Implementació d'un Algoritme Genètic per millorar les simulacions d'incendis forestals

Sergio A. Jaimes Guzmán

9 de febrer de 2026

Resum – Aquest treball tracta el problema de la predicció d'incendis forestals. Des de fa dècades, s'ha treballat en models de propagació d'incendis que, a partir de dades com la topografia, meteorologia i flora de l'àrea de l'incendi, volen simular el comportament del foc en el temps i l'espai. Un dels simuladors més utilitzats en aquest àmbit és FARSITE. L'objectiu d'aquest treball és millorar la precisió de les simulacions de FARSITE 5.0 fent servir la tècnica d'optimització i cerca anomenada Algoritme Genètic. Implementat en C++, selecciona els paràmetres meteorològics d'entrada més òptims pel simulador segons una funció d'error, que busca l'individu amb el perímetre de foc més semblant. El programa es paral·lelitza fent servir MPI i s'executa tot plegat en entorns de computació d'altes prestacions (HPC). Els resultats mostren millores a la predicció de la propagació del foc, així com una disminució significativa del temps d'execució respecte a la seva versió seqüencial.

Paraules clau – Incendis forestals, Predicció incendis, FARSITE, Algoritme Genètic, funció error, MPI, Computació Altes Prestacions.

Abstract – This work addresses the problem of wildfire prediction. For several decades, research has been conducted on fire spread models that, based on data such as topography, meteorology, and vegetation of the fire area, aim to simulate fire behavior over time and space. One of the most widely used simulators in this field is FARSITE. The objective of this work is to improve the accuracy of FARSITE 5.0 simulations by using the optimization and search technique known as the Genetic Algorithm. Implemented in C++, it selects the most optimal meteorological input parameters for the simulator according to a fitness function that seeks the individual with the most similar fire perimeter. The program is parallelized using MPI and is executed in high-performance computing (HPC) environments. The results show an improvement in fire spread prediction, as well as a significant reduction in execution time compared to its sequential version.

Keywords – Wildfires, Fire prediction, FARSITE, Genetic Algorithm, fitness function, MPI, HPC.

1 Introducció - Context del treball

Els incendis forestals són un problema d'actualitat a tot el món. Representen una de les principals amenaces per als ecosistemes naturals i també són un perill cada vegada més freqüent per a la seguretat de les persones. En aquest context, des de mitjans del segle XX s'han proposat diversos models de predicció de la propagació dels incendis forestals. Aquests models tenen en compte factors com la topografia, meteorologia i la vegetació.

Un dels models de simulació més utilitzats en aquest àmbit és FARSITE. Aquest software permet simular la propagació d'incendis forestals a escenaris reals: a partir la introducció dels paràmetres abans mencionats (meteorologia, topologia i vegetació) genera simulacions de comportament del foc durant un determinat rang de temps. Tot i així, el resultat de les simulacions depèn fortament de la qualitat i el calibratge dels paràmetres d'entrada. La predicció meteorològica no és una ciència exacta i sempre es basa en estimacions. Per tant, la incertesa de les condicions meteorològiques és un obstacle important. Als incendis forestals, la incertesa és molt més alta perquè es genera (a dins del perímetre) un microclima que afecta directament la meteorologia i, com a conseqüència, a les dades que s'han d'introduir al simulador [1]. Aquesta qüestió complica l'obtenció de prediccions més fiables.

• E-mail de contacte: searlogu0910@gmail.com
• Menció: Enginyeria de Computadors
• Treball tutoritzat per: Carles Carrillo Jordan (CAOS)
• Curs 2025/26

D'altra banda, l'Algoritme Genètic (AG) és un mètode d'optimització que es basa en els principis de la selecció natural. L'objectiu és millorar progressivament un conjunt de solucions potencials, de generació en generació, fins a trobar la més adequada per resoldre un problema.

Amb aquesta descripció, es pot deduir que característiques d'aquest mètode són útils per trobar les millors solucions a un simulador. En aquest cas, al simulador FARSITE.

2 OBJECTIUS

2.1 Objectiu general

L'objectiu general d'aquest treball és obtenir, a la versió final, un Algoritme Genètic en C++ que seleccioni els millors paràmetres meteorològics d'entrada al simulador FARSITE 5.0, utilitzant Message Passing Interface (MPI) per executar-lo en entorns de computació d'altes prestacions (HPC) per millorar la precisió de les simulacions respecte FARSITE natiu.

2.2 Objectius específics

Al principi d'aquest treball es van definir 4 objectius específics, organitzats per ordre d'assoliment:

- Aconseguir una major precisió a les simulacions, reduint la discrepància entre els resultats inicials de FARSITE i les dades de referència (incendis del passat).
- Accelerar significativament el procés de simulació, canviant de l'execució seqüencial amb una CPU a la paral·lelització amb MPI fent servir un clúster HPC
- Assegurar escalabilitat a la versió final, permetent a l'usuari generar una determinada quantitat de simulacions en funció del nombre de processadors disponibles al clúster
- Obtenir, al final d'aquest treball, una eina amb el propòsit de ser útil per a la recerca i la gestió d'incendis forestals.

Aquests objectius són adients per quantificar les millores de precisió, velocitat i escalabilitat de la solució respecte al punt inicial del treball.

3 ESTAT DE L'ART

3.1 Orígens del modelatge d'incendis forestals: Rothermel i BEHAVE

Un dels primers models computacionals per incendis forestals es va originar a l'any 1972. Va ser desenvolupat per l'enginyer aeronàutic Richard C. Rothermel i es conegut com el model de propagació de Rothermel. Estableix equacions semi-empíriques per la velocitat de propagació del front de flames basat en 3 grups de característiques: combustible, condicions meteorològiques i topografia.

El model de propagació del foc proposat per Rothermel simplifica el comportament dels incendis forestals transformant-los en un conjunt d'equacions suposades en un entorn idealitzat, on el foc es propaga únicament sobre combustibles morts, homogenis i de petites dimensions situats al sòl forestal. Aquest model no requereix informació detallada sobre les espècies vegetals presents en una determinada àrea, sinó que les considera únicament en funció del tipus de combustible que representen. L'objectiu del model és descriure de manera matemàtica els processos físics i químics que tenen lloc a la propagació del foc [2].

Una dècada després, el model va donar lloc a la creació dels primers sistemes de predicció de propagació i comportament del foc. BEHAVE (1984) va ser el primer sistema de predicció del comportament dels incendis forestals desenvolupat pel U.S. Forest Service. Aquest sistema integrava els subsistemes FUEL, orientat al modelatge de les característiques del combustible, i BURN, destinat a la predicció de la velocitat de propagació, la intensitat del foc i la longitud de la flama. BEHAVE utilitzava el model de Rothermel com a nucli de càlcul, però es trobava limitat a simulacions puntuals perquè no tenia capacitat de càlcul de propagació espacial contínua. Aquest fet sovint obligava a fer ajustos manuals realitzats sobre el terreny, recalculant BEHAVE quan canviaven les condicions meteorològiques, topogràfiques o de combustible [3].

El model de Rothermel continua sent el nucli computacional de la majoria dels simuladors actuals.

3.2 FARSITE

Al 2004, Mark A. Finney va desenvolupar FARSITE (Fire Area Simulator) al Laboratori de Ciències del Foc de Missoula (USDA Forest Service) com un sistema de simulació explícit tant en l'espai com en el temps, capaç d'integrar diversos submodels de comportament d'incendis forestals en una única plataforma. El funcionament del model es basa en el principi de Huygens, aplicat a la propagació del foc, segons el qual cada punt del front d'incendi actua com una font independent de propagació. D'aquesta manera, FARSITE genera perímetres poligonals al llarg d'un interval de temps determinat [4].

La implementació de FARSITE integra quatre models principals del comportament del foc: la propagació del foc de superfície mitjançant l'equació de Rothermel, la propagació del foc a la part superior de la vegetació basada en els models de Van Wagner, l'acceleració de propagació del foc (canvis de velocitat de propagació) i el model del spotting o propagació de brases mitjançant les equacions d'Albini. La sortida del simulador consisteix en perímetres poligonals vectorials que contenen informació sobre la velocitat de propagació, la intensitat de la línia de foc i la longitud de la flama en cada vèrtex. A partir d'aquesta informació, es generen mapes del comportament del foc mitjançant processos d'interpolació [4].

Des de la seva versió inicial el 1998 fins les posteriors revisions, FARSITE ha incorporat millores a la resolució espacial i temporal, així com la integració de dades meteorològiques en funció del temps. El model requereix com a entrada un conjunt mínim de cinc capes: elevació, orientació, pendent, model de combustible i cobertura de les capes superiors de les copes dels arbres. Aquestes capes s'utilitzen per realitzar ajustos topogràfics i realitzar els càlculs per simular la propagació del foc. L'estructura de dades basada en la informació geogràfica (GIS) permet simular incendis

en paisatges heterogenis sota condicions meteorològiques variables, diferenciant-se de simuladors anteriors com BEHAVE, que assumien condicions uniformes [4].

Després de la versió 4.0, FARSITE es va incorporar a l'aplicació FlamMap i es va discontinuar com a eina independent, tot i que encara rep millores com a part de FlamMap.

3.3 Algoritme Genètic

Els algoritmes genètics són mètodes d'optimització i cerca inspirats en els processos de l'evolució d'espècies i la selecció natural, introduïts formalment per John H. Holland a mitjans de la dècada dels setanta [5]. Formen part de la família dels algorismes evolutius i s'han aplicat àmpliament en problemes d'optimització complexos caracteritzats per tenir rangs de cerca extensos, no lineals o amb múltiples òptims locals.

Un algoritme genètic opera sobre una població inicial d'individus, on cada individu representa una possible solució al problema mitjançant una codificació genètica. A cada iteració, o generació, els individus són avaluats mitjançant una funció de fitness (o també coneguda com funció d'error) que mesura la qualitat de cada individu/solució segons els criteris definits a la funció fitness [6]. Els individus amb millor fitness tenen una major probabilitat de ser seleccionats per participar en la generació de noves solucions.

Després d'acabar l'etapa d'avaluació, finalment es seleccionen un nombre determinat dels individus amb les millors fitness, els quals són utilitzats per generar una nova població mitjançant operadors genètics com el encreuament (crossover) i la mutació. El encreuament permet combinar informació genètica de diferents individus amb l'objectiu d'explorar noves regions de l'espai de cerca, mentre que la mutació introdueix variacions aleatòries per mantenir la diversitat genètica dels individus i evitar la convergència prematura cap a solucions subòptimes [7].

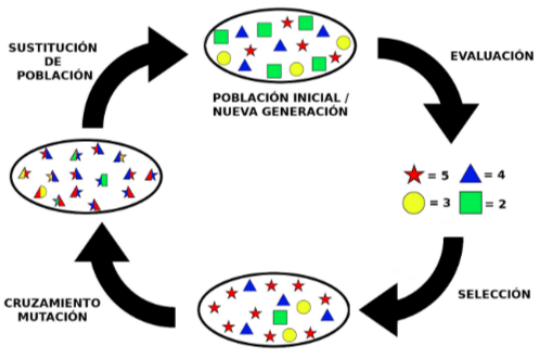

Fig. 1: Diagrama iteratiu de l'algoritme genètic

El procés evolutiu descrit es repeteix durant un nombre determinat de generacions o fins que es compleix un criteri arbitrari, com pot ser l'assoliment d'un valor llindar predefinit o l'estabilització dels valors de la funció de fitness. Gràcies a la seva naturalesa aleatòria i poblacional, els algorismes genètics presenten una elevada capacitat d'exploració, fet que els fa especialment adequats per a problemes on els mètodes deterministes tradicionals de cerca resulten ineficients.

3.4 L'Algoritme Genètic a les simulacions d'incendis forestals

Els simuladors d'incendis forestals depenen de paràmetres els quals la seva estimació exacta es difícil en condicions reals, ja que els factors meteorològics, combustible o la heterogeneïtat de la vegetació són variables amb el pas del temps. En aquest context, els algoritmes genètics (AG) s'utilitzen com a tècniques de cerca i optimització per calibrar automàticament aquests paràmetres.

Un exemple representatiu d'aquesta aplicació es troba a treballs com [1], on s'aplica el paradigma DDDAS (Dynamic Data Driven Application Systems) a la predicció de la propagació d'incendis forestals. Amb aquest paradigma la simulació del foc no es realitza una sola vegada, sinó que es va corregint a mesura que es disposa de noves observacions de l'incendi. En l'esquema DDDAS de tipus Algoritme Genètic (DDDAS-GA), l'evolució real del perímetre de l'incendi s'utilitza directament per recalibrar el sistema, és a dir, per executar el simulador amb informació actualitzada i reduir l'error de predicció.

En termes funcionals, el paper de l'algoritme genètic és actuar com a eina d'ajust: per cada actualització de l'evolució de l'incendi, l'algoritme genera múltiples solucions candidates (conjunts de paràmetres) i executa el simulador amb cadascuna per produir perímetres simulats futurs. A continuació, cada candidat s'avalua comparant la seva sortida amb el perímetre real observat, i utilitzant una funció d'error es seleccionen els millors candidats per crear una nova generació, fent servir el operadors genètics de l'algoritme. Aquest procés iteratiu permet explorar automàticament combinacions de paràmetres òptimes sense dependre d'ajustos manuals i, principalment, adaptar la predicció a les condicions específiques de l'incendi en curs.

Trobem un altre exemple d'aplicació del AG a simuladors d'incendis a l'article [8]. En aquest cas, el simulador és específicament FARSITE, i també es realimenta periòdicament amb dades observades de l'incendi i amb informació meteorològica cada cert temps. Amb aquestes dades i passant per la funció d'evacuació, es recalculen diverses simulacions per obtenir pronòstics més consistents. Així, a partir de la discrepància entre els perímetres simulats i els observats, s'ajusten els paràmetres del model. Una vegada obtinguts els paràmetres corregits, es realitza la predicció del següent interval futur. En aquest cas d'ús però, s'introdueix la dinàmica del vent permet evitar la hipòtesi simplista de "vent constant". Per tant, s'afegeix als paràmetres informació meteorològica com la direcció i la velocitat d'avanç del front de foc.

En aquest treball, l'algoritme genètic comparteix el mateix propòsit del cas d'ús anterior: produir noves generacions de solucions que tendeixen a reduir l'error. D'aquesta manera, s'automatitza la cerca de condicions meteorològiques que expliquen el comportament de l'incendi.

Amb aquests exemples, es pot deduir que l'aplicació de l'algoritme genètic a la predicció d'incendis és una millora significativa. Treballs que recolzen aquesta afirmació, com per exemple [9], expliquen que comparat amb altres tècniques de optimització global (Simulated Annealing, Tabu Search), l'AG es superior tant a la precisió de resultats com a l'escalabilitat paral·lela. No obstant, s'ha de tenir en compte el cost computacional que comporta la seva im-

plementació. El nombre d'iteracions de l'algoritme genètic està directament relacionat amb la qualitat de l'ajust obtingut, però també amb el temps total d'execució. Un nombre més elevat d'iteracions permet assolir una calibració més precisa, però comporta un increment del cost computacional. De manera similar, poblacions d'individus més grans tendeixen a convergir en menys iteracions cap a solucions adequades, però requereixen una major disponibilitat de recursos computacionals per poder avaluar tots els individus en paral·lel.

3.5 MPI

Message Passing Interface (MPI) és un estàndard de programació paral·lela basat en el pas de missatges, al qual la informació es transfereix entre diferents processos mitjançant operacions coordinades per les dues partes. A més del model clàssic, l'estàndard incorpora extensions per operacions col·lectives, accés remot a memòria, creació dinàmica de processos i mecanismes d'entrada/sortida paral·lela [10]. El propòsit principal de una implementació MPI a un programa és habilitar la paral·lelització del còmput en nodes amb el seu propi processament, memòria i emmagatzemament, és a dir en un sistema distribuït. Com a conseqüència, es pot aconseguir una reducció significativa del temps d'execució.

MPI és necessari per desenvolupar implementacions HPC basades en el paradigma Master/Worker, amb l'objectiu d'augmentar la capacitat de còmput i reduir els temps d'execució que comporten les diverses simulacions. A la fase inicial, el procés Master o procés 0, crea la població inicial de solucions i la distribueix entre els processos Worker. Posteriorment, els Workers executen les simulacions d'individus en paral·lel (El Master pot participar-hi també depenent del disseny) i calculen el valor de la funció de fitness corresponent [11].

4 METODOLOGIA

Per assolir els objectius definits, es va adoptar una metodologia de treball de tipus incremental i iterativa. Aquesta metodologia consisteix en la implementació progressiva de cadascuna de les tasques del projecte, seguida immediatament de la seva validació. Es continua aplicant de manera cíclica i progressiva a al llarg de totes les etapes o fites del treball. Aquesta metodologia garanteix que el procés de treball sigui fiable i ordenat.

El treball es va dividir en 2 tasques principals:

- Implementar una versió seqüencial en C++ de l'Algoritme Genètic. La implementació principal havia d'anar precedida de la creació d'un algoritme per generar els primers N individus. Una vegada fet, es va procedir a implementar el bucle de l'algoritme genètic prenent les decisions de disseny corresponents. Per acabar, es va afegir una simulació amb el millor individu per executar una simulació calibrada amb l'objectiu de millorar la precisió.

- Implementació de directives MPI per paral·lelitzar l'AG Seqüencial. Primer, s'havien de fer adaptacions al codi original pel correcte acoblament de MPI al AG. També es tenia previst fer ajustos al codi per garantir l'escalabilitat del programa. Una vegada solucionats aquests 2 problemes, es podien implementar les directives MPI necessàries.

La validació de cada tasca es va fer de dues maneres:

- A les tasques on es va treballar l'algoritme genètic, es va fer una validació qualitativa mitjançant la comparació amb dades de referència (perímetres d'incendis ja coneguts).

- A les tasques on es va treballar la implementació paral·lela, es va fer una validació quantitativa mitjançant mètriques com el temps d'execució, Speedup i l'escalabilitat en funció del nombre de processos.

Els recursos que es van fer servir per desenvolupar aquestes tasques són els següents:

- El programa base de FARSITE, la versió de Linux.

- Un cas d'incendi real, digitalitzat i formatat per FARSITE. Sobre aquest cas s'aplicarà l'algoritme genètic.

- El programa Visual Studio Code.

- El clúster Wilma de la UAB.

El clúster Wilma un és un clúster dedicat que està format per 12 nodes. cada nodé té 2 processadors AMD Opteron 4180 de 6 nuclis (6 cores * 2 CPU = 12 cores lògics). Té una memòria de 128 GB pel 6 primers nodes i de 16 GB per la resta. El clúster utilitza Linux com a sistema operatiu i SLURM (Simple Linux Utility for Resource Management) com a planificador de tasques. Per tant, s'han de fer les execucions mitjançant scripts que s'envien a una cua per ser executats.

5 IMPLEMENTACIÓ ALGORITME GENÈTIC

5.1 Generador de la població

El procés de implementació va començar per crear el generador de la població de l'algoritme genètic. El simulador FARSITE s'executa a partir de la lectura de diversos fitxers de dades. Per executar el simulador, llegeix un fitxer en format text que recull tota la configuració i les rutes als fitxers d'entrada necessaris. Els fitxers més rellevants són un mapa de la vegetació de la zona de l'incendi, un mapa topogràfic de l'àrea de l'incendi i un fitxer input amb dades. El fitxer input conté dades de combustió de la vegetació de l'àrea de l'incendi, dades topogràfiques com l'altura sobre el nivell del mar i dades meteorològiques a l'incendi. Pel que respecta a l'àmbit d'aquest treball, ens interessen crear múltiples individus a partir de la modificació de, únicament, dos fitxers: el fitxer de text per executar el simulador i el fitxer `.input`. Al fitxer input, només es modifiquen les línies de dades del temps i les de vent perquè la meteorologia és el paràmetre que té més impacte a la propagació del foc [4].

Cada línia de dades del temps al fitxer `.input` segueix el format:

```
[Mes] [Dia] [Precip.]
[Hora-Tmin] [Hora-Tmax]
[Tmin-(°F)] [Tmax-(°F)] [HRmax]
[HRmin]
```

Les dades de vent segueixen el format:

```
[Mes] [Dia] [Hora]
[Velocitat-(mph)] [Direcció]
[Núvols]
```

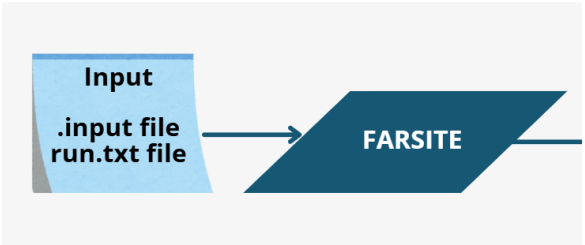

Fig. 2: Diagrama dels fitxers d'entrada per individu

Mitjançant la llibreria `fstream` de C++, es pot processar el fitxer input que conté els paràmetres capturats originalment i extreure els seus valors per ser modificats. La intenció del AG és que es generin N individus diferents entre ells, llavors s'han de generar valors aleatoris diferents dins d'uns rangs per mantenir la lògica. El generador només modificarà dades de meteorològiques i de vent, deixant intactes les dades topogràfiques i de combustible perquè queden fora de l'abast d'aquest treball. Els valors limit de rang de cada paràmetre a modificar s'han assignat arbitràriament, seguint una lògica de valors realistes i tenint en compte que els valors mínims no poden ser majors als màxims i viceversa. Finalment, el flux del procés del generador consisteix en llegir i capturar les línies amb dades del temps i de vent, canviar els seus valors aleatòriament dins dels límits establerts i escriure-ls en un fitxer amb format idèntic juntament amb el fitxer de text necessari per passar la ruta dels fitxers necessaris per cada individu. A la fig. 3 es pot veure una representació gràfica.

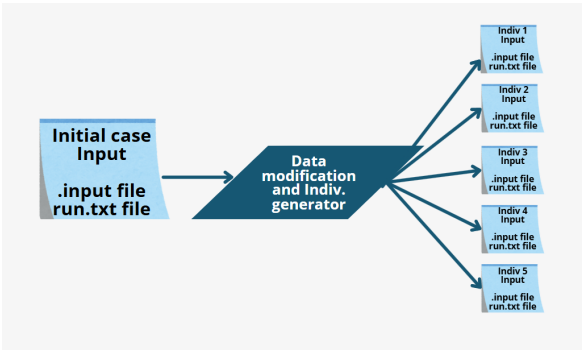

Fig. 3: Diagrama explicatiu del generador

5.2 Bucle evolutiu

El següent pas va ser dissenyar el bucle evolutiu. Una vegada generada la població inicial, es simula FARSITE per cadascun dels individus. La simulació genera diversos fitxers de sortida per cada individu. Per tant, el primer que fa el programa és obtenir la ruta per guardar els fitxers de sortida a una carpeta individual per cada solució (generada abans amb el programa generador). El següent és executar una comanda de sistema per executar FARSITE. Una vegada acabada la simulació de l'individu, es generen 2 fitxers necessaris per calcular la fitness: el fitxer amb el perímetre de l'incendi en format poligonal i un fitxer amb el temps d'arribada (Time of Arrival) del foc a cada punt de l'àrea de l'incendi. Una vegada carregat algun dels dos fitxers, es pot calcular la funció d'avaluació del AG. Al següent subapartat s'explica amb més detall el disseny de la funció fitness.

Després de calcular l'error de cada individu, es fa servir una estructura anomenada `indiv` per emmagatzemar l'identificador de cada individu i el valor de la seva puntuació fitness. Tots els resultats individuals es guarden a un vector i s'ordenen de major (més semblança al perímetre observat) a menor. Els primers N individus s'utilitzaran per crear la següent generació. El nombre d'individus seleccionats per passar a la següent generació serà arbitràriament decidit per l'usuari. Aquestes possibles solucions es converteixen en pares de la següent generació i passen a la fase de encreuament. Es creuen a partir d'una mascara de 32 bits generada cada vegada que els pares s'acaben de creuar per parelles entre tots ells. Si el valor del bit es 0, agafarà el component genètic del pare 0 i si és 1, del pare 1. El procés continua fins haver generat el mateix nombre d'individus de la població inicial.

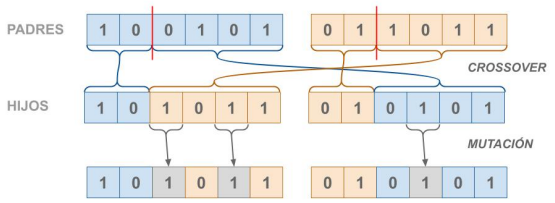

Fig. 4: Exemple visual de l'encreuament i mutació. En aquest diagrama no es fa servir mascara

El següent pas a l'algoritme és la fase de mutació, on les dades meteorològiques de la nova població es tornen a alterar aleatòriament. En aquest cas, però, s'apliquen límits molt més restrictius, ja que es vol mantenir l'estabilitat de les millors solucions obtingudes i evitar desviacions excessives respecte a la fitness dels individus "pares". Per exemple, si per modificar la temperatura màxima a la generació de la població inicial es fa servir `TMax ± 6`, per modificar aquest paràmetre a la mutació es fa servir `TMax ± 2`. Un cop acabada la mutació dels individus, es tanca una iteració del AG i torna a començar el bucle passant a simular i avaluar la 2a generació.

Així doncs, és com es passa per totes les fases de l'algoritme genètic: avaluació, selecció, encreuament i mutació. El bucle AG es para quan s'acaben de simular totes les generacions que l'usuari hagi determinat o bé perquè el valor de la fitness d'un individu ha superat el valor llindar, el que vol dir que és bastant acceptable com a solució.

5.3 Funció de fitness

La funció d'avaluació és la part més decisiva de l'algoritme genètic. Si la funció no és adequada, el resultat de l'algoritme no és optim i perd la seva eficàcia. Tampoc n'hi ha un procediment únic per calcular-la: el càlcul de la fitness o del error depèn del desenvolupador i del problema a tractar amb el AG. Al cas d'ús d'incendis forestals i de FARSITE, segons la literatura revisada, el consens és que la millor forma de tractar aquest problema es mitjançant l'avaluació de

la comparativa entre el perímetre simulat i el perímetre real observat de l'incendi. Tot i així, també n'hi ha diverses formes de dissenyar aquesta comparativa. En aquest treball s'han desenvolupat 2 funcions diferents pel càlcul de la fitness:

- La primera compara els polígons generats per la solució candidata i el perímetre observat plasmat en un polígon GIS. Aquesta comparació es fa mitjançant la llibreria GDAL/OGR, i es calcula fent servir la fórmula Intersection over Union (també coneguda com Índex de Jaccard) (1).

- La segona compara els fitxers que contenen el Time of Arival (ToA) dels 2 perímetres. En el cas del perímetre observat, no tenim informació de quan arriba el foc a un determinat píxel de la projecció al mapa, però rasteritzant el fitxer del polígon (en format `.shp`), podem saber si a cada cel·la ha arribat el foc o no. Amb aquesta informació, podem comparar punt per punt l'extensió del foc dels 2 i calcular la fitness fent servir la formula Intersection over Union (IoU) (1).

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

El disseny de la primera proposta és el següent: fent servir les funcions de la llibreria GDAL/OGR disponible per C++ [12], es converteix el perímetre observat emmagatzemat a un fitxer `.shp` en un únic polígon de 2 dimensions. Després es fa el mateix amb el perímetre simulat, tot i que en aquest cas és una operació més costosa ja que el fitxer inicial està compost per més de 100 capes que s'han de fusionar amb funcions de la llibreria OGR. una vegada obtingut els polígon fusionat, es procedeix al càlcul de la fitness fent servir les funcions Intersection i Union per obtenir l'àrea que tenen en comú i l'àrea que no coincideix. Es calcula el valor de les 2 àrees i s'aplica la fórmula. La funció retorna el resultat final i es guarda al vector de individus.

El disseny de la segona proposta en comptes de fer servir els fitxers `.shp`, fa servir els fitxers `.asc` que contenen una matriu de valors on representa tots els píxels de la projecció al mapa del perímetre de l'incendi. El procés requereix la conversió del perímetre observat a tipus ASC (ASCII). Es pot fer mitjançant la comanda `rasterize` de GDAL. Pels perímetres simulats, FARSITE genera un ASC per cada simulació. El següent pas és guardar el valor de cada pixel (tant del perímetre simulat com de l'observat) a un vector. Al cas de que no hagi arribat el foc, el valor d'aquest punt serà -9999.0, que correspon al predeterminat per NO DATA. Si el foc sí que s'ha propagat fins aquest punt, es valor serà diferent a NO DATA. Es recorre punt per punt els 2 vectors a la vegada. Si coincideixen en tenir valors diferents a -9999.0, és un punt d'intersecció i unió (als 2 casos el foc ha arribat al punt comparat). Si només a un dels dels 2, és un punt d'unió. Una vegada obtingut el nombre total de cel·les on ha arribat el foc als 2 perímetres i el nombre de cel·les on ha arribat el foc d'algun dels dos perímetres, dividim els 2 valors (IoU) i obtenim el valor fitness d'aquella solució. la funció retorna el resultat final i es guarda al vector de individus.

La decisió sobre quina de les 2 funcions d'avaluació es farà servir l'haurà de fer l'usuari. Tindrà l'opció de executar el programa amb ambdues opcions però no a la vegada.

Més endavant, a la secció de resultats i de conclusions, es fa un anàlisi sobre quina de les dos opcions és més recomanable.

5.4 Simulació post-calibratge amb el millor individu

Una vegada calculat el millor individu i els paràmetres meteorològics que expliquen millor el perímetre observat hem acabat la fase que anomenarem fase de calibratge. Ara, la idea és veure l'efecte que té aquest procés a les simulacions de FARSITE. Una de les decisions més importants que es va prendre durant la fase de disseny del programa és fer servir la metodologia de 2 etapes, explicada en més d'un exemple a la literatura revisada [1] [8]. Aquest mètode primer aplica l'algoritme genètic amb l'últim perímetre de l'incendi visualitzat per després, a la següent etapa, predir la seva evolució a partir de les ultimes dades meteorològiques calibrades. Això significa que hem de fer una simulació amb el millor individu, suposant que les últimes dades meteorològiques calibrades es mantenen constants fins al temps limit de la simulació. Com que en aquest cas no es compta amb un DDDAS, haurem de permetre l'usuari entrar manualment l'últim perímetre de referencia vist de l'incendi, o bé el perímetre que vulgui provar per experimentar amb el simulador.

6 IMPLEMENTACIÓ PARAL·LELA

6.1 Adaptacions al codi

Previ a la implementació MPI, s'havia de fer adaptacions al codi seqüencial per que MPI funcioni correctament. Tot i que MPI permet paral·lelitzar el treball computacional d'un bucle en CPUs independents (com un sistema distribuït), cada CPU fa servir la seva memòria local. Si no adaptem les iteracions dels bucles a la porció local que volem assignar a cada procés, cada procès faria el nombre d'iteracions totals i canviaria el comportament del programa. Per això, hem de modificar els 2 bucles d'avaluació: el primer per la primera generació, i el segon per la simulació de les següents generacions.

Pel bucle de la primera generació, es defineix `int i = MPIRank + 1; i <= numIndiv; i +=MPISize` que permet repartir equitativament les iteracions del bucle entre els processos MPI disponibles (`MPIRank` correspon al identificador d'un procés MPI i `MPISize` al total de processos MPI disponibles). Tal i com es va explicar al disseny de l'algoritme genètic, per cada generació passen un nombre determinat dels millors individus a la següent. D'aquests no cal tornar a repetir la simulació perquè ja sabem la seva puntuació fitness. Per tant, comencem a avaluar els individus des de la posició del primer individu "fill". Per aquest motiu, simplement s'afegeix a l'inici de la declaració del bucle el nombre d'individus (`NIndiv`) que passen a la següent generació amb l'objectiu de començar a iterar després de les posicions on estan guardats aquests al vector: `int i = NIndiv + MPIRank + 1; i <= numIndiv; i += MPISize`. Amb aquestes adaptacions, ja es poden començar a implantar les directives MPI.

6.2 Disseny MPI i escalabilitat

Per explicar el disseny de la implementació MPI, la millor forma és explicar primer perquè és necessari. Al executar la versió seqüencial, es pot veure que el coll d'ampolla del programa es la simulació dels incendis a FARSITE. Llavors, l'objectiu principal és paral·lelitzar les simulacions de FARSITE. A més, podem deduir que el càlcul de la funció d'avaluació és independent entre processos, ja que no es necessita cap informació fora del procés local per a què un procés calculi el seu valor fitness. Amb aquests dos punts, ja es pot plantejar una proposta adequada per paral·lelitzar el programa, amb l'estratègia de paral·lelisme de tasques.

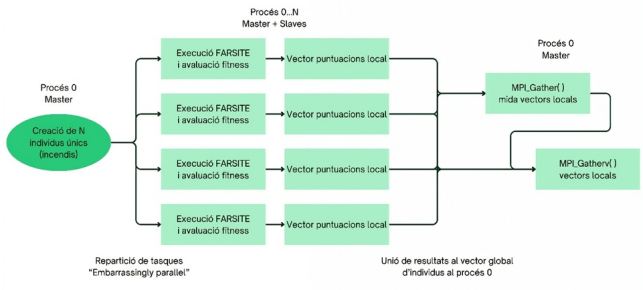

Fig. 5: Diagrama del disseny MPI Implementat

Com es mostra a la fig. 5, una vegada creada la primera generació s'assigna a tots els processos disponibles un individu per simular a FARSITE. Una vegada acabada la simulació, cada procés s'encarrega de carregar els seus fitxers `.shp` o `.asc` corresponents i calcular el seu valor de fitness. Un cop obtingut el valor de la fitness, guarda el valor dins del vector d'individus juntament amb el seu identificador (per saber de quina solució es la millor puntuació).

Si el nombre de processos no és igual al nombre d'individus, quedaran individus per simular. Per tant, amb el disseny de bucle proposat a l'apartat anterior, els processos continuaran fent simulacions fins que s'hagin cobert totes les solucions. Degut a que una implementació on es busqui processos inactius per assignar-lis simulacions és complicada i pot comportar condicions de carrera, es va descartar aquesta opció. També es descarta repartir el nombre d'individus entre el nombre de processos per divisió directa, perquè donaria lloc a reparticions desequilibrades. El primer procés o rank faria moltes més simulacions que els altres si la divisió no és entera. Per això, es va decidir que la solució més eficient era la següent: cada procés MPI té assignada una simulació per identificador i la següent serà `MPIRank + MPISize`. D'aquesta forma, el balanceig de carrega és més acceptable.

Tot i així quedava un gran inconvenient per solucionar. La funció de comunicació col·lectiva més eficaç per ajuntar els valors fitness calculats per cada procés, `MPI Gather`, no permet ajuntar els blocs de dades de tots els processos si no són de la mateixa mida. Precisament, aquest inconvenient passa si el nombre de individus no és exactament divisible entre el nombre de processos. Per tal de no limitar la funcionalitat del programa, es va decidir buscar alternatives. La decisió final va ser utilitzar un comptador de la mida dels vectors parcials amb els resultats per procés. Amb aquest valor, es pot fer servir la directiva `MPI Gatherv`, que sí que permet ajuntar dades de diferents mides per procés, però abans coneixent la mida (en bytes) de les dades que rebrà de cada procés. Com es sap la quantitat de elements per procés i la mida de l'estructura del vector (amb la funció `sizeof(indiv)`), es pot calcular la mida i passar exitosament tots els resultats parcials al vector global que porta el procés Master.

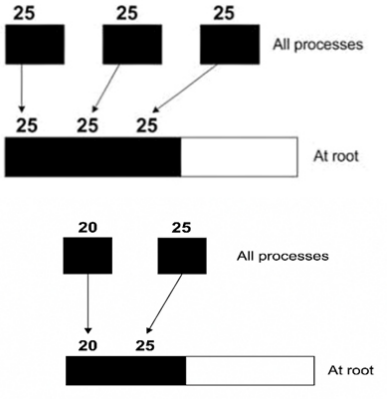

Fig. 6: Exemple comparatiu de `MPI Gather` (a dalt) i `MPI Gatherv` (abaix)

Quan el procés Master rep i emplena el vector global d'individus amb els resultats locals, ordena els individus de millor a pitjor fitness, com es va explicar abans. Aquesta feina continua sent exclusiva del Master per evitar inconsistències al resultat. Es torna a repetir el mateix procés (`Simulacions paraleles + MPI Gather + MPI Gatherv`) per avaluar les següents generacions fins al final de AG. Les fases d'encreuament i mutació també són executades exclusivament pel procés Master. També es fan servir dos funcions més de MPI: la funció `MPI Bcast` per que el procés Master pugui enviar a tots els processos la solució amb millor fitness i veure si el valor d'aquest supera el llindar, i la funció `MPI Barrier` per evitar que els processos worker comencin a executar simulacions de FARSITE abans que el Master hagi acabat l'encreuament i la mutació de la nova població.

En quant a l'escalabilitat del sistema, amb l'estratègia de repartició de simulacions implementada, no hauria de haver problemes en escalar a qualsevol xifra entera de simulacions. El bucle reparteix de la manera més equitativa possible qualsevol nombre enter d'individus. La idea es que l'usuari només hagi de tenir en compte els recursos de còmput amb els que compta.

7 Resultats

7.1 Resultats qualitatius

El resultat del treball, un cop finalitzat, és un programa capaç de simular un nombre d'individus i escollir quina és la millor solució per executar una simulació ja precalibrada. El programa dona l'opció a l'usuari d'escollir 2 maneres de calcular el valor de la funció fitness: la comparació geomètrica dels polígons i la comparació amb el temps d'arribada del foc a una cel·la de la projecció al mapa. El codi final segueix el diagrama de flux de la fig.7, on la part d'execució paral·lela són les fases de simulació i avaluació, que són el coll d'ampolla de la versió seqüencial.

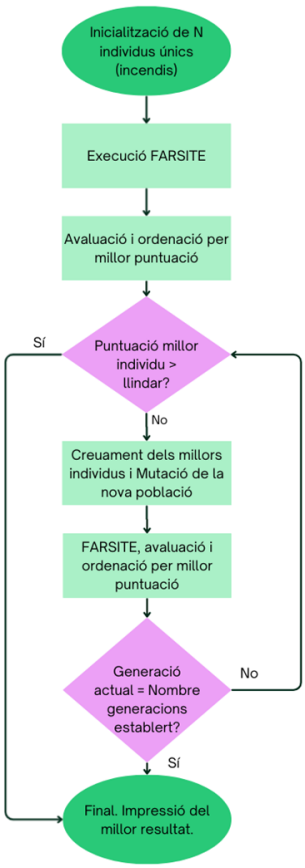

Fig. 7: Diagrama de flux del AG

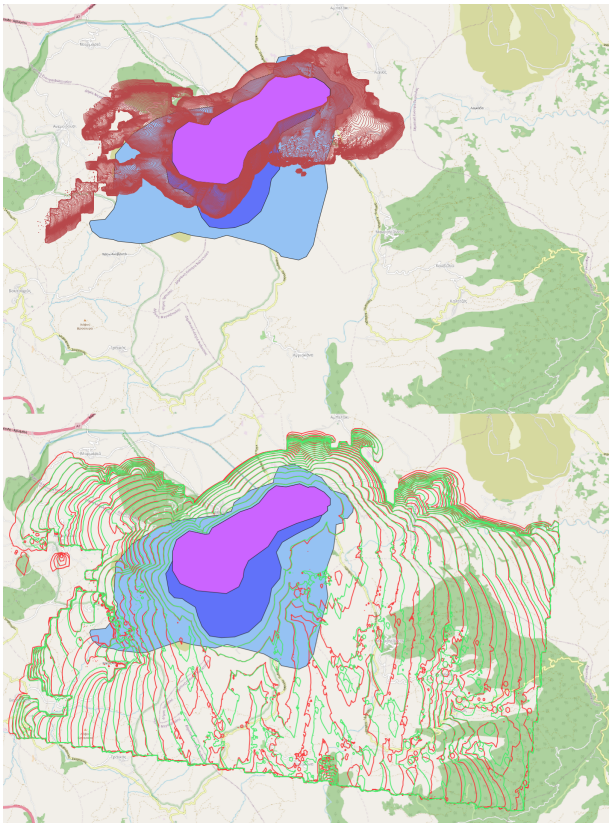

Fig. 8: Comparació de simulacions: simulació de 23 hores d'incendi sense AG a dalt. Simulació amb AG-GDAL/OGR (vermell) i simulació amb AG-Time of Arrival (verd) abaix

Tal i com es va veure a la literatura relacionada amb aquest problema, la millor forma per tractar el problema de l'avaluació és comparar el perímetre simulat i el perímetre observat. En aquesta solució també s'ha optat per seguir aquesta estratègia. També es va decidir fer servir el paradigma Master/Worker proposat als treballs revisats prèviament [11], tot i que en aquest cas també s'utilitza el node Master per fer simulacions. Cada procés que participa a l'execució s'encarrega d'un nombre de simulacions de FARSITE. El resultat també compta amb un breu programa que genera tots els individus de la primera generació, que si l'usuari vol, pot ajuntar dins del script d'execució SLURM per que es faci automàticament abans de executar l'AG.

Per validar la qualitat del resultat de l'algoritme, s'han de fer proves d'execució del programa. Es van fer simulacions de 250 individus i 15 generacions, utilitzant 36 processos MPI distribuïts en 3 nodes de Wilma. Comparant les solucions abans d'implementar l'AG i després d'implementar l'AG, es poden veure resultats mixtos però satisfactoris: A la fig. 8, es pot observar que el perímetre simulat utilitzant com a funció de fitness la comparació de perímetres mitjançant GDAL/OGR (àrea vermella) es propaga en una direcció més realista a comparació del cas inicial sense algoritme genètic. Quan la funció de fitness es basa en els temps d'arribada del foc a cada punt de la projecció sobre el mapa (àrea verda), el perímetre simulat presenta un resultat similar. Tot i així, podem veure als 2 casos una evident sobreestimació del perímetre final. Després de provar amb més simulacions i revisar els paràmetres calibrats resultants, es va arribar a la conclusió que aquesta sobreestimació es presenta per 2 causes principals:

- La fórmula d'error Intersection over Union tendeix a seleccionar individus que sobreestimen el perímetre final: Si comparem 2 individus presents en una simulació del AG-TOA (Time of Arrival), un individu que obté 538 punts d'intersecció i 794 d'unió obté una pitjor puntuació fitness que un amb 575 punts d'intersecció i 831 d'unió (0.6775 vs 0.6919). Si calculem els punts de discrepància entre la intersecció i la unió, veiem que són els mateixos: 256 punts. Per tant, surt més afavorit un perímetre més gran tot i tenir el mateix nombre de "errades" a la simulació.

- L'exemple de simulació (incendi a Arcàdia, 2011) presenta un primer perímetre observat a les 9:43 del 26 d'agost. Després, el perímetre de calibratge es va capturar a les 11:27, 1 hora i 45 minuts després. L'ultim i més gran (en turquesa) és el perímetre de simulació, capturat a les 8:49 del dia següent. Per tant, la simulació és de 21 hores i 22 minuts després. Com vam explicar al disseny, el mètode de 2 fases suposa la persistència de les últimes condicions meteorològiques calibrades. Per tant, s'està suposant que les condicions meteorològiques que expliquen el perímetre de calibratge es mantenen constants 21 hores, fet poc probable. Aquesta metodologia de simulació és millor si es disposen de més dades de calibratge al llarg del incendi i no només una. A més temps de simulació, més incertesa. Per demostrar aquest segon punt, a la fig. 9 es pot veure una simulació post-calibratge de 2 hores. Comparant el resultat simulat del AG-TOA amb el cas inicial sense AG, es veu que la sobreestimació baixa i

que l'error de simulació és pitjor al cas sense AG (que s'expandeix en direcció contrària) que al cas calibrat. Així doncs, el resultats de les simulacions calibrades s'ajusten a les dades de calibratge i al temps de simulació.

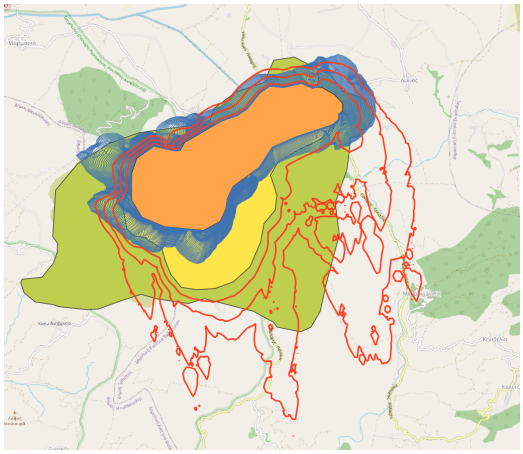

Fig. 9: Comparació de simulacions 2 hores després del calibratge: simulació amb AG (Time of Arrival) en vermell i simulació sense AG en blau

Les condicions que expliquen l'expansió de l'incendi durant les primeres 2 hores corresponen a ratxes de vent moderat/fort en direcció sud-est. Si al següent perímetre de calibratge es veu menys dispersió en aquesta direcció (com va passar realment), l'AG ajustarà la simulació a velocitats més baixes de vent i per conseqüència, a un perímetre amb menys sobreestimació.

Respecte la comparació entre les dos maneres de calcular la funció de error, es pot observar que les dos obtenen resultats qualitativament similars. Això ens demostra que no hi ha diferencies qualitatives significatives entre els resultats dels dos mètodes i que es pot fer servir qualsevol dels dos per arribar a una solució acceptable.

7.2 Resultats quantitatius

Igual que amb el resultats qualitatius del programa, per validar quantitativament la solució s'han de veure els resultats de rendiment de la implementació MPI respecte la seqüencial i també comparar entre les dos versions propostes per l'AG. A la següent taula es mostren les mètriques corresponents a cada cas (on TOA correspon a Time of Arrival i polígons a la funció que compara polígons amb GDAL/OGR):

TAULA 1: COMPARACIÓ DE RENDIMENT

Versió	Temps Execució	Speedup
Fitness polígons sèrie	32 min 53 s	-
Fitness polígons + MPI	4 min 37 s	7,12
Fitness TOA sèrie	30 min 10 s	-
Fitness TOA + MPI	4 min 56 s	6,12

Com es pot veure a la taula comparativa, el temps d'execució es redueix satisfactòriament a la versió final paral·lela de les 2 versions del programa (36 processos) en comparació a les versions sèrie. Per tant es pot assegurar que es compleix l'objectiu de reducció del temps d'execució. Tots els 4 cassos s'han executat amb 180 individus, simulats a 8 generacions. També cap dir que, tot i que el Speedup és significatiu, està limitat per sincronització. La paral·lelització d'aquest procés comporta aplicar regles de sincronització entre processos. En una tasca computacionalment costosa com és fer simulacions de FARSITE, és normal que el rendiment de la sincronització vegi afectat, ja que en casos on els individus no siguin exactament divisibles entre el nombre de processos, hi haurà processos MPI que tindran més treball computacional que altres. El mateix si una de les simulacions s'allarga més que les dels altres processos. Es per això que tampoc s'esperava un Speedup proper al valor màxim possible de `Nprocesos` $\approx$ 36 (segons la llei d'Amdahl). Les directives `MPI Gather, Gatherv, Bcast i Barrier` són bloquejants, per tant tenim 4 punts de parada on cada procés ha d'esperar l'arribada dels altres per continuar l'execució.

Pel que fa la comparació entre els dos modes de calcular la funció d'error o fitness, es pot observar que les 2 versions paral·leles triguen un temps similar en aquest cas de simulació (incendi d'Arcàdia). Cal comentar que s'han observat casos en els que, si el polígon que forma l'incendi de calibració és relativament gran, el mètode de càlcul de fitness que fa servir GDAL/OGR és més lent que el que fa servir els temps d'arribada del foc. Això s'explica amb el temps que requereix unir totes les capes que conformen la figura poligonal en una per poder fer la comparació IoU. Moltes simulacions de FARSITE acaben generant perímetres amb cents de capes MultiLineString que s'han de fusionar per poder se tractades com a polígon. Aquest procés triga un temps i provoca un retard que no es produeix al cas de la comparació amb els TOA, ja que en aquest cas simplement es compara valors decimals dins d'un bucle. Tot i així, aquests casos són molt concrets i amb els temps de la taula (validats amb diverses simulacions) es pot concloure que, als casos d'ús normals, Es pot fer servir qualsevol dels 2 mètodes d'avaluació obtenint resultats en temps molt acceptables.

Respecte a l'escalabilitat del programa, executant diverses combinacions de individus i processos es demostra que és correcta. L'algoritme ha funcionat amb normalitat amb les següents combinacions d'individus i processos MPI:

- 50 individus per 16 processos MPI.
- 100 individus per 36 processos MPI.
- 180 individus per 36 processos MPI.
- 250 individus per 36 processos MPI.

En totes les combinacions, el programa respon satisfactòriament i dona un resultat vàlid.

8 CONCLUSIONS

Aquest treball ha reforçat la idea de que l'algoritme genètic és una solució adequada al problema d'incertesa dels paràmetres meteorològics. També ha aportat un exemple d'aplicació de l'algoritme genètic al simulador FARSITE, concretament per la versió 5.0. Per altre banda, s'ha aconseguit aportar explícitament formes de calcular la funció de fitness o error, especialment fent servir el mètode de comparacions entre els polígons que formen el perímetre, que

és una opció poc comuna. En general, s'ha aconseguit entregar una eina útil per trobar els paràmetres meteorològics que expliquen l'expansió i el comportament de un perímetre del foc observat. Amb això també podem dir que el treball realitzat compleix el quart objectiu que s'havia marcat: s'ha entregat una eina útil per a la recerca i la gestió d'incendis forestals.

Tot i així, tal i com s'ha anat explicant durant el document, hi ha punts de millora a la solució global que s'ha implementat. S'ha parlat de possibles millores a l'eficiència de la execució paral·lela i als resultats de predicció que resulten en sobreestimació. També es va trobar un lleu problema al disseny d'una part del programa: a la funció d'avaluació que compara els polígons, la llibreria GDAL/OGR pot generar detectar com invàlids alguns polígons (perímetres) generats per FARSITE. La conseqüència d'aquest error és que al fusionar les capes que conformen el perímetre simulat, algunes figures no poden generar correctament el polígon unificat. Això provoca la pèrdua de la figura original i una baixa puntuació fitness, que acaba descartant aquest individu. L'impacte d'aquest problema és mínim (s'ha detectat només en una o dos execucions de desenes) però seria interessant de revisar.

8.1 Possibles línies de millora

Les possibles millores o línies de continuació que es poden fer a partir d'aquest programa són diverses. Algunes sorgeixen com a resposta pels problemes lleus trobats, altres es poden plantejar a partir dels resultats del programa:

- Cercar fórmules de càlcul de la fitness o error més complexes que la Intersecció sobre Unió, amb l'objectiu de corregir l'esbiaix detectat pels perímetres que sobreestimen la simulació final.
- Explorar si n'hi ha un rang de valors per cada paràmetre meteorològic que sigui més justificable (a partir d'estudis de la temperatura als incendis forestals, de la humitat o de corrents de vent, per exemple). El mateix es podria aplicar per l'etapa de mutació dins l'algoritme genètic.
- Provar aquest simulador amb casos on la distancia de simulació sigui menor, de forma que pugui ser útil per la predicció d'evacuacions de poblacions.
- Cercar alternatives de disseny MPI que redueixin l'espera i sincronització entre processos, com la possibilitat d'incorporar clàusules no bloquejants.
- Per últim, prenent com a referencia la literatura consultada i els resultats, el programa seria encara més útil si s'incorporés el paradigma DDDAS, que permetria ajustar periòdicament els paràmetres meteorològics de calibratge i s'obtindrien prediccions més fiables i precises de la propagació del foc [1].

Ús de la IA Generativa

Per aquest informe s'ha fet servir el model d'IA generativa GPT-5/5.2, OpenAI exclusivament per millorar la claredat de redacció del document i per corregir el llenguatge massa informal o faltes ortogràfiques. També per cercar bones fonts bibliogràfiques per l'explicació teòrica de l'algoritme genètic.

Per la realització del treball pràctic, s'ha fet servir també GPT-5/5.2 com a suport per comprendre el funcionament i la sintaxi de les funcions de la llibreria GDAL/OGR, degut a que la quantitat de documentació i tutorials per C++ és molt limitada. També per validar la correcta implementació de la clàusula `MPIGatherv`, perquè es troben pocs casos d'ús (explicats) amb aquesta clàusula MPI.

Agraïments

Aprofito per agrair a aquelles persones que han fet possible la realització d'aquest treball. En primer lloc, voldria agrair als meus pares tot l'esforç que han fet per arribar a aquest moment, així com tots els consells durant el camí, que m'han allunyat de decisions que avui hauria lamentat.

També vull agrair al meu tutor, Carles Carrillo, pels seus consells i la supervisió durant la realització d'aquest treball. He tingut la sort de tenir un tutor que domina molt el tema exposat en aquest article. El treball no hauria tingut la qualitat necessària sense la seva experiència i punts de vista.

Referències

[1] T. Artés, A. Cardil, A. Cortés, T. Margalef, D. Molina, L. Pelegrín and J. Ramírez, "Forest Fire Propagation Prediction Based on Overlapping DDDAS Forecasts," Procedia Computer Science, vol. 51, pp. 1623–1632, 2015, doi: 10.1016/j.procs.2015.05.294

[2] G. Wells, "The Rothermel Fire-Spread Model: Still Running Like a Champ," Fire Science Digest, no. 2, Joint Fire Science Program, 2008.

[3] D. Weinstein, "Fire growth modeling in an integrated GIS environment," in Proceedings of the 1995 ESRI User Conference, Palm Springs, CA, USA, May 22–26, 1995. https://library.esri.com/docs/proc95/to100/p092.html (accessed Jan. 15, 2026).

[4] M. A. Finney, "FARSITE: Fire Area Simulator—Model Development and Evaluation," U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, Research Paper RMRS-RP-4, Mar. 1998 (revised Feb. 2004), doi: 10.2737/RMRS-RP-4.

[5] J. H. Holland, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence," Cambridge, MA, USA: MIT Press, 1992.

[6] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Reading, MA, USA: Addison-Wesley, 1989.

[7] M. Mitchell, "An Introduction to Genetic Algorithms," Cambridge, MA, USA: MIT Press, 1998.

[8] C. Brun, T. Artés, T. Margalef and A. Cortés, "Coupling Wind Dynamics into a DDDAS Forest Fire Propagation Prediction System," Procedia Computer Science, vol. 9, pp. 1110–1118, 2012, doi: 10.1016/j.procs.2012.04.120.

[9] A. Cencerrado, A. Cortés, and T. Margalef, "On the Way of Applying Urgent Computing Solutions to Forest Fire Propagation Prediction," Procedia Computer Science, vol. 9, pp. 1657–1666, 2012, doi: 10.1016/j.procs.2012.04.183

[10] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard," Version 5.0, Jun. 2025. https://www.mpi-forum.org/docs/mpi-5.0/mpi50-report.pdf (accessed Jan. 19, 2026).

[11] T. Artés, A. Cencerrado, A. Cortés, and T. Margalef, "Relieving the Effects of Uncertainty in Forest Fire Spread Prediction by Hybrid MPI-OpenMP Parallel Strategies," Procedia Computer Science, vol. 18, pp. 2278–2287, 2013, doi: 10.1016/j.procs.2013.05.399

[12] GDAL/OGR contributors, "GDAL/OGR Geospatial Data Abstraction Software Library," Open Source Geospatial Foundation, 2026. https://gdal.org/en/stable/api/index.html. doi: 10.5281/zenodo.5884351.