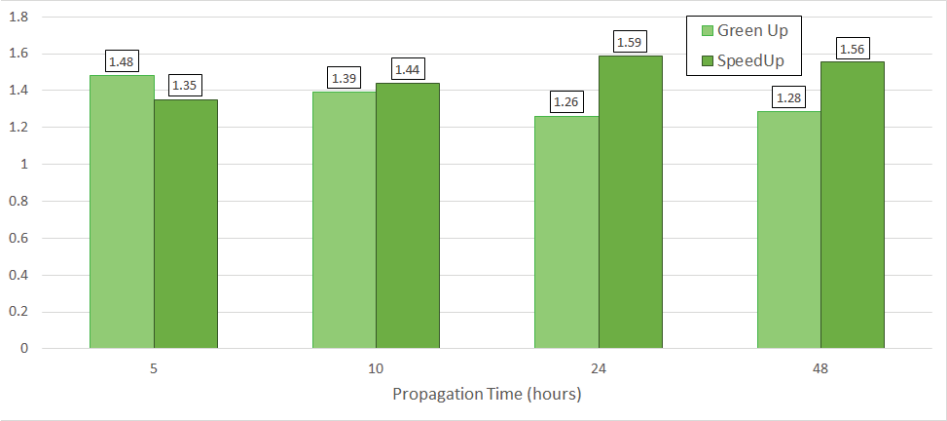


Figure 5.11: Speed up (dark green) of the *Xavier* in the EDP Mode and *Green Up* (light green) in the Mode 2 (15W mode).

increases. In contrast, the speed up raises with the propagation time. It can be seen that for 5 hours of fire evolution, *Short-Term Simulation*, the consumption of energy is not compensated by the reduction of the execution time. We found that the *Green Up* is $1, 1.48$ while the Seep Up is 1.35. These results illustrate that the speed up does not compensate the increment of the energy consumption of Mode 0; therefore, in this case, Mode 2 would be the best option. On the contrary, for propagation time higher than 10 hours, the results suggest that the speed up compensates the *Green Up*; hence, Mode 0 would be the best choice to perform this kind of forest fire spread simulation. Nonetheless, the *Green Up* and the Seep Up of both energy Modes are very close. In the following section, the different energy Modes of the *Xavier* are compared using a real forest fire.

We analysed the different characteristics needed in edge computing using an ideal scenario. *Edge scenario* has been defined as the best scenario to be simulated in edge computing. In order to identify this *Edge scenario* the *Green Up* and the speed up are compared. The *Edge scenario* is those that the *Green Up* is higher than the speed up. In the case of a synthetic fire, the obtained results demonstrate that the *Edge scenario* is when the *Short-term Simulation*, 5 hours of fire evolution, with a *Perimeter Resolution* equal to 5 meters. In the following section, a real fire is used to analyse the best energy mode to respond to the computational requirement that edge computing requires.

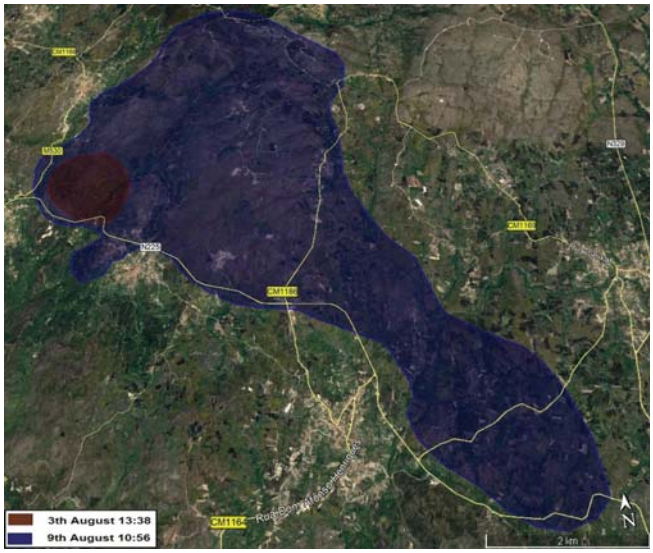


Figure 5.12: Digital Elevation map of São Joaninho fire area. The *Red Perimeter* was used as initial perimeter (ignition Perimeter). The perimeter to be predicted is *Blue Perimeter*, [23].

5.1.2 Real fire

In the second experiment, we focus on the 5 hours of fire evolution scenario to evaluate the performance of our target edge computing case, *Edge Scenario*. In this scenario, the features of the *Xavier* make it stand above the *GeForce 2080* for edge computing. As a study case, we use a forest fire that took place in 2013 in the region of São Joaninho, Portugal. The forest fire began on August 3rd, and the total burnt area was 1,973ha. Figure 5.12 shows the fire perimeters at two different time instants: t_0 (August 3th at 13:38am) and t_1 (August 9th at 10:56am).

Because, *Short-term fire propagation* is the most relevant information that allows firefighters to assess the short-term risk in order to quickly optimize the strategy to reduce the impact of the wildfire, we simulate the evolution of the fire front for five different propagation times, 1 hour, 2 hours, 3 hours 4 hours and 5 hours, and for five different *Perimeter Resolution*, 5, 10, 25, 50, and 100 meters. The performance of different energy Modes of the *Xavier* is analyzed. The main objective is to analyse how the energy consumption affects the performance of the

simulations and which is the most efficient combination of *Jetson ARG Xavier developer kit* energy Mode to apply the edge computing to the forest fire spread simulation in the *Short-Term Simulation*.

Figure 5.13 summarizes the execution time for the different propagation time and different *Perimeter Resolution* in Mode 2 (default), see 5.2. We can see that for propagation time of 1, 2, and 3 hours, the execution time invested into performing the simulation is under 2.5 seconds for all *Perimeter Resolution*. In addition, we can observe that the difference between simulating 1 hour or 3 hours is less than a second. When the propagation time is fixed to 4 hours, the execution time is between 3.69 and 9.30 seconds. However, the execution time notably increases when we simulate 5 hours of propagation time. The simulation that needs more time to perform the simulation is the simulation with a *Perimeter Resolution* equal to 5 meters, 57.09 seconds. Nonetheless, we simulated the near future of the forest fire in less than a minute, which is close to simulating the forest fire spread near real-time. Consequently, in the following paragraphs, we consider work with a *Perimeter Resolution* equal to 5 meters and a fire propagation time of 5 hours.

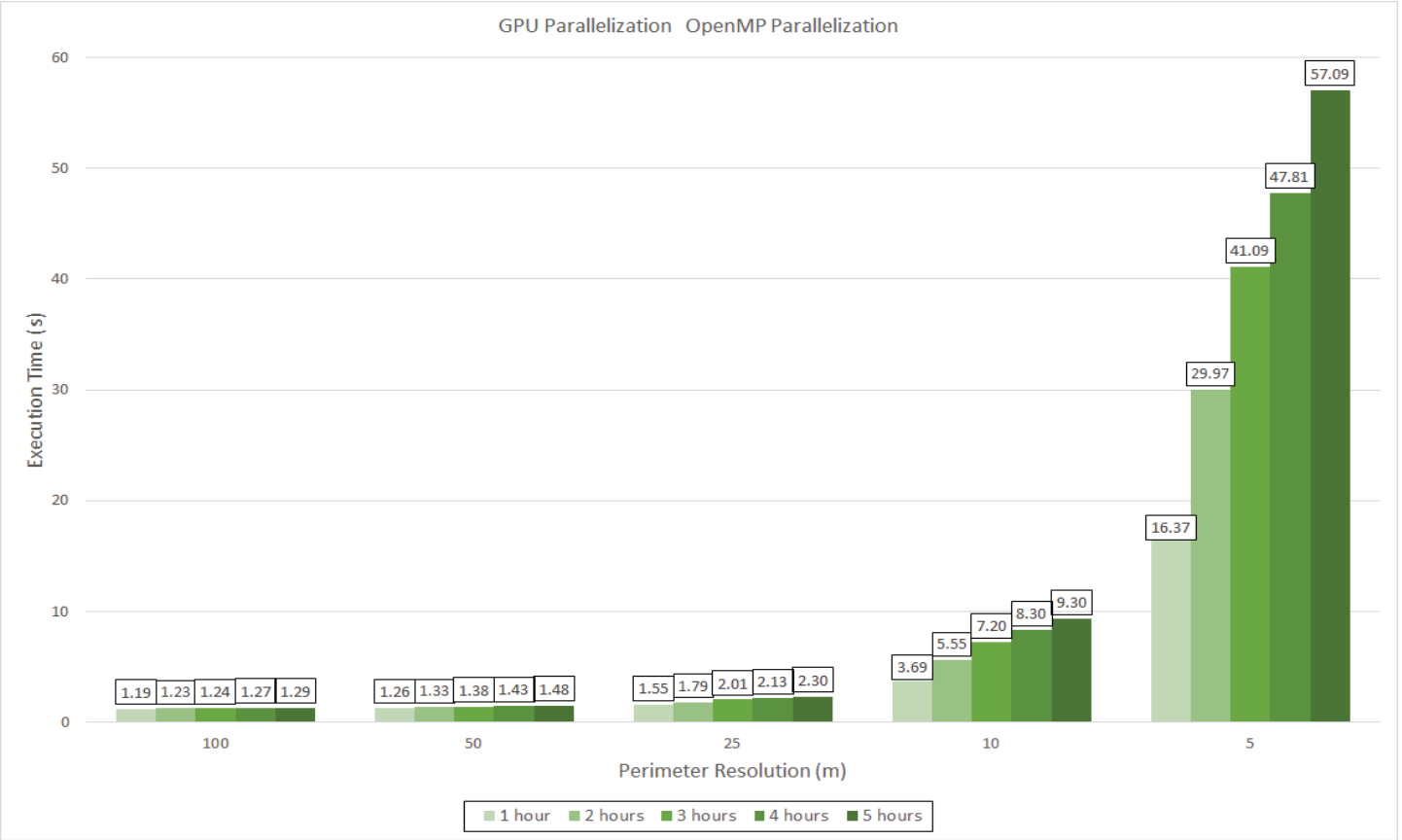


Figure 5.13: Execution time for five different propagation times and for five different *Perimeter Resolution* in *Xavier* with Mode 2 (15W).

Table 5.4: Execution time invested for different energy Mode, to perform the simulations with *Perimeter Resolution* equal to 5 meters for five different propagation time, 1, 2, 3, 4 and 5 hours.

Energy Mode	TDP	1 hours	2 hours	3 hours	4 hours	5 hours
Mode 0	(30W)	9.745	17.678	23.673	27.706	32.566
Mode 2	(15W)	16.373	29.973	41.091	47.808	57.087
Mode 3	(30W)	12.745	21.512	33.079	46.079	54.598
Mode 4	(30W)	13.944	25.553	34.534	40.486	47.002
Mode 5	(30W)	11.672	21.918	29.232	34.167	39.770
Mode 6	(30W)	10.650	19.373	26.035	30.441	35.877

Table 5.4 presents the execution time for the different energy Modes of the *Xavier*, see Figure 5.2, and different fire evolution times for both parallel implementations. In this case, the *Mode* $10W$ (Mode 1) is not used because, in this energy Mode, the GPU remains off. As we expect, the lowest execution time is obtained when Mode 0 is used. In this energy Mode, the *Xavier* works at a maximum capacity. One interesting aspect when we analyze the performance of the GPU-FARSITE implementation is that the second fastest energy Mode is Mode 6. This is an unexpected result because Modes 3, 4, and 5 allow the utilization of more CPU cores than Mode 6; 8, 6, and 4 CPU cores, respectively. While the code executed in the GPU is similar in all Modes, there are parts of the code that runs in the CPU. These Modes limit the maximal frequency of the CPU. While Modes 3, Modes 4 and 5 have $1,200MHz$, $1,450MHz$ and $1780MHz$, Mode 6 has a CPU Maximal Frequency equal to $2,100$. This frequency increment compensates the reduction of the CPU cores available; therefore, the Mode with the higher maximal frequency has the best performance.

As we said, in order to measure the energy efficiency of the different Modes, the *Energy Capability* is computed, see Equation 5.1. The *Energy Capability* reflects the work necessary to propagate a single point from the initial perimeter to the final perimeter. Figure 5.14 reveals the *Energy Capability* for different energy Modes when the *Perimeter Resolution* is 5 meters and the propagation time is 5 hours. We see that the maximum *Energy Capability* is obtained with Mode 2. However, the second energy Mode with the highest *Energy Capability* is Mode 0. Until Mode

0 consumes 30W, double than Mode 2, its energy capability is only 12% less than Mode 2. This is because Mode 0 only needs 32.57 seconds to simulate the forest fire evolution, around 42% less execution time than Mode 2. On the contrary, the Mode with the worst *Energy Capability* is Mode 3. This seems to have no sense because Mode 3 allows the utilization of the 8 cores of the CPU. However, in this case, the CPU Maximal Frequency, 1200Mhz, is the lowest of all the Modes that have a TDP of 30W, see Figure 5.2. The increment of the CPU cores does not compensate this reduction of the CPU Maximal Frequency; therefore, it is the Mode with the lowest *Energy Capability*.

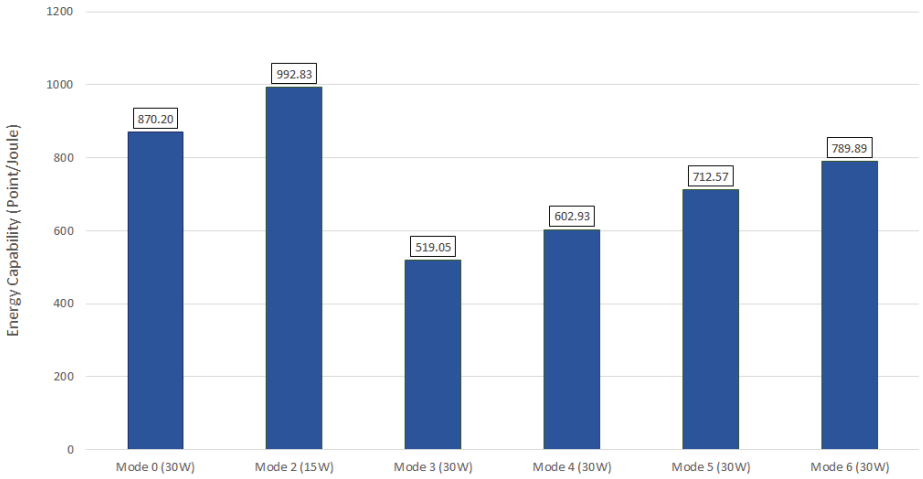


Figure 5.14: *Energy Capability* of the GPU-FARSITE parallelization for five different propagation times and a *Perimeter Resolution* equal to 5 meters.

As in the previous section, we use the *Green Up* and the*Speed Up* to evaluate in which circumstances we can suggest the utilization of one energy Mode or another. As a reference Mode, we utilize Mode 2 because it is the Mode with the highest *Energy Capability* and the highest execution time. Consequently, we calculate the *Green Up* of energy Mode 2 over the other Modes and the *Speed Up* of the other Modes over Mode 2. Figure 5.15 reveals the comparison of the *Green Up* and the speed up obtained. In the case of energy Modes 3 and 4, the Green Up is higher than the speed up. Therefore, these energy modules do not compensate for the increment of energy consumption. Nonetheless, when Mode 0 is utilized, the speed up is higher than the Green UP. This result suggests that the energy Mode

with the best balance between energy consumption and computational power is Mode 0. This is a consequence of the increment of the perimeter points. When we work with a synthetic fire, the maximum number of points to be processed is around 68,000 perimeter points. This number increases in the real fire case since 850,000 perimeter points. Therefore, the computational work to simulate the behaviour of the real forest fire is 12.5 times higher than the synthetic fire. For that reason, in this particular case, the best energy Mode is the one that has the highest computational capability.

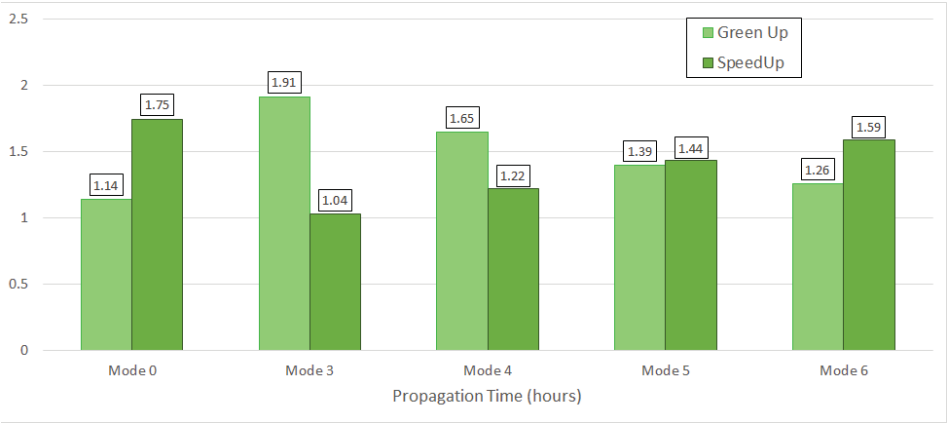


Figure 5.15: Green Up (light green) and speed up (dark green) for five Different propagation times and a *Perimeter Resolution* equal to 5 meters.

The new embedded systems, like *Xavier*, open a new perspective where the monitoring and prediction of the fire behaviour could be made in the same place where the fire is taking place. Clearly, we can distinguish two different situations. On one side, when we are interested in minimizing the execution time to perform the simulation of the forest fire, in which case, the obtained results show that Mode 0 is the best option. On the other side, when we want to maximize energy efficiency, the best choice is Mode 2. However, we have to sacrifice some performance, which translates into an increment in execution time. In the particular case, where the embedded system is combined with a drone to reduce input data uncertainty, "extreme edge computing", energy optimization could be a crucial aspect. Suppose the fire burns in a zone with difficult access and low connectivity. In that case, the computational system must be as energy efficient as possible, although this implies increasing the execution time in a few seconds. In this case,

the utilization of the *Xavier* with energy Mode 2 is the best choice. The obtained results highlight that the computational and consumption characteristics *Xavier* make them the perfect platform when we need to assess the short-term risk of a wildfire. However, depending on the characteristics of the urgent response to a hazard, like a forest fire, the simulation requirements can change, and the balance *time/cost* may change. Under these circumstances, the users should consider the peculiarities of their problem, taking into account their limitations to choose the best energy Mode.

Chapter 6

Conclusions and OpenLines

In the last years, the augment of the global temperature caused by climate change has intensified the number and damage of forest fires. Models and their implementation in simulators can estimate the behaviour of the wildfires, but they are not exempt from a certain degree of error. The quality results of these models depend not only on the propagation equations describing the behaviour of the fire but also on the input data required to initialize the model. Typically, this data is subjected to a high degree of uncertainty and variability during the evolution of the event due to the dynamic nature of some of them, such as meteorological conditions or moisture contents in the vegetation. Consequently, it is necessary to find strategies to minimize this uncertainty in order to provide better predictions. Previous studies have demonstrated that the use of the *Two-stage Methodology* increases undoubtedly the accuracy of the predictions. However, due to its characteristics, *Two-stage Methodology* implies that the execution time invested in performing a prediction of the fire propagation increases significantly. Because the forest fire spread simulators based on the *Elliptical Wave Propagation Algorithm* cannot distinguish between burned and not burned zones, the simulation of the forest fire evolution can be classified into big blocks: the propagation of the fire front and the reconstruction of the fire perimeter, which represent around the 7% and 60% of the execution time respectively for big fires. In this study, the two blocks are analyzed separately. In particular, we focus on the forest fire spread simulator FARSITE, which is a widely accepted forest fire spread simulator in the scientific

community related to this field; however, the proposed methodology is simulator independent and could be applied to any other forest fire spread simulator. During this thesis, we want to achieve three main objectives: The first objective is to reduce the execution time invested in the GA. The second objective is to improve the accuracy of the forest fire spread simulator without excessive execution time penalties, and the last objective is to accomplish forest fire spread simulations near real-time and near the location where the fire is taking place. To achieve these goals, three different strategies were developed. The main conclusions of each strategy are described in the following sections.

6.1 Mixed Precision Methodology Conclusions

For historical reasons, a large part of the scientific codes overestimate the numerical precision that a model needs; for that reason, the generalized use of double-precision in scientific codes has been put into revision as not all variables require it. Different works expose that the generalized use of double-precision in most scientific codes could be notably reduced. The use of mixed-precision would payback in terms of performance improvement, requiring little coding effort. For this reason, our first strategy consists of applying the mixed-precision methodology to simulate forest fire propagation using the *Two-stage Methodology*, which is based on an evolutionary scheme (GA) without damaging the accuracy of the final simulation. A relevant outcome from the experimental study is that around 74% of the variables used to calculate the point expansion algorithm can be defined in single-precision. On the other hand, only 15% of the variables used in the fire front reconstruction module could be used with single-precision. This is an essential restriction because that part represents, in some cases, around 60% of the total simulation time. For that reason, in those particular cases where the Fire Front reconstruction algorithm has a higher weight, the performance improvement using mixed-precision is limited.

The proposed mixed-precision methodology has been tested two types of sce-

narios instanced by Arkadia's, *Fire Front Reconstruction Oriented*, and Nuñomoral's fires, *Point Expansion Oriented*. We saw that the local conditions of the scenario where the fire occurred have a notable impact on the performance improvement provided by the mixed-precision methodology. The experimental study performed analyzes the impact of the mixed-precision in the classic prediction scheme where a single simulation is done to predict the evolution and also it has been tested on the *Two-stage Methodology* where a genetic algorithm is used in a data-driven way to determine the most suitable values of the input simulator parameters based on the current evolution of the forest fire. As it was expected, the application of the *Two-stage approach* improves notably the prediction of the evolution of the fire front. In both simulated scenarios, the computed error of the prediction was lower when we used the *Two-stage approach* than the classic scheme. However, this improvement in terms of quality has a high cost in terms of execution time. We saw that, in general, the *Two-stage approach* consumes around 10 times more execution time than the classic scheme. The obtained results, when combining the *Two-stage Methodology* with the mixed-precision implementation, show that the mixed-precision implementation does not compromise the quality of the forest fire spread simulation because of the slight differences in the fire evolution.

However, this difference is accumulative; therefore, this difference is more discernible when the fire propagation increases. This is because the usage of less precision introduces a tiny error for each time loop; consequently, when the propagation of the fire front raises, the error becomes slightly detectable even values are always below predefined error limits. Due to the forest fires being very complex systems, we are interested in predicting the general evolution tendency of fire. In both scenarios, both implementation, the double and the mixed-precision predict the same fire propagation tendency. The performance improvement of the mixed-precision implementation depends on the wildfire scenario. The Point Expansion Oriented fires, Nuñomoral's, tends to have a better performance improvement than in the case of Fire Front Reconstruction Oriented fires, like Arkadia's case. This improvement variation could be caused due to the higher terrain complexity, like in the Arkadia's fire. When the wildfire takes place in complex terrain, the weight

of the fire front reconstruction algorithm raises. In the fire front reconstruction algorithm, only the 15% of the variables can use single-precision; for this reason, the performance benefit is limited. Thus, the performance improvement of the mixed-precision approach is more significant for those scenarios where the fire is more Point Expansion Oriented.

6.2 Fine-Grained Paralelization Conclusions

In order to improve the accuracy of the forest fire spread simulation without penalizing the execution time excessively, we apply a novel fine-grained parallelization. The computing power of GPUs make them the ideal device to implement the fine-grain parallelization of the forest fire spread effectively. In Chapter 4 a novel GPU implementation for forest fire spread simulators based on the Elliptical Wave Propagation is described. The proposed GPU implementation has been validated utilizing a real scenario that took place in Arkadia (Greece) in 2011. To analyze the efficiency of the novel GPU implementation, it has been tested against an OpenMP implementation. First of all, the two simulation blocks, the Fire Front Propagation algorithm, and the Fire Front Reconstruction algorithm, are evaluated separately. Finally, the performance of the whole simulator is analyzed.

When the Fire Front Propagation algorithm is evaluated, we see that for *Perimeter Resolution* equal to 100, 50, and 25 meters, less than $730,000$ perimeter points, the implementation in OpenMP is faster than the GPU implementation. This is because GPUs have an ongoing start-up cost caused by their initialization and the time invested in the data transfer from the *Host* to *device* memories. In this situation, the start-up and copies times are not compensated by the computational work. However, for lower *Perimeter Resolution*, the GPU becomes more efficient than the OpenMP implementation. We see that the speed up increases as the *Perimeter Resolution* decreases. The maximum speed up is reached for the lowest *Perimeter Resolution*, 5 meters. This is due to the complexity of the Fire Front Propagation algorithm being $O(n)$, where n is the number of perimeter

points. We saw that for low *Perimeter Resolution* the GPU is the most efficient device to execute the Fire Front Propagation algorithm. In order to normalize the computational work, we compute the number of propagated points per second. For *Perimeter Resolution* equal to 5, the GPU parallelization propagates around 54% more perimeter points per second than the OpenMP implementation.

When the Fire Front Reconstruction algorithm is analyzed, we see that the execution time consumed is much higher than in the Fire Front Propagation algorithm. The obtained results reveal that, except for *Perimeter Resolution* equal to 100 meters, the GPU parallelization is the fastest in all situations. However, when the speed up is analyzed, we found that the maximum speed up is reached with a *Perimeter Resolution* equal to 10 meters. In this situation, we obtain a speed up round of 5.48. Under this *Perimeter Resolution* the speed up decays to 3.02.

The performance improvement is notable if we combine the GPU implantation of the Fire Front Propagation and the Fire Front Reconstruction in a GPU-FARSITE fire spread simulator. For *Perimeter Resolution* of 100 meters, the serial implementation is the fastest one. As we mention, in this situation, computational work cannot compensate for the start-up time of the parallel implementations, and the copy data time, in the case of the GPU application. For this *Perimeter Resolution*, the GPU implementation is the slowest implementation. For the other *Perimeter Resolution*, both parallel implementations are more efficient than the serial FARSITE. The maximum speed up of both parallel implementations is obtained with the lowest *Perimeter Resolution*, 11.99 for the GPU and 3.96 for the OpenMP applications. Except for a *Perimeter Resolution* of 100 meters, the GPU implementation is faster than the OpenMP one in all different scenarios. When we compare both parallel implementations, we observe that the GPU application is the most efficient to perform simulations with high accuracy, low *Perimeter Resolution*. We obtain the maximum speed up for *Perimeter Resolution* equal to 10 meters. This is because the bandwidth of the GPU is saturated. Therefore, we went from a compute-bound problem, where the number of perimeter points can be raised without increasing the execution time significantly, to a memory-bound problem, where the increment of perimeter points has a higher impact on the ex-

ecution time. In order to analyze the data transfer between memories, the copy time was measured.

We obtain that the execution time invested in the data transfer decrease exponentially as the number of perimeter point raise. When the transfer work is normalized, we see that above $16,500$ points per second, the bandwidth of the GPU is saturated. Hence, the increment of the perimeter points directly impacts the execution time needed to simulate the forest fir spread. As a take-home message, we can conclude that the new GPU-FARSITE simulator possibilities the forest fire spread propagation simulation with high accuracy because it significantly improves the performance of the serial implementation and the OpenMP parallelization.

6.3 Edge Computing Conclusions

As we saw, one of the principal problems of the forest fire spread simulators is related to the input data uncertainty. Different strategies had been developed in order to reduce this input data uncertainty, like *Two-stage Methodology*. These strategies focus their effort on calibrating the input parameter; however, none of these are oriented to improve the quality of input measurements. In the last years, the advances in the UAVs field allowed the utilization of drones to collect the data parameters of the forest fire needed for the forest fire spread simulator. The direct measurement of these data by a drone at the same place and at the same time when a fire is occurring could reduce the uncertainty of the input data parameters significantly. However, this data has to be sent somewhere in order to be processed, and the evolution of the forest fire simulation has to be sent back to the firefighters. This is time expensive and impossible in those zones with low connectivity. For this reason, the optimal solution is to perform the simulation of the near future of the fire spread in the same place, or close, where the fire is taking place. To accomplish this objective, we utilize Edge Computing. The main requirements that edge computing has to accomplish are three: it has to work in

a low connectivity environment, has short execution times, and be very energy efficient.

The new Embedded Systems with our novel GPU-FARSITE implementation opens a new way to exploit maximum edge computing. In the last part of the thesis, we test the GPU-FARSITE implementation in an embedded system with a low consumption GPU the *Short-term Simulation*, less than 5 hours of fire propagation. In our case, we utilize an NVIDIA Jetson AGX Xavier developer kit, *Xavier*. For the purpose of studying the possibilities of these kinds of devices, we tested the *Xavier* into two different studies. In the first experiment, we used a synthetic fire in flat terrain, with homogeneous fuel and a constant wind speed and wind direction throughout the simulation. The performance of the *Xavier* has been compared against a desktop GPU, GeForce RTX 2080 Ti, *GeForce*. In general, the *GeForce* needs less execution time to perform the forest fire spread simulation than the *Xavier*. However, *GeForce* has 8.7 more computational power than the *Xavier*, but *Xavier* is not 8.7 times slower. According to the study carried out, when we work with high *Perimeter Resolution*, 100 and 50 meters, and less than 10 hours of fire propagation, the *Xavier* is faster than the *GeForce*. This is because the *GeForce* has a higher start-up time than the *Xavier*; therefore, in those situations in which the number of perimeter points to be processed is not large enough, the computational work cannot compensate this time. As a consequence, the *Xavier* is the fastest in those simulations in which the execution time is less than a second.

In the other scenarios, the *GeForce* needs less execution time to perform the forest fire spread simulation than the *Xavier*. However, we want to study the possibility of using the low consumption GPUs to create a new device to simulate the evolution of the fire *in situ*. For that reason, the consumption of energy is a crucial point to take into account. When the *Energy Capability* is computed, we saw that the *Xavier* processed more perimeter points per energy unit than the *GeForce* or, we need less energy to compute the evolution of a single point in the *Xavier* than in the *GeForce*. This is due the *Energy Capability* of the *Xavier* is higher than the *Energy Capability* of the *GeForce*. This result highlight that the *Xavier* is more energetically efficient than the *GeForce*. Consequently,

even the *Xavier* is slower than the *GeForce*, the obtained execution time proves that we can use a low consumption GPU to simulate the evolution of the front fire with high accuracy, with operational times. This is illustrated by comparing the *Green Up* and the speed up. We saw that for *Short Term simulation* the Green Up is 10 higher than the speed up, which indicates that the increment of the simulation velocity does not compensate for the increment of the energy consumption. Analysing the balance between *Green Up* and the speed up, we can define this situation as the *Edge Scenario*, as the best candidate to be simulated with edge computing. The *Edge scenario* is those that the *Green Up* is higher than the speed up. The obtained results highlight that the *Edge Scenario* is when the propagation time is 5 hours of fire evolution, and a *Perimeter Resolution* equal to 5 meters. Or in other words, the *Edge Scenario* is *Short-term Simulation*, with high accuracy. We clearly saw that the power efficiency of the *Xavier* makes it the perfect platform to apply edge computing to the simulation of the forest fire spread. Therefore, the utilization of the *Xavier* can help to assess the short-term risk of a forest fire *in situ* where the fire is burning, favoring a rapid response to possible fire variations and optimizing firefighting resources.

Because *Xavier* has different operating modes, we are interested in studying the optimal balance between the *Simulation Time* and energy consumption. For this purpose, in the second experiment, the different energy modes of the *Xavier* are tested in the *Short Term simulation*. In particular, we focus on the fire evolution time of 1, 2, 3, 4, and 5 hours. We use a real fire to analyse the efficiency of the different power configurations of the *Xavier*. From the six possible power configurations, the most interesting is the Mode 2, default Mode, with a Thermal Design Power of 15W, and Mode 0, in which the *Xavier* works at full capacity, with a Thermal Design Power of 30W. Mode 0 is the energy mode that allows the faster simulation, around 30 seconds, but it has a higher energy consumption than Mode 2. On the contrary, with Mode 2, the obtained execution time is the highest. However, with this power configuration, we reach the maximum energy capability of the *Xavier*. In order to determine which is the optimum energy mode, we compare the Green Up of Mode 2 against the speed up of the other Modes. We observed that Mode 0 has a speed up higher than the *Green Up* of Mode 2. The

other energy configuration with a higher speed up than *Green Up* is Mode 6, but this Mode is slower and has less energy capability than Mode 0.

The new embedded systems, like *Xavier*, opens a new perspective where the monitoring and prediction of the fire behaviour could be made in the *in situ* where the fire is burning. We distinguished two different situations. On one side, when the minimization of the execution time to perform the forest fire simulation is the priority, the obtained results show that Mode 0 is the best option. The second case is in those situations where the maximization of energy efficiency is of paramount importance. In these scenarios, the best choice is Mode 2. However, we have to sacrifice some performance. If an embedded system is combined with a drone in order to reduce input data uncertainty, "extreme edge computing", the optimization of the energy consumption is critical. In addition, If the fire takes place in a zone with low connectivity, the computational system must be as energy efficient as possible, although this implies increasing the execution time in a few seconds. In this case, the utilization of the *Xavier* with energy Mode 2 is the best choice. These results highlight that the computational and consumption characteristics *Xavier* makes them the perfect platform when we need to assess short-term risk of a wildfire. However, depending on the characteristics of the urgent response to the fire propagation, the simulation requirements can change, and the balance *time/cost* may change. Under these circumstances, the users should consider the peculiarities of their problem, taking into account their limitations to determine the best energy configuration.

6.4 Open Lines and Future Work

At the final of this thesis, some Open Lines must be solved in the near future. First of all, the mixed-precision methodology has to be applied to GPU-FARSITE parallel implementation, which will improve its performance and reduce the weight of the data transfer between memories. Finally, the most relevant open line is to find a new Fire Front Reconstruction algorithm with low complexity. The complexity of the current Fire Front Reconstruction algorithm is $O(n^4)$. Our GPU parallel implementation of the Fire Front Reconstruction algorithm reduces its complexity to $O(n^3)$. If the number of perimeter points is high enough, we can replace the Crosswalk algorithm with an enveloping algorithm, like α-Shape, with low error. This new algorithm reduces the complexity of the Fire Reconstruction to $O(n \cdot \log n)$. *Increase the accuracy to go faster.*

References

[1] Fatemeh Afghah, Abolfazl Razi, Jacob Chakareski, and Jonathan Ashdown. Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 835–840, 2019. 87

[2] Moulay A. Akhloufi, Andy Couturier, and Nicolás A. Castro. Unmanned aerial vehicles for wildland fires: Sensing, perception, cooperation and assistance. *Drones*, 5(1), 2021. 87

[3] Vassil N. Alexandrov, Michael Lees, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors. *Proceedings of the International Conference on Computational Science, ICCS 2013, Barcelona, Spain, 5-7 June, 2013*, volume 18 of *Procedia Computer Science*. Elsevier, 2013. 121, 127

[4] Hal E. Anderson. Aids to determining fuel models for estimating fire behavior. *Gen. Tech. Rep. INT-122. U.S.Department of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station. 22p.*, April 1982. 3

[5] Patricia L. Andrews. Current status and future needs of the behaveplus fire modeling system. *International Journal of Wildland Fire.*, 23:21–33, 2014. 4

[6] Tomàs Artés, Andrés Cencerrado, Ana Cortés, and Tomàs Margalef. Relieving the effects of uncertainty in forest fire spread prediction by hybrid mpi-openmp parallel strategies. In Alexandrov et al. [3], pages 2278–2287. 13

[7] Tomàs Artés, Andrés Cencerrado, Ana Cortés, and Tomàs Margalef. Relieving the effects of uncertainty in forest fire spread prediction by hybrid mpi-openmp parallel strategies. *Procedia Computer Science*, 18:2278–2287, 2013. 57

[8] Tomàs Artés, Andrés Cencerrado, Ana Cortés, and Tomàs Margalef. Enhancing computational efficiency on forest fire forecasting by time-aware genetic algorithms. *The Journal of Supercomputing*, 71(5):1869–1881, 2015. 60

[9] Marc Baboulin, Alfredo Buttari, Jack J. Dongarra, Jakub Kurzak, Julie Langou, Julien Langou, Piotr Luszczek, and Stanimire Tomov. Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications*, 180(12):2526–2533, 2009. 26

[10] Neil D. Bennett and Barry et alter. Characterising performance of environmental models. *Environmental Modelling & Software*, 40:1–20, 2013. 10, 26

[11] Carlos Brun, Ana Cortés, and Tomàs Margalef. Coupled dynamic data-driven framework for forest fire spread prediction. *Lecture Notes Computer Science*, 8964:54–67, 2015. 5

[12] Carlos Brun, Tomàs Margalef, Ana Cortés, and Anna Sikora. Enhancing multi-model forest fire spread prediction by exploiting multi-core parallelism. *The Journal of Supercomputing*, 70(2):721–732, 2014. 57

[13] Alfredo Buttari, Jack Dongarra, Julie Langou, Julien Langou, Piotr Luszczek, and Jakub Kurzak. Mixed precision iterative refinement techniques for the solution of dense linear systems. *The International Journal of High Performance Computing Applications*, 21(4):457–466, 2007. 25

[14] A. Espinosa C. Carrillo, T. Margalef and A. Cortés. Accelerating wild fire simulator using gpu. *International Conference on Computational Science 2019, ICCS 2019, 12-14 June 2019, Faro, Algarve, Portugal (Accepted)*, 2019. 76

[15] Copernicus atmosphere monitoring service (cams), 2019. 2

[16] Carlos Carrillo, Tomàs Artés, Ana Cortés, and Tomàs Margalef. Error function impact in dynamic data-driven framework applied to forest fire spread

prediction. In *International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA*, pages 418–427, 2016. 9

[17] Carlos Carrillo, Ana Cortés, Tomàs Margalef, Antonio Espinosa, and Andrés Cencerrado. Reducing data uncertainty in forest fire spread prediction: A matter of error function assessment. In Yong Shi, Haohuan Fu, Yingjie Tian, Valeria V. Krzhizhanovskaya, Michael Harold Lees, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science - ICCS 2018 - 18th International Conference, Wuxi, China, June 11-13, 2018 Proceedings, Part III*, volume 10862 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2018. 9

[18] Carlos Carrillo, Ana Cortés, Tomàs Margalef, Antonio Espinosa, and Andrés Cencerrado. Relevance of error function in input parameter calibration in a coupled wind field model-forest fire spread simulator. In *2018 International Conference on High Performance Computing & Simulation, HPCS 2018, Orleans, France, July 16-20, 2018*, pages 772–779. IEEE, 2018. 9

[19] Carlos Carrillo, Tomàs Margalef, Antonio Espinosa, and Ana Cortés. Accelerating wild fire simulator using GPU. In João M. F. Rodrigues, Pedro J. S. Cardoso, Jânio M. Monteiro, Roberto Lam, Valeria V. Krzhizhanovskaya, Michael Harold Lees, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science - ICCS 2019 - 19th International Conference, Faro, Portugal, June 12-14, 2019, Proceedings, Part V*, volume 11540 of *Lecture Notes in Computer Science*, pages 521–527. Springer, 2019. 31, 60

[20] Andrés Cencerrado, Tomàs Artés, Ana Cortés, and Tomàs Margalef. Relieving uncertainty in forest fire spread prediction by exploiting multicore architectures. In *Proceedings of the International Conference on Computational Science, ICCS 2015, Computational Science at the Gates of Nature, Reykjavík, Iceland, 1-3 June, 2015, 2014*, pages 1752–1761, 2015. 57

[21] Andrés Cencerrado, Ana Cortés, and Tomàs Margalef. Genetic algorithm

characterization for the quality assessment of forest fire spread prediction. *Procedia Computer Science*, 9:312–320, 2012. 5

[22] Andrés Cencerrado, Ana Cortés, and Tomàs Margalef. On the way of applying urgent computing solutions to forest fire propagation prediction. In Hesham H. Ali, Yong Shi, Deepak Khazanchi, Michael Lees, G. Dick van Albada, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Proceedings of the International Conference on Computational Science, ICCS 2012, Omaha, Nebraska, USA, 4-6 June, 2012*, volume 9 of *Procedia Computer Science*, pages 1657–1666. Elsevier, 2012. 6

[23] Joint Research Centre. European forest fire information system. http://forest.jrc.ec.europa.eu/effis/, August 2011. 31, 32, 43, 71, 103

[24] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998. 59

[25] Donato D'Ambrosio, Salvatore Di Gregorio, Giuseppe Filippone, Rocco Rongo, William Spataro, and Giuseppe A. Trunfio. A multi-gpu approach to fast wildfire hazard mapping. In *Simulation and Modeling Methodologies, Technologies and Applications - International Conference, SIMULTECH 2012 Rome, Italy, July 28-31, 2012 Revised Selected Papers*, pages 183–195, 2012. 31, 58

[26] Mónica Denham, Ana Cortés, and Tomàs Margalef. Computational steering strategy to calibrate input variables in a dynamic data driven genetic algorithm for forest fire spread prediction. *Lecture Notes Computer Science*, 5545:479–488, 2009. 5

[27] B Ebert. Forecast verification:issues, methods and faq. http://www.cawcr.gov.au/projects/verification, June 2012. 9

[28] Angel Farguell, Ana Cortés, Tomàs Margalef, Josep Ramon Miró, and J. Mercader. Data resolution effects on a coupled data driven system for forest fire propagation prediction. In Petros Koumoutsakos, Michael Lees, Valeria V.

Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland*, volume 108 of *Procedia Computer Science*, pages 1562–1571. Elsevier, 2017. 4

[29] M. A. Finney. Farsite: Fire area simulator—model development and evaluation. *Research Paper RMRS-RP-4 Revised*, 236, 1998. 4, 9, 13

[30] Farsite tutorial website, 2007. 14, 20, 22, 23, 60

[31] Fire information for resource management system, 2021. xix, 2

[32] Odegaard V. Gjertsen U. The water phase of precipitation: a comparison between observed, estimated and predicted values. http://dx.doi.org/10.1016/j.atmosres.2004.10.030, October 2005. 9

[33] Stef Graillat, Fabienne Jézéquel, Romain Picot, François Févotte, and Bruno Lathuilière. Auto-tuning for floating-point precision with discrete stochastic arithmetic. working paper or preprint, June 2016. 30

[34] Salvatore Di Gregorio, Giuseppe Filippone, William Spataro, and Giuseppe A. Trunfio. Accelerating wildfire susceptibility mapping through GPGPU. *J. Parallel Distrib. Comput.*, 73(8):1183–1194, 2013. 58

[35] Roger Viet Hoang. *Wildfire Simulation on the GPU*. PhD thesis, university of Nevada, 2008. 31, 58

[36] Nvidia jetson agx xavier developer kit, 2018. 89

[37] Nvpmodel – nvidia jetson agx xavier developer kit, 2018. 89, 90

[38] Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, Tiegang Chen, Guangxiao Hu, Shaohuai Shi, and Xiaowen Chu. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *CoRR*, abs/1807.11205, 2018. 25

[39] Fabienne Jézéquel, Jean-Luc Lamotte, and Issam Saïd. Estimation of numerical reproducibility on cpu and gpu. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 675–680, 2015. 73

[40] I Knight and J Coleman. A fire perimeter expansion algorithm-based on huygens wavelet propagation. *International Journal of Wildland Fire*, 3, 01 1993. 14

[41] Maja Kucharczyk and Chris H. Hugenholtz. Remote sensing of natural hazard-related disasters with small drones: Global trends, biases, and research opportunities. *Remote Sensing of Environment*, 264:112577, 2021. 87

[42] Jan Mandel, Jonathan D. Beezley, Adam K. Kochanski, Volodymyr Y. Kondratenko, Lin Zhang, Erik W. Anderson, Joel Daniels II, Cláudio T. Silva, and Christopher R. Johnson. A wildland fire modeling and visualization environment. *CoRR*, abs/1111.4610, 2011. 4

[43] C.M. Maynard and D.N. Walters. Mixed-precision arithmetic in the endgame dynamical core of the unified model, a numerical weather prediction and climate model code. *Computer Physics Communications*, 2019. 25

[44] National interagency fire center, 2020. 1

[45] Vasileios G. Ntinas, Byron E. Moutafis, Giuseppe A. Trunfio, and Georgios Ch. Sirakoulis. Parallel fuzzy cellular automata for data-driven simulation of wildfire spreading. *J. Comput. Science*, 21:469–485, 2017. 31, 58

[46] Nvidia toolkit documentation, November 23, 2021. 59, 69

[47] Nvidia visual profiler, 2022. 83

[48] E Pastor, L Zárate, E Planas, and J Arnaldos. Mathematical models and calculation systems for the study of wildland fire behaviour. *Progress in Energy and Combustion Science*, 29(2):139 – 153, 2003. 2

[49] Karl E. Prikopa and Wilfried N. Gansterer. On mixed precision iterative refinement for eigenvalue problems. In Alexandrov et al. [3], pages 2647–2650. 26

[50] Gwynfor D. Richards. An elliptical growth model of forest fire fronts and its numerical solution. *International Journal for Numerical Methods in Engineering*, 30(6):1163–1179, 1990. 4

[51] R. Rothermel. A mathematical model for predicting fire spread in wildland fuels, US Department of Agriculture, Forest Service, Inter- mountain Forest and Range Experiment Station Ogden, UT, USA 1972. 17

[52] Flávio A. Sousa, Ricardo J. N. dos Reis, and José C. F. Pereira. Simulation of surface fire fronts using firelib and gpus. *Environmental Modelling and Software*, 38:167–177, 2012. 31, 58

[53] A wildfire simulation toolkit for researchers and experts in the disaster resilience field, 2017. 4

[54] Copernicus atmosphere monitoring service (cams), 2015. 65

[55] A. L. Sullivan. A review of wildland fire spread modelling, 1990-present 3: Mathematical analogues and simulation models. *arXiv e-prints*, page arXiv:0706.4130, Jun 2007. 2, 4

[56] O. Tintó Prims, M. C. Acosta, A. M. Moore, M. Castrillo, K. Serradell, A. Cortés, and F. J. Doblas-Reyes. How to use mixed precision in ocean models: exploring a potential reduction of numerical precision in nemo 4.0 and roms 3.6. *Geoscientific Model Development*, 12(7):3135–3148, 2019. 30

[57] Kerstin Wendt, Ana Cortés, and Tomàs Margalef. Knowledge-guided genetic algorithm for input parameter optimisation in environmental modelling. In Peter M. A. Sloot, G. Dick van Albada, and Jack J. Dongarra, editors, *Proceedings of the International Conference on Computational Science, ICCS 2010, University of Amsterdam, The Netherlands, May 31 - June 2, 2010*, Procedia Computer Science, pages 1367–1375. Elsevier, 2010. 6