Universitat Autònoma de Barcelona

Paralelización del cálculo del campo de vientos para la predicción de la propagación de incendios forestales

Gemma Sanjuan Gómez

Escola d'Enginyeria
Departament d'Arquitectura de Computadors i Sistemes Operatius

Paralelización del cálculo del campo de vientos para la predicción de la propagación de incendios forestales

Tesis doctoral presentada por Gemma Sanjuan Gómez,
correspondiente al programa de doctorado en Informática,
realizada bajo la dirección del doctor Tomás Manuel Margalef Burrull,
para optar al grado de Doctor por la Universitat Autònoma de Barcelona

Mayo 2016

Paralelización del cálculo del campo de vientos para la predicción de la propagación de incendios forestales

Tesis doctoral presentada por Gemma Sanjuan Gómez para optar al grado de Doctor por la Universitat Autònoma de Barcelona. Esta tesis doctoral ha sido realizada en el Departament d'Arquitectura de Computadors i Sistemes Operatius de la Universitat Autònoma de Barcelona, dentro del programa de Doctorado en Informática, bajo la dirección del Dr. Tomàs Manuel Margalef Burrull.

Fecha: _____

El Doctorando: _____

El Director: _____

La investigación llevada a cabo en esta tesis doctoral ha sido financiada por el Ministerio de Ciencia e Innovación bajo contrato TIN2011-28689-C02-01 y por el Ministerio de Economía y Competitividad bajo contrato TIN2014-53234-C2-1-R.

"Todavía no se han levanto las barreras que le digan al genio:"De aquí no pasarás"."

Ludwig Van Beethoven

Abstract

Forest fires are natural disasters that every year cause significant losses. Knowing in advance their evolution is of great importance to mitigate their effects. There are several models that provide a prediction of fire propagation. The wind is a fundamental parameter of these models, but it is modified by the terrain, and it is necessary to couple wind field models. Wind field simulators, such as WidnNinja, discretize the terrain in a mesh of points and apply the corresponding equations to constitute a system of equations of the form $Ax = b$. In the particular case of WindNinja, mass conservation equations and Dirichlet boundary conditions are applied.

When the map is very large, the system of equations has hundreds of millions of unknowns and cannot be solved by direct methods, so that iterative methods are applied, such as Precondcionated Conjugate Gradient (PCG). Solving such systems takes too much time that cannot be taken in operational situations. Therefore, it is necessary to apply parallelization techniques. Specifically, three techniques have been applied to accelerate the calculation of wind field with WindNinja:

- Data parallelism has been exploited, using partition map. A methodology that determines the partition map, to reduce the execution time without losing accuracy in the calculation of wind field, beyond acceptable limits, has been developed.
- Domain decomposition has been applied. It allows parallelizing the resolution of the system of equations by applying methods with overlaping (Schwarz) or methods without overlaping (Schur).
- It has been determined that the most time consuming operation of the PCG is the sparse matrix-vector multiplication and a storage format (*Vectorizing Diagonal Sparse Matrix VDSpM*) speeding up the operation has been developed.

The three methods are able to reduce execution times, but when the maps are very large, none of the three has the scalability to reduce the time to get below 100 seconds. However, the methods developed can be integrated into a hybrid MPI-OpenMP aplication, getting achieve the excution time objectives for very large maps.

Resumen

Los incendios forestales son desastres naturales que todos los años causan cuantiosas pérdidas. Conocer de antemano su evolución resulta de gran importancia para mitigar sus efectos. Existen diversos modelos que proporcionan una predicción de la propagación del incendio. El viento es un parámetro fundamental de estos modelos, pero hay que considerar que se ve modificado por la orografía del terreno, y es necesario acoplar modelos de campo de vientos. Los simuladores de campo de vientos, como es el caso de WidnNinja, discretizan el terreno formando una malla y aplican las ecuaciones correspondientes para establecer un sistema de ecuaciones de la forma $Ax = b$. En el caso particular de WindNinja, se aplican las ecuaciones de conservación de la masa y las condiciones de contorno de Dirichlet.

Cuando el mapa es muy grande, el sistema de ecuaciones llega a tener centenares de millones de incógnitas y no puede ser resuelto por métodos directos, de modo que se aplican métodos iterativos, como es el Gradiente Conjugado con Precondcionador (PCG). Resolver tales sistemas toma un tiempo que no puede ser asumido en situaciones operacionales. Por tanto, es necesario aplicar técnicas de paralelización. En concreto, se han aplicado tres técnicas para acelerar el cálculo del campo de vientos con WindNinja:

- Se ha explotado el paralelismo de datos, aplicando partición del mapa. Se ha desarrollado una metodología que determina la partición del mapa, para reducir el tiempo de ejecución sin perder precisión en el cálculo del campo de vientos, más allá de unos límites aceptables.
- Se ha aplicado descomposición del dominio que permite paralelizar la resolución del sistema de ecuaciones aplicando métodos con solapamiento (Schwarz) o métodos sin solapamiento (Schur).
- Se ha determinado que la operación más costosa del PCG es la multiplicación matriz dispersa-vector y se ha desarrollado un formato de almacenamiento (*Vectorizing Diagonal Sparse Matrix VDSpM*) que permite acelerar dicha operación.

Los tres métodos consiguen reducir los tiempos de ejecución, pero cuando los mapas son muy grandes, ninguno de los tres presenta la escalabilidad necesaria para conseguir reducir el tiempo por debajo de 100 segundos. Sin embargo, los métodos desarrollados pueden ser integrados en una aplicación híbrida MPI-OpenMP, que consigue alcanzar los objetivos de tiempo establecidos para mapas muy grandes.

Resum

Els incendis forestals són desastres naturals que tots els anys causen quantioses pèrdues. Conèixer per endavant la seva evolució resulta de gran importància per mitigar-ne els efectes. Hi ha diversos models que proporcionen una predicció de la propagació de l'incendi. El vent és un paràmetre fonamental d'aquests models, però cal considerar que es veu modificat per l'orografia del terreny, i cal acoblar models de camp de vents. Els simuladors de camp de vents, com és el cas de WidnNinja, discretitzen el terreny formant una malla i apliquen les equacions corresponents per establir un sistema d'equacions de la forma $Ax = b$. En el cas particular de WindNinja, s'apliquen les equacions de conservació de la massa i les condicions de contorn de Dirichlet.

Quan el mapa és molt gran, el sistema d'equacions arriba a tenir centenars de milions d'incògnites i no pot ser resolt per mètodes directes, de manera que s'apliquen mètodes iteratius, com és el Gradient Conjugat amb Precondcionador (PCG). Resoldre aquests sistemes pren un temps que no pot ser assumit en situacions operacionals. Per tant, cal aplicar tècniques de paral·lelització. En concret, s'han aplicat tres tècniques per accelerar el càlcul del camp de vents amb WindNinja:

- S'ha explotat el paral·lelisme de dades, aplicant partició del mapa. S'ha desenvolupat una metodologia que determina la partició del mapa, per reduir el temps d'execució sense perdre precisió en el càlcul del camp de vents, més enllà d'uns límits acceptables.
- S'ha aplicat descomposició del domini que permet paral·lelitzar la resolució del sistema d'equacions aplicant mètodes amb solapament (Schwarz) o mètodes sense solapament (Schur).
- S'ha determinat que l'operació més costosa del PCG és la multiplicació matriu dispersa-vector i s'ha desenvolupat un format d'emmagatzematge (*Vectorizing Diagonal Sparse Matrix VDSpM*) que permet accelerar aquesta operació.

Els tres mètodes aconsegueixen reduir els temps d'execució, però quan els mapes són molt grans, cap dels tres presenta l'escalabilitat necessària per aconseguir reduir el temps per sota de 100 segons. No obstant això, els mètodes desenvolupats poden ser integrats en una aplicación híbrida MPI-OpenMP, que aconsegueix assolir els objectius de temps establerts per a mapes molt grans.

Agradecimientos

Escribiendo esta tesis me he acordado de unas palabras que dijo un buen amigo: “Escribir una tesis es un conjunto de factores y hacer una tesis, es un trabajo cansado, laborioso, metódico... y cumplir los objetivos no depende sólo de ti”. Ahora sólo puedo darle la razón y, por eso, quiero darle las gracias a todas las personas que han hecho posible esta tesis, pero especialmente, a mi director Tomás Margalef, ya que esta tesis no hubiera sido posible sin él: En primer lugar, por qué creyó en mí y me dio la oportunidad de llevar a cabo esta investigación. Pero lo que verdaderamente debo agradecerle, es que siempre me ha ayudado a ordenar y organizar mis ideas por muy caóticas que fueran, y esto ha hecho posible que sacara lo mejor de mi. Por todo esto, siempre le estaré agradecida.

A Anna Cortes por haber estado ahí ayudándome a mejorar y haber sido la parte más humana del equipo, ya que siempre hemos podido contar con ella para cualquier cosa.

A Tomás Artés y Carles Brun, por qué, desde el inicio, siempre estuvieron ahí para ayudarme. Han sido para mi los mejores compañeros que una podía esperar. Aunque ellos ya se doctoraron, nunca se lo he dicho y aprovecho este momento para darles las gracias.

A Dani y Gemma, por haber estado ahí ayudándome siempre.

A mi Familia, que siempre creyó en mi y no dejaron nunca que me rindiera. Siempre han estado ahí en mis buenos y malos momentos. Y por muy malos y difíciles que fueran nunca dejaron de creer en mi.

Y no me puedo olvidar a mis nuevos compañeros, que, aunque no entienden nada de lo que hago, siempre están ahí animándome.

Índice

Listado de Figuras

Listado de Tablas

A mis padres y a mis hermanos, porque sin ellos esta tesis no hubiera sido posible

Capítulo 1

Los incendios forestales

1.1 Introducción

Los incendios forestales son una de las peores catástrofes que sufren los bosques de todo el mundo. El problema no son sólo las hectáreas de bosque destruidas por el incendio, sino su efecto global sobre los ecosistemas en los que se desarrollan y sobre el medio ambiente en general.

Desde un punto de vista medioambiental, los incendios provocan la muerte de animales y plantas, poniendo en peligro la supervivencia de ciertas especies; generan una importante contaminación medioambiental, tanto atmosférica, por la gran cantidad de agentes contaminantes liberados a la atmósfera, como de las aguas, por los restos que van a para a ellas; favorecen la erosión del terreno por la pérdida de especies vegetales que ayudan a mantener la cohesión del suelo. En resumen, los incendios forestales conllevan la disminución de zonas proliferas y la degradación medioambiental.

Desde un punto de vista socio-económico, los incendios forestales provocan importantes pérdidas en bienes e infraestructuras, con un importante impacto económico; exigen un elevado gasto en medios y tareas de extinción; y por desgracia, en muchos casos hay que lamentar la afectación de la población por inhalación de humo e incluso, en algunos casos, la pérdida de vidas humanas.

Por todos estos factores, la lucha contra los incendios forestales de la forma más eficaz y eficiente es una necesidad de nuestra sociedad a nivel global.

Los incendios son un problema globalizado, ya que cada año se generan incendios enormes, que destruyen millones de hectáreas a nivel mundial, desde la cuenca del Mediterráneo a las llanuras de Australia, desde California a la selva amazónica o desde la jungla del sureste asiático a la sabana subsahariana.

Hay muchos factores que pueden causar o favorecen la propagación de los incendios forestales como largas estaciones secas, elevadas temperaturas, tormentas eléctricas,... pero más del 90% de los incendios son causados, de forma consciente o inconsciente, por la actuación humana.

Por ello, el primer punto para luchar contra los incendios forestales se basa en una buena concienciación de los ciudadanos de los riesgos asociados a los incendios y a las actividades que pueden desencadenarlos, pero aun así los incendios siguen siendo un fenómeno "natural" cuya propagación se ve favorecida por largos periodos de sequía, temperaturas extremas, acumulación de combustible (masa forestal), fuertes vientos, etc.

En este contexto, el cambio climático al que nos enfrentamos todavía empeora más la situación, ya que provoca un aumento de la temperatura y conlleva periodos de sequía, que ayudan a la propagación de los incendios forestales. Pero a su vez, los incendios forestales, son también una importante fuente de cambio climático, ya que emiten grandes cantidades de carbono (monóxido y dióxido de carbono) a la atmósfera y aumentan el efecto invernadero. Más aún, las zonas quemadas tienden a desertizarse, con lo que la función que realizaban las plantas de fijar carbono de la atmósfera mediante la fotosíntesis, ya no se puede realizar, incrementando todavía más el efecto invernadero y el cambio climático.

En España, el fenómeno de los incendios forestales viene siendo uno de los problemas ecológicos más significativos debido al clima mediterráneo de buena parte del territorio, el tipo de vegetación, el abandono de muchas tierras de cultivo, etc. En este sentido, se han realizado importantes campañas de concienciación de la población, pero desgraciadamente todavía se siguen produciendo incendios por conductas o actos negligentes, como el lanzamiento de colillas por la ventanilla del coche, que acaban provocando incendios, a menudo, difíciles de controlar. De hecho, en los últimos años se viene observando que, en la mayoría de los casos, la cantidad total de hectáreas quemadas en España y en Catalunya se concentra en unos pocos incendios forestales. Es decir, hay un gran número de incendios que se consiguen controlar en pocas horas y acaban quemando pocas hectáreas, mientras que otros se dan en unas condiciones extremas de temperatura, viento, vegetación, orografía y dan lugar a grandes incendios que acumulan miles de hectáreas quemadas, como fue el caso del incendio del Alt Emporda (cerca de La Jonquera) en Julio de 2012, que calcinaron más de 13.000 hectáreas y provocaron la muerte de 4 personas, así como 3 heridos muy graves y 6 heridos graves. En la Figura 1.1 se muestra la virulencia de las llamas durante ese incendio de julio de 2012.

En este contexto, la actuación para la rápida extinción de los incendios forestales se convierte en una cuestión crucial que requiere el uso de los medios disponibles de la

FIGURA 1.1: Incendio forestal cerca de La Jonquera (Julio 2012)

mejor manera posible, para mitigar, en la medida de lo posible, las consecuencias de los incendios, minimizando los daños medioambientales y las pérdidas materiales, así como, evitando desgracias personales. Esto implica la gestión, por parte de la autoridades y centros de control, de los medios humanos y de los recursos materiales involucrados en las tareas de extinción. Sin embargo, para poder organizar los medios disponibles del mejor modo posible, es necesario disponer de información fiable sobre el comportamiento y evolución de los incendios forestales. Esta información sobre el comportamiento de los incendios debe, pues, incorporarse para su integración en sistemas de ayuda a la toma de decisiones ("Decision Support System - DSS"), para que los gestores de los centros de control dispongan de una ayuda fiable a la hora de la toma de decisiones.

1.2 Modelos de propagación y simuladores

Para predecir el comportamiento de los incendios forestales y su evolución, se ha investigado en modelos de propagación de incendios forestales, que tomando como parámetros de entrada el entorno y las condiciones en las que se desarrolla el incendio, proporcionan una predicción de su propagación a lo largo del tiempo. En la literatura podemos encontrar modelos semi-empíricos, como los propuestos en [1], [2], [3], [4] y [5], entre otros, o modelos basados en principios físicos de la propagación de los incendios, como los propuestos en [6], [7] y [8]. Pero, de entre todos estos modelos, el que ha tenido

mayor éxito y ha constituido el núcleo central de la mayoría de las aproximaciones es el modelo propuesto por Rothermel [9] con aportaciones de Albini [10].

En el modelo de propagación de Rothermel, la vegetación se clasifica en 13 modelos, correspondientes a modelos de vegetación propios de Norte América. La vegetación existente en el terreno donde se desarrolla el incendio debe clasificarse de acuerdo a esos 13 modelos. La descripción precisa de los 13 modelos de combustible puede consultarse en [11].

Estos modelos de propagación se implementan en aplicaciones computacionales, denominadas simuladores, que permiten obtener los resultados de la predicción de una forma mucho más rápida que si hubiera que calcularse de forma manual. Algunos de estos simuladores son: BEHAVE [12], CARDIN [13], FARSITE [14], FireStation [15], BehavePlus [16][17], PHOENIX RapidFire [18] o Wildfire Analyst [19]. De entre todos estos simuladores, el que ha tenido una mayor difusión entre la comunidad y ha sido considerado como simulador operacional, ha sido FARSITE [14], desarrollado en el Rocky Mountain Research Station del Missoula Fire Sciences Laboratory en Missoula (Montana). Este simulador incorpora como núcleo de simulación el modelo de propagación propuesto por Rothermel [9].

Estos modelos de propagación requieren un conjunto de parámetros de entrada, que describe el entorno y las condiciones en las que se desarrolla el incendio. Aunque cada modelo o simulador requiere unos parámetros de entrada específicos, y en un formato particular, en general, pueden clasificarse en unos pocos tipos de parámetros:

- Mapa digital de elevación del terreno.
- Mapa de combustible con los modelos de vegetación existentes en cada punto del terreno.
- Condiciones meteorológicas.
- Condiciones de la vegetación.
- Perímetro inicial del incendio.

El simulador toma los parámetros de entrada proporcionados y genera la evolución del frente del incendio para los instantes de tiempo posteriores. Este proceso se representa en la Figura 1.2.

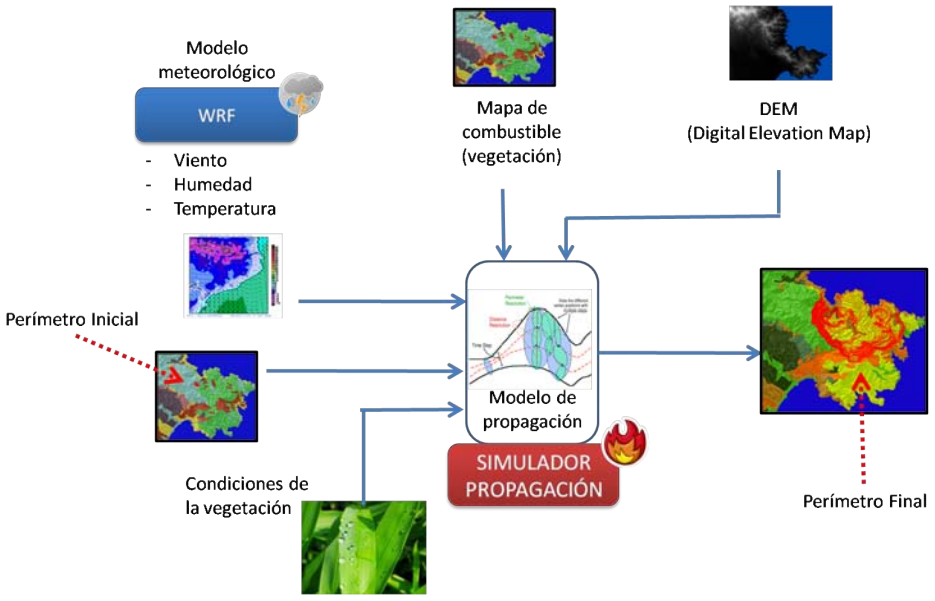


FIGURA 1.2: Esquema de predicción de la propagación de incendios forestales

1.2.1 Variabilidad de los parámetros de entrada

Estos parámetros de entrada pueden clasificarse en función de su variabilidad a lo largo del tiempo:

- Parámetros estáticos: Son aquellos parámetros que permanecen constantes a lo largo del tiempo. Entre ellos podemos destacar:

  1. Mapa digital de elevación: Describe la orografía del terreno en el que se desarrolla el incendio forestal. La resolución de dicho mapa afecta de modo muy significativo a los resultados de la predicción, ya que cuando la resolución es baja, la orografía se ve suavizada y se pierde precisión, hecho que en algunos casos puede resultar determinante. Hoy en día es posible conseguir mapas digitales de elevación de diversas fuentes y repositorios públicos, como es el caso de ASTER-GDEM de la NASA [20] que ofrece un mapa digital global de elevación a una resolución de 30 metros. En la Figura 1.3 se muestra un recorte del mapa digital de ASTER correspondiente a la zona del Mediterráneo.

- Parámetros quasi-estáticos: Son aquellos parámetros, que aunque sufren variación a lo largo del tiempo, su escala temporal es lo suficientemente amplia como para ser considerados constantes durante la evolución del incendio.

  1. Mapa de combustible o vegetación: El mapa de vegetación correspondiente al terreno en el que se desarrolla el incendio forestal, incluye los modelos

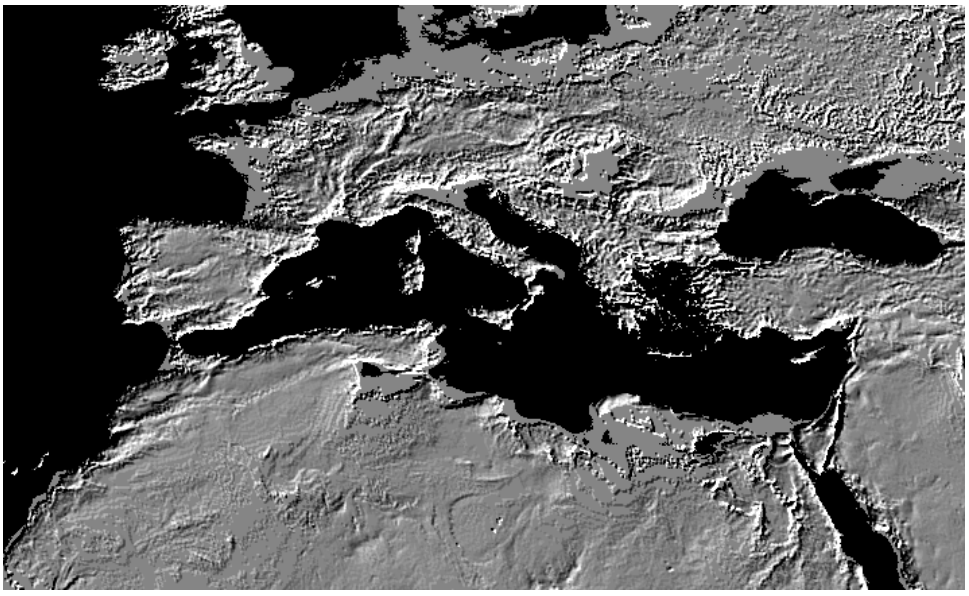


FIGURA 1.3: Recorte del mapa digital global de elevación proporcionado por la NASA

de vegetación existentes en cada punto o celda del terreno. Este modelo de combustible varía según la evolución propia de la vegetación, pero esos cambios se van produciendo de un modo gradual. La vegetación va creciendo, se van separando las copas y el sotobosque, se producen periodos de sequía que hacen que desaparezcan determinados tipos de vegetación, etc. Por otro lado, hay otros factores que hacer cambiar los modelos de vegetación. Por ejemplo, el abandono de tierras de cultivo provoca su progresiva transformación en bosques que hace que varíen los modelos de combustible presentes. Pero, todos estos cambios requieren una escala temporal mucho mayor que el tiempo de propagación del incendio, con lo cual se suele considerar que los modelos de vegetación son constantes durante el tiempo de propagación del incendio. Estos mapas de vegetación, se pueden obtener de repositorios públicos, como puede ser el inventario forestal de Catalunya, elaborado por el Centre de Recerca Ecològica i Aplicacions Forestals (CREAF) [21]. En la Figura 1.4 se muestra la imagen correspondiente al inventario forestal de la comarca del Ripollés (Catalunya). Los mapas elaborados en este inventario tienen una resolución de 1 kilómetro.

Otra fuente más genérica que proporciona los mapas de combustible es el proyecto CORINE Land Cover que clasificó el suelo de Europa según 44 tipos de uso del mismo [22]. A partir de esta información se puede realizar un ajuste para asimilar los modelos de uso del suelo a modelos de vegetación. Los datos de CORINE Land Cover se obtuvieron a partir de imágenes de satélite con una resolución de píxel de 100 metros. La Figura 1.5 muestra la

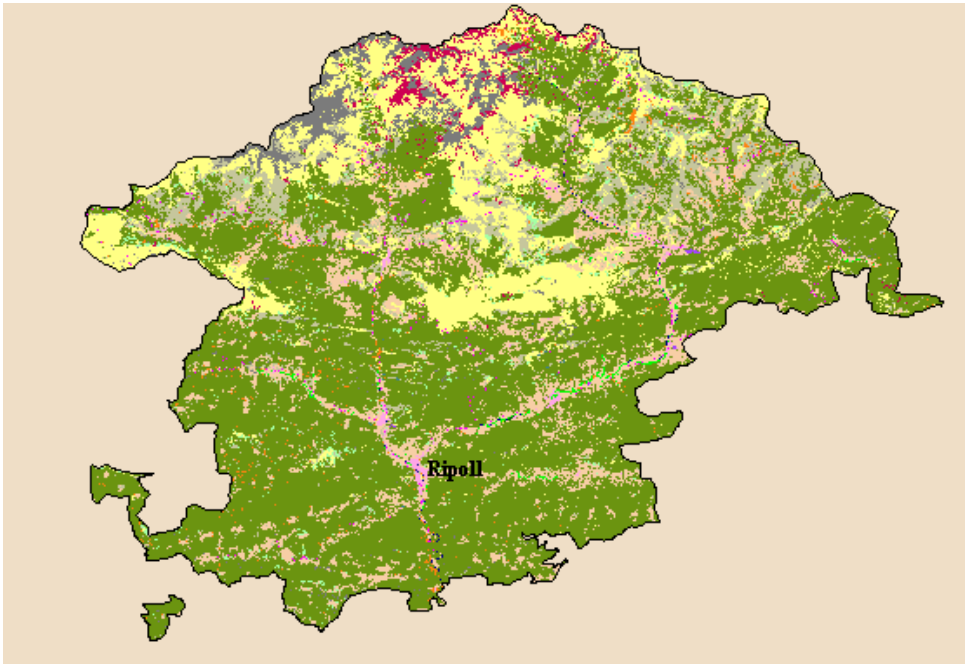


FIGURA 1.4: Inventario Forestal correspondiente a la comarca del Ripollés (Catalunya)

zona del Mediterráneo según la clasificación de los usos del suelo del proyecto CORINE.

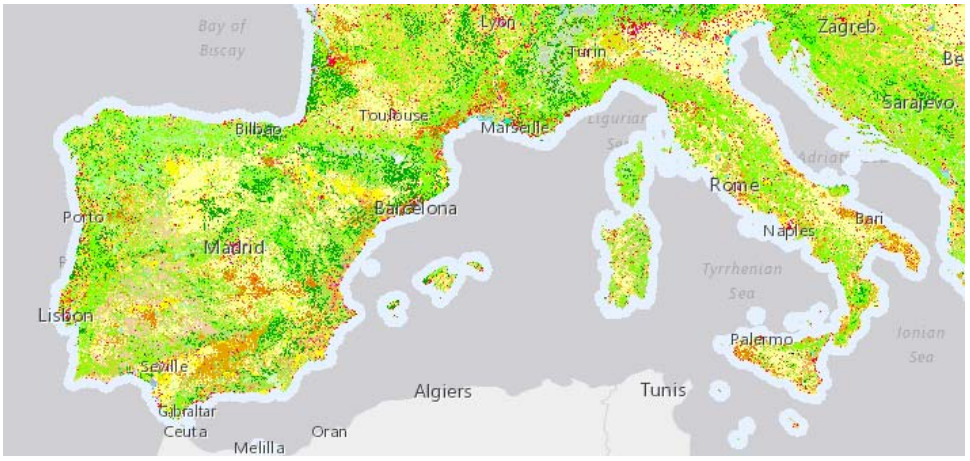


FIGURA 1.5: Mapa de usos del suelo correspondiente al Mediterráneo occidental

En los últimos años, se han desarrollado técnicas, como es el caso de la tecnología LIDAR (*Laser Imaging Detection and Ranging*) que permite obtener un mapa estratificado de la vegetación, y no se limita a la vista zenital, de modo que los mapas de vegetación realizados utilizando esta tecnología incorporan para cada punto 8 características de la vegetación con una resolución de 20 metros. Sin embargo, estas imágenes deben tomarse con vuelos muy precisos, que generan gran cantidad de datos que deben ser preprocesados para poder ser utilizados. Esto implica muchas horas de vuelo para cubrir

grandes extensiones de terreno, con lo cual el proceso de adquisición de las imágenes y procesado de las mismas se vuelve costoso, de modo que aún no es frecuente el uso de dicha información en los modelos de vegetación utilizados en los modelos propagación de incendios forestales.

En cualquier caso, estos datos de los uso del suelo o modelos de combustible a gran escala son tomados y actualizados con escasa frecuencia. La recolección, procesamiento y compresión de los datos son tareas costosas, en tiempo y esfuerzo, que requieren una buena planificación y metodología estrictas. Este hecho, combinado, con la relativamente evolución de los usos del suelo, hace que se actualice el inventario forestal o mapa de usos del suelo tras varios años (10 o 15 años), de modo que es posible que los mapas de modelos de combustible no se encuentren actualizados en el momento de ocurrencia de un incendio. La mejora en los métodos automáticos de captación de datos procedentes de imágenes de satélite de alta resolución va a favorecer el uso de datos más actualizados y de mayor precisión.

2. Características de la vegetación: La vegetación varía sus características, especialmente su contenido hídrico, según las condiciones meteorológicas, las lluvias, la humedad atmosférica, el grado de insolación, etc., de modo que un mismo modelo de vegetación va a arder de una u otra manera en función de la humedad interna del combustible. Esta humedad de la vegetación varía de una forma relativamente rápida, pero presenta una escala de tiempo mayor que el tiempo de propagación del incendio. Por ello, los distintos modelos de propagación requieren unos valores relativos al contenido hídrico o humedad del combustible. Hay que tener en cuenta, que en este caso el tiempo de variación de los valores es mucho más corto que el caso de los modelos de uso del suelo o combustible. El uso de índices que permiten estimar el estado de la vegetación, como puede ser el NDVI (Normalized Difference Vegetation Index), a partir de imágenes aéreas o de satélite pueden favorecer una mejor estimación de estos valores de los parámetros referentes al estado de la vegetación [23]. Otra posibilidad, para estimar el estado de la vegetación, consiste en utilizar modelos biológicos que intentan estimar el contenido hídrico de las plantas, dadas unas condiciones climatológicas [24].

- Parámetros dinámicos: Son aquel conjunto de parámetros que experimentan una variación rápida de su valor a lo largo del desarrollo del incendio.

1. Variables meteorológicas: Entre los parámetros dinámicos destacan de forma muy significativa las variables meteorológicas. La temperatura ambiente, la humedad relativa del aire, y, muy especialmente, la dirección y la velocidad del

viento influyen en la propagación de los incendios forestales. Es bien sabido que estas variables van variando a lo largo de las horas, y en algunos casos, como la velocidad del viento, los cambios se producen en cuestión de minutos. Estas variables pueden obtenerse de estaciones meteorológicas situadas cerca del lugar en el que se ha declarado un incendio, pero si se quiere realizar una predicción de la propagación de un incendio en las próximas horas, necesitamos saber de antemano el valor de las variables meteorológicas, con lo cual, no es posible utilizar las medidas de una estación meteorológica, sino que deben tomarse los valores proporcionados por modelos meteorológicos, como WRF (*The Weather Research and Forecasting model*) [25]. Estos modelos meteorológicos son ejecutados en centros meteorológicos que pueden proporcionar resultados a distintas resoluciones espaciales y temporales. Habitualmente la resolución espacial que utilizan es de algunos metros y la proporcionan salida cada hora con un horizonte confiable de unas 36 horas. Incrementar la resolución espacial y temporal comporta un aumento en los tiempos de cómputo que hacen que sea completamente inoperante.

2. Frente del incendio: Otro parámetro de entrada necesario es el estado del incendio en el momento a partir del cual se desea realizar la predicción. Evidentemente, el incendio va evolucionando y el estado inicial que consideremos para realizar la predicción debe corresponderse con el estado real del frente. Si en un instante posterior, deseamos iniciar otra predicción, necesitaremos conocer el nuevo estado del frente del incendio. Información relativa al estado del incendio en un instante determinado puede obtenerse a partir de imágenes de satélite, imágenes o datos proporcionados por medios aéreos, datos proporcionados por les efectivos de campo, etc. Sin embargo, la mayoría de las fuentes presentan distintos inconvenientes. Así, por ejemplo, los satélites suelen tener una frecuencia de paso relativamente elevada, con lo cual no es factible disponer de una información actualizada del estado del frente de una forma más o menos regular. Además, ciertos satélites no captan imágenes hiperespectrales, con lo que el frente puede quedar enmascarado por nubes ambientales o por el propio humo del incendio. Así por ejemplo, los satélites Aqua y Terra de la NASA captan imágenes MODIS (*Moderate Resolution Imaging Spectroradiometer*) con una frecuencia de paso próxima a las 24 horas [26]. A su vez, las imágenes obtenidas de medios aéreos ofrecen problemas de georeferenciación ya que hay que tener en cuenta la inclinación del avión, de la cámara, del terreno, etc. para poder posicionar de forma efectiva el frente del incendio. Por otra parte, la información proporcionada por los medios terrestres, también adolece de problemas de georeferenciación. En

la Figura 1.6 se muestra el perímetro generado a partir de un vuelo con un aeronave, proyectado sobre el mapa topográfico del terreno.

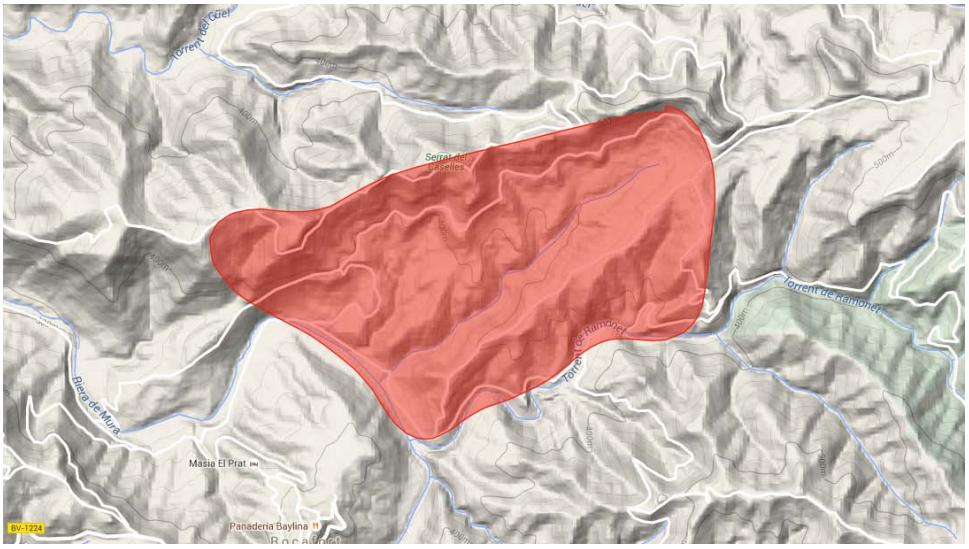


FIGURA 1.6: Perímetro de un incendio forestal visualizado sobre el mapa topográfico del terreno

Además de esta variabilidad temporal, algunos parámetros presentan una variabilidad espacial, mientras que otros tienen un valor uniforme a lo largo de todo el terreno. Así, por ejemplo, se puede considerar que la temperatura es uniforme en todos los puntos del terreno (si bien es cierto que el grado de insolación puede ser diferente en unas partes que en otras), o las características de un mismo modelo de vegetación son las mismas para todas las zonas del mapa con ese modelo de vegetación.

Otros parámetros varían de un punto a otro. Por ejemplo, el mapa de vegetación suele incluir distintos modelos de vegetación, aunque esta es una variación conocida a priori. Sin embargo, aparecen parámetros de los que se puede tener una medida o estimación para un punto del terreno, que no sea valida para otros puntos del terreno. El caso más paradigmático es el caso del viento. Tal como se ha comentado anteriormente, los modelos meteorológicos proporcionan un valor de dirección y velocidad del viento sobre una malla de una resolución de algunos kilómetros. Este viento meteorológico se ve modificado por la orografía del terreno, de modo que las colinas, valles, desfiladeros y montículos hacen que el viento sea distinto en todos los puntos del terreno cuando se considera un mapa de una resolución de 30 metros.

Todos estos factores generan un alto grado de incertidumbre en los parámetros de entrada de los modelos de propagación de incendios forestales que provocan que los resultados de las predicciones no sean completamente fiables y, a menudo, incorporen un cierto grado de discordancia con la propagación real del incendio. Esta discordancia puede resultar determinante para conseguir mitigar o no los daños y pérdidas causadas por los

incendios. Por lo tanto, resulta de crucial importancia poder reducir la incertidumbre de los parámetros de entrada.

Con el fin de superar estos inconvenientes, se propuso una metodología de predicción de la propagación de los incendios forestales basado en dos etapas. En esta metodología, se introduce una etapa de calibración de los parámetros que presentan una mayor incertidumbre [27].

1.2.2 Predicción en dos etapas

Para llevar a cabo la calibración de los parámetros de entrada, inicialmente se analiza la evolución del incendio durante un intervalo de tiempo (de t_0 a t_1) y se lleva a cabo la etapa de calibración de parámetros en la que se aplica un algoritmo de optimización que determina los valores de los parámetros que mejor se ajustan al comportamiento del incendio real.

Como estrategias de calibración o ajuste de los parámetros se pueden utilizar diversas formas como pueden ser los algoritmos genéticos [27][28], métodos de *tabu search*, métodos estadísticos [29] o combinaciones de diversos métodos [30].

En el caso de los algoritmos genéticos, se genera una población inicial en la que cada individuo está formado por un conjunto de valores de los parámetros que representan un escenario de la propagación del incendio. Para cada individuo se lleva a cabo una predicción del comportamiento del incendio en el intervalo t_0 - t_1 y se compara el resultado con la propagación del frente real del fuego en el instante de tiempo t_1.

Los individuos son ordenados en función del error entre el resultado de la simulación y la evolución real del incendio. La función de error que se utiliza es la diferencia simétrica entre la predicción y el comportamiento real[31]. Con la población ordenada se aplican los operadores genéticos de elitismo, selección, cruzamiento y mutación para obtener una nueva generación de individuos.

Este proceso se repite un cierto número de iteraciones y al final de este proceso de calibración de parámetros, el individuo con el mejor resultado se utiliza para llevar a cabo la predicción de la propagación del incendio entre los instantes t_1 y t_2 [32]. Este proceso se muestra en la Figura 1.7.

El hecho de que se deba realizar una predicción para cada individuo de la población y se deban realizar un cierto número de iteraciones para que converja el algoritmo genético implica que el método de calibración puede llevar un tiempo sustancial. Sin embargo, el hecho de tener que realizar una *predicción* efectiva requiere que el tiempo de

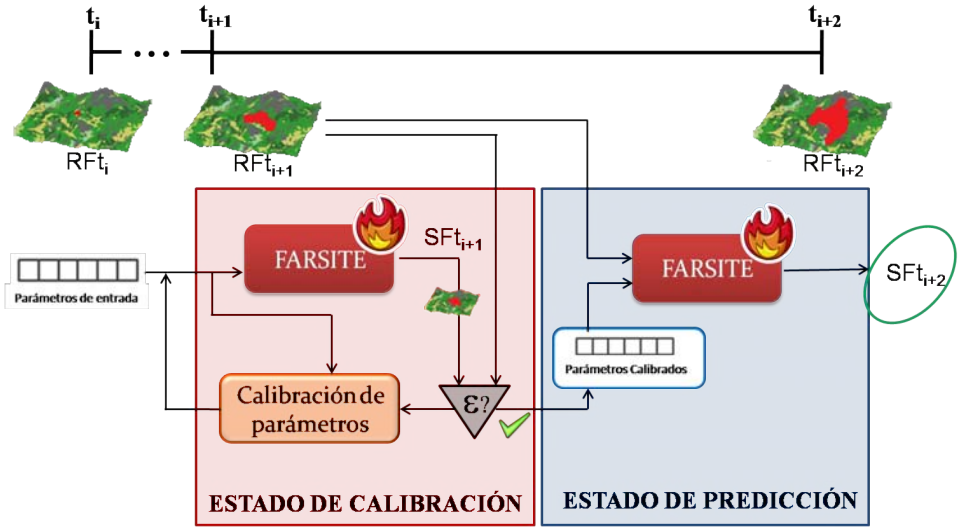


FIGURA 1.7: Método de predicción en dos etapas

la misma sea lo suficientemente pequeño como para que se puedan tomar las decisiones y acciones adecuadas en función de los resultados de la predicción. Por tanto, es necesario utilizar métodos de computación de altas prestaciones para mantener el tiempo de predicción acotado dentro de los márgenes deseados. Así pues, se puede explotar el cómputo paralelos y la propia estructura de los algoritmos genéticos para cumplir con los objetivos. El esquema del algoritmo genético encaja de forma natural en el paradigma Master/Worker [33][34]. El proceso Master genera la población inicial y envía los individuos a los procesos worker. Los workers realizan la evaluación de cada uno de los individuos y devuelven el error obtenido al Master que, a su vez, aplica los operadores genéticos para alcanzar la siguiente población. Este esquema ofrece la ventaja de que los workers pueden ejecutarse de forma simultánea, sin que haya ningún intercambio de información entre ellos. Los únicos intercambios de información que se producen son los que se llevan a cabo del Master a los workers, para distribuir los individuos, y de los workers al Master, para devolver el error de cada individuo. Este método implica la necesidad de disponer de un número relativamente elevado de recursos computacionales para satisfacer los requerimientos de tiempo. Se han desarrollado técnicas para utilizar los recursos disponibles de la manera más eficiente, aprovechando el paralelismo interno de la simulación de la propagación de los incendios forestales [35].

Este método de predicción en dos etapas se basa en la hipótesis que las condiciones en las que se desarrolla el incendio forestal, no varían drásticamente entre la etapa de calibración y la etapa de predicción. Así, el método funciona perfectamente para fuegos sintéticos o prescritos, donde todos los factores del fuego se encuentran controlados, y por tanto ni las condiciones meteorológicas ni las condiciones de terreno varían de forma notable durante el tiempo de propagación. Sin embargo, este método presenta

ciertas limitaciones, ya que se supone que los parámetros de entrada no varían con el tiempo y, por otro lado, supone que los parámetros son uniformes en todo el terreno. En consecuencia, las dos restricciones más significativas son la uniformidad espacial de los parámetros y la sensibilidad a los cambios de dichos parámetros a lo largo del tiempo.

Tal como se ha comentado anteriormente, las variables meteorológicas, y, en particular, el viento, presentan una gran variabilidad tanto temporal como espacial, con lo cual, es necesario introducir modelos complementarios que representen la evolución temporal y la distribución espacial del mismo.

1.2.3 Acoplamiento de modelos complementarios al modelo de propagación

La metodología de predicción en dos etapas requiere unas condiciones de uniformidad espacial y de constancia temporal en los valores de los parámetros para poder contrarrestar, de una manera satisfactoria, la incertidumbre propia de los valores de los parámetros. Pero, tal como hemos visto en los puntos anteriores, parámetros como el viento (con velocidad y dirección), presentan una rápida variación temporal, debida a las fluctuaciones meteorológicas, y una clara distribución espacial, debida al efecto de la orografía propia del terreno sobre el viento meteorológico. Para tomar en consideración estos grados de variabilidad espacial y temporal, y poder minimizar sus efectos, es necesario introducir y acoplar al modelo de propagación de incendios forestales, modelos meteorológicos y de campo de vientos que tengan en cuenta los mencionados efectos. Por otra parte, es necesario mencionar el hecho que la dirección y velocidad del viento, junto con la pendiente del terreno, son los parámetros que más influyen en la propagación de los incendios forestales [36], de modo que determinar los valores de estos parámetros de la forma más precisa posible permite mejorar la calidad de la predicción de la propagación.

- Para tomar en consideración la evolución temporal de las variables meteorológicas, es necesario, tomar los datos procedentes de modelos meteorológicos para utilizarlos a la hora de realizar las predicciones de la propagación de los incendios. Estos modelos son ejecutados por los centros meteorológicos y se pueden asimilar los datos proporcionados por estos modelos en el modelo de propagación. En el caso concreto de la metodología de predicción en dos etapas, durante la fase de calibración de los parámetros, es posible tomar como datos de entrada, tanto las predicciones de los modelos meteorológicos, como las medidas de las estaciones, ya que cuando se lleva a cabo la calibración de los parámetros estos datos ya están disponibles. Sin embargo, en la fase de predicción, las medidas de las estaciones no están disponibles, ya que se refieren a instantes venideros, con lo cual, es obligatorio utilizar datos procedentes de los modelos meteorológicos.

- Por lo que se refiere a la distribución espacial, es necesario considerar que tanto las estaciones meteorológicas, como los modelos meteorológicos, proporcionan valores a una resolución muy baja que no toma en consideración el efecto de la orografía del terreno. Por lo tanto, para poder disponer de los valores de la dirección y velocidad del viento a la resolución deseada (por ejemplo, 30 metros), debe acoplarse un modelo de campo de vientos que tome los valores proporcionados por los modelos meteorológicos y los utilice para calcular en valor del viento en cada punto del terreno, con la resolución deseada.

Acoplando estos modelos complementarios en la metodología de predicción de la propagación de incendios forestales en dos etapas [37], el esquema operativo queda como se muestra en la Figura 1.8. En esta figura, se observa que los datos meteorológicos, ya no son calibrados como parámetros participantes en el algoritmo genético, sino que son proporcionados por los modelos complementarios.

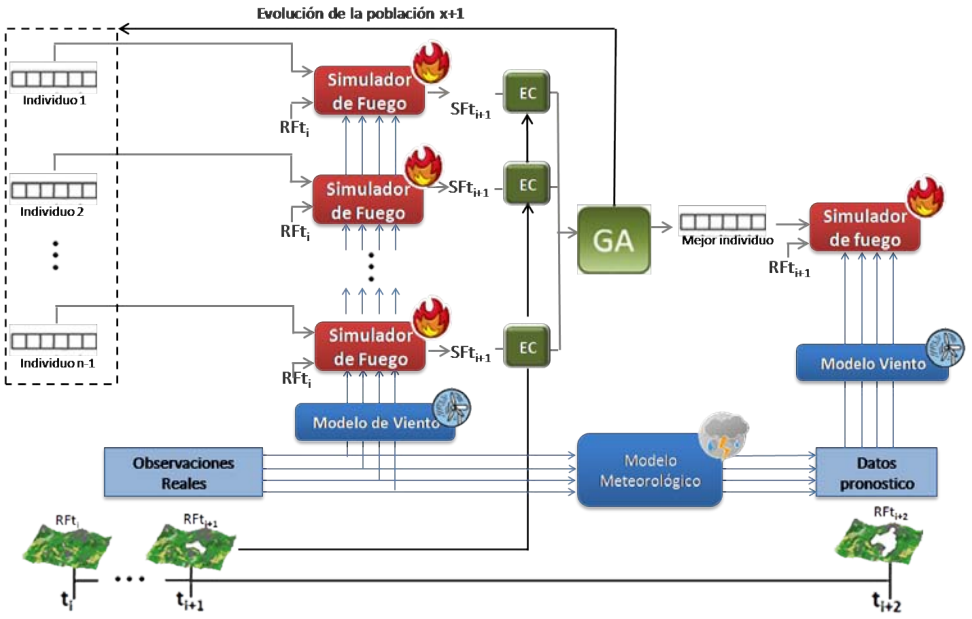


FIGURA 1.8: Método de predicción en dos etapas con modelos meteorológico y de campo de vientos acoplados

El modelo meteorológico más comúnmente utilizado hoy en día por los centros internacionales es WRF (*The Weather Research & Forecasting Model*) [25]. A su vez, el simulador de propagación más extendido entre la comunidad es FARSITE [38]. Como simulador de campo de vientos, se ha seleccionado en este trabajo, WindNinja [39][40], por haber sido desarrollado en el mismo laboratorio que FARSITE, y permitir una conexión directa de ambos simuladores.

Así pues, el esquema funcional descrito en la Figura 1.8 implica la toma de datos meteorológicos proporcionados por los centros correspondientes, la ejecución del modelo

de campo de vientos (en nuestro caso, WindNinja) para cada valor de velocidad y dirección del viento proporcionada por el modelo de predicción meteorológica, y la posterior ejecución del simulador de propagación de incendios forestales (en nuestro caso, FARSITE), y todo ello ejecutado para una población de un cierto número de individuos e iterado por un cierto número de iteraciones. Así pues, se puede vislumbrar que el método de calibración y predicción requiere el uso de técnicas de computación de altas prestaciones para poder ser operativo en tiempo real. Por tanto, es necesario analizar cada uno de los componentes del mismo y determinar la manera de ejecutarlo que permita cumplir las restricciones temporales derivadas del funcionamiento operacional.

En este contexto, esta tesis doctoral se ha centrado en llevar a cabo un estudio exhaustivo del simulador de vientos WindNinja para reducir su tiempo de ejecución mediante la aplicación de técnicas de paralelismo y computación de altas prestaciones. La metodología utilizada, en esta tesis doctoral, es aplicable a muchas otras aplicaciones de diversos campos de la ciencia y la ingeniería.

1.3 Objetivos de la tesis doctoral

El objetivo general de esta tesis doctoral consiste en la reducción del tiempo de ejecución del simulador de campo de vientos WindNinja mediante la aplicación de técnicas de paralelismo y computación de altas prestaciones. Este es un simulador que se basa en la conservación de la masa para describir el flujo de aire generado en cada punto del terreno, a partir de un viento atmosférico. Para ello, el terreno se discretiza, formando una malla irregular, y se aplican las ecuaciones de conservación de la masa, con las restricciones pertinentes, para formar un sistema de ecuaciones de grandes dimensiones, que debe resolverse utilizando algún método de resolución.

Para conseguir este objetivo general, se plantea el trabajo a diferentes niveles:

1. Particionado del mapa digital de elevaciones: Esta aproximación representa un particionado del problema, para la resolución de problemas más "*pequeños*" que pueden resolverse en un tiempo mucho menor en paralelo. Esto implica, montar un sistema de ecuaciones para cada partición del terreno y la resolución de cada partición. Sin embargo, las características del problema implican un conjunto de restricciones que han de ser tenidas en cuenta para conseguir que el particionado del problema permita alcanzar una solución muy próxima y equivalente a la solución del problema original.

2. Descomposición del dominio: Aunque pudiera parecer que el objetivo es el mismo que el del punto anterior, la aproximación es sustancialmente diferente. En este

caso, se monta la malla que describe el terreno y el sistema de ecuaciones correspondiente, y lo que se pretende es dividir la malla (o el sistema de ecuaciones equivalente) para resolverlos por métodos de descomposición del dominio, que permiten resolver el sistema de ecuaciones por subdominios.

3. Reducción del tiempo de ejecución del método de resolución (*solver*). A más bajo nivel, el cálculo del campo de vientos implica la resolución de un sistema de ecuaciones, mediante la utilización de un solver, como es el gradiente conjugado con precondicionador (*Preconditioned Conjugate Gradient* PCG). Este es un solver iterativo que va aproximando la solución mediante un conjunto de operaciones algebraicas con matrices de grandes dimensiones. Sin embargo, las matrices que describen el sistema son matrices dispersa con una muy baja densidad, y en este caso, los métodos de almacenamiento, acceso y operación con estas matrices determinan el tiempo de resolución del sistema de ecuaciones. Por tanto, es necesario adaptar los métodos de almacenamiento y operaciones a las características de las matrices generadas por WindNinja para poder reducir el tiempo de ejecución y mejorar la escalabilidad.

4. Sistema híbrido: las aproximaciones comentadas en los puntos anteriores no son excluyentes, sino que pueden combinarse para conseguir aprovechar las ventajas de las distintas técnicas y reducir el tiempo de ejecución, haciendo el uso más eficiente posible de los recursos disponibles.

1.4 Organización de la tesis doctoral

En este capítulo se ha introducido el problema de la predicción de la propagación de los incendios forestales y las restricciones temporales impuestas por los requisitos operacionales. Asimismo, se han descrito los parámetros necesarios para poder realizar la predicción y la necesidad de incorporar modelos complementarios para reducir la incertidumbre de los parámetros de entrada, como es el caso de la dirección y velocidad del viento, y mejorar la calidad de la predicción. En concreto, se plantea el uso del simulador de campo de vientos WindNinja para ajustar el viento en cada punto del terreno a partir del viento meteorológico. Como se ha descrito en la sección anterior, el objetivo que se persigue consiste en la paralelización y reducción del tiempo de ejecución de dicho simulador de campo de vientos.

La presente tesis doctoral se ha organizado de la siguiente manera:

- En el capítulo 2, se explican las ecuaciones que gobiernan el movimiento del aire y que aplica WindNinja. También se expone, el método de discretización de la

malla y de generación del sistema de ecuaciones. Una vez que se ha generado el sistema de ecuaciones es necesario estudiar el método de almacenamiento de las matrices y los métodos de operación que utiliza WindNinja para aplicar el solver del Gradiente Conjugado con Precondicionador. Este capítulo va a proporcionar una visión general de la complejidad del sistema que abordamos.

- En el capítulo 3, se introduce el particionado del mapa del terreno como método para paralelizar el cálculo del campo de vientos. Este método implica el particionado del mapa del terreno para resolver el campo de vientos para cada parte del mapa en cuestión. Sin embargo, debido a las condiciones de contorno, es necesario incluir un cierto grado de solapamiento entre las distintas partes para conseguir un campo de vientos lo más parecido posible al campo de vientos generado a partir del mapa completo. En este capítulo, se describe la metodología necesaria para conseguir un campo de vientos lo más parecido posible al campo de vientos correspondiente al mapa completo, y se presentan los resultados, en cuanto a tiempo de ejecución, diferencia en el campo de vientos y diferencia en al predicción de la propagación.

- En el capítulo 4, se describe el método de descomposición del dominio para conseguir paralelizar la resolución del campo de vientos mediante la descomposición en subdominios. En esta aproximación, se describen en la literatura dos métodos un tanto diferentes: Por un lado, la descomposición del dominio sin solapamiento (Método de Schur) y, por otro, la descomposición del dominio con solapamiento (Método de Schwarz). Se describen y se aplican ambos métodos al cálculo del campo de vientos y se analizan los resultados obtenidos con tiempo de ejecución, SpeedUp y escalabilidad.

- En el capítulo 5, se lleva a cabo un estudio del método de resolución del sistema de ecuaciones que aplica WindNinja, el gradiente conjugado con precondicionador (PCG). En este estudio, se analizan las características de la matrices propias de WindNinja, los métodos de almacenamiento de las mismas, las operaciones involucradas en la resolución del problema, los precondicionadores utilizados, etc. y se diseñan o adaptan nuevos métodos para conseguir que dicho proceso se ejecute en un menor tiempo y alcance una mejor escalabilidad.

- En el capítulo 6, se diseña una aplicación híbrida que combina las técnicas expuestas en los capítulos anteriores. De este modo, se pretende conseguir aunar las ventajas de cada una de ellas y poder extender la escalabilidad de la aplicación más allá de la escalabilidad de cada una de las aproximaciones por separado. Se describe el método que permite combinar las distintas aplicaciones y se analizan los resultados alcanzados con distintas configuraciones.

- En el capítulo 7, se presentan las conclusiones de la tesis doctoral y se analizan las líneas abiertas que deja la tesis para su posible continuación.

Capítulo 2

WindNinja

2.1 Introducción

WindNinja es un simulador de campo de vientos que, a partir de un viento meteorológico y un mapa de elevaciones, calcula el módulo de la velocidad, y las direcciones y sentidos del viento para cada punto del mapa de elevaciones introducido a la resolución deseada.

Este simulador se basa en la resolución de las ecuaciones diferenciales que describen el movimiento del aire en la atmósfera. Concretamente, en las ecuaciones de conservación de la masa, inicializadas con las condiciones de contorno. Por tanto, para conocer el viento en un punto es necesario conocer el viento en los puntos adyacentes. Esto implica, debido a las condiciones de contorno, que los resultados obtenidos en las fronteras del mapa, presenten una diferencia con la realidad, siendo necesario que transcurran un conjunto de celdas para que los valores no sean afectados por las condiciones de contorno y sean próximos a los valores reales [41][42]. Los modelos de conservación de la masa plantean un problema de mínimos cuadrados en el dominio Ω para ajustar las velocidades del viento $\vec{u}(u, v, w)$ a partir de los valores observados o estimados por un modelo meteorológico $\vec{v_0}(u_0, v_0, w_0)$[39].

La función a minimizar se construye a partir de la diferencia entre los valores calculados y los valores observados, como se muestra en la Ecuación 2.1,

$$E(u, v, w) = \int_{\Omega} [\alpha_1^2(u - u_0)^2 + \alpha_1^2(v - v_0)^2 + \alpha_2^2(w - w_0)^2] d\Omega \tag{2.1}$$

donde *u, v, w* son las componentes de la velocidad del viento en las direcciones x (positivo hacia el Este), y (positivo hacia el Norte) y z (positivo hacia arriba), respectivamente;

u_0, v_0, w_0 son los valores iniciales de la velocidad del viento, y α_i es el módulo de precisión de Gauss que se utiliza para controlar la cantidad de cambio inducida por el modelo en las direcciones horizontal y vertical.

La minimización de la Ecuación 2.1 está sujeta a la fuerte restricción de la conservación de la masa, que puede ser expresada como se muestra en la Ecuación 2.2,

$$H(u_x, v_y, w_z) = (\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}) = 0 \tag{2.2}$$

Esta restricción puede ser expresada aplicando la teoría de los multiplicadores de Lagrange. Así, la función puede expresarse como se muestra en la Ecuación 2.3,

$$F(u,v,w,\lambda) = \int_{\Omega} [\alpha_1^2(u-u_0)^2 + \alpha_1^2(v-v_0)^2 + \alpha_2^2(w-w_0)^2 + \lambda(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z})]d\Omega \tag{2.3}$$

donde $\lambda(x, y, z)$ es el multiplicador de Lagrange.

Si analizamos WindNinja, podemos observar que el proceso para obtener el campo de viento se puede resumir en 5 partes, como muestra la Figura 2.1.

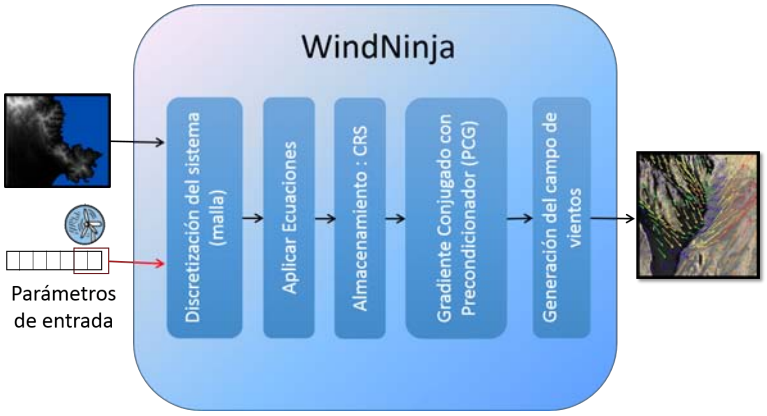


FIGURA 2.1: Simulador WindNinja

Por tanto, para calcular el campo de vientos, WindNinja debe llevar a cabo los siguientes pasos:

1. WindNinja toma como parámetros de entrada el mapa de elevaciones y los valores de las variables meteorológicas relativas al viento, generando una malla tridimensional del terreno.

2. Aplica las ecuaciones de la conservación de la masa (2.3) en cada punto de la malla, para generar un sistema de ecuaciones lineales $Ax = b$.

3. En el sistema lineal $Ax = b$ generado, la matriz A es una matriz simétrica y dispersa, que se almacena en formato CRS (*Compressed Row Storage*). En este sistema, la matriz A tiene una baja densidad y un patrón diagonal [43].

4. Una vez que se ha completado el almacenamiento del sistema lineal, WindNinja aplica el método del gradiente conjugado con precondicionador (*Preconditioned Conjugate Gradient* PCG) [44] para resolver el sistema de ecuaciones. El PCG es un método que usa una matriz M como precondicionador e iterativamente aproxima la solución. WindNinja incorpora como precondicionadores SSOR (*Symmetric Succesive Overrelaxation*), y Jacobian para aquellos casos en los que el uso de SSOR no alcanza la convergencia.

5. La solución del solver PCG es utilizada para construir el campo de vientos, con velocidad, dirección y sentido para cada punto del terreno.

2.2 Implementación de WindNinja

En esta sección se analiza la implementación realizadas por los desarrolladores de WindNinja, para llevar a cabo los pasos descritos en el punto anterior.

2.2.1 Almacenamiento de la matriz *A*

La matriz A generada por el sistema de ecuaciones se almacena en formato *Compressed Row Storage format* (CRS)[45], también llamado Compressed Sparse Row (CSR).

Se utiliza este método de almacenamiento porque tiene en cuenta la dispersión de la matriz. En este formato CRS, la matriz dispersa se almacena en 3 vectores: el primer vector contiene los elementos diferentes de cero de la matriz, el segundo vector indica la posición de la fila de cada elemento diferente de cero y un tercer vector que contiene la posición del vector inicial en el que comienza cada fila. Esta estructura se muestra en la Figura 2.2. Usando este método de almacenamiento, la cantidad de memoria utilizada respecto a la matriz original es mucho menor, pero los acceso para acceder a los distintos elementos de la matriz se vuelven más complejos, y este hecho complica los algoritmos que usan estas matrices de CRS y degrada el rendimiento de la memoria.

2.2.2 Gradiente conjugado con precondicionador (PCG)

Introducir el PCG es una buena solución para los sistema de ecuaciones que están expresados por $Ax = b$, ya que los métodos directos requieren una gran cantidad de

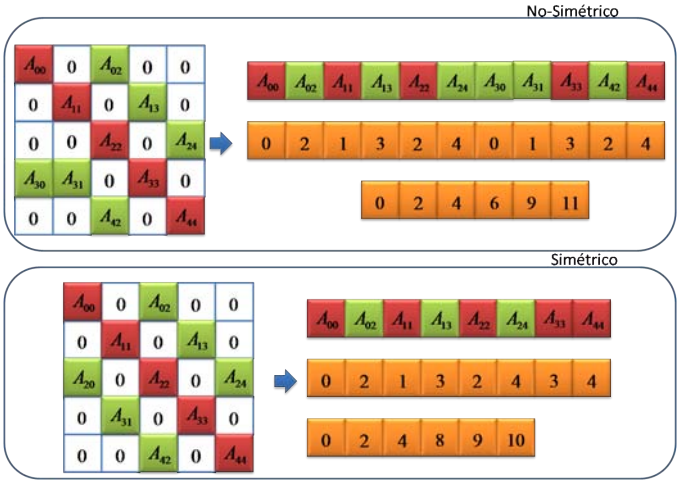


FIGURA 2.2: Formato de almacenamiento Compressed Row Storage (CRS)

memoria y de tiempo. Aplicar el PCG pretende reducir los tiempos de resolución, reduciendo considerablemente el uso de memoria para resolver el sistema. El algoritmo que describe cómo encontrar el vector x al aplicar el PCG se muestra en el Algoritmo 2.1.

Iniciamos con x_0
Calculamos $g_0 = Ax_0 - b$,
que es la diferencia entre el valor inicial y el valor real
Considerando que M es el precondicionar,
evaluamos $q_0 = M^{-1}g_0$
establecemos el valor inicial de p como $p_0 = -q_0$

Para k=1, . . . , n:

$$\alpha_k = \frac{(g_k, q_k)}{(p_k, Ap_k)}$$
$$x_{k+1} = x_k + \alpha_k p_k$$
$$g_{k+1} = g_k - \alpha_k A p_k$$
$$q_0 = M^{-1} g_0$$
$$\beta_k = \frac{(g_{k+1}, q_{k+1})}{(g_k, q_k)}$$
$$p_{k+1} = q_{k+1} + \beta_k p_k$$

ALGORITMO 2.1: Gradiente conjugado con precondicionador (PCG)

El vector x es la solución del sistema que se encuentra en el punto de intersección de todos los hiperplanos creados por la forma cuadrática de cada ecuación del sistema. x se inicializa con x_0. A cada iteración, el valor de x es sustituido por el resultado encontrado en esta iteración. Cada iteración realizada, se aproxima más a la solución real. x es modificada tal como se muestra en el Algoritmo 2.1. Se puede observar que se forma un vector ortogonal, g en la superficie de x, y, a continuación, se obtiene un vector ortogonal a g y q. Los dos vectores ortogonales provienen del vector p, que se aplican en x. Este

proceso se repite de manera iterativa hasta que la diferencia entre ellos es más pequeña que la restricción impuesta por el sistema. Este concepto puede representarse como se muestra en la Figura 2.3.

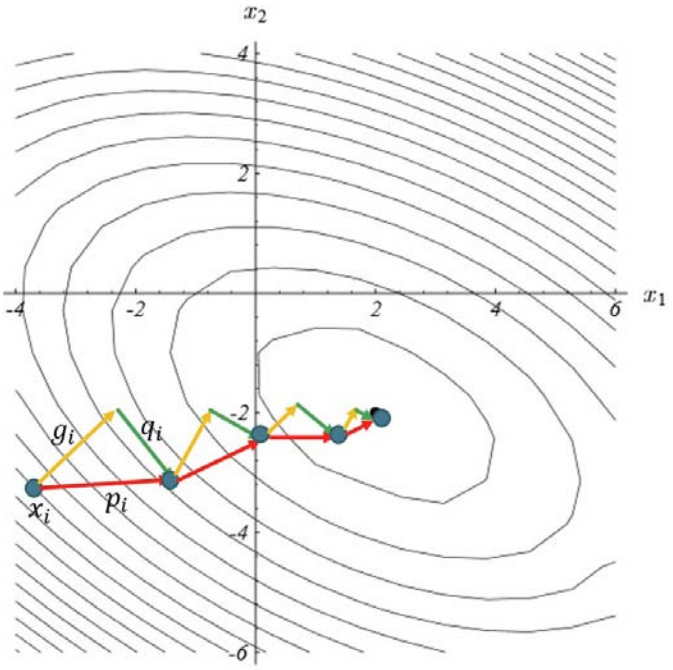


FIGURA 2.3: Representación de la evolución del Gradiente Conjugado con precondicionador

En este algoritmo, las operaciones más costosas son:

1. El producto matriz-vector (en concreto, matriz dispersa-vector), que se realiza en la operación Ap_k, donde la matriz A multiplica al vector p_k.

2. Dependiendo del precondicionador utilizado, la operación $M^{-1}g_{k+1}$, que implicaría realizar el cálculo de una matriz inversa, y un producto, matriz-vector.

En la multiplicación Matriz dispersa-Vector, cada término de la fila se multiplica por el término correspondiente del vector, y los resultados parciales se van acumulando para obtener un elemento en particular del vector resultante, como se muestra en la Ecuación 2.4. El esquema de dicha multiplicación, en formato CRS, se muestra en la Figura 2.4. En este esquema, se pone de manifiesto que los accesos a los distintos elementos del vector b no se realizan de forma lineal o consecutiva, sino que se va accediendo a las posiciones indicadas por el índice de posición de cada término de la fila correspondiente de la matriz A de forma discontinua, con lo cual se generan accesos irregulares a memoria.

$$C_0 = A_{00}B_0 + A_{01}B_1 + \ldots + A_{(n-1)0}B_{(n-1)}$$
$$\ldots$$
$$C_{(n-1)} = A_{0(n-1)}B_0 + A_{1(n-1)}B_1 + \ldots + A_{(n-1)(n-1)}B_{(n-1)}$$

$$C_j = \sum_{i=0}^{n-1} A_{ji}B_i \tag{2.4}$$

Esta irregularidad en los accesos a memoria provoca un gran número de fallos de memoria cache y, por tanto el rendimiento se degrada significativamente. La Tabla 2.1 muestra el número de fallos de cache y la tasa de aciertos (*Hit ratio*) obtenido para diferentes tamaños de mapa. Como se puede observar, la tasa de aciertos a memoria cache se encuentran un poco por encima de 0.70 (70%), lo que implica una tasa de fallos próxima al 30%, lo cual resulta una tasa de fallos extremadamente elevada.

Mapa	Memoria (MB)	Fallos de Cache	Hit Ratio
400x400	353	2971774	0,74
500x500	797	7257159	0,72
600x600	1383	12922181	0,72
700x700	2166	20083338	0,72
800x800	5556	55717760	0,70
900x900	7034	69377314	0,71
1000x1000	8687	87208715	0,70

TABLA 2.1: Rendimiento de la memoria Cache para la implementación en CRS

2.2.3 Precondicionadores

El método del gradiente conjugado está restringido a sistemas lineales cuya matriz de coeficientes es simétrica y definida positiva. Los precondicionadores son imprescindible para la resolución de los sistemas lineales. El uso de un precondicionador permite sustituir el sistema original, por otro con idéntica solución, pero con mejores propiedades de cara a la convergencia de los métodos iterativos. La matriz M es una matriz cuadrada de rango igual al tamaño de la matriz y del mismo orden que A. El sistema precondicionado es [46]:

$$M^{-1}Ax^* = M^{-1}b \tag{2.5}$$

y presenta la misma solución que el sistema $Ax = b$.

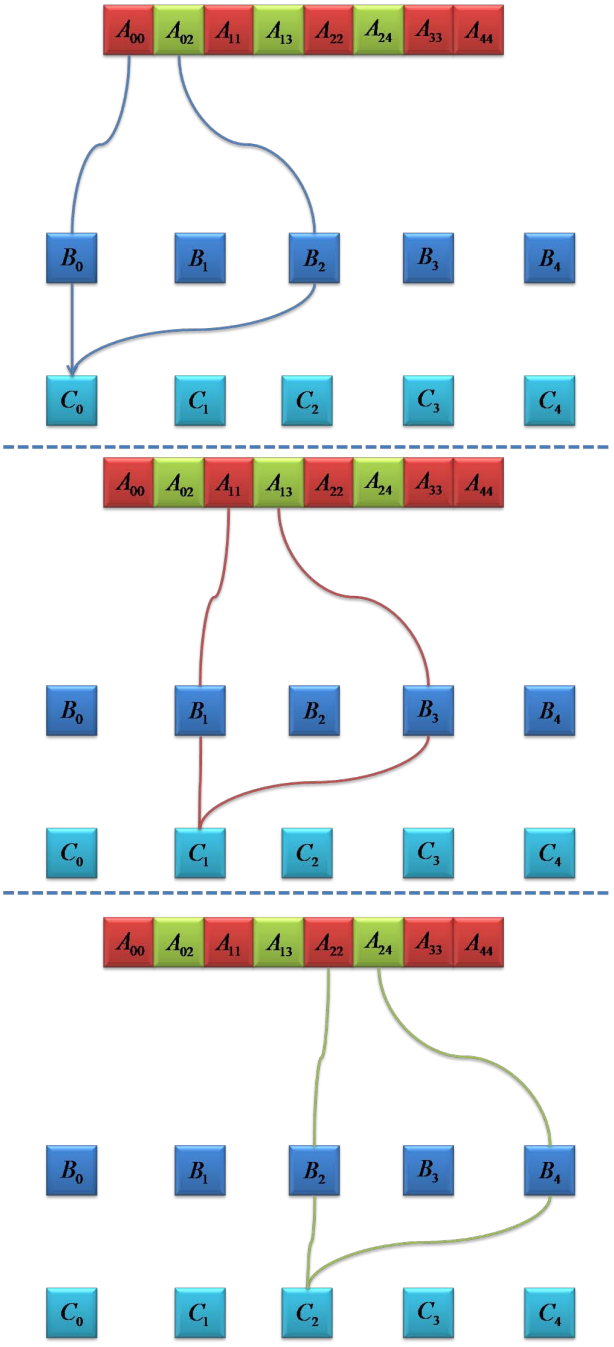


FIGURA 2.4: Multiplicación Matriz vector para matrices dispersas considerando el formato CRS

El objetivo del precondicionador es reducir el número de iteraciones requeridas para que el sistema converja, de modo que el sistema original $Ax = b$ se sustituye por un sistema $\tilde{A}x = \tilde{b}$, satisfaciendo las siguientes propiedades:

- Resolver $\tilde{A}x = \tilde{b}$ no debe incrementar el número de operaciones que se requieren para resolver $Ax = b$.
- $Ax = b$ y $\tilde{A}x = \tilde{b}$ tienen la misma solución por tanto, $\tilde{A}^{-1}\tilde{b} = A^{-1}b$

La matriz $\tilde{A}$ y el vector $\tilde{b}$ se obtienen por multiplicación por una matriz M, que recibe el nombre de precondicionador. El precondicionador se escoge con la condición que sea invertible. Por medio del precondicionador, se transforma el sistema $Ax = b$ en otro equivalente con condiciones más favorables para la convergencia, sin incrementar significativamente los cálculos por iteración.

En los métodos iterativos estacionarios, dado el precondiconador M para el sistema $Ax = b$, obtenemos un sistema equivalente $M^{-1}Ax = M^{-1}b$. Si el precondicionador M se entiende como un aproximación a A, se puede escribir $A = M - N$, y por tanto, $Ax = b$ se puede escribir como $Mx = Nx + b$.

Si la matriz M se escribe como una composición de D, L_A y U_A, donde D es la diagonal principal, L_A es la parte de A estrictamente triangular inferior y U_A es la parte de A estrictamente triangular superior, se pueden aplicar distintos precondicionadores para resolver el sistema con métodos iterativos como el PCG:

1. Si $M = D = diag(A)$, se conoce como método de Jacobian. Se supone que los valores en la diagonal de A son diferentes de cero para que M^{-1} pueda estar definida.

2. Si $M = D + L_A$, se llama método de Gauss-Seidel.

3. Si M es una combinación más compleja, como

$$M = \frac{\omega}{2-\omega}(\omega^{-1}D + L_A)D^{-1}(\omega^{-1}D + U_A), \tag{2.6}$$

   donde ω es un parámetro en el intervalo (0,2), se denomina *Symmetric Successive Over-Relaxation o SSOR*.

Los precondicionadores utilizados por WindNinja son Jacobian y el SSOR. En la Tabla 2.1 se muestra, para diferentes tamaños de mapa, el tiempo de ejecución de WindNinja y el número de iteraciones necesarias para que el sistema converja. El precondicionador Jacobian necesita un mayor número de iteraciones, pero las iteraciones en si son más sencillas, de modo que para mapas pequeños el tiempo de ejecución es similar o ligeramente inferior. En cambio, para mapas grandes el número de iteraciones que requiere el uso del precondicionador Jacobian crece de forma sustancial y el método se ve claramente perjudicado.

Mapa	Memoria	SSOR		Jacobian	
	MB	Iteraciones	Tiempo (s)	Iteraciones	Tiempo (s)
400x400	353	645	87	890	79
500x500	797	531	148	761	142
600x600	1383	634	296	1116	287
700x700	2166	671	442	1259	480
900x900	5556	796	1499	1299	1491
1000x1000	7034	839	1499	1355	1491
1200x1200	8687	875	2198	1465	2525

TABLA 2.1: Número de iteraciones y tiempo de ejecución de WindNinja considerando el precondicionador SSOR y Jacobian

2.3 Limitaciones de WindNinja

Cuando el tamaño del mapa es relativamente pequeño (100x100 celdas), el campo de vientos se genera rápidamente y no presenta limitaciones de memoria. Sin embargo, cuando el tamaño del mapa se incrementa, el tiempo de ejecución y los requisitos de memoria pueden convertirse en una limitación significativa. Debe tenerse en cuenta que el cálculo del campo de vientos de un mapa de 100x100 celdas implica un sistema de 10^5 ecuaciones, mientras que, en cambio, un mapa grande requiere resolver un sistema de 10^8 ecuaciones. Por tanto, WindNinja presenta 2 limitaciones principales [47][48]:

1. Requerimientos de memoria: los requerimientos de memoria de WindNinja para almacenar y resolver el sistema dependen del tamaño del mapa [47]. La cantidad de memoria requerida para resolver un mapa de NxM celdas puede expresarse como se muestra en la ecuación 2.7, donde N es el número de filas y M es el número de columnas del mapa. Esta cantidad de memoria puede contrastarse con los datos reales obtenidos a partir de la resolución de diferentes mapas y graficarse tal como se muestra en la Figura 2.5. En esta figura, puede observarse que, por ejemplo, para resolver un mapa de 1500x1500 celdas se necesitan 25GB de memoria.

$$M(Bytes) = 20480 + 15360 * N + 15360 * N + 11520 * N * M \tag{2.7}$$

2. Tiempo de ejecución: El tiempo de ejecución para resolver el sistema de ecuaciones con el solver PCG representa el 80% del tiempo total de ejecución del WindNinja. Este tiempo de ejecución depende del tamaño del mapa y, evidentemente, de la capacidad de cálculo de la arquitectura computacional utilizada. Se ha estudiado esta dependencia y se ha determinado que es una dependencia completamente lineal con el número de celdas del mapa [48]. Por tanto, el tiempo de ejecución

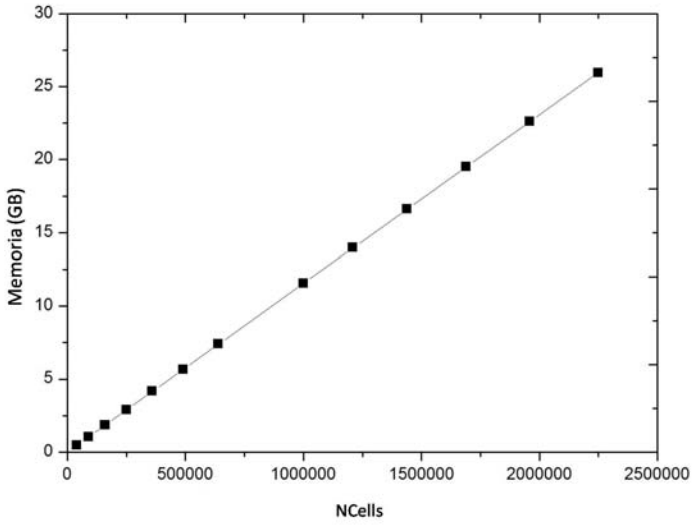


FIGURA 2.5: Requerimientos de memoria de WindNinja

depende linealmente del número de celdas del mapa ($NCells$=NxM), y la pendiente de la recta depende de las características de la arquitectura sobre las que se ejecuta el simulador. Esta dependencia se muestra en la Ecuación 2.8, donde a y b son parámetros que dependen de la arquitectura, y deben determinarse experimentalmente para cada una de ellas. La Figura 2.6 muestra esta dependencia para el caso concreto de una arquitectura basada en procesadores Dual-Core Intel(R) Xeon(R) 5150 a 2,66GHz. Para esta arquitectura, puede observarse que un mapa de 1500x1500 celdas requiere un tiempo de más de 3000 segundos para ser resuelto, lo cual hace el sistema completamente inaplicable en entornos operacionales reales.

$$t = aNCells + b \tag{2.8}$$

2.4 Escalabilidad de WindNinja

Para paliar, en cierta medida, estas limitaciones, los desarrolladores de WindNinja integraron una paralelización básica en OpenMP que permitiera explotar los núcleos (cores) disponibles en los procesadores actuales, y reducir así el tiempo de ejecución.

La Tabla 2.2 muestra el tiempo de ejecución y el SpeedUp para mapas de diferente tamaño y utilizando distinto número de cores, cuando se han ejecutado en un cluster

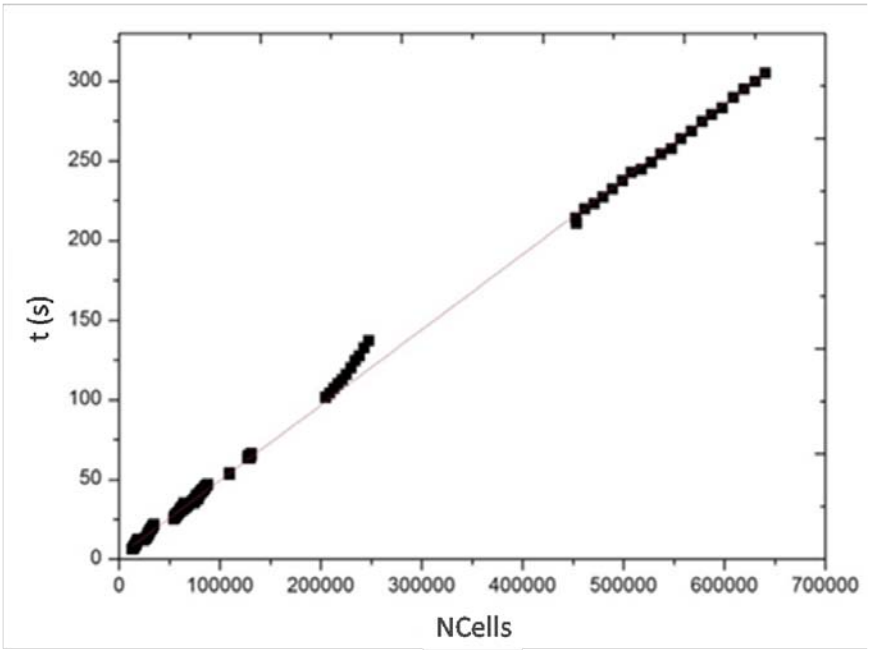


FIGURA 2.6: Tiempo de ejecución de WindNinja para mapas con distinto número de celdas sobre una arquitectura basada en procesadores Dual-Core Intel(R) Xeon(R) 5150 a 2,66GHz

basado en procesadores de 8 cores Intel(R) Xeon(R) E5-2650. Estos resultados se muestran en forma gráfica en las Figuras 2.7 y 2.8.

Mapa	Tiempo (s) N Cores					Speedup N Cores				
	1	2	4	8	16	1	2	4	8	16
400x400	237	194	182	207	182	1	1,22	1,30	1,14	1,30
600x600	567	410	399	442	464	1	1,38	1,42	1,28	1,22
800x800	804	681	558	659	659	1	1,18	1,44	1,22	1,22
1000x1000	1831	1743	1592	1500	1505	1	1,05	1,15	1,21	1,22
1200x1200	2939	2525	2300	2.111	2.113	1	1,02	1,12	1,23	1,21
1500x1500	3831	3793	3390	3140	3140	1	1,02	1,13	1,21	1,21

TABLA 2.2: Tiempo de ejecución y Speedup de WindNinja en un cluster basado en procesadores de 8 cores Intel(R) Xeon(R) E5-2650

También se realizó una comparación entre el desempeño de los dos precondicionadores implementados. Las Tablas 2.3 y 2.4 muestran el tiempo de ejecución de WindNinja en un clúster basado en AMD Opteron 6376 procesadores con 16 cores, considerando precondicionadores SSOR y Jacobian, respectivamente.

Se puede observar que la reducción del tiempo de ejecución no es muy significativa. En el caso de la precondicionador SSOR no hay una reducción significativa, y en el caso de Jacobian, la reducción de tiempo de ejecución es más apreciable, pero más allá de 8 cores, no hay reducción de tiempo. Considerando 8 cores el SpeedUp alcanzado es de 1,5, para el mejor de los casos. También se puede observar que la reducción de tiempo de

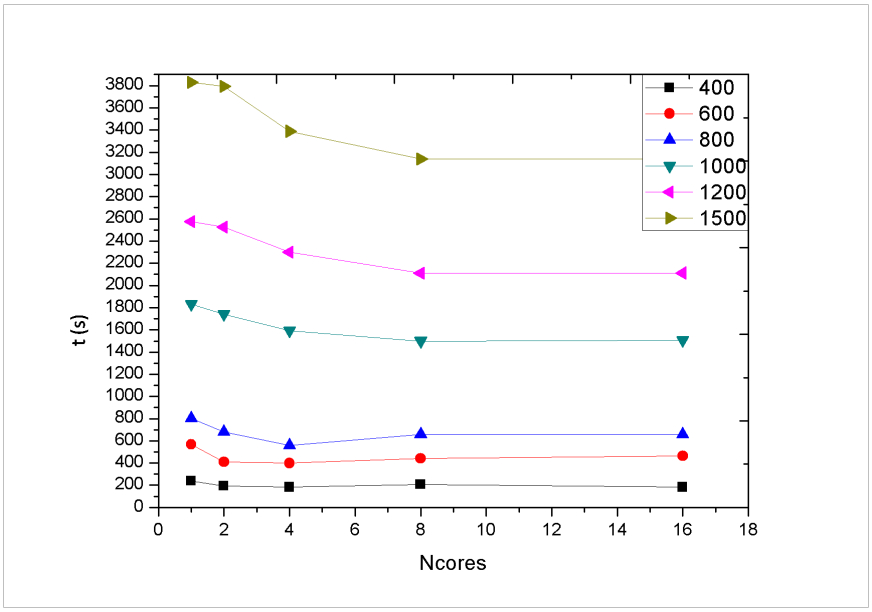


FIGURA 2.7: Tiempo de ejecución de WindNinja para mapas de distinto tamaño y distinto número de cores sobre un cluster basado en procesadores de 8 cores Intel(R) Xeon(R) E5-2650

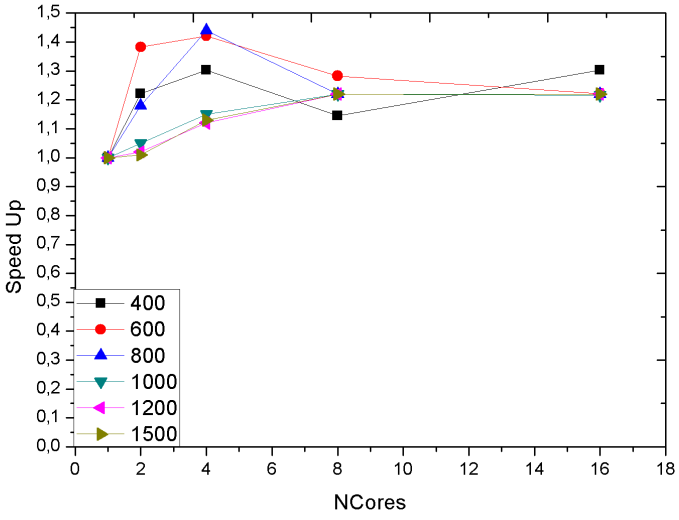


FIGURA 2.8: Speedup de WindNinja para mapas de distinto tamaño y distinto número de cores sobre un cluster basado en procesadores de 8 cores Intel(R) Xeon(R) E5-2650

ejecución obtenido mediante el uso de 4 cores es casi el mismo que el tiempo de ejecución obtenido con 8 cores. Se puede concluir que la implementación actual de WindNinja con PCG requiere mucho tiempo en solucionarlo y no escala correctamente.

Mapa	Tiempo de ejecución (s) Cores			
	1	2	4	8
400x400	87	87	86	87
500x500	142	141	139	142
600x600	303	300	298	301
800x800	502	498	500	499
1000x1000	1673	1669	1670	1671
1200x1200	2105	2099	2100	2099
1500x1500	2923	2920	2918	2916

TABLA 2.3: Tiempo de ejecución WindNinja usando el precondcionador SSOR sobre un cluster basado en un procesador de 16 cores AMD Opteron 6376

Mapa	Tiempo de ejecución (s) Cores			
	1	2	4	8
400x400	79	63	55	56
500x500	142	116	101	125
600x600	356	320	308	308
800x800	480	390	340	340
1000x1000	1968	1881	1460	1371
1200x1200	2725	2627	2121	2054
1500x1500	3954	3730	2881	2824

TABLA 2.4: Tiempo de ejecución WindNinja usando el precondcionador Jacobian sobre un cluster basado en un procesador de 16 cores AMD Opteron 6376

2.5 Librerías matemáticas

Para intentar mejor los resultados se planteo la integración de diferentes librerías matemáticas como Intel MKL[49], ViennaCL [50] y cuSparse [51], que incluyen el gradiente conjugado con precondicionador, esperando que las implementaciones de estas librerías mejorasen sustancialmente los tiempos de ejecución y la escalabilidad de la implementación original de WindNinja [43]. Los resultados, para el caso de un mapa de 1000x1000 celdas se muestran en la Figura 2.9. En estos resultados puede observarse que el tiempo de ejecución no se reduce considerablemente y no se alcanza una escalabilidad significativamente mejor. Para mapas de otros tamaños, los resultados son muy similares y siguen la misma tendencia.

Analizando las causas de este mal rendimiento y baja escalabilidad, incluso en implementaciones contrastadas del gradiente conjugado con precondicionador, se determinó que el principal problema radica en las características de las matrices generadas por WindNinja, y que utiliza durante el proceso de resolución.

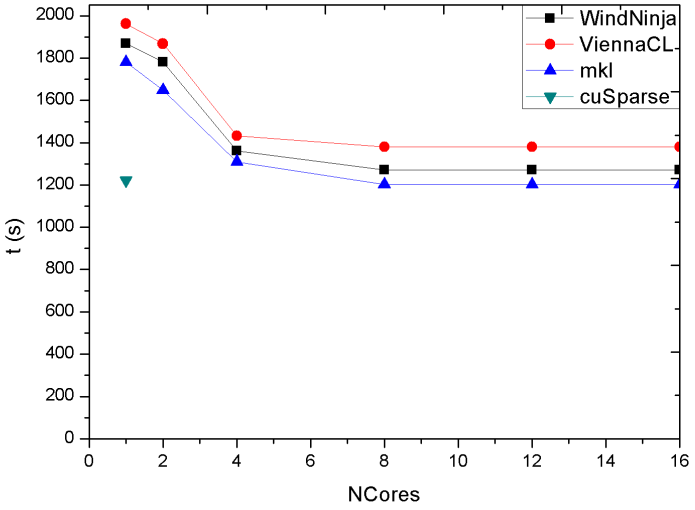


FIGURA 2.9: Tiempo de ejecución de WindNinja considerando diferentes librerías matemáticas sobre un cluster basado en procesadores de 8 cores Intel(R) Xeon(R) E5-2650 para un mapa de 1000x1000 celdas

Las matrices generadas por WindNinja son matrices simétricas en las que cada elemento de la matriz representa la interacción entre dos nodos de la malla. La diagonal principal describe las interacciones en cada uno de los nodos de la malla, mientras que el resto de términos se distribuyen en un total de 26 subdiagonales (13 si consideramos que son matrices simétricas), ya que únicamente se consideran las interacciones con los nodos adyacentes de la malla. Realmente, esta estructura de la matrices es muy común en diversos problemas que toman en cuenta características geofísicas [52]. Lo que cambia de un caso a otro, es la posición relativa de las subdiagonales respecto a la diagonal principal.

Para terrenos muy pequeños (100x100 celdas), las matrices resultantes son matrices dispersas, pero con una densidad apreciable. Sin embargo, a medida que se van considerando mapas de mayor tamaño, la densidad de las matrices va disminuyendo, a pesar de que se siguen manteniendo la diagonal principal y las 26 subdiagonales. En la Figura 2.10 se muestra la disminución de la densidad de la matriz a medida que se consideran mapas de mayor tamaño. De hecho, para mapas de 600x600 celdas, la densidad de la matriz es del orden de $0,002$, lo que es una densidad extremadamente baja que se encuentra muy alejada de las densidades de las matrices dispersas consideradas en muchos otros problemas. Una colección de matrices dispersas representativas de muchos problemas esta disponible en [53].

Esto es debido a que la mayoría de puntos que genera WindNinja son cero y sólo los elementos diferentes de cero son aquellos que se multiplican por el vector. Esto implica

que sea una matriz extremamente dispersa (Figura 2.10). En esta operación, el formato de almacenaje juega un papel importante en el rendimiento del sistema y la baja densidad afecta considerablemente a la escalabilidad del sistema.

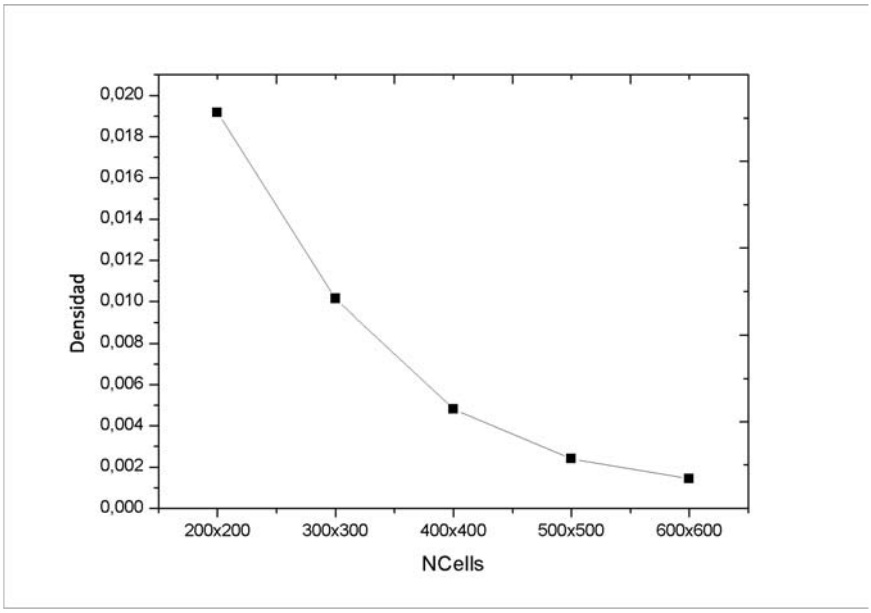


FIGURA 2.10: Densidad de la matriz dependiendo del tamaño del mapa

2.6 Conclusiones del capítulo

Todo lo expuesto pone de manifiesto que la generación del campo de vientos, y específicamente en el caso de WindNinja, requiere un estudio en profundidad y la aplicación de técnicas de computación de altas prestaciones para llegar a ser una aplicación útil en un entorno operacional, en el que se traten situaciones de emergencia reales. Por ello, esta tesis se ha centrado en la paralelización de WindNinja aplicando diferentes técnicas, e incluso combinándolas entre sí para alcanzar un mejor rendimiento. En los próximos capítulos se presentan las técnicas aplicadas: Particionamiento del mapa, descomposición del dominio y paralelización a bajo nivel.

Capítulo 3

Explotando el paralelismo de datos: Particionamiento del mapa

3.1 Introducción

Tal como se ha visto en el capítulo anterior, WindNinja es un simulador de campo de vientos que permite obtener los valores de la velocidad y dirección del viento a una resolución elevada, pero que consume importantes recursos de memoria y, además, requiere un tiempo de ejecución muy elevado cuando el mapa tiene un tamaño considerable. Se ha comprobado que el tiempo de ejecución crece de forma lineal con el número de celdas del mapa, de modo que, si el cálculo del campo de vientos de un terreno puede dividirse en el cálculo del campo de vientos para fragmentos de ese terreno, y cada fragmento procesarse en paralelo, se reducirá el tiempo de forma proporcional al número de fragmentos en los que se haya dividido el terreno. Con esta idea, se ha diseñado una metodología de particionado del mapa, del mapa objeto de estudio. En esta aproximación, se considera que la resolución de un problema global puede alcanzarse mediante la resolución del mismo problema considerando las partes que constituyen los datos globales.

Sin embargo, hay algunos factores que hemos de considerar, y que, en cierta medida, van a limitar las posibilidades de la metodología. Estos factores son los siguientes:

1. Tal como se ha comentado en el capítulo anterior, las ecuaciones de variación de la velocidad del viento y de conservación de la masa, que utiliza WindNinja, introducen un cierto grado de incertidumbre en los valores del campo de viento en las proximidades de los bordes del mapa. Esto es debido a que, en el fondo, no se conoce lo que hay más allá de los bordes del mapa, y por tanto, las celdas del

borde del mapa, o muy próximas a dicho borde, van a disponer de una información limitada que impide que se puedan obtener los valores de la velocidad y dirección del viento en esas celdas.

Si se realiza un particionamiento del mapa, dividiendo el mapa en un conjunto de fragmentos, tendremos que hay un borde o frontera entre los distintos fragmentos, de modo que los valores de la velocidad y dirección del viento en las celdas próximas a ese borde, calculados como se tratase de un mapa individual, presentan un cierto grado de imprecisión, con lo que el campo de vientos va a adolecer de un elevado grado de imprecisión o error.

2. Por otro lado, debe considerarse, que a medida que se va particioinando el mapa del terreno, se va perdiendo, en cierto sentido, la información global del problema, pasando a tener una visión local del mismo, con lo cual, la solución global, va a resentirse de esa falta de visión global de la información.

3. Cada partición que se realiza introduce una nueva frontera, con su grado de imprecisión asociado, y restringe un poco más la información de la que se dispone, siendo más local y focalizada.

Un método para reducir el efecto de los bordes o fronteras consiste en añadir un conjunto de celdas alrededor de cada partición de modo que las particiones tengan un cierto grado de solapamiento. Así, cuando se calcula el mapa de modo que la parte calculada correspondiente a cada partición se le añade un conjunto de celdas de las particiones vecinas (Figura 3.1). Para obtener el campo de vientos total se unen los campos de viento obtenidos en cada partición, descartando los vientos calculados en la zona solapada correspondiente a la zona vecinas del mapa. Aunque hay que tener en cuenta que la introducción de un cierto grado de solapamiento entre las particiones del mapa va a provocar un aumento en el tiempo de procesamiento de cada partición, con la consiguiente perdida de rendimiento, el efecto en la precisión del cálculo es mucho más significativo que la pérdida en rendimiento.

Este esquema de particionamiento del mapa, para el cálculo por separado del campo de vientos en cada una de las partes, encaja con un esquema Master/Worker. Por ello, se ha implementado en una aplicación MPI, en la que el proceso Master particiona el mapa y reparte las distintas partes del mismo a los procesos worker. Estos procesos worker reciben la parte correspondiente del mapa, calculan el campo de vientos correspondiente a esa parte y devuelven al proceso Master el campo de viento generado. El proceso Master va recibiendo y agregando los campos de vientos correspondientes a cada una de las partes calculados por los procesos worker.

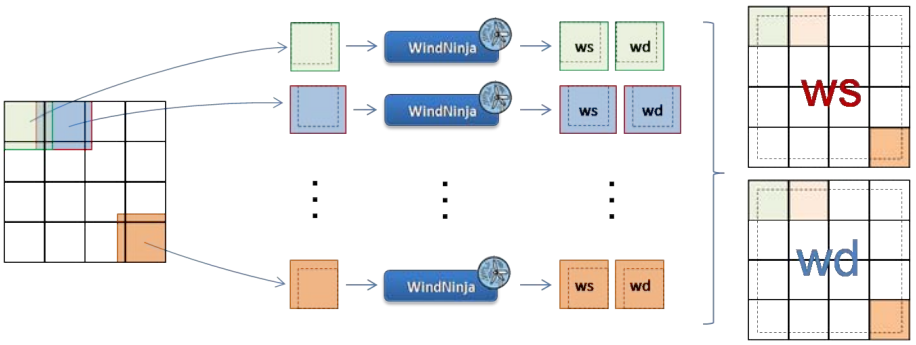


FIGURA 3.1: Partición de mapas con solapamiento

Para determinar la diferencia entre el campo de vientos obtenido con el mapa original, por tanto sin particionar, y el campo de vientos obtenido a partir del particionamiento del mapa, se utiliza el RMSE (Root Mean Square Error), ya que permite cuantificar la diferencia entre los dos campos de viento, a partir de la expresión 3.1:

$$RMSE_{AxB}(ws) = \sqrt{\frac{\sum_{i=0}^{N} (NP(ws)_i - SC_{AxB}(ws)_i)^2}{N}}, \tag{3.1}$$

donde AxB hace referencia al tipo de partición que presenta el mapa, N representa el numero de celdas totales que tiene el mapa e i hace referencia a cada una de las celdas del mapa. Los términos del sumatorio $NP(ws)_i$ son los valores obtenidos de la velocidad del viento en el mapa sin particionar para la celda i y $SC_{AxB}(ws)_i$ son los valores obtenidos de la velocidad del viento para cada celda i del mapa particionado AxB. El objetivo es realizar una comparación entre los valores de la velocidad del viento obtenido con el mapa global o con el mapa particionado. De forma análoga, se puede evaluar el error para la dirección, simplemente cambiando los términos ws por wd, es decir, la velocidad del viento e cada punto del mapa, por la dirección del viento [54].

El objetivo principal es garantizar el tiempo de ejecución de WindNinja, para poder ejecutarlo en tiempo real. Pero, para su aplicación en problemas como la predicción de la propagación de los incendios forestales, tan importante como el tiempo de predicción es la exactitud de la predicción de la propagación de incendios forestales. Por tanto, también es necesario calcular la diferencia entre la predicción de la propagación de los incendios obtenidos utilizando WindNinja con particionamiento del mapa obtenido con respecto a la predicción cuando WindNinja se ejecuta con el mapa sin particionar. La diferencia entre las áreas quemadas predichas utilizando diferentes configuraciones WindNinja se estima mediante la aplicación de la Ecuación (3.2). Esta ecuación calcula la diferencia simétrica normalizada, es decir, la diferencia entre las áreas predichas por dos configuraciones WindNinja diferentes dividida por el número de celdas quemadas

predicho utilizando el campo de vientos generado por WindNinja considerando un mapa global. En este caso, el área predicha usando un campo de viento sin particionar (GMWF) corresponde con la propagación de referencia. Formalmente, esta ecuación corresponde a la diferencia simétrica entre el área del campo de mapa sin particionamiento del mapa del campo de vientos (GMWF) y el mapa particionado por en el área GMWF. ∪(GMCell, Pcell) es la unión del número de celdas quemadas en el GMWF y las celdas quemadas en el mapa viento con particiones, ∩(GMCell, Pcell) es la intersección entre el número de celdas quemadas en el propagación y en el mapa del campo de viento con particiones GMWF, y GMCell es el número de celdas quemadas utilizando el campo de mapa viento sin particionar.

$$\mathrm{D}=\frac{\cup(\mathrm{GMCell},\mathrm{PCell})-\cap(\mathrm{GMCell},\mathrm{PCell})}{\mathrm{GMCell}} \tag{3.2}$$

Debe tenerse en cuenta que el objetivo final es proporcionar predicciones lo más exactas posibles de los incendios forestales y la diferencia en la predicción de incendios forestales debe limitarse en el valor $0,1$. De esta manera no se observan variaciones perceptibles en la predicción de la propagación de incendios entre un caso y otro [54].

Así, se debe buscar un compromiso entre el grado de paralelismo y el grado de precisión del cálculo, con lo cual es necesario realiza un estudio completo que permita determinar la mejor forma de particionar el mapa para aprovechar el paralelismo al máximo, y reducir el tiempo de ejecución, sin introducir un error significativo en el cálculo del campo de vientos. En este capítulo se va a desarrollar este estudio, y de él se derivará una metodología [47][48] para determinar el mejor modo de particionar el mapa para ajustarse al compromiso establecido.

3.2 Tamaño de la partición

Como se indica en el capítulo 2, el tiempo de ejecución y los requisitos de memoria de WindNinja presentan una dependencia lineal con respecto al número de celdas del mapa. A partir de la ecuación $t = aNCells + b$, es posible determinar el máximo número de celdas, para una condición de tiempo dada (Ecuación 3.3).

$$NCells = \left(\frac{t}{a} - \frac{b}{a}\right) \tag{3.3}$$

El máximo número de celdas, $NCells$, que puede tener una partición para ser resuelta en un tiempo t, puede ser expresado como una función ($\lambda(t)$), que depende del tiempo disponible y de las características de la arquitectura donde se va a ejecutar (a y b):

$$\lambda(t) = \left(\frac{t}{a} - \frac{b}{a}\right) \tag{3.4}$$

A partir de las consideraciones operacionales de los estamentos de coordinación de la extinción de los incendios forestales, se ha determinado que el tiempo máximo para el cálculo del campo de vientos es de 100 segundos. Así, aplicando la Ecuación (3.4), para un cluster basado en procesadores Dual-Core Intel(R) Xeon(R) 5150 a 2,66GHz, se deduce que el tamaño máximo del mapa debe estar alrededor 160000 celdas. Con unos requisitos de memoria de aproximadamente de 2GB (Figura 2.5), que es completamente factible en los nodos actuales.

3.3 Forma de las partes

Para determinar cual es la forma más adecuada de particionamiento se han realizado un conjunto de experimentos para un mapa de 1500x1500 celdas, donde cada parte contenga aproximadamente 160000 celdas, incluyendo 50 celdas de solapamiento por cada lado. La Tabla 3.1 muestra el número de partes en las que se ha dividido el mapa horizontalmente ($Horizontal$), el número de partes en las que se ha dividido el mapa verticalmente ($Vertical$), el número de filas ($Filas$) y columnas ($Columnas$) de cada partición excluyendo el solapamiento y el error RMSE, expresado en m/s.

Horizontal	Vertical	Filas	Columnas	RSME (m/s)
2	15	750	100	3,510
3	9	500	166	0,579
5	5	300	300	0,559
9	3	166	500	0,564
15	2	100	750	3,516

Tabla 3.1: Diferentes métodos de particionamiento para un mapa de 1500x1500 celdas

En los resultados se ha podido observar que la partición cuadrada es la que proporciona mejores resultados en cuanto al valor del RMSE. Este resultado es consistente con el hecho que, ya que las fronteras son las partes que van a representar una mayor fuente de error, las particiones del mapa deben ofrecer el perímetro mínimo para el área máxima, ya que de esta manera se minimiza el error producido por las condiciones de frontera.

Si el área (A_p) de una partición viene dado por la expresión:

$$A_p = N_R * N_C, \tag{3.5}$$

donde N_R es el número de filas y N_C es el número de columnas de cada parte. La expresión del perímetro (P_p) correspondiente a ese área es la que se muestra en la Ecuación 3.6.

$$P_p = 2 * N_R + 2 * N_C \tag{3.6}$$

N_C puede ser aislado de la expresión 3.5 y substituido en la expresión 3.6, de modo que se alcanza la Ecuación 3.7.

$$P_p = 2 * N_R + 2 * \frac{A}{N_R} \tag{3.7}$$

Minimizando la expresión del perímetro 3.6, con respecto a NR, se obtiene:

$$A_p = (N_R)^2, \tag{3.8}$$

y, por tanto,

$$N_C = N_R \tag{3.9}$$

Por lo tanto, la figura geométrica que cumple esta propiedad es el cuadrado, debido a que presenta el menor número de celdas expuestas en la partición y el perímetro más pequeño. Si el tamaño máximo de la partición es 160000 celdas esto implica que cada lado debe tener un tamaño de 400x400, incluyendo el solapamiento. Es interesante que el mapa completo sea tomado como un mapa cuadrado. Este hecho, por lo general, no es una limitación, ya que, cuando el fuego se esta propagando en un lugar particular, es posible definir el centroide del incendio y recortar un mapa cuadrado alrededor de este centro de gravedad de 40 o 50 kilómetros por lado, a partir de los mapas de elevaciones digitales disponibles en los distintos repositorios públicos.

3.4 Cantidad de solapamiento

Dada una parte del mapa con MxN celdas, la cantidad de celdas de solapamiento por lado podría variar. Esto implica que, para un tamaño fijo de parte con solapamiento

incluido, si el solapamiento aumenta, el campo de viento efectivo calculado en la partición disminuye, y, por tanto, hay más particiones para un mismo mapa. La Figura 3.2 muestra una partición de 400x400 celdas con diferente cantidad de solapamiento (área oscura) y distinta zona efectiva de campo de viento calculada (área blanca). Considerando que el mapa total es de 1500x1500 celdas, la figura también indica el número de partes en las que debería dividirse el mapa para cada caso.

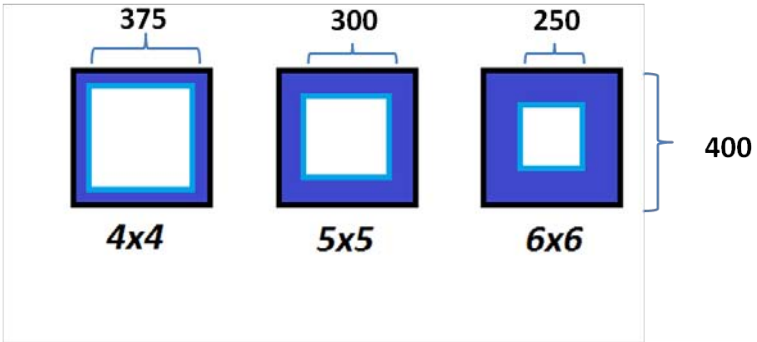


FIGURA 3.2: Diferente grado de solapamiento

La Figura 3.3 muestra el RMSE obtenido en función de la cantidad de solapamiento (Ov) introducido y se pueden observar claramente dos zonas diferenciadas [47]: La primera parte de la gráfica corresponde al error ocasionado por las condiciones de frontera y la segunda parte corresponde a la diferencia que existe, debido a que la partición es una singularidad del mapa completo, haciendo que los resultados encontrados sean parecidos pero no idénticos. Como se puede observar, una vez superadas las 25 celdas de solapamiento, el error cometido se estabiliza, de modo que el error debido a las condiciones de frontera, prácticamente se hace inapreciable, y únicamente queda el error residual.

3.5 Número de partes

El número de partes en las que se divide el mapa influye directamente en el error del campo de viento. Las particiones no deben superar las 400x400 celdas para respetar el límite de tiempo establecido, y con un solapamiento de 50 celdas por lado se garantiza que se han eliminado los efectos de frontera. Así, la parte útil de cada partición, como máximo, será de 300x300 celdas. Por lo tanto, dado un mapa, el número de particiones horizontales será como mínimo el número filas del mapa dividido por 300, y el número de particiones verticales será, a su vez, el número de columnas del mapa dividido por 300.

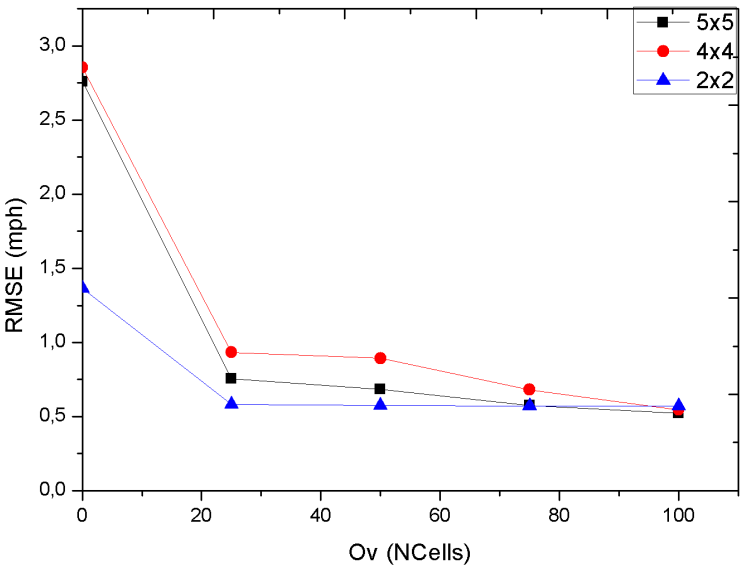


FIGURA 3.3: RMSE en función del grado de solapamiento

Para determinar el tamaño de partición más adecuada para un mapa dado, se han realizado experimentos con diferentes tipos de particiones. Las particiones usadas se muestran en la Tabla 3.2. Esta tabla muestra las particiones del mapa en AxB particiones. Para cada tipo de partición, se han probado un solapamiento de 25 y 50 celdas. El $RMSE$ es el error de media cuadrática de la velocidad del viento medida en m/s, T_{TI} es el tiempo de ejecución del mapa sin particiones en un cluster basado en procesadores Dual-Core Intel(R) Xeon(R) 5150 a 2,66GHz y T_I es el tiempo de ejecución para el mapa particionado sobre el mismo cluster.

Los resultados (Tabla 3.2) muestran que el tiempo de ejecución se reduce significativamente cuando se aplica el método del particionamiento del mapa. La reducción del tiempo varía entre el 50 % y el 95 % dependiendo del tipo de partición realizada. Por ejemplo, un mapa de 750x750 celdas, que resuelto como un mapa único conlleva un tiempo de ejecución de 271 segundos, se resuelve en tan solo 13 segundos cuando se aplica un particionamiento en 7x7, es decir, 49 partes. Hay que comentar que el mapa de 1500x1500 celdas no puede resolverse en esta arquitectura, por lo cual no se incluyen los tiempos de la resolcuión de WindNinja con un mapa completo de ese tamaño. Los resultados correspondientes a RSME mostrados corresponden a otra arquitectura con mayor cantidad de memoria por nodo, pero con unos resultados de mejora en el tiempo de ejecución muy similares.

La Figura 3.4 muestra el SpeedUp alcanzado al realizar distintas particiones y el RMSE correspondiente (en función del número de partes generado) para un mapa de 1500x1500 celdas con un solapamiento de 50 celdas por lado. Puede observarse que a medida que va

Mapa	AxB	Ov	RMSE (mph)	t_{TI} (s)	t_I (s)
200x200	2x2	50	0,097	21	12
300x300	2x2	25	0,324	60	21
	3x3	50	0,358	60	15
500x500	2x2	50	0,164	98	38
	3x3	25	0,241	98	16
	5x5	50	0,262	98	15
750x750	2x2	25	0,160	271	38
	4x4	25	0,230	271	16
	7x7	50	0,310	271	13
1500x1500	2x2	50	0,330	–	405
	4x4	25	0,563	–	78
	5x5	50	0,559	–	60
	6x6	25	0,642	–	38
	7x7	50	0,677	–	36
	10x10	25	0,795	–	16
	15x15	50	0,772	–	13

TABLA 3.2: Diferencia del viento (RMSE) y el tiempo de ejecución considerando diferentes particiones del mapa

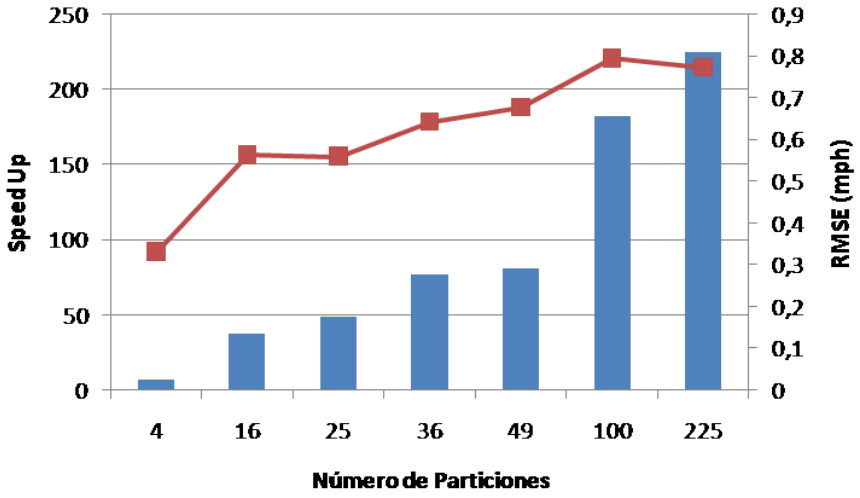


FIGURA 3.4: SpeedUp y RMSE dependiendo del número de partes

aumentando el número de particiones, el RMSE se va incrementando, aunque no crece de forma muy repentina.

Por lo tanto, el método del particionamiento del mapa permite reducir los requisitos de memoria y el tiempo de ejecución de WindNinja explotando los arquitectura de los diferentes nodos del cluster.

3.6 SpeedUp

Los resultados de tiempo de ejecución y de Speedup se muestran en la Tabla 3.3. La tabla muestra el tiempo de ejecución y Speedup obtenido para diferentes particionamientos (2x2, 3x3 y 4x4) y diferentes solapamientos (0, 25, 50, 75 y 100 celdas por lados). Los resultados muestran una muy buena escalabilidad, aunque el aumento de solapamiento reduce significativamente el SpeedUp. Así, un mapa sin particionar de 1500x1500 es ejecutado en 3831 segundos (aproximadamente 3 horas) en un cluster basado en procesadores AMD Opteron 6376 con 16 cores. Aplicando un particionamiento de 2x2 y 50 celdas de solapamiento el sistema se resuelve en un tiempo de 1171 segundos, lo que representa un SpeedUp de $3,27$. Si ese mismo mapa se particiona en 4x4 partes con 50 celdas de solapamiento por lado, es ejecutado en 412 segundos, que representa un SpeedUp de $9,28$. Los resultados también se muestran en forma gráfica en las Figuras 3.5 y 3.6.

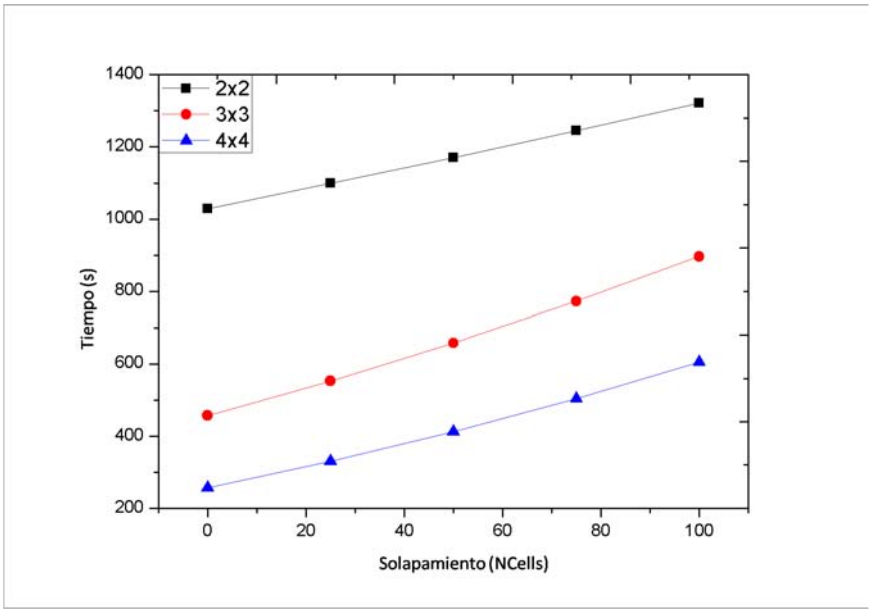


FIGURA 3.5: Tiempo de ejecución aplicando particionamiento para un mapa de 1500x1500 celdas

3.7 Resolución de mapa, solapamiento y número de partes

En las secciones previas, se ha determinado que el número máximo por partición debería ser de 400x400 celdas y cada parte debería ser cuadrada, con un solapamiento mínimo de 25 celdas por lado. Sin embargo, si nos fijamos en los resultados de la Tabla 3.3 podemos observar que, en la mayoría de los casos, los tiempos de ejecución superan con creces el tiempo límite que se ha establecido en 100 segundos, ya que las particiones superan el

Mapa	Solapamiento	WN	Tiempo (s) Particionamiento 2x2	3x3	4x4	SpeedUp Particionamiento 2x2	3x3	4x4
400x400	0	237	73	32	18	3,24	7,28	12,95
	25		92	61	41	2,56	3,85	5,75
	50		104	99	73	2,07	2,38	3,24
	75		138	147	114	1,71	1,61	2,07
	100		164	203	164	1,44	1,17	1,44
600x600	0	567	164	73	41	3,44	7,74	13,77
	25		193	114	73	2,93	4,96	7,74
	50		224	164	112	2,53	3,44	4,96
	75		257	224	164	2,20	2,53	3,44
	100		292	292	224	1,94	1,94	2,53
800x800	0	804	292	130	73	2,75	6,18	10,98
	25		330	183	102	2,43	4,38	7,03
	50		370	246	164	2,17	3,27	4,88
	75		412	317	224	1,95	2,53	3,59
	100		457	398	292	1,76	2,02	2,75
1000x1000	0	1836	457	203	118	4,01	9,03	16,05
	25		504	269	164	3,64	6,83	11,15
	50		553	343	224	3,32	5,34	8,19
	75		605	427	292	3,03	4,29	6,27
	100		658	520	370	2,79	3,53	4,95
1200x1200	0	2939	658	292	164	4,46	10,04	17,84
	25		714	370	224	4,11	7,93	13,11
	50		773	457	292	3,80	6,42	10,04
	75		833	553	370	3,52	5,31	7,93
	100		896	658	457	3,28	4,46	6,42
1500x1500	0	3831	1029	457	257	3,72	8,37	14,88
	25		1099	553	330	3,48	6,92	11,59
	50		1171	658	412	3,27	5,81	9,28
	75		1245	773	504	3,08	4,95	7,59
	100		1322	896	605	2,90	4,27	6,33

TABLA 3.3: Tiempo de ejecución y Speedup aplicando particionamiento para un mapa de 1500x1500 celdas

tamaño de 400x400 celdas. Por otro lado, si observamos la Tabla 3.2 podemos apreciar que el error en el campo de vientos va aumentando a medida que vamos incrementando el número de particiones. Así pues, es necesario realizar un estudio más completo y detallado que permita cumplir con todas las restricciones.

Hay que tener en cuenta que la obtención de un campo de viento del mapa de particionado genera un campo de viento un poco diferente del campo de viento obtenido resolviendo el mapa sin particionar y, en consecuencia, la predicción de la propagación del fuego en los dos casos también será un poco diferente. La diferencia depende del número de partes, del solapamiento entre las partes y de la resolución de mapa utilizado. Estos

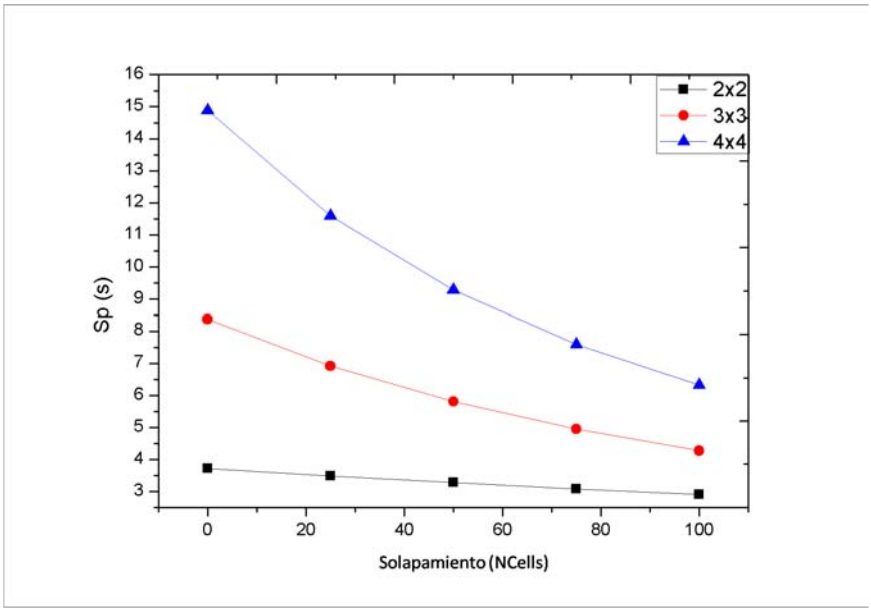


FIGURA 3.6: Speedup obtenido aplicando particionamiento de mapa de 1500x1500 celdas

factores están intrínsecamente interrelacionados, de modo que no son independientes y no es posible elegir o fijar los valores de los tres de forma independiente.

Así, se plantea definir un conjunto de ecuaciones matemáticas que, dado un tamaño del mapa de la zona que cubre un área (*Area*) y un máximo tiempo de ejecución t, nos permite determinar la resolución del mapa, el grado de solapamiento y la cantidad de partes que deben ser utilizados para cumplir con las limitaciones de tiempo y alcanzar la menor diferencia posible en la predicción de la propagación de incendios forestales.

Por ejemplo, un mapa de 45 x 45 kilómetros con una resolución de 30 x 30 metros tiene un total de 1500 x 1500 celdas. Este mapa se puede dividir de diferentes formas. Se puede dividir en 4 x 4, 5 x 5 o 6 x 6 partes. Teniendo en cuenta que el tamaño máximo por cada parte debe ser de 400 x 400, en los 4 x 4 partición, el solapamiento 12 celdas por lado, para 5x5 particiones, es de 50 celdas por lado y en los 6 x 6 particiones, con un solapamiento de 75 celdas por cada lado. Esta situación es análoga a la que se representa en la Figura 3.1. Particionar el mapa en más partes nos permite introducir un solapamiento más grande, pero la partición en un mayor número de partes provoca un error más grande que compensan negativamente el efecto de introducir más solapamiento.

Por otra parte, el mismo mapa de 45 x 45 kilómetros se puede representar mediante una resolución más baja (50 x 50 metros) con un total de 900 x 900 celdas. Dicho mapa se puede dividir en 3 x 3 partes con un solapamiento de 50 celdas por lado, o en 4 x 4 partes, con un solapamiento de 87 celdas por cada lado. De esta manera, la partición tiene menos partes y más solapamiento, pero con una resolución inferior.

Cuando se particiona el mapa con solapamiento, cada parte resultante del mapa se compone de un cierto número de celdas que son las celdas efectivas de esa parte del mapa (NCP_{ef}) y un cierto número de celdas que conforman el solapamiento (NCP_{ov}). El número total de celdas que se debe calcular para cada parte es $NCP = NCP_{ef} + NCP_{ov}$. NCP debe ser como máximo $\lambda(t)$, ya que esto representa el número de celdas que se pueden procesar en t segundos. NCP_{ef} se puede calcular a partir de la ecuación 3.10,

$$NCP_{ef} = \frac{NT}{N_p}, \tag{3.10}$$

donde NT es el número total de celdas del mapa y N_p es el número de partes. El número total de celdas de un mapa que representa un cierto área de terreno con una resolución dada puede ser calculado por la ecuación 3.11,

$$NT = \frac{Area}{R^2}, \tag{3.11}$$

donde $Area$ es el área total del mapa y R es la resolución.

En primer lugar, se ha analizado el efecto del solapamiento en los resultados de tiempo de ejecución y de la diferencia del campo de vientos y a su vez como esta diferencia afecta a la predicción de la propagación, manteniendo constante el $Area$, el número de particiones (N_p), la resolución (R) y la velocidad del viento (Sp) y variando exclusivamente solapamiento de la partición (Ov). Una muestra de los resultados obtenidos se presenta en la Tabla 3.4. En esta tabla, se puede observar que para un mapa de 24x24 km, manteniendo fijos el particionamiento, la resolución y la velocidad del viento, el hecho de aumentar el solapamiento por lado, aumenta el número total de celdas por parte ($NCells$ o $\lambda(t)$), y, en consecuencia, el tiempo de ejecución (t), pero reduce la diferencia en el campo de viento (NC_1 y $RMSE$) y en la predicción de la propagación del fuego (D). En este caso se ha introducido una nueva medida de la diferencia en el campo de vientos que hemos denominado NC_1, que representa el número de celdas con una diferencia mayor de 1 km/h en la velocidad del viento. La diferencia en la predicción de la propagación D representa la diferencia simétrica normalizada expuesta en la Ecuación 3.2. Aumentar el solapamiento a 50, 75 o incluso 100 celdas por cada lado no mejora sensiblemente el campo de viento ni la predicción de la propagación de incendios forestales.

El siguiente estudio consiste en mantener fijo el mapa de elevaciones ($Area$), la velocidad del viento (Sp), el número total de celdas por parte ($NCells$ o $\lambda(t)$) y la resolución del mapa (R), y cambiar el número de particiones (N_p) para dividir el mapa en 4 x 4, 5 x 5 y 6 x 6 partes. Cuando el número de partes se incrementa, el número efectivo de celdas por parte disminuye y la cantidad de solapamiento se puede aumentar sin exceder

Area	N_p	NCP_{ef}	R (m)	Ov	$NCells$ (o $\lambda(t)$)	t (s)	NC_1	$RMSE$ (km/h)	D
24x24	4	160000	30	0	160000	100	0,140	9,28	0,38
24x24	4	160000	30	25	200625	104	0,080	2,98	0,07
24x24	4	160000	30	50	242500	106	0,080	2,93	0,07
24x24	4	160000	30	75	285625	110	0,070	2,86	0,06
24x24	4	160000	30	100	330000	115	0,067	2,83	0,06

Tabla 3.4: Variación de la diferencia del campo de vientos y de la predicción de la propagación al incrementar el solapamiento

el límite de tiempo de ejecución. Un ejemplo de estos de resultados se muestran en la Tabla 3.5 y en la Figura 3.7. Se puede observar que hay un efecto compensatorio entre el aumento de solapamiento, que tiende a reducir la diferencia en la predicción de la propagación, y el aumento del número de partes, que tiende a aumentar la diferencia. En el ejemplo de la Tabla 3.5, se puede observar que dividir el mapa en 4 x 4 partes requiere un solapamiento de 25 celdas por lado y la diferencia obtenida es de $0,07$, mientras que dividir el mapa en 6 x 6 partes requiere un solapamiento de 88 celdas por lado para conseguir una diferencia de $0,08$. En el primer caso (4 x 4 particiones), sólo se requieren 16 nodos, mientras que en el segundo caso (6 x 6 particiones), es necesario el uso de 36 nodos para llegar a una diferencia similar. Por lo tanto, se puede concluir que el aumento del número de partes no mejora la diferencia y requiere más recursos.

Area	Sp (km/h)	$NCells$ (o $\lambda(t)$)	t (s)	R (m)	N_p	NCP_{ef}	Ov	NC_1	$RMSE$ (km/h)	D
45x45	20	180625	100	30	16	140625	25	0,22	3,73	0,07
45x45	20	180625	100	30	25	90000	63	0,20	3,48	0,07
45x45	20	180625	100	30	36	62500	88	0,21	3,58	0,08

Tabla 3.5: Variación de la diferencia del campo de vientos y de la predicción de la propagación al cambiar el número de partes

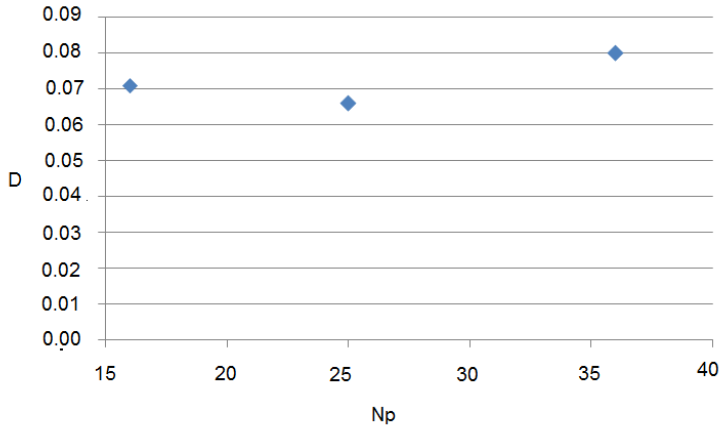


Figura 3.7: Diferencia en la propagación del fuego cuando cambiamos el número de partes

El siguiente estudio fue llevado a cabo para analizar el efecto de la resolución del mapa. Por lo tanto, para un mapa determinado de área ($Area$), dividido en N_p partes con un solapamiento de Ov celdas por lado y teniendo en cuenta una velocidad del viento Sp, la resolución del mapa R se modifica para determinar su efecto sobre el tiempo de ejecución del cálculo del campo de viento, sobre la diferencia en el campo de vientos y sobre la diferencia de propagación del fuego. La Tabla 3.6 muestra un ejemplo de este análisis. En este caso, el mapa en cuestión es de $39x39$ Km. Se divide en 9 partes (3x3) con un solapamiento de 25 celdas por lado. La resolución del mapa se cambia de 30 a 80 metros para calcular el campo de viento, aunque, una vez que el campo de viento se ha calculado la resolución dada, se interpola a 30 metros para comparar los resultados. En esta tabla, se puede observar que, con 9 partes y una resolución de mapa de 30 metros, la diferencia de propagación de incendios forestales es muy pequeña, pero el tiempo de ejecución supera ligeramente 100 segundos. En el otro extremo, una resolución de 80 metros resuelve el campo de viento en sólo 24 segundos, pero genera una diferencia en la propagación del incendios de $0,14$, siendo demasiada elevada. Estos resultados también se presentan en la Figura 3.8.

Area	Sp (km/h)	N_p	Ov	R (m)	NCP_{ef}	$NCells$ (o $\lambda(t)$)	t (s)	NC_1	$RMSE$ (km/h)	D
39x39	20	9	25	30	187777	233611	130	0,14	4,58	0,03
39x39	20	9	25	40	105625	140625	62	0,27	10,48	0,05
39x39	20	9	25	50	67600	96100	45	0,29	10,73	0,07
39x39	20	9	25	60	46945	71111	34	0,30	10,73	0,07
39x39	20	9	25	80	26460	45227	24	0,35	12,73	0,14

TABLA 3.6: Variación de la diferencia del campo de vientos y de la predicción de la propagación al cambiar la resolución del mapa

En la Figura 3.8 se muestra como varía el error de la propagación del incendio en función de la resolución. Si se ha marcado que la diferencia límite entre propagaciones es de $0,1$, podemos observar que con la mínima resolución que se puede trabajar para un mapa es de 50 metros.

La Figura 3.9 presenta la influencia de diferentes parámetros. En esta figura, cada línea corresponde a diferentes particionamientos y diferentes resoluciones y, para cada caso, hay varios solapamientos considerados. En esta figura, el tiempo de ejecución de 100 segundos se ha indicado (160000 celdas) y la diferencia máxima tolerable en la propagación del fuego de 0,1. Los puntos por encima de la línea horizontal representan un error inaceptable y los puntos en la parte derecha de la figura representan configuraciones que requieren tiempos de ejecución que son demasiado grandes.

Teniendo en cuenta todos estos estudios, se ha aplicado una regresión múltiple para determinar las ecuaciones que nos permiten determinar la mejor manera de dividir el

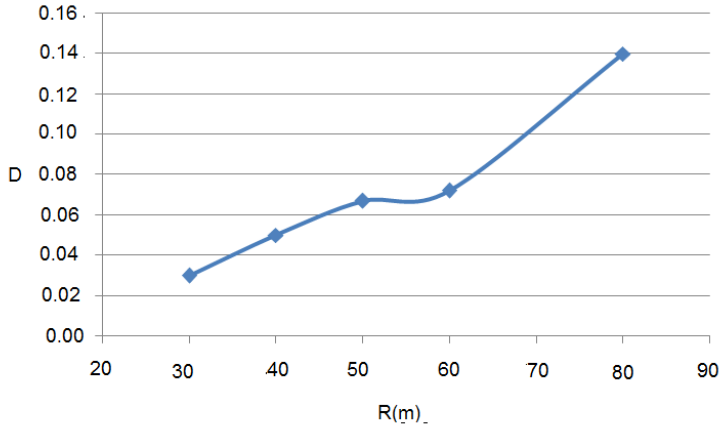


FIGURA 3.8: Diferencia en la propagación del fuego al variar la resolución del mapa

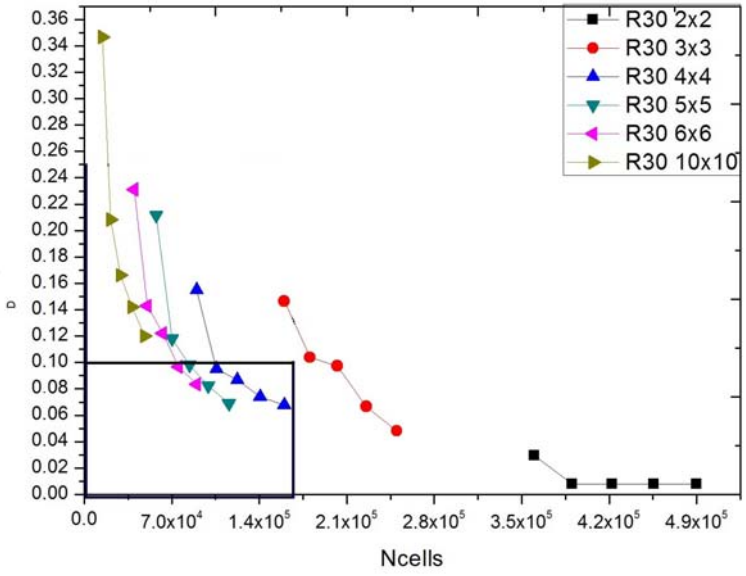


FIGURA 3.9: Diferencia en la propagación del fuego al variar diferentes parámetros

mapa. Para llevar a cabo esta regresión múltiple, se ha tomado como parámetros de entrada considerados son: el área del mapa ($Area$), expresada en metros cuadrados, y el tiempo de ejecución (t) en segundos, que fija el número máximo de las celdas en una partición ($\lambda(t)$). La diferencia en la propagación de incendios se consideró como un parámetro limitante ya que siempre debe ser inferior a $0,1$. A partir de esta regresión múltiple, se obtuvieron las ecuaciones de la resolución del mapa y del solapamiento. En la ecuación 3.12, se fija que la resolución del mapa debe estar entre 30 y 60 metros. Esto se debe al hecho de que la resolución normal del mapa es de 30 metros y que no tiene sentido establecer un mapa de resolución mayor, ya que los datos no suele estar disponible hoy en día. Por otra parte, se ha observado que, cuando la resolución del mapa

es de más de 60 metros (por ejemplo, 80 metros), el efecto de campo de viento se suaviza a causa de la baja resolución y la aplicación del campo de viento no es significativa. En cuanto a la ecuación 3.13, se requiere un solapamiento mínimo de 25 celdas por lado para asegurar que los efectos de borde se reducen al mínimo.

$$\begin{aligned} R = & Min\{Max\{30, \\ & 32,91 - 4,092 \cdot 10^{-5}\lambda(t) + 9,684 \cdot 10^{-9}Area - 3,963 \cdot 10^{-14}\lambda(t) \cdot Area\}, \\ & 60\} \end{aligned} \tag{3.12}$$

$$Ov = Max\{38,70 + 2,32 \cdot 10^{-4} \cdot \lambda(t) - 1,41 \cdot 10^{-13}\lambda(t) \cdot Area, 25\} \tag{3.13}$$

A partir de estas ecuaciones, es posible determinar el número de partes (N_p) como se muestra en la Ecuación 3.14.

$$N_p = \sqrt{\frac{Area}{R^2(\lambda(t) - 2 \cdot Ov)}} \tag{3.14}$$

A continuación, se combinan las ecuaciones 3.10, 3.11, 3.12, 3.13 y 3.14, y es posible expresar el total del número de celdas por partición como una función de los anteriores, como se muestra en la ecuación 3.15.

$$NCP = \left(\sqrt{\frac{Area}{R^2 N_p}} + 2 \cdot Ov\right)^2 \tag{3.15}$$

3.7.1 Efecto de la velocidad del viento

Si la velocidad del viento es baja en general (5 mph), la diferencia en la propagación de incendios forestales no es demasiado significativa, en el caso del particionamiento del mapa, debido a que la velocidad del viento obtenida en cada celda presenta ligeras variaciones haciendo que el error generado para este mapa sea pequeño. Sin embargo, para velocidades del viento superiores, como 15 y 20 mph, los valores obtenidos en el campo de viento tendrán un rango de mayor variación.

Por lo tanto, es necesario determinar el efecto de la velocidad del viento respecto a la diferencia generada por el particionamiento del mapa. Se ha llevado a cabo una extensa experimentación para determinar la correlación existente la velocidad del viento y el particionamiento del mapa. Para diferente tamaño de mapa y diferente velocidad del

viento, se puede determinar el número mínimo de celdas por partición para mantener la propagación de incendios forestales con una diferencia de menos de $0,1$. Este resultado queda representado de forma gráfica en la Figura 3.10. El gráfico muestra, para cada velocidad del viento en km/h, el número mínimo de celdas por parte, incluyendo el solapamiento, en función del tamaño del mapa original, para asegurar que la diferencia en la propagación de incendios forestales sea menor de $0,1$.

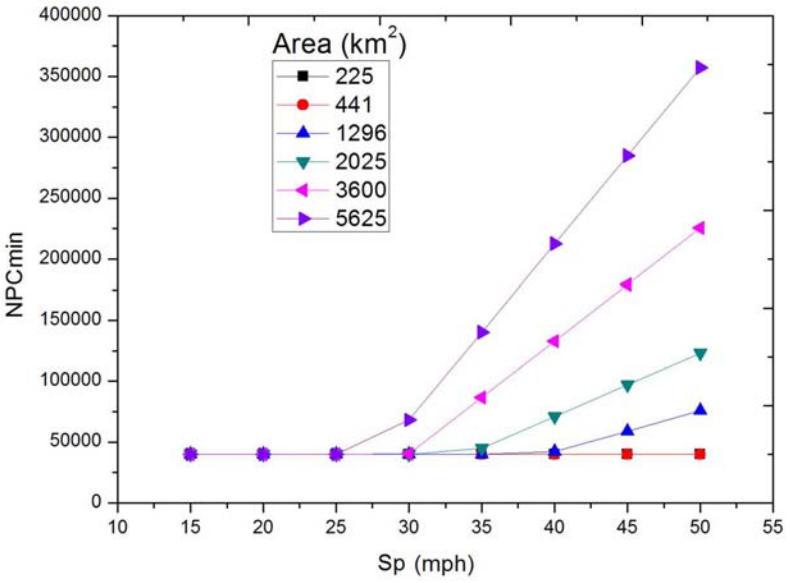


FIGURA 3.10: Mínimo número de celdas por partición para un tamaño de mapa dado y una velocidad del viento.

La regresión lineal, ha determinado que el número mínimo de celdas por partición se puede expresar como se muestra en la Ecuación 3.16,

$$NCP_{min} = Max\{40000; x + y \cdot NT + z \cdot NT \cdot Sp\}, \tag{3.16}$$

donde NT es el número total de celdas del mapa, Sp es la velocidad del viento, x, y y z son los términos obtenidos de la regresión múltiple. A partir de los experimentos, se ha obtenido que $x = -8,63 \cdot 10^3$, $y = -5,71 \cdot 10^{-2}$ y $z = +2,31 \cdot 10^{-3}$. A partir de esta ecuación, se obtiene el número mínimo de celdas que debe tener una partición (incluyendo el solapamiento) para que la diferencia máxima para la predicción de la propagación de incendios sea inferior $(0,1)$. Este número mínimo de celdas no debe exceder el número máximo de celdas por partición que ha sido previamente determinados para lograr el tiempo máximo de ejecución (por ejemplo, para 100 segundo, significa 160000 celdas). Por tanto, es necesario comprobar la viabilidad de mantener ambos límites.

Sustituyendo la ecuación 3.11 en la ecuación 3.16, esta ecuación se puede expresar como se muestra en la Ecuación 3.17.

$$NCP_{min} = x + y\frac{Area}{R^2} + z\frac{Area}{R^2} \cdot Sp \tag{3.17}$$

Debe cumplir que:

$$NCP_{min} \leq NCP \leq \lambda(t) \tag{3.18}$$

Se puede observar que, para los mapas pequeños o de baja velocidad del viento, el número mínimo está por debajo del máximo determinado por el tiempo de ejecución. La aplicación de la ecuación 3.17 para un mapa 36 x 36 Km y una velocidad del viento de 40 km/h, el número mínimo de celda por partición obtenidos a partir de la ecuación 3.17 es de 20060 celdas, que es claramente inferior a 160000 celdas.

Sin embargo, cuando el mapa es más grande (75 x 75 Km) y la velocidad del viento es significativa (40 km / h), el número mínimo de celdas obtenidas a partir de la ecuación 3.17 es 212745, que es mayor que el límite 160000 celdas. Esto significa que, en este caso, no es posible obtener un resultado que cumpla ambas limitaciones y no hay partición posible que cumpla tanto el tiempo de ejecución y la diferencia predicción de la propagación. En esta situación, no es factible aplicar el cálculo del campo de viento a las predicciones de la propagación del fuego en caso de emergencia operativas reales.

3.8 Metodología para determinar el particionamiento del mapa

A partir de todo el estudio y análisis que se ha llevado a cabo, se ha definido una metodología [48] que, aplicando las ecuaciones descritas en los apartados anteriores, permite minimizar el error producido por el particionamiento del mapa, cumpliendo el límite del tiempo de ejecución y la restricción de la limitación en la diferencia en la predicción de la propagación de los incendios. Esta metodología puede resumirse en los siguientes puntos:

Dado un mapa de área $Area$ y un tiempo de ejecución máximo t, es posible estimar:

1. el valor de $\lambda(t)$ para la expresión 3.4,
2. la resolución del mapa R, determinada por la ecuación 3.12 y

3. el solapamiento Ov, mediante la ecuación 3.13.

4. Una vez que se han determinado la resolución y el solapamiento, es posible calcular el número de partes N_p, utilizando la ecuación 3.14.

5. El número de celdas por partición (NCP) se calcula aplicando la ecuación 3.15.

6. Después de eso, es necesario para determinar el número mínimo de celdas por partición NCP_{min} debido a la velocidad del viento a partir de la Ecuación 3.17

7. Por último, es necesario estimar la ecuación 3.18 para determinar si el mapa se puede resolver dentro del tiempo límite de t con una diferencia inferior a $0,1$.

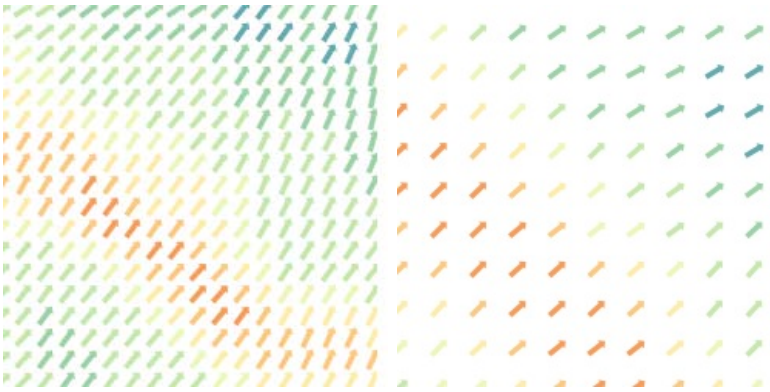
3.8.1 Validación de la metodología

Para llevar a cabo la validación de la metodología desarrollada, se llevó a cabo una experimentación amplia con mapas de distintos tamaños, distintas orografías, distintas velocidades de viento, etc.

El primer experimento se llevó a cabo en un mapa del terreno correspondiente a la Jonquera (España), en un lugar donde hubo un incendio en julio de 2012. El área del mapa es de aproximadamente 50 x 50 Km. El tiempo de ejecución requerido para calcular este mapa completo con una resolución de 30 m es 1402 segundos. Teniendo en cuenta un tiempo máximo de 100 s, la ecuación 3.4 ofrece $\lambda(t)$ en 206533. La resolución del mapa se determina que es 30 m mediante la aplicación de la ecuación 3.12. El solapamiento obtenido de la fórmula 3.13 es 50, y el número de partes a partir de la ecuación 3.14 es 16. Teniendo en cuenta una velocidad del viento de 40 Km / h, la condición de la ecuación 3.17 también se cumple. La diferencia en la predicción de la propagación del fuego teniendo en cuenta el campo de vientos generado a partir del mapa completo o el campo de vientos con el mapa de particionado es $0,8$.

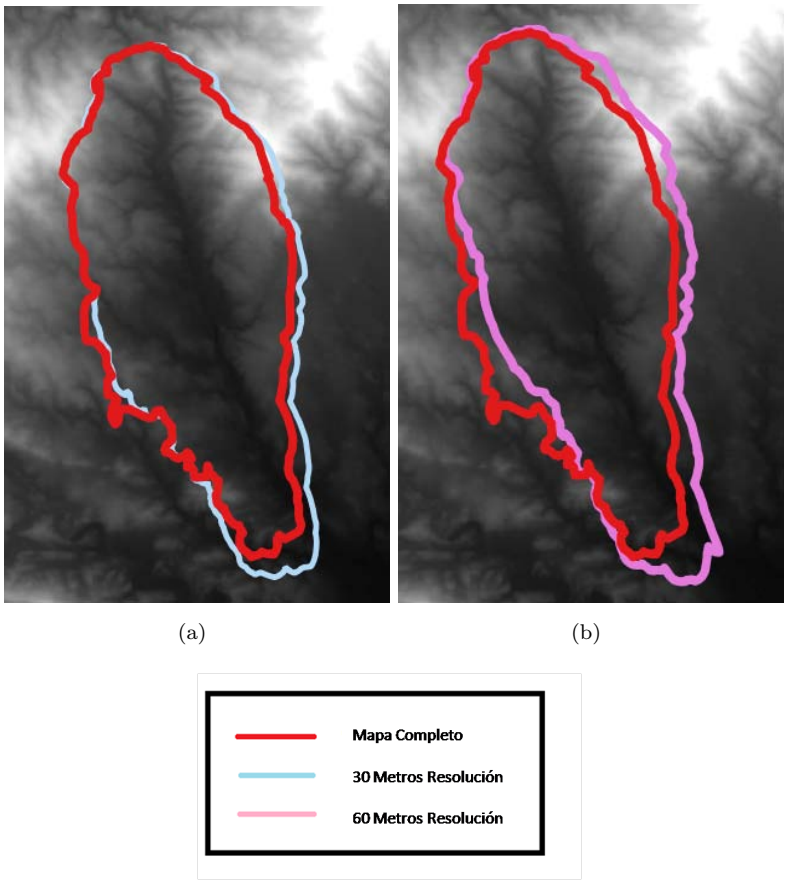
Si el tiempo máximo de ejecución considerado es ahora 50 s, el valor de $\lambda(t)$ obtenido de la fórmula 3.4 es 100000, la resolución es de 40 m, el solapamiento es de 40 celdas por cada lado y el número de partes es 25. En este caso, la diferencia en la propagación de incendios forestales es de $0,09$. Un detalle de los campos de viento obtenidos en ambos casos se muestra en la Figura 3.11. Se puede observar que el campo de viento a una resolución 30 m es más detallada que la que se obtiene en una resolución de 60 m. Ambas propagaciones de fuego se muestran en la Figura 3.12.

El segundo experimento se llevó a cabo con un fuego sintético. El área del mapa es de aproximadamente 45x45 Km. El tiempo de ejecución requerido para calcular este mapa completo a una resolución de 30 m es de 1253 segundos. Teniendo en cuenta un



(a) (b)

FIGURA 3.11: Detalle del campo de vientos para una zona correspondiente al incendio de La Junquera de 2012 a una resolución de 30m y 60m.



(a) (b)

FIGURA 3.12: Propagación del incendio forestal para una zona correspondiente al incendio de La Junquera de 2012 a una resolución de 30m y 60m.

tiempo máximo de 78 s, la ecuación 3.4 ofrece $\lambda(t)$ en 144400. La resolución del mapa se determina que es 30 m al aplicar la ecuación 3.12. El solapamiento obtenido de la ecuación 3.13 es de 50 celdas por lado, y el número de partes a partir de la ecuación 3.14 es de 25. Teniendo en cuenta una velocidad del viento de 20 Km / h, la condición de la

ecuación 3.17 también se cumple. La diferencia en la propagación del fuego teniendo en cuenta el mapa completo y el número de particiones es $0,7$.

Si el tiempo máximo de ejecución considerado es de 20 s, el valor de $\lambda(t)$ obtenido de la ecuación 3.4 es 10000, la resolución es de 60 m, el solapamiento es de 40 celdas por cada lado y el número de partes es de 49. En este caso, la diferencia en la propagación de incendios forestales es de $0,16$, que no es factible. Se puede observar que el número mínimo de celdas por partición obtenida a partir de la expresión 3.17 es mayor que el resultado obtenido de la fórmula 3.4. Esto significa que la ecuación 3.18 no se ha satisfecho, y el sistema no puede ser resuelto en el tiempo especificado.

Un detalle de los campos de viento obtenidos en ambos casos se muestra en la Figura 3.13. A su vez, las predicciones de la propagación del incendio en ambos casos se muestran en la Figura 3.14. Se puede observar que existe una gran diferencia entre ambas propagaciones. Esto se debe al hecho de que el campo de viento es más suave con una resolución de 60 m. que con una resolución de 30 m.

(a) (b)

FIGURA 3.13: Detalle del campo de vientos para un terreno sintético a una resolución de 30m y 60m.

3.9 Conclusiones del capítulo

El particionamiento del mapa de terreno para el cálculo de campo de vientos con solapamiento entre las partes es una aproximación que permite explotar el paralelismo inherente en la aplicación, aunque requiere la introducción de un cierto grado de solapamiento entre las partes en las que se divide el mapa del terreno. Por otra parte, el incremento del numero de partes en las que particiona el mapa provoca la aparición de un cierto error en el campo de vientos generados, que no se corrige con la introducción del solapamiento entre las partes. La metodología desarrollada permite determinar, si dado un mapa que cubre un área, un límite de tiempo y una velocidad del viento, puede hallarse una

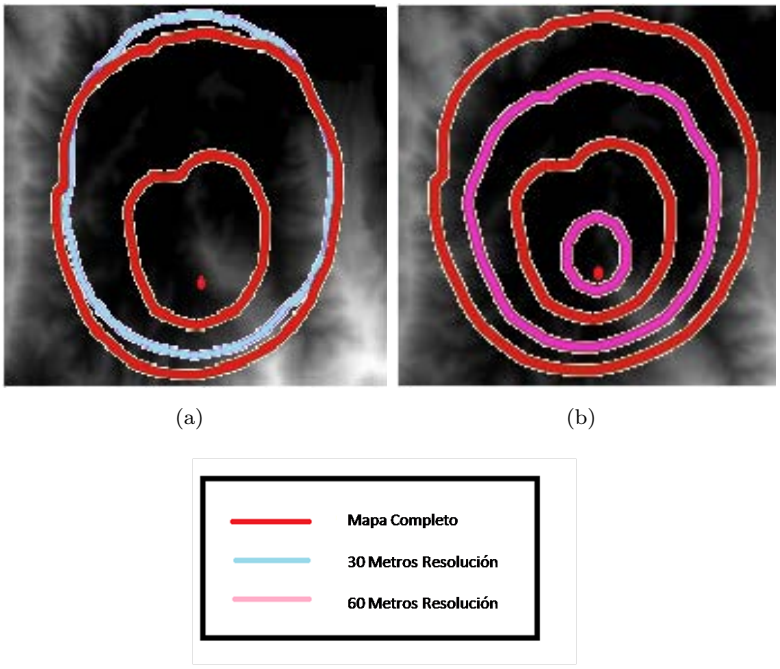


FIGURA 3.14: Propagación del incendio forestal para un terreno sintético a resolución de 30m y 60m.

partición que cumpla con dichas condiciones y proporcione una predicción que no se aleje más de un cierto límite de la predicción obtenida considerando un mapa global.

Esta metodología funciona de forma satisfactoria para un un número de casos, pero cuando el mapa del terreno tiene un número de celdas muy elevado o el límite de tiempo es muy restrictivo, no se puede conseguir cumplir con todos los condicionantes, de modo que es necesario aplicar más técnicas de paralelización para conseguir cumplir con todos los requerimientos del problema.

Capítulo 4

Descomposición del dominio

4.1 Introducción

En el capítulo anterior se ha propuesto una metodología para particionar el mapa del terreno para el cual se desea calcular el campo de vientos, con el fin de realizar el cálculo del campo de vientos para cada una de las particiones en paralelo, y así reducir el tiempo de ejecución. Se ha mostrado que esta metodología tiene ciertas limitaciones y es necesario explotar otras formas de paralelismo para obtener el campo de vientos en los límites de tiempo establecidos.

Tal como se ha mostrado, el simulador de campo de vientos, WindNinja, realiza una discretización del espacio, formando una malla y aplica las ecuaciones correspondientes en los distintos puntos de la malla para obtener un sistema de ecuaciones, que puede representarse como $Ax = b$, donde A es la matriz que representa la malla del problema.

Cuando se desea calcular el campo de vientos para un terreno, se realiza una discretización del sistema en millones de elementos que constituyen la malla donde se describen los diferentes elementos del sistema y las relaciones o interacciones que existen entre ellos. Los elementos de la malla interaccionan entre ellos dando lugar al sistema lineal que ha medida que se vaya incrementado el tamaño del mapa, se aumenta el sistema a construir y, por tanto, el tiempo de resolución y la complejidad de la misma.

Una aproximación que se ha usado en la literatura para paralelizar la resolución de estos sistemas, y así reducir el tiempo de ejecución, consiste en realizar una descomposición del dominio (*Domain Decomposition*). En esta aproximación, lo que se pretende es paralelizar la resolución del sistema de ecuaciones. Esto es distinto del particionamiento del mapa, el objetivo era plantear sistemas de ecuaciones más pequeños para cada parte

del mapa, mientras que en la aproximación de Descomposición del Dominio, lo que se pretende es paralelizar la resolución del sistema completo.

Existen dos formas de aplicar Descomposición del Dominio: con solapamiento y sin solapamiento. En los dos casos es necesario dividir el dominio en P subdominios, que corresponden a P bloques de la malla. La matriz A puede ser representada como un conjunto de P submatrices A^P. Cada subdominio o submatriz puede ser resuelto aplicando métodos convencionales como el gradiente conjugado (CG) o el gradiente conjugado con precondicionador (PCG). A continuación se presentan ambos métodos y se analizan los resultados obtenidos con cada uno de ellos.

4.2 Descomposición del Dominio con solapamiento: Método de Schwarz

El método de Descomposición del Dominio con solapamiento se conoce como método de Schwarz *Schwarz alternating method* [55] [56]. Este método consiste en resolver independientemente cada subdominio, respectando las condiciones de frontera de Dirichlet con los subdominios adyacentes. El método de Schwarz es una generalización del método iterativo tradicional de Gauss-Seidel por Bloques. El sistema de ecuaciones $Ax = b$ se cumple en todo el dominio Ω, y las condiciones de frontera $\partial\Omega$ satisfacen las condiciones de frontera de Dirichlet $x = c$.

La Figura 4.1 muestra un esquema simplificado del método de Descomposición del dominio con solapamiento. En este esquema, se asume que todo el dominio Ω se puede dividir en dos subdominios Ω_1 y Ω_2, con fronteras $\partial\Omega_1$ y $\partial\Omega_2$, respectivamente. Así, se cumple que Ω=$\Omega_1 \cup \Omega_2$. El solapamiento entre los dos subdominios se define como $\Omega_1 \cap \Omega_2$. La parte de $\partial\Omega_1$ dentro de Ω_2 define una frontera artificial Γ_1. Análogamente, Γ_2 la frontera artificial dentro Ω_1.

El efecto de dividir la malla es equivalente a realizar una división de la matriz A en submatrices que representan los distintos subdominios. En forma resumida podemos decir que una vez que el dominio ha sido dividido en subdominios, el sistema se resuelve para cada subdominio de forma independiente. Los valores obtenidos en los nodos correspondientes a las fronteras Γ_1 y Γ_2 son intercambiados entre los subdominios contiguos. Si dichos valores tienen una diferencia mayor que la tolerancia ϵ establecida, los valores obtenidos en el solapamiento en la resolución de un subdominio son transferidos al subdominio adyacente y el nuevo valor es incorporado para la siguiente iteración, hasta que se alcanza una solución que tenga una diferencia en los nodos frontera inferior al valor de tolerancia. Este método se muestra en el Algoritmo 4.1.

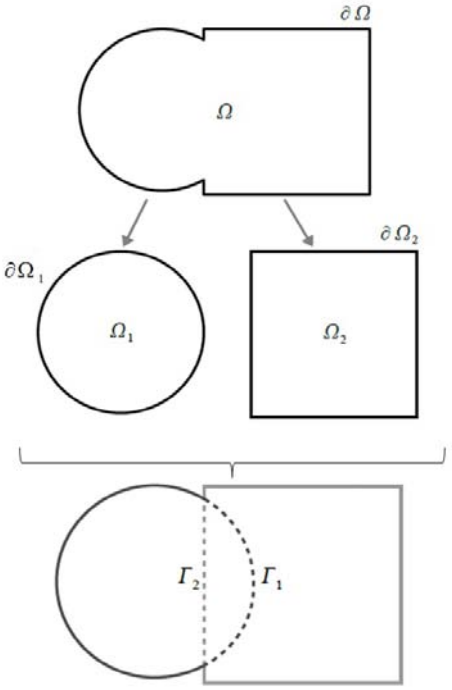


FIGURA 4.1: Descomposición de la malla en dos subdominios con solapamiento

En los siguientes apartados de esta sección se mostrará la aplicación del método de Descomposición del Dominio con solapamiento al caso concreto del cálculo del campo de vientos, y de WindNinja en particular.

4.2.1 Aplicando Descomposición del Dominio con solapamiento a WindNinja

La malla de WindNinja se forma a partir del mapa de elevaciones al aplicar la ecuación de la conservación de la masa. En la Figura (Figura 4.2) se muestra una malla esquemática a modo de ejemplo.

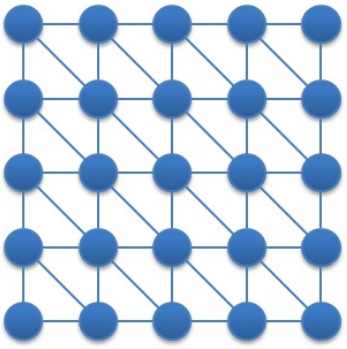


FIGURA 4.2: Malla de WindNinja

Para esta malla, el sistema de ecuaciones obtenido es $Ax = b$, donde A es una matriz dispersa. Los nodos de la malla se corresponden con los elementos de la diagonal

x_1^0,x_2^0 aproximación inicial
ϵ tolerancia
i $\leftarrow$ 0 número de iteraciones
mientras $\| x_1^i - x_1^{i-1} \| \leq \epsilon$ o $\| x_2^j - x_2^{j-1} \| \leq \epsilon$

Resolver

$$Ax_1^i = b \text{ en } \Omega_1$$
$$\text{con } x_1^i = c \text{ en } \partial\Omega_1$$
$$x_2^i \leftarrow x_1^i \text{ en } \Gamma_1$$
$$\text{i} \leftarrow \text{i+1}$$

Resolver

$$Ax_2^j = b \text{ en } \Omega_2$$
$$\text{con } x_2^j = c \text{ en } \partial\Omega_2$$
$$x_1^j \leftarrow x_2^j \text{ en } \Gamma_2$$
$$j \leftarrow j + 1$$

ALGORITMO 4.1: Algoritmo de resolución mediante Descomposición del Dominio con solapamiento

principal de la matriz y las iteraciones entre nodos se encuentran, de tal manera que cada elemento A_{ij} de la matriz A representa las interacciones entre los nodos i y j de la malla. En el caso particular de WinNinja, la matriz A tiene dos cualidades: la primera es que es simétrica y definida positiva, y segundo es que tiene un patrón independiente del terreno. Este patrón consiste en la diagonal principal y 26 subdiagonales (13 en cada triangular) distribuidas a lo largo de la matriz. Las posiciones que ocupan las subdiagonales dependen del mapa de elevación.

Para aplicar el método de Schwarz, el primer paso consiste particionar la malla, tratando de tener el mismo número de nodos en todos los subdominios y el mínimo número de nodos posibles en la frontera. El particionamiento debe cumplir las siguiente normas: Los nodos en un subdominio sólo tienen interrelación con los otros nodos en el mismo subdominio o con los nodos de la frontera. Los nodos frontera entre dos subdominios adyacentes son considerados como nodos de ambos subdominios. Estas reglas se muestran en la Figura 4.3, donde los nodos de la frontera se muestran en color rojo.

Una vez se ha definido los subdominios y las fronteras, es necesario determinar el solapamiento entre los subdominios adyacentes. El mínimo solapamiento de un subdominio está formada por todos los nodos del subdominio adyacente que están relacionados con los nodos de la frontera. Esta situación se muestra en la Figura 4.4, donde los nodos frontera están representados en color rojo y los nodos de solapamiento en color verdes. Sin embargo, es posible definir un solapamiento más grande que contenga más nodos

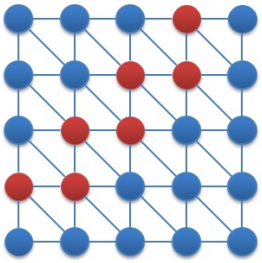


FIGURA 4.3: Frontera entre subdominios adyacentes

del subdominio adyacente. En general, aumentar el solapamiento genera que el sistema global converja más rápidamente.

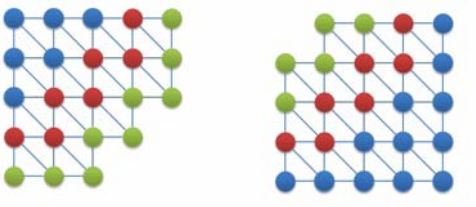


FIGURA 4.4: Solapamiento entre subdominios adyacentes

Por lo tanto, cada subdominio está formado por tres partes: los nodos internos del subdominio, los nodos frontera y los nodos solapamiento.

El hecho que las matrices generadas por WindNinja respondan a un patrón concreto, muy bien definido, implica que no es necesario el uso de una herramienta para el particionamiento de la malla, como METIS [57], que debe llevar a cabo un profundo análisis de la matriz para determinar la partición óptima del subdominio. En el caso particular de WindNinja, es posible determinar los subdominios del patrón de la matriz, sin realizar un análisis previo, lo cual simplifica la operación de forma considerable.

El número mínimo de nodos(n_{min}) de un subdominio es la distancia entre la diagonal principal y la última subdiagonal de la matriz original A. El número de nodos en la frontera es por tanto $n_{min} - 1$ y el número de nodos en el solapamiento es también $n_{min} - 1$. De esta manera, el número mínimo de nodos en un subdominio se puede expresar como se muestra en la Ecuación 4.1.

$$\begin{array}{ll} 3n_{min} - 2 & A^0, A^{P-1} \\ 5n_{min} - 4 & A^1 \dots A^{P-2} \end{array} \tag{4.1}$$

La partición de la matriz obtenida a partir de la aplicación del método se muestra en la figura 4.5.

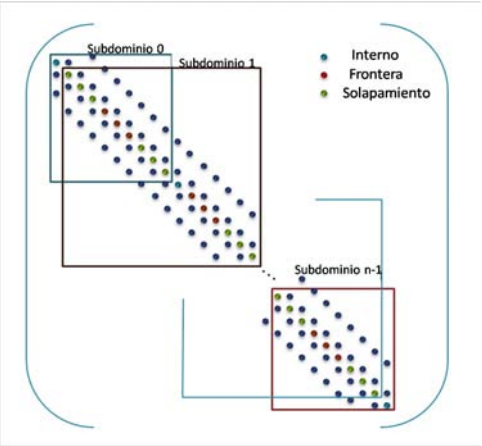


FIGURA 4.5: Particionamiento de la matriz A

Una vez que la malla, y por tanto la matriz A, ha sido particionada, el sistema debe ser resuelto. Cada subdominio es resulto independiente usando el gradiente conjugado con precondicionador (PGC). Este método ya ha sido descrito en el Algoritmo 2.1.

Una vez que cada subdominio ha sido resuelto, el valor correspondiente a la frontera es comparado con los valores de la frontera del subdominio adyacente. Si el valor obtenido es superior al valor de tolerancia establecida, entonces los valores correspondientes a los nodos de solapamiento de un subdominio son enviados al subdominio adyacente y se vuelve a solucionar el sistema utilizando estos nuevos valores. Dicho intercambio de valores se muestra en las Figuras 4.6 y 4.7

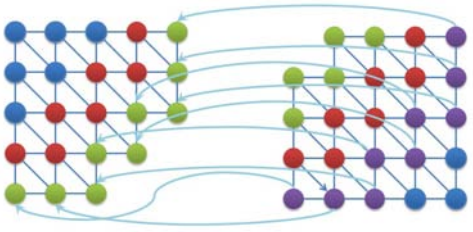


FIGURA 4.6: Intercambio de los valores del solapamiento entre el subdominio 1 al 2

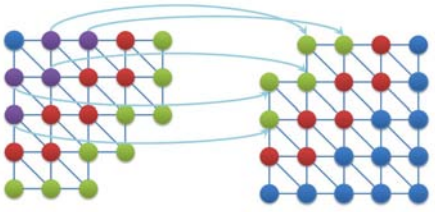


FIGURA 4.7: Intercambio de los valores del solapamiento entre el subdominio 2 al 1

El proceso finaliza cuando el valor correspondiente a los nodos frontera de un subdominio y su adyacente presentan un valor de tolerancia menor al impuesto.

4.2.2 Paralelización del método de Schwarz

Como se describió anteriormente, el método de Schwarz, presenta diferentes ventajas y posibilidades de explotar el paralelismo del sistema. El método ha sido implementado como una aplicación MPI Master-Worker. La organización general se muestra en la Figura 4.8.

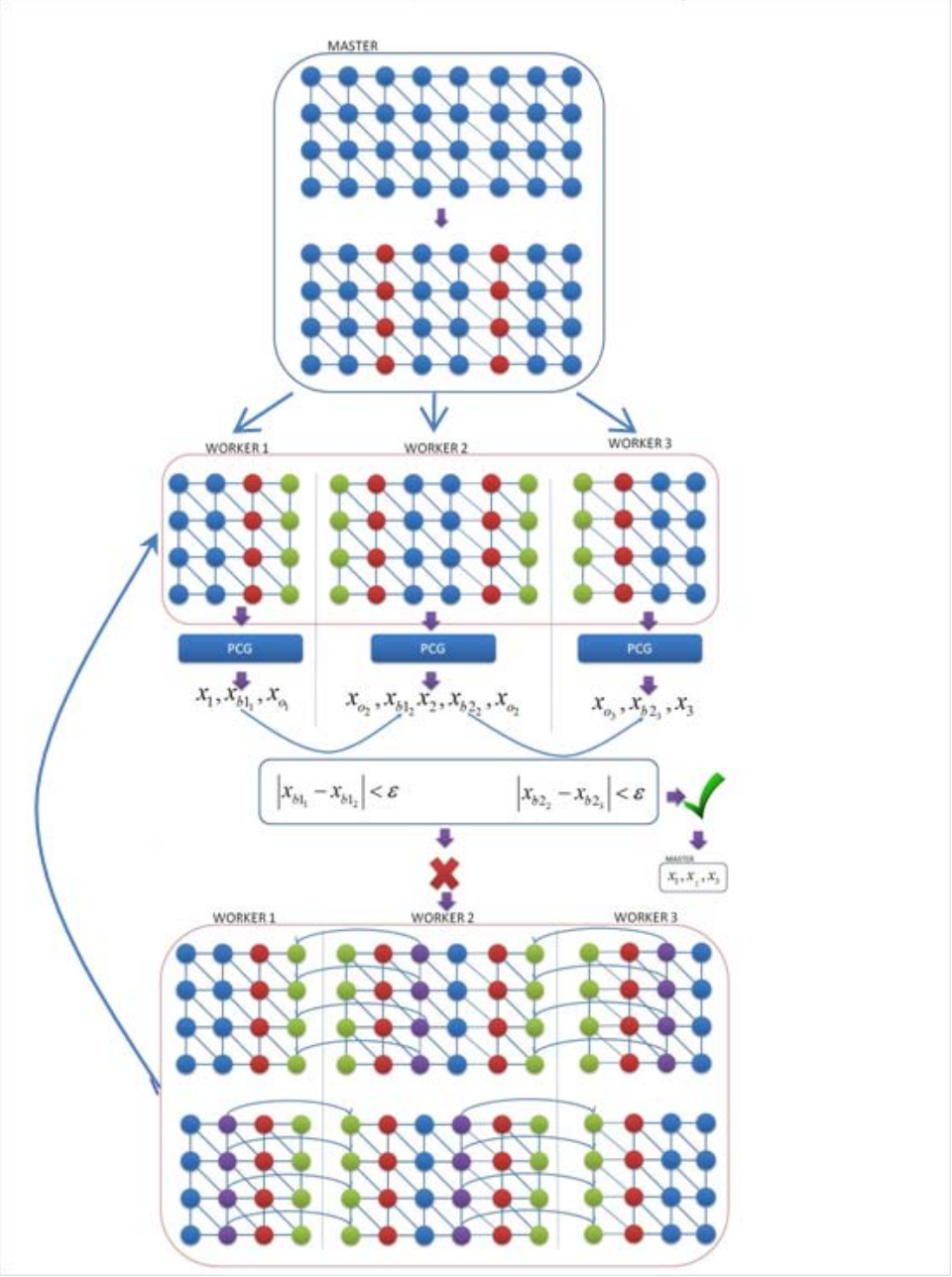


FIGURA 4.8: Paralelización Master/Worker del método de Schwarz

Primeramente, en el Master se descompone la malla y cada uno de sus subdominios es enviado a un Worker. En cada Worker se ejecuta el solver de forma independiente para cada subdominio. Una vez la solución inicial ha sido encontrada, cada Worker envía al

Worker con subdominios adyacentes los valores obtenidos en la frontera. Estos valores son comparados con los valores obtenidos en el propio subdominio y si la diferencia es mayor a la tolerancia impuesta, los Worker con subdominios adyacentes sustituyen los valores del solapamiento por los valores del solapamiento proporcionados por el subdominio adyacente y se vuelve a resolver de nuevo con estos nuevos valores. En el caso que la tolerancia sea inferior, se envía al Master la solución del sistema. Este esquema es diferente al clásico Master-Worker, en el sentido de que el Master distribuye los datos entre los Workers, pero luego estos procesos worker van intercambiando información entre ellos, para comprobar que se ha alcanzado la convergencia del sistema, y únicamente en el caso en el que se ha conseguido alcanzar la convergencia, se le envían los resultados al Master.

4.2.3 Análisis del método de Schwarz para WindNinja

El primer paso del método Schwarz consiste en particionar la malla en subdominios. El objetivo es dividir la malla entre un cierto número de subdominios de tamaño similar. Dado que el patrón de las matrices de WindNinja es conocido y muy regular, el tiempo de particionamiento de la matriz y del envío de cada subdominio de la matriz no es significativo en relación al tiempo de resolución del sistema.

La Tabla 4.1 recoge los resultados experimentales utilizando un Cluster basado en procesadores Dual-Core Intel Xeon 5150. Esta tabla muestra el tiempo de ejecución para resolver el campo de viento para mapas de 500x500 a 900x900 celdas. $ElemD$ es el número de elementos en cada subdominio, $\%Elem$ es el decremento en porcentaje del número de nodos respecto al dominio original, NSd es el número de subdominios de la malla, I es el número máximo de iteraciones necesarias para resolver el sistema, T_t es el tiempo total para calcular el campo de viento y SP es el SpeedUp respecto al mapa sin particionar.

Se puede observar que el tiempo de ejecución disminuye cuando el número de subdominios se incrementa hasta alcanzar 6 subdominios. Este comportamiento es debido al hecho de que aumentar el número de partes no se reduce el tamaño de las fronteras y el solapamiento entre los subdominios, y por lo tanto, los nodos internos representan una parte menos significativa del cálculo. Al aumentar el tamaño del mapa, el número de iteraciones requeridas para resolver el sistema aumenta significativamente. Para los mapas de 500x500 el SpeedUp máximo es de alrededor $5,3$. Este aumento de SpeedUp se reduce a medida que aumenta el tamaño de los mapas: para 600x600 es de alrededor de $4,9$, para 800x800 es de alrededor de $4,4$, y finalmente para 900x900 es de alrededor de $4,3$.

Mapa (NCells)	$ElemD$	$\%Elem$	NSd	I	T_t (s)	SP
500x500	1708266	46,62	3	1140	95	2,76
500x500	1441599	54,95	4	1137	62	4,20
500x500	1281599	59,95	5	1115	55	4,75
500x500	1174933	63,28	6	1085	54	4,80
500x500	1098742	65,66	7	1070	49	5,27
500x500	1041600	67,45	8	1010	49	5,29
500x500	997155	68,84	9	992	51	5,16
600x600	2668666	46,63	3	1229	168	2,71
600x600	2251999	54,96	4	1227	121	3,78
600x600	2001999	59,96	5	1236	104	4,40
600x600	1835333	63,29	6	1229	100	4,56
600x600	1716285	65,67	7	1240	94	4,85
600x600	1627000	67,46	8	1203	93	4,90
600x600	1557555	68,85	9	1199	97	4,70
800x800	6829866	46,64	3	1353	501	2,80
800x800	5763199	54,97	4	1373	364	3,86
800x800	5123199	59,97	5	1369	371	3,79
800x800	4696533	63,31	6	1355	318	4,42
800x800	4391771	65,69	7	1371	316	4,44
800x800	4163200	67,47	8	1322	318	4,42
800x800	3985422	68,86	9	1335	319	4,40
900x900	8643599	46,64	3	1423	690	2,70
900x900	7293599	54,98	4	1447	494	3,77
900x900	6483599	59,98	5	1452	492	3,78
900x900	5943600	63,31	6	1439	432	4,31
900x900	5557885	65,69	7	1450	435	4,28
900x900	5268600	67,48	8	1403	436	4,27
900x900	5043600	68,87	9	1423	433	4,30

TABLA 4.1: Tiempo de ejecución de WindNinja aplicando el método de Schwarz

4.2.4 Explotando el paralelismo OpenMP en el método de Schwarz

En este apartado se va a realizar un análisis de los precondicionadores SSOR y Jacobian, que son los que dispone Windninja, y se intentará explotar la paralilización OpenMP que incorpora WindNinja para cada una de los subdominios que han sido distribuidos utilizando MPI.

Las Tablas 4.2 y 4.3 resumen los resultados experimentales cuando se utiliza un cluster AMD Opteron 6376 de 16 cores con un procesador 5150. Estas tablas muestran el tiempo de ejecución para resolver el campo de viento para los mapas de 400 x 400, 1200 x 1200 celdas, teniendo en cuenta los precondicionadores SSOR y Jacobian, respectivamente. WN es el tiempo de ejecución de WindNinja original, $Tiempo$ es el tiempo total para calcular el campo de viento y $SpeedUp$ es el SpeedUp obtenido en comparación con la implementación de WindNinja sin descomposición de dominio.

Mapa	Tiempo (s) Subdomimios					Speedup Subdominios			
	WN	2	3	4	5	2	3	4	5
400x400	148	58	50	44	42	2,54	2,96	3,36	3,52
600x600	296	130	114	78	75	2,27	2,60	3,79	3,95
800x800	442	232	168	111	108	1,91	2,63	3,98	4,09
1000x1000	1677	698	470	378	354	2,40	3,57	4,44	4,74
1200x1200	2198	880	615	600	578	2,50	3,57	3,66	3,80

Tabla 4.2: Tiempo de ejecución y SpeedUp obtenido aplicando el método de Schwarz usando el precondicionador SSOR

Map Size	Tiempo (s) Subdominios					Speedup Subdominios			
	WN	2	3	4	5	2	3	4	5
400x400	142	79	54	53	37	1,81	2,65	2,67	3,79
600x600	287	166	115	90	82	1,73	2,50	3,18	3,50
800x800	480	294	193	149	138	1,63	2,49	3,21	3,47
1000x1000	1968	1028	635	503	454	1,91	3,10	3,91	4,33
1200x1200	2525	1072	742	610	624	2,36	3,40	4,14	4,05

Tabla 4.3: Tiempo de ejecución y SpeedUp obtenido aplicando el método de Schwarz usando el precondicionador Jacobian

Estos resultados muestran que el método de Schwarz ofrece una mejora significativa en el tiempo de ejecución y un buen SpeedUp. En algunos casos, se llega a un SpeedUp lineal debido a la reducción en el tamaño del sistema a resolver. La Figura 4.9 muestra el tiempo de ejecución cuando se cambia el número de subdominios considerando los precondicionadores SSOR y Jacobian para un mapa 1200 x 1200 celdas. Se puede observar que el precondicionador SSOR proporciona un mejor tiempo de ejecución, aunque ambos precondicionadores siguen la misma tendencia.

Sin embargo, desde un punto de vista operativo, un tiempo de ejecución de 610 o 624 segundos es demasiado grande para una operación real. Por lo tanto, es necesario mejorar el tiempo de ejecución.

Una posibilidad consiste en explotar el paralelismo OpenMP de la aplicación WindNinja original del PCG. Por lo tanto, una aplicación híbrida con la paralelización de descomposición de dominios MPI y OpenMP paralelización PCG puede mejorar el tiempo de ejecución. Esta aplicación ha sido implementada y probada en diferentes mapas de terreno. Las Tablas 4.4 y 4.5 muestran el tiempo de ejecución y velocidad obtenidos para un mapa 800 x 800 al considerar diferente número de subdominios y aplicando diferentes números de cores para resolver cada subdominio .

En estas tablas, se puede observar que, usando un precondicionador Jacobian con 4 cores y 4 subdominios, el tiempo de ejecución se reduce desde 567 segundos con WindNinja

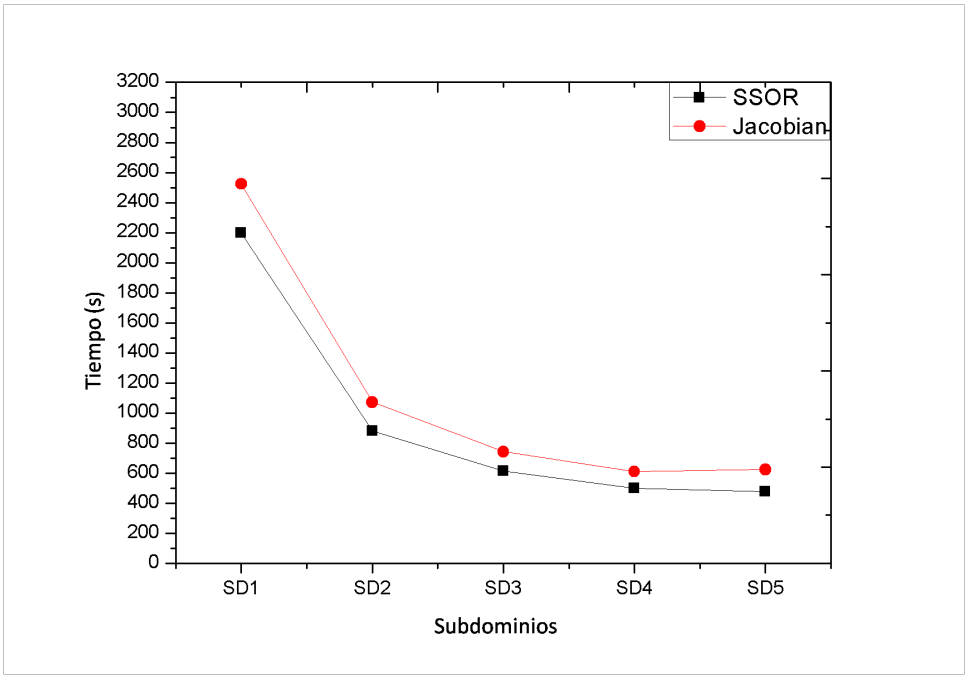


FIGURA 4.9: Tiempo de ejecución de WindNinja para un mapa de 1200x1200, considerando el precondicionador SSOR y Jacobian para diferentes subdominios

Subdominios	Tiempo (s) Cores			Speedup Cores		
	1	2	4	1	2	4
2	232	221	210	2,45	2,57	2,70
3	168	152	148	3,38	3,73	3,83
4	111	100	95	5,11	5,67	5,97
5	108	98	80	5,25	5,79	7,09

TABLA 4.4: Tiempo de ejecución de WindNinja para un mapa de 800x800, considerando el precondicionador SSOR para diferentes subdominios y cores

Subdominios	Tiempo (s) Cores			Speedup Cores		
	1	2	4	1	2	4
2	294	214	144	1,93	2,65	3,93
3	193	141	107	2,94	4,03	5,30
4	149	111	63	3,80	5,09	8,96
5	138	101	59	4,10	5,62	9,68

TABLA 4.5: Tiempo de ejecución de WindNinja para un mapa de 800x800, considerando el precondicionador Jacobian para diferentes subdominios y cores

Original (Tabla 2.1) a 63 segundos, que es un tiempo razonable para la operación real. En este caso, 4 nodos con 4 núcleos se utilizan (16) y del SpeedUp obtenido es de $8,96$.

La Figura 4.11 muestra el tiempo de ejecución WindNinja en un mapa 1200 x 1200 celdas teniendo en cuenta 3 subdominios y aplicando 1, 2, 4 y 8 cores para resolver el sistema con los precondicionadores SSOR y Jacobian precondicionadores. Se puede

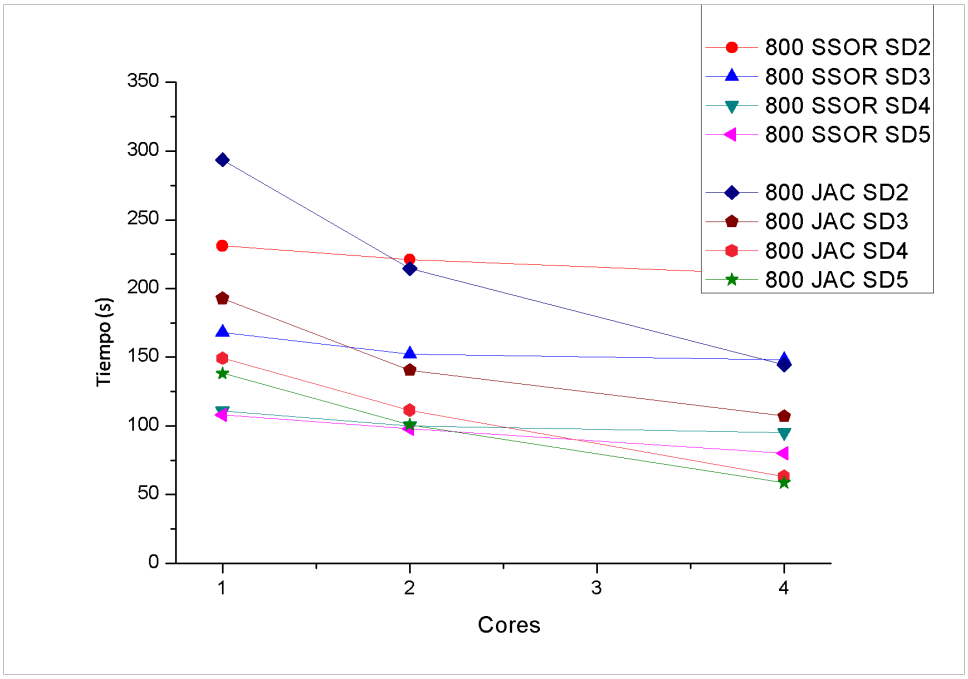


FIGURA 4.10: Tiempo de ejecución de WindNinja para un mapa de 800x800, considerando el precondicionador Jacobian y SSOR para diferentes subdominios y cores

observar que el precondicionador Jacobian escala mucho mejor que SSOR y proporciona un mejor tiempo de ejecución cuando se utilizan 2 y 4 cores. El uso de 8 cores no introduce ninguna mejora significativa en el tiempo de ejecución.

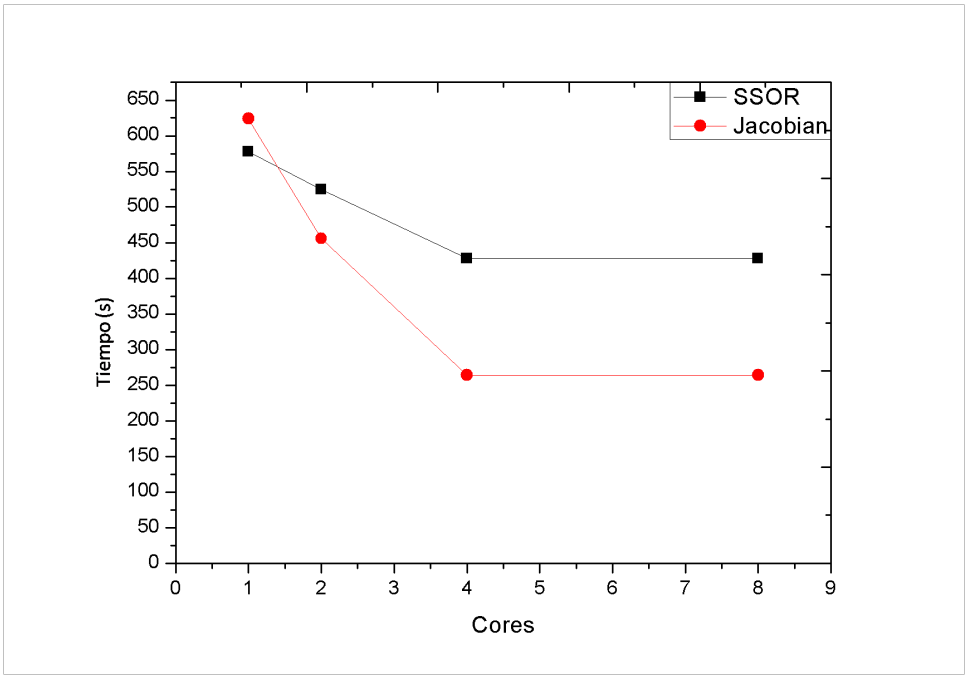


FIGURA 4.11: Tiempo de ejecución de WindNinja para un mapa de 800x800, considerando 3 subdominios para el precondicionador Jacobian y SSOR

4.3 Descomposición del dominio sin solapamiento: El método de Schur

El otro método de Descomposición del Dominio es el método sin solapamiento, o método de Schur. El objetivo consiste en la resolución de un sistema lineal: $Ax = b$, donde A es una matriz dispersa simétrica y definida positiva. El problema a resolver se divide en un conjunto de subdominios, más un subdominio frontera s. La frontera s se define de modo que para cualquier par de subdominios p_i y p_j, ningún nodo de p_i presenta ningún tipo de iteración con cualquier nodo de p_j , y sólo presentan interacción entre sus nodos y con los de la frontera s. En la Figura 4.12 se muestra un esquema de la descomposición de la malla en subdominios. Esto significa que, en el método de Schur, la malla se divide en un cierto número de subdominios y un subdominio frontera. Los nodos en un subdominio en particular no tienen ningún interrelación con los nodos en los otros subdominios pero si con el subdominio frontera. Por lo tanto, existen interrelaciones entre los nodos de un subdominio, y entre los nodos de otro subdominio y los nodos del subdominio frontera. Sin embargo, no hay interrelaciones entre los nodos de dos subdominios diferentes.

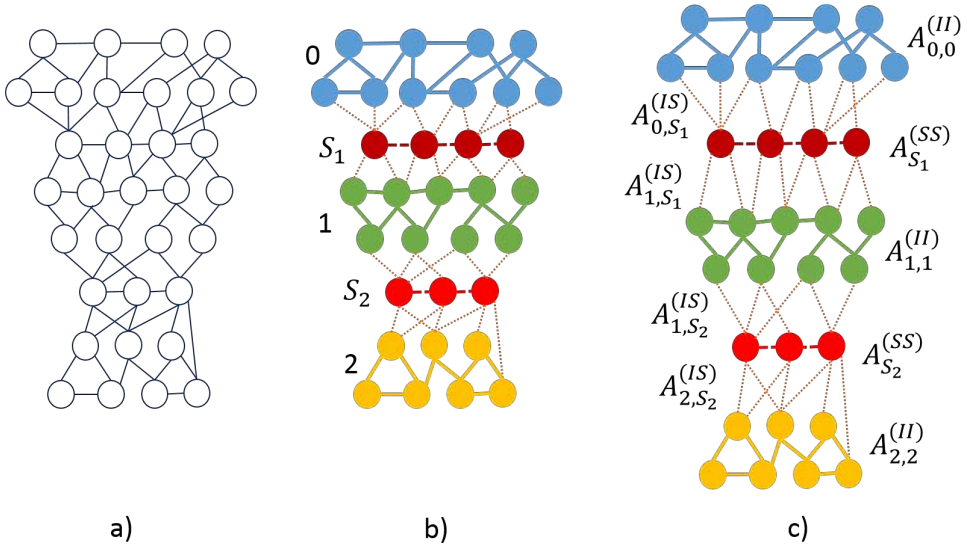


FIGURA 4.12: Descomposición de la malla aplicando el método de Schur

Esta organización de los subdominios implica que la matriz A pueda estructurarse de una forma análoga, tal como se muestra en la Figura 4.13.

Esta matriz puede reorganizarse, de modo que las submatrices cuadradas de interacción interna de las fronteras se sitúen en la parte inferior de la diagonal principal de la matriz, y las submatrices correspondientes de interacción entre un subdominio y la frontera, queden en las últimas filas y columnas de la matriz, como se muestra en la Figura 4.14.

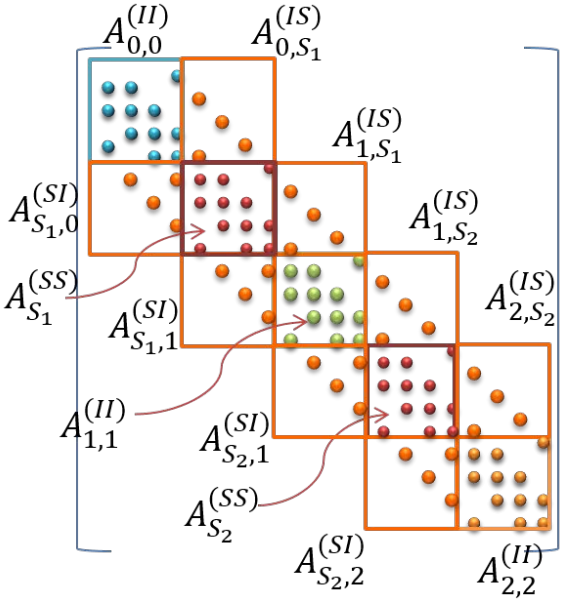


FIGURA 4.13: Organización de la matriz A aplicando el método de Schur

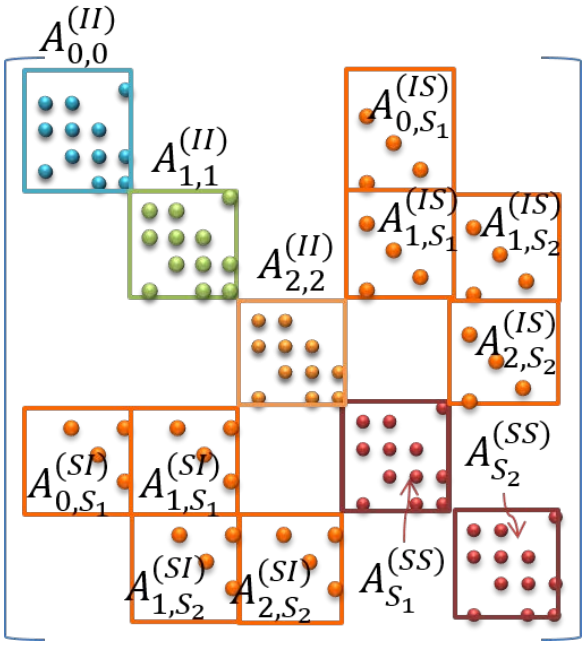


FIGURA 4.14: Reorganización de la matriz A

Ahora las submatrices de la frontera se organizan como una única submatriz de frontera y las submatrices de interrelación entre cada subdominio y cada una de las partes de la frontera también se agrupen en una única matriz rectangular. Esta organización se muestra en la Figura 4.15

Si tomamos en consideración la nueva organización de la matriz A que representa al sistema, se puede ver que dicha matriz se puede entender organizada como sigue:

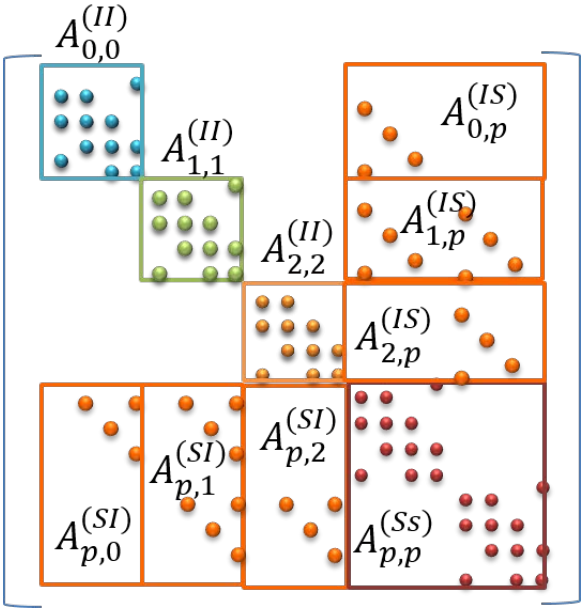


FIGURA 4.15: Matriz A resultante de considerar el método de Schur

1. Un conjunto submatrices cuadradas y dispersas $(A^{(II)}_{0,0} \dots A^{(II)}_{p-1,P-1})$. La dimensión de cada uno de estos submatrices depende del número de nodos en ese subdominio particular. Estas matrices se organizan a lo largo de la diagonal del sistema.

2. Una submatriz cuadrada y dispersa $(A^{(SS)}_{p,p})$ que representa las interrelaciones entre los nodos de la frontera. Esta submatriz está situada en la parte inferior de la diagonal del sistema.

3. Un conjunto de submatrices ubicadas en las últimas columnas $(A^{(IS)}_{0,s} \dots A^{(IS)}_{p-1,s})$ (o filas $(A^{(SI)}_{s,0} \dots A^{(SI)}_{s,p-1})$, debido a que la matriz es simétrica) del sistema, que representa las interrelaciones entre los nodos en un subdominio en particular y los nodos del subdominio frontera.

4. El resto de los elementos del sistema de la matriz son iguales a cero, ya que no existen interrelaciones entre los nodos de un subdominio en particular y los nodos de cualquier otro subdominio.

Teniendo en cuenta esta nueva organización, el sistema se puede expresar en términos de la matriz de bloques (Ecuación 4.2).

$$\begin{bmatrix} A^{(II)}_{0,0} & 0 & \cdots & A^{(IS)}_{0,p} \\ 0 & A^{(II)}_{1,1} & \cdots & A^{(IS)}_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ A^{(SI)}_{p,1} & A^{(SI)}_{p,2} & \cdots & A^{(SS)}_{p,p} \end{bmatrix} \begin{bmatrix} x^{(I)}_0 \\ x^{(I)}_1 \\ \vdots \\ x^{(S)}_p \end{bmatrix} = \begin{bmatrix} b^{(I)}_0 \\ b^{(I)}_1 \\ \vdots \\ b^{(S)}_p \end{bmatrix} \tag{4.2}$$

Aplicamos la eliminación de Gauss, de modo que la última fila del sistema puede reescribirse y convertirse en el siguiente:

$$\begin{bmatrix} A_{0,0}^{(II)} & 0 & \cdots & A_{0,p}^{(IS)} \\ 0 & A_{1,1}^{(II)} & \cdots & A_{1,p}^{(IS)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & S \end{bmatrix} \begin{bmatrix} x_0^{(I)} \\ x_1^{(I)} \\ \vdots \\ x_p^S \end{bmatrix} = \begin{bmatrix} b_0^{(I)} \\ b_1^{(I)} \\ \vdots \\ s_S \end{bmatrix} \tag{4.3}$$

donde S y s_S son, respectivamente:

$$S = A_{p,p}^{(SS)} - \sum_{i=0}^{P-1} A_{p,i}^{(SI)} A_{i,i}^{(II)-1} A_{i,p}^{(IS)} \tag{4.4}$$

$$s_S = b_p^{(S)} - \sum_{i=0}^{P-1} A_{p,i}^{(SI)} A_{i,i}^{(II)-1} b_i^{(I)} \tag{4.5}$$

Para resolver el sistema, el primer paso es resolver el subsistema:

$$S x_p^{(S)} = s_s \tag{4.6}$$

Sustituyendo, se puede expresar como:

$$(A_{p,p}^{(SS)} - \sum_{i=0}^{P-1} A_{p,i}^{(SI)} A_{i,i}^{(II)-1} A_{i,p}^{(IS)}) x_p^{(S)} = b_p^{(S)} - \sum_{i=0}^{P-1} A_{p,i}^{(SI)} A_{i,i}^{(II)-1} b_i^{(II)} \tag{4.7}$$

Este sistema es mucho más pequeño que el sistema original y se puede resolver mediante la aplicación del solver de gradiente conjugado con precondicionado.

Para resolver el sistema mediante PCG, es necesario evaluar $w = sp_k$. Esta multiplicación matriz-vector se puede expresar

$$w = A_{p,p}^{(SS)} p_k - \sum_{i=0}^{P-1} A_{p,i}^{(SI)} A_{i,i}^{(II)-1} A_{i,p}^{(IS)} p_k \tag{4.8}$$

Por lo tanto, esta operación se descompone en un sumatorio que se puede calcular en paralelo. Los vectores resultantes de cada multiplicación se pueden reagrupar, y el

resultado se utiliza en el siguiente paso del algoritmo PCG para obtener el nuevo valor de p_{k+1}. Este proceso se itera hasta que el sistema se resuelve.

Resolviendo el sistema, se obtiene $x_p^{(S)}$. Una vez obtenido $x_p^{(S)}$ todos los $x_i^{(I)}$ se obtiene el resultado del sistema (Ecuación 4.3).

$$x_i^{(I)} = A_{i,i}^{(II)-1}(b_i^{(I)} - A_{s,i}^{(IS)}x_p^{(S)}) \tag{4.9}$$

Se ha demostrado que la aplicación del método Schur descomposición dominio presenta varias ventajas:

1. Un sistema extremadamente grande con un gran número de variables se transforma en varios subsistemas más pequeños con un número más pequeño de variables que se pueden resolver mucho más rápido.

2. Para generar el primer subsistema por el método de eliminación de Gauss, es necesario evaluar el sumatorio de varios términos que incluyen operaciones con las matrices. Sin embargo, todos estos términos son independientes y se pueden calcular en paralelo.

3. Una vez que el primer subsistema se ha resuelto, todos los otros subsistemas pueden ser resueltos de forma independiente. Por lo tanto, esta es otra manera de explotar el paralelismo.

4.3.1 Paralelización del método de Schur

Como se describió anteriormente, el método Schur presenta varias ventajas y posibilidades de explotar el paralelismo. Por lo tanto, el método ha sido implementado como una aplicación MPI Master-Worker. La organización general se muestra en la Figura 4.16.

En primer lugar, el Master divide la malla y distribuye las submatrices correspondientes a cada subdominio a un Worker. Por tanto el Worker recibe la matriz que describe las interrelaciones internas de ese subdominio y las interrelaciones entre el subdominio y la frontera. Cada Worker evalúa uno de los términos del sumatorio de la Ecuación 4.4 y se calcula la matriz inversa correspondiente. Entonces, el Worker multiplica el resultado por el vector p_k y envía el vector resultante al Master. El Master lleva a cabo una de las iteraciones del algoritmo PCG y envía el nuevo p_{k+1} a los Workers. Los Workers evalúan la nueva multiplicación de la matriz-vector y se envían de vuelta el resultado para el Master. Este proceso se repite para resolver $x_p^{(S)}$. Una vez que se ha resuelto

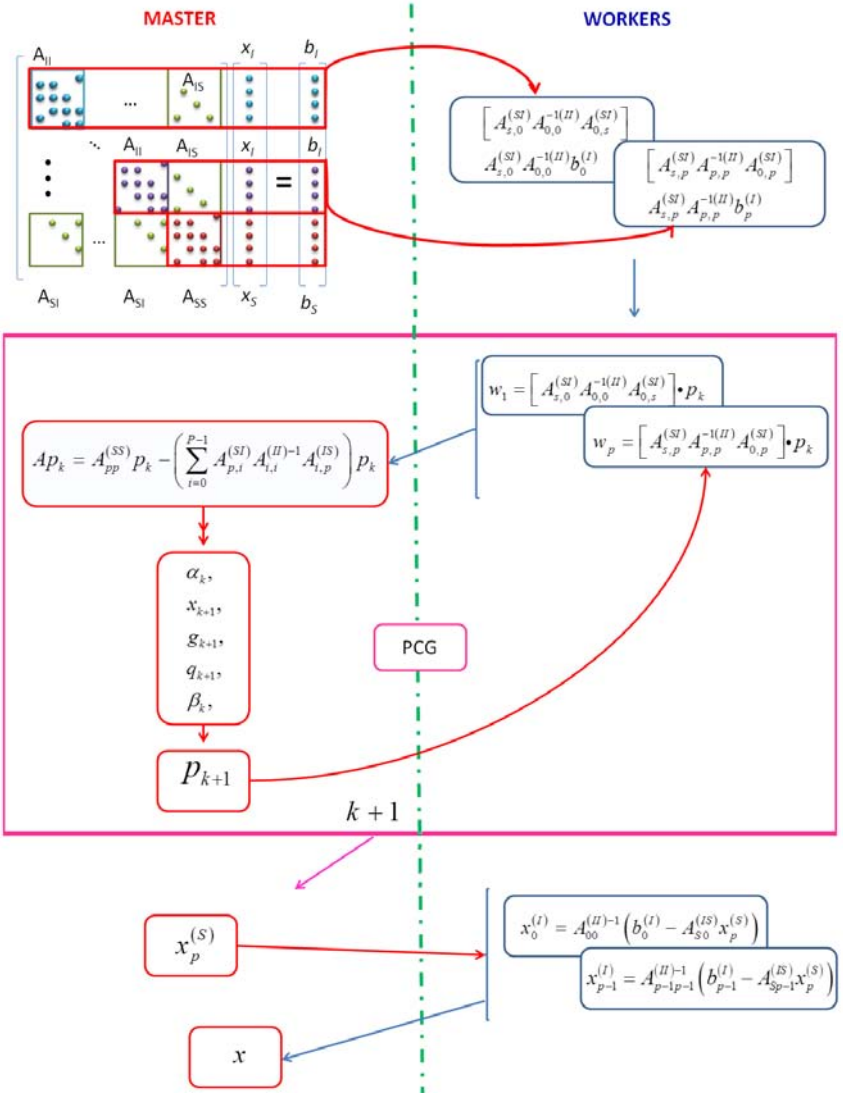


FIGURA 4.16: Paralelización del método Schur

el resultado se envía a los trabajadores que calculan los restantes $x_i^{(I)}$ términos de la solución.

En la siguiente sección, se describe la aplicación del método de Schur para calcular el campo de vientos y se muestran los resultados obtenidos.

4.3.2 Aplicación del método Schur para acelerar el cálculo del campo de vientos

Como se muestra en el capítulo 2, WindNinja crea la matriz que representa el sistema lineal que debe ser resuelto para calcular el campo de viento. El enfoque es aplicar el método de descomposición de dominio de Schur. Como se ha visto, este método

introduce un paralelismo inherente que puede ser explotado para reducir el tiempo de ejecución y acelerar el cálculo del campo de viento.

La primera etapa del método de Schur consiste en dividir la malla en subdominios disjuntos y el subdominio frontera. El objetivo es dividir la malla entre un cierto número de subdominios de tamaño similar y con una frontera también de tamaño similar. La partición de la malla entre muchos subdominios, por lo general, genera un subdominio frontera demasiado grande que toma demasiado tiempo para ser resuelto. Una vez que la malla ha sido dividido, es necesario cambiar el orden de los bloques de matriz para aplicar la eliminación de Gauss. Un estudio en profundidad de las matrices WindNinja a determinado que las matrices WindNinja siempre presentan el mismo patrón que se diferencian unas de las otras por las posiciones de las iteraciones. Por lo tanto, se ha propuesto un método de partición de la matriz basado en el patrón de la matriz WindNinja. La Figura 4.17 muestra el patrón general de las matrices WindNinja. La mayor parte de los elementos, que representan las interacciones entre los nodos de la malla, se concentran alrededor de la diagonal principal y sólo unos pocos elementos se encuentran alejadas de esta. Cada nodo presenta un máximo 13 interrelaciones con otros nodos en el mismo subdominio o en el subdominio frontera.

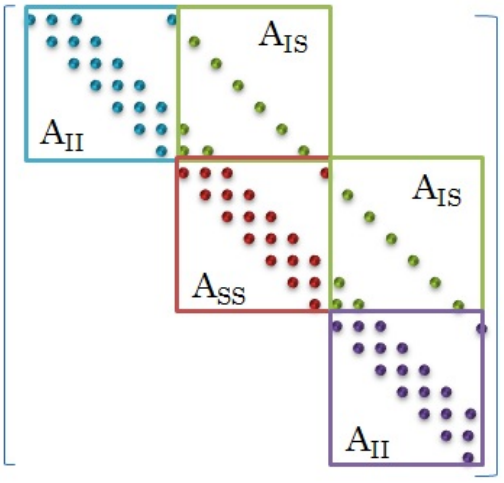


FIGURA 4.17: Patrón de la matriz WindNinja

Por lo tanto, las matrices se han dividido y reordenado simplemente teniendo en cuenta este patrón de forma automática.

Una vez que se divide la matriz, es necesario evaluar los términos de las sumas de las Ecuaciones 4.4 y 4.5. Estos términos se pueden calcular en paralelo en diferentes nodos. Sin embargo, el cálculo de cada uno de estos términos requiere la inversión de la matriz, la multiplicación de matrices y multiplicación de la matriz-vector, que son operaciones intensivas complejos y costosas computacionalmente. En particular, la inversión de la

matriz es una operación compleja y se ha utilizado el paquete Pardiso [58] para invertir las matrices.

S y S_s se calculan y es posible resolver $x_p^{(S)}$ mediante la aplicación del PCG. Posteriormente, se puede calcular cada $x_i^{(I)}$.

La Tabla 4.6 muestra el tiempo de ejecución y la memoria necesaria para resolver el campo de viento para mapas desde 200x200 celdas hasta 800x800 celdas. $Memoria$ es la memoria necesaria para construir la matriz, $NSd.$ es el número de subdominios, $NA_{i,i}$ es el número de elementos de la diagonal de la matriz $A_{i,i}^{(II)}$, $NA_{p,p}$ es el número de elementos de la diagonal de la matriz $A_{p,p}^{(SS)}$, T_p el tiempo necesario para construir las matrices, T_i es el tiempo de inversión de la matriz y T_t es el tiempo total para calcular el campo de vientos.

Map size (cells)	Memoria (MB)	NSd.	$NA_{i,i}$	$NA_{p,p}$	T_p (s)	T_i (s)	T_t (s)	Sp
200x200	194	2	284999	30000	1,00	80,56	305,56	0,09
200x200	194	3	179999	60000	1,00	32,00	232,00	0,11
200x200	194	4	127499	90000	1,00	10,20	185,20	0,14
200x200	194	5	96000	119999	1,00	4,50	154,50	0,17
200x200	194	6	75000	149999	1,00	2,08	127,08	0,21
200x200	194	7	60000	179999	1,00	0,71	100,71	0,26
200x200	194	8	48750	209999	1,00	0,33	75,33	0,35
200x200	194	9	40000	239999	1,00	0,25	50,25	0,52
200x200	194	10	33000	269999	1,00	0,24	40,12	0,66
200x200	194	11	27273	299999	10,21	0,21	47,12	0,56
200x200	194	12	22500	329998	15,77	0,22	55,12	0,48
300x300	353	2	379999	40000	1,00	119,56	389,56	0,14
300x300	353	3	239999	80000	1,00	48,10	288,10	0,20
300x300	353	4	169999	120000	1,00	10,20	220,20	0,26
300x300	353	5	128000	159999	1,00	4,50	184,50	0,31
300x300	353	6	100000	199999	1,00	2,08	152,08	0,37
300x300	353	7	80000	239999	1,00	0,71	120,71	0,47
300x300	353	8	65000	279999	1,00	0,33	90,33	0,62
300x300	353	9	53333	319999	1,00	0,25	60,25	0,93
300x300	353	10	44000	359999	1,00	0,24	40,12	1,40
300x300	353	11	36364	399999	16,69	0,25	48,12	1,17
300x300	353	12	30000	439998	21,23	0,24	54,12	1,04
400x400	793	2	854999	90000	1,00	409,36	724,36	0,14
400x400	793	3	539999	180000	1,00	173,26	453,26	0,22

Sigue en la página siguiente

Map size (cells)	Memoria (MB)	NSd.	$NA_{i,i}$	$NA_{p,p}$	T_p (s)	T_i (s)	T_t (s)	Sp
400x400	793	4	382499	270000	1,00	35,92	280,92	0,35
400x400	793	5	288000	359999	1,00	15,53	225,53	0,44
400x400	793	6	225000	449999	1,00	6,99	181,99	0,54
400x400	793	7	180000	539999	1,00	5,20	145,20	0,68
400x400	793	8	146250	629999	1,00	2,38	107,38	0,92
400x400	793	9	120000	719999	1,00	1,09	71,09	1,38
400x400	793	10	99000	809999	1,00	0,77	40,77	2,41
400x400	793	11	81818	899999	18,56	0,54	49,77	1,97
400x400	793	12	67500	989998	23,89	0,57	65,77	1,47
600x600	1562	2	1519999	160000	1,00	1859,71	2219,71	0,10
600x600	1562	3	959999	320000	1,00	492,29	812,29	0,27
600x600	1562	4	679999	480000	1,00	128,22	408,22	0,53
600x600	1562	5	512000	639999	1,00	49,52	289,52	0,75
600x600	1562	6	400000	799999	1,00	18,46	218,46	1,00
600x600	1562	7	320000	959999	1,00	12,01	172,01	1,27
600x600	1562	8	260000	1119999	1,00	4,98	124,98	1,75
600x600	1562	9	213333	1279999	1,00	2,07	82,07	2,66
600x600	1562	10	176000	1439999	1,00	1,61	61,20	3,57
600x600	1562	11	145454	1599999	27,00	0,98	86,20	2,53
600x600	1562	12	120000	1759998	35,00	–	–	–
700x700	2620	2	2374999	250000	1,00	3730,00	4135,00	0,08
700x700	2620	3	1499999	500000	1,00	1770,00	2130,00	0,15
700x700	2620	4	1062499	750000	1,00	210,00	525,00	0,62
700x700	2620	5	800000	999999	1,00	104,00	374,00	0,87
700x700	2620	6	625000	1249999	1,00	45,34	270,34	1,20
700x700	2620	7	500000	1499999	1,00	11,58	191,58	1,70
700x700	2620	8	406250	1749999	1,00	5,36	140,36	2,32
700x700	2620	9	333333	1999999	1,00	4,25	94,25	3,45
700x700	2620	10	275000	2249999	1,00	3,58	64,00	5,08
700x700	2620	11	227273	2499999	39,00	2,53	–	–
700x700	2620	12	187500	2749998	46,00	2,45	–	–
800x800	5556	2	6079999	640000	1,00	–	–	–
800x800	5556	3	3839999	1280000	1,00	–	–	–
800x800	5556	4	2719999	1920000	1,00	2111,00	2461,00	0,20
800x800	5556	5	2048000	2559999	1,00	1148,00	1448,00	0,34
800x800	5556	6	1600000	3199999	1,00	450,89	700,89	0,71

Sigue en la página siguiente

Map size (cells)	Memoria (MB)	NSd.	$NA_{i,i}$	$NA_{p,p}$	T_p (s)	T_i (s)	T_t (s)	Sp
800x800	5556	7	1280000	3839999	1,00	110,26	310,26	1,60
800x800	5556	8	1040000	4479999	1,00	50,22	200,22	2,48
800x800	5556	9	853333	5119999	1,00	40,25	140,25	3,54
800x800	5556	10	704000	5759999	1,00	13,82	90,20	5,50
800x800	5556	11	581818	6399999	52,45	12,21	–	–
800x800	5556	12	480000	7039998	86,23	9,32	–	–

TABLA 4.6: Tiempo de ejecución de WindNinja aplicando el método de Schur con diferentes número de subdominios

La Figura 4.18 muestra el tiempo de ejecución al incrementar el número subdominios y la Figura 4.19 muestra el aumento de Speedup obtenido. Se puede observar que para número de subdominios pequeños y mapas grandes, el tiempo necesario para llevar a cabo la inversión de la matriz representa la parte más significativa del tiempo total de ejecución. En realidad, para un mapa de las $700x700$ celdas y una descomposición en tan solo 2 subdominios el tiempo para invertir la matriz es más de 3700 segundos y el tiempo de ejecución total es 4090 segundos. Este tiempo de ejecución es mucho mayor que el tiempo secuencial ejecución de WindNinja para el mismo mapa, que es de alrededor de 325 segundos. Para el mismo mapa, utilizando una descomposición en 10 subdominios, la inversión de la matriz toma $3,58$ segundos y el tiempo de ejecución total es de 64 segundos. Esto representa un aumento de velocidad de $4,56$ en comparación con la versión secuencial WindNinja. Esto significa que hasta 800x800 celdas, el tiempo de ejecución se ha reducido por debajo de los 100 segundos límite.

Sin embargo, aumentar el número de subdominios de mapas más grande que 600x600 celdas a 11 o 12 subdominios, provoca que la matriz de frontera se vuelve demasiado grande y el sistema no se puede resolver, como se indica en la tabla 4.6.

Los resultados obtenidos al aplicar este método de Schur para diferente tamaños de mapa considerando 10 subdominios se muestran en la Tabla 4.7 y en la Figura 4.20. En realidad, el tiempo total de ejecución está por debajo del límite de 100 segundos impuesta por las condiciones de funcionamiento para los mapas hasta $800x800$ celdas.

Sin embargo, para mapas de más de 800x800 celdas el método no es aplicable en los nodos disponibles ya que aparecen limitaciones debidas a la memoria necesaria para almacenar las matrices necesarias.

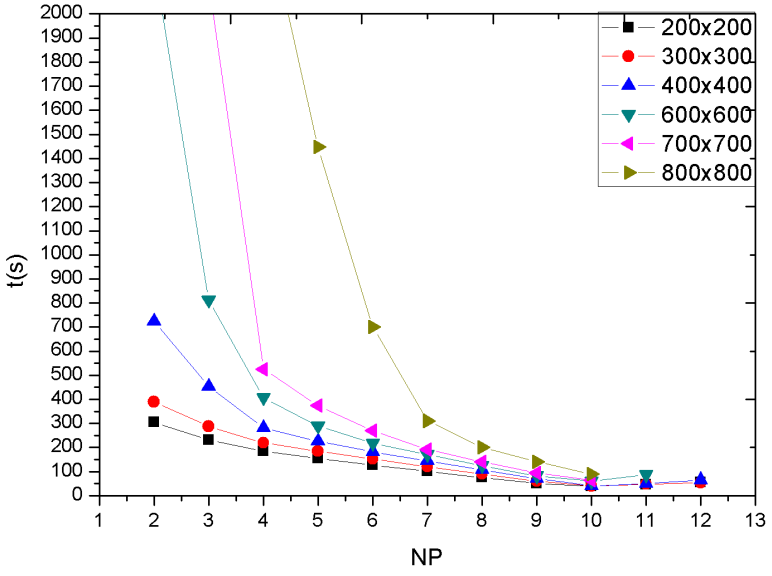


FIGURA 4.18: Tiempo de ejecución aplicando el método Schur.

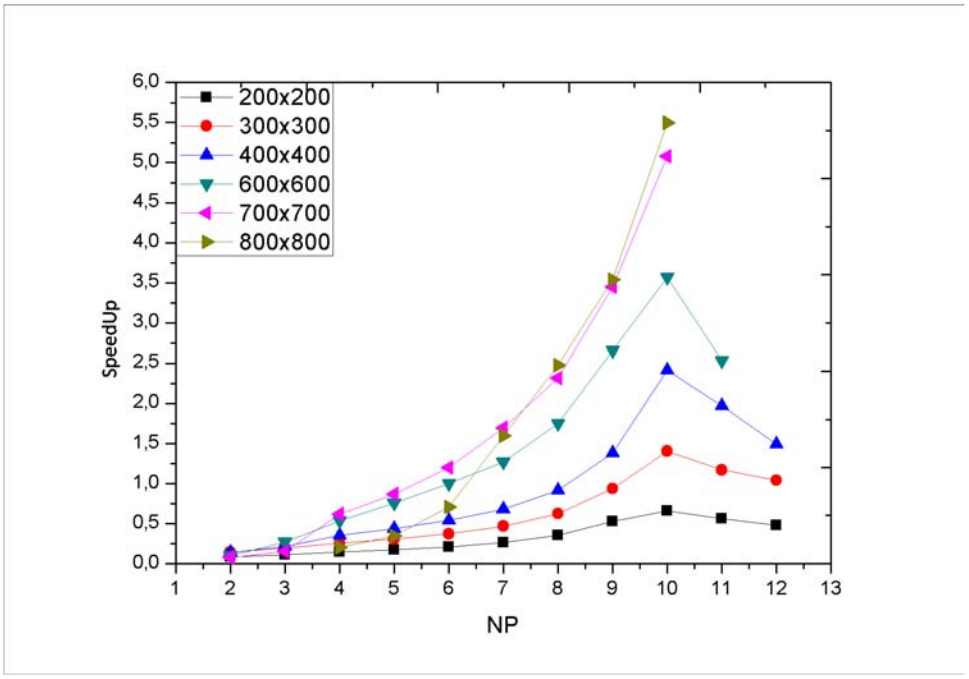


FIGURA 4.19: SpeedUp aplicando el método Schur.

4.4 Conclusiones del capítulo

Los métodos de descomposición del dominio con solapamiento (método de Schwarz) o sin solapamiento (método de Schur) se han mostrado como otro método para introducir paralelismo en el cálculo del campo de vientos, y, de esta manera, reducir el tiempo de ejecución. De todos modos, tanto un método como otro, presentan limitaciones,

Map size (cells)	Memory used (MB)	NSub	$NA_{i,i}$	$NA_{p,p}$	T_p (s)	T_i (s)	T_t (s)
200x200	194	10	33000	269999	1,00	0,24	40,12
300x300	353	10	44000	359999	1,00	0,24	40,12
400x400	793	10	99000	809999	1,00	0,77	40,77
500x500	1212	10	140720	1407200	1,00	1,40	51,40
600x600	1562	10	176000	1439999	1,00	1,61	61,20
700x700	2620	10	275000	2249999	1,00	3,58	64,00
800x800	5556	10	704000	5759999	1,00	13,82	90,20

Tabla 4.7: Tiempo de ejecución considerando diferentes tamaños de mapa y 10 subdominios

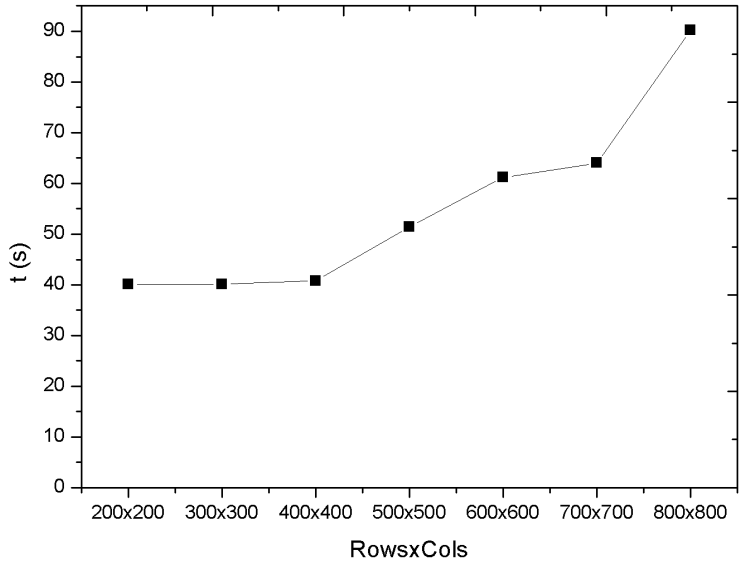


Figura 4.20: Tiempo de ejecución considerando diferentes tamaños de mapa y 10 subdominios

principalmente debidas a que el grado de paralelismo va asociado al número de subdominios definido, pero al aumentar el número de subdominios, crece el tamaño de la frontera entre los subdominios y esto introduce una limitación significativa.

Capítulo 5

Aceleración del Gradiente Conjugado con Precondicionador (PCG)

5.1 Introducción

Tal como se ha descrito en el capítulo 2, las operaciones que involucran más tiempo en la ejecución del solver del gradiente conjugado con precondicionador (PCG), son las operaciones de multiplicación matriz dispersa-vector, conocida comúnmente como SpMV. El objetivo de este capítulo es poder reducir el tiempo de ejecución de esta operación, y para ello se ha realizado un estudio de los diferentes sistema de almacenamiento y de su repercusión en los métodos de multiplicación matriz dispersa-vector.

5.2 Formato de almacenamiento de matrices dispersas y multiplicación matriz dispersa-vector

En el capítulo 2 se ha descrito el formato de almacenamiento de matrices dispersas utilizado en WindNinja (CSR) y se ha mostrado el efecto negativo de este formato para la realización de la multiplicación de matriz dispersa-vector con matrices de tan baja densidad como son las de WindNinja, donde la densidad está por debajo de $0,02$.

Un formato alternativo de almacenar la matriz dispersa que puede ser utilizado es el Compressed Diagonal Storage (CDS) [59]. Este formato de almacenamiento es mayoritariamente utilizada para elementos finitos. En este tipo de almacenamiento, las subdiagonales de

la matriz son almacenadas como filas de una matriz y un elemento adicional indica la posición relativa de cada subdiagonal respecto de la diagonal principal (Figura 2.2). Este formato se utiliza cuando los elementos distintos de cero se encuentran organizados en subdiagonales, como es el caso de WindNinja. Sin embargo, este método CDS está pensado para sistemas con subdiagonales muy cercanas a la diagonal principal, y WindNinja no presenta esta característica, ya que las subdiagonales de la matriz de WindNinja se encuentran lejos de la diagonal principal y requeriría almacenar un gran número de ceros. Por tanto, este sistema no es apto para representar las matrices generadas por WindNinja, debido a que incrementaría los requerimientos de memoria.

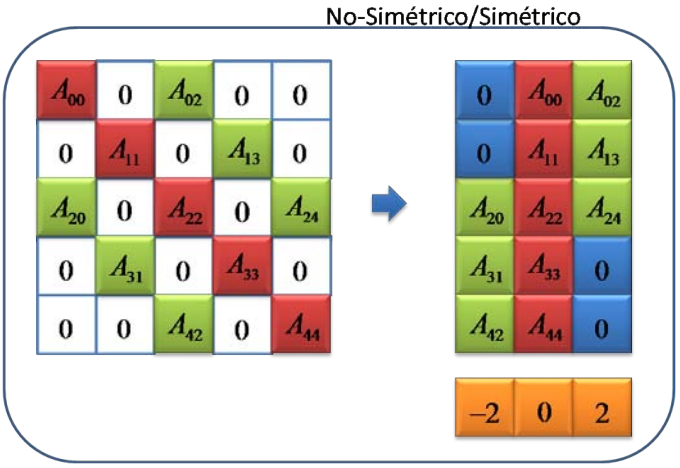


FIGURA 5.1: Compressed Diagonal Storage CDS

5.3 Formato de almacenamiento VDSpM: Vectorization of Diagonal Sparse Matrix y paralelizacón de la multiplicación matriz dispersa-vector

Para hacer frente a este problema y evitar la asignación de memoria innecesaria, se presenta un nuevo tipo de almacenamiento llamado Vectorization of Diagonal Sparse Matrix (VDSpM) [60]. En este formato las subdiagonales se almacenan en un conjunto de vectores (unidimensional arrays) del tamaño preciso para incluir los elementos de la subdiagonal y un índice que indica la posición de la subdiagonal en la matriz. De esta manera, los elementos de cada subdiagonal son almacenado en posiciones consecutivas de memoria. Este formato de almacenamiento se muestra en la Figura 5.2.

A la matriz WindNinja, almacenada en CRS, se le debe realizar un preanálisis para transformarla al formato VDSpM. En este análisis es necesario determinar las posiciones de cada subdiagonal y, a partir de la posición, determinar la cantidad de elementos de cada subdiagonal. Si una matriz tiene n filas y n columnas (nxn), la diagonal principal

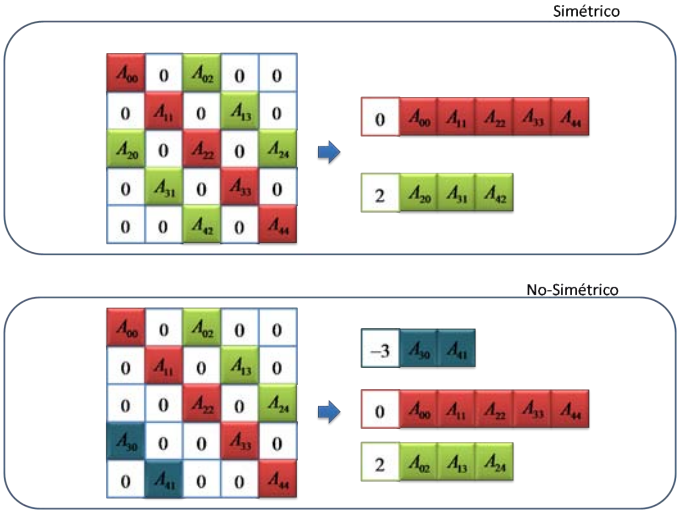


FIGURA 5.2: Vectorization of Diagonal Sparse Matrix (VDSpM)

tiene n elementos y la subdiagonal en la posición i tiene $n - i$ elementos. De esta manera es posible determinar el tamaño del vector para almacenar cada subdiagonal. Se debe considerar que la matriz resultante es simétrica. En este caso, el número de subdiagonales a almacenar corresponde únicamente a la submatriz triangular superior (o inferior).

El análisis preliminar de la vectorización de la matriz dispersa y la construcción de la VDSpM matriz implican un cierto tiempo. Después de realizar diferentes medidas con diferentes matrices de tamaños diferentes, los resultados muestran que el tiempo requerido para llevar a cabo estos pasos depende exponencialmente del tamaño de la matriz. Los resultados se muestran en la Tabla 5.1. En esta tabla, *Mapa* representa el número de filas y columnas del mapa original de Windninja, *NE* es el número de elementos diferentes de cero en la matriz, *Densidad* es el porcentaje de elementos diferentes de cero, *Memoria* es la memoria requerida para el almacenaje en CRS y t es el tiempo requerido para construir la matriz en formato VDSpM.

Map Size	NE	Density (%)	Memory (MB)	t (s)
400x400	10770511	0,0017	353	0,1
500x500	24285711	0,00075	797	0,1
600x600	43220911	0,00042	1383	0,2
700x700	67576111	0,00027	2166	0,4
800x800	173161711	0,00011	5556	1,1
900x900	270652111	0,00009	7034	3,2
1000x1000	270652111	0,00007	8657	3,8

TABLA 5.1: Tiempo de construcción de la matriz en VDSpM

Los datos de la Tabla 5.1 pueden ser representados en una gráfica, como se muestra en la Figura 5.3. A partir de estos datos, es posible determinar el tiempo requerido para construir la matriz VDSpM. Este tiempo puede ser expresado como se muestra en la Ecuación 5.1.

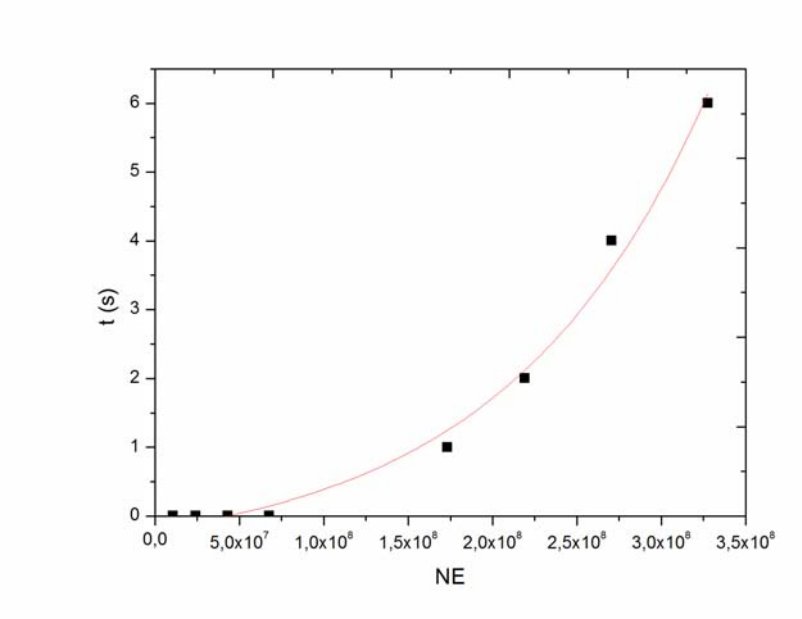


FIGURA 5.3: Tiempo de construcción de la matriz en VDSpM

$$t = 0,44 \cdot e^{\frac{-NE}{1,21 \cdot 10^8}} - 0,64 \tag{5.1}$$

Usando Vectorized Diagonal Sparse Matrix (VDSpM), es posible implementar la multiplicación matriz dispersa-vector. La principal ventaja de este formato de almacenamiento es que los subdiagonales de la matriz se almacenan como vectores secuenciales. Cada término de la subdiagonal de la matriz debe multiplicarse por los términos correspondientes del vector de multiplicación y los resultados parciales se debe agregar para crear los términos del vector resultante. Este método de multiplicación de la matriz dispersa-vector está representado en la Figura5.4. En esta figura, la matriz se ha considerado simétrica, como es el caso de las matrices de WindNinja.

El esquema que se muestra en la Figura 5.4, se ha desglosado en varias etapas que muestran la multiplicación de cada vector subdiagonal por el vector de multiplicación. Estas fases se muestran en la Figura 5.5. Cada una de estas fases se puede ejecutar al mismo tiempo y, así, el paralelismo interno también puede ser explotado.

Otra forma de representarlo, que tal vez pueda resultar también más aclaradora sea la que se muestra en la Figura 5.6, donde puede verse que los accesos a memoria son completamente lineales.

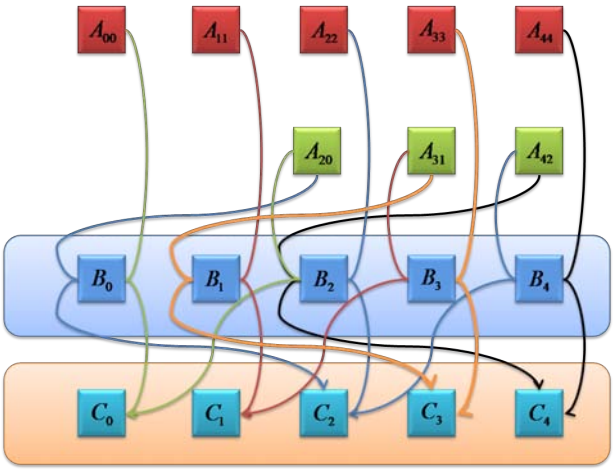


FIGURA 5.4: Multiplicación Matriz-Vector considerando el formato de almacenamiento VDSpM

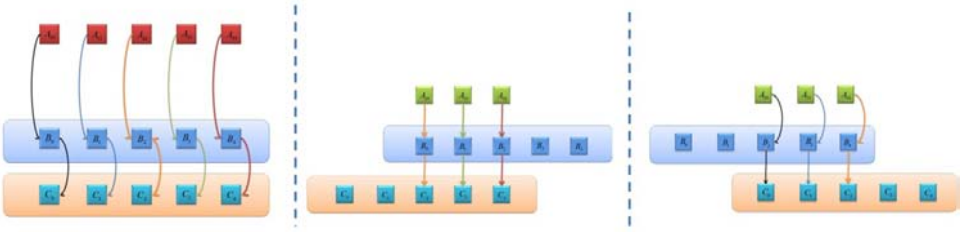


FIGURA 5.5: Multiplicación Matriz-Vector considerando el formato de almacenamiento VDSpM paso por paso

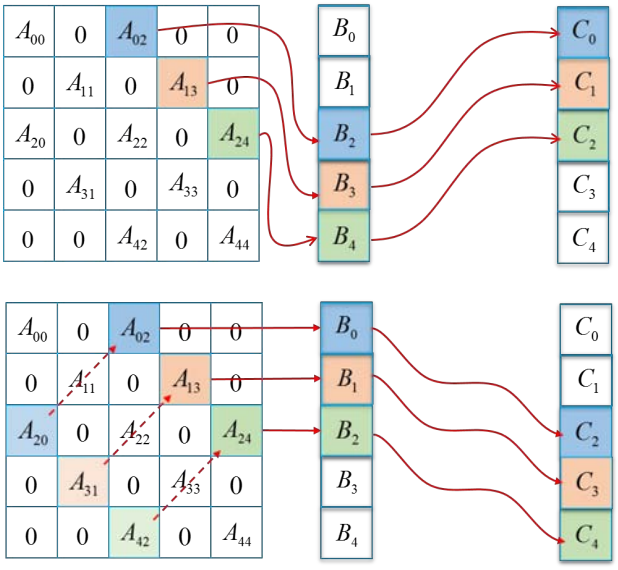


FIGURA 5.6: Multiplicación Matriz-Vector considerando el formato de almacenamiento VDSpM

De este modo, utilizando este formato de almacenamiento es posible realizar la multiplicación en paralelo y minimizando los fallos de cache. Esta multiplicación ha sido implementada en OpenMP y MPI. En los siguientes apartados se muestran las implementaciones

llevadas a cabo y los resultados obtenidos [43]

5.3.1 Paralelización OpenMP

El método de multiplicaición propuesto ha sido paralelizados usando OpenMP. Cada thread calcula el termino correspondiente a algunos elementos de la diagonal principal y subdiagonales. Los elementos de las subdiagonales son distribuidos en threads, mientras que el vector resultante se mantiene en la memoria compartida, y por tanto, puede ser accedida por todos los threads. Esta implementación se muestra en la Figura 5.7.

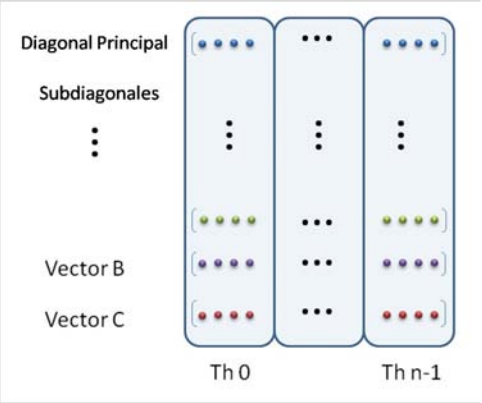


FIGURA 5.7: Paralelización OpenMp de VDSpMV

Los resultados para diferentes tamaños de matriz y diferentes número de cores se muestra en la Tabla 5.3. En esta tabla, se incluyen el tiempo de ejecución y el SpeedUp comparado con el tiempo de ejecución de WindNinja original (WN). El tiempo de ejecución y SpeedUp se muestran en la Figura 5.8 y en la Figura 5.9, respectivamente. Se puede observar que el SpeedUp con 2 cores alcanza $1,7$, con 4 cores alcanza $2,4$ y con 8 y 16 cores alcanza $2,5$. Pero, el punto más significativo es que cuando se ejecuta un mapa de 1000x1000 celdas, con un tiempo 1831 segundo en un core, se ejecuta en sólo 852 segundos utilizando 8 cores, y el tiempo de ejecución del mapa de 1500x1500 es reducido de 3831 segundos a 1824 segundos. Esto implica una significativa mejoría, pero los resultados obtenidos en el tiempo de ejecución no son suficientemente bajos para ser usado en operaciones a tiempo real.

5.3.2 Paralelización MPI

Este sistema de almacenamiento permite utilizar MPI. Para este proceso, han sido considerados dos casos diferentes. En el primero los Workers envían parte del vector resultante al Master. El SpeedUp obtenido no es bueno y el tiempo de ejecución se incrementa. Sin embargo, el mayor tiempo consumido es como consecuencia de formar

Mapa	Tiempo (s) NCores					SpeedUp NCores				
	WN	1	2	4	8	16	1	2	4	8	16
400x400	236	215	191	135	141	143	1,10	1,24	1,76	1,68	1,66
600x600	567	505	325	231	225	227	1,12	1,74	2,45	2,52	2,50
800x800	804	769	608	507	465	478	1,05	1,32	1,59	1,73	1,68
1000x1000	1831	1658	1300	967	852	863	1,10	1,41	1,89	2,15	2,12
1200x1200	2939	2714	1871	1300	1260	1290	1,08	1,57	2,26	2,33	2,28
1500x1500	3831	3080	2488	2006	1824	1856	1,24	1,54	1,91	2,10	2,06

TABLA 5.2: Tiempo de ejecución y SpeedUp considerando el formato de almacenamiento VDSpM y paralelización OpenMP

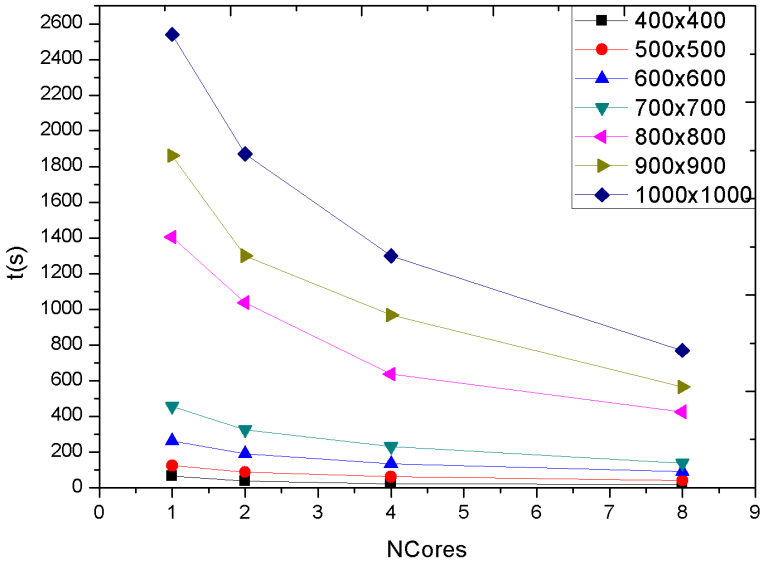


FIGURA 5.8: Tiempo de ejecución de WindNinja considerando el almacenamiento VDSpM y la paralelización OpenMP

un único vector completo. El segundo experimento mide el tiempo requerido para calcular cada parte del vector de resultado, pero sin considerar la agregación. El SpeedUp obtenido supera considerablemente la velocidad obtenida por los otros enfoques. Este enfoque es interesante porque muchos métodos iterativos no requieren la agregación del vector resultado para proceder a la siguiente iteración, como sucede en el caso del gradiente conjugado con precondicionador de WindNinja .

Este método puede aplicarse al PCG de WindNinja en MPI. En este caso, la estructura Master/Worker ha sido implementada, de modo que cada Worker recibe una porción de la matriz y del vector. Para ser más precisos, se corresponde a una partición de cada subdiagonal. Esta implementación se muestra en la Figura 5.10.

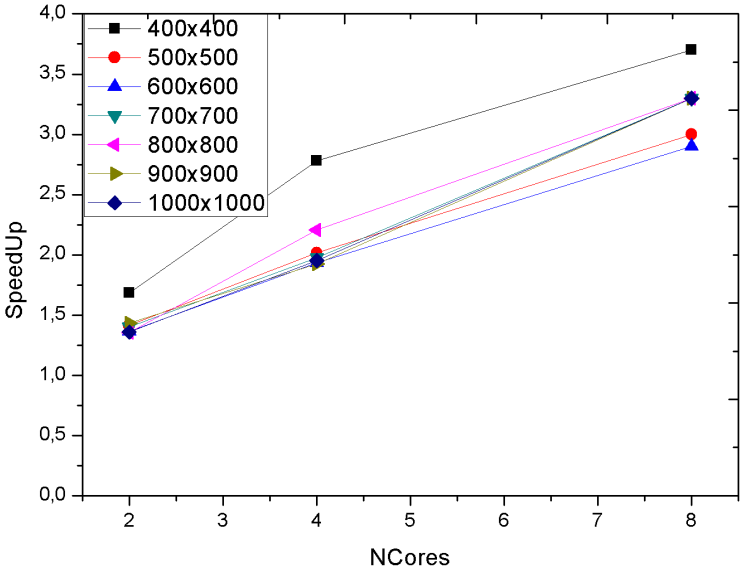


FIGURA 5.9: SpeedUp considerando el formato de almacenamiento VDSpM y la paralelización OpenMP

NumFilas	NE	Densidad (%)	Memoria (MB)	Con agregación Num Workers 2	4	8	12	Sin agregación Num Workers 2	4	8	12
1800000	24285711	0.00075	797	0,91	1,12	1,15	0,78	3,05	4,24	7,41	10,79
3200000	43220911	0.00042	1383	0,87	1,34	1,06	0,90	3,03	5,17	6,02	7,00
5000000	67576111	0,00027	2166	0,92	1,16	1,20	1,28	3,25	5,12	8,85	10,44
12800000	173161711	0,00011	5556	0,86	1,09	1,23	1,20	2,93	4,17	5,45	7,39
16200000	270652111	0,00009	7034	2,88	0,78	1,05	1,17	0,77	4,04	6,44	9,19
20000000	270652111	0,00007	8657	0,70	1,06	1,60	1,16	2,76	5,82	7,72	6,12
24200000	327527311	0,00006	10513	0,85	1,16	1,27	1,09	2,95	4,16	5,47	5,65

TABLA 5.3: Speedup obtenido con la paralelización MPI

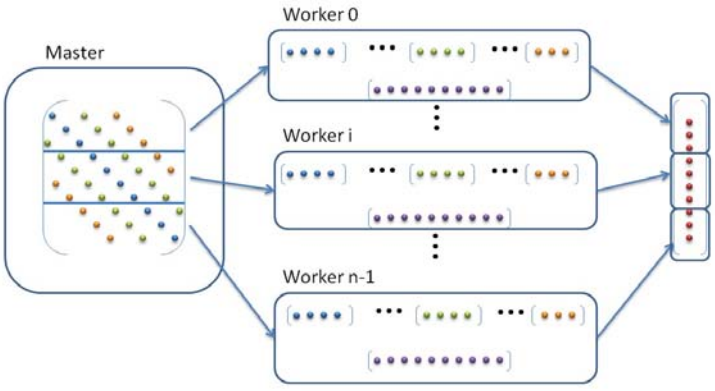


FIGURA 5.10: Paralelización MPI de VDSpMV

El tiempo de ejecución y el SpeedUp se muestran en la Tabla 5.4. Los resultados obtenidos no son tan buenos como los obtenidos con la paralelización OpenMP. Con 2 Workers el SpeedUp obtenido está alrededor de $1,4$, con 4 workers esta alrededor $1,8$ y con 8 procesadores está alrededor 2.0. Estos resultados se presentan en las Figuras 5.11 y 5.12.

NRowsxNCols	Memory (MB)	VDSpMV time Num Workers 1	2	4	8	VDSpMV SpeedUp Num Workers 2	4	8
400x400	353	64	40	30	21	1,6	2,1	3,0
500x500	797	123	81	45	37	1,5	2,7	3,3
600x600	1383	261	187	142	131	1,4	1,8	2,0
700x700	2166	456	320	248	210	1,4	1,8	2,2
800x800	5556	1405	1027	806	740	1,4	1,7	1,9
900x900	7034	1862	1382	1082	910	1,3	1,7	2,0
1000x1000	8687	2540	1888	1463	1338	1,3	1,7	1,9

TABLA 5.4: Tiempo de ejecución y SpeedUp de la paralelización MPI

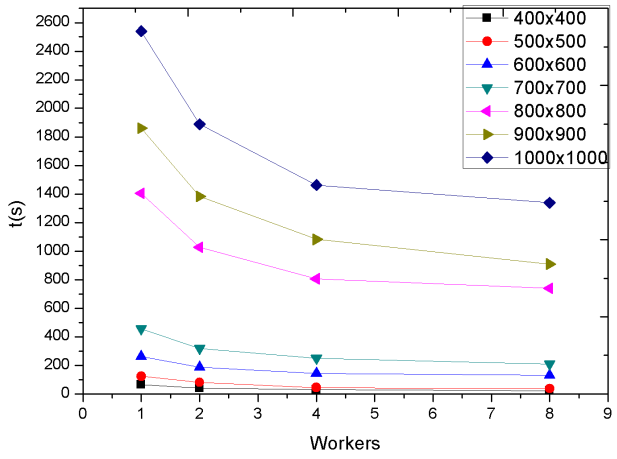


FIGURA 5.11: Tiempo de ejecución de WindNinja con paralelización MPI

5.4 Efecto del precondicionador

En esta sección se pretende determinar como afecta la selección del precondicionador en el rendimiento del solver, y se van comparar y analizar la implementación del PCG con SSOR y Jacobian. La multiplicación de matriz dispersa-vector (SpMV) desarrollada utilizando el formato de almacenamiento VDSpM ha sido implementada en ambos precondcionadores. Las Tablas 5.5 y 5.6 muestran el tiempo de ejecución que se obtiene al utilizar el nuevo formato de almacenamiento VDSpM para almacenar la matriz A. En

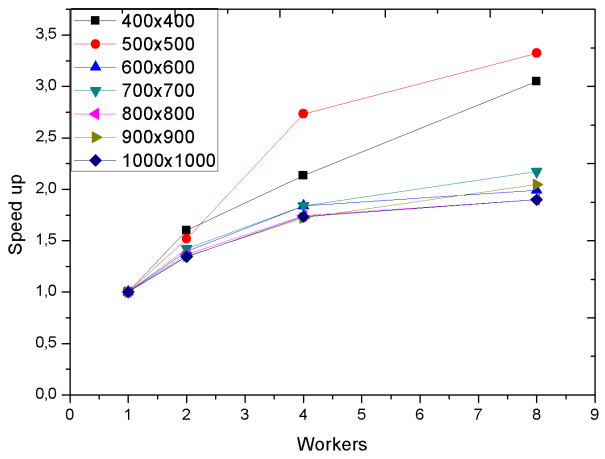


FIGURA 5.12: SpeedUp de WindNinja con paralelización MPI

ambos casos, el precondicionador M se almacena utilizando el formato de almacenamiento CRS.

Mapa	Tiempo (Segundos) Ncores			
	1	2	4	8
400x400	87	88	87	87
500x500	140	135	132	133
600x600	317	299	237	240
800x800	477	405	327	330
1000x1000	1391	1122	1022	1133
1200x1200	1738	1565	1181	1269
1500x1500	2426	2022	1737	1896

TABLA 5.5: Tiempo de ejecución de WindNinja utilizando el precondicionar SSOR y el formato de almacenamiento VDSpM del cluster basado en AMD Opteron 6376 procesadores con 16 cores

Comparando los resultados obtenidos usando este formato de almacenamiento de la matriz A con los obtenidos en la implementación original de WindNinja (Tablas 2.3 y 2.4), se puede observar que el tiempo de ejecución se reduce en ambos casos, y el precondicionador Jacobian proporciona una mejor escalabilidad.

El tiempo de ejecución para un terreno de 1500x1500 celdas se reduce, en el caso SSOR, desde 2923 segundos a 2426 segundos, esto significa SpeedUp de $1,20$. Cuando se utilizan 4 cores, el tiempo de ejecución se reduce a 1737 segundos, lo que significa un SpeedUp de $1,68$.

En el caso de la precondicionador Jacobian, el tiempo de ejecución, para el mismo mapa, se reduce de 3954 segundos a 3560 segundo cuando se aplica el formato de

Mapa	Tiempo (Segundos) Ncores			
	1	2	4	8
400x400	80	45	33	35
500x500	139	82	58	60
600x600	330	182	138	135
800x800	559	304	233	228
1000x1000	1850	1046	771	765
1200x1200	2650	1498	1104	1106
1500x1500	3560	1978	1483	1489

TABLA 5.6: Tiempo de ejecución de WindNinja utilizando el precondicionar Jacobian y el formato de almacenamiento VDSpM del cluster basado en AMD Opteron 6376 procesadores con 16 cores

almacenamiento VDSpM de la matriz A. Por otra parte, cuando el número de cores se incrementa hasta a 4, el tiempo de ejecución se reduce a 1483 segundos, lo que significa un SpeedUp de $2,66$.

Por lo tanto, el precondicionador Jacobian presenta una mejor escalabilidad que SSOR. Esto es debido al hecho de que, en el caso de precondicionador jacobian, la operación $q_0 = M^{-1}g_0$ del algoritmo es muy simple, ya que la matriz M es una matriz diagonal. Por lo tanto, cuando se utiliza este precondicionador, el cuello de botella del algoritmo es el paso Ap_k que corresponde a la multiplicación SpMV. Sin embargo, en el caso de SSOR, la operación $q_0 = M^{-1}g_0$ también es un cuello de botella, que requiere una gran cantidad de cálculo que no se evita cuando se utiliza el formato de almacenamiento VDSpM. Por lo tanto, se ha llevado a cabo un análisis más profundo de esta operación.

5.4.1 Aceleración del PCG con SSOR

Como se mencionó anteriormente (capítulo 2) el precondicionador de SSOR es una matriz que es el resultado de multiplicar tres matrices tal como se expresa en la Ecuación 2.6. Después, esta matriz resultante debe ser invertida, y esta operación no es posible a nivel computacional a causa del gran número de recursos necesarios. Por lo tanto, la expresión $q_{k+1} = M^{-1}g_{k+1}$ se transforma en $Mq_{k+1} = g_{k+1}$ que puede ser expresado como: $L'D'U'q_{k+1} = g_{k+1}$ (Ecuación 2.6), donde el término que se calcula es el vector de q_{k+1}. Un primer paso para calcular q_{k+1}, es calcular z_{k+1} de $L'z_{k+1} = g_{k+1}$. Cuando se obtiene z_{k+1}, es posible calcular q_{k+1} de $U''q_{k+1} = z_{k+1}$.

Por lo tanto, es necesario empezar por llevar a cabo el procedimiento iterativo se describe en la sección 2.2.2 para calcular z_{k+1}. Como se describió anteriormente, en este sistema, es posible determinar la primera componente del vector z_{k+1} directamente. A continuación, se utiliza este valor para calcular el segundo, y así sucesivamente. Un

ejemplo de este proceso se representa en las Figuras 5.13 y 5.14. En la Figura 5.13, se muestra el cálculo del componente z_2. Una vez z_0 se ha calculado, el valor obtenido se multiplica por $L'_{2,0}$ y el resultado se resta de g_2. El nuevo valor obtenido se divide por $L_{2,2}$ y se obtiene el valor de z_2.

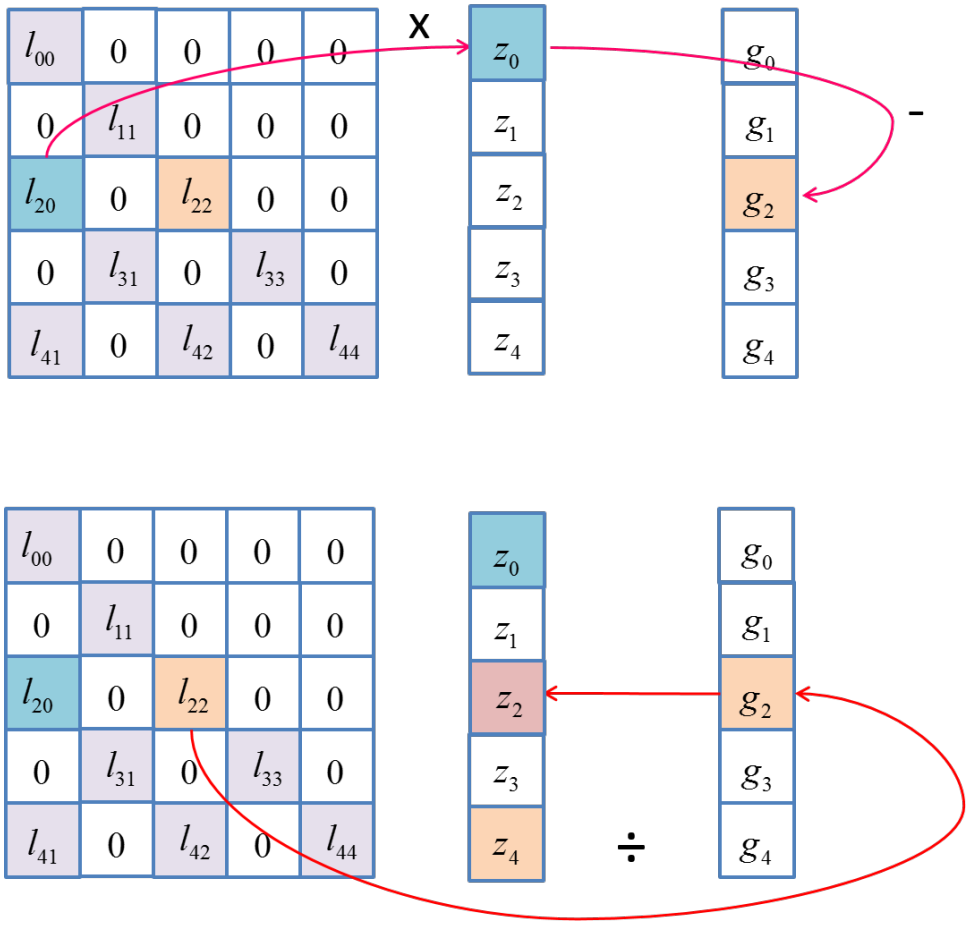


FIGURA 5.13: Cálculo de z_2 usando el formato CRS

Una vez z_2 ha sido calculado, es posible calcular los siguientes valores (en este ejemplo, z_4). Para el cálculo de z_4, es necesario multiplicar $L'_{4,0}$ por z_0 y $L'_{4,2}$ por z_2, y restar los valores obtenidos de g_4 . Por último, este resultado se debe dividir por $L'_{4,4}$ para obtener z_4.

Este proceso debe ser repetido en la dirección opuesta, para calcular los componentes de q_{k+1} utilizando la matriz triangular superior como se describe en la sección 2.2.2.

El almacenamiento de las matrices en formato CRS implica el cálculo por fila, que es en realidad la aplicación directa del método propuesto. Sin embargo, esta forma de tratamiento de los datos introduce algunas serializaciones y genera en muchos errores de caché. Por lo tanto, se propone para almacenar las matrices relacionadas con el preacondicionador en CCS [61] y se realiza el cálculo de forma diferente.

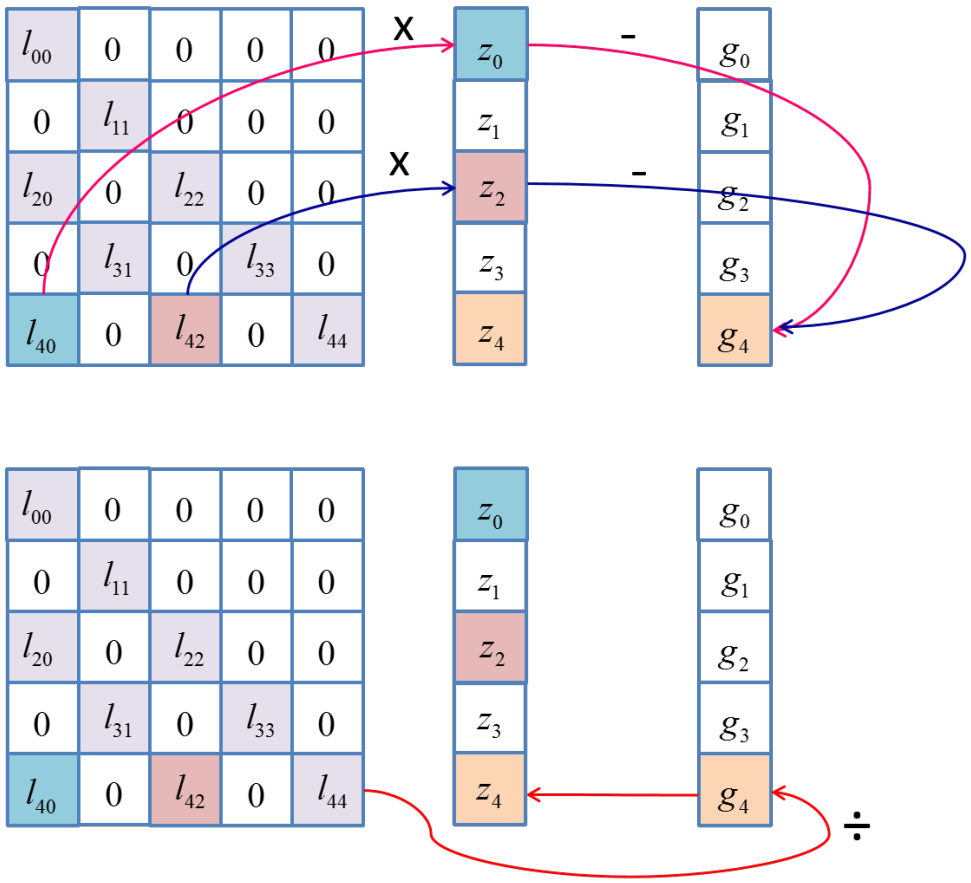


FIGURA 5.14: Cálculo de z_4 usando el formato CRS

El formato CCS es equivalente a formato CRS, pero la matriz se almacena por columnas en lugar de filas. La Figura 5.15 muestra la organización de los tres vectores de formato de almacenamiento de CCS.

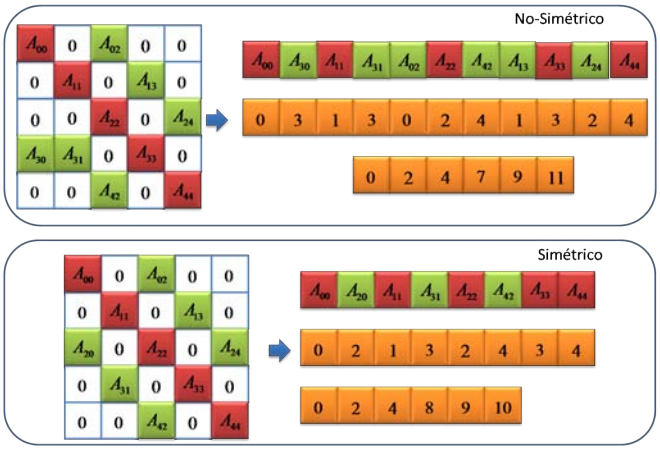


FIGURA 5.15: Formato de almacenamiento para matrices dispersas CCS

Usando el formato de almacenamiento CCS es posible organizar el cálculo de los componentes de z de una manera diferente. Una vez z_1 ha sido calculado, es posible multiplicar el

valor obtenido por todos los valores de la primera columna de la matriz L', y restar cada valor obtenido de la componente correspondiente de g, como se muestra en el ejemplo de la Figura 5.16. En cada paso (columna) un nuevo componente de z se obtiene dividiendo el componente correspondiente de g por el componente de la diagonal. Este proceso se repite columna por columna, pero los elementos de la columna se acceden de forma lineal, y sólo un componente de z es accedido a cada paso, lo que reduce los errores de caché.

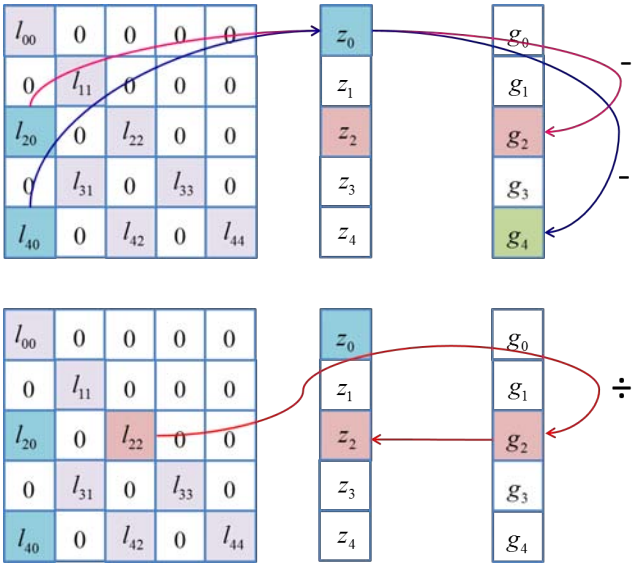


FIGURA 5.16: Cálculo de z usando el formato CCS

La misma estrategia se aplica para calcular el vector q_{k+1}, pero en este caso, la matriz triangular superior u'' se almacenan como triangular inferior y los componentes de z y q son también invertido para llevar a cabo el cálculo de la misma forma.

El tiempo de ejecución de diferente tamaño del mapa y diferente número de cores se muestra en la Tabla 5.7. Se puede observar que el tiempo de ejecución de un mapa 1500 x 1500 celdas se reduce a 1153 segundos. Esto significa un SpeedUp de 2,53 en comparación con la aplicación WindNinja SSOR original en CRS, y una velocidad en marcha de 1,5 en comparación con la versión mejorada de la aplicación SSOR con el formato de VDSpM para la matriz A.

5.5 Resultados experimentales

Los resultados obtenidos se han presentado en los anteriores puntos, pero en esta sección se comparan estos resultados.

Map	Execution Time (Seconds) Ncores			
	1	2	4	8
400x400	92	81	74	81
500x500	140	123	112	120
600x600	201	189	161	166
800x800	340	325	271	276
1000x1000	1008	935	806	906
1200x1200	1142	1125	902	924
1500x1500	1457	1424	1153	1166

TABLA 5.7: Tiempo de ejecución de WindNinja utilizando el precondicionar CCS y el formato de almacenamiento VDSpM del cluster basado en AMD Opteron 6376 procesadores con 16 cores

La Figura 5.17 muestra el tiempo de ejecución WindNinja en un sólo core teniendo en cuenta diferentes tamaños del mapa y el uso de diferentes implementaciones de PCG. Estas implementaciones son:

- WindNinja original con precondicionador SSOR
- WindNinja original con precondicionador Jacobian
- Precondicionador SSOR con las matrices de SSOR almacenadas en CRS y la matriz A en formato VDSpM.
- Precondicionador Jacobian con almacenamiento de la matriz A en formato VDSpM.
- Precondicionador SSOR con las matrices SSOR almacenadas en formato en CCS y la matriz A en formato VDSpM.

Los resultados muestran que el precondicionador SSOR es más rápido que preacondicionador Jacobian. SSOR converge en un número inferior de iteraciones y las mejoras introducidas en el formato y las operaciones de almacenamiento hace referencia en las diferencias entre Jacobian y SSOR.

La Figura 5.18 muestra el tiempo de ejecución para un mapa de 1500x1500 celdas teniendo en cuenta las cinco implementaciones y diferentes números de cores. En este caso, se puede observar que las implementaciones con Jacobian presentan una mejor escalabilidad que las implementaciones SSOR, reduciendo el tiempo de ejecución más significativamente para cada aplicación. Sin embargo, la aplicación SSOR con formato de matriz VDSpM para A y CCS para el precondicionador proporciona el mejor tiempo de ejecución.

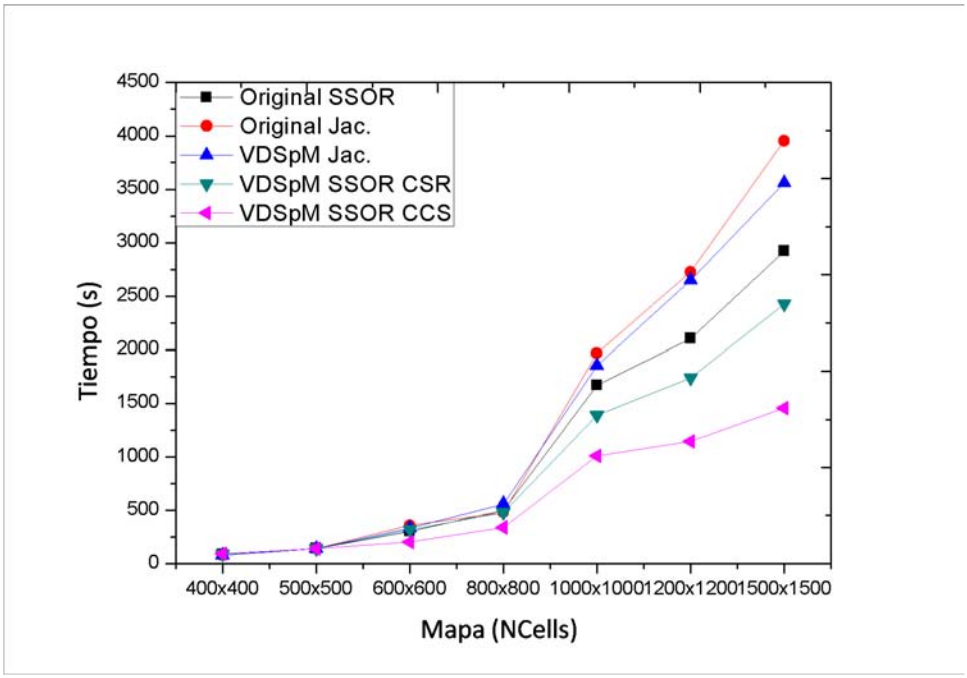


FIGURA 5.17: Tiempo de ejecución usando diferentes implementaciones para el cluster basado en AMD Opteron 6376 procesadores con 16 cores considerando diferentes tamaños de mapa

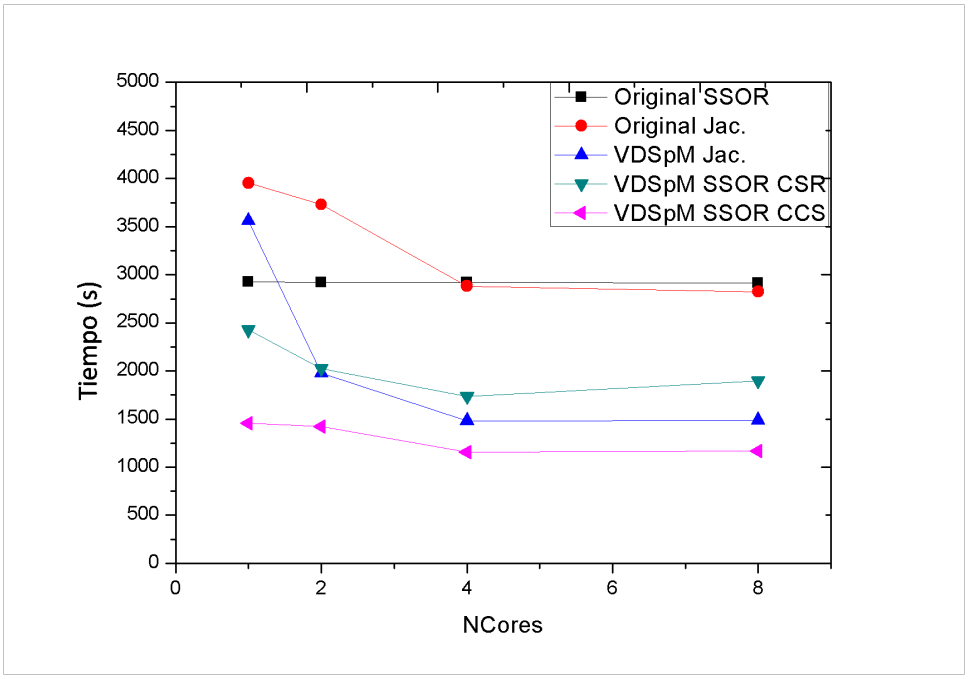


FIGURA 5.18: Tiempo de ejecución de WindNinja para un mapa de 1500x1500 celdas considerando diferentes implementaciones y número de cores para un cluster basado en AMD Opteron 6376 procesadores con 16 cores

5.6 Conclusiones del capítulo

En este capítulo se ha analizado la implementación del solver gradiente conjugado con precondicionador que incorpora WindNinja y se han propuesto mejoras en el método de almacenamiento y en las operaciones que implican un mayor tiempo de procesamiento. Se ha propuesto un nuevo método de almacenamiento de las matrices dispersas basado en la vectorización de las diagonales, y se ha propuesto un método paralelo de multiplicación matriz dispersa-vector que explota las características del formato de almacenamiento desarrollado. También se ha analizado las implementaciones y cuellos de botella de los precondicionadores SSOR y Jacobian integrados en WindNinja, de modo que se ha mejorado el tiempo de ejecución y la escalabilidad del algoritmo. De todos modos, la reducción conseguida en el tiempo de ejecución para mapas grandes, aunque significativas, no permiten alcanzar la solución del campo de vientos en un tiempo operacionalmente aceptable.

Capítulo 6

Integración de las distintas aproximaciones: Aplicación híbrida para el cálculo del campo de vientos

6.1 Introducción

Todos los métodos recogidos en los capítulos anteriores han demostrado que mejoran el tiempo de ejecución de WindNinja. Pero cuando los mapas considerados son muy grandes, ninguno de los métodos consigue reducir el tiempo de ejecución por debajo de los límites establecidos para la operación real. Como se ha visto, ninguno de los métodos ofrece una escalabilidad suficiente más allá de 6-10 procesos. Sin embargo, las distintas aproximaciones abordan fases y procedimientos diferentes en el cálculo del campo de viento. Por ello es imprescindible combinar los diferentes métodos en una aplicación híbrida que combine los beneficios de las tres aproximaciones mostradas, y, así, se consiga reducir el tiempo de ejecución por debajo de los límites establecidos.

Por tanto, en este capítulo se analizan las posibilidades de integrar diferentes métodos en una misma aplicación híbrida MPI-OpenMP que explote el paralelismo y los métodos acelerados desarrollados. En una primera aproximación se combinará el método de Schwarz de Descomposición del Dominio con la aceleración del gradiente conjuga con precondicionador. En un segundo paso, se integrarán los tres métodos de mejora introducidos en los tres capítulos anteriores en una única aplicación híbrida.

6.2 Integración de la aceleración del PCG en el Método de Schwarz de Descomposición del Dominio

En este caso, a la aplicación paralela MPI que resolvía el sistema de ecuaciones mediante el método de Schwarz (Capítulo 4, se le han incorporado las mejoras realizadas para acelerar el gradiente conjugado con precondicionador (PCG) (Capítulo 5). Tal como se mostró en el capítulo 4, el método de Schwarz admite una paralelización en un esquema parecido al Master-Worker, en la que cada worker ejecuta el algoritmo del PCG para resolver la parte de la matriz que les corresponde. Para resolver este sistema mediante PCG, es posible aplicar las mejoras introducidas en el capítulo 5 para conseguir reducir el tiempo de ejecución.

En esta aplicación, el Master lleva a cabo la descomposición de la malla en subdominios y distribuye los subdominios a los Workers. A continuación, los Workers almacenan la matriz en formato VDSpM y ejecutan el PCG, utilizando el método de multiplicación matriz dispersa-vector descrito en el capítulo 5. Este método de multiplicación ha sido paralelizado en MPI y en OpenMP, pero para su integración con el método de descomposición del dominio, se ha escogido la paralelización con OpenMP, ya que resultaba más fácil de integrar y además ofrecía unas prestaciones ligeramente mejores.

De esta manera, se ha implementado una aplicación híbrida MPI-OpenMP (Schwarz-VDSpMV). Algunos de los resultados se resumen en la Tabla 6.1. El número de Workers (subdominios) que se consideran son 1, 2, 3 y 4, y el número de cores el número de Workers que se aplica es 1, 2 y 4. Estos resultados muestran que un mapa de 800 x 800 se pueden resolver en sólo 63 segundos aplicando 4 subdominios (4 nodos) y 4 cores por subdominio con un SpeedUp de $12,76$.

Este método es capaz de resolver muchos mapas en menos de 100 segundos, pero, cuando los mapas son muy grandes (por ejemplo, 1500 x 1500 celdas), el tiempo que se consigue no respeta los límites establecidos para el funcionamiento operacional del método y es necesario ir un paso más allá e integrar el método de paralelización mediante particionamiento del mapa, expuesto en el capítulo 3.

			Time (s) NCores			Speedup NCores		
Map Size	Subd.	WN	1	2	4	1	2	4
400x400	1	237	215	191	135	1,10	1,24	1,75
	2		79	57	53	3,00	4,15	4,47
	3		54	39	30	4,38	6,07	7,90
	4		53	39	23	4,47	6,07	10,30
600x600	1	567	505	325	231	1,12	1,74	2,45
	2		166	122	90	3,41	4,64	6,30
	3		115	85	64	4,93	6,67	8,85
	4		90	67	38	6,30	8,46	14,92
800x800	1	804	769	738	637	1,04	1,08	1,26
	2		294	214	144	2,73	3,75	5,58
	3		193	141	107	4,16	5,70	7,51
	4		149	111	63	5,39	7,24	12,76
1000x1000	1	1836	1658	1300	967	1,10	1,40	1,89
	2		1028	751	436	1,78	2,43	4,20
	3		635	463	353	2,88	3,95	5,18
	4		503	376	213	3,64	4,87	8,59
1200x1200	1	2939	2714	1871	1300	1,08	1,57	2,26
	2		1072	782	454	2,74	3,75	6,47
	3		842	614	468	3,49	4,78	6,28
	4		710	534	301	3,76	5,50	9,76
1500x1500	1	3831	3080	2488	2006	1,24	1,54	1,91
	2		1250	962	712	3,06	3,98	5,38
	3		984	757	547	3,89	5,06	7,00
	4		781	601	501	4,90	6,37	7,64

TABLA 6.1: Tiempo de ejecución y Speedup de la aplicación Híbrida Schwarz-VDSpMV

6.3 Integración del particionameinto del mapa con el método de Schwarz de Descomposición del Dominio y la aceleración del PCG

El objetivo consiste en aprovechar la paralelización mediante el particionamiento del mapa, con el fin de tener que trabajar con mapas de unas dimensiones más acotadas, de modo que puedan ser resueltos aplicando el método de descomposición del dominio y la aceleración del PCG en unos tiempos que respeten los límites establecidos.

En este caso, el mapa inicial se divide en un cierto número de partes mediante una aplicación de Master/Worker MPI. Entonces, cada Worker aplica el método de Schwarz de descomposición del dominio usando un cierto número de subdominios que se distribuyen a subworkers. Finalmente, cada subworker utiliza la multiplicación matriz dispersa-vector basada en VDSpM. Esta estructura se muestra en la Figura 6.1.

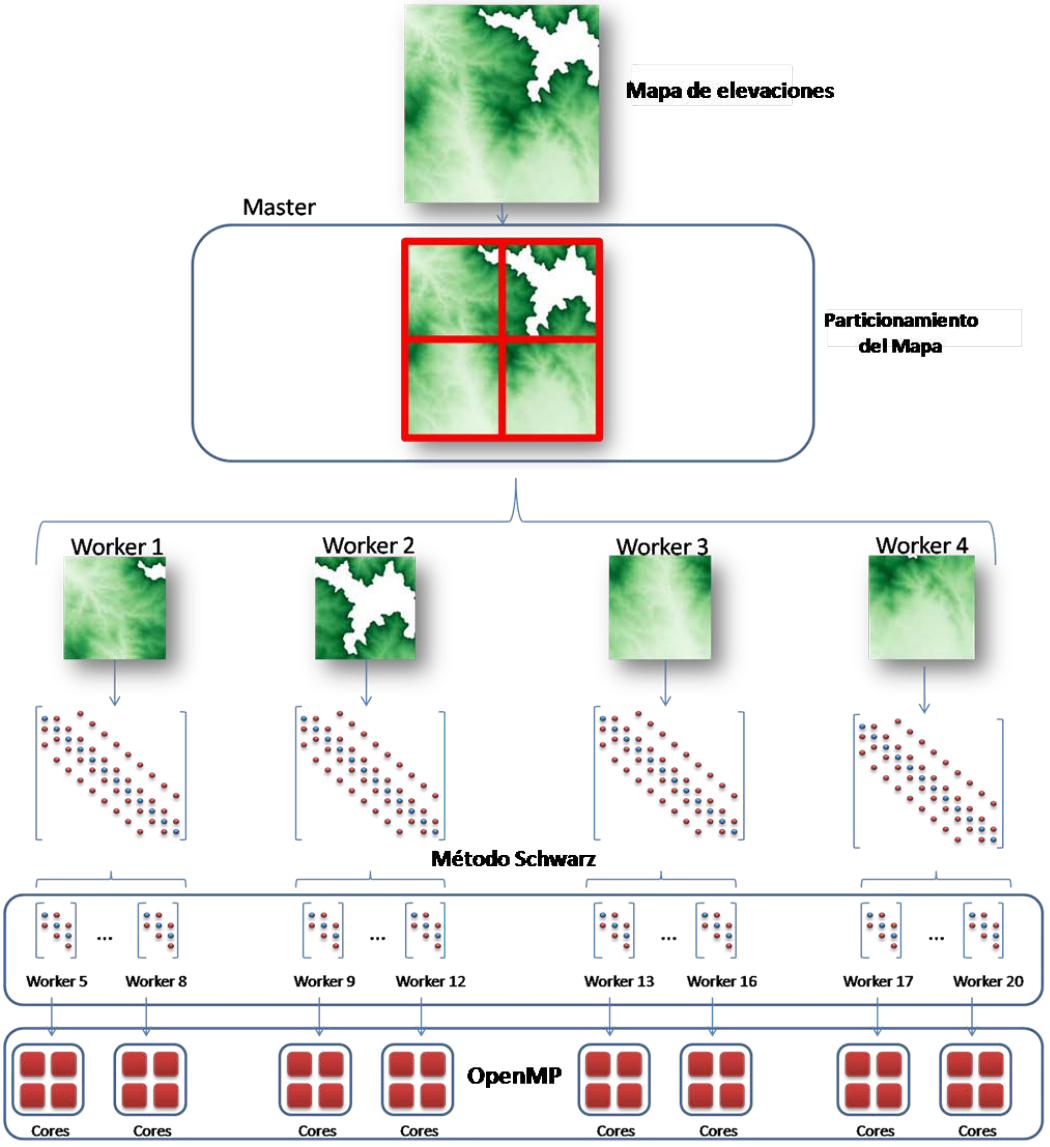


FIGURA 6.1: Aplicación híbrida MPI-OpenMP integrando particionamiento del mapa, descomposición del dominio y aceleración de PCG

Algunos resultados experimentales para un mapa de 1500 x 1500 se muestran en la Tabla 6.2. En estos experimentos, los resultados de resolver el mapa completo se comparan con diferentes particionamientos 2 x 2, 3 x 3 y 5 x 5 (*Partes*). En cada partición se ha considerado un (*Solapamiento*) de 50 celdas. Para cada partición, se consideran diferentes configuraciones referentes a la descomposición del dominio y número de cores por subdominio para explotar la aceleración del PCG. El número de subdominios (*Subd.*) Se establece en 1 o 4 y el número de cores (*NCores*) utilizado por cada subdominio varía también entre 1 y 4. Así, en el caso máximo, la aplicación utiliza 400 cores (*TotCores*), ya que son 25 partes, con 4 subdominios por parte y 4 cores por subdominio. En este caso, el tiempo de ejecución se reduce de 3831 segundos a 30, 3 segundos, con un RMSE 0.56 m/s para la velocidad del viento y la diferencia de la predicción de la propagación estimada

en $0,12$. Este valor $(0,12)$ es bastante grande y la diferencia en la propagación del fuego se puede detectar a simple vista mediante la comparación de las dos propagaciones. Sin embargo, cuando el mapa se divide en 2 x 2 y en 4 subdominios por cada parte y se utilizan 4 cores por cada subdominio, el tiempo de ejecución es de $91,1$ segundos, el RMSE para la velocidad del viento es de $0,31$ y la diferencia de la propagación del fuego es $0,009$ que no es significativa. En este caso, el número de cores requeridos es de 64 (4 x 4 x 4). Por lo tanto, este método permite reducir el tiempo de ejecución y cumple con las limitaciones de tiempo real, sin una pérdida significativa en la precisión de la predicción de propagación de fuego y sin necesidad de una gran cantidad de recursos.

Mapa	Partes	Subd.	NCores	TotCores	Tiempo (s)	RMSE	Dif.
1500x1500	1	1	1	1	3831	0	0
1500x1500	1	1	4	4	2006	0	0
1500x1500	1	4	1	4	867	0	0
1500x1500	1	4	4	16	501	0	0
1500x1500	2x2	1	1	4	1171	0,31	0,009
1500x1500	2x2	1	4	16	377	0,31	0,009
1500x1500	2x2	4	1	16	144	0,31	0,009
1500x1500	2x2	4	4	64	91	0,31	0,009
1500x1500	3x3	1	1	9	658	0,35	0,03
1500x1500	3x3	1	4	36	223	0,35	0,03
1500x1500	3x3	4	1	36	103	0,35	0,03
1500x1500	3x3	4	4	144	61	0,35	0,03
1500x1500	5x5	1	1	25	190	0,56	0,12
1500x1500	5x5	1	4	100	63	0,56	0,12
1500x1500	5x5	4	1	100	43	0,56	0,12
1500x1500	5x5	4	4	400	30	0,56	0,12

Tabla 6.2: Tiempo de ejecución, RMSE de la velocidad del viento y diferencia en la propagación del fuego para un mapa de 1500x1500 celdas, considerando diferentes configuraciones.

6.4 Conclusiones del capítulo

Con el fin de conseguir ejecutar el simulador de campo de vientos dentro de los límites temporales establecidos por los requerimientos operacionales, se ha desarrollado una aplicación híbrida MPI-OpenMP que integra los tres métodos desarrollados a lo largo de la investigación. Con esta aplicación, es posible resolver mapas muy grandes (1500x1500 celdas) en un tiempo inferior a los 100 segundos, utilizando un conjunto de recursos computacionales que pueden estar disponibles en muchos sistemas actuales de gama media. Los tiempos de ejecución alcanzados permiten llevar esta aplicación a un entorno operacional de tiempo real.

Capítulo 7

Conclusiones y líneas abiertas

7.1 Conclusiones

La investigación llevada a cabo se ha centrado en la aceleración (reducción del tiempo de ejecución) de un simulador de campo de vientos para su acoplamiento con un simulador de propagación de incendios forestales, con el fin de proporcionar una predicción lo más ajustada posible a la realidad de la propagación de los incendios forestales.

El cálculo del campo de vientos a alta resolución (30 metros) para terrenos de dimensiones importantes (45x45 km) implica la resolución de sistemas de ecuaciones enormes, con centenares de millones de incógnitas, que deben ser resueltos por métodos iterativos, como es el gradiente conjugado con precondicionador (PCG). Esta resolución puede llevar un tiempo de ejecución que hace que el cálculo del campo de vientos no pueda ser utilizado en el funcionamiento operacional de los medios de extinción.

Por tanto, en esta tesis doctoral se han propuesto 3 aproximaciones para reducir el tiempo de ejecución que explotan distintas formas de paralelismo que ofrece la aplicación.

- Una primera aproximación consistió en la división y reparto de los datos de entrada, en este caso el mapa de elevaciones del terreno, entre distintos procesos Worker. Esta aproximación requería un cierto grado de solapamiento e introducía una diferencia con respecto al cálculo del campo de vientos considerando un mapa global, que podía afectar a la predicción de la propagación del incendio forestal. Para minimizar estos problemas, se desarrolló una metodología completa que permitía determinar:

  1. Forma de las partes en las que se divide el mapa
  2. Tamaño de las partes para cumplir las limitaciones temporales

3. Grado de solapamiento entre las partes
4. Resolución adecuada del mapa
5. Efecto de la velocidad del viento

Esta investigación dio lugar a una publicación en una revista indexada de segundo quartil, dos publicaciones en congresos internacionales clasificados como CORE:A, una publicación en un congreso internacional clasificado como CORE:B, una publicación en un congreso internacional clasificado como CORE:C, dos publicaciones en congresos internacionles de referencia en el campo de la modelización de la propagación de incendios forestales y una publicación en un congreso nacional. Estas publicaciones son las siguientes:

1. Gemma Sanjuan, Carlos Brun, Tomás Margalef, Ana Cortés
   Determining map partitioning to minimize wind field uncertainty in forest fire propagation prediction
   Journal of Computational Science (en impresión) 2016. **IF: 1,231 (Q2 - 35/102)**
2. Gemma Sanjuan, Carlos Brun, Tomás Margalef, Ana Cortés
   Wind field uncertainty in forest fire propagation prediction
   International Conference on Computational Science 2014. Procedia Computer Science, 29, pp. 1535-1545. 2014. **CORE A**
3. Gemma Sanjuan, Tomás Margalef, Ana Cortés.
   Adapting map resolution to accomplish execution time constraints in wind field calculation
   International Conference on Computational Science 2015. Procedia Computer Science, 51, pp. 2749-2753. 2015. **CORE A**
4. Gemma Sanjuan, Carlos Brun, Tomás Margalef, Ana Cortés
   Determining map partitioning to accelerate wind field calculation
   International Conference on High Performance Computing & Simulation (HPCS 2014), pp. 96-103. 2014. **CORE B**
5. Gemma Sanjuan, Carlos Brun, Tomás Margalef, Ana Cortés
   Effect of wind field parallelization on forest fire spread prediction
   14th International Conference on Computational Science and Applications (ICCSA-2014). Lecture Notes in Computer Science, 8582, pp. 538-549. 2014. **CORE C**
6. Gemma Sanjuan, Carlos Brun, Tomás Margalef, Ana Cortés
   Map Partitioning to accelerate wind Field calculation for Forest Fire Propagation Prediction
   International Conference on Forest Fire Research 2014, pp. 336-343. 2014.

7. Gemma Sanjuan, Tomás Margalef, Ana Cortés
   Effect of map resolution on wind field accuracy Prediction
   International Conference on Fire Behaviour Research 2015.
8. Gemma Sanjuan, Carlos Brun, Tomás Margalef, Ana Cortés
   Paralelizacion del cálculo del campo de vientos para predicción de la propagación de incendios forestales
   XXIII Jornadas de Paralelismo 2013. pp. 259-264. 2013.

- En una segunda aproximación se abordo la paralelización mediante métodos de descomposición del dominio. En estos métodos, la malla y el sistema de ecuaciones que se genera se divide en subdominios, para los que hay que resolver partes del sistema de ecuaciones, con lo cual se consigue reducir el tiempo de ejecución. En estos métodos, se definen los subdominios y la frontera entre los distintos subdominios, ya sea con solapamiento (método de Schwarz) o sin solapamiento (método de Schur). La limitación que presentan es que a medida que aumenta el número de subdominios, aumenta la frontera entre los mismos, de modo que la resolución del subsistema que representa la frontera se convierte en la limitante del método. Se aplicaron tanto el método de Schwarz como el de Schur y, en general, para descomposiciones en 6-10 subdominois ofrecen una escalabilidad bastante razonable, pero más allá de estos valores el rendimiento comienza a decrecer.

  Esta línea de investigación ha dado lugar a 1 artículo aceptado, 1 artículos que se encuentran en evaluación (uno de ellos en segunda revisión) en revistas indexadas de primer quartil, un artículo publicado en un congreso internacional clasificado como CORE:B, y dos artículos en congresos centrados en métodos matemáticos. Estas publicaciones son:

  9. Gemma Sanjuan, Tomás Margalef, Ana Cortés
     Applying domain decomposition to wind field calculation
     Parallel Computing. **IF: 1,511 (Q1 - 23/102)**
  10. Gemma Sanjuan, Tomás Margalef, Ana Cortés
      Wind Field parallelization based on domain decomposition
      Future Generation Computer Systems. **IF: 2,786 (Q1 - 8/102)** (Enviado))
  11. Gemma Sanjuan, Tomás Margalef, Ana Cortés
      Applying Domain Decomposition Schwarz Method to Accelerate Wind Field Calculation
      International Conference on High Performance Computing & Simulation 2015 (HPCS 2015), pp. 484-490. 2015. **CORE B**

12. Gemma Sanjuan, Tomás Margalef, Ana Cortés
Applying Domain Decomposition to Wind Field Calculation
International Workshop on Parallel Matrix Algorithms and Applications.
13. Gemma Sanjuan, Tomás Margalef, Ana Cortés
Accelerating Schur Complement Domain Decompostion Method for Wind Field Calculation
16th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2016) (enviado).

- En la tercera aportación se analizó en profundidad le solver del gradiente conjungado con precondicionador (PCG) y se estudió el formato de almacenamiento de las matrices dispersas involucradas y los métodos de multiplicación matriz dispersa-vector. Se observó que las matrices generadas por WindNinja son matrices dispersas, simétricas, definidas positivas, con tan solo una diagonal principal y 13 subdiagonales en cada triangular. Para representar esta estructura se propuso un nuevo formato de almacenamiento, denominado *Vectorization of Diagonal Sparse Matrix (VDSpM)* y se propuso un método de multiplicación matriz dispersa-vector que permite se paralelizado en OpenMP y MPI y que introduce un nuevo punto de mejora en el tiempo de ejecución.

  Este estudio dio lugar a una publicación en una revista de tercer quartil, a una publicación en un congreso internacional clasificado como CORE:B y a una publicación en un congreso sobre métodos matemáticos.

14. Gemma Sanjuan, Carles Tena, Tomás Margalef, Ana Cortés
Applying vectorization of diagonal sparse matrix to accelerate wind field calculation
The Journal of Supercomputing (en impresión). **IF: 0,858 (Q3 - 25/50)**
15. Gemma Sanjuan, Tomás Margalef, Ana Cortés
Accelerating Preconditioned Conjugate Gradient solver in Wind Field Calculation
International Conference on High Performance Computing & Simulation 2016 (HPCS 2016). **CORE B** (Aceptado)
16. Gemma Sanjuan, Carles Tena, Tomás Margalef, Ana Cortés
Applying vectorization of diagonal sparse matrix to accelerate wind field calculation
15th International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2015), pp. 1011-1022. 2015.

- Todas estas aproximaciones fueron combinadas en una aplicación híbrida MPI-OpenMP que aplica las tres metodologías y consigue reducir el tiempo de ejecución, sin un uso demasiado excesivo de recursos computacionales. Esta aplicación ha dado lugar a un artículo que se encuentra en segunda revisión en un revista de segundo quartil.

17. Gemma Sanjuan, Tomás Margalef, Ana Cortés
Hybrid application to accelerate Wind Field calculation
Journal of Computational Science. **IF: 1,231 (Q2 - 35/102)** (2a revisión)

7.2 Lineas Abiertas

Fruto de la investigación llevada a cabo, se dispone de un aplicación que explota paralelismo a distintos niveles para conseguir reducir el tiempo de ejecución por debajo de los límites establecidos por los requerimientos operacionales de tiempo real. Podría parecer que con estos métodos que se han desarrollado no es necesario realizar ninguna mejora más sobre el campo, pero esto no es así. Hay muchos puntos en los que todavía es importante poder mejorar:

- Las nuevas arquitecturas con procesadores, y muy especialmente aceleradores, *many-core*, ofrecen unas posibilidades de cómputo y unos ahorros energéticos que pueden resultar muy útiles para este tipo de aplicaciones. Las GPUs y aceleradores como el Intel Xeon Phi, son arquitecturas que pueden reducir sustancialmente las necesidades de tiempo y energéticas de este tipo de aplicaciones.

- El uso de matrices de gran tamaño implica una gestión de memoria muy detallada para optimizar el uso de la misma y minimizar los fallos de memoria cache. Este análisis detallado es especialmente importante cuando se utilicen aceleradores, como las GPUs, con un sistema de memoria con unas características muy particulares.

- La inclusión en los modelos de campo de vientos de elementos más precisos, como son los modelos de dinámica de fluidos (CFD), van a generar modelos mucho más detallados que van a ser computacional mente más exigentes y deberán ser analizados y paralelizados, explotando las características de los sistemas actuales y futuros, para conseguir cumplir con todos los requerimientos.

- El modelo de campo de vientos que incorpora WindNinja está pensado para introducir en la evolución de los incendios forestales, el efecto del viento teniendo en cuenta la orografía del terreno, pero no tienen en cuenta el efecto del incendio sobre el viento. Actualmente, se dispone de unos primeros modelos que combinan el efecto bidireccional, del viento sobre le fuego y del viento sobre la atmósfera, pero son muy costosos desde un punto de vista computacional y no pueden ser utilizados en tiempo real. Estos modelos, cada vez más detallados van a requerir de un trabajo muy importante desde el área de la computación de altas prestaciones para llevarlos a modelos operacionales.

Bibliografía

[1] WR Catchpole, EA Catchpole, BW Butler, RC Rothermel, GA Morris, and DJ Latham. Rate of spread of free-burning fires in woody fuels in a wind tunnel. *Combustion Science and Technology*, 131(1-6):1–37, 1998.

[2] D.X. Viegas. Forest fire propagation. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 356(1748): 2907–2928, 1998. ISSN 1364-503X. doi: 10.1098/rsta.1998.0303. URL http://rsta.royalsocietypublishing.org/content/356/1748/2907.

[3] V. Mallet, D.E. Keyes, and F.E. Fendell. Modeling wildland fire propagation with level set methods. *Computers Mathematics with Applications*, 57(7):1089 – 1101, 2009. ISSN 0898-1221. doi: http://dx.doi.org/10.1016/j.camwa.2008.10.089. URL http://www.sciencedirect.com/science/article/pii/S0898122108006329.

[4] J.K. Adou, A.D.V. Brou, and B. Porterie. Modeling wildland fire propagation using a semi-physical network model. *Case Studies in Fire Safety*, 4:11 – 18, 2015. ISSN 2214-398X. doi: http://dx.doi.org/10.1016/j.csfs.2015.05.003. URL http://www.sciencedirect.com/science/article/pii/S2214398X15000084.

[5] S. B. Santra and I. Bose. Dynamics of forest fire under rotational constraint. *Zeitschrift für Physik B Condensed Matter*, 89(2):247–250. ISSN 1431-584X. doi: 10.1007/BF01320943. URL http://dx.doi.org/10.1007/BF01320943.

[6] Peter G. Baines. Physical mechanisms for the propagation of surface fires. *Mathematical and Computer Modelling*, 13(12):83 – 94, 1990. ISSN 0895-7177. doi: http://dx.doi.org/10.1016/0895-7177(90)90102-S. URL http://www.sciencedirect.com/science/article/pii/089571779090102S.

[7] J.-L. Dupuy and D. Valette, J.-C.and Morvan. A complete physical model of forest fire behaviour as a tool to manage the forest fuel on wui. In *Forest fires in th wildland-urban interface and rural areas in Europe: an integral planning and management challenge*, pages 105–112, 2004.

[8] E. Koo, P. Pagni, S. Stephens, J. Huff, J. Woycheese, and D.R. Weise. A simple physical model for forest fire spread rate. *Fire Safety Science*, 8:851–862, 2005. doi: 10.3801/IAFSS.FSS.8-851.

[9] R.C. Rothermel. *How to predict the spread and intensity of forest and range fires.* Intermountain Forest and Range Experiment Station Ogden, 1983.

[10] F.A. Albini, Intermountain Forest, Utah) Range Experiment Station (Ogden, and United States. Forest Service. *Estimating wildfire behavior and effects.* General technical report INT. Dept. of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station, 1976.

[11] Joe H. Scott and Robert E. Burgan. *Standard fire behavior fuel models: a comprehensive set for use with Rothermel's surface fire spread model.* Gen. Tech. Rep. RMRS-GTR-153. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. 72 p. 2005.

[12] Patricia L. Andrews. *BEHAVE: fire behavior prediction and fuel modeling system - BURN subsystem, part 1.* General Technical Report INT-GTR-194. Ogden, UT: USDA Forest Service, Intermountain Research Station. 1986.

[13] J. Martinez Millan, S. Vignote, J. Martos, and D. Caballero. Cardin: a computer system for the simulation of wildland fire spread. *Forest Systems*, pages 121 – 133, 1991. doi: doi:http://dx.doi.org/10.5424/1299.

[14] M. A. Finney. *FARSITE, Fire Area Simulator–model development and evaluation.* Res. Pap. RMRS-RP-4, Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. 1998.

[15] A.M.G. Lopes, M.G. Cruz, and D.X. Viegas. Firestation — an integrated software system for the numerical simulation of fire spread on complex topography. *Environmental Modelling Software*, 17(3):269 – 285, 2002. ISSN 1364-8152. doi: http://dx.doi.org/10.1016/S1364-8152(01)00072-X.

[16] Patricia L. Andrews. Behaveplus fire modeling system: past, present, and future. In *Proceedings of 7th Symposium on Fire and Forest Meteorology*, page 13, 2007.

[17] Patricia L. Andrews. Current status and future needs of the behaveplus fire modeling system. *International Journal of Wildland Fire*, 23(1):21 – 33, 2013.

[18] K.G. Tolhurst and Derek Chong. Assessing potential house losses using phoenix rapidfire. http://www.bushfirecrc.com/resources/pages-74-86-assessing-potential-house-losses, 2011. ISSN 978-0-9806759-9-3.

[19] David Buckley, Joaquin Ramírez, and Santiago Monedero. New approaches in fire simulations analysis with wildfire analyst. In *5th International Wildland Fire Conference*, 2011.

[20] Aster global digital elevation map. https://asterweb.jpl.nasa.gov/gdem.asp, 2011 (accessed 27-Abril-2016).

[21] Inventari forestal i ecològic de catalunya. http://www.creaf.uab.cat/iefc/index.htm, 2011 (accessed 27-Abril-2016).

[22] Corine land cover. http://land.copernicus.eu/pan-european/corine-land-cover, 2012 (accessed 27-Abril-2016).

[23] M. Castro, J.C. Parra, L.J. Morales, and C. Salas. Establishment of empirical relations between fuel moisture content and the normalised difference vegetation index. *Journal of soil science and plant nutrition*, 14(3):670 – 675, 2014.

[24] Hamlyn G. Jones. Monitoring plant and soil water status: established and novel methods revisited and their relevance to studies of drought tolerance. *Journal of experimental botany*, 58(2):119 – 130, 2006.

[25] J. Michalakes, J. Dudhia, D.O. Gill, T.B. Henderson, J.B. Klemp, W. Skamarock, and W. Wang. The weather research and forecast model: software architecture and performance. In *11th Workshop on the Use of High Performance Computing in Meteorology*, 2004.

[26] J.P. Kerekes and D.A. Landgrebe. An analytical model of earth-observational remote-sensing systems. *IEEE Transaction on systems man and cybernetics*, 21 (1):125 – 133, 1991.

[27] Baker Abdalhaq, Ana Cortés, Tomàs Margalef, and Emilio Luque. Enhancing wildland fire prediction on cluster systems applying evolutionary optimization techniques. *Future Generation Computer Systems*, 21(1):61–67, 2005. ISSN 0167-739X.

[28] Mónica Denham, Ana Cortés, Tomàs Margalef, and Emilio Luque. Applying a dynamic data driven genetic algorithm to improve forest fire spread prediction. In Marian Bubak, Geert van Albada, Jack Dongarra, and Peter Sloot, editors, *Computational Science – ICCS 2008*, volume 5103 of *Lecture Notes in Computer Science*, pages 36–45. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-69388-8. 10.1007/978-3-540-69389-5_6.

[29] Germán Bianchini, Mónica Denham, Ana Cortés, Tomàs Margalef, and Emilio Luque. Wildland fire growth prediction method based on multiple overlapping

solution. *J. Comput. Science*, 1(4):229–237, 2010. doi: 10.1016/j.jocs.2010.07.005. URL http://dx.doi.org/10.1016/j.jocs.2010.07.005.

[30] Germán Bianchini, Paola Caymes-Scutari, and Miguel Méndez-Garabetti. Evolutionary-statistical system: A parallel method for improving forest fire spread prediction. *J. Comput. Science*, 6:58–66, 2015. doi: 10.1016/j.jocs.2014.12.001. URL http://dx.doi.org/10.1016/j.jocs.2014.12.001.

[31] 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.

[32] Andŕes Cencerrado, Ana Cortes, and Tom'as Margalef. Prediction time assessment in a dddas for natural hazard management: Forest fire study casei. *Procedia Computer Science*, 4:1761–1770, 2011.

[33] William Gropp. Mpich2: A new start for mpi implementations. In Dieter Kranzlmüller, Jens Volkert, Peter Kacsuk, and Jack Dongarra, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, volume 2474 of *Lecture Notes in Computer Science*, pages 37–42. Springer Berlin / Heidelberg, 2002. ISBN 978-3-540-44296-7.

[34] B. Chapman, G. Jost, and R. Van Der Pas. *Using OpenMP: portable shared memory parallel programming*, volume 10. The MIT Press, 2007.

[35] Tomàs Artés, Andrés Cencerrado, Ana Cortés, and Tomàs Margalef. Real-time genetic spatial optimization to improve forest fire spread forecasting in high-performance computing environments. *International Journal of Geographical Information Science*, 30(3):594–611, 2016. doi: 10.1080/13658816.2015.1085052. URL http://dx.doi.org/10.1080/13658816.2015.1085052.

[36] Baker Abdalhaq, Ana Cortés, Tomàs Margalef, Germán Bianchini, and Emilio Luque. Between classical and ideal: enhancing wildland fire prediction using cluster computing. *Cluster Computing*, 9(3):329–343, 2006. doi: 10.1007/s10586-006-9745-4. URL http://dx.doi.org/10.1007/s10586-006-9745-4.

[37] Carlos Brun, Tomàs Artées, Tomàs Margalef, and Ana Cortées. Coupling wind dynamics into a dddas forest fire propagation prediction system. *Procedia Computer Science*, 9:1110–1118, 2012.

[38] Farsite fire area simulator-model. http://www.firelab.org/project/farsite, 1995 (accessed 27-Abril-2016).

[39] JM Forthofer, K Shannon, and BW Butler. Initialization of high resolution surface wind simulations using nws gridded data. In *Proceedings of 3rd Fire Behavior and Fuels Conference*, pages 25–29, 2010.

[40] Windninja wind field simulator. http://www.firelab.org/project/windninja, 2009 (accessed 27-Abril-2016).

[41] Jason Forthofer, Kyle Shannon, and Bret Butler. 4.4 simulating diurnally driven slope winds with windninja. *USDA Forest Service, Rocky Mountain Research Station, Missoula, MT*, 2009.

[42] J. M. Forthofer, K. Shannon, and B. W. Butler. Simulating diurnally driven slope winds with windninja. In *8th Symposium on Fire and Forest Meteorological Society*, 2009.

[43] Gemma Sanjuan, Carles Tena, Tomàs Margalef, and Ana Cortés. Applying vectorization of diagonal sparse matrix to accelerate wind field calculation. *The Journal of Supercomputing*, pages 1–19, 2016. ISSN 1573-0484. doi: 10.1007/s11227-016-1696-9. URL http://dx.doi.org/10.1007/s11227-016-1696-9.

[44] Jorge Nocedal and Stephen J Wright. Conjugate gradient methods. *Numerical Optimization*, pages 101–134, 2006.

[45] Youcef Saad. Sparskit: A basic tool kit for sparse matrix computations. 1990.

[46] A Suárez, E Rodríguez, G Montero, MD García, and E Flórez. Precondicionamiento de sistemas de ecuaciones variables.

[47] Gemma Sanjuan, Carlos Brun, Tomas Margalef, and Ana Cortes. Determining map partitioning to accelerate wind field calculation. In *High Performance Computing & Simulation (HPCS), 2014 International Conference on*, pages 96–103. IEEE, 2014.

[48] Gemma Sanjuan, Carlos Brun, Tomàs Margalef, and Ana Cortés. Determining map partitioning to minimize wind field uncertainty in forest fire propagation prediction. *Journal of Computational Science*, pages –, 2016. ISSN 1877-7503. doi: http://dx.doi.org/10.1016/j.jocs.2016.01.006. URL http://www.sciencedirect.com/science/article/pii/S1877750316300060.

[49] MKL Intel. Intel math kernel library, 2007.

[50] Karl Rupp et al. Viennacl. *Web Page: http://viennacl. sourceforge. net*, 2011.

[51] CUDA NVIDIA. Cusparse library. *NVIDIA Corporation, Santa Clara, California*, 2014.

[52] S.R. Bernabeu, V. Puzyrev, M. Hanzich, and S. Fernandez. Efficient sparse matrix-vector multiplication for geophysical electromagnetic codes on xeon phi coprocessors. pages 61–65, 2015. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-84957937424&partnerID=40&md5=2a85145789961308bb112063f848b330. cited By 0.

[53] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *Transactions on Mathematical Software*, 38(1):1 – 14, 2011.

[54] Gemma Sanjuan, Carlos Brun, Ana Cortés, and Tomàs Margalef. *Computational Science and Its Applications – ICCSA 2014: 14th International Conference, Guimarães, Portugal, June 30 – July 3, 2014, Proceedings, Part IV*, chapter Effect of Wind Field Parallelization on Forest Fire Spread Prediction, pages 538–549. Springer International Publishing, Cham, 2014. ISBN 978-3-319-09147-1. doi: 10.1007/978-3-319-09147-1_39. URL http://dx.doi.org/10.1007/978-3-319-09147-1_39.

[55] Pierre-Louis Lions. On the schwarz alternating method. i. In *First international symposium on domain decomposition methods for partial differential equations*, pages 1–42. Paris, France, 1988.

[56] Pierre-Louis Lions. On the schwarz alternating method. iii: a variant for nonoverlapping subdomains. In *Third international symposium on domain decomposition methods for partial differential equations*, volume 6, pages 202–223. SIAM Philadelphia, PA, 1990.

[57] George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.

[58] Olaf Schenk and Klaus Gärtner. Solving unsymmetric sparse systems of linear equations with pardiso. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[59] Jack Dongarra, Andrew Lumsdaine, Xinhiu Niu, Roldan Pozo, and Karin Remington. A sparse matrix library in c++ for high performance architectures, 1994.

[60] Gemma Sanjuan, Carles Tena, Tomàs Margalef, and Ana Cortés. Applying vectorization of diagonal sparse matrix to accelerate wind field calculation. In *Proceedings of the 15th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE 2015*, pages 1011–1022. IEEE, 2015.

[61] Youcef Saad. Sparskit: a basic tool kit for sparse matrix computations - version 2, 1994.