UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

# COMPARACIÓN DE METAHEURÍSTICAS PARA LA UBICACIÓN DE CORTAFUEGOS EN EL COMBATE DE INCENDIOS FORESTALES

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN GESTIÓN DE OPERACIONES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

DAVID ISRAEL PALACIOS MENESES

PROFESOR GUÍA:
ANDRÉS WEINTRAUB POHORILLE

PROFESOR CO-GUÍA:
JAIME CARRASCO BARRA

MIEMBROS DE LA COMISIÓN:
SEBASTIÁN DÁVILA GÁLVEZ
RAFAEL EPSTEIN NUMHAUSER
CRISTÓBAL PAIS MARTÍNEZ

SANTIAGO DE CHILE
2022

MEMORIA PARA OPTAR AL TÍTULO DE:
Ingeniero Civil Industrial
TESIS PARA OPTAR AL GRADO DE:
Magíster en Gestión de Operaciones
POR: David Israel Palacios Meneses
AÑO: 2022
PROFESOR GUÍA: Andrés Weintraub Pohorille
PROFESOR CO-GUÍA: Jaime Carrasco Barra

# COMPARACIÓN DE METAHEURÍSTICAS PARA LA UBICACIÓN DE CORTAFUEGOS EN EL COMBATE DE INCENDIOS FORESTALES

En esta investigación se exponen, implementan y comparan metaheurísticas con el fin de resolver el problema de ubicación de cortafuegos para ayudar al combate de incendios forestales mediante el simulador Cell2Fire.

El problema considera el uso de un determinado porcentaje de la superficie del bosque como cortafuego, cuyo fin es detener el avance de un eventual incendio forestal. Para este trabajo se simplifican los costos operativos y los costos asociados a la pérdida económica y social a causa del incendio mediante la normalización del costo de cada cortafuego y la omisión de las pérdidas.

Considerando que en este problema existe un número importante de variables aleatorias, como el punto de inicio del incendio, dirección y velocidad del viento, temperatura ambiente, entre otras, se utilizan aproximaciones del desempeño real de los algoritmos mediante simulaciones de Monte Carlo a los resultados de las metaheurísticas. Para enfrentarlas a condiciones cercanas a las reales y considerando el tiempo que le tomaría a un equipo de combate de incendios el llegar a la zona afectada, se simulan incendios de 30 horas de duración en las que no hay intervenciones de dichos equipos. Para definir el desarrollo de las metaheurísticas, se consideran 3 fases metodológicas.

La primera fase metodológica consiste en un estudio de un amplio espectro de metaheurísticas para analizar aquellas que teóricamente mejor se adapten al problema o de las cuales existen investigaciones que las hayan aplicado en el problema o alguno similar.

La segunda fase de la metodología se basa en la implementación de las metaheurísticas escogidas en bosques pequeños con poca aleatoriedad para definir el desempeño de estas en ambientes controlados, y así obtener una idea del eventual desempeño que pudiesen tener en problemas menos triviales. De esta etapa se obtiene que las metaheurísticas seleccionadas en este trabajo son: Algoritmo Genético, *Greedy Randomized Adaptive Search Procedure* (o GRASP) y Búsqueda Tabú.

Finalmente, la tercera etapa de la metodología consiste en modificar las metaheurísticas para adaptarlas de acuerdo al problema en cuestión e implementarlas en bosques más reales y con condiciones meteorológicas más cercanas a la realidad. Para esto se compara el desempeño de los algoritmos con 3 variaciones distintas.

Los resultados obtenidos muestran un buen desempeño de las metaheurísticas al ser adaptadas para resolver este problema y enfrentándose a condiciones no totalmente aleatorias. Dichos resultados son obtenidos en pruebas de 2 horas de duración para cada metaheurística. Mediante un análisis de sensibilidad se logra determinar que al aumentar la superficie de cortafuegos o aumentando el tiempo de ejecución de los algoritmos se obtienen mejores resultados, con un mayor impacto en el primer caso. Finalmente se proponen modelos alternativos al problema planteado y se concluye que algunas de las metaheurísticas seleccionadas podrían ayudar a combatir eventuales incendios forestales en ciertas zonas geográficas como Chile.

# Tabla de Contenido

# Lista de Figuras

# Lista de Tablas

# Lista de Algoritmos

# 1 Introducción

## 1.1 Antecedentes

Los incendios forestales han ido al aumento en los últimos años y según como avanza el cambio climático, se espera que la magnitud y la frecuencia con que se generan estos eventos se mantendrá al alza en los años venideros [Westerling, 2016] [Running, 2016]. A pesar de que los incendios forestales presentan un factor positivo en algunas especies de la flora y fauna del planeta [Mabuhay et al, 2003] [Liu et al, 2007] [Ivanauskas et al, 2003], los efectos negativos tienden a opacar a los positivos, y al haber una menor ventana de tiempo entre incendios y un mayor impacto de estos, los efectos negativos son mucho más numerosos que los positivos [Jhariya y Raj, 2014]. Así, urge encontrar medidas para aplacar el impacto negativo que genera un incendio tanto en la biodiversidad del bosque como también en la que existe a su alrededor.

Una de las formas más eficientes para aplacar el impacto de un incendio forestal, es reducir su avance y por ende, la zona del bosque que se ve afectada por el mismo. En este sentido, los cortafuegos son uno de los métodos que se utilizan para frenar dicho avance y consisten en intervenciones al terreno del bosque para modificar la vegetación del mismo y así impedir un avance fácil del incendio.

Diversos métodos se han utilizado para decidir la ubicación óptima de cortafuegos en un bosque, algunos de las técnicas que se han usado son: Programación Lineal Entera Mixta, *Machine Learning*, Optimización de funciones tipo *Black-Box*, Metaheurísticas, entre otras.

Por lo tanto, el objetivo de este trabajo es estudiar, implementar, adaptar y comparar metaheurísticas para así determinar si estas tienen un desempeño mejor que una estrategia aleatoria en la ubicación de cortafuegos, con el fin de lograr disminuir el porcentaje de bosque perdido debido a un incendio forestal. Para lograr el propósito mencionado, se utiliza el simulador *Cell2Fire* que permite recrear diversos escenarios de incendios forestales en base a diferentes condiciones meteorológicas y topográficas en las que se genera el evento [Pais et al, 2019]. Con el fin de resolver el problema propuesto, es necesario en primer lugar definir un subconjunto de metaheurísticas que teóricamente prometan mejores resultados que otras.

Las metaheurísticas, en particular, son técnicas que se aplican en problemas de optimización para obtener resultados de forma rápida pero sacrificando la calidad del resultado. Así, es posible resolver problemas que las técnicas convencionales no son capaces de solucionar en un tiempo suficientemente bueno mediante el uso de algoritmos que resuelven dicho problema en un tiempo menor, pero sin lograr asegurar que el resultado obtenido es el mejor posible. Algunas de las metaheurísticas más conocidas son: Algoritmo Genético, Búsqueda Tabú, *Simulated Annealing*, Algoritmos de Enjambre, entre otras.

En general, las metaheurísticas son altamente adaptables por lo que pueden modificarse para mejorar su estrategia de funcionamiento de cara al problema. Es por esto que en este trabajo se incluyen variaciones de la estructura base de las metaheurísticas con el fin de lograr mejores resultados.

Una de las mayores complejidades que tiene este problema es la aleatoriedad del mismo, ya que un determinado incendio se propagará de forma distinta según las condiciones que existen al momento que se genera y se propaga este. Dentro de las principales fuentes de aleatoriedad del problema se cuentan: velocidad y dirección del viento, punto de ignición del incendio, temperatura ambiente, humedad relativa, entre otras. Para resolver la aleatoriedad del problema se utilizan simulaciones de Monte Carlo que permiten darle una mayor robustez a los resultados generados por

las metaheurísticas.

La metodología de este trabajo se construye en base a 3 etapas. En la primera etapa, en base a un estudio de la literatura existente, se estudia un amplio espectro de metaheurísticas con el fin de evaluar teóricamente como funcionarían estas en el problema, considerando su uso en problemas similares o del ámbito forestal, para así considerar metaheurísticas de las que se sabe que han tenido un buen desempeño en el área.

En la segunda etapa metodológica, se implementan los algoritmos seleccionados en ambientes sencillos, es decir, en bosques pequeños y con poca aleatoriedad. Por ejemplo, un incendio del cual se conoce su punto de ignición y en el que existe sólo una dirección del viento. De esta etapa se espera realizar un segundo filtro de metaheurísticas que más allá de adaptarse teóricamente al problema, se adapten en su aplicación real.

Una vez realizados los dos filtros anteriores, se espera adaptar las metaheurísticas para que funcionen de acuerdo al problema en cuestión. Para esto se proponen 3 variaciones en el funcionamiento de los algoritmos según su esquema básico para así evaluar si estas modificaciones mejoran los resultados obtenidos frente a la estructura básica de las metaheurísticas.

Finalmente, se propone una comparación de las metaheurísticas en distintos escenarios para así evaluar su desempeño frente a variadas condiciones. Para esto, se evalúa el desempeño del porcentaje promedio de bosque quemado de cada metaheurística en diferentes bosques, por medio de una serie de simulaciones de Cell2Fire. Dicho promedio será comparado con las soluciones obtenidas al ubicar cortafuegos aleatoriamente. Con esto, se desea concluir la o las metaheurísticas que mejor se comportan ante diversos escenarios y de la misma forma, determinar si el uso de estas es mejor que una distribución aleatoria de cortafuegos. Así, se desea que este trabajo permita brindar una revisión teórica y práctica para resolver este problema, además de asistir, desde el punto de vista de la planificación, al diseño de terrenos que sean resistentes a incendios forestales.

Para lograr esto la presente tesis desarrolla en el Capítulo 1 la presentación y algunos antecedentes del trabajo. En el Capítulo 2 se entrega el marco teórico que engloba al problema, describiendo la problemática de los incendios forestales, el simulador utilizado para generar los escenarios con los que se trabajará y la descripción de metaheurísticas, profundizando en algunas de las más conocidas. En el capítulo 3 se describe el problema que se intenta resolver junto al estado del arte que permite enfocar el presente trabajo, además de indicar los principales aportes del trabajo. En el Capítulo 4 se expone la metodología y desarrollo del trabajo, mientras que en el Capítulo 5 se presentan los resultados más relevantes de la investigación. En el Capítulo 6 se discuten y analizan los resultados presentados en la sección anterior y se describen posibles extensiones al trabajo. Finalmente, en el Capítulo 7 se exponen las principales conclusiones del trabajo realizado.

## 1.2 Objetivos del Estudio

El objetivo general de este trabajo es:

*"Implementar y comparar metaheurísticas para así determinar si estas tienen un desempeño mejor que una estrategia aleatoria en la ubicación de cortafuegos, con el fin de lograr disminuir el porcentaje de bosque perdido debido a un incendio forestal"*

Los objetivos específicos son:

1. Estudiar en la literatura las metaheurísticas y sus aplicaciones en el problema o problemas similares
2. Determinar un conjunto de metaheurísticas que mejor se adapten al problema
3. Adaptar el comportamiento de las metaheurísticas para mejorar su estrategia de funcionamiento ante el problema
4. Medir la cantidad de bosque quemado ante óptimos encontrados por las metaheurísticas bajo un conjunto de simulaciones con distintas condiciones climáticas a las que se puede ver enfrentado
5. Comparar entre sí los resultados obtenidos por las metaheurísticas y además, comparar dichos resultados frente a disposiciones aleatorias de cortafuegos

# 2 Marco Teórico

## 2.1 Incendios Forestales

En los primeros 10 meses del año 2019, Bolivia triplicó su área quemada debido a incendios forestales con respecto al mismo período del año 2018, mientras que Venezuela superó en un 24% su área quemada con respecto al mismo período, en tanto Brasil tuvo un 70 % más de área quemada comparando dichos lapsos de tiempo. Todo esto provocado por una serie de incendios que tuvo lugar en el bosque Amazonas en agosto de 2019. A pesar de esto, la porción quemada no superó en demasía el promedio anual de área quemada entre 2001 y 2018 [Lizundia-Loiola et al, 2020], pero la importancia del impacto que generó este evento se debe a que afectó áreas que con anterioridad no habían sufrido incendios. Esto posiblemente se produjo debido a la suma de diversos factores: una serie de incendios forestales en los 3 años previos, la constante deforestación en los 5 años previos, y una mayor deforestación y posterior ocupación de los territorios en el año 2019 sumado a condiciones climáticas que si bien se habían dado en años anteriores, nunca se habían dado con las variables mencionadas [Silveira et al, 2020].

A nivel nacional, los días 12 y 13 de abril del año 2014 se generó un incendio en los cerros de Valparaíso, que dejó cerca de 3.000 viviendas destruidas y más de 12.000 personas damnificadas. Dentro de las principales causas de este evento y su rápida propagación, está la irregularidad (de forma y norma) del desarrollo urbano de las zonas altas en las que estaban ubicadas las viviendas (la mayor parte de ellas construidas por los propios pobladores), fenómeno estrechamente relacionado con la alta vulnerabilidad social [Orozco y Rivera, 2015]. Este evento demuestra que los desastres naturales, lejos de ser episodios puntuales, son una construcción donde no sólo está presente el entorno geográfico como factor de riesgo, sino que condiciones estructurales y sociales de las familias que acentúan dichos riesgos [Bailey y Zenteno Torres, 2015]. Este contexto se torna especialmente de cuidado en un escenario post-pandemia debido al virus COVID-19, la cual ha incrementado los índices de precariedad afectando directamente la incidencia, severidad e intensidad de la pobreza, tanto en Estados Unidos como en países en desarrollo [Sumner et al, 2020] [Parolin, 2020], levantando una alarma en el área de los incendios forestales, los cuales se generan en más del 90% de los casos debido a la acción humana [Paton et al, 2014].

Sumado al factor antropológico como origen de los incendios forestales, se encuentra el cambio climático que también genera condiciones más favorables para la ocurrencia de estos eventos. Por ejemplo, en California, las temperaturas han aumentado en aproximadamente 1°C y las precipitaciones han bajado un 30% en las últimas 4 décadas, lo que ha generado que la frecuencia de días con condiciones climáticas extremas (que se han asociado con ocurrencias de incendios forestales) se ha duplicado con respecto a inicios de la década de los 1980s [Goss et al, 2020]. Por otro lado, en el sur de Europa, el riesgo de incendio debido a la emisión de gases de invernadero ha aumentado en promedio desde un 2% a un 4% y el riesgo de área quemada ha subido de un 2% hasta un 50% [Dupuy et al, 2020]. De igual manera, diversas investigaciones han señalado que la ocurrencia de incendios y el área que afectan, han aumentado debido a los efectos del cambio climático en la variación de las temperaturas, la cantidad de precipitaciones y la humedad relativa del suelo [Westerling, 2016] [Running, 2016].

El aumento en la cantidad e impacto de los incendios forestales es una señal de alerta para la comunidad, ya que estos eventos afectan al ecosistema de múltiples maneras. A modo de ejemplo, el año 2012, de haberse prevenido todos los incendios forestales de las regiones al oeste del Amazonas, particularmente en los estados bolivianos y brasileños de Rondônia, Acre, y Mato Grosso, se estima que se podrían haber evitado 16.000 muertes prematuras y 641.000 años de vida ajustados por

discapacidad (*DALYs*) [Butt et al, 2020]. El ya mencionado incendio del Amazonas disminuyó la productividad del bosque modificando totalmente el ciclo del carbono y el flujo del $CO_2$ de dicho año [Bandopadhyay y Sánchez, 2020]. De igual manera, en Chile, en el año 2017 se quemaron más de 500.000 hectáreas, lo que generó emisiones de $CO_2$ equivalentes al 90% del total de emisiones generadas en el país en el año 2016 (asociadas a energía, transporte, residuos, etc.). Esto es equivalente a 23 años de emisiones de $CO_2$ de todos los vehículos livianos de la Región Metropolitana [González et al, 2020].

Algunos efectos que genera un incendio forestal son:

- **Flora**: El impacto del daño depende de la especie, edad, intensidad del incendio y el tipo de vegetación. En árboles por ejemplo, la cantidad de biomasa de un árbol disminuye al encontrarse en zonas que periódicamente enfrentan incendios forestales. Un incendio forestal que afecta a un árbol provoca un impacto negativo en el ciclo del carbono, ya que los nutrientes esenciales (como $N$, $S$, $K$, $Na$, etc) se ven alterados por el incendio [Jhariya, 2014]. Del mismo modo, la quema de la biomasa de un árbol altera también la emisión de $CO_2$, principal causante del efecto invernadero [Klock y Grier, 1979]. Así, el bosque genera un papel primordial en la mantención del nivel de $CO_2$, el secuestro del carbono y la protección del ecosistema frente al calentamiento global [Jhariya y Raj, 2014].

- **Fauna silvestre**: Un incendio afecta a los animales debido a la muerte directa de estos a causa del fuego o mediante la modificación de la estructura y composición de su hábitat [Lyon et al, 2000]. Algunos incendios modifican la vegetación del bosque, la que actúa como un refugio y escondite para ciertos animales así como también se ve alterada la disposición espacial de esta vegetación que otorga los recursos para vivir y reproducirse. Por ejemplo, los troncos muertos del bosque son usados por algunos pájaros, insectos, mamíferos pequeños u osos [Bull y Blumton, 1999]. Un incendio también puede provocar cambios conductuales en ciertas especies, por ejemplo luego de grandes incendios en California, se observó que búhos sobrevivían menos y tenían una menor tendencia tanto a asentarse en lugares como en establecer relaciones sexuales con otros búhos y por ende, se reproducían exitosamente en menor medida [Bond et al, 2002].

- **Suelo**: Los incendios modifican al suelo físicamente, químicamente y biológicamente. Físicamente, el suelo puede modificar su estructura, disminuyendo su porosidad y permeabilidad [Certini, 2005]. Del mismo modo, un incendio puede modificar la repelencia al agua del suelo, lo que provoca una mayor erosión del suelo disminuyendo la posibilidad de reforestar dicha área [Letey, 2001] [DeBano, 1990].
  Químicamente, el suelo de los bosques guarda aproximadamente un 45% del carbono existente bajo tierra del mundo [Dixon Y Turner, 1991]. Incendios de baja intensidad provocan pequeñas emisiones de carbono desde el suelo, pero incendios de alta intensidad pueden desencadenar importantes emisiones de carbono al medio ambiente [Johnson, 1992].
  Biológicamente, el fuego puede afectar organismos invertebrados y micro-organismos de forma directa o indirecta. Los invertebrados de suelo juegan un papel importante en descomponer materia orgánica, mineralizar nutrientes y carbono, y en la formación estructural del suelo [Neary et al, 1999]. Los micro-organismos juegan un rol trascendental en los ciclos naturales de los nutrientes y en el flujo energético del ecosistema forestal. Los efectos que generan los incendios forestales a estos pueden ser positivos [Mabuhay et al, 2003] [Liu et al, 2007], negativos [Choromanska y DeLuca, 2001] [Rodriguez et al, 2009] o neutrales [Rutigliano et al, 2007].

- **Salud humana**: La evidencia muestra que la salud de las personas puede verse afectada debido a los incendios forestales [Finlay et al, 2012]. El humo de la madera quemada tiene altos niveles de material particulado y toxinas. El problema más directo proviene de las

quemaduras que pueden sufrir las personas al contacto directo con el fuego, ya sean residentes afectados o personal de socorro. Dichas quemaduras pueden necesitar cuidado intensivo e internación hospitalaria, ya que dependiendo de su gravedad es probable que existan fallas multi-orgánicas. Aparte de esto, si bien el síntoma más común son problemas respiratorios, también existen efectos a nivel cardiovascular, oftalmológico y psiquiátricos. Además, la emisión de gases y residuos debido al incendio puede ocasionar contaminación en el aire, agua, y tierra alrededor del bosque, lo que también puede generar complicaciones sanitarias. En el verano de 2006, por ejemplo, una serie de incendios en Galicia, España, amenazó como en muy pocas ocasiones anteriores a granjas, casas y vidas humanas. Lo anterior generó que en los meses posteriores a estos eventos se experimentara un aumento en el consumo de medicamentos para tratar enfermedades respiratorias obstructivas en pensionados que vivían en municipios afectados por los incendios comparado con pensionados que no vivían en estos municipios en el mismo período. De la misma manera, se evidenció un aumento en el consumo de ansiolíticos y somníferos en hombres (pensionados o no) que habitaban estos municipios comparado con hombres que no habitaban estos municipios al momento de los incendios [Camaano-Isorna et al, 2011].

- **Otras**: Modificación al clima [De Sales et al, 2019], impacto en el turismo [Castillo, 2017], impacto social [Schumann et al, 2020], modificación demográfica [Heck, 2007], etc.

Es por todo esto, que se hace menester el incorporar políticas para enfrentar un incendio forestal, de acuerdo a [Julio y Bosnich 2005], estas deberían constar de 4 pilares: Prevención, Pre-supresión, Combate y Uso del Fuego.

- **Prevención**: Corresponde a todas las medidas que están destinadas a impedir la generación y propagación de incendios forestales. Estas medidas controlan tanto el riesgo como el peligro, el primero corresponde al agente que provoca el incendio, mientras que el segundo corresponde al nivel de conflictividad que pudiese alcanzar un incendio una vez propagado. La prevención dependerá de las condiciones topográficas, de la vegetación y del estado atmosférico.

- **Pre-supresión**: Corresponde a la planificación y programación realizada con anterioridad a la ocurrencia del incendio, con el fin de evaluar los problemas que puedan llegar a surgir debido a este.

- **Combate**: Es la actividad destinada al control del incendio forestal. Su objetivo es la extinción del foco del fuego y se ejecuta de acuerdo a lo planificado en la etapa de pre-supresión.

- **Uso del Fuego**: Se realiza en operaciones agrícolas y forestales, las cuales pueden ser ejecutadas utilizando el fuego como herramienta de trabajo. Debe ser usado de forma cuidadosa, por lo que es necesario que se realice en conjunto a un plan de quema.

En este trabajo, se estudiará una aplicación del primer punto de los señalados, es decir, la prevención de un incendio forestal. De acuerdo a [Curt et al, 2016], a causa de los recientes incidentes, se deben tomar medidas preventivas para así disminuir la probabilidad de que determinado territorio enfrente un incendio forestal, y de este modo proteger bosques, condiciones climáticas y vidas humanas.

La prevención de un incendio forestal puede elaborarse desde distintas perspectivas, como quemas permitidas, campañas públicas para la prevención, campañas en escuelas, endurecimiento de condenas por generación y propagación de incendios forestales o mediante manejo de combustible [1] [Abt et al, 2015] [Cary et al, 2009] [Vélez, 1982]. El objetivo de la metodología de manejo

[1] Combustible en este trabajo hace referencia al tipo de vegetación presente en el terreno.

de combustible y ubicación de cortafuegos,[2] es modificar la composición de la vegetación del bosque para así prepararse de mejor manera ante futuros incendios. En este sentido, se han propuesto muchas técnicas para alterar el combustible de un bosque como: corte y despeje de madera, incendios planificados o raleo[3] [Agee y Skinner, 2005]. Todas estas acciones también tienen un impacto en el ecosistema del bosque, ya que podrían afectar la diversidad de las especies [Shinneman et al, 2019], así como también mejorar la salud de los ecosistemas [Hirsch et al, 2001], entre otras [Moghaddas y Craggs, 2008].

Con tal de mitigar el efecto de los incendios forestales, se han estudiado diversos enfoques como: programación lineal, técnicas de *Machine Learning*, algoritmos aproximados o metaheurísticas (estas últimas se revisarán en la sección 3). Por ejemplo, en [Pais et al, 2021] se presenta el concepto de *Downstream Protection Value* para detectar áreas críticas a las cuales tratar para disminuir el riesgo de un incendio forestal y en [Elimbi Moudio et al, 2021] se extiende este concepto. En [Guo et al, 2016] se propone un modelo que utiliza la función $k$ de *Ripley* y regresión logística para determinar espacialmente las zonas con mayor probabilidad de ocurrencia de incendios. Por su parte, en [Brakeall, 2013], dos de sus objetivos son determinar qué meses del año son los más seguros para implementar quemas planificadas y qué tipo de vegetación o sub-ecosistema dentro del bosque presentará una tasa de propagación de incendios más rápida y cual experimentará una más lenta. A su vez, en [Zheng, 2010] se utiliza un modelo de programación lineal para ubicar de forma óptima diversos recursos de cara a incendios forestales. Algo similar se plantea en [Chan et al, 2020], en donde se diseña y utiliza el algoritmo *Firefly* para ubicar de forma semi-óptima recursos (bomberos humanos y bomberos tipo dron) para el combate de incendios forestales.

## 2.2 Cell2Fire

Cell2Fire es un simulador de incendios forestales de código abierto diseñado e implementado en [Pais et al, 2019]. Se trata de un modelo basado en celdas (o *celullar automata*), por lo que cada bosque se compone de $N$ celdas, cada una homogénea en su interior, por las que el fuego se propaga desde el centro de una determinada celda, al centro de otra celda de acuerdo a una probabilidad de ignición. Por lo general, en la simulación de incendios forestales se utiliza el modelo *cellular automata* (como en [Alexandridis et al, 1990]), o el modelo de propagación de *Huygens*, en donde se modela el incendio como un polígono que avanza a lo largo del bosque y se propaga mediante el contacto de una ola principal con la superficie tangencial a la circunferencia de una ola secundaria. Este último modelo se utiliza por ejemplo en [Richards, 1990].

El objetivo de Cell2Fire a diferencia de otros simuladores como el de [Arca et al, 2019] no es asistir a equipos operacionales que deben actuar en el momento de la ocurrencia del incendio, sino que busca generar estrategias preventivas para reducir el impacto del fuego. Para esto se intenta construir mapas del comportamiento del incendio (o cicatrices del incendio) lo más realistas posibles y así realizar planes de manejo de combustible o remoción de vegetación. Todo esto con una alta eficiencia computacional, ya que para ejecutar dichos planes puede ser necesario contar con varios escenarios de cicatrices de incendios para evaluar distintas estrategias.

En Cell2Fire el estado de una celda viene dado por una determinada probabilidad modelada por una cadena de Markov, dichos estados pueden ser:

[2]Un cortafuego corresponde a un área con combustible modificado cuya misión es detener o estorbar la propagación del fuego.

[3]El raleo es una técnica que consiste en eliminar la vegetación excedente de una planta o árbol. En el área forestal corresponde a la remoción de ramas y hojas de los árboles para así disminuir su volumen.

- ***Avaliable***: Este estado corresponde al de una celda que contiene un combustible inflamable, y que de ser alcanzada por el fuego, tiene la capacidad de arder.
- ***Burning***: Corresponde a una celda que contiene un incendio activo y que puede propagar el incendio a otra celda .
- ***Burned***: Son las celdas por las que el incendio ya ha pasado.
- ***Harvested***: Estas celdas son celdas tipo "cortafuego", las cuales son entregadas por el usuario al simulador y no pueden ser quemadas por el incendio.
- ***Non-Fuel***: Son celdas que contienen superficie no inflamable como lagos, ríos o piedras.

Para predecir el comportamiento del fuego, Cell2Fire utiliza 2 parámetros: *rate of spread* (ROS) y la intensidad del incendio, ambos dados por el tipo de combustible de la celda, temperatura, humedad, velocidad y dirección del viento, entre otras. El simulador utiliza el *Canadian Forest Fire Behavior Prediction System* (FBP), sistema que maneja datos empíricos de propagación de incendios, los cuales pueden usarse para predecir ROS y la intensidad del incendio debido a ciertas condiciones de humedad, estación del año, condiciones climáticas y condiciones topográficas de celdas con determinados tipos de combustible. Los combustibles (o tipo de vegetación) utilizados por el FBP son extraídos del ecosistema canadiense, los cuales difieren de los combustibles de otras zonas geográficas, por lo que el simulador no es directamente aplicable en zonas como Chile, a diferencia de simuladores como KITRAL [Castillo, 1998] que sí utilizan combustibles de la zona.

El simulador utiliza 3 variables que le otorgan una naturaleza estocástica a Cell2Fire. La primera de ellas es el punto de ignición del incendio, este corresponde al nodo en el cual se inicia el fuego. Puede ser dado por el usuario de acuerdo a una cierta distribución de probabilidad, o puede tener una probabilidad uniforme a lo largo de todas las celdas. La otra fuente de incertidumbre es mediante la variación de los valores básicos del FBP para determinar ROS, así, se pueden simular distintos escenarios de acuerdo a la forma de propagación del incendio. Finalmente, el usuario puede simular incendios con sus propios escenarios meteorológicos y así estudiar la evolución del fuego frente a distintas condiciones climáticas.

El funcionamiento del simulador es de acuerdo a un algoritmo que en cada período actualiza el estado de las celdas según la propagación del fuego. Inicialmente, el incendio parte con un punto de ignición. Desde este y en cada período, el fuego puede propagarse a celdas adyacentes (se asume que cada celda tiene a lo más 8 celdas vecinas, ver Figura 2.1) desde del centro de la celda hasta el centro de una celda vecina. Para determinar esta probabilidad se utilizan las métricas *Head Rate of Spread* (HROS), *Flank Rate of Spread* (FROS) y *Back Rate of Spread* (BROS), donde la primera indica la tasa de propagación hacia adelante, la segunda la tasa de propagación hacia los lados, y la tercera la tasa de propagación hacia atrás. Con estas métricas se construye un modelo elipsoidal del fuego, el cual permite determinar mediante su geometría, si el fuego que viene desde del centro de una celda alcanza a tocar el centro de otra celda, evento que puede cambiar el estado de la segunda a *Burning* (ver Figura 2.1 para observar este modelo de propagación). Los valores HROS, FROS y BROS son calculados por el FBP de acuerdo al procedimiento descrito en [Tymstra et al, 2010] y a las ecuaciones:

$$a = \frac{HROS + BROS}{2} \cdot t$$

$$b = \frac{2 \cdot FROS}{2} \cdot t$$

$$e = \sqrt{1 - \frac{(FROS \cdot t)^2}{\frac{(HROS+BROS)\cdot t^2}{2}}}$$

Donde a, b y e corresponden al eje semi-mayor, eje semi-menor y la excentricidad de la elipse que describe al incendio, respectivamente. Mientras que t representa el período del incendio.

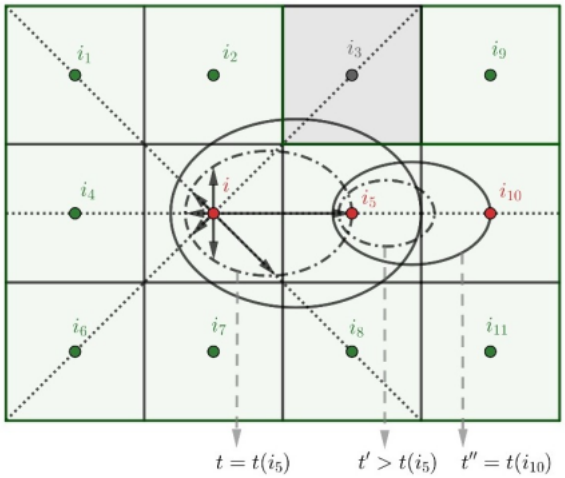



Figura 2.1: Modelo de Propagación Cell2Fire. De [Pais et al, 2019]

En cuanto la elipse de una celda toca el centro de una celda vecina[4], se calcula el ROS de la segunda celda, y si este valor es mayor a un determinado valor dado por el usuario entonces la celda pasa a estar en el estado *Burning*. El estado del fuego se calcula en cada período de acuerdo al estado de las celdas *Burning*. En el momento en que (i) ya no existen celdas vecinas en estado *Available* a una celda en estado *Burning*; ó (ii) el FROS, BROS, HROS son menores a determinado valor empírico $\delta > 0$; ó (iii) el combustible de la celda ya no es suficiente para propagar el incendio; ó (iv) la intensidad delantera del incendio ($HROS \cdot t$) es menor a un valor dado por el usuario, entonces la celda pasa a estar en estado *Burned*, y ya no es tomada en cuenta nuevamente en el simulador. Este proceso se repite hasta que ha pasado el tiempo simulado por el usuario o hasta que ya no queden celdas en estado *Available*

Cell2Fire tiene la capacidad de simular distintos escenarios para un mismo bosque las veces que el usuario lo requiera. En el algoritmo 1 se describe el funcionamiento de una simulación de Cell2Fire, donde $i$ corresponde a la celda $i$ del bosque. $Adj(i)$ corresponde al conjunto de celdas *Available* adyacentes a la celda $i$. $BurningCells$ corresponde al conjunto de celdas que actualmente se encuentran en estado *Burning* y $i \rightarrow_j$ indica que la celda $i$ alcanza a la celda $j$ en el tiempo $t$.

El modelo utilizado por Cell2Fire principalmente asume:

1. El crecimiento del fuego depende del ROS del fuego de una celda hacia una celda adyacente. Se asume que una celda se incendia cuando el fuego alcanza su centro y ciertas condiciones se cumplen (punto 5. de esta lista). Además, cada celda posee a lo más 8 vecinos.

2. HROS, FROS y BROS se calculan en base al sistema canadiense FBP (podrían utilizarse otros sistemas que modificarían la forma de propagación del incendio). El valor de cada dirección de la elipse depende de la dirección del viento, específicamente HROS corresponde a la dirección que apunta el viento, BROS corresponde a la dirección contraria y FROS apunta en las direcciones perpendiculares al viento.

[4]No se consideran celdas más allá de las 8 vecinas, ya que se asume que el simulador utiliza períodos de tiempo suficiente pequeños como para que esta propagación no sea tan extensa

**Algorithm 1** Pseudo-código de una simulación de Cell2Fire, de [Pais et al, 2019]

**procedure** $Sim(ForestData, FTypes, Ignitions, Weather, TMax, Options)$
**Step 0: Initialize Cell2Fire**
$ParseInputs, options, readData, initializeObjects$
$FPeriod \leftarrow 0, nsim \leftarrow 0$
**Step 1: Ignition**
$ic \leftarrow Choice(Ignitions)$
$BurningCells \cup \{ic\}$
$FPeriod \leftarrow 1$
**Step 2: Fire Dynamics (Send-Receive)**
$Let\ i \in BurningCells, if\ i \rightarrow_{FPeriod} j, where\ j \in Adj(i):$
$BurningCells \cup \{j\}$
$FPeriod \leftarrow FPeriod + 1, UpdateForest, Weather$
**Repeat, until** $FPeriod < TMax$
**Step 3: Results and Outputs generation**
$GenerateGrids, Statistics, Outputplots$
**end procedure**

3. Las celdas que se queman sirven como una nueva fuente de fuego. Además, la propagación ocurre a un nivel de celdas cuyo tamaño depende de tanto la resolución espacial utilizada como de los datos disponibles.

4. No se modela el efecto de la supresión del fuego.

5. Se usan 2 formas para finalizar un escenario de incendio: a nivel celular y según las condiciones generales de la evolución del incendio. A nivel celular una celda termina de quemarse (pasa a estado *Burned*) si se cumple una de las siguientes condiciones:

   (a) ya no existen celdas vecinas en estado *Available* a una celda en estado *Burning*,

   (b) el FROS, BROS, HROS son menores a determinado valor empírico $\delta > 0$,

   (c) el combustible de la celda ya no es suficiente para propagar el incendio,

   (d) la intensidad delantera del incendio ($HROS \cdot t$) es menor a un valor dado por el usuario.

   A pesar de esto, si la simulación lleva más del tiempo de incendio determinado por el usuario o si ya no quedan celdas combustibles (es decir, no hay celdas en estado *Available*) en el bosque, entonces la simulación se concluye.

Luego de simular, Cell2Fire es capaz de entregar 3 resultados: *Burn-Grids*, gráficos y estadísticas.

- ***Burn-Grids***: Son archivos que representan al bosque y en los que cada celda es representada según 3 valores: un 1 para celdas quemadas, -1 para celdas tratadas como cortafuegos y un 0 para celdas que no son quemadas (y que no son cortafuegos). Estos archivos son útiles para calcular estadísticas de comparación con respecto a otros simuladores o con respecto a otros experimentos de Cell2Fire.

- **Gráficos**: Cell2Fire genera gráficos de: estado inicial del bosque, evolución de la cicatriz de fuego y gráficos de comunicación de celdas, útiles para observar qué celdas fueron las que más transmitieron fuego.

- **Estadísticas**: Contienen información relevante sobre el estado final del bosque, como momento de ignición, evolución del fuego (valores de ROS), número promedio de celdas quemadas en una serie de iteraciones, porcentaje de celdas no quemadas en una serie de simulaciones, etc.

Los resultados obtenidos por Cell2Fire han mostrado un desempeño similar al del simulador *Prometheus* [Tymstra et al, 2010], uno de los 2 modelos que han tenido mejor desempeño en simular incendios de los que se tienen registros históricos (el otro corresponde a *FARSITE* [Finney, 1995]). Estos resultados son visibles en las cicatrices de incendio generadas por ambos simuladores.

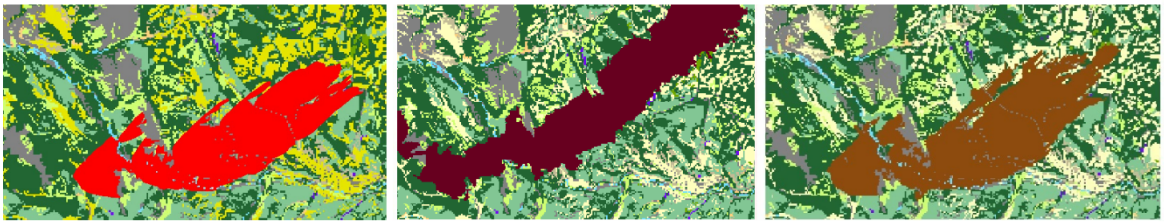



Figura 2.2: Comparación cicatrices Cell2Fire y Prometheus. a) Cicatriz generada por simulador Prometheus de incendio en la región de Dogrib, comparada con b) incendio real proyectado en formato grilla creado en 2002 y c) cicatriz generada por Cell2Fire.

## 2.3 Metaheurísticas

Un problema de optimización se define como un problema que busca encontrar el o los mejores valores de una determinada función objetivo. La forma estándar de estos problemas tiene la forma [Boyd y Vandenberghe, 2004 ]:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad \forall i = 1, ..., m \\ & h_i(x) = 0, \quad \forall i = 1, ..., r \end{aligned}$$

con $f(x)$ la función a minimizar (o maximizar) y las funciones $g_i(x)$ y $h_i(x)$ son funciones restrictivas cuyo objetivo es lograr que la función $f(x)$ tome su mínimo (máximo) valor sujeta a ciertas condiciones en la variable $x$. Este resultado se conoce como el óptimo del problema y se denota por $p^*$ y formalmente se define para un problema de minimización como:

$$p^* = inf\{f(x)|g_i(x) \leq 0, i = 1, ..., m, h_i(x) = 0, i = 1, ..., r\}$$

El conjunto de todas las soluciones $x$ posibles se conoce como el espacio de soluciones o espacio de búsqueda. Dentro este espacio de soluciones existen soluciones mejores que otras. Específicamente, la (o las) solución (es) que permiten alcanzar el óptimo es (son) conocida (s) como solución (es) global (es) y se denota (n) como $x^*$. Formalmente, el conjunto de soluciones globales de un problema de optimización es definido como:

$$X_{opt} = \{x|g_i(x) \leq 0, i = 1, ..., m, h_i(x) = 0, i = 1, ..., r, f(x) = p^*\}$$

A su vez, una solución que alcanza un desempeño menor que el óptimo es conocida como solución sub-óptima. Una solución que alcanza un óptimo en una determinada región del espacio de solu-

ciones se le conoce como solución local. Formalmente se define una solución local en una vecindad $R > 0$ si:

$$f(x) = inf\{f(z)|g_i(z) \leq 0, i = 1, ..., m, h_i(z) = 0, i = 1, ..., r, \|z - x\|_2 \leq R\}$$

Generalmente los problemas de optimización son demasiado difíciles de resolver y se utilizan técnicas para llegar a soluciones sub-óptimas de estos.

Diversos métodos se han estudiado y aplicado a lo largo de los años para resolver problemas de optimización. Usualmente estos problemas son resueltos mediante métodos iterativos. Dichos métodos pueden ser deterministas, estocásticos o estocásticos con aprendizaje. Dentro del grupo de los métodos deterministas es posible hallar algoritmos que resuelven problemas mediante programación lineal, optimización convexa o métodos de gradientes. En el grupo de métodos estocásticos están las heurísticas y metaheurísticas, mientras que en el grupo de métodos estocásticos con aprendizaje se encuentran técnicas de aprendizaje supervisado y no supervisado. En la figura 2.3 se puede ver una tabla resumen con la clasificación de algunos algoritmos de resolución de problemas de optimización.

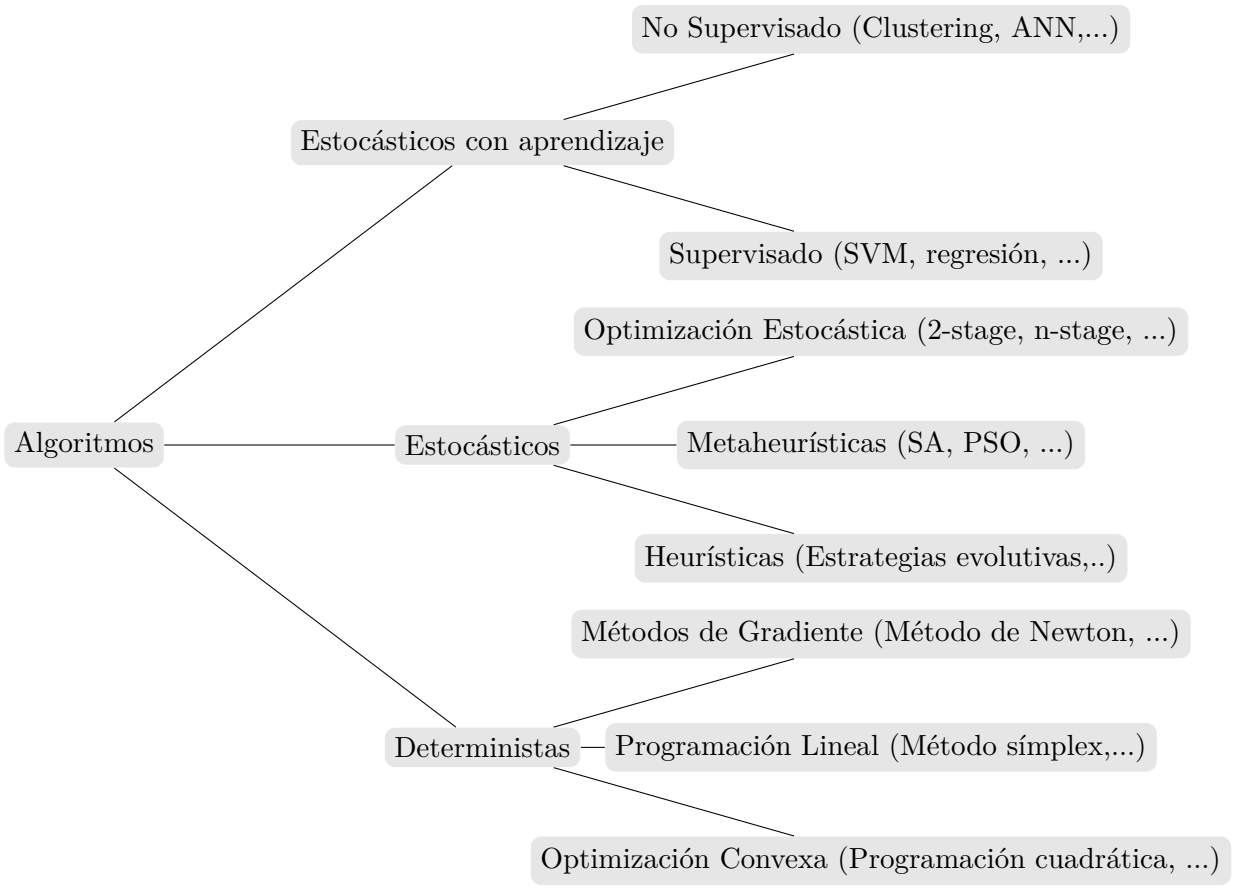



Figura 2.3: Clasificación de Algoritmos de Optimización. De [Yang, 2021]

El concepto de heurística proviene del griego *heuriskein* que quiere decir "encontrar algo o descubrir", en psicología David Myers las define como atajos mentales que facilitan la carga cognitiva de tomar una decisión [Myers, 2010]. En términos computacionales y algorítmicos, Romanyci y Pelletier definen una heurística en Inteligencia Artificial como cualquier dispositivo, ya sea un programa, regla, porción de conocimiento, etc, que no con toda confianza será capaz de otorgar una

solución practica, pero que del cual se tiene alguna razón para creer que será útil y que se le es añadido a un sistema que soluciona problemas con la esperanza de que en promedio su desempeño mejorará [Romanyci y Pelletier, 1985].

Las heurísticas en problemas de optimización son técnicas usadas para conseguir no necesariamente las mejores soluciones de dicho problema (aunque podrían eventualmente obtener el óptimo global), sino que se utilizan para encontrar valores relativamente buenos en un tiempo menor al que le tomaría a una técnica determinista. Estos métodos se han utilizado desde hace varias décadas, el concepto de resolver problemas de optimización mediante heurísticas fue presentado por George Pólya en 1945 [Pólya, 1945], mientras que el algoritmo símplex, que puede ser considerado la primera heurística de búsqueda local para problemas de programación lineal fue creado por George Dantzig en 1947 [Dantzig, 1963].

Unas de las heurísticas más conocidas son las heurísticas golosas (o *Greedy Algorithms*). Estas fueron esquematizadas para varios problemas de grafos en los años 1950s. En 1959 Edsger Djikstra construyó un algoritmo para generar un árbol de mínimo crecimiento con el fin de disminuir el número de rutas dentro de la ciudad de Ámsterdam [Djikstra, 1959].

Otro método heurístico famoso son las heurísticas evolutivas, estas tienen su origen también en los años 1950s, período en el que Nils Aall Barricelli publica *Esempi Numerici di processi di evoluzione*, un artículo que se basa en vida artificial para utilizar simulación evolucionaria y así resolver problemas generales de optimización [Barricelli, 1954].

El concepto de metaheurísticas comienza a surgir algunos años después, cercano a los años 1970s, cuando John Henry Holland propone la primera versión de un algoritmo genético [Holland, 1975]. Generalmente, se dice que las heurísticas y metaheurísticas se diferencian según su complejidad y flexibilidad [El-Omari, 2021], puesto que las primeras tienen una menor performance y están sujetas a resolver problemas definidos, mientras que las segundas tienen un mejor desempeño (al igual que las heurísticas, no aseguran un óptimo) y son aplicables y moldeables a distintos tipos de problemas.

Las heurísticas en el área de la programación y la resolución de problemas de optimización se utilizan cuando la solución exacta de un problema no es lo suficientemente fácil de obtener, para esto, se utilizan medidas de complejidad que clasifican a los problemas de programación lineal de acuerdo a la complejidad temporal que le toma al mejor algoritmo posible resolver el problema respectivo.

Dentro de la clase de problemas más simples están los *P-Problems*, para los cuales existe un algoritmo capaz de resolver el problema en una cantidad de tiempo de orden polinomial. Un ejemplo de esto, es encontrar en una lista de $n$ números un valor mayor al entero $k$, esto puede ser realizado en un tiempo de orden lineal recorriendo cada elemento de la lista y revisando si es mayor a $k$. Luego están los *Nondeterministically Polynomial problems* (NP), para los cuales existe un algoritmo no determinista que resuelve el problema en tiempo polinomial (dentro de esta clasificación se incluyen los *P-Problems*), en esta categoría podríamos encontrar el Problema del isomorfismo de grafos [Schöning, 1987].

Luego de estas categorías encontramos los tipo NP-Completo y NP-Duro. Para probar que un problema es NP-Duro se utiliza una transformación entre el problema (que se asume que es de tipo NP) y algún problema NP conocido (la mayor cantidad de demostraciones utiliza el problema *SAT* o *Boolean Satisfiability Problem* [Battiti, 2008]), de existir dicha transformación, entonces el problema es NP-Duro. Si además, dicho problema es NP, entonces el problema es NP-Completo,

para esto es necesario encontrar un algoritmo que resuelva el problema en un tiempo de orden polinomial. Dentro de los problemas NP-Completo se puede encontrar el *0/1 Knapsack problem* o el ya mencionado *SAT* y en la categoría de problemas NP-Duro está el problema de *Halting* [Davis, 1958]. Abierta está la discusión sobre la equivalencia entre los problemas de tipo P y NP, que en caso de ser cierta, implicaría un cambio en el paradigma de estudio de estos problemas [Fortnow, 2009].

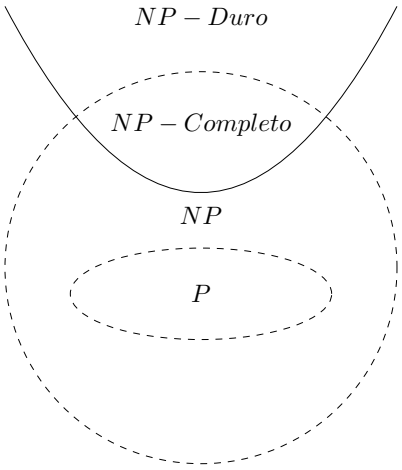



Figura 2.4: Complejidad de problemas de decisión. Este diagrama asume P!=NP.

| | P | NP | NPC | NPD |
|---|---|---|---|---|
| Se puede resolver en tiempo de orden polinomial | ✓ | X | X | X |
| Se puede comprobar su solución en tiempo de orden polinomial | ✓ | ✓ | ✓ | X |
| Reduce cualquier problema NP en tiempo de orden polinomial | X | X | ✓ | ✓ |

Tabla 2.1: Propiedades de clases de problemas. Esta tabla asume P!=NP.

Teniendo en cuenta la complejidad de los algoritmos, el proceso de decisión del uso de una metaheurística viene dado tanto por esta variable, como también por el tamaño del problema, ya que problemas del tipo NP-Duro con pequeñas instancias aún son posibles de resolver con métodos deterministas, sin correr el riesgo de penalizar nuestro problema. Del mismo modo, no es aconsejable utilizar metaheurísticas en problemas donde existen métodos exactos de resolución, por ejemplo en problemas de tipo P. Así, se debería utilizar una metaheurística cuando tanto la complejidad del problema como su tamaño provocan una incapacidad de resolución exacta en un tiempo prudente.

Es posible encontrar aplicaciones de metaheurísticas en un sinfín de problemas de distintas áreas, por ejemplo, aerodinámica, dinámica de fluidos, telecomunicaciones, robótica, *Machine Learning* y *Data Mining* en bioinformática o finanzas, procesamiento de imágenes, logística y transporte, *Supply Chain*, etc [Torres-Jiménez y Pavón, 2014].

Los 2 conceptos básicos en la construcción del diseño de una metaheurística son la diversificación y la intensificación. La primera genera distintas soluciones de un determinado problema para así recorrer la mayor cantidad posible del espacio de soluciones, mientras que la segunda explota una determinada solución para así buscar asegurar el descubrimiento de buenas soluciones en un determinado espacio. El equilibrio de estos 2 conceptos es crucial, ya que si existe un exceso de

diversificación, probablemente la metaheurística nunca encontrará una buena solución, y si existe un exceso de intensificación, la metaheurística se quedará atrapada en soluciones similares y no será capaz de encontrar quizás mejores soluciones.

Hay varios tipos de clasificaciones para definir las metaheurísticas [Talbi, 2009]:

- **Inspiradas en la naturaleza versus no inspiradas en la naturaleza**: Existen algunas metaheurísticas que se basan en distintos efectos naturales o sociales, por ejemplo los algoritmos evolutivos. También, algunas se basan en el comportamiento de animales como algoritmos de colonia de abejas o el algoritmo de hormigas, así como también se puede encontrar el *Simulated Annealing* que tiene su origen en una analogía con la física.

- **Iterativa o Greedy**: Al momento de plantear el diseño de una metaheurísticas, se puede representar las etapas de construcción de una solución desde una solución vacía, o con una solución inicial dada. En el caso de un diseño iterativo, las soluciones partirán con un determinado valor y en cada iteración irá recorriendo el espacio de soluciones. En las metaheurísticas *greedy*, se inicia con soluciones vacías y en cada paso del algoritmo se van tomando decisiones que provocan que esta solución vaya tomando forma.

- **Consumidoras de memoria o libres de uso de memoria**: En el caso de metaheurísticas que no utilizan memoria (o muy reducida) en cada paso del algoritmo no es necesario utilizar información de iteraciones anteriores (por lo general sólo de la iteración anterior), aquí caen algoritmos como GRASP o *Simulated Annealing*. Mientras que las metaheurísticas que consumen memoria en cada iteración utilizan información de iteraciones pasadas para tomar una decisión, por ejemplo, la Búsqueda Tabú.

- **Deterministas o estocásticas**: Esto hace referencia a que dependiendo de la metaheurística, en cada iteración se tomarán soluciones aleatorias o deterministas. Por ejemplo, la Búsqueda Tabú tradicional cae en una metaheurística determinista, ya que no existen decisiones aleatorias que tomar, mientras que *Simulated Annealing* es una metaheurística estocástica debido a que dependiendo del estado actual del algoritmo toma decisiones sobre las soluciones.

- **Basadas en población o en individuo único**: Las metaheurísticas pueden considerar un sólo individuo que va siendo modificado iteración a iteración, o un conjunto de individuos que en cada iteración van siendo alterados. La primera permite una alta diversificación buscando soluciones en una mayor porción del espacio mientras que la segunda técnica logra explotar la intensificación mediante búsquedas locales.

### 2.3.1 Ejemplos de Metaheurísticas

Existe actualmente una multitud de metaheurísticas. A continuación se presentarán algunas de las más conocidas de acuerdo a lo desarrollado en [Martí, Pardalos y Resende, 2018] [5].

#### 2.3.1.1 Algoritmo Genético

Los algoritmos genéticos (GA) fueron presentados por primera vez por Jhon Holland en 1975 pero no como algoritmos resolutivos de problemas sino que como instrumentos para la simulación de la evolución biológica en sistemas naturales adaptables [Holland, 1975]. Basan su funcionamiento

[5]Es importante destacar que las metaheurísticas aquí presentadas representan la estructura más básica de las mismas, ya que en la práctica, existen muy pocas implementaciones que emulen completamente los esquemas expuestos debido a que por su naturaleza, las metaheurísticas son y deben ser altamente adaptables según el problema y los requerimientos del usuario.

en la evolución natural de las especies y la teoría evolutiva de Charles Darwin. Sus principales ideas son:

- La supervivencia de una especie viene dado por una competencia entre sus individuos en cuanto a quién es más apto para sobrevivir.
- Los individuos más aptos de la población son los que sobreviven.
- Las descendencias de una generación son combinaciones entre los individuos de la generación actual.
- Eventualmente los individuos podrían sufrir mutaciones lo que modifica su material genético y provoca que se enfrente de una manera distinta al entorno.

Así, la lógica base de los algoritmos genéticos se compone de los siguientes principios:

- Una población se compone de un grupo de individuos, también llamados cromosomas, que representan una solución del problema en cuestión.
- La capacidad del individuo para sobrevivir, o *fitness* (valor de la función objetivo del problema evaluada según el cromosoma) depende de cuán bien se adapte el individuo al problema y este define si dicho individuo tiene mayores probabilidades de sobrevivir.
- Iterativamente en cada generación existen individuos que dejan la población y otros nuevos que ingresan, esto viene dado por su *fitness*.
- En cada generación existen individuos que son elegidos como padres (generalmente aquellos con mejor *fitness*) que dan lugar a hijos mediante combinaciones entre dichos padres, esto permite la intensificación de buenas soluciones.
- Existe también una componente asociada a la mutación que modifica a los individuos de una población. Esto permite una cierta variabilidad en los individuos, y así dar lugar a una diversificación en las soluciones para poder explorar más zonas del universo de soluciones.

Hay diversas variaciones de algoritmos genéticos, lo que da lugar a los algoritmos evolutivos, pero todos siguen la misma estructura base. Además, cada etapa del algoritmo genético se enfrenta a decisiones, las cuales deben ser tomadas por el diseñador del algoritmo y son flexibles según el problema, las principales decisiones son las siguientes:

- **Representación**: Generalmente existen 3 representaciones de un individuo, puede ser binaria, mediante permutación o mediante números reales. La representación binaria puede ser utilizada en un *Knapsack Problem* [6] [Pisinger y Toth, 1998] y se representa como un *string* binario de la forma

  $$[1, 0, 0, 1, 0, 1]$$

  en donde cada 1 representa la decisión de arrojar dicho elemento al saco y cada 0 representa el no hacerlo. La representación mediante permutación puede utilizarse en el *Travelling Salesman Problem* o *TSP* [7] [Applegate et al, 2006] y puede mostrarse como un *string* de valores enteros

[6] El *Knapsack Problem* es un problema de optimización que enfrenta un individuo que debe seleccionar un número de ítems para introducir en su mochila. Cada uno de estos elementos tiene un cierto valor y la mochila tiene una capacidad limitada, por lo que el problema buscar elegir los mejores ítems a introducir en la mochila para que el individuo saque más provecho, a la vez que no sobrepasa la capacidad máxima de la mochila.

[7] El *TSP* es un problema de optimización que busca encontrar la mejor ruta que debe recorrer un individuo para lograr visitar una serie de lugares. En su forma más simple, el individuo debe visitar todos los lugares y el problema busca minimizar el costo asociado a la distancia que se debe recorrer entre los lugares, para así, ubicar la ruta en la que el individuo menos se demora.

de la forma

$$[4, 3, 1, 5, 2, 6]$$

en donde el orden de los valores representa el orden en que se visitarán las ciudades. Es importante notar que esta misma notación podría utilizarse en el *Knapsack Problem* para mostrar qué elementos insertaremos en el saco, por lo que la representación de un problema no es excluyente a otro. Finalmente, la representación mediante valores reales puede utilizarse para resolver un problema con valores reales, por ejemplo, la maximización de una curva, dicha representación puede ser de la forma

$$[3.21, -11, 0.23, 4.2]$$

en donde cada valor corresponde al valor de una coordenada de la función a maximizar.

- **Solución Inicial**: El escoger de que forma se construirá la solución inicial del problema puede llegar a ser un tema crucial del G.A. (y en general de todas las metaheurísticas), ya que el hacerlo de una forma "inteligente" puede hacer que se resuelva el problema más rápido, pero por otro lado, puede tener un costo alto el obtener dicha solución inicial. Por lo general, hay 3 formas de obtener una solución inicial, puede ser completamente aleatoria, aleatoria con diversidad y con heurísticas. Las soluciones completamente aleatorias son las menos costosas, ya que no requieren información previa del problema, pero al mismo tiempo, puede que ninguna otorgue buenas soluciones, lo que provocará que el algoritmo tarde más en encontrar buenos individuos. Las soluciones aleatorias con diversidad se aseguran de que desde un inicio haya soluciones aleatorias, pero lo suficientemente distintas para asegurar la búsqueda en una amplia zona del espacio de soluciones. Esta opción es un poco más costosa que la solución completamente aleatoria, pero existen técnicas que lo hacen en un tiempo prudente, por ejemplo la técnica del hipercubo latino [Iman y Conover, 1982]. Finalmente, las soluciones iniciales con heurísticas son las más costosas, ya que se basan en el diseño previo de una o más heurísticas (por ejemplo una heurística *Greedy*) que permiten hallar buenas soluciones iniciales con el fin de ayudar al algoritmo a converger más rápido. Kazimipour, Li y Qin proponen en [Kazimipour, Li y Qin, 2014] diversas técnicas para generar soluciones iniciales.

- **Evaluación de un individuo**: Generalmente es posible calcular la función objetivo de un problema, pero esto no siempre sucede así, además en algunos casos no es de interés calcular directamente la función objetivo de un problema para evaluar a un individuo, por ejemplo se puede desear calcular penalizaciones en el *TSP* por viajes demasiado largos. Es por esto que existen algunas técnicas que permiten enfrentar estos casos y así lograr obtener valores (aunque no sean los reales) de la función objetivo, por ejemplo evaluaciones según el ranking del *fitness* de un individuo [Whitley, 1989] las que permiten no evaluar directamente a un individuo según su *fitness* sino que según un ranking comparativo entre individuos.

- **Selección**: Existen distintos métodos para seleccionar a los individuos de la población que sobrevivirán a la siguiente generación. El método más simple es simplemente escoger los $P$ mejores individuos de la generación que se convertirán en los futuros padres. Otro método simple es mediante torneo. En este se escoge un subconjunto de $n$ individuos de la población y se selecciona aquel con mayor *fitness* de dicho subconjunto. Posteriormente, se repite este procedimiento con otro subconjunto las veces necesarias hasta obtener el número de padres deseados. Otro método conocido es el *roulette-wheel* que asigna distintas probabilidades a cada individuo según su *fitness* y posteriormente selecciona al azar a los $P$ padres. Con este método los mejores individuos tienen mayores probabilidades de ser seleccionados en la siguiente generación.

- **Orden de las operaciones**: No es directo el decidir si realizar la etapa de la mutación previa a la de cruce (*crossover*) de los padres. Generalmente el procedimiento consiste en seleccionar a los padres y en primer lugar cruzarlos para posteriormente mutar a los hijos resultantes de este procedimiento. Un método alternativo es mutar en primer lugar a los padres y posteriormente cruzarlos para dar lugar a los hijos.

- **Cruce**: Existen varias estrategias de cruce entre individuos $P_1$ y $P_2$. Una de ellas es mediante el *crossover* en un punto, esto se basa en escoger un punto (puede ser el punto medio del vector que define al individuo) y cortar a $P_1$ y $P_2$ en 2 partes cada uno. Así, se da lugar a 2 hijos con una parte de $P_1$ y una parte de $P_2$ y otro hijo con la otra parte de $P_2$ y la otra parte de $P_1$. Otra estrategia de cruce es mediante *uniform crossover*, técnica que consiste en tomar algunos valores aleatorios de $P_1$ y otros valores aleatorios de $P_2$ para dar lugar al hijo. Cabe notar que la selección de la estrategia a utilizar es bastante sensible al problema, por ejemplo, las antes mencionadas pueden generar múltiples individuos infactibles en el *Knapsack Problem* o soluciones sin sentido en el *TSP*. En el caso de este último, es posible utilizar *partially mapped crossover* [Goldberg y Lingle, 1985].

- **Mutación**: Al igual que en el cruce, la mutación es bastante sensible al problema en cuestión. Por ejemplo, en el *Knapsack Problem*, la mutación podría consistir en modificar la decisión de llevar o no un ítem o podría ser el decidir no llevar un ítem y llevar otro, mientras que en el *TSP* la mutación podría basarse en intercambiar el orden en que se visitan 2 ciudades. De cualquier manera, la mutación debiera basarse en modificar los individuos comparado con su estado actual con el fin de explotar la diversificación en la población.

El algoritmo 2 se presenta la estructura básica de un algoritmo genético:

**Algorithm 2** Pseudo-código de un algoritmo genético

$tasaMutacion \leftarrow \mu$
$tasaCrossover \leftarrow \lambda$
$numGeneraciones \leftarrow n$
$poblacion \leftarrow generarPoblacionInicial(TamañoPoblacion)$
**while** $i < numGeneraciones$ **do**
  $fitness \leftarrow calcularFitness(poblacion)$
  $padres \leftarrow seleccion(poblacion, fitness)$
  $hijos \leftarrow crossover(padres, fitness, tasaCrossover)$
  $hijos \leftarrow mutacion(hijos, tasaMutacion)$
  $poblacion \leftarrow padres \cup hijos$
  i++
**end while**
**return** $fittestMember(poblacion)$

#### 2.3.1.2 GRASP

El *greedy randomized adaptive search procedure* o *GRASP* es una heurística *greedy* presentada en 1989 por Mauricio Resende y Thomas Feo en un *seminal paper* [Feo y Resende, 1989] [Feo y Resende, 1995] [Resende y Riberio, 2016]. Se trata de una metaheurística en la que en cada iteración usualmente se utiliza una fase de construcción donde se construye una solución factible sobre la que se realiza una búsqueda local hasta encontrar una buena solución local. Los elementos

especialmente sensibles de la heurística están asociados a la fase de construcción en la cual normalmente se ejecutan algoritmos *greedy* pero también pueden ser utilizados otros métodos.

La estructura típica de un algoritmo *GRASP* para un problema de minimización se presenta en el algoritmo 3:

**Algorithm 3** Pseudo-código de un algoritmo GRASP para un problema de minimización

```
numIteraciones ← n
g() ← greedyFunction
fitness() ← fitnessFunction
x_best ← ∅
fitness(x_best) ← +∞
while i < numIteraciones do
    x ← randomGreedyRandomizedSolution(g())
    if x infactible then
        x ← reparar(x)
    end if
    x ← busquedaLocal(x, fitness())
    if fitness(x) < fitness(x_best) then
        x_best ← x
    end if
    i++
end while
return x_best
```

Como es posible observar, las 2 etapas principales de este algoritmo son tanto la fase de construcción (definida por la función *randomGreedyRandomizedSolution*) y la fase de búsqueda local (definida por la función *randomGreedyRandomizedSolution*)

- *Fase de Construcción*: La fase de construcción se genera a partir de una solución vacía y consiste en que en cada iteración se insertan elementos de acuerdo a una lista de candidatos $C$ ordenada según una función *greedy* $g()$. Esta heurística de construcción se considera adaptable, ya que en cada iteración de la función de construcción los valores de los candidatos $C$ son actualizados según el beneficio que otorga añadirlos sujeto a haber añadido ya algunos miembros. La lista de los mejores candidatos de la iteración recibe el nombre de *restricted candidate list* (*RCL*). El uso de esta lista permite añadir buenos candidatos, pero no necesariamente al mejor candidato posible, y así no se cae en una sobre explotación de la intensificación del algoritmo. Generalmente, la construcción de la *RCL* se hace mediante las *min max-$\alpha$ percentage rules* que utilizan los valores máximos y mínimos de la función $g()$ para así seleccionar a un número estático (cuando se usa un enfoque cardinal) o dinámico (cuando se usa un enfoque no cardinal) de elementos a incorporar en la *RCL*. Prais and Ribeiro en [Prais y Ribeiro, 1999] [Prais y Ribeiro, 2000] analizan distintos métodos de construcción de la *RCL*. El algoritmo 4 describe un pseudocódigo de la fase de construcción.

**Algorithm 4** Pseudo-código de fase de construcción de algoritmo GRASP para un problema de minimización

$enfoqueCardinal \leftarrow True \vee False$
$numeroSeleccionados \leftarrow k$
$parametroSeleccion \leftarrow \alpha$
$Solucion \leftarrow \emptyset$
$g() \leftarrow greedyFunction$
$C \leftarrow candidatos$
$costoIncremental_i \leftarrow g(i, Solucion)\ \forall i \in C$
**while** $C > 0$ **do**
  $g_{min} \leftarrow min(costoIncremental_i)\ \forall i \in C$
  $g_{max} \leftarrow max(costoIncremental_i)\ \forall i \in C$
  **if** $enfoqueCardinal \leftarrow True$ **then**
    $Elements \leftarrow Sorted(C, costoIncremental)$
    $RCL \leftarrow Elements[1, ..., k]$
  **else**
    $RCL \leftarrow \{i \in C | costoIncremental_i \leq g_{min} + parametroSeleccion \cdot (g_{max} - g_{min})\}$
  **end if**
  $randomElement \leftarrow Random(RCL)$
  $Solucion \leftarrow Solucion \cup randomElement$
  $C \leftarrow C - \{randomElement\}$
  $costoIncremental_i \leftarrow g(i, Solucion)\ \forall i \in C$
**end while**
**return** $Solucion$

- *Fase de Búsqueda Local*: La fase de búsqueda local es similar a todos los otros tipos de búsqueda local que se revisarán en este documento. El objetivo de esta es poder modificar levemente las soluciones para asegurar una búsqueda diversificada y así no caer en óptimos locales. El procedimiento consiste en escoger estados vecinos de la solución actual y escoger algún elemento que mejore la solución. El pseudocódigo de esta se muestra en el algoritmo 5.

**Algorithm 5** Pseudo-código de fase de búsqueda local de algoritmo GRASP para un problema de minimización

$Vecinos \leftarrow V(x)$
$f \leftarrow fitnessFunction()$
$Vecinos \leftarrow f(Vecinos)$
$x \leftarrow min(Vecinos)$
**return** $x$

#### 2.3.1.3 Búsqueda Tabú

La Búsqueda Tabú tiene su origen en el *seminal paper* de Fred Glover *Heuristics for Integer Programming Using Surrogate Constraints* [Glover, 1977]. Al igual que *GRASP*, la Búsqueda Tabú es una metaheurística que realiza búsquedas locales. Para permitir una diversificación utiliza 3 conceptos, la memoria a corto plazo, la memoria a medio plazo y la memoria a largo plazo. Un algoritmo de Búsqueda Tabú puede diseñarse utilizando sólo la primera, pero generalmente se utilizan las 2 primeras y en problemas complejos se aconseja utilizar las 3.

- **Memoria a corto plazo**: La memoria a corto plazo es el concepto más conocido de la

Búsqueda Tabú, por lo general la mayor cantidad de estas metaheurísticas incorporan este concepto. Se basa en mantener por varias iteraciones un recuento de ciertos movimientos hechos con anterioridad para asegurarse de no volver a ellos y así no tener que evaluar nuevamente un estado ya evaluado. Esto elemento se conoce como *Lista Tabú*, y su construcción varía según diversas técnicas. El proceso de adición de un elemento a la lista tabú puede darse debido a que es un buen elemento o simplemente por haber sido visitado (esto último consume más memoria), y en cada iteración debiese verificarse no volver a dichos estados. El tamaño de la lista puede ser dinámico o estático. En el caso dinámico el tamaño de la lista tabú varía según evaluaciones dentro del algoritmo, por ejemplo, si se encuentra una nueva solución óptima global, se añade a la lista tabú por determinadas iteraciones, pasadas estas iteraciones este elemento se retira de la lista. Esto permite una menor rigidez del tamaño de la lista, pero al mismo tiempo puede llevar a un mayor consumo de memoria si repetidamente se encuentran óptimos globales o si se decide dejar a los elementos por muchas iteraciones. En el caso del tamaño estático, se le da previamente un largo a la lista tabú, y apenas esta se ve llena y se desea insertar otro elemento, se retira generalmente el elemento más antiguo de la lista.

- **Memoria a medio plazo**: La memoria a medio plazo permite una intensificación de las mejores soluciones encontradas por el algoritmo. La idea consiste en extraer ciertas componentes de las mejores soluciones y luego intensificar la búsqueda en soluciones que tengan también estas componentes. Un método popular se basa en reiniciar la búsqueda con la mejor solución local y luego modificar esta solución con los componentes con mejor desempeño de la solución global.

- **Memoria a largo plazo**: La memoria a largo plazo permite una mayor diversificación en la búsqueda. Su objetivo es forzar el explorar regiones no exploradas del universo de soluciones. Generalmente se utiliza guardando una lista con la frecuencia de visita de un componente de la solución en cada una de las soluciones visitadas. De esta forma, es posible incluir diversas formas de diversificación [Gendreau, 2002]. Por ejemplo, se puede decidir en cierta iteración, insertar en la mejor solución global o local el componente menos visitado y evaluar la nueva solución. También, en cada iteración se podría manejar mediante una tasa de probabilidad $\mu$ la decisión de incluir una de estas componentes en base a su bajo número de visitas.

El algori6 presenta el pseudocódigo de un algoritmo de Búsqueda Tabú:

**Algorithm 6** Pseudo-código de Búsqueda Tabú con tamaño de lista estático para un problema de minimización

$numIteraciones \leftarrow n$
$solucionGlobal \leftarrow \emptyset$
$fitness() \leftarrow fitnessFunction$
$V() \leftarrow vecinosFunction$
$f_{solucionGlobal} \leftarrow +\infty$
$solucionGlobal \leftarrow \emptyset$
$listaTabu \leftarrow \emptyset$
$tamañoMax \leftarrow tamañoMaximoListaTabu$
$x \leftarrow generarSolucionInicial()$
$f_x \leftarrow fitness(x)$
**while** $i < numIteraciones$ **do**
  $vecinos \leftarrow V(x)$
  **for** $v \in vecinos$ **do**
    **if** $v \notin listaTabu$ & $fitness(v) < f_x$ **then**
      $x \leftarrow v$; $f_x \leftarrow fitness(v)$
    **end if**
  **end for**
  **if** $f_x < f_{solucionGlobal}$ **then**
    $solucionGlobal \leftarrow x$
    $f_{solucionGlobal} \leftarrow f_x$
    $listaTabu = listaTabu \cup solucionGlobal$
  **end if**
  **if** $tamaño(listaTabu) > tamañoMax$ **then**
    $listaTabu = listaTabu - \{listaTabu[0]\}$
  **end if**
  i++
**end while**
**return** *solucionGlobal*

#### 2.3.1.4 *Simulated Annealing*

La metaheurística *Simulated Annealing* (SA) se utiliza en problemas de optimización por primera vez en [Cerny, 1985] y [Kirkpatrick, Gelatt y Vecchi, 1983] mediante una aplicación en grafos con el *TSP* y en un diseño a gran escala del mismo problema, respectivamente. Posteriormente, en los años 1980s tuvo un mayor uso debido a su simpleza y su eficiencia para resolver problemas de optimización [Talbi, 2009].

El SA se basa en un principio físico el cual señala que el proceso de enfriamiento requiere en un inicio un calentamiento y luego lentamente enfriar un elemento para obtener una buena estructura cristalina. Si se le da una temperatura inicial no muy alta o si su enfriamiento es muy veloz, la estructura será inestable. El SA simula este proceso mediante el cambio en la energía de un proceso pasando de un estado de mayor calor a uno de menor calor hasta que converge a un estado de equilibrio. Este estado de equilibrio representa la solución óptima del problema a resolver, mientras que óptimos locales representan estructuras inestables.

El algoritmo se considera sin memoria, ya que no utiliza información pasada, sólo va necesitando la temperatura actual del sistema. En cada iteración se mueve la solución actual hacia un estado

vecino, si este vecino mejora la solución se acepta, si no, se utilizan distintos criterios de aceptación de estados en donde se usa la temperatura actual del sistema y la calidad del vecino, denotado por $\Delta E$ que representa la pérdida en la función objetivo de este vecino comparado con el mejor resultado global. A medida que el algoritmo avanza, la probabilidad de escoger estos movimientos va disminuyendo. Generalmente se utiliza el criterio de aceptación según la distribución *Boltzmann* que es de la forma:

$$P(\Delta E, T) = e^{\frac{|\Delta E|}{T}}$$

En un determinado nivel de temperatura pueden darse múltiples evaluaciones de vecinos. Una vez que se alcanza una cierta estabilidad en una región, la temperatura se disminuye según una regla de enfriamiento para así ir aceptando cada vez menos vecinos de forma estocástica. Debido a esto, los 2 aspectos cruciales en el diseño de un algoritmo SA son la probabilidad de aceptación y la regla de enfriamiento:

- **Probabilidad de aceptación**: El objetivo de esta probabilidad es permitir que el algoritmo pueda escapar de óptimos locales aceptando "malos" movimientos. Esta probabilidad en la distribución *Boltzmann* es directamente proporcional a la temperatura e inversamente proporcional al empeoramiento de la función objetivo. A altas temperaturas, la probabilidad de aceptar una mala solución es alta: si $T = \infty$, cualquier movimiento será aceptado, lo que es equivalente a una búsqueda aleatoria de soluciones. Por el contrario, a bajas temperaturas se aceptarán menos soluciones: si $T = 0$, la probabilidad de aceptación será 0 y sólo serán aceptados valores que mejoran la solución global, lo que corresponde a una búsqueda local. Otros criterios de aceptación han sido propuestos, por ejemplo [Ceperley y Dewing, 1999] incluye un término $\sigma_{\Delta E}$ para otorgar mayor variabilidad en la probabilidad de aceptación, o SANE, propuesto en [Branke, Meise y Schmidt, 2008] que incorpora tanto el criterio de *Ceperley* y *Dewing* como el de la distribución *Boltzmann* junto a un muestreo secuencial para decidir si aceptar o no a un movimiento, esto en la búsqueda de criterios más flexibles para enfrentar problemas estocásticos.

- **Regla de enfriamiento**: Esta define en cada iteración el valor de la temperatura. Debido a la probabilidad de aceptación, la temperatura es de vital importancia para asegurar un buen desempeño del algoritmo. Los principales componentes a considerar son la temperatura inicial, el estado de equilibrio, la función de enfriamiento y la temperatura final que define el criterio de término.

  - **Temperatura Inicial**: Si la temperatura inicial es demasiado alta, se aceptarán muchos elementos, lo que impedirá una buena intensificación. Por el contrario, si la temperatura inicial es muy baja, no se aceptarán muchos vecinos y no habrá suficiente diversificación. Para definir una buena solución inicial pueden utilizarse algunos criterios como simplemente utilizar una temperatura inicial alta y así aceptar a un elevado número de soluciones. Esto provocará probablemente una mayor demora en la convergencia del algoritmo, pero se asegurará de moverse en un vasto espacio de soluciones. Otro criterio es utilizar como temperatura inicial el valor:

    $$T_0 = \frac{\Delta}{ln(m_1(a_0 - 1)/m_2 + a_0)}$$

    cuyo objetivo es aceptar soluciones iniciales con un valor mayor a $a_0$. Los valores $m_1$ y $m_2$ vienen dados por el rango esperado de aceptación, por ejemplo entre el 30% y 50% de las soluciones, y $\Delta$ corresponde al valor medio de la función objetivo evaluada en una serie de simulaciones [Aarts y Van Laarhoven, 1987]. Otros criterios son explorados en [Huang, Romeo y Sangiovanni-Vincentelli, 1986] o [Walid, 2004].

- **Estado de Equilibrio**: A cada nivel de temperatura, debiese estabilizarse el algoritmo con el fin de encontrar un estado de equilibrio. Para lograr este equilibrio es necesario visitar un determinado número de vecinos. La teoría sugiere que el número de iteraciones en cada temperatura debiese ser exponencial al tamaño del problema, lo que normalmente es imposible. Así, el número de iteraciones debe ser consecuente con el universo de estados vecinos a visitar. Este número puede ser estático o dinámico. En el caso estático, existe un número $i$ de vecinos a visitar en cada temperatura, si $i$ es más grande, se debiese llegar a mejores resultados pero a un costo computacional alto. En el caso dinámico, en cada iteración se debe tomar la decisión de cuantos vecinos visitar, esta decisión puede tomarse simplemente señalándole que cambie la temperatura apenas mejore el resultado comparado con el estado actual. Otro posible método es ir modificando el número de iteraciones según el mejor y el peor valor encontrado de la función objetivo en la iteración actual [Ali, Torn y Viitanen, 2002].
- **Función de Enfriamiento**: Debido a la probabilidad de aceptación, la forma en que va descendiendo impacta directamente a la calidad del algoritmo. Las principales formas de descenso de la temperatura $T$ son:
  - **Lineal**: La temperatura desciende linealmente en cada iteración de la forma:

    $$T = T - \alpha,$$

    con $\alpha$ la constante de enfriamiento.
  - **Geométrica**: Es la forma más común de enfriamiento y desciende la temperatura de la forma:

    $$T = \alpha \cdot T,$$

    , donde $\alpha$ es un escalar entre 0 y 1, generalmente entre 0.5 y 0.99.
  - **Logarítmica**: Es más lento de aplicar, pero la teoría señala que asegura un óptimo global [Blum, Dan y Seddighin, 2020]:

    $$T_i = \frac{T_0}{log(i)},$$

    donde $T_i$ corresponde a la temperatura de la iteración $i$, $T_0$ corresponde a la temperatura inicial e $i$ corresponde a la iteración actual.
  - **Adaptable**: En el caso de un esquema de enfriamiento adaptable, la tasa de descenso de la temperatura no es constante y depende de información obtenida a lo largo del algoritmo. Se utiliza este esquema cuando se desea realizar pocas evaluaciones a temperaturas altas y muchas iteraciones a temperaturas bajas [Ingber, 1996].
- **Temperatura Final**: Generalmente el SA termina cuando se alcanza una temperatura igual a 0 (no siempre se logra así, por lo que se utiliza un valor bajo, por ejemplo $T_F$=0.001). Pero también pueden utilizarse otros criterios como detener el algoritmo cuando han pasado un número suficiente de iteraciones sin mejorar la solución global. También, se puede decidir detener cuando se han recorrido predeterminados niveles de temperatura [Johnson et al, 1989].

El algoritmo 7 presenta un pseudocódigo de una metaheurística SA:

**Algorithm 7** Pseudo-código de *Simulated Annealing* para un problema de minimización

$numIteraciones \leftarrow n$
$solucionGlobal \leftarrow \emptyset$
$fitness() \leftarrow fitnessFunction$
$V() \leftarrow vecinosFunction$
$fitness_{solucionGlobal} \leftarrow +\infty$
$T \leftarrow temperaturaInicial$
$enfriamiento() \leftarrow funcionEnfriamiento$
$x \leftarrow generarSolucionInicial()$
$f_x \leftarrow fitness(x)$
**while** $i < numIteraciones$ **do**
  $vecinos \leftarrow V(x)$
  $v \leftarrow Random(vecinos)$
  $T_i \leftarrow enfriamiento(i, T)$
  **if** $fitness(v) < f_x$ **then**
    $x \leftarrow v$
    $f_x \leftarrow fitness(v)$
  **else**
    **if** $e^{\frac{|fitness(v)-f_x|}{T_i}} > Random(0,1)$ **then**
      $x \leftarrow v$
      $f_x \leftarrow fitness(v)$
    **end if**
  **end if**
  **if** $fitness(v) < f_{solucionGlobal}$ **then**
    $solucionGlobal \leftarrow v$
    $f_{solucionGlobal} \leftarrow fitness(v)$
  **end if**
  i++
**end while**
**return** *solucionGlobal*

#### 2.3.1.5 Algoritmos de Enjambre de Partículas

Los algoritmos de optimización mediante enjambre de partículas (*Particle Swarm Optimization* o PSO) fueron creados por Russell Eberhart y James Kennedy [Eberhart y Kennedy, 1995] [Kennedy y Eberhart, 1995] y se basan en el comportamiento colectivo de algunos organismos como hormigas, abejas, peces, libélulas, entre otros. En su teoría incluyen también conceptos de la física de las partículas, así como también sobre procesos estocásticos y la teoría de las probabilidades.

Se basan en el uso de partículas como elementos de búsqueda que se mueven constantemente en el espacio de soluciones. Los estados de estas partículas otorgan información para iteraciones futuras en las que dicha información se utiliza para dar movimientos hacia zonas en las que existe una mayor probabilidad de encontrar buenas soluciones.

El esquema básico de un PSO se compone de un conjunto de partículas las cuales poseen una serie de dimensiones (también llamadas componentes del problema [8]). Cada una de estas partículas,

[8]Componente del problema hace referencia a una determinada coordenada del universo de soluciones, por ejemplo en el *Knapsack Problem* una componente es un valor asociado al elemento $i$, con $i$ un determinado elemento del universo de probables elementos a insertar al saco

en cada una de sus componentes, guardan 4 datos, estos son su posición actual, su velocidad, su mejor posición y su vecindad:

- **Posición actual**: Define el estado actual de la partícula en una determinada componente. Para términos del problema representa una coordenada de la solución, por lo que en cada iteración existirá un universo de $N \cdot D$ componentes (siendo $N$ el número de partículas y $D$ el número de componentes) las que variarán su estado de acuerdo a las propiedades del algoritmo.
- **Velocidad**: La velocidad define el movimiento de la partícula, mientras mayor sea la velocidad, mayor será el movimiento que se realiza. Por lo general, se limita sus valores según ciertos parámetros $v_{min}$ y $v_{max}$ que se señalan el valor mínimo y máximo que puede tomar este parámetro, respectivamente.
- **Mejor posición**: Este concepto representa la mejor posición en la que ha estado la partícula a lo largo de sus dimensiones. Su función es sesgar los movimientos futuros hacia zonas que se espera tengan un mejor comportamiento. Sin embargo, tener sólo esta información probablemente sesgará en demasía a la partícula y limitará su movimiento hacia zonas en que tan solo ella ha tenido buenos resultados. Es por esto que se incluye el siguiente concepto de esta lista.
- **Vecindad**: El objetivo de la vecindad es intercambiar información entre las partículas para realizar movimientos no tan solo basado en su propia mejor posición sino que también en base a la mejor posición de otras partículas. Al igual que otras metaheurísticas, normalmente se escoge un conjunto pequeño de posibles vecinos por temas de capacidad computacional.

Considerando estos conceptos, el modelo básico del PSO modifica la posición de una partícula en cierta dimensión con las siguientes fórmulas,

$$
\begin{aligned}
v_{i,j}^{t+1} &= v_{i,j}^{t} + C_1 \cdot (p_{i,j}^{t} - x_{i,j}^{t}) + C_2 \cdot (p_{b_i,j}^{t} - x_{i,j}^{t}) \\
x_{i,j}^{t+1} &= x_{i,j}^{t} + v_{i,j}^{t+1} \\
p_i^t &= \begin{cases} x_i^{t+1}, \text{ si } f(x_i^{t+1}) \leq f(p_i^t) \\ p_i, \text{ si no} \end{cases}
\end{aligned}
$$

en donde $x_{i,j}^t$ representa la posición de la partícula $i$ en la dimensión $j$ en la iteración $t$; $v_{i,j}^t$ representa la velocidad de la partícula $i$ en la dimensión $j$ en la iteración $t$; $p_{i,j}^t$ representa la mejor posición que haya alcanzado la componente $j$ de la partícula $i$ hasta la iteración $t$; y $p_{b_i,j}^t$ representa la mejor posición que haya alcanzado la componente $j$ a lo largo de todas las partículas hasta la iteración $t$. La actualización del término $p_i$ está ejemplificada para un problema de minimización. Las componentes $C_1$ y $C_2$ son las componentes estocásticas del problema y son calculadas en base a una distribución uniforme: $C_1 = \mathbf{U}(0, c_1)$ y $C_2 = \mathbf{U}(0, c_2)$, donde $c_1$ y $c_2$ son llamadas las *constantes de aceleración* y definen la magnitud del cambio en la velocidad debido a $p_{i,j}^t$ y $p_{b_i,j}^t$ respectivamente. Dichas constantes son ajustadas por el usuario. El término $(p_{i,j}^t - x_{i,j}^t)$ es conocido como el *término cognitivo* y representa el cambio en la posición de la componente debido a la información dada por la partícula en sí misma, y el término $(p_{b_i,j}^t - x_{i,j}^t)$ es conocido como el *término social* y representa la variación en la componente debido a la información otorgada por el enjambre.

El pseudocódigo que explica el modelo base de un PSO se presenta en el algoritmo 8:

**Algorithm 8** Pseudo-código de algoritmo de enjambre de partículas

```
numIteraciones ← I
numParticulas ← n
numDimensiones ← d
actualizarVel() ← funcionActualizarVelocidad
actualizarPos() ← funcionActualizarPosicion
x_{i,j} ← solucionesInicial()
v_{i,j} ← velocidadesInicial()
p_i ← 0
while h < numIteraciones do
    for i : 1...numParticulas do
        for j : 1...numDimensiones do
            v_{i,j} ← actualizarVel(v_{i,j})
            if v_{i,j} > v_max ∨ v_{i,j} < v_min then
                v_{i,j} ← corregir(v_{i,j})
            end if
            x_{i,j} ← actualizarPos(v_{i,j})
            if x_{i,j} infactible then
                x_{i,j} ← corregir(x_{i,j})
            end if
        end for
    end for
    for i : 1...numParticulas do
        p_i ← actualizarMejor(x_{i,j})
    end for
    h++
end while
return Best((x_{i,j}))
```

#### 2.3.1.6 Algoritmo de Colonia de Abejas

El algoritmo de colonia de abejas es una metaheurística estocástica que pertenece a la categoría de algoritmos de enjambre de partículas o PSO. El algoritmo surge en el año 2005 y fue creado por Dervis Karaboga a partir del comportamiento que presentan las colonias de abejas [Karaboga, 2005]. A partir de este año, diversas variaciones del algoritmo se han utilizado para resolver problemas combinatoriales y en optimización continua [Bitam, Batouche y Talbi, 2008]. Las cualidades fundamentales que utiliza el algoritmo para emular el comportamiento de las abejas son la búsqueda de comida y la búsqueda de ubicación de panal. El algoritmo *marriage in honeybees optimization* o *MBO* [Abbas, 2001] se inspira también en el comportamiento del matrimonio dentro de una colonia de abejas. Para comprender el algoritmo de abejas se deben definir algunos conceptos en la naturaleza de las abejas.

Una colonia de abejas posee entre 60.000 y 80.000 miembros, cuya misión principal es sobrevivir mediante la recolección y producción de miel. La colonia generalmente posee una única abeja reina que vive entre 3 a 5 años. Pasado este tiempo, se selecciona dentro de un grupo selecto de huevos a la nueva abeja reina la que es alimentada con jalea real, mientras que al mismo tiempo matan a la antigua reina. La abeja reina dará lugar a aproximadamente 2.000 huevos diarios, de los que nacerán zánganos y obreras. Los zánganos en una colonia son entre 300 y 3.000 miembros que corresponden a huevos no fertilizados de la abeja reina y su función consiste en fertilizar a la abeja

reina para dar lugar a otros huevos. Las abejas exploradoras, son las abejas más conocidas, ya que son las que más frecuentemente son observables, su número en la colmena varía según la estación, desde 30.000 en invierno, hasta 80.000 en el verano. Estas abejas buscan alimento (polen y néctar de las flores), construyen y protegen la colmena , limpian, hacen correr el aire batiendo sus alas, entre otras tareas [Snodgrass, 1953].

Los principales procesos que ocurren en una colmena son los siguientes:

- **Búsqueda de ubicación para panal**: A finales de la primavera o inicios del verano se divide la población en 2 mitades. Un grupo corresponde a la reina, junto con la larva reina y la mitad de las obreras, y el otro grupo corresponde a la otra mitad de las obreras. El primer grupo va en búsquedas de nuevos sitios para ubicar un panal, y la segunda mitad se encarga de producir miel en el panal actual. Muchas abejas obreras son enviadas a distintas zonas para evaluar la ubicación de un nuevo panal y se comparten dicha información entre ellas mediante patrones llamados *danza del meneo* en donde la forma y la velocidad del movimiento son las variables para evaluar la calidad del panal. Posteriormente, estas abejas se reúnen para llegar un consenso sobre el mejor sitio encontrado para luego informar al resto del grupo sobre dicha decisión.

- **Búsqueda de comida**: La búsqueda de lugares con buen néctar y polen se realiza con 2 tipos de abejas: desempleadas (o exploradoras) y empleadas. Las primeras son enviadas a lugares sin información y deben ser capaces de ubicar buenas zonas en donde explotar recursos. Para esto se comunican con la *danza del meneo* para evaluar estos lugares. Las abejas empleadas son enviadas a determinados sitios en donde se sabe que hay comida. En el momento en que este sitio se queda sin comida, la abeja se convierte en desempleada, por el contrario, si el sitio tiene demasiado néctar, las abejas comunican esta información y más abejas son enviadas al lugar. El método de comunicación es similar al que utilizan durante la búsqueda de panal. Las abejas desempleadas van a sitios en búsqueda de comida y al regresar al panal informan sobre el lugar a otras abejas, mientras que las abejas empleadas al visitar un lugar para extraer néctar deben ser capaces de recordar el sitio al que fue enviada, recoger el néctar y evaluar la cantidad de néctar que posee dicho sitio, para que así al volver al panal puedan compartir dicha información al resto de la colonia con el fin de ó enviar más abejas, ó convertirse en una abeja desempleada.

  Este comportamiento es el concepto clave en el diseño del algoritmo de colonia de abejas, ya que estos dos métodos se utilizan como una analogía entre la diversificación e intensificación del espacio de soluciones.

El algoritmo se construye con un universo de $n$ abejas desempleadas que van en búsqueda de distintas soluciones mediante una búsqueda aleatoria. Cada abeja se encarga de evaluar $m_n$ distintos estados vecinos para realizar búsquedas locales. De aquí saldrán los $e_{m,n}$ mejores lugares de cada grupo $m_n$ que representarán la mejor valoración de cada abeja. Este proceso puede hacerse de forma directa (simplemente tomar el máximo valor de $e_{m,n}$) o en base a probabilidades, en donde los mejores estados tendrán una mayor probabilidad de ser escogidos como los representantes de la abeja $n$. Al mismo tiempo, otro grupo de abejas se encarga de realizar búsquedas aleatorias para así no caer en óptimos locales.

El algoritmo 9 representa esta técnica:

**Algorithm 9** Pseudo-código de Algoritmo de Abejas

$numAbejas \leftarrow n$
$numVecinos \leftarrow m$
$numRepresentantes \leftarrow e$
$abejasDesempleadas \leftarrow ds$
$abejasEmpleadas \leftarrow numAbejas - abejasDesempleadas$
$numIteraciones \leftarrow I$
$fitness() \leftarrow fitnessFunction$
$actPobl() \leftarrow ActualizarPoblacion$
$V() \leftarrow vecinosFunction$
$poblacion \leftarrow solucionInicialRandom(numAbejas)$
$f \leftarrow fitness(poblacion)$
**while** $i < numIteraciones$ **do**
  **for** $j : 1...abejasEmpleadas$ **do**
    $vecinos_j \leftarrow V(poblacion, numVecinos)$
    $v_j \leftarrow mejorVecino(fitness(vecinos_j))$
    $poblacion \leftarrow actPobl(poblacion, v_j)$
  **end for**
  **for** $k : 1...abejasDesempleadas$ **do**
    $sites_k \leftarrow randomSolution()$
    $v_k \leftarrow mejorVecino(fitness(sites_j))$
    $poblacion \leftarrow actPobl(poblacion, v_k)$
  **end for**
  $f \leftarrow fitness(poblacion)$
  i++
**end while**
**return** $Max(fitness(poblacion))$

#### 2.3.1.7 Algoritmo de Colonia de Hormigas

El algoritmo de colonia de hormigas es una metaheurística que al igual que el algoritmo de colonia de abejas cae en la clasificación de algoritmos de enjambre de partículas o PSO. Este algoritmo construye soluciones mediante métodos probabilísticos. Para lograrlo usa 2 técnicas informativas: información heurística dada directamente por el problema, así como también una información asociada a la analogía con las hormigas, en particular, los *senderos artificiales de feromonas* los cuales corresponden a un método de aprendizaje que intenta identificar las componentes que guían a soluciones de alta calidad [Dorigo, 2007] [Dorigo y Stützle, 2004].

Fue inspirado por experimentos realizados en colonias de hormigas. En dichos experimentos se descubrió que las hormigas son capaces de encontrar el camino más corto entre su colonia y una fuente de comida mediante un mecanismo activado gracias al sendero de feromonas que dejan otras hormigas y su comportamiento [Deneubourg et al, 1990].

Al igual que GRASP, el algoritmo de colonia de hormigas originalmente no parte con una solución inicial sino que construye soluciones en cada iteración, lo que no es común en las metaheurísticas. Pero a diferencia de GRASP, este algoritmo trabaja en base a una población (como GA) y construye las soluciones mediante esta feromona, la cual es actualizada dinámicamente. Los aspectos más importantes de este algoritmo son:

- **Información heurística y feromona**: La información heurística en un algoritmo de hormi-

gas asigna a cada componente del problema un determinado valor. Este valor puede ser asignado de forma estática (al inicio del problema) o de forma dinámica utilizando información extraída a lo largo del algoritmo, aunque su cálculo debiese ser rápido, ya que es necesario dicho valor en cada iteración.

La feromona al igual que la información heurística es un valor que se le asigna a cada componente del problema. Esto se realiza dinámicamente a lo largo del algoritmo de acuerdo a la calidad de las soluciones mediante las técnicas de depósito de feromona y evaporación de feromona. El objetivo de la feromona es ayudar a que las soluciones generadas vayan teniendo en mayor medida ciertas componentes en desmedro de otras. Un mayor valor de feromona representa una mayor probabilidad de insertar determinada componente en la solución.

- **Construcción de la solución**: En cada iteración del algoritmo son generadas varias soluciones en función del número de hormigas dado por el usuario, por lo que al inicio de la iteración se presenta un conjunto de soluciones iniciales vacía y en cada paso de construcción un elemento es añadido a cada solución mediante una cierta probabilidad calculada en base a la feromona y la información heurística. Generalmente esta probabilidad es tomada de acuerdo a la presentada en la metaheurística *AS* [Dorigo, 1992] y es de la forma:

$$Pr(c_j) = \frac{\tau_j^\alpha \cdot \nu_j^\beta}{\sum_{k=1}^{|N_i|} \tau_k^\alpha \cdot \nu_k^\beta} \ \ \forall j \in \{1, ..., |N_i|\},$$

en donde $c_j$ corresponde a la componente $j$, $N_i$ corresponde al conjunto de componentes candidatas en la fase de construcción $i$; $\tau_h$ corresponde a la feromona del individuo $h$; $\nu_h$ es la información heurística del individuo $h$; y $\alpha$ y $\beta$ son parámetros de control de la importancia relativa de la feromona y la información heurística respectivamente. Otros métodos para el cálculo de esta probabilidad pueden ser utilizados [Maniezzo, 1999] [Maniezzo y Carbonaro, 2000] [Dorigo y Gambardella, 1997].

La construcción de la solución es un paso muy sensible y que puede tomar mucho tiempo en problemas muy grandes. Es por esto que es posible utilizar una lista de "mejores candidatos" (similar a la *RCL* del algoritmo GRASP) para no evaluar a todos los candidatos sino que a un set de candidatos prometedores según la estructura del problema (por ejemplo, en el *Travelling Salesman Problem* utilizar como mejores candidatos a un set de ciudades cercanas ya añadidas a la solución parcial) [Dorigo y Di Caro, 1999]. Otro método alternativo respecto a la solución clásica es no iniciar la construcción con soluciones vacías, sino que con soluciones parcialmente construidas para en primer lugar ahorrar algunos pasos en la fase de construcción y así tomar menos tiempo, y en segundo lugar, para destruir parcialmente una buena solución y a partir de esta crear en teoría una mejor [Acan, 2004] [Tsutsui, 2007] [Wiesemann y Stützle, 2006].

- **Actualización global de la feromona**: La feromona se actualiza en cada iteración mediante una evaporación y una deposición de feromona.
  - **Evaporación**: Busca ayudar a olvidar el hecho de que algunas componentes tengan un bajo valor reduciendo el efecto de la feromona previa. Este factor puede ser aplicado a todas las componentes o solamente a algunas. Generalmente se utiliza un factor fijo para todas las componentes de la forma

$$\tau_j = (1 - \rho) \cdot \tau_j \ \ \forall j \in \{1, ..., |N_i|\}$$

    con $\tau_j$ la feromona de la componente $j$, $N_i$ el conjunto de feromonas de las componentes

en la fase de construcción $i$ y $\rho$ el factor de evaporación. Mientras mayor sea $\rho$, mayor evaporación habrá. Existen variaciones del algoritmo que aplica esta evaporación a un conjunto $S \subset N$, este conjunto $S$ lo define arbitrariamente el usuario. Por ejemplo, en el *Ant Colony System* [Dorigo y Gambardella, 1997], el conjunto $S$ se define como el conjunto de componentes para las cuales en la iteración actual existió deposición de feromonas.

– **Deposición**: Su objetivo es incrementar la feromona de las componentes de ciertas soluciones. Este conjunto de soluciones varía entre algoritmos, por ejemplo los algoritmos más simples (*Ant System*) escogen a todo el universo de soluciones construidas en la iteración. Otros métodos sólo actualizan la feromona de las componentes de la mejor solución de la iteración. Un paso intermedio es escoger algunas de las mejores soluciones de la iteración [Dorigo y Stützle, 2004]. Existen también algunos métodos más sofisticados para la creación de este conjunto [Bullnheimer, Hartl y Strauss, 1999] o [Stützle y Hoos, 2000], así como también técnicas dinámicas en las que se usan distintos criterios a lo largo del algoritmo para definir el subconjunto de soluciones ( [Stützle, 1998] [Merkle y Middendorf, 2001]).

En general, se utiliza una fórmula del tipo

$$\tau_j = \tau_j + \sum_{s_k \in S^* | c_j \in s_k} w_k \cdot F(s_k) \;\; \forall j \in \{1, ..., |N_i|\}$$

con $N_i$ el set de componentes candidatas en la fase de construcción $i$, $\tau_j$ la feromona de la componente $j$, $S^*$ el subconjunto de soluciones seleccionadas para actualizar la feromona de sus componentes, $s_k$ la solución $k$ del conjunto $S^*$, $c_j$ la componente $j$ de la solución $s_k$, $w_k$ un escalar asociado a la solución $s_k$ y $F(s_k)$ una función directamente proporcional a la calidad de la solución $s_k$. Hay distintas formulaciones de la función $w_k \cdot F(s_k)$, la más común es:

$$w_k \cdot F(s_k) = 1/f(s_k)$$

para problemas de minimización (el inverso en caso de maximización), con $f(s_k)$ el $fitness$ de la solución $s_k$. El ya mencionado *Ant Colony System* utiliza la forma:

$$w_k \cdot F(s_k) = \rho/f(s_k)$$

donde $\rho$ corresponde a la tasa de evaporación de feromona. Otros métodos son utilizados en [Dorigo et al, 1991a], [Dorigo et al, 1996] o [Cordón et al, 2000].

- **Valor inicial de las feromonas**: En general se utilizan 2 posibilidades como valor inicial de feromonas: un valor muy pequeño (como en el *Ant Colony System*) o uno muy grande (*MMAS* [Stützle y Hoos, 2000]), en donde pequeño y grande son valores relativos a la cantidad de feromona depositada en la actualización global de acuerdo al término $w_k \cdot F(s_k)$ de la fórmula. Valores pequeños conllevan a sesgos más grandes en la decisión de elección de candidatos, mientras que valores grandes provocan una exploración inicial más grande. En general estos valores son dejados en libertad para ser especificados por el usuario.

- **Reinicio de las feromonas**: La literatura ha demostrado que reiniciar las feromonas a un determinado nivel ayuda a la diversificación al moverse en más zonas del universo de soluciones [Stützle y Hoos, 2000]. Esto es equivalente a inicializar nuevamente el algoritmo, pero guardando en la memoria la mejor solución global. Generalmente se utiliza esta técnica en problemas como el *TSP* en donde una intensa búsqueda global ayuda a una rápida convergencia.

El *MMAS* utiliza el reinicio de feromonas calculando una métrica llamada *branching factor* que representa el número de alternativas potenciales guardadas en el sendero de feromona. Cuando este valor desciende de un determinado valor, las feromonas son reiniciadas a su máximo valor. Para evitar un reinicio muy seguido, se prohíbe reiniciar la feromona por un determinado número de iteraciones.

- **Actualización local de la feromona**: Este método modifica el valor de la feromona de una componente durante la fase de construcción. Esta técnica es de difícil implementación ya que requiere algunas modificaciones con respecto al algoritmo clásico, una de ellas es que no tiene evaporación de feromona. Los resultados tampoco han sido muy efectivos [Colorni, Dorigo y Maniezzo, 1992] [Dorigo, Maniezzo y Colorni, 1991b] y se debe ser cuidadoso cuando se implementa una construcción de hormigas en paralelo y no secuencialmente, ya que la primera conlleva más complicaciones para la implementación de esta técnica. Un algoritmo que utiliza este método dando resultados es el *Ant Colony System* que la usa de la siguiente forma:
$$\tau_j = (1 - \epsilon) \cdot \tau_j + \epsilon \cdot \tau_0$$
con $\tau_j$ el valor de la feromona de la componente $c_j$, $\tau_0$ el valor inicial de la feromona (que debe ser mucho menor que el típico valor inicial) y $\epsilon$ un parámetro que controla la medida de actualización local.

- **Límites de los valores de feromona**: Este criterio fue incluido en el algoritmo *MMAS* e impide que las feromonas tomen valores fuera de un cierto rango generalmente dinámico, esto con el propósito de que las feromonas no tomen valores tan bajos que provoquen que una componente nunca sea seleccionada y que tampoco tomen valores tan altos que provoque que una componente nunca salga del universo de soluciones.

- **Búsqueda local**: Luego de la construcción de las soluciones es posible realizar una búsqueda local recorriendo soluciones vecinas de las construidas, para así explorar mejores soluciones dentro de una determinada vecindad. Es importante tomar la decisión sobre el modo en que se hará esta búsqueda local, una opción es hacerla a todas las soluciones construidas (opción más cara pero con mejores resultados), a las mejores soluciones (opción relativamente más barata y con resultados medios) o sólo a la mejor solución construida (opción más barata pero probablemente con peores resultados).

En el algoritmo 10 se presenta un pseudocódigo del algoritmo de colonia de hormigas.

**Algorithm 10** Pseudo-código de Algoritmo de Hormigas

```
numIteraciones ← I
numHormigas ← n
fitness() ← fitnessFunction
V() ← vecinosFunction
construirSolucion() ← funcionConstruccion()
evaporacion() ← funcionEvaporacion()
deposicion() ← funcionDeposicion()
feromonas ← inicializarFeromonas()
while i < numIteraciones do
    for j : 1...numHormigas do
        solucion_j ← construirSolucion(feromonas)
    end for
    for k : 1...numHormigas do
        vecinos_k ← V(solucion_j)
        solucion_k ← Max(fitness(vecinos_j))
    end for
    feromonas ← evaporacion(feromonas)
    feromonas ← deposicion(feromonas)
    i++
end while
return Max(fitness(solucion))
```

#### 2.3.1.8 Otras metaheurísticas

Debido a la incapacidad de exponer todo el universo existente de metaheurísticas (incluso la literatura dedicada a la enseñanza de estas técnicas es incapaz de exponerlas), este marco teórico se enfocó en algunas de las más conocidas. Algunas otras metaheurísticas conocidas y que no fueron desarrolladas aquí son: *Memetic Algorithms* [Moscato y Cotta, 2003], *POPMUSIC* o *partial optimization metaheuristic under special intensification conditions* [Taillard y Voß, 2002], *Scatter Search* [Laguna y Martí, 2002], *Iterated Local Search* [Lourenco, Martin y Stutzle, 2002], *Differential Evolution* [Price, Storn y Lampinen, 2006], *Artificial Inmune Systems* [Hunt y Cooke, 1995], Algoritmo de libélulas [Yang, 2008], etc.

### 2.3.2 Metaheurísticas en ambientes estocásticos

La complejidad del problema abordado en este trabajo hace menester el utilizar una metaheurística capaz de resolver problemas estocásticos. Se han probado diferentes modificaciones de metaheurísticas para resolver problemas estocásticos, por ejemplo en [Bianchi et al, 2009] se analiza un enfoque estocástico en el algoritmo de colonia de hormigas, *Simulated Annealing*, Búsqueda Tabú y Computación evolucionaria. En [Juan et al, 2021] se revisa el desempeño de distintas heurísticas en programación estocástica. Algunos ejemplos de metaheurísticas aplicadas a un problema estocástico, en este caso el *vehicle routing problem* (VRP) [9] son:

- En [Mendoza et al, 2015] se utiliza GRASP para resolver un VRP con una demanda estocástica. El método utilizado ocupa un set de rutas aleatorias y un enfoque heurístico para generar

[9] El VRP es un problema que busca distribuir de forma óptima la ruta de una serie de vehículos para satisfacer la demanda de los clientes mientras se minimiza el costo total de enviar la flota.

soluciones en la etapa de construcción. Para la búsqueda local, se utiliza un procedimiento de descenso de vecinos.

- En [Ferone et al, 2018] también se utiliza GRASP en un VRP con demanda estocástica. En este modelo, se sesga la probabilidad de elección en la *RCL* por sobre la probabilidad uniforme que utiliza tradicionalmente esta etapa. Además, utilizan un enfoque de 2 etapas para la evaluación de las soluciones. En una primera etapa, los óptimos locales obtenidos a lo largo de todas las iteraciones de GRASP son evaluados en pequeñas simulaciones. Luego, sólo el mejor óptimo local obtenido en estas evaluaciones es evaluado completamente lo que entrega una solución "más real" del método.

- En [Gendreau et al, 1994] se utiliza Búsqueda Tabú para resolver también un VRP. El mayor aporte de este trabajo es que crean un enfoque para calcular de manera más rápida la función objetivo. Para la definición de un estado vecino utilizan la inserción de rutas y el tamaño de la lista tabú es dado aleatoriamente dentro de un intervalo predefinido de acuerdo al tamaño del problema.

- En [Li y Li, 2020] se resuelve un VRP con ventanas de tiempo utilizando Búsqueda Tabú. Se otorga una solución inicial con un enfoque *greedy* y posteriormente se crean estados vecinos mediante 3 alternativas: 2-opt [10], *swap*[11] y *reallocate* [12].

- En [Mak y Guo, 2004] se resuelve un VRP con un algoritmo genético modificado para resolver un problema estocástico. En este caso, se utiliza lo que llaman *Age-GA* en el cual los individuos no son reemplazados por los hijos, sino que pueden seguir generando hijos. Un individuo deja de pertenecer a la población después de un determinado número de iteraciones, dado por un parámetro llamado *tasa de supervivencia*. De la misma forma, el número de individuos en cada generación viene dado por el parámetro *tasa de nacimiento*. Así, el tamaño de la población es dinámico y se compone de individuos de distinta edad, lo que permite que los individuos puedan otorgar información en más de una generación.

Una gran complejidad que se enfrenta al resolver problemas estocásticos es que en ocasiones el tiempo que toma resolver una instancia puede ser muy alto. Una alternativa es aproximar la función objetivo y luego modificar la metaheurística reemplazando la función objetivo por la versión aproximada. Esta técnica es conocida como *meta-modeling*. Algunas técnicas para modelar la función objetivo de problemas de optimización emplean Redes Neuronales [Reed y Marks, 1999], modelos *Kriging* [Cressie, 1993], *Support Vector Machine* [Vapnik, 1998], entre otros [Powell, 1987] [Santner et al, 2003].

A pesar de esto, en algunos problemas una solución analítica de la función objetivo es imposible de calcular. En estos casos la función objetivo se aproxima mediante una serie de soluciones creadas mediante experimentos físicos o simulaciones. Por ejemplo, [Costa y Silver, 1998] utilizan *Noise Tabu Search* (NTS) para buscar soluciones rápidamente saltándose el cálculo de la función objetivo. Sus resultados muestran que la Búsqueda Tabú convencional toma más tiempo, pero su solución es solamente un poco mejor que NTS. A pesar de esto, recomiendan que si es posible calcular de forma cerrada el valor de la función objetivo del problema, entonces es preferible calcularlo (aún si es costoso computacionalmente) que utilizar NTS.

[10] Método que intercambia 2 arcos de un grafo entre sí.
[11] Método que intercambia 2 nodos adyacentes entre sí.
[12] Método que ubica un nodo en otro lugar de la ruta.

### 2.3.3 Comparación de metaheurísticas

Finalmente, cabe mencionar el *No-Free-Lunch Theorem* propuesto por David Wolpert y William Macready [Wolpert y Macready, 1997], el cual señala que si bien un algoritmo puede funcionar de manera excepcional en un problema y superar a todos los otros algoritmos, dicho algoritmo tendrá un desempeño menor comparado con otro algoritmo en todos los otros problemas. Debido a esto, concluyen que todos los algoritmos de *black-box optimization* (aplicable también a metaheurísticas) tendrán exactamente el mismo desempeño. Gracias a esto, 2 aspectos se deben considerar de cara a este trabajo:

1. Si bien una metaheurística $A$ ha funcionado históricamente bien para un problema $P$, esto no indica que la misma metaheurística $A$ funcionará bien para un problema $P'$. Es más, puede haber una metaheurística $B$ que supere en todo sentido a la metaheurística $A$ en el problema $P'$.

2. Una determinada configuración de los parámetros del algoritmo $A$ no necesariamente funcionará igual de bien o mal en el problema $P$ que en el problema $P'$.

# 3 Problema y Estado del Arte

## 3.1 Descripción del Problema

La ubicación de cortafuegos en un bosque es un método que se utiliza con el objetivo de ser una barrera que disminuya el impacto de un incendio forestal, siendo su efecto ampliamente investigado [Pinto et al, 2020] [Lin, Liu y Huang, 2021] [Curran et al, 2018] [Bachinger et al, 2016]. Estos cortafuegos pueden ser de origen natural, como un río, un lago, un terreno sin vegetación o una zona rocosa, o pueden ser de origen humano, mediante carreteras construidas, modificaciones al terreno o reforestación con especies menos inflamables [Xinglei et al, 2019]. En este trabajo se revisará el efecto de utilizar metaheurísticas para ubicar de buena manera cortafuegos en el terreno en la forma de tratamientos al combustible existente.

El costo de ubicar y mantener un cortafuego es considerable, ya que operacionalmente involucra un equipo de personas que deben encargarse de planificar, ejecutar y mantener las modificaciones al terreno, además de un costo social al modificar el paisaje, o debido a las molestias a los habitantes de la zona circundante al bosque. Sumado a esto, el modificar un bosque conlleva también un impacto ecológico tanto por la biomasa perdida del bosque como por los efectos que podrían alterar el comportamiento y vida de distintos organismos que habitan el bosque [Depietri y Orenstein, 2021]. Obviamente, estos efectos debiesen ser comparados con los efectos que genera un incendio en esta misma zona. Es por esto, que la decisión de incorporar cortafuegos a una zona no es trivial.

En este trabajo se normaliza el costo de ubicar un cortafuego a un valor unitario y se omiten otros costos positivos y negativos asociados a tanto la generación de un cortafuego como las consecuencias de un incendio. Para controlar la cantidad de cortafuegos a ubicar sólo se permite poner a lo más $P$ celdas cortafuegos, con $P$ un valor correspondiente a entre un 5% y 10% de las celdas totales del bosque, valores normalmente utilizados en el combate forestal.

Como ya se mencionó, el objetivo de esta investigación es encontrar una buena planificación en la ubicación de cortafuegos en un bosque mediante el uso de metaheurísticas. Para esto se plantea el siguiente problema de optimización:

$$\begin{aligned} \min \quad & BosqueQuemado(c) \\ \text{subject to} \quad & |c| \leq P \end{aligned} \tag{3.1}$$

Donde $BosqueQuemado(c)$ corresponde a la porción de bosque que se incendia en un escenario con determinadas condiciones dadas por las variables $c$ y parámetros medioambientales al que se ve enfrentado el bosque (dirección y magnitud del viento, temperatura ambiental, topografía del terreno, tipo de pasto de determinada zona del bosque, ROS, punto de ignición del incendio, etc.). Mientras $c$ corresponde a la configuración de cortafuegos otorgada al bosque y $P$ corresponde a un parámetro asociado a la cantidad máxima de cortafuegos a ubicar en el bosque.

Para calcular la porción de bosque quemado en cada simulación, se utiliza la fórmula:

$$BosqueQuemado(c) = \sum_{i=1}^{N} \frac{CeldasQuemadas_i^X(c)}{CeldasTotales} \tag{3.2}$$

Donde $CeldasQuemadas_i(c)$ es el número de celdas que se incendian en el bosque en la simulación $i$, mientras $CeldasTotales$ es el total de celdas que posee el bosque y $X$ son los parámetros medioambientales del escenario $i$. $N$ corresponde al número de simulaciones que ejecuta Cell2Fire, valor ajustado por el usuario.

La mayor complejidad de este problema viene dado por los parámetros medioambientales $X$, los que tienen un comportamiento estocástico muy difícil de predecir, de hecho, si la función *BosqueQuemado*() tuviese un comportamiento determinista, el problema se convertiría en una aplicación directa del *Knapsack Problem*. Un ejemplo de un incendio forestal con características deterministas sería un incendio en el cual se conoce su punto de ignición, por lo que la solución trivial óptima es ubicar un cortafuego en dicho punto y de esta manera se salvaría todo el bosque (a excepción de la celda utilizada como cortafuego). Como en la mayoría de los incendios forestales esta situación no es así, el problema es de carácter estocástico.

Algunas aproximaciones que se pueden realizar para disminuir la incertidumbre del problema es utilizar métodos de análisis de datos para descubrir ciertas distribuciones de probabilidad de los parámetros $X$, pero determinar esto escapa del alcance de esta investigación. Por ejemplo, en [D'Este et al, 2020] se utilizan *Negative Binomial Hurdle models* para determinar ciertos lugares más probables en donde se iniciaría un incendio en la zona de Cataluña, Cerdeña y Apulia.

Para lograr resultados robustos se realizan simulaciones de Monte Carlo, método que simula un determinado número de veces distintos escenarios aleatorios para generar una aproximación más o menos estable del valor esperado de los parámetros $X$ [Metropolis y Ulam, 1949]. El tiempo que le toma a Cell2Fire simular los distintos escenarios de incendio, es directamente proporcional al número de simulaciones que se usan en Monte Carlo, por lo que un alto número de simulaciones promete resultados más robustos pero muy costosos computacionalmente, y un bajo número de simulaciones promete resultados no robustos pero rápidamente procesables. A pesar de esto, en [Barnett et al, 2016] se muestra que históricamente en Estados Unidos, la mayor cantidad de tratamientos realizados a los bosques no coinciden de buena manera con los escenarios que realmente ocurren.

De acuerdo al funcionamiento del simulador Cell2Fire, un bosque es un cuadrado formado por un conjunto de nodos simétricos en tamaño, pero variables entre ellos en su composición (tipo de combustible, pendiente, etc.). Una celda cortafuego es equivalente a un nodo del bosque incombustible, es decir, al que el fuego no puede incendiar (por lo que los estados *Non-Fuel* y *Harvested* del simulador cumplen la misma función). En consecuencia, una celda de un lago o una zona rocosa del bosque son considerados también como cortafuegos, pero en este caso son cortafuegos naturales del paisaje, por lo que no son considerados como cortafuegos en la restricción del problema y sólo son incluidos aquellos cortafuegos creados artificialmente.

Dentro de las capacidades que tiene el simulador es posible incluir estrategias de planificación temporales. Específicamente, el simulador es capaz de generar escenarios en los cuales en cierta ventana de años se pueden llevar a cabo operaciones de ubicación de cortafuegos año a año. Para simplificar esto, el problema que se resuelve en este trabajo sólo utiliza una ventana temporal de un año de planificación en donde se ubican los cortafuegos propuestos por las metaheurísticas y se simulan escenarios de incendios sujeto solamente a estos cortafuegos.

## 3.2 Estado del Arte

Las investigaciones llevadas hasta la fecha en materia de estrategia de ubicación de cortafuegos se pueden clasificar en 2 tipos para los fines de este trabajo: métodos no metaheurísticos y métodos metaheurísticos, que serán descritos a continuación.

### 3.2.1 Métodos No Metaheurísticos

La literatura existente en este ámbito es amplia comparada con los métodos metaheurísticos. Uno de los investigadores más nombrados en esta área es Mark Finney, creador del simulador *FARSITE* [Finney, 1995]. En [Finney et al, 2000] prueba que los *shaded fuelbreaks*[13] junto con un tratamiento en el combustible de la vegetación de un área puede reducir el tamaño, intensidad y efectos de incendios forestales. En [Finney, 2006] elabora un modelo computacional que, utilizando datos del suelo del bosque, logra ubicar mejores zonas en donde ubicar los cortafuegos comparado con escoger estas zonas aleatoriamente.

Otros trabajos son [Bar-Massada, 2011], en donde se utiliza un algoritmo propio en conjunto con el simulador *FARSITE* y el algoritmo *Minimal Travel Time* (propuesto en [Finney, 2002]) para ubicar cortafuegos de forma óptima para proteger estructuras en la interfaz urbano-rural[14]. En [Demange et al, 2021] plantean un enfoque teórico al modelar un bosque como un grafo en donde cada arista corresponde a la probabilidad de propagación entre distintas zonas del bosque, representadas como nodos. En este estudio prueban la dureza del problema y plantean un algoritmo que resuelve este en un tiempo polinomial para ciertas instancias. A su vez, en [Russo et al, 2016] también se modela el bosque como un grafo, pero obtienen la probabilidad de ignición de un nodo mediante cadenas de Markov y un modelo Autómata Celular para así incorporar métricas de medida de centralidad de los nodos para descubrir cuales son aquellas áreas que favorecen la propagación del incendio y así poder modificarlas.

Sumado a las investigaciones recién presentadas, en [Palma et al, 2007] es utilizado el algoritmo *Shortest Path* para identificar conexiones críticas que unen potenciales puntos de ignición de un incendio forestal con valores de riesgo asociados a cuanto tiempo se quemará un punto y el modo en que se propagará, y así, reducir tanto la inflamabilidad del bosque como las pérdidas a causa de los incendios. En [Wei et al, 2008] se plantea un modelo de programación entera mixta que incluye información espacial sobre riesgo de ignición, probabilidades condicionales de propagación del fuego entre celdas, intensidad del incendio y valores de riesgo para poder ubicar tratamientos de combustible en un bosque con el fin de disminuir la probabilidad de incendio a lo largo del bosque. Luego, en [Wei, 2012] se extiende este modelo para considerar incendios futuros. Otra investigación es la de [Minas et al, 2015] que combina tanto la planificación como el combate de los incendios forestales al crear un modelo de programación entera que incluye la ubicación de cortafuegos y la supresión del incendio para así proporcionar un enfoque de planificación integrado.

Algunos trabajos muy importantes para esta investigación son: [Finney, 2001] en donde muestra que figuras agrupadas de cortafuegos, más específicamente rectangulares, son más eficientes que figuras aleatorias. Además, en [Finney et al, 2008] se comprueba que mediante patrones regulares de cortafuegos es necesario utilizar entre un 1% al 2% de la superficie del bosque como cortafuego anualmente, mientras que si se utilizan patrones aleatorios, se necesita el doble de la superficie para lograr los mismos resultados. Otro de los trabajos importantes para esta investigación es

[13] Los *shaded fuelbreaks* son cortafuegos que son construidos con una técnica que no elimina toda la vegetación de un área sino que consiste en eliminar la vegetación más espesa y alta del lugar, para así evitar que el fuego se propague hacia árboles cercanos de gran tamaño

[14] Límite entre la ciudad y el bosque.

[Martínez, 2021], en donde se encuentran diseños de patrones de cortafuegos que permiten proteger de mejor manera al bosque frente a un incendio forestal que una disposición aleatoria de cortafuegos.

### 3.2.2 Métodos Metaheurísticos

En esta área existen 2 trabajos destacables, el primero corresponde a [Rytwinski y Crowe, 2010] en donde utilizan la metaheurística *Scatter Search* para determinar buenas ubicaciones de cortafuegos gracias a un simulador creado en esta investigación llamado *FastFire*. El desempeño de esta técnica la comparan con un algoritmo *greedy* que selecciona como celdas prioritarias aquellas que significarían una pérdida monetaria más alta en el caso de ser incendiadas. Sus resultados probaron que su mejor solución, lograda en 87 horas con 8.000 iteraciones del algoritmo, fue significativamente mejor que una solución aleatoria. Además, dicha solución fue mejor que el algoritmo *greedy* en un 5% y un 7% mejor que el mismo *Scatter Search* con 2.000 iteraciones. Estos resultados comprueban que un mayor número de iteraciones proporcionan con una significancia estadística un mejor resultado que un menor número de iteraciones.

Otro trabajo en el que se aplican estas técnicas es el de [Chung et al, 2013], en el que se utiliza *Simulated Annealing* y el simulador *Optfuels*, logran una disminución de aproximadamente un 40% de la pérdida monetaria asociada al incendio en una zona de 20.000 hectáreas cuando se aplica el algoritmo a una alta intensidad. En este caso el algoritmo ejecuta un poco más de 38.000 iteraciones en 8 días, mientras que a una baja intensidad el algoritmo es capaz de ejecutar 400 iteraciones en 2 horas. En este algoritmo construyen las soluciones de acuerdo a patrones conexos de cortafuegos y no soluciones al azar como sí lo hace la primera metaheurística expuesta.

## 3.3 Aportes del trabajo

Para identificar los aportes del presente trabajo se deben definir las diferencias más importantes entre este trabajo y los 2 presentados en la sección anterior, las cuales son:

- **Simulador**: En ambos casos utilizan simuladores distintos a Cell2Fire (a pesar de que comparten ciertas características de modelación), en el primer trabajo utilizan un simulador creado para la investigación, mientras que en el segundo trabajo la metaheurística es incluida al simulador para optimizar la ubicación de los cortafuegos. En este caso, Cell2Fire no posee ninguna extensión que sea capaz de llevar a cabo una buena estrategia de combate de incendios mediante manejo de combustible, por lo que el uso de una metaheurística se plantea como una posible mejora al simulador.

- **Función Objetivo**: Los 2 simuladores incluyen a su función objetivo los costos asociados a la pérdida de madera por el incendio. Además, el primer trabajo incluye costos operativos por construir un cortafuego en determinado lugar (aunque es normalizado a $1 para todos los cortafuegos) y un presupuesto para este propósito. En este trabajo, no se incluyen costos asociados ni a pérdidas económicas por madera, ni a pérdidas ecológicas por bosque nativo perdido o animales o plantas afectadas, ni a pérdidas humanas o sociales por el efecto del fuego y sus gases contaminantes ni costos operativos por la ubicación de cortafuegos. El objetivo de esta tesis es comparar metaheurísticas enfrentándose al problema base, y a partir de este es posible incorporar otros costos a la función objetivo.

- **Períodos de implementación**: En este trabajo no se simula la posibilidad de construir cortafuegos año a año a pesar de que Cell2Fire lo permite. Esta decisión sí es incluida en los 2 trabajos antes expuestos. El objetivo de incluir una planificación separada por períodos

es distribuir de forma más realista el trabajo operacional que conllevaría el ubicar todos los cortafuegos en un solo período anual dándole prioridad a celdas más sensibles. Lo anterior no es tomado en cuenta en este trabajo para simplificar las simulaciones.

- **Metaheurísticas a utilizar**: El objetivo de este trabajo es probar una serie de metaheurísticas y compararlas entre ellas para así obtener la que mejor se adapta al problema presentado aquí. En los otros trabajos expuestos se han aplicado directamente 2 metaheurísticas: *Scatter Search* y *Simulated Annealing*, sin haber mencionado la forma en que se comportan otras metaheurísticas frente al mismo problema. Para lograr comparar un universo más grande de metaheurísticas, es necesario incorporar más de estos algoritmos, por lo que se intentará evitar replicar el uso (al menos con su mismo diseño) de las metaheurísticas utilizadas en los 2 trabajos mencionados. Finalmente cabe destacar que la comparación de resultados con los trabajos presentados se hace difícil ya que no se cuenta con los costos (tanto económicos, ecológicos y sociales) asociados a cada celda de los bosques utilizados en este trabajo, así como tampoco con los costos operacionales que conlleva el ubicar un cortafuego, por lo que la comparativa de la función objetivo no tendría sentido debido a que los problemas no tienen el mismo foco.

Considerando lo recién expuesto, el aporte de este trabajo es encontrar metaheurísticas que logren minimizar la expansión (y daño) del incendio en un caso hipotético ideal, es decir, sólo teniendo como restricción la capacidad operativa de situar cortafuegos. Por lo tanto, al no considerar los costos reales en la función objetivo ni un horizonte de implementación, este trabajo busca entregar la cota superior del problema real de ubicación de cortafuegos. Además, el trabajo se presenta como una opción para ser implementada en un simulador de incendios forestales que actualmente no posee ninguna extensión que permita distribuir de forma óptima tratamientos de combustibles en un bosque. Finalmente, el trabajo permite explotar de forma más amplia las técnicas metaheurísticas al no incorporar ninguna de las ya estudiadas en la literatura para este problema (al menos con el mismo diseño que ya fueron implementadas).

# 4 Metodología y Desarrollo

La metodología de este trabajo se separa en una fase de elección de metaheurísticas para así encontrar las que mejor se adapten a este problema, luego se prueban distintas combinaciones de parámetros para afinar de buena manera el desempeño, posteriormente se evalúan modificaciones a las metaheurísticas seleccionadas para enfocarlas de mejor forma en el contexto del problema, luego se presenta la estructura definitiva que tienen las metaheurísticas a utilizar, y finalmente se definen y se presentan los parámetros y bosques a utilizar en el simulador Cell2Fire. En las secciones 4.1 y 4.2 de la metodología se prueba con instancias pequeñas de bosques simulados, mientras que en las etapas siguientes se utilizan instancias más grandes. En la sección 4.3 se definen las principales variaciones que serán comparadas en este trabajo, mientras que en la sección 4.4 se define la selección final de las metaheurísticas a utilizar. Finalmente, en la sección 4.5 se muestran los parámetros con los que se usa Cell2Fire y en la sección 4.6 se exponen los bosques en los que se generarán los incendios en el simulador.

## 4.1 Elección de las Metaheurísticas

La elección de las metaheurísticas se separa en 2 etapas: estudio e implementación. La fase de estudio considera la posibilidad de incluir determinada metaheurística, mientras que en la segunda la metaheurística es implementada en una pequeña instancia y se analiza su comportamiento en comparación con ubicar cortafuegos en un bosque de forma aleatoria.

### 4.1.1 Estudio de metaheurísticas

En esta fase se estudia la posibilidad de inclusión de distintas metaheurísticas. A continuación se mencionan algunas de ellas y si es que fueron consideradas para la segunda etapa o no:

- **Algoritmo Genético** (***A utilizar***): La decisión de incluir el algoritmo genético como una metaheurística a utilizar viene dada por el trabajo de [Rytwinski y Crowe, 2010], en donde se menciona que se decide utilizar *Scatter Search*, ya que es una forma más simplificada de algoritmo genético que está diseñado para reducir el número de iteraciones necesarias para encontrar soluciones de alta calidad. Dicha decisión se toma debido a que la evaluación de una determinada configuración de cortafuegos requiere un alto tiempo de ejecución computacional para ser evaluada. Como Cell2Fire permite realizar un elevado número de iteraciones en poco tiempo, se toma la decisión de evaluar el desempeño de un algoritmo genético en este problema.

- ***Simulated Annealing*** (***A utilizar***): Siguiendo el trabajo de [Chung et al, 2013] se decide estudiar la implementación de una versión alternativa a la del trabajo mencionado de *Simulated Annealing*, para así comprobar si bajo otra construcción de las soluciones es posible llegar a buenos resultados o no.

- **Búsqueda Tabú** (***A utilizar***): A pesar de que hasta ahora no existen investigaciones en las que se haya utilizado Búsqueda Tabú para ubicar cortafuegos en un bosque, sí existen bastantes publicaciones sobre el uso de esta metaheurística en otras aplicaciones forestales, por ejemplo en [Richards y Gunn, 2011] se utiliza para diseñar caminos y la cantidad de cosecha a extraer en un bosque. En [Legües et al, 2007] es aplicado para hallar buenas locaciones para ubicar maquinaria destinada al transporte de madera y generar caminos para mover la madera desde donde están ubicadas las máquinas hasta la carretera. Otros trabajos pueden ser mencionados como por ejemplo: [Laroze, 1999], [Aruga, 2005], [Brumelle et al, 1998], [Caro et al, 2003] o

[Bettinger et al, 2007]. Considerando esto, se decide estudiar la implementación de Búsqueda Tabú en este problema para así determinar si es útil en esta aplicación forestal.

- **GRASP** (***A utilizar***): Al igual que en el caso de la Búsqueda Tabú, no hay trabajos en los que GRASP haya sido utilizado para ubicar cortafuegos. Sin embargo, en el trabajo de [Díaz-Romero et al, 2013] se usa un algoritmo híbrido de GRASP junto a Búsqueda Tabú para ubicar recursos de combate de incendio en una forma adecuada para que cualquier incendio forestal en una cierta zona pueda ser combatido en un tiempo de respuesta determinado. El uso de GRASP como metaheurística para la resolución de un problema en donde se busca ubicar de buena manera recursos para combatir incendios forestales motiva a considerarla como un algoritmo que potencialmente podría traer buenos resultados.

- **Algoritmo de Colonia de Hormigas** (***No se utiliza***): El algoritmo de colonia de hormigas ha demostrado buen desempeño en problemas estocásticos [Doerr et al, 2012], además, debido a sus características permite detectar zonas en las que podría ser conveniente ubicar cortafuegos (zonas con mayor feromona de acuerdo al algoritmo). A pesar de esto, es complejo implementar el algoritmo en este problema, ya que cada solución ejecuta un movimiento según la probabilidad de moverse hacia vecinos (basado en la cantidad de feromona que tiene ese estado y la cantidad de bosque que se quema en esas soluciones). El problema en este caso es que cada solución cuenta con un elevado número de vecinos (cada nodo del bosque es eventualmente un vecino), por lo que calcular la porción de bosque quemado para cada uno de los vecinos para cada una de las hormigas y en cada iteración conlleva un costo computacional elevado. Una solución es construir una *restricted candidate list* (*RCL*), pero se determina que debido a la sensibilidad del algoritmo frente al número de candidatos, el costo de utilizarlo conlleva un alto riesgo de error. Finalmente, considerando que el algoritmo de colonia de abejas es un algoritmo similar en el que no es tan necesario evaluar a un alto número de estados vecinos, se decanta por este último sopesando todo lo mencionado.

- **Algoritmo de Colonia de Abejas** (***A utilizar***): Como se mencionó anteriormente, el algoritmo de colonia de abejas es un algoritmo similar al de colonia de hormigas, ya que ambos evalúan a ciertos nodos de acuerdo al porcentaje de bosque que se quema y en base a esto construyen métricas para generar un ranking de mejores nodos. Una importante diferencia entre ambos algoritmos y que en este caso juega a favor de este trabajo es que el algoritmo de colonia de abejas no necesita evaluar a un alto número de estados vecinos de una solución. Dicha evaluación es controlada por el usuario mediante un parámetro que señala a cuantos vecinos se les calculará su función objetivo para actualizar los mejores nodos. Considerando esto y el hecho de que este algoritmo y el algoritmo de colonia de hormigas son similares (y este último ha tenido un buen desempeño en problemas estocásticos), se decide estudiar el desempeño del algoritmo de colonia de abejas.

### 4.1.2 Implementación de metaheurísticas

- **Diseño básico de las metaheurísticas**

Antes de implementar las metaheurísticas, de acuerdo a [Talbi, 2009] es necesario definir si estas serán *from scratch*, con *code reuse* o *code reuse and design*. La primera opción es crear el código computacional del algoritmo completamente desde cero, sin ayuda de ningún software y sólo con el pseudocódigo del algoritmo. El segundo enfoque es utilizar un algoritmo de libre acceso (o no) de la web, implementado por algún desarrollador. Generalmente estos códigos son parte de una librería de códigos más amplia que contiene varios métodos de resolución de problemas de optimización. La tercera opción es una mezcla entre los 2 enfoques mencionados

y consiste en modificar un algoritmo creado por algún desarrollador para ser aplicado en un problema de forma personalizada y así ser adaptado de mejor forma al mismo.

En este trabajo, todos los algoritmos son construidos con el enfoque *from scratch*. La razón de esto es que diseñar un algoritmo con la técnica señalada permite ser más flexible en modificaciones a la forma en que se comportan los algoritmos. Por ejemplo, si se quisieran realizar mutaciones en el algoritmo genético con un cierto sesgo hacia determinadas zonas del bosque, sería mucho más complejo realizarlo en un algoritmo ya creado porque está diseñado para trabajar con cierto tipo de mutaciones. Lo mismo ocurriría si se desease modificar la probabilidad de aceptación de un *Simulated Annealing*. Es difícil encontrar algún algoritmo incluido en algún software que permita utilizar probabilidades de aceptación personalizadas más allá de las que usualmente se utilizan en dicha metaheurística.

Tampoco se considera utilizar un criterio de *code reuse and design* ya que se prefiere armar el algoritmo de forma totalmente personalizada para que así de ser necesario realizar modificaciones, se tenga total conocimiento del algoritmo construido y sus partes. Para esto todos los algoritmos fueron construidos a partir de los pseudocódigos de la sección 1.3.1 de este trabajo.

La obvia desventaja de este enfoque es no ser capaz de construir un algoritmo que converja a una buena solución. Pero para esto se realizan pruebas que permiten determinar si un algoritmo es bueno al compararlo con soluciones al azar y otras configuraciones de cortafuegos que han demostrado buenos resultados.

Todas las metaheurísticas inicialmente se construyen a partir de los pseudocódigos expuestos en la sección 1.3.1 de este trabajo y con soluciones iniciales (si es que el algoritmo las utiliza) aleatorias. Algunos detalles de cada metaheurística son:

- **Algoritmo genético**: Para este algoritmo, es necesario tomar algunas decisiones previas para definir la forma en que funcionará, estas son:
  - Representación: Se representa cada solución como un vector binario de un largo equivalente al tamaño del bosque. En dicho vector, cada valor corresponde a la decisión de ubicar un cortafuego en la celda respectiva a la posición en el vector.
  - Selección: Se decide escoger el método clásico de selección al elegir a los mejores $N/2$ individuos de la generación como padres, con $N$ el tamaño de la generación. La generación siguiente se compone de una mitad de padres, y de la otra mitad de hijos (resultantes del cruce) mutantes.
  - Cruce: Para el cruce se consideran 2 opciones: en primer lugar se utiliza *crossover* en un punto y posteriormente se modifica a *uniform crossover*. La primera opción se descarta debido a que esta técnica produce muchas soluciones infactibles, lo que complica la convergencia de la solución. Esto se produce en casos en los que en un padre existe un grupo de cortafuegos que se concentran en el inicio del bosque, y en otro padre estos cortafuegos se concentran al final del bosque. Por ende, al mezclarse ambas soluciones, la solución resultante tendrá un número elevado de cortafuegos que sobrepasará la capacidad máxima de cortafuegos a utilizar.
    Es por esto que se decide utilizar *uniform crossover*, teniendo cada nodo de cada padre la misma probabilidad de ser seleccionado para construir a un hijo.
  - Mutación: Se decide mutar inicialmente modificando simplemente uno o dos nodos de la solución, esta modificación puede ser removiendo el cortafuego ubicado en

ese lugar, añadiendo un cortafuego en esa posición, o removiendo el cortafuego y ubicándolo en otro lugar.

- ***Simulated Annealing***: En el caso de *Simulated Annealing*, se decide utilizar la misma estructura básica, es decir, con la probabilidad de aceptación en base a la distribución *Boltzmann* y con un enfriamiento geométrico. Para el movimiento a un estado vecino, se representan las soluciones como un vector conteniendo los nodos que poseen cortafuegos, y se define el movimiento como extraer un cortafuego, ubicar un cortafuego (si la capacidad máxima no ha sido superada) o extraer y ubicar un cortafuego.
- **Búsqueda Tabú**: En la Búsqueda Tabú, la gran decisión corresponde a utilizar o no la memoria a largo plazo. Por temas de capacidad de memoria (especialmente en bosques más grandes), se decide no utilizarla. Además, la lista tabú inicialmente se construye de tamaño dinámico y no estático. En el caso de los movimientos a estados vecinos, se utiliza el mismo criterio que en *Simulated Annealing*. Además, se incluye un criterio de diversificación que modifica la solución cada $n$ iteraciones sin mejorar la solución global, lo que permite diversificar la solución para no estancarse en un óptimo local. Así, cada $n$ iteraciones en las que la solución global no ha mejorado, se modifica la solución construyendo una solución híbrida entre una configuración aleatoria y una configuración construida a partir de información de iteraciones anteriores (proceso detallado en la sección 4.4.2).
- **GRASP**: Para la implementación de GRASP, se decide utilizar un criterio *greedy* de acuerdo a la zona en la que el incendio afecta con mayor impacto. Así, se simula el comportamiento frente a un bosque sin cortafuegos y se ubican cortafuegos en alguna de las zonas que más se queman (las que componen la *RCL*) en una serie de simulaciones. Así, sucesivamente se van ubicando cortafuegos y simulando para ir atacando el incendio a partir de la zona que se ve más afectada.
  Para la fase de búsqueda local, se ejecutan visitas a vecinos de la misma forma que lo hace *Simulated Annealing* y Búsqueda Tabú.
- **Algoritmo de Colonia de Abejas**: Los algoritmos de enjambre fueron creados principalmente en ambientes de funciones reales, es por esto que es necesario adaptarlos cuando se trabaja con problemas binarios. En este caso, el algoritmo de colonia de abejas aplicado en este trabajo se construye a partir de la investigación de [Zhang, 2019], en donde implementan una metodología para aplicar el algoritmo en ambientes binarios, específicamente en el *Knapsack Problem*.
  En el caso de este trabajo, se representa de la misma forma que en el algoritmo genético, es decir, como un vector de tamaño igual al tamaño del bosque, el cual contiene $1's$ y $0's$ correspondiente a la decisión de poner un cortafuego o no respectivamente en la posición señalada por el vector.
  La gran modificación de esta metodología comparada con el algoritmo tradicional, es que se reemplaza la función de la actualización de la solución $x$ por:

$$\begin{aligned} u_{i,j} &= x_{best,j} + \phi_{i,j} * (x_{best,j} - x_{k,j}) \\ u_{i,j} &= \lceil u_{i,j} \rceil , \end{aligned}$$

  con $x_{best,j}$ la coordenada j de la mejor solución $x$ encontrada hasta ahora, $x_{k,j}$ la coordenada j de una solución distinta a la mejor solución encontrada hasta ahora, $\phi_{i,j}$ un número aleatorio entre -1 y 1, $u_{i,j}$ la coordenada $j$ de la solución $i$, y con $\lceil \cdot \rceil$ la función

cajón superior. Además, se considera:

$$u_{i,j} = \begin{cases} 1, & u_{i,j} > 1 \\ 0, & u_{i,j} < 0. \end{cases}$$

Esta formulación, permite que el algoritmo genere exclusivamente soluciones binarias.

- **Comparación de las metaheurísticas en ambientes pequeños**
  En primera instancia, las metaheurísticas son probadas en bosques simples y con condiciones climáticas con baja variabilidad para comprobar su convergencia a soluciones triviales. Con este fin, se utilizan bosques de 10 celdas de ancho y 10 celdas de largo, representando 100 hectáreas. El combustible utilizado es heterogéneo (distinto tipo de vegetación a lo largo del bosque) y vientos con poca variabilidad en su ángulo de dirección. Además, se utiliza un único punto posible de ignición. Estas variables son definidas así para reducir la aleatoriedad del problema y así facilitar la convergencia. Los resultados de estos experimentos no son presentados ya que no generan relevancia significativa para la investigación del uso de metaheurísticas en incendios forestales y sólo sirven como referencia para definir si las metaheurísticas tienen facilidad para trabajar en formas simplificadas de este problema, los resultados significativos sí son presentados en la siguiente sección de este documento.

  En estos experimentos previos, la mayoría de las metaheurísticas logran resultados satisfactorios ante distintos valores de sus parámetros claves. La única metaheurística que no logra tener buenos resultados es Algoritmo de Colonia de Abejas, la que no es capaz de generar de buena forma las soluciones construidas a partir de la información de las otras soluciones. Esto se debe principalmente a la alta sensibilidad de construcción de soluciones infactibles, ya que ubicando más de un determinado porcentaje de cortafuegos, se considera una solución infactible con un valor objetivo nulo. En este trabajo, dicho porcentaje suele ser bajo (aproximadamente se permite un máximo de un 5% de la superficie total del bosque a utilizar como cortafuego), por lo que la posibilidad de construir soluciones infactibles es alta y ante esto, el algoritmo antes mencionado no es capaz de funcionar correctamente incluso variando sus distintos parámetros (número total de abejas, número de abejas exploradoras, número de generaciones, etc.).

  Lo anterior ocurre debido a la forma en que se construyen las soluciones del Algoritmo de Colonia de Abejas, en donde una nueva solución es una combinación convexa de 2 soluciones (la mejor histórica y una al azar) a partir de un parámetro $\phi$, a diferencia de las otras metaheurísticas, en donde su construcción es más simple y permite manejar de mejor manera las soluciones infactibles. Es así, que para este trabajo se descarta el uso de dicho algoritmo y sólo se trabajará en adelante con Búsqueda Tabú, GRASP, *Simulated Annealing* y Algoritmo Genético.

## 4.2 Configuración de Parámetros

Existen parámetros globales que utilizan cada uno de los algoritmos. Estos parámetros son el tiempo/número de iteraciones (o generaciones en el caso del Algoritmo Genético) y las simulaciones a considerar en Cell2Fire.

- Tiempo máximo de iteración/ Número de iteraciones: El número de iteraciones (o generaciones para el Algoritmo Genético) viene dado por el máximo tiempo que se desea dejar operar al algoritmo. Dicho tiempo se deja variable por prueba, pero se define entre 2 y 6 horas, lo que

le permite al algoritmo lograr cierta convergencia sin permitir un amplio lapso de tiempo de espera. Este tiempo resulta en un número variable de iteraciones en función del tamaño del bosque: bosques más pequeños tienen un mayor número de iteraciones en el mismo tiempo que bosques más grandes ya que los primeros requieren menor tiempo de simulación en Cell2Fire, por lo que el número de iteraciones varía en función del bosque con el que se experimenta. Del mismo modo, de acuerdo al funcionamiento del algoritmo, no todos realizan las mismas iteraciones en el mismo tiempo, por lo que se decide usar como criterio de parada un tiempo máximo y no el número de iteraciones a realizar. Cabe señalar que una iteración se define como la evaluación de una generación en el caso del Algoritmo Genético, mientras que para Búsqueda Tabú una iteración corresponde a la visita de los estados vecinos definidos y la posterior comparación con la solución global, y en el caso de GRASP una iteración corresponde a una fase de construcción completa y la visita de los estados vecinos previamente determinados.

- Simulaciones en Cell2Fire: El número de simulaciones a usar en Cell2Fire es un elemento crucial en este trabajo. Este valor hace referencia a cuantos escenarios distintos de incendios será sometida una solución. La importancia de este número se debe a la naturaleza aleatoria del problema. Como se señaló en la sección 3, para asegurar robustez en las soluciones son sometidas a simulaciones de Monte Carlo y por ende, mientras más simulaciones se realicen, más realista será el escenario. El problema de esto es que el realizar un número elevado de simulaciones conlleva un alto tiempo de ejecución. Así, es crucial encontrar un buen número de simulaciones a realizar en Cell2Fire para asegurar resultados realistas en poco tiempo.

  Para determinar este número, cada algoritmo es sometido a distintas pruebas, logrando determinar que el Algoritmo Genético necesita un número de simulaciones más elevado que los otros algoritmos ya que su funcionamiento se basa en la confianza otorgada por las mejores configuraciones, mientras que los otros algoritmos de búsqueda local son menos dependientes de las iteraciones pasadas (este extremo se evidencia en GRASP, el cual sólo utiliza las soluciones de iteraciones anteriores para actualizar el máximo global). Por ende, en esta tesis, el algoritmo genético realizará un número mayor de simulaciones que Búsqueda Tabú, *Simulated Annealing* y GRASP.

  Búsqueda Tabú y *Simulated Annealing* realizarán más o menos el mismo número de visitas a estados vecinos en cada iteración, por lo que el número de simulaciones en Cell2Fire se fijará similar entre ambas metaheurísticas.

  Finalmente, GRASP debe considerar aparte de la búsqueda local (que realizan de forma similar Búsqueda Tabú y *Simulated Annealing*) una fase de construcción, que se considera aún más crucial que la fase de búsqueda local. Por ende, se realizará un mayor número de simulaciones en la fase de construcción de GRASP y un menor número en la fase de búsqueda local, siempre buscando que el total de simulaciones realizadas en cada iteración del algoritmo sea más o menos similar al número de iteraciones realizadas en Búsqueda Tabú y *Simulated Annealing* ya que como los 3 algoritmos corresponden a metaheurísticas de búsqueda local, se busca fijar tiempos de iteración similar entre ellas.

Además, cada metaheurística posee parámetros propios que deben ser definidos de acuerdo al problema que se intenta resolver.

- **Algoritmo Genético**: En el caso del algoritmo genético, existen 3 parámetros que deben ser definidos: Tasa de mutación, Tasa de cruce y Tamaño de la población.

1. Tasa de mutación: La tasa de mutación se varía entre los valores 80%, 60% y 40%. Finalmente se determina que 60% es una buena tasa para este problema ya que permite una alta diversificación entre los hijos generados. Un valor más alto genera que se pierda la información de los 2 padres seleccionados, y un valor más bajo tiende a no generar suficiente variabilidad de las soluciones y sólo genera hijos como combinación de padres.
2. Tasa de cruce: En este caso se utiliza una tasa de cruce de 100%, es decir, todos los hijos de la generación son generados por una combinación entre los padres y ninguno corresponde al mismo individuo de la generación anterior. Este método se utiliza debido a que ya se guarda la información de individuos sin alterar de la generación anterior gracias al guardado de los mejores individuos (padres en la generación futura), por lo que conservar nuevamente individuos sin alterar de la generación sería o redundar en individuos ya considerados, o guardar individuos no tan buenos.
3. Tamaño de la población: Para definir el tamaño de la población se varía entre 40, 30, 20, 10 y 5 individuos. Se logra determinar que 10 individuos por generación permite mantener variabilidad al mismo tiempo que se realizan iteraciones rápidas del algoritmo.

- ***Simulated Annealing***: En *Simulated Annealing* es necesario determinar el número de vecinos a visitar en cada iteración, la temperatura inicial y la constante de enfriamiento. En el caso de los estados vecinos, se evalúan 5, 10 y 20 vecinos, logrando determinar que un buen número de estados vecinos a visitar es 10, logrando iteraciones rápidas visitando un número aceptable de posibilidades. Para la temperatura se determina que $100^{o}$ y una tasa de enfriamiento de 0.9 otorga buenos resultados.

- **Búsqueda Tabú**: Para Búsqueda Tabú se debe determinar el número de vecinos a visitar en cada iteración, el número de iteraciones de espera para diversificar la solución y el número de iteraciones que estará cada movimiento en la lista tabú.

  1. Vecinos a visitar: En el caso de los vecinos a visitar, se utiliza el mismo criterio de *Simulated Annealing* ya que ambos algoritmos visitan el mismo número de estados en cada iteración, es decir, se visitan 10 vecinos en cada iteración.
  2. Número de iteraciones de espera para diversificación de solución: En este caso se probó entre 5, 10 y 15 iteraciones de espera para diversificar las soluciones. Dicho número depende del tiempo de simulación en Cell2Fire, ya que en bosques más grandes las simulaciones requieren más tiempo, provocando que tome más tiempo iterar, por lo que se debe diversificar más rápidamente y por ende, el número de iteraciones de espera debe ser menor. Como en este trabajo se experimenta con bosques más grandes, se decide utilizar 5 iteraciones de espera para la diversificación, lo que permite explorar un número suficiente de estados posibles de cada solución como para tener una intensificación suficiente y así justificar un cambio para encontrar una mejora en la solución global.
  3. Número de iteraciones de estadía de cada movimiento en la lista tabú: Como se utiliza un tamaño dinámico de la lista tabú, cada movimiento estará por un determinado número de iteraciones dentro de la lista. Este número fue variado entre 10, 15 y 20. Al igual que el número de iteraciones de espera para la diversificación, este valor depende del tamaño del bosque. Considerando el problema en cuestión, se utilizan 15 iteraciones de estadía para cada movimiento, lo que permite explorar distintas soluciones para alcanzar una diversificación suficiente como para converger.

- **GRASP**: Finalmente, en GRASP es necesario definir 3 parámetros: número de iteraciones en la fase de construcción, número de vecinos a visitar en la fase de búsqueda local y tamaño de la *restricted candidate list* (*RCL*).

1. Número de iteraciones en la fase de construcción: Como cada configuración de cortafuegos se compone de $c$ celdas cortafuegos correspondientes al 5% del tamaño del bosque, el mejor caso sería realizar $c$ iteraciones en la fase de construcción, y así ubicar en cada iteración un cortafuego. Debido a la complejidad temporal, este enfoque es muy difícil de realizar, por lo que se decide ubicar en cada iteración $\lfloor \frac{c}{4} \rfloor$ cortafuegos, es decir, el 25% de los cortafuegos a ubicar. Dicho porcentaje permite realizar iteraciones (considerando también la fase de búsqueda local) en tiempos similares a Búsqueda Tabú y a *Simulated Annealing*, algoritmos que funcionan también en base a búsqueda local.

   Además, en cierto porcentaje de las ocasiones, se inicializa el algoritmo con el 25% de cortafuegos de una solución aleatoria. Este porcentaje corresponde al 40% de las ocasiones, lo que permite cierta variabilidad al punto de inicio del algoritmo, ya que si dicho porcentaje fuera del 0%, es decir, siempre se inicia con una solución vacía, el comportamiento de las soluciones tendería a ser similar entre algunas de ellas, ya que todas parten del mismo escenario y bastaría que ciertas condiciones climáticas coincidan para que converjan a la misma solución quitando variabilidad a las soluciones.

   De esta forma, en la fase de construcción se realizan 4 o 5 iteraciones dependiendo de si se inicia parcialmente con el 25% del bosque cubierto, o si se inicia con un bosque sin cortafuegos, respectivamente.
2. Número de vecinos visitados en fase de búsqueda local: En este caso, se visitan 5 estados vecinos en cada iteración, dicho número se determina en conjunto con el porcentaje de cortafuegos ubicados en cada iteración de la fase de construcción para así lograr iteraciones en un tiempo aceptable y similar al de Búsqueda Tabú y *Simulated Annealing*.
3. La *RCL* permite obtener un conjunto de opciones a considerar para realizar movimientos en GRASP. En este caso, se decide que un tamaño de entre 10 a 20 candidatos es un número suficiente de opciones, considerando que en cada iteración de la fase de construcción se seleccionará sólo un candidato, por lo que el tener entre 5% y 10% de probabilidad de seleccionar cada nodo parece ser un valor suficiente.

## 4.3 Modificación de Metaheurísticas para Aplicación al Problema

Para lograr personalizar de mejor forma a las metaheurísticas según el problema a resolver, estos algoritmos son modificados. En la sección Resultados de este trabajo, son comparados los resultados de estas metaheurísticas con y sin dichas modificaciones. Las 3 modificaciones son:

- **Tipo de la solución**: En un inicio, se construyen soluciones en base a nodos independientes ubicados en un bosque, tal como se muestra en la figura 4.3:

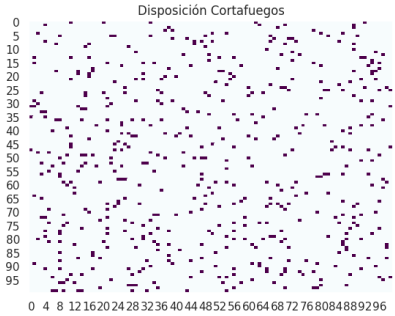



Figura 4.1: Disposición de cortafuegos individuales

De acuerdo a [Finney et al, 2008], las figuras agrupadas de cortafuegos son más eficientes al momento de proteger contra un incendio forestal que figuras independientes. En los primeros experimentos se espera que dichos resultados tiendan a converger a soluciones agrupadas en el bosque, situación que no ocurre consistentemente. Es por esto, que se utiliza el mismo método que [Chung et al, 2013], en donde cada cortafuego es construido como un conjunto de cortafuegos agrupados, y no se consideran cortafuegos individuales como una posibilidad para el algoritmo. Una consecuencia importante de esta modificación es el descarte de la metaheurística *Simulated Annealing*, ya que como se señaló en la sección 3.3, la justificación de la inclusión de este algoritmo se basa en la diferencia en la implementación con respecto al trabajo en donde ya fue aplicada. Pero como se utilizará un enfoque similar, su uso ya no es considerado.

Para la construcción de estos *clusters* de cortafuegos, se utiliza la investigación realizada por [Martínez, 2021], en donde se encuentra que una determinada geometría de cortafuegos protege mejor al bosque que otras figuras bajo la misma disposición espacial en el bosque, dicha figura y disposición espacial pueden observarse en la figura 5.2.

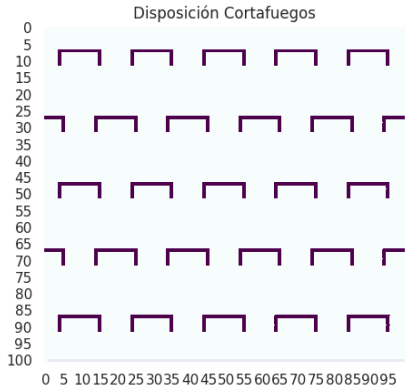



Figura 4.2: Disposición de cortafuegos según [Martínez, 2021]

Tomando en cuenta este trabajo, en la presente tesis se utilizan cortafuegos simétricos a los expuestos. Estas estructuras tienen forma de arco, poseen una dimensión de 12 nodos en sentido horizontal, 5 nodos en dirección vertical al lado derecho, y 5 nodos en dirección vertical al lado izquierdo, por lo que su dimensión total es de 20 nodos de cortafuegos (ya que los 2 nodos horizontales de los extremos pertenecen también a los de las figuras verticales). Así, las soluciones de los algoritmos aquí expuestos contarán con $c$ *clusters*, donde:

$$c = \lfloor \frac{\text{Capacidad máxima de cortafuegos a ubicar}}{20} \rfloor$$

Además, en este trabajo estos cortafuegos son implementados de tal forma que se pueden ubicar en 4 orientaciones distintas: 0°, 90°, 180° y 270°, estas 4 disposiciones son presentadas en la figura 4.3.

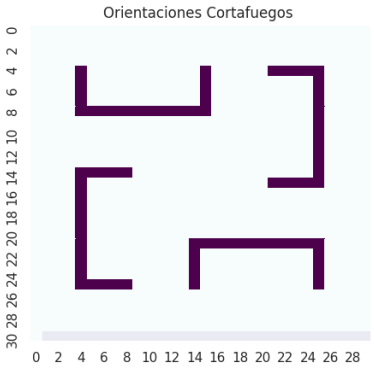



Figura 4.3: Posibles orientaciones de los cortafuegos a utilizar

- **Forma de la solución inicial**: La solución inicial se considera de 2 tipos posibles: de acuerdo a una solución totalmente aleatoria, o de acuerdo a la configuración de la figura 5.2. Como se sabe que esta última solución tiene un desempeño bastante mejor que una solución aleatoria, se propone utilizarla como punto de partida para los algoritmos. Una posible desventaja de esto es que no permita mucha mejora a la metaheurística, lo que genere un estancamiento de la misma, por lo que se deja la posibilidad también de una solución inicial aleatoria. Esto último nace de acuerdo a [Talbi, 2009], en donde se señala que en algunos problemas el otorgar una solución inicial generada a partir de un enfoque *greedy* tiene peores resultados que al utilizar una solución inicial aleatoria.

- **Sesgo en los movimientos**: Una hipótesis de este problema es que ubicar un cortafuego en la zona en donde más ocurren incendios es más efectivo que ubicarlo en otra zona. En sentido esta idea, se propone que los movimientos de los algoritmos a estados vecinos sean considerando dicho efecto. Así, en los movimientos de las configuraciones en cada algoritmo se utiliza un parámetro $\phi$, que puede tomar valores entre 0 y 1. Dicho parámetro representa el sesgo en el movimiento a dar, cuando $\phi$ vale 0, el movimiento a dar corresponde a ubicar un cortafuego en una zona aleatoria del bosque, mientras que cuando $\phi$ vale 1, el movimiento a dar corresponde a ubicar un cortafuego en alguna de las zonas más quemadas a lo largo de todas las simulaciones que ha realizado el algoritmo.

  Para otorgar cierta variabilidad entre qué nodos se escogerán en cada movimiento, se construye una distribución de probabilidad entre los nodos según su frecuencia de incendio. En este caso, se decide optar por la probabilidad geométrica con parámetro $p = \frac{1}{2}$. Para realizar un movimiento se realiza un ranking que contiene al 10% de los nodos que más se han quemado. A partir de esta lista, se modela la probabilidad de moverse de acuerdo al nodo que está en la posición $x$ de dicho ranking de la forma:

$$P[X = x] = p(1-p)^{x-1}, \text{ con x } \in [1, 2, 3, ...]$$

  Así, el nodo que más se ha quemado tiene una probabilidad de ser escogido de $\frac{1}{2}$, mientras que el nodo que más se ha quemado en segundo lugar tiene una probabilidad de ser escogido de $\frac{1}{4}$, etc. Con esta forma de modelar la chance de escoger los movimientos realizados, se logra incluir una pequeña aleatoriedad que permite no siempre moverse hacia el nodo que más se quema y de esta forma no se sesga de forma extrema los movimientos vecinos. Además, se experimentará el resultado de tener un sesgo de movimiento del 60% y no tenerlo. No se prueba el escenario con $\phi = 1$ ya que genera soluciones muy poco diversificadas.

## 4.4 Implementación Final de las Metaheurísticas

Considerando lo anterior y en base a las modificaciones realizadas además del descarte de *Simulated Annealing* justificado en la sección anterior, a continuación se describe el procedimiento de ejecución de cada metaheurística.

### 4.4.1 Algoritmo Genético

En la figura 4.4, se presenta la estructura base del algoritmo genético. Cuando se desea analizar el impacto de una solución inicial distinta, se modifica uno de los individuos de la generación inicial y se reemplaza por la solución inicial de acuerdo a [Martínez, 2021].

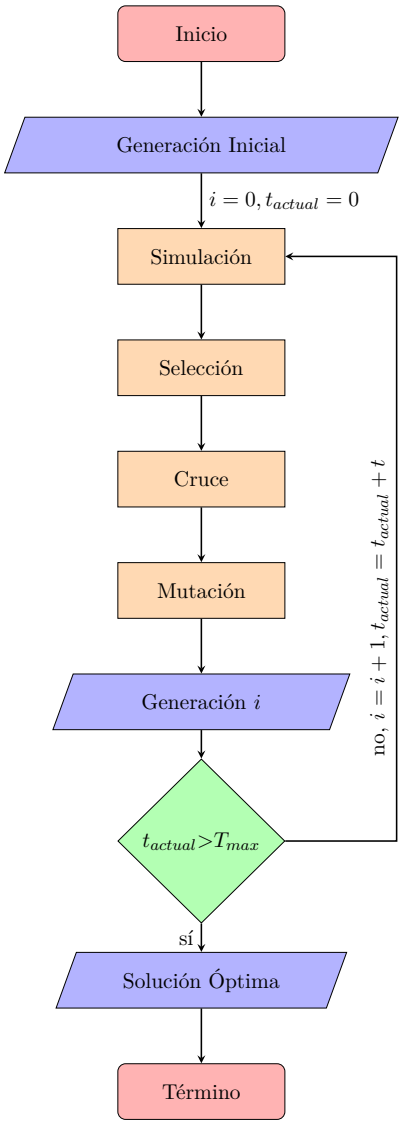



Figura 4.4: Algoritmo Genético

La primera etapa de una iteración de este algoritmo corresponde a la simulación, en donde la población es evaluada de acuerdo a su desempeño en Cell2Fire, posteriormente, las soluciones son seleccionadas de acuerdo a dicho desempeño. Luego, las configuraciones escogidas son combinadas entre ellas para dar lugar a los hijos, los cuales serán mutados en la etapa siguiente. Posteriormente, se construye la generación siguiente con una mitad correspondiente a los padres, obtenidos

en la etapa de selección, y la otra mitad correspondiente a los mutantes, obtenidos en la etapa de mutación. Finalmente, al haber superado el tiempo de ejecución, se entrega al mejor individuo de la última generación, correspondiente a la solución óptima.

A continuación, en la figura 4.5, se muestra el procedimiento de la etapa de simulación.

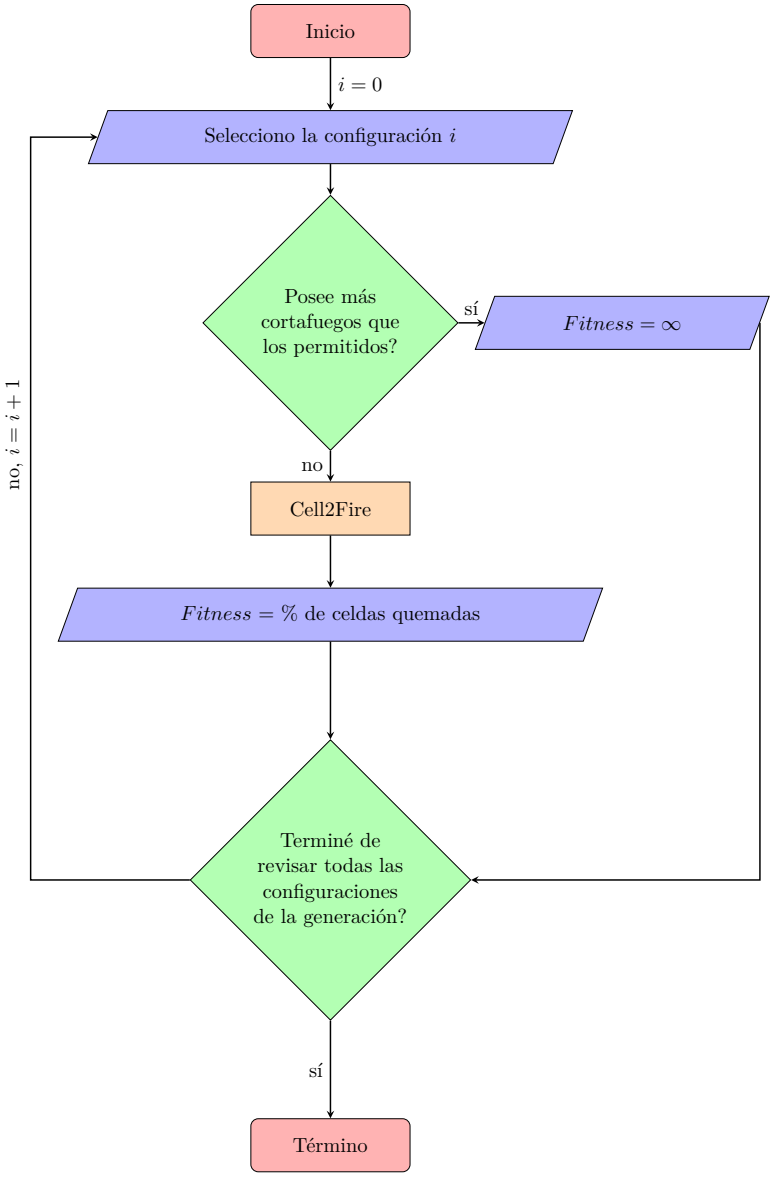



Figura 4.5: Simulación en Algoritmo Genético

En esta etapa, cada individuo es evaluado de acuerdo a su configuración de cortafuegos. El mecanismo de evaluación de un individuo es de acuerdo al porcentaje de celdas que se queman en una serie de incendios a los que se ve enfrentado. En el caso que un individuo posea más cortafuegos que lo máximo a ubicar, se fija su desempeño como nulo y no es evaluado por el simulador.

En la figura 4.6, se describe el proceso de selección de los individuos.

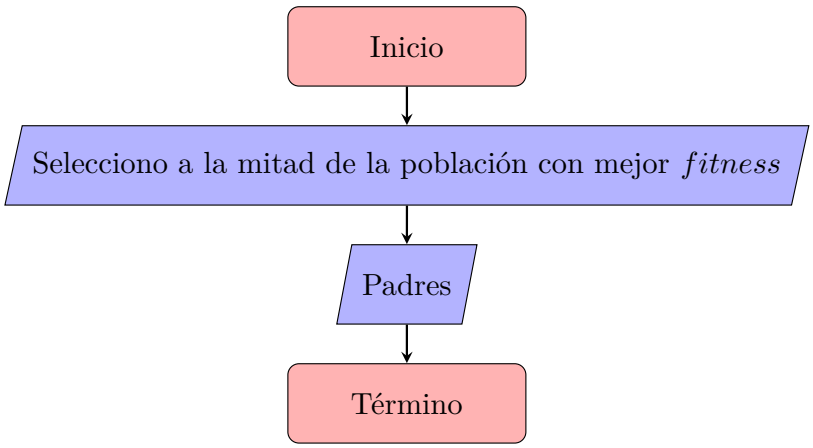



Figura 4.6: Selección en Algoritmo Genético

En la selección, cada individuo es elegido de acuerdo al *fitness* obtenido en la fase de simulación. De acuerdo a lo señalado en la sección 4.1.2, se ordenan los elementos según su desempeño (de menor a mayor celdas quemadas) y se selecciona a la mitad de la generación como padres de la siguiente generación, los cuales serán utilizados tanto en la fase de cruce como para construir la mitad de la generación futura.

En la figura 4.7, se expone la etapa de cruce de los padres, la cual da a lugar a los hijos que serán mutados para la siguiente generación.

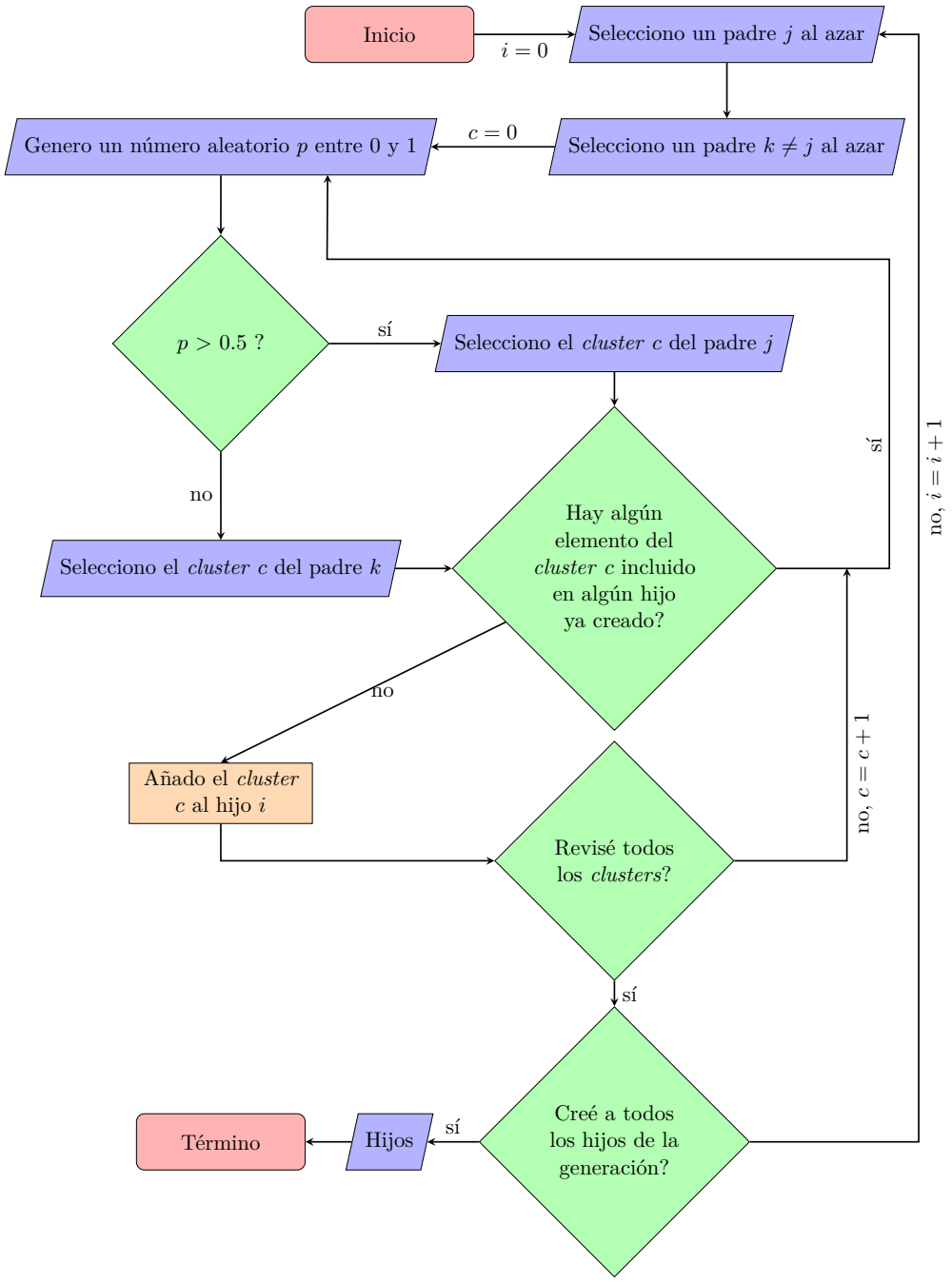



Figura 4.7: Cruce en Algoritmo Genético

En esta etapa, 2 padres distintos son elegidos al azar, posteriormente, se construye cada *cluster* de cada hijo seleccionando dicho *cluster* de alguno de los 2 padres de acuerdo a una probabilidad equivalente. Para permitir variabilidad en la generación de los hijos, no se construyen *clusters* que posean cortafuegos en común con algún hijo ya creado en la generación actual.

En la figura 4.8, se muestra el funcionamiento del proceso de mutación de los hijos obtenidos en la fase de cruce.

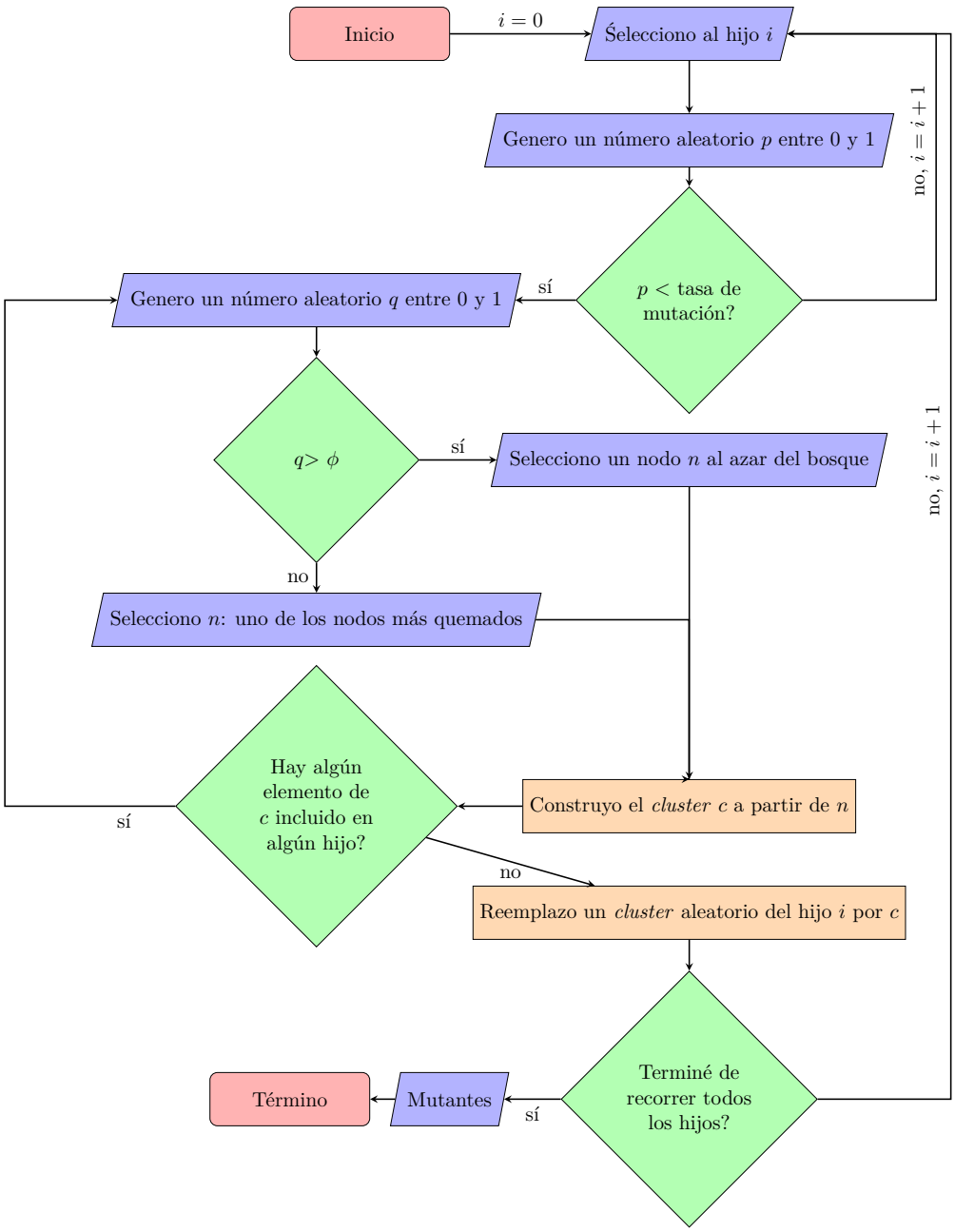



Figura 4.8: Mutación en Algoritmo Genético

Finalmente, en la mutación se modifican los hijos. En primer lugar, se evalúa si se cambiará la configuración del hijo o no de acuerdo a la tasa de mutación. Posteriormente, se reemplaza uno de los *clusters* de los hijos por un *cluster* que se construye a partir de un nodo aleatorio, o uno de los nodos más quemados en la historia del algoritmo, esto de acuerdo al sesgo de selección según la tasa $\phi$.

Es importante destacar que en el caso que el método de construcción de las soluciones sea en base a nodos independientes y no a *clusters*, el procedimiento en general del algoritmo es el mismo, con la diferencia de que en la etapa de mutación y cruce, se seleccionan una cantidad de nodos equivalente al tamaño de un *cluster* con el fin de mantener el impacto de los cambios realizados en cada versión de la metaheurística. Así, en la etapa de cruce, en vez de seleccionar un *cluster* de cada padre, se seleccionan $k$ nodos, con $k$ el número de nodos que posee un *clusters*. De la misma

forma, en la mutación, se modifican $k$ nodos de cada hijo.

### 4.4.2 Búsqueda Tabú

En el caso de la Búsqueda Tabú, la figura 4.9 muestra su estructura base.

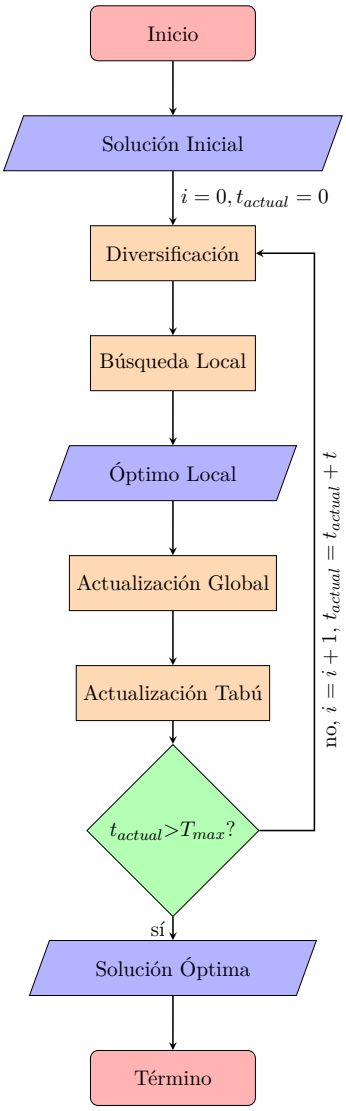



Figura 4.9: Búsqueda Tabú

La primera etapa de esta metaheurística se compone de una diversificación, la cual ocurre cuando ha pasado un determinado número de iteraciones en las que no ha mejorado la solución global del problema. Luego de esta, la solución es evaluada de acuerdo a la visita de un cierto número de estados vecinos, proceso del cual se obtiene un óptimo local, el cual permite actualizar la solución global y la lista tabú. Al superar el tiempo máximo de simulación, el algoritmo entrega la mejor solución que se haya encontrado hasta el momento.

A continuación, en la figura 4.10, se expone la primera etapa de este algoritmo: Diversificación.

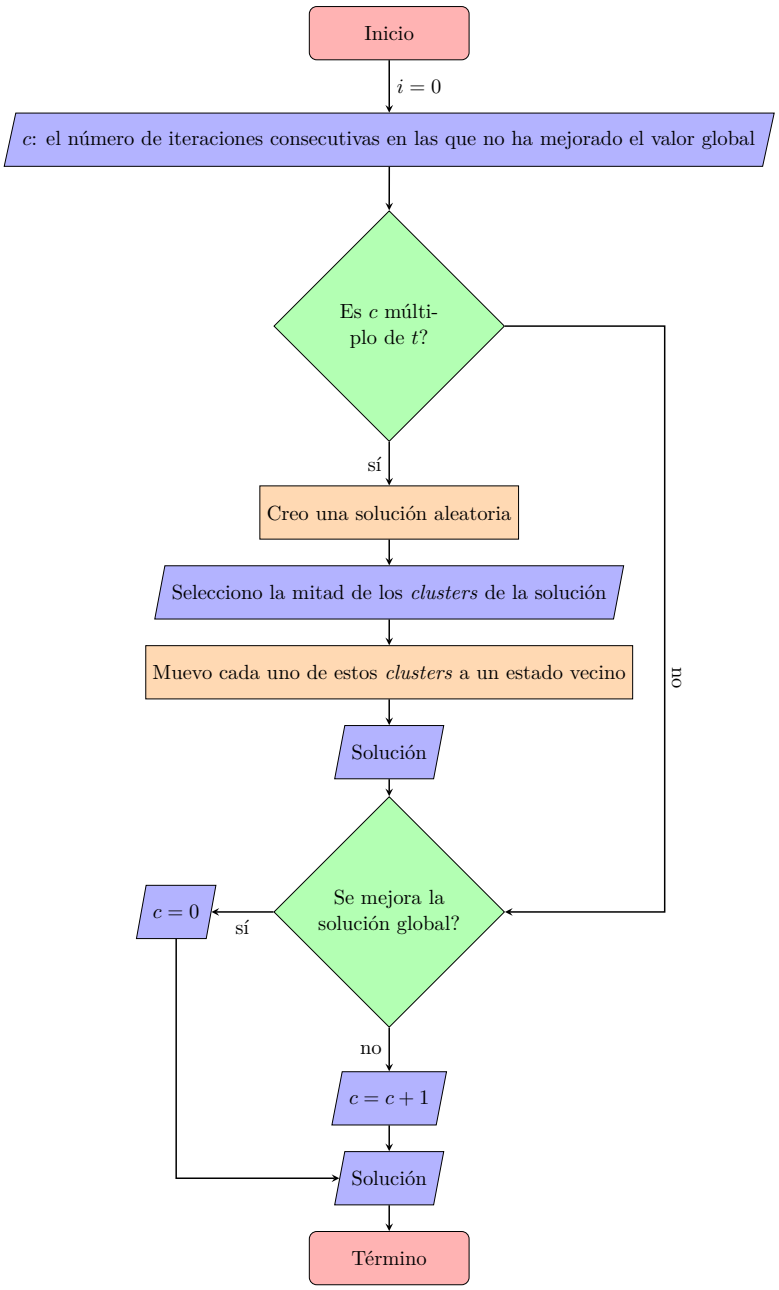



Figura 4.10: Diversificación en Búsqueda Tabú

En esta etapa, se evalúa si han pasado $t$ iteraciones en las que la función objetivo de la solución global no ha mejorado. Si es así, se genera una solución aleatoria. Luego se escoge aleatoriamente a la mitad de esta solución y se modifican estos *clusters* a estados vecinos, con lo que se tendrá una solución híbrida con una mitad aleatoria, y la otra generada de acuerdo a movimientos realizados a partir de la otra mitad de esta solución según el sesgo de movimiento $\phi$. Si posteriormente, en la etapa de actualización global, se mejora la solución global, entonces se reinicia el contador del proceso de diversificación.

En la figura 4.11, se muestra el proceso de búsqueda local de este algoritmo.

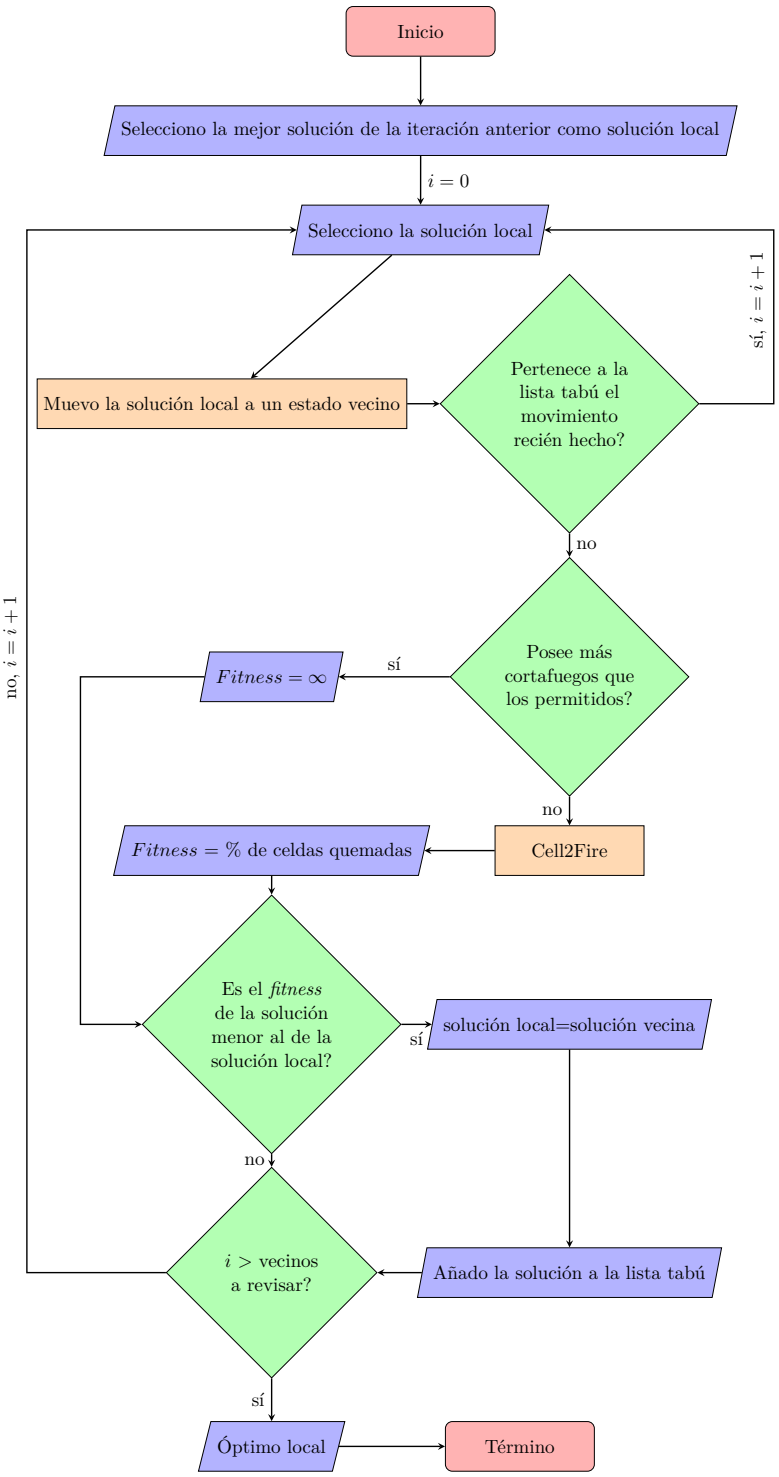



Figura 4.11: Búsqueda Local en Búsqueda Tabú

En esta fase, la mejor solución de la iteración anterior corresponde a la nueva solución de la iteración actual. A partir de esta solución local, se construye un determinado número de vecinos, los cuales son evaluados solamente si el movimiento realizado para su construcción no pertenece a la lista tabú, ni si el número de cortafuegos que posee supera al máximo posible a ubicar. En el último caso su porcentaje de celdas quemadas es nulo. Si la solución es viable, se simula su desempeño de acuerdo al porcentaje promedio de celdas quemadas en una serie de simulaciones en Cell2Fire. Luego, si dicha solución mejora a la solución local, entonces se guarda como la nueva solución local

de la iteración. Finalmente, el movimiento realizado para obtener el nuevo óptimo local es agregado a la lista tabú y se entrega dicho óptimo, es decir, el óptimo local de la iteración.

A continuación, en la figura 4.12, es posible visualizar el procedimiento de actualización global.

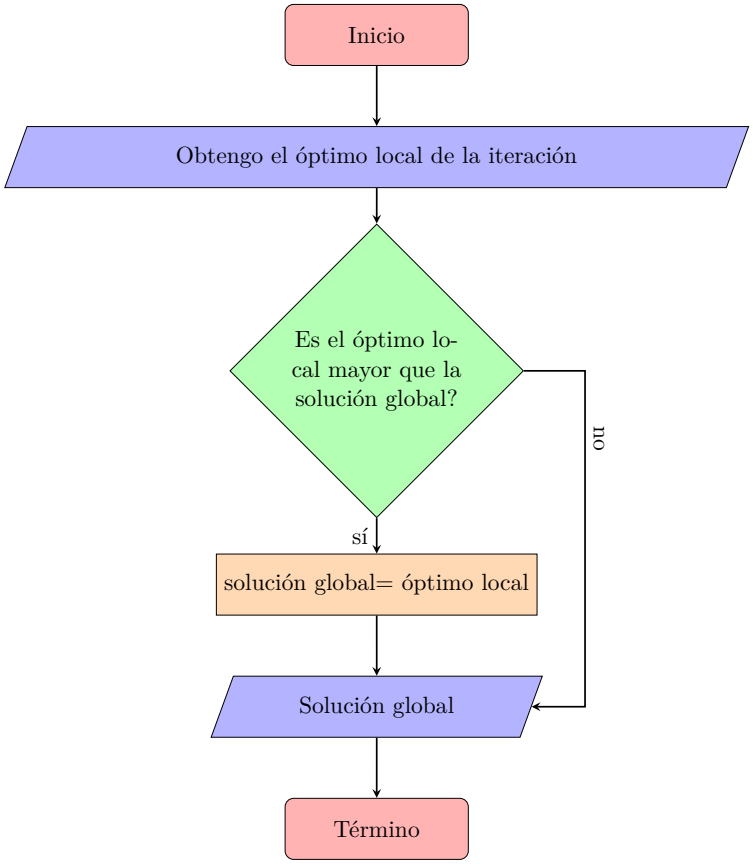



Figura 4.12: Actualización Global en Búsqueda Tabú

En esta etapa, el óptimo local de la iteración es comparado con el óptimo global encontrado hasta el momento. De ser superado, se actualiza el nuevo óptimo global como el óptimo local encontrado en la iteración actual.

El proceso de actualización de la lista tabú es descrito en la figura 4.13.

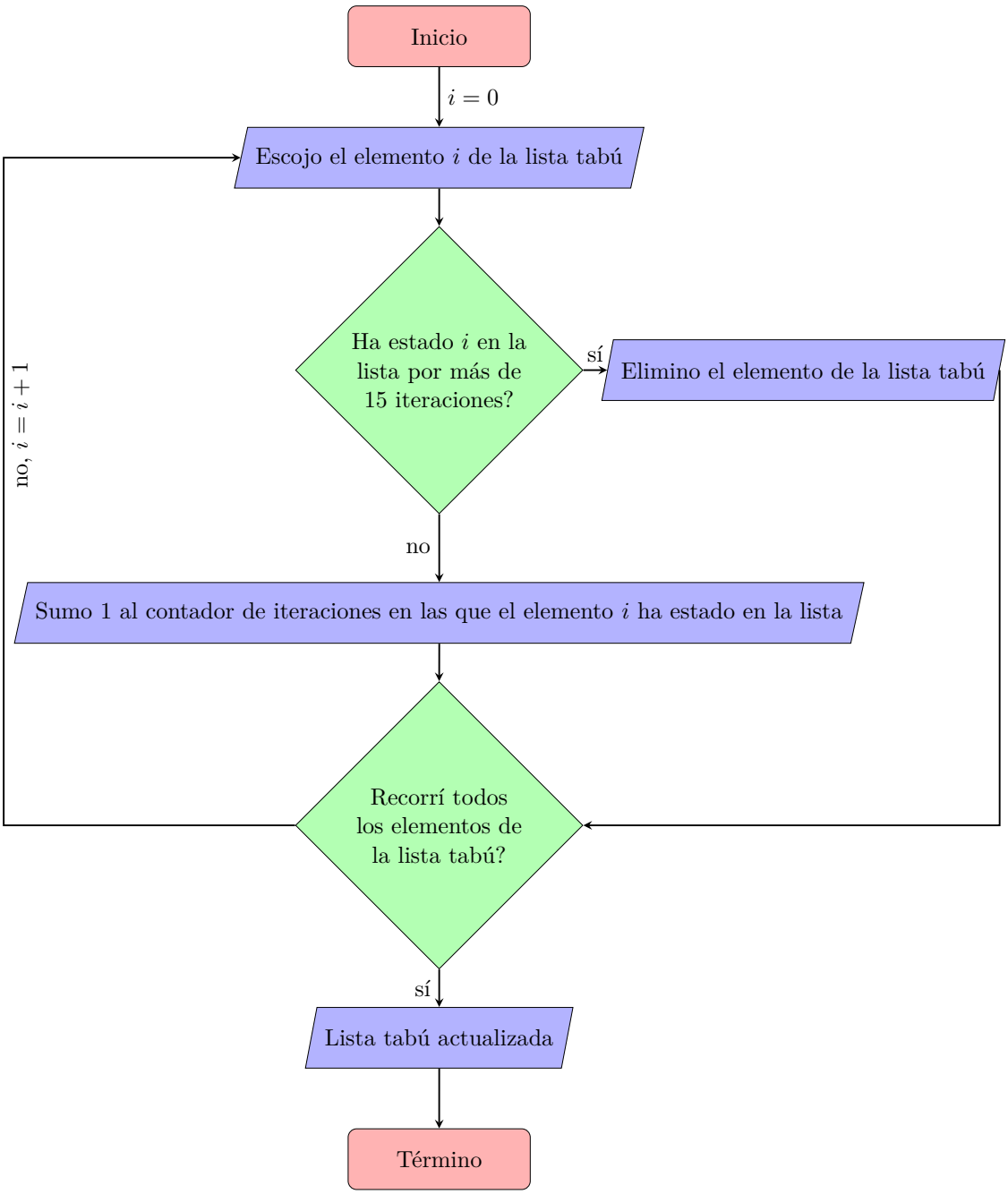



Figura 4.13: Actualización Tabú en Búsqueda Tabú

La actualización de la lista tabú se basa en eliminar los elementos que hayan estado más de 15 iteraciones dentro de la lista, por lo que en cada iteración se revisa cada elemento que esté en la lista tabú. Si dicho elemento ha estado más de las iteraciones señaladas es eliminado, sino, se suma 1 a su contador de iteraciones.

Finalmente, en la figura 4.14 se presenta la estructura de los movimientos realizados en la Búsqueda Tabú para poder obtener estados vecinos.

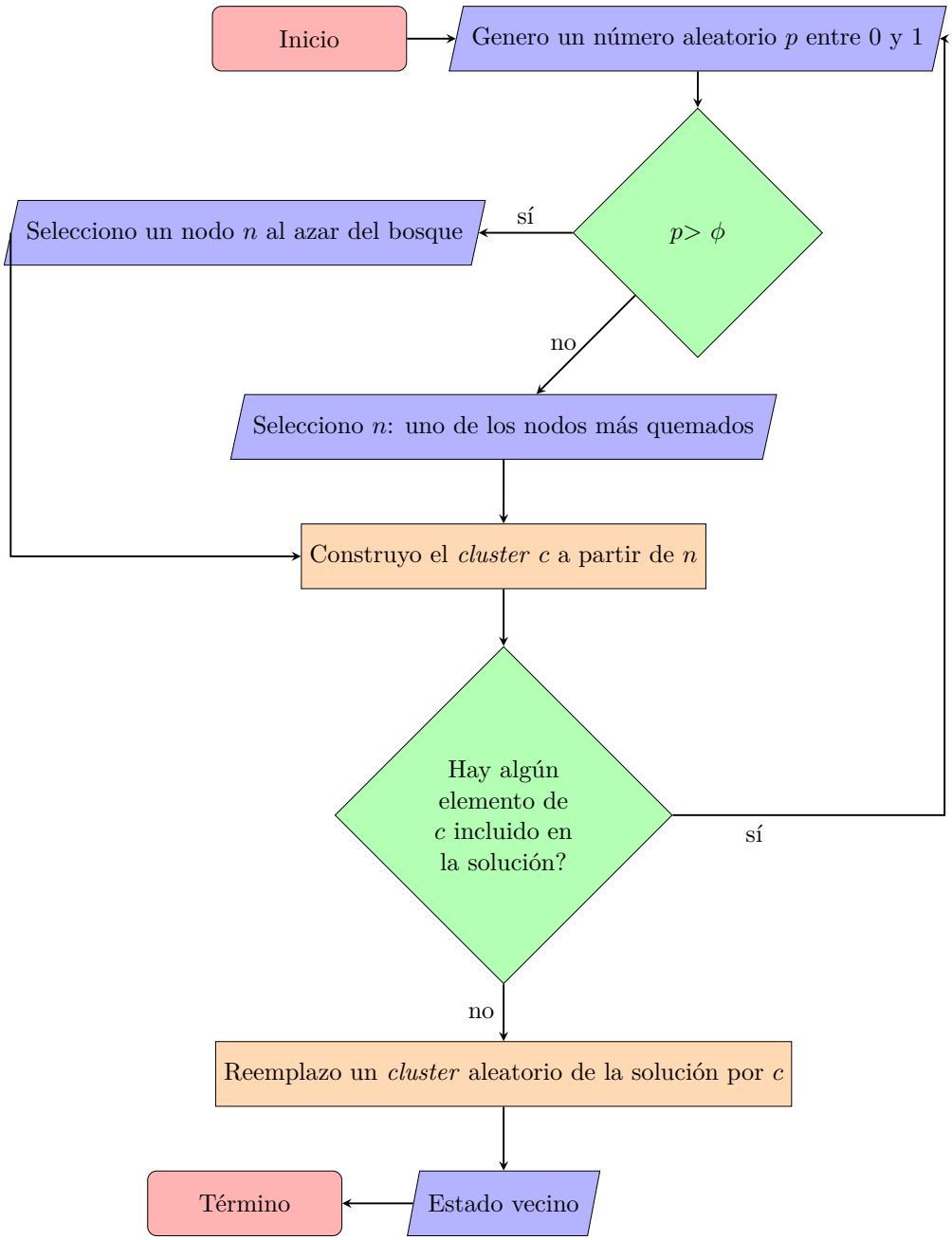



Figura 4.14: Movimiento a estado vecino en Búsqueda Tabú

En esta etapa del algoritmo, se modifican las soluciones de acuerdo al sesgo de movimiento $\phi$: si un número aleatorio es mayor a dicho sesgo, entonces se construye un *cluster* a partir de un nodo aleatorio del bosque, si ocurre lo contrario, entonces se construye un *cluster* a partir de uno de los nodos más quemados del bosque a lo largo de toda la ejecución del algoritmo. Posteriormente, se verifica si ya existe algún nodo del *cluster* creado en alguno de los otros nodos de la solución, de ser así, entonces se decide no construir este *cluster* y buscar un *cluster* alternativo. Esta verificación se hace para no construir *clusters* que se superpongan entre sí, ya que no tiene sentido el construir cortafuegos encima de otro. Finalmente, se reemplaza un *cluster* aleatorio de la solución por el *cluster* creado y se entrega la solución vecina.

Al igual que en el Algoritmo Genético, si se trabaja con nodos independientes, estas etapas son

equivalentes a diferencia de la diversificación y el movimiento. En la primera se selecciona un número de nodos equivalentes a la mitad de los *clusters* de la solución, y dichos nodos son modificados a estados vecinos. Mientras que en la etapa de movimiento, se escogen y se reemplazan los $c$ nodos más quemados, con $c$ el número de nodos que posee un *cluster*.

### 4.4.3 GRASP

El esquema básico del funcionamiento de GRASP es presentado en la figura 4.15.

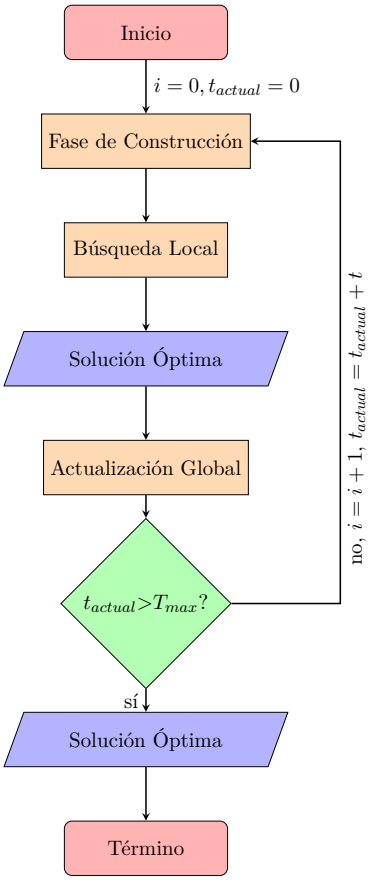



Figura 4.15: GRASP

GRASP inicia con una fase de construcción la cual puede generar una solución desde cero, o a partir de un cierto número de cortafuegos, evento que será analizado más adelante. Posteriormente, la solución es evaluada de acuerdo a la visita de un determinado número de estados vecinos, proceso del cual se obtiene un óptimo local, el cual permite actualizar la solución global. Al terminar todas las iteraciones, el algoritmo entrega la mejor solución que se haya encontrado hasta el momento.

En la figura 4.16 se describe la fase de construcción del algoritmo.

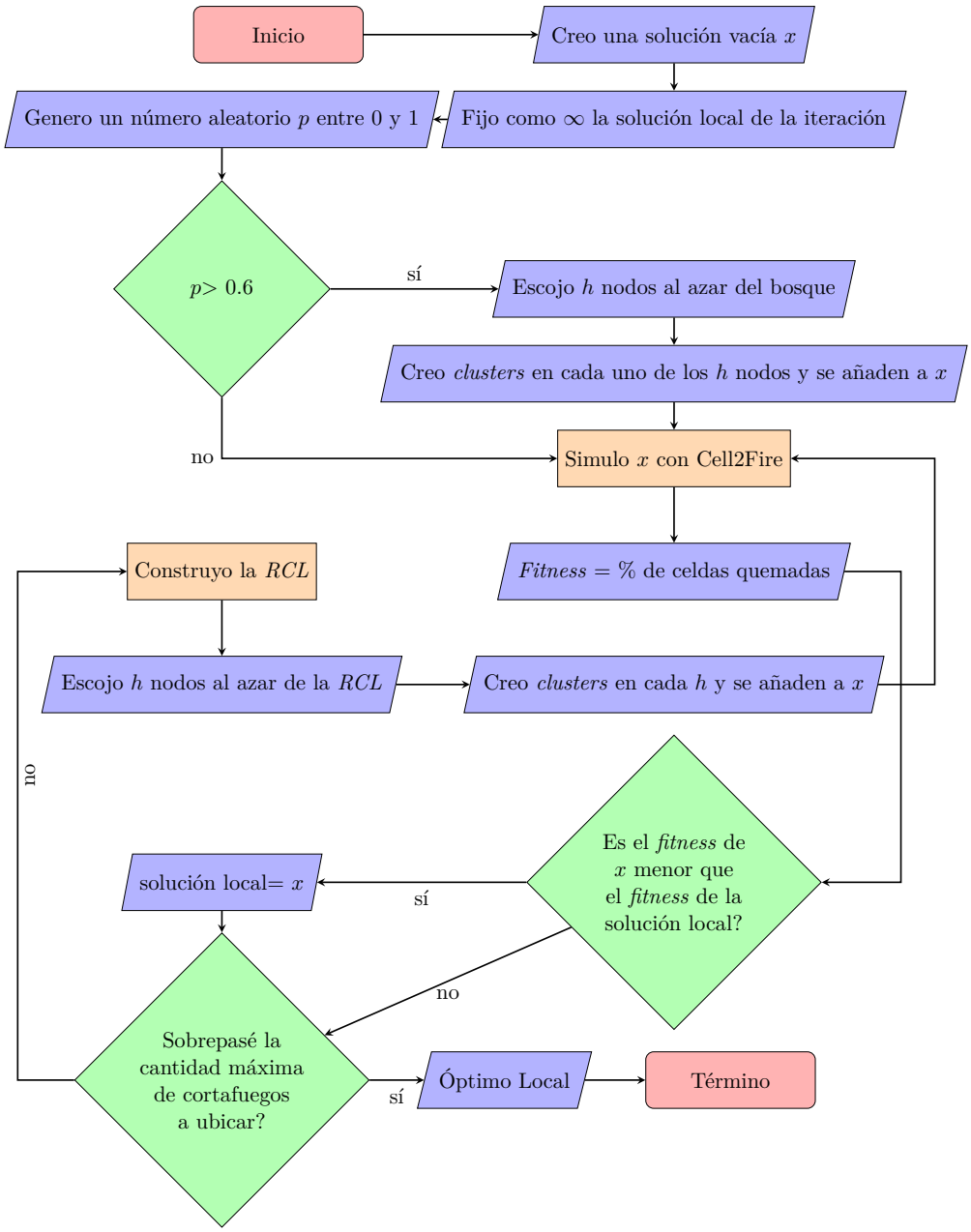



Figura 4.16: Fase de Construcción GRASP

En esta etapa se inicializa la iteración con un óptimo local de valor nulo. Posteriormente, en el 60% de los casos se simula el bosque con una solución vacía, y en 40% de los casos se simula con una solución parcial que contiene $h$ *clusters*, con $h = \lfloor \frac{c}{4} \rfloor$ de acuerdo a lo señalado en la sección 4.2. Mientras sea posible añadir *clusters* a la solución, se incluyen $h$ *clusters* a partir de $h$ nodos seleccionados desde la *RCL*. Además, se chequea cada una de las soluciones creadas para actualizar el óptimo local encontrado en la etapa de construcción, dicho óptimo suele ser la última solución construida en esta fase, ya que es el que posee más cortafuegos, pero esto no siempre ocurre así por lo que se debe revisar en cada etapa.

A continuación se presenta la fase de construcción de la *RCL* en la figura 4.17:

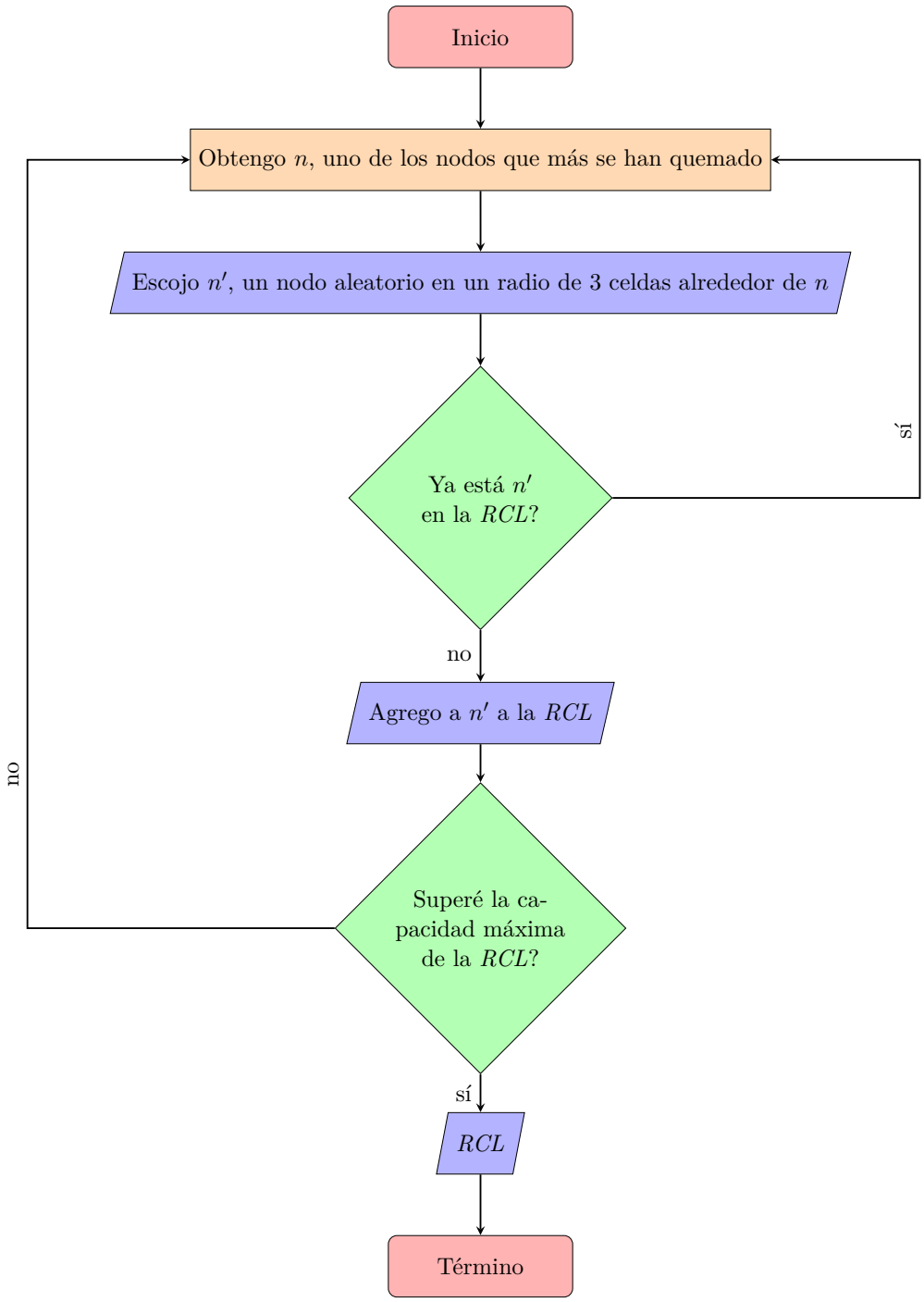



Figura 4.17: Construcción $RCL$ en GRASP

La $RCL$ se construye en cada iteración de la fase de construcción. Para seleccionar los mejores candidatos que componen esta lista, se escoge aleatoriamente uno de los nodos que más se han quemado a lo largo de la historia del algoritmo. Posteriormente, se selecciona un nodo al azar dentro de un radio de 3 celdas adyacentes al nodo seleccionado. Si este nodo no ha sido agregado a la $RCL$ de esta iteración y si tampoco se ha superado el tamaño máximo de la $RCL$, entonces se añade dicho nodo a la lista. Finalmente, se entrega la lista con los mejores candidatos a añadir a la solución.

En la figura 4.18 se expone la estructura de la fase de búsqueda local en GRASP:

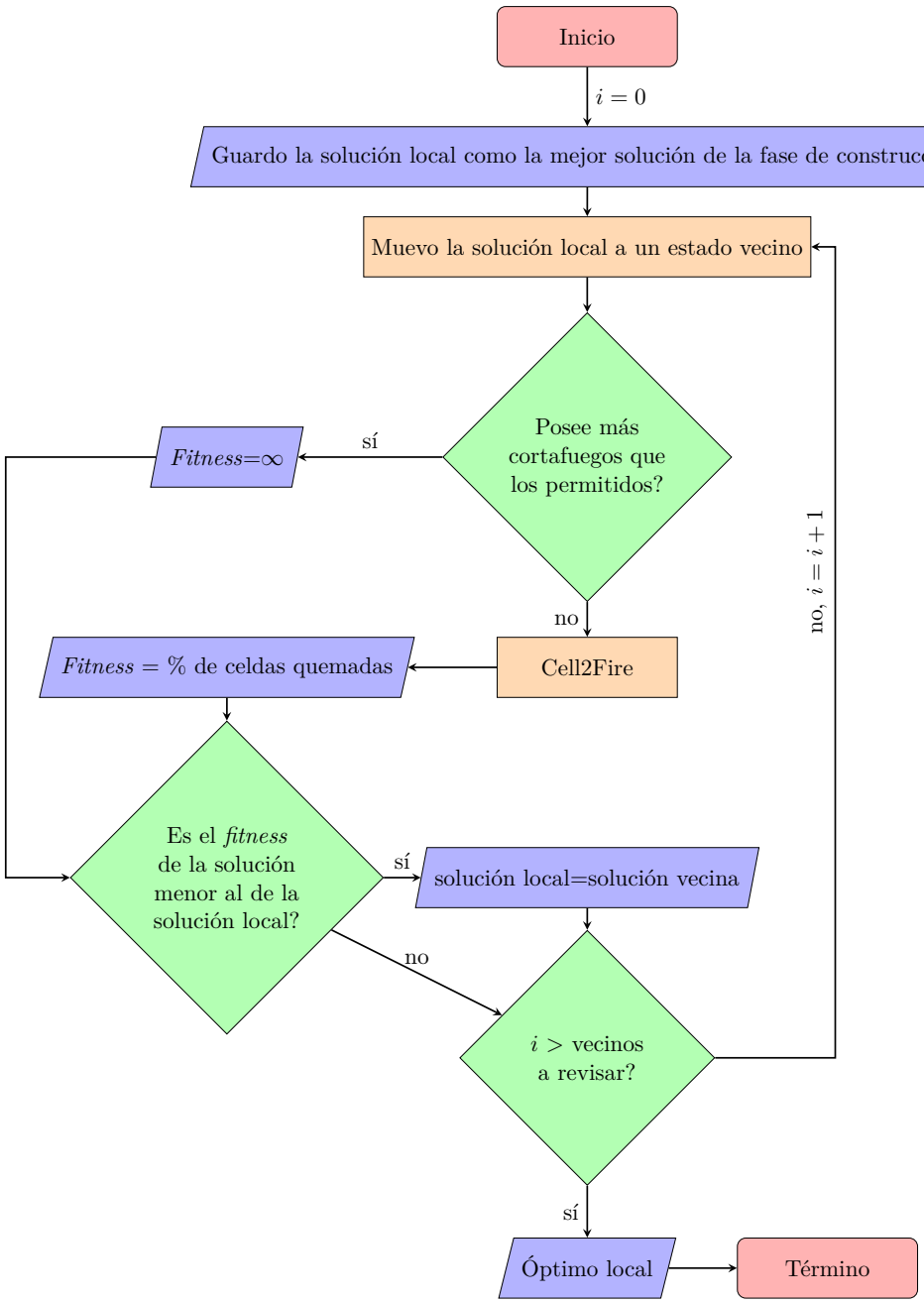



Figura 4.18: Búsqueda Local en GRASP

La búsqueda local en GRASP es similar a la de Búsqueda Tabú, salvo por el hecho de que no existe un chequeo debido a la lista tabú, ni se hace el mismo número de simulaciones en Cell2Fire (de acuerdo a lo señalado en la sección 4.2). El procedimiento es mover la solución local obtenida de la fase de construcción hacia distintos estados vecinos y evaluar su desempeño. Si se encuentra algún vecino con un desempeño mejor que la mejor solución local encontrada en esta iteración, entonces se actualiza dicho valor.

El procedimiento para mover una solución a un estado vecino es descrito en la figura 4.19:

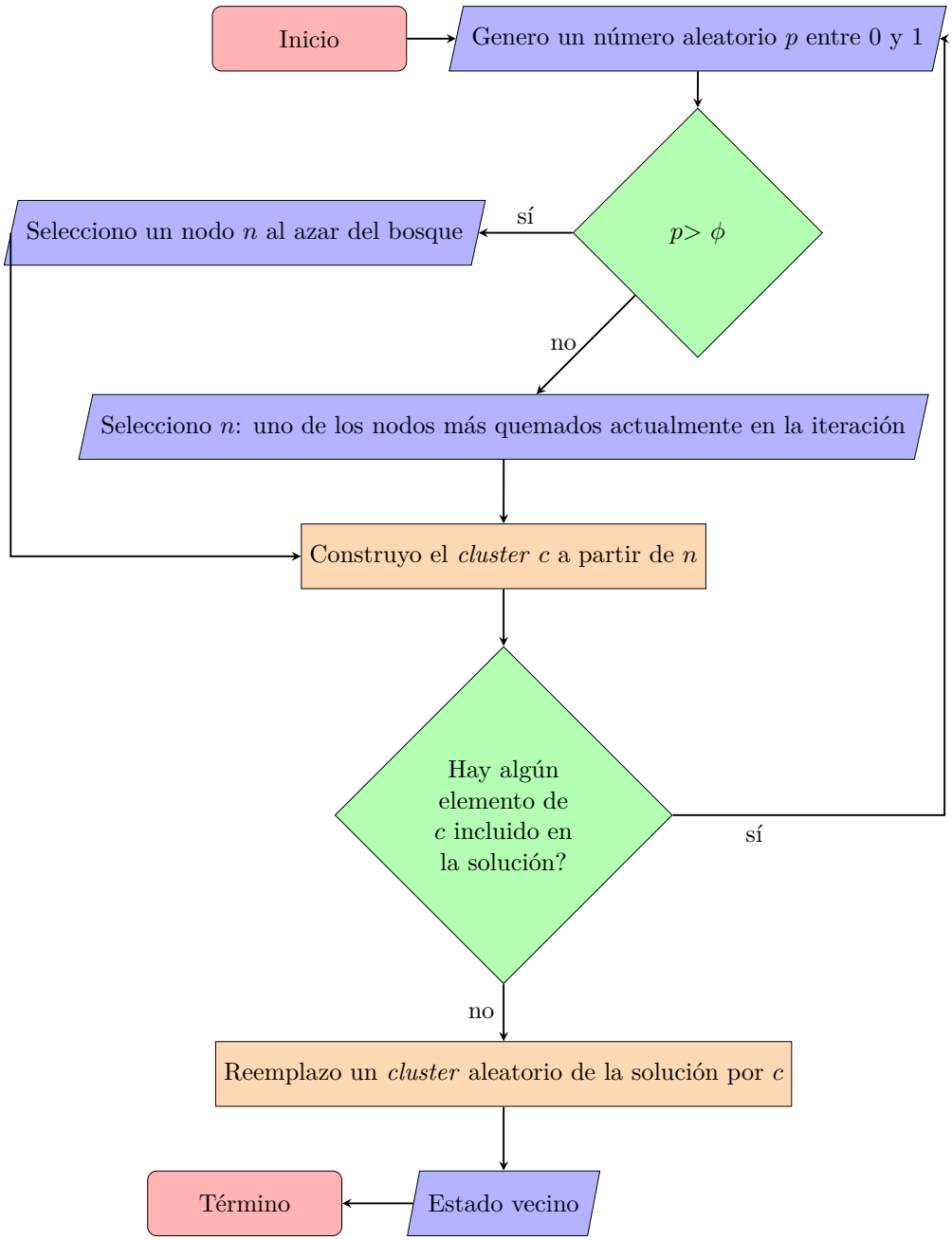



Figura 4.19: Movimiento a estado vecino en GRASP

En este caso, el movimiento se realiza de la misma forma que en Búsqueda Tabú. Es decir, las soluciones son modificadas según el sesgo de movimiento $\phi$, el cual indica si el movimiento será aleatorio o de acuerdo al historial de incendios anteriores. De existir cortafuegos en la solución que estén superpuestos con el *cluster* seleccionado, entonces se descarta dicho *cluster* y se busca otro. Si esto no ocurre, entonces se reemplaza un *cluster* aleatorio de la solución por el *cluster* recién creado y se entrega la solución vecina.

Finalmente, en la figura 4.20 se presenta la actualización global de esta metaheurística:

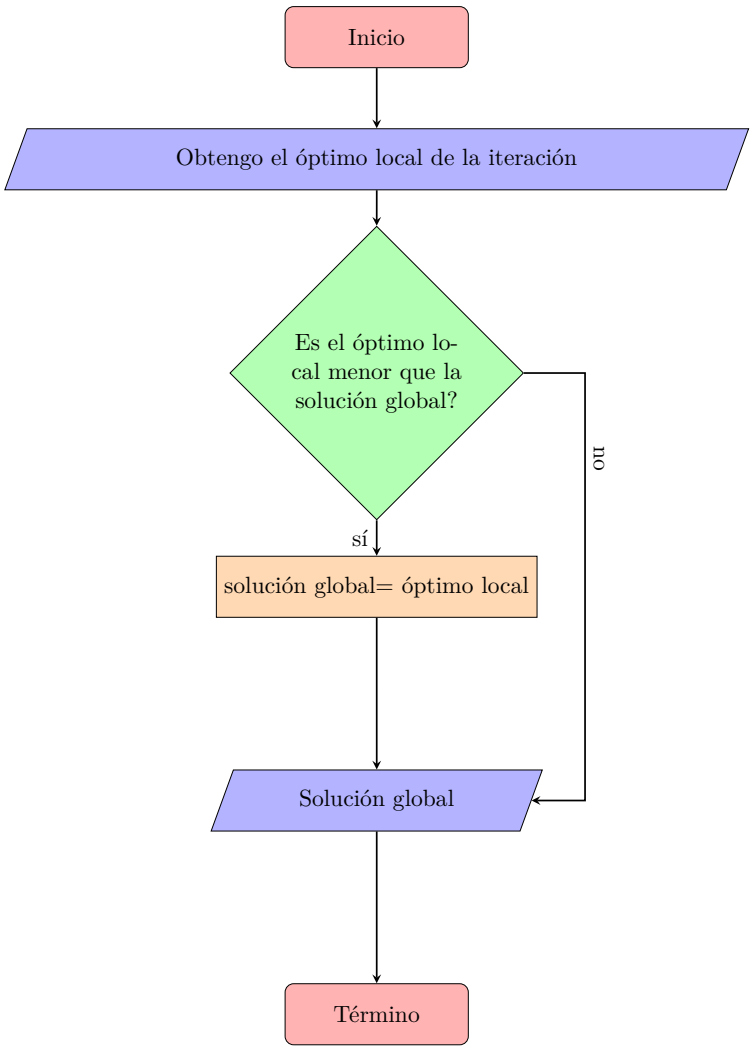



Figura 4.20: Actualización Global en GRASP

En esta etapa se compara al óptimo encontrado en la iteración frente a la mejor solución encontrada hasta el momento a lo largo del algoritmo. Si la solución de la iteración actual supera (es decir se quema menos bosque) a la mejor solución, entonces se reemplaza la mejor solución encontrada hasta el momento por la solución de la iteración.

Al igual que en las metaheurísticas anteriores, si se trabaja con nodos independientes, estas etapas funcionan de la misma forma. En la fase de construcción se ubican en cada iteración $h \cdot c$ nodos, con $c$ el tamaño del *cluster*. Situación similar ocurre en el movimiento a estados vecinos, en donde se reemplazan $c$ nodos.

## 4.5 Parámetros de Cell2Fire

El simulador Cell2Fire posee una serie de parámetros que pueden ser modificados de acuerdo a los requerimientos del usuario y así otorgar variabilidad y/o realismo a los escenarios simulados. En este trabajo se modificarán 2 parámetros: Dirección del viento y Punto de Ignición.

- Dirección del Viento: La dirección del viento modifica la dirección en la que se propagará el incendio. Para este trabajo, se considera más importante la dirección del viento que la

velocidad del mismo (u otras variables meteorológicas [15]) ya que define la topología que debe tener la configuración de cortafuegos para cubrir el crecimiento del incendio. Como ejemplo, si se asume una velocidad del viento y una temperatura ambiente constante y se genera un incendio cuando existe un viento con dirección al sur, los cortafuegos planificados pueden tener un desempeño distinto a si el viento apunta al este. Para el primer caso una barrera que cubra el bosque de este a oeste probablemente impedirá el avance del fuego de manera más óptima, mientras que dicha configuración de cortafuego no ayudará tanto en el segundo caso. En cambio, si se presenta un escenario con dirección del viento y temperatura ambiente constante pero velocidad del viento variable, probablemente el mismo cortafuego vaya a tener un desempeño similar en ambos casos, el mismo escenario ocurriría si hubiese una temperatura variable.

Así, en este trabajo se utilizarán 2 variaciones de viento: una con alta variabilidad y otra con menor variabilidad. En ambos casos se simula la misma dirección del viento a lo largo del horizonte de simulación, y las otras condiciones meteorológicas varían cada hora de la misma manera en cada uno de los escenarios de posibles direcciones. Para todos estos escenarios se utilizan condiciones climáticas extremas de la región de Alberta, Canadá. En el caso con alta variabilidad se utilizarán 8 vientos, cada uno apuntando a una determinada dirección de la rosa de los vientos de acuerdo a la figura 4.21.

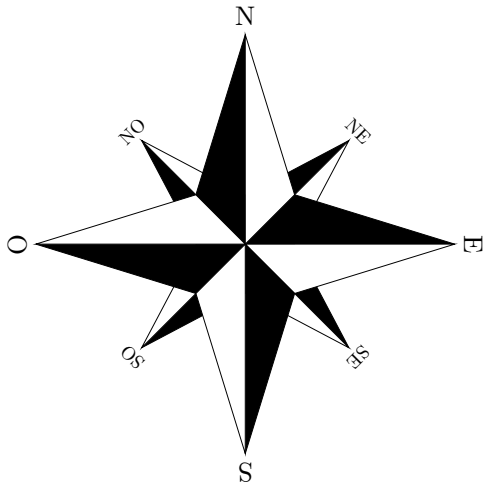



Figura 4.21: Rosa de los vientos representando las 8 posibles direcciones de los vientos.

El otro caso utiliza 30 vientos que apuntan en cada una de las posibles direcciones desde los 0º hasta los 29º, con el origen en la dirección sur y avanzando en dirección horaria, es decir, de acuerdo a la zona más oscura de la figura 4.22.

[15]Cell2Fire utiliza como variables metereológicas la temperatura, la humedad relativa, la velocidad y la dirección del viento y los valores asociados al *Canadian Forest Fire Weather Index System* (FWI) que usa 6 componentes (como la humedad del combustible) que determinan el comportamiento del fuego.

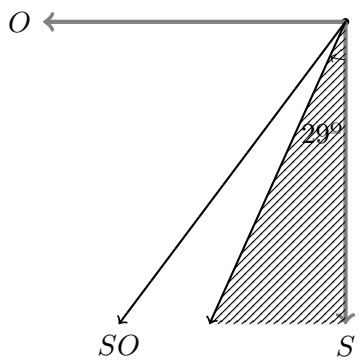



Figura 4.22: Representación de los posibles ángulos de dirección de los vientos.

Estas son las 2 posibles disposiciones de los vientos que se utilizan en este trabajo. La primera aporta alta variabilidad al problema y por ende una mayor robustez en los resultados, mientras que la segunda permite una menor variabilidad logrando así una convergencia más fácil para las metaheurísticas.

- Punto de ignición: El punto de ignición define el origen del incendio y por ende, el lugar desde donde se propagará el mismo. De existir solamente un punto de ignición, la solución es trivial y basta rodear dicho punto para reducir al mínimo el efecto del incendio. Por el contrario, de existir un número elevado de puntos de ignición, le costará más trabajo a las metaheurísticas el obtener buenas soluciones.

  En esta tesis, al igual que para las direcciones del viento, se utilizarán 2 enfoques para el punto de ignición: punto de ignición aleatorio a lo largo de todo el bosque y punto de ignición aleatorio dentro de un subconjunto de celdas del bosque.

  El primer escenario es el más común para incendios de origen no intencional, por ejemplo, ante caídas de rayos. Por ser completamente aleatorio le otorga mayor dificultad al problema pero más realismo y robustez a las soluciones en determinadas zonas geográficas donde los incendios tienden a provocarse por efectos no humanos, una de ellas Canadá [Stocks et al, 2002].

  Para el segundo caso, se utiliza un subconjunto de nodos del bosque de acuerdo a una determinada zona del mismo. Dicha zona puede observarse en la figura 4.23:

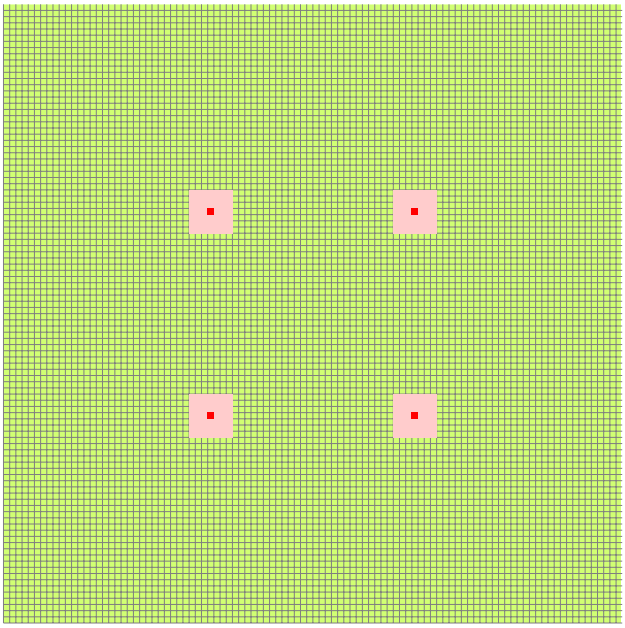


Figura 4.23: Zonas de Ignición en un bosque de dimensión 100x100 celdas

Se puede distinguir 4 zonas distintas en el bosque. Cada zona es generada a partir de un nodo (celda más roja) que permite una aleatoriedad de 3 celdas a la redonda desde las cuales puede partir el incendio. Es decir, el incendio tiene la posibilidad (equiprobable) de iniciarse tanto en la celda central de la zona (celda roja) como en las celdas adyacentes (celdas más claras) en un radio de 3 celdas a partir de dicho punto central. Cabe destacar que con el objetivo de no obtener soluciones triviales, se le prohíbe al algoritmo ubicar cortafuegos en cualquier punto de estas zonas mediante una penalización en la función objetivo de la misma forma que se hace al haber un número de cortafuegos mayor al máximo permitido. En la figura 4.24 se puede apreciar una vista ampliada de una de las zonas de ignición.

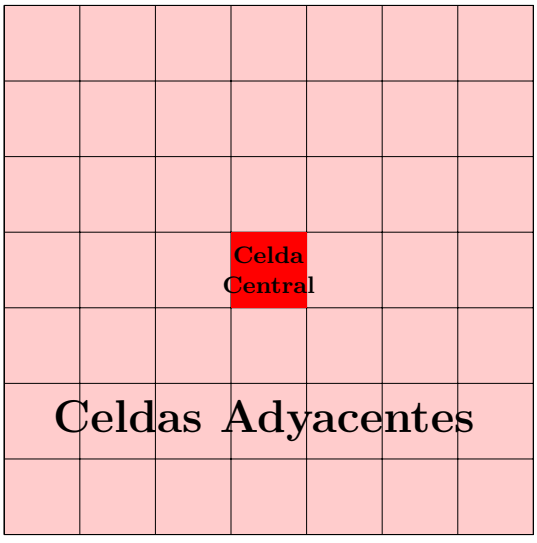



Figura 4.24: Vista ampliada de zona de ignición

Con este enfoque es posible obtener resultados con menos variabilidad y por ende más mane-

jables para los algoritmos. A pesar de que esto pueda ser visto como una simplificación del problema, existen zonas geográficas en donde los incendios forestales más significativos no se generan aleatoriamente en el bosque, sino que tienen su origen en intervenciones humanas. Por ejemplo en Chile, en donde los incendios generados en la interfaz entre las urbes y el bosque son los que generan mayor riesgo a la población [Úbeda y Sarricolea, 2016]. Por lo tanto, es posible obtener resultados realistas y significativos al trabajar con un subconjunto de puntos de ignición.

## 4.6 Bosques Utilizados

En este trabajo, se utilizan 4 bosques distintos para probar el desempeño de las metaheurísticas en diferentes escenarios. Así, se utilizan 2 bosques heterogéneos (con diversa vegetación) y 2 bosques homogéneos (con una sola vegetación).

- **Bosques heterogéneos**: Como Cell2Fire utiliza combustibles en base al sistema canadiense FBP, se utiliza el incendio *Dogrib*, el cual inició el 21 de septiembre de 2001 en un campamento abandonado en *Rocky Mountain*, al suroeste de la región de Alberta en Canadá. En la figura 4.25 se observa el paisaje del bosque incendiado previo al evento, el cual se compone de diversa vegetación, así como también zonas incombustibles (de color gris).

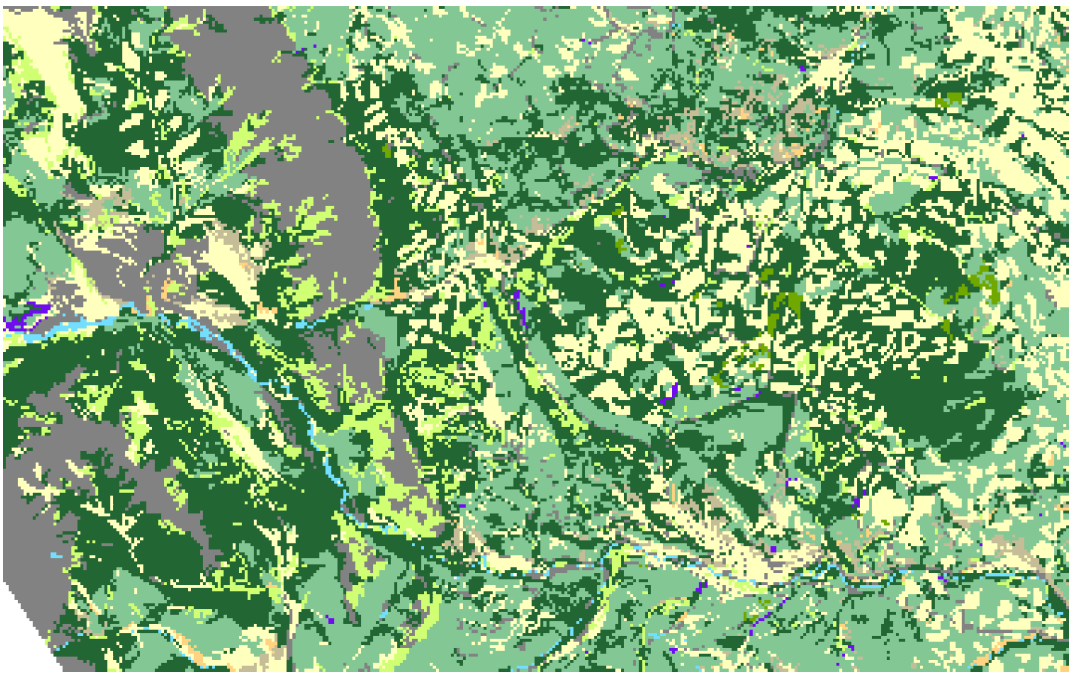
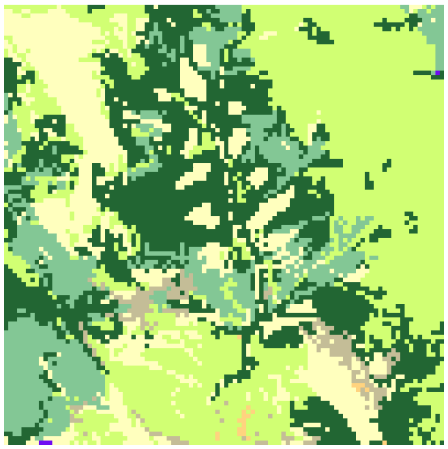
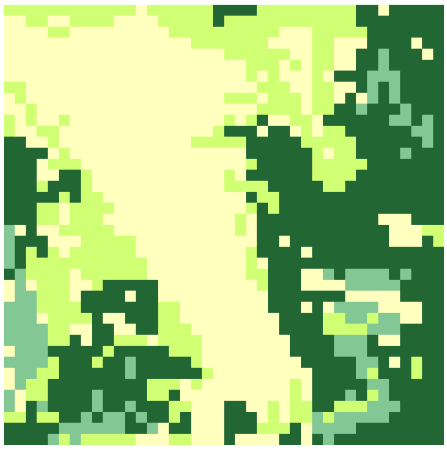


Figura 4.25: Paisaje previo a incendio *Dogrib*.

Este bosque es de dimensión 357x223 celdas, por lo que contiene 79.611 celdas, cada una de ellas de dimensión real 1 hectárea. En esta tesis se utilizarán sólo porciones de este bosque, ya que simular incendios para 79.611 celdas requiere un alto consumo de tiempo. La primera porción será una de dimensión 100x100 celdas, denominada bosque **a)**. Esta porción contiene celdas incombustibles, las cuales son reemplazadas por una de las vegetaciones más combustibles del FBP, el *Spruche-Liche Woodland*, denominada $C1$ en dicho sistema. Además, también será utilizada otra porción más pequeña del bosque **a)** de dimensión 40x40 celdas, la que será llamada bosque **b)**. A continuación se pueden apreciar ambos bosques:



(a) Porción de bosque *Dogrib* de 100x100



(b) Porción de bosque *Dogrib* de 40x40

Figura 4.26: Bosques **a)** y **b)**

- **Bosques Homogéneos**: En el caso de los bosques homogéneos, se utiliza una porción de la misma dimensión que los bosques heterogéneos, pero con una vegetación uniforme a lo largo del paisaje. Dicha vegetación es la ya mencionada $C1$. En la figura 4.27 se puede observar el bosque homogéneo de 100x100 celdas, denominado bosque **c)**. El bosque homogéneo de 40x40 celdas, denominado bosque **d)** es un bosque más pequeño que contiene la misma vegetación (su paisaje no es incluido ya que posee las mismas características que el bosque **c)**).



Figura 4.27: Bosque homogéneo de *Spruche-Liche Woodland* de 100x100 celdas, bosque **c)**

# 5 Resultados

Todas las simulaciones fueron realizadas en un procesador Intel® Core™ i5-8265U CPU @ 1.60GHz con 12 GB de RAM. Los códigos fueron escritos en el lenguaje de programación Python y el sistema operativo utilizado es Ubuntu 18.04 no nativo.

## 5.1 Comparación de algoritmos según variación

A continuación se presentan los resultados obtenidos por cada uno de los algoritmos según variaciones descritas en la sección 4.3. Todos estos experimentos son realizados para un incendio de 50 horas de duración que ocurre en el bosque **a)**, con punto de ignición aleatorio en el subconjunto de celdas señalado en la sección 4.5 y con vientos en cualquiera de las 8 direcciones señaladas en la misma sección. Cada una de las pruebas son realizadas en 2 horas y se realizan 5 pruebas con distintas semillas aleatorias por cada algoritmo en cada variación. Finalmente, se debe señalar que todos los resultados aquí expuestos son la evaluación del óptimo obtenido por cada algoritmo en cada prueba. Este óptimo se evalúa en base a 40 simulaciones en Cell2Fire con una misma semilla aleatoria para cada óptimo de cada algoritmo.

### 5.1.1 Algoritmo Genético

Como se señala en la sección 4.2, en el Algoritmo Genético se utiliza una población de 10 individuos. Además, se evalúa a cada individuo de la generación con 20 simulaciones en Cell2Fire, logrando un promedio de aproximadamente 27 generaciones en cada prueba.

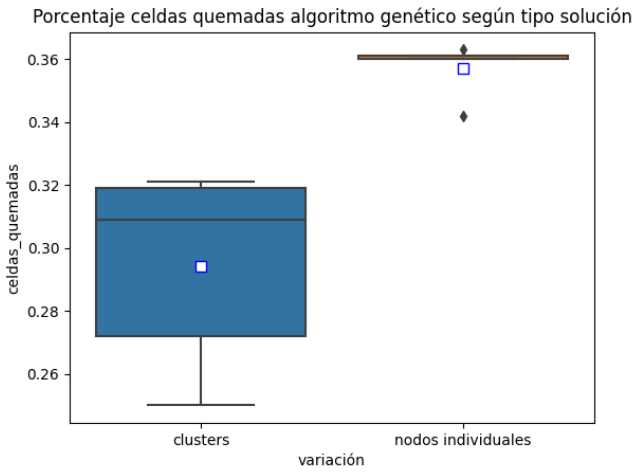



Figura 5.1: Desempeño Algoritmo Genético según tipo de solución.

Es posible apreciar que el uso de soluciones tipo *clusters* logran un desempeño mucho mejor que al usar soluciones como nodos no necesariamente conexos, teniendo en el primer caso un promedio de aproximadamente 29.5% del bosque quemado, mientras que en la segunda situación se obtiene un promedio de casi un 36% de la superficie total del bosque incendiada.

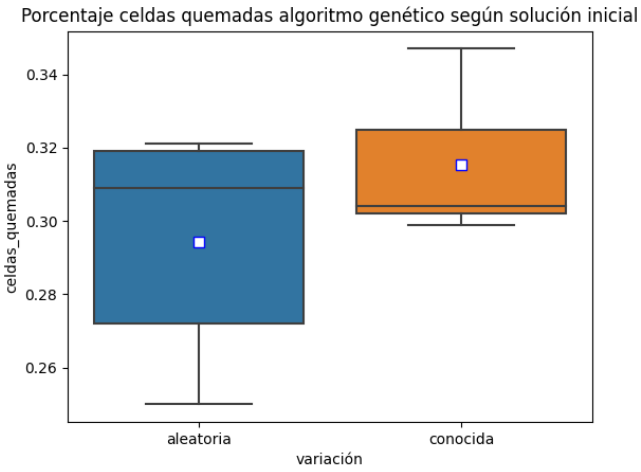



Figura 5.2: Desempeño Algoritmo Genético según solución inicial.

En este caso, al utilizar una solución inicial aleatoria de los individuos de la primera generación se obtiene un desempeño levemente mejor que al utilizar la solución utilizada en [Martínez, 2021], con un 29.5% del bosque quemado en el primer escenario y un 31.5% en el segundo.

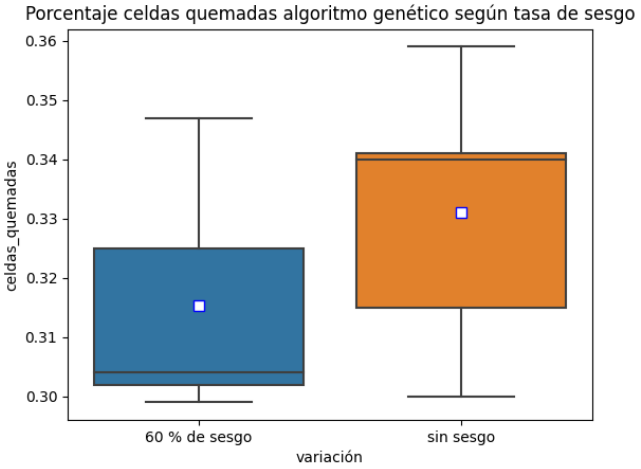



Figura 5.3: Desempeño Algoritmo Genético según sesgo de movimiento.

Finalmente, al utilizar la información de las generaciones anteriores para realizar la mutación de la generación actual se logran también mejores resultados que si se realiza un movimiento aleatorio, logrando un 31.6% de las celdas quemadas en el primer caso y un 33.3% en el segundo.

Por lo tanto, se concluye que la mejor versión del Algoritmo Genético para este escenario se obtiene utilizando soluciones tipo *clusters*, con solución inicial aleatoria y con mutaciones en base al comportamiento de las generaciones anteriores.

### 5.1.2 Búsqueda Tabú

En la Búsqueda Tabú se realizan 10 simulaciones en Cell2Fire a cada uno de los 10 vecinos encontrados en la fase de búsqueda local. En este algoritmo se ejecutan aproximadamente 80 iteraciones.

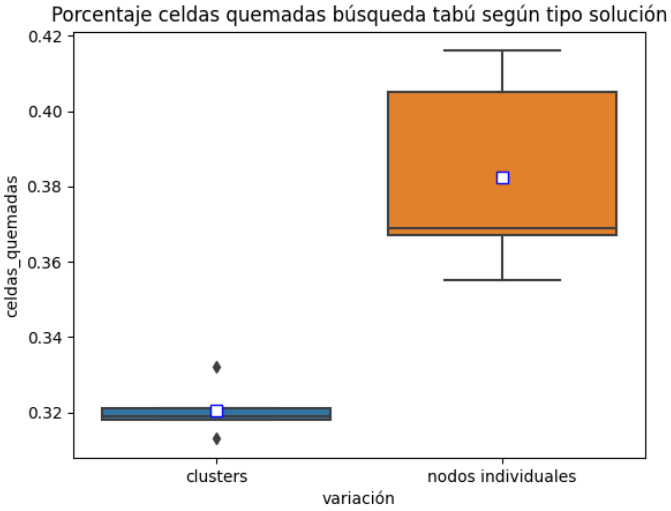



Figura 5.4: Desempeño Búsqueda Tabú según tipo de solución.

En la primera figura, se puede notar que al igual que en el algoritmo genético se logran mejores resultados utilizando soluciones tipo *clusters* en desmedro de soluciones inconexas, quemando un 29% de la superficie en el primer caso y un 38% en el segundo.

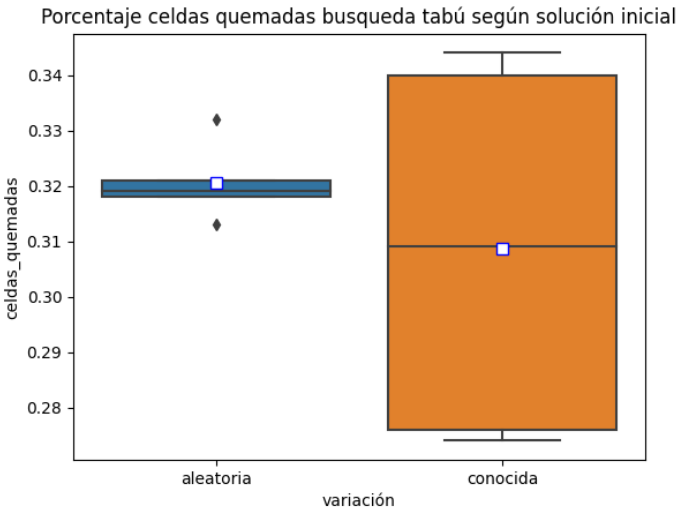



Figura 5.5: Desempeño Búsqueda Tabú según solución inicial.

En el caso de la solución inicial del algoritmo, es posible notar que se logra un promedio menor cuando se utiliza la solución inicial conocida, pero con un amplio margen de error comparado con el escenario que se utiliza una solución inicial aleatoria.

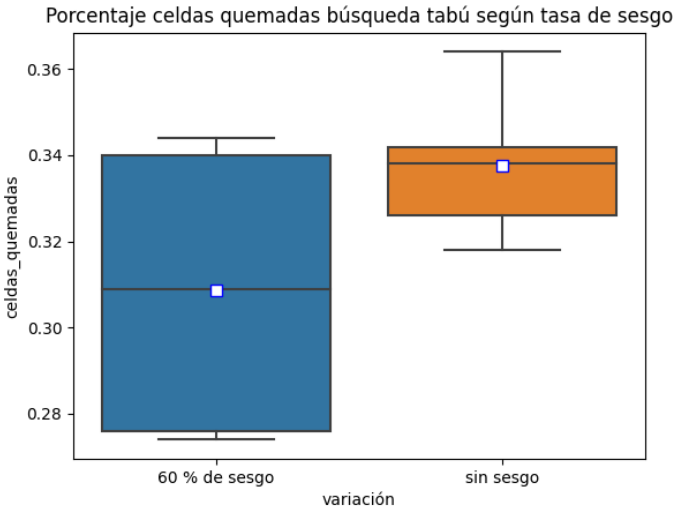



Figura 5.6: Desempeño Búsqueda Tabú según sesgo de movimiento.

Al igual que en el Algoritmo Genético, el utilizar la información de iteraciones anteriores para saber qué estados vecinos visitar permite salvar una mayor porción de bosque, con un promedio de 31% de las celdas quemadas, mientras que al realizar movimientos totalmente aleatorios se quema un 34% del bosque.

Así, se define la mejor versión del algoritmo con soluciones tipo *clusters*, con solución inicial aleatoria (ya que posee una mejor dispersión de los datos y con un promedio similar) y utilizando la información de las iteraciones anteriores para realizar los movimientos.

### 5.1.3 GRASP

En el caso de GRASP, como el algoritmo no cuenta con una solución inicial (y se construye en base a una metodología *greedy*) no se compara esta variación. El algoritmo utiliza 5 simulaciones en Cell2Fire para el bosque vacío para definir dónde ubicar los primeros cortafuegos, posteriormente realiza 3 simulaciones en Cell2Fire en cada una de las 4 o 5 etapas (según el caso explicado en la sección 4.2) de la fase de construcción para ir colocando cortafuegos y utiliza 1 simulación en Cell2Fire para evaluar a cada uno de los 5 estados vecinos a considerar. Finalmente, al igual que en Búsqueda Tabú, se ejecutan 20 simulaciones en el simulador al mejor candidato de la generación para re-evaluar su función objetivo.

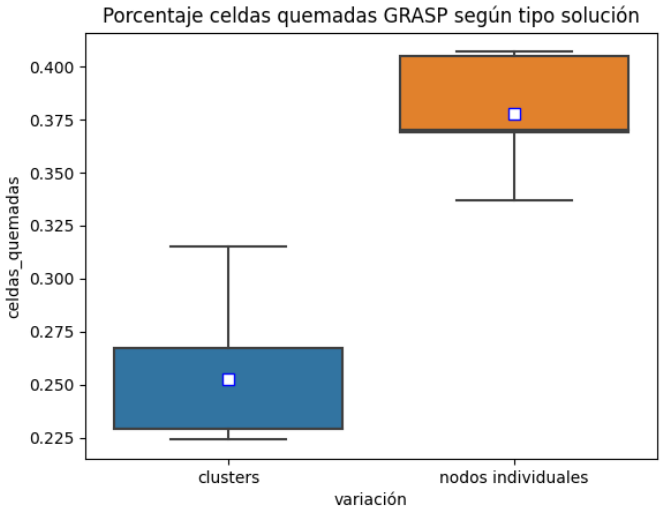



Figura 5.7: Desempeño GRASP según tipo de solución.

Al igual que en los algoritmos anteriores, la solución tipo *cluster* logra mejores resultados que la solución como nodos individuales, logrando en promedio 26% del bosque quemado en el primer escenario y 38% en el segundo.

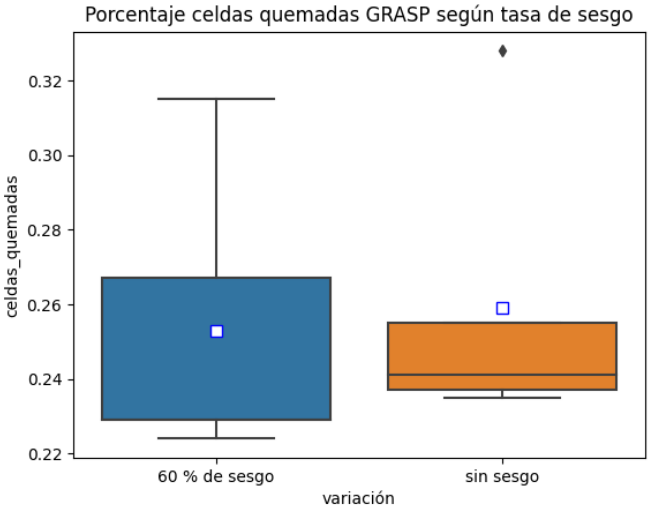



Figura 5.8: Desempeño GRASP según sesgo de movimiento.

Finalmente, el utilizar el sesgo de movimiento en la búsqueda local de GRASP no aporta tanto como en los algoritmos anteriores, ya que logra un desempeño en promedio similar (25.5% vs 26%) pero con una amplia dispersión al utilizar dicho sesgo.

Por ende, la mejor versión de GRASP para este caso es utilizar soluciones tipo *cluster* y sin utilizar la información de iteraciones anteriores.

## 5.2 Comparación de mejor variación en distintos escenarios

A continuación se presentan los resultados obtenidos por las metaheurísticas en diferentes escenarios en los que se modifica: el bosque, las direcciones del viento y el tipo de punto de ignición. Se busca crear escenarios con alta, media y baja variabilidad [16]. El caso con mayor variabilidad consiste en un incendio en un bosque homogéneo con vientos en las 8 direcciones de la rosa de los vientos y con punto de ignición aleatorio, mientras que el caso con menor variabilidad ocurre en un bosque heterogéneo con vientos en una apertura de 30º y con punto de ignición en las zonas cuadradas descritas en la sección 4.5.

Para estas pruebas se utilizan las mejores versiones de los algoritmos a partir de lo realizado en la sección anterior, mientras que el número de simulaciones, pruebas y tiempo total de ejecución se mantiene igual. Para comparar los resultados se usan las 3 metaheurísticas con: (i) un bosque con el 5% de su superficie utilizada como cortafuegos tipo *clusters* en forma de arco dispuestos aleatoriamente en el bosque; (ii) un bosque totalmente libre de cortafuegos y (iii) un bosque con la disposición de cortafuegos de la figura (llamada a partir de ahora solución *Martínez*), la cual también posee un 5% de la superficie total del bosque como cortafuegos. Para los bosques **b)** y **c)** no existe un tipo de solución *Martínez*, por lo que dicha configuración no se compara en aquellos escenarios que usan los bosques mencionados.

### 5.2.1 Bosque a) con vientos en 8 direcciones y punto de ignición en cuadrado

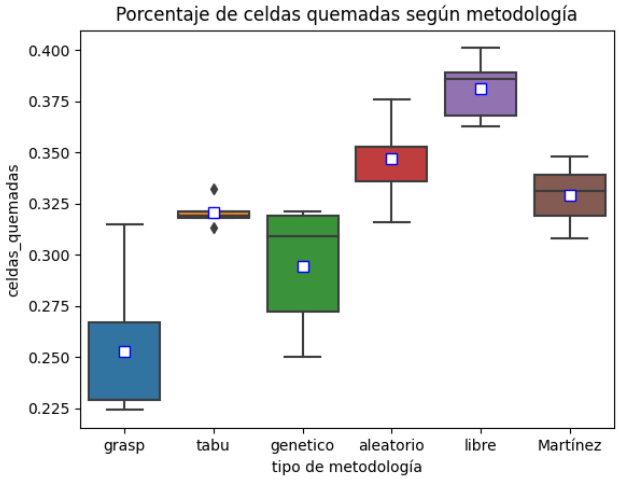



Figura 5.9: Desempeño bosque **a)** con vientos en 8 direcciones y punto de ignición en cuadrado.

En este escenario de variabilidad media se logra apreciar que la metaheurística GRASP tiende a tener un mejor desempeño que los otros tipos de metodología. Al mismo tiempo, tanto el Algoritmo Genético como la Búsqueda Tabú logran un desempeño algo mejor que la solución aleatoria y la solución *Martínez*. En promedio, las 3 metaheurísticas otorgan una mejora con respecto a la mejor solución de las metodologías anteriores (en este caso la solución *Martínez*), con aproximadamente 7%, 3.6% y 0.5% menos de superficie quemada usando la metaheurística GRASP, Algoritmo Genético y Búsqueda Tabú, respectivamente. De la misma forma, las 3 metaheurísticas suponen

[16]La definición de alta, media y baja variabilidad van de acuerdo a las variables utilizadas en este trabajo, ya que pueden existir escenarios con una aleatoriedad aún mayor si se incluyen otras variables.

una mejora promedio de aproximadamente 13%, 9% y 6% respectivamente en la porción de bosque quemado con respecto a dejar el bosque sin cortafuegos.

### 5.2.2 Bosque a) con vientos en 8 direcciones y punto de ignición aleatorio

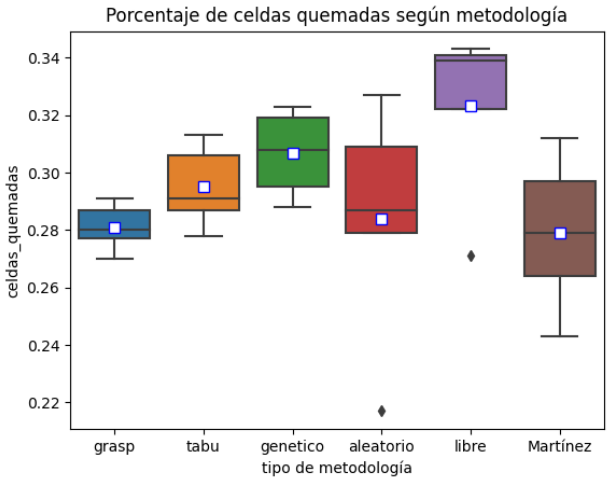



Figura 5.10: Desempeño bosque **a)** con vientos en 8 direcciones y punto de ignición aleatorio.

Este caso corresponde a un escenario de variabilidad alta. Aqui, se puede observar que ninguna de las metaheurísticas logra en promedio un desempeño mejor que la solución *Martínez*. Al compararlas entre ellas, GRASP es la que tiene resultados más cercanos a la mejor solución, quemando aproximadamente un 0.1% más de bosque. Mientras que tanto la Búsqueda Tabú como el Algoritmo Genético no logran superar la solución aleatoria, quemando aproximadamente un 2% y un 1% más de bosque respectivamente.

### 5.2.3 Bosque a) con vientos en 30 direcciones y punto de ignición en cuadrado

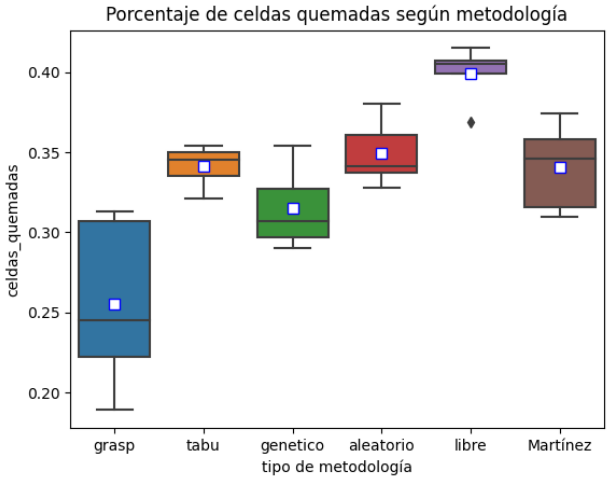



Figura 5.11: Desempeño bosque **a)** con vientos en 30 direcciones y punto de ignición en cuadrado.

Este escenario posee una baja variabilidad y en él se logra apreciar que GRASP alcanza mejores resultados aunque con un alto margen de error. Aún así, en su peor prueba logra resultados similares a la mejor prueba del Algoritmo Genético y de la solución *Martínez*. Búsqueda Tabú tiene un desempeño en promedio similar a la solución *Martínez* y levemente mejor a la solución aleatoria. En promedio, GRASP logra superar a la solución *Martínez* en aproximadamente 9.5% menos de superficie quemada y un 7% comparado con el Algoritmo Genético. Las 2 mejores metaheurísticas de esta prueba logran salvar aproximadamente 7% (en el caso del Algoritmo Genético) y 14.5% (en el caso de GRASP) más de superficie del bosque al compararlas con el escenario en que no se ubican cortafuegos.

### 5.2.4 Bosque a) con vientos en 30 direcciones y punto de ignición aleatorio

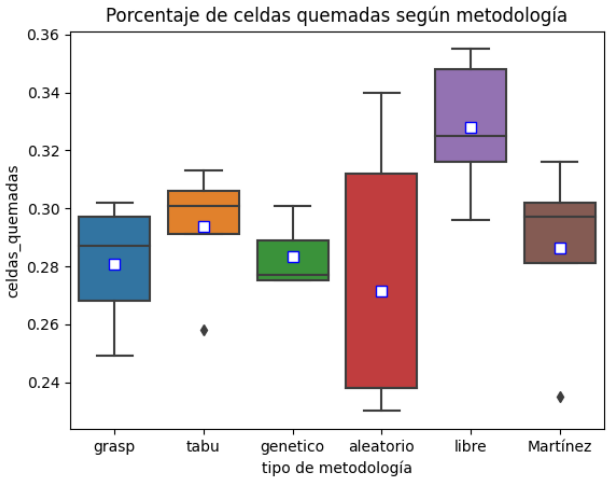



Figura 5.12: Desempeño bosque **a)** con vientos en 30 direcciones y punto de ignición aleatorio.

En este caso de variabilidad media, ninguna de las metaheurísticas logra superar en promedio a la solución aleatoria y tanto Búsqueda Tabú como el Algoritmo Genético tienen un desempeño similar a la solución *Martínez*, teniendo la primera resultados peores que la segunda. GRASP nuevamente supera a las otras metaheurísticas pero en menor magnitud (quema 0.5% menos de superficie de bosque que el Algoritmo Genético y 1.5% menos que Búsqueda Tabú) y no logra superar por 1% el desempeño de la solución aleatoria aunque sí es capaz de mejorar en un 0.8% la superficie quemada con la solución *Martínez*.

### 5.2.5 Bosque c) con vientos en 8 direcciones y punto de ignición en cuadrado

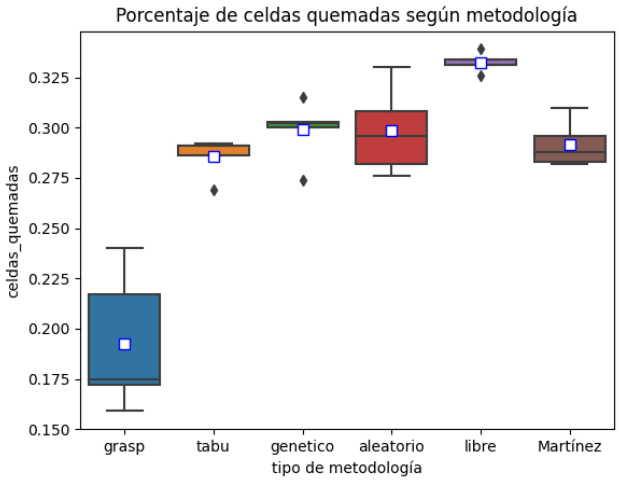



Figura 5.13: Desempeño bosque **c)** con vientos en 8 direcciones y punto de ignición en cuadrado.

En este bosque homogéneo de variabilidad media GRASP vuelve a superar al resto de metaheurísticas, logrando quemar aproximadamente 9.5% menos de superficie que la mejor otra metodología (Búsqueda Tabú). De la misma forma, quema 14% menos de bosque que la alternativa de dejar el bosque sin cortafuegos. Búsqueda Tabú y el Algoritmo Genético tienen resultados similares entre sí y entre la solución aleatoria y la *Martínez*. En el caso de Búsqueda Tabú logra superar a las otras mencionadas pero no más allá de un 0.5% (al compararla con la solución *Martínez*), mientras que el Algoritmo Genético no es capaz de obtener mejores resultados que el resto de metodologías.

### 5.2.6 Bosque c) con vientos en 30 direcciones y punto de ignición en cuadrado

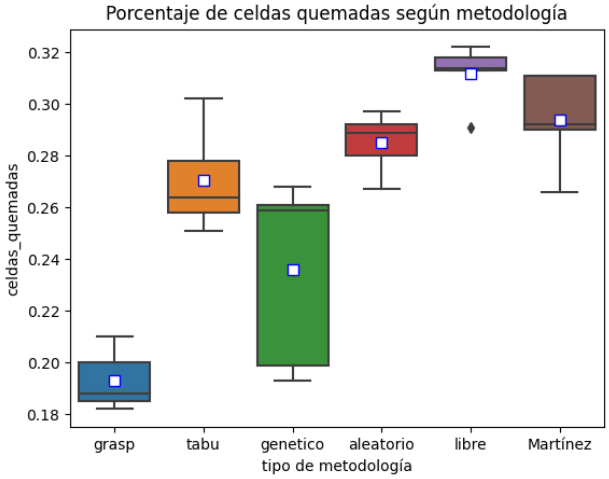



Figura 5.14: Desempeño bosque **c)** con vientos en 30 direcciones y punto de ignición en cuadrado.

Este escenario posee una variabilidad baja y al igual que en las otras pruebas GRASP logra los mejores resultados. En este caso todas las metaheurísticas logran superar en promedio a las otras metodologías, pero con un alto margen de error para la Búsqueda Tabú y uno aún mayor en el Algoritmo Genético. En promedio, GRASP salva aproximadamente 9.5% y 10% más de superficie que la solución aleatoria y la *Martínez* respectivamente, y un 12% más que la alternativa libre de cortafuegos. Mientras que supera por 8% y 4% en la porción de bosque quemado a la Búsqueda Tabú y al Algoritmo Genético respectivamente.

### 5.2.7 Bosque b) con vientos en 8 direcciones y punto de ignición en cuadrado

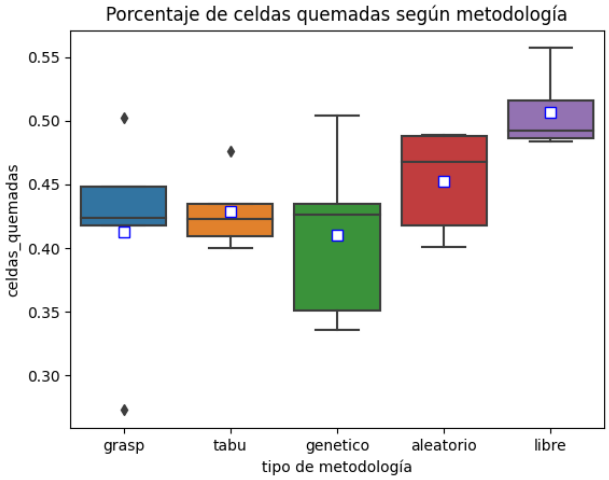



Figura 5.15: Desempeño bosque **b)** con vientos en 8 direcciones y punto de ignición en cuadrado.

En este escenario de variabilidad media el Algoritmo Genético y GRASP tienen un desempeño similar, logrando el primero un mejor promedio pero con un mayor error a lo largo de las pruebas. En general las 3 metaheurísticas salvan en promedio más bosque que la solución aleatoria (4%, 3% y 1.5% para el Algoritmo Genético, GRASP y Búsqueda Tabú respectivamente), pero no logran superarla en demasía y sus peores pruebas tienen resultados similares e incluso peores (en el caso de GRASP y el Algoritmo Genético) que dicha solución. Finalmente, GRASP, Búsqueda Tabú y el Algoritmo Genético queman en promedio 8%, 6.5% y 9% menos de superficie que la opción libre de cortafuegos, respectivamente.

### 5.2.8 Bosque d) con vientos en 8 direcciones y punto de ignición en cuadrado

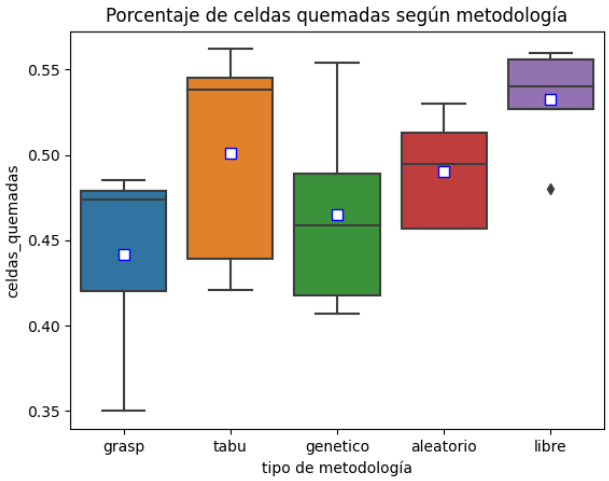



Figura 5.16: Desempeño bosque **d)** con vientos en 8 direcciones y punto de ignición en cuadrado.

En el pequeño bosque homogéneo de media variabilidad GRASP y el Algoritmo Genético superan en promedio a la solución aleatoria en 5% y 2% respectivamente. Búsqueda Tabú no logra mejorar con respecto a la solución aleatoria por 1% de superficie quemada. En promedio, GRASP, Búsqueda Tabú y el Algoritmo Genético superan en 9%, 3% y 6% respectivamente a la porción de bosque quemado en la solución sin cortafuegos.

## 5.3 Análisis de sensibilidad en porcentaje de cortafuegos

A continuación se presentan los resultados obtenidos al alterar el máximo porcentaje permitido de cortafuego a ubicar, variándolo entre 5% (caso base), 7% y 10% de la superficie total del bosque. Estas pruebas se realizaron en el bosque **a)** con vientos en las 8 direcciones de la rosa de los vientos y con un punto de ignición aleatorio, por lo tanto el caso base es similar al de la figura 5.10. Al igual que en las pruebas anteriores, estos resultados se obtienen con las mejores variaciones de los algoritmos encontradas según lo expuesto en la sección 5.1 y con tiempo máximo de ejecución de 2 horas de simulación para cada una de las 5 pruebas realizadas a cada uno de los 3 algoritmos.

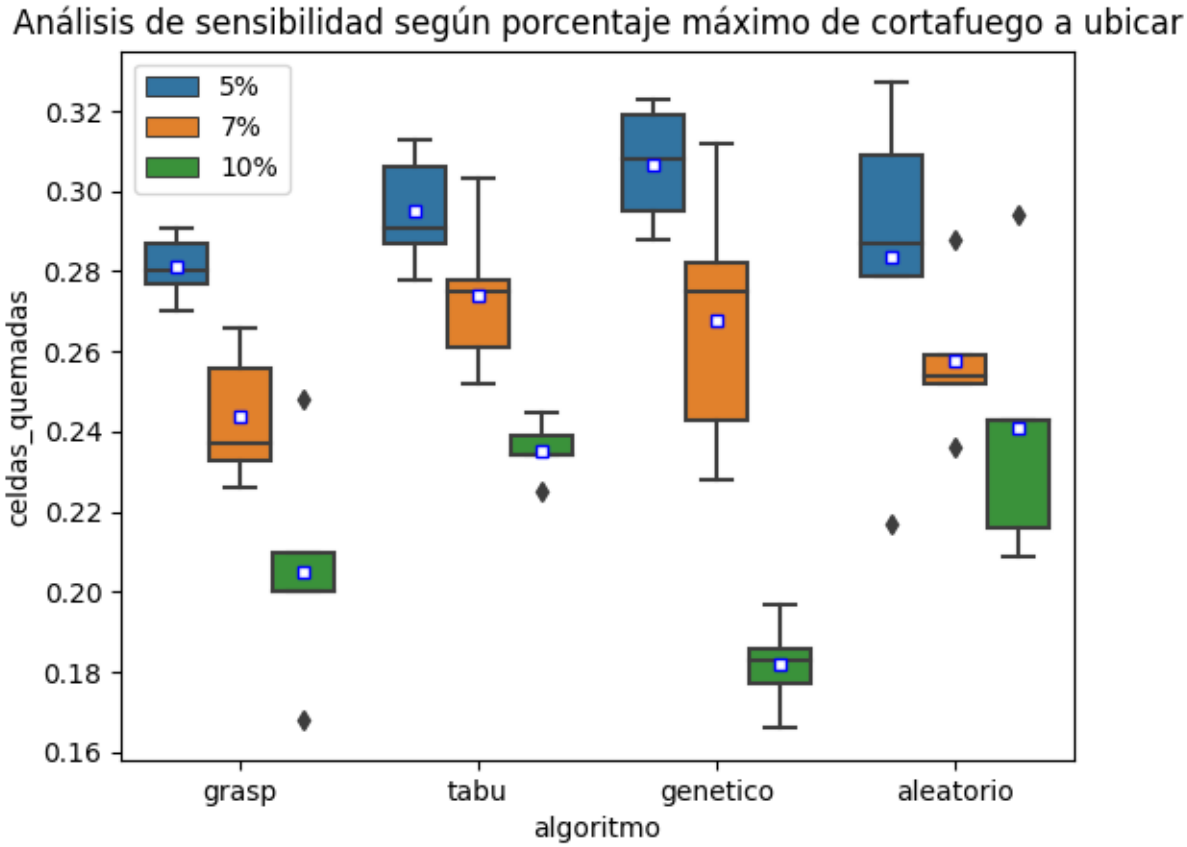



Figura 5.17: Desempeño para distintos porcentajes de máximo cortafuego a ubicar.

En la figura se logra ver que todas las metaheurísticas escalan positivamente a medida que aumenta la cantidad de cortafuego a utilizar, con el Algoritmo Genético teniendo un resultado destacable al tener un 10% de la superficie total ded bosque como cortafuego. Por otro lado, si bien la Búsqueda Tabú aumenta el porcentaje de bosque que salva a medida que aumenta el porcentaje de cortafuego permitido, en promedio no logra superar a la solución aleatoria en los casos que se ocupa un 5% y un 7% de cortafuegos, mientras que al tener 10% la supera pero sólo levemente (aproximadamente quema un 0.6% menos de bosque que la solución aleatoria). GRASP logra superar a todas las otras metodologías en los escenarios con 5% y 7% de superficie total de bosque como cortafuego, no así al utilizar un 10%, en cuyo caso el Algoritmo Genético logra superarlo. Además, cabe recordar de la figura 5.10 que el porcentaje promedio de superficie quemada al dejar el bosque sin cortafuegos es de aproximadamente 33%. Así, al utilizar un 5% de superficie total de bosque como cortafuego, Algoritmo Genético, GRASP y Búsqueda Tabú logran superar en promedio al escenario libre en aproximadamente 2.3%, 4.9% y 3.5% respectivamente. Al usar un 7% de la superficie total de bosque en dichas metaheurísticas, se logra mejorar el escenario libre en promedio en 6.2%, 8.6% y 5.6%. Mientras tanto, al ocupar un 10% de la superficie total del bosque como cortafuego, las metaheurísticas salvan en promedio 14.8%, 12.5 y 9.5% más superficie que el caso libre.

## 5.4 Análisis de sensibilidad en tiempo de ejecución

En las siguientes pruebas se aumenta el tiempo de ejecución a 6 horas. De acuerdo al desempeño de las pruebas anteriores, existía una alta probabilidad de que las metaheurísticas no mejoraran mucho debido a la imprecisión del valor de la función objetivo a causa del número de simulaciones de Monte Carlo. Por esto, se modifica también el número de simulaciones de Monte Carlo que realiza cada algoritmo: Algoritmo Genético pasa desde las 20 simulaciones que realizaba en pruebas de 2 horas a realizar ahora 30 simulaciones a cada individuo. La Búsqueda Tabú realiza ahora 20 simulaciones a cada vecino versus las 10 que hacía anteriormente, y GRASP verifica el óptimo local con 30 simulaciones y no con 20 como en su versión de 2 horas. Los experimentos se realizaron en el bosque **a)** con vientos en las 8 direcciones de la rosa de los vientos y con un punto de ignición en cuadrado, para así presentar un escenario que facilita la convergencia a lo largo del tiempo. El resto de los parámetros se mantiene igual que en las pruebas anteriores.

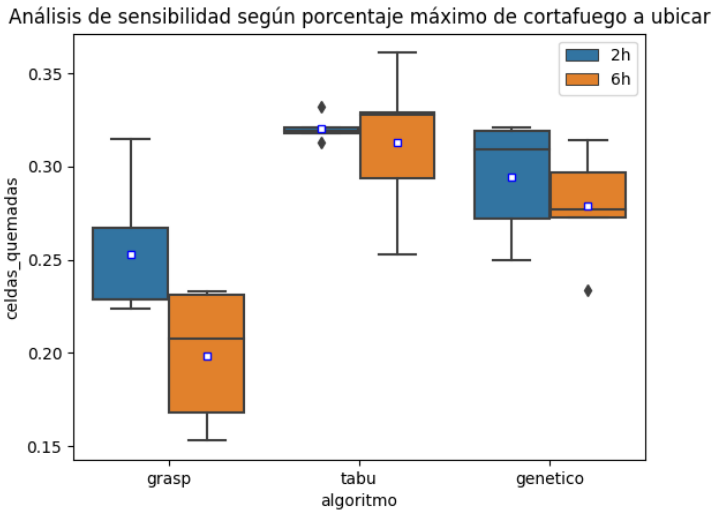



Figura 5.18: Desempeño según tiempo máximo de ejecución.

Es posible apreciar que las metaheurísticas en pruebas de 6 horas tienden a mejorar al ser comparadas con los experimentos de 2 horas, con GRASP el algoritmo que presenta una mejor escalabilidad, pasando de un 25.2% de bosque quemado en las pruebas de 2 horas, a un 20% en las pruebas de 6 horas. Por otro lado, tanto Búsqueda Tabú como el Algoritmo Genético si bien mejoran en su desempeño promedio a lo largo de las 5 instancias, dicha mejora no es significativa, además el error en el caso de la Búsqueda Tabú es bastante superior en las pruebas de 6 horas.

# 6 Discusión

A partir de lo expuesto en la sección anterior, es posible notar que las metaheurísticas tienen buenos desempeños en algunos escenarios, y un mal desempeño en otros escenarios.

En primer lugar, se debe analizar el resultado obtenido en la sección 5.1, es decir, las mejores variaciones de los algoritmos. La hipótesis consistía en que la mejor variación se construía en base a tener soluciones tipo *cluster* con sesgo en los movimientos, moviéndose hacia zonas que más frecuentemente se incendian. La primera hipótesis se cumple para todas las metaheurísticas y la segunda se cumple en el Algoritmo Genético y la Búsqueda Tabú, es decir, las soluciones que consisten en nodos agrupados son más efectivas que soluciones con nodos independientes (como ya había sido expuesto en [Finney et al, 2008]), y realizar movimientos no totalmente aleatorios y hacia zonas que más se han visto afectadas por incendios permite mejores resultados en el caso del Algoritmo Genético y la Búsqueda Tabú. Por otro lado, la hipótesis de otorgar una buena solución inicial a Búsqueda Tabú y al Algoritmo Genético no se cumple (como ya se mencionó, GRASP no utiliza solución inicial). Una posible explicación a esto viene dado por lo expuesto en [Talbi, 2009], en donde se señala que el otorgar una buena solución inicial puede provocar que el algoritmo no sea capaz de encontrar mejores soluciones y así se quede estancado en dicha solución. Finalmente, el sesgo de movimiento no es efectivo en la búsqueda local en GRASP. La causa probable es debido a que GRASP ya utiliza información de las zonas más quemadas en su fase de construcción, por ende el utilizar el mismo criterio para buscar estados vecinos sólo logra repetir información anterior y no se permite diversificar en demasía las configuraciones de la metaheurística. Cabe destacar que esto último no implica que el sesgo de movimiento no sirva en GRASP, sino que señala que el sobre utilizarlo en 2 etapas distintas de la metaheurística no es efectivo y basta emplearlo para armar las soluciones en la fase de construcción.

Los resultados de la sección 5.2 muestran un desempeño mejor en situaciones de baja y media variabilidad en bosques grandes, pero un mal resultado en escenarios de variabilidad alta y en los bosques pequeños. Los 2 peores resultados en bosques grandes corresponden a un incendio con punto de ignición aleatorio y otro en donde además de tener punto de ignición aleatorio el viento es libre de moverse en cualquier dirección a la redonda. Esto demuestra que las metaheurísticas tienen una convergencia más lenta en escenarios aleatorios, y se hace necesario elevar el número de simulaciones de Monte Carlo para buscar soluciones más robustas (sección 5.4) o aumentar el porcentaje de cortafuegos para cubrir más superficie de bosque (sección 5.3). Por otro lado, la diferencia entre los casos 5.9 y 5.12 evidencian una mayor sensibilidad de las metaheurísticas frente al punto de ignición que frente a una variación en la dirección del viento, ya que la primera prueba tiene mejores resultados que la segunda, y en la primera se simula al bosque **a)** con vientos totalmente aleatorios pero punto de ignición con baja variabilidad, mientras que en la segunda se simula el mismo bosque **a)** pero con vientos menos aleatorios (apertura en $30^o$) y punto de ignición totalmente aleatorio. Este mismo resultado se puede obtener al comparar los casos de las figuras 5.9 versus 5.11 y los casos de las figuras 5.13 versus 5.14. En la primera comparación se logran apreciar resultados similares en las metaheurísticas a pesar de que en un caso se utilizan vientos totalmente aleatorios y en el otro vientos en apertura de $30^o$. Un resultado similar se obtiene en la segunda comparación. En este bosque homogéneo las metaheurísticas logran resultados similares en ambos escenarios (a excepción del Algoritmo Genético pero posee un amplio margen de error). Estos resultados ayudan a concluir lo mismo que ya se señaló: las metaheurísticas tienden a ser más sensibles en su desempeño cuando el punto de ignición es aleatorio a cuando el viento posee mayor variabilidad.

Tanto el bosque heterogéneo como el homogéneo de dimensión 40x40 celdas no logran resul-

tados satisfactorios (especialmente el bosque **b)**), probablemente por la poca superficie cubierta con cortafuegos debido a la geometría de los *clusters* como se puede ver en una de las soluciones óptimas, expuesta en la figura 6.1. Una posible solución a esto es modificar la estructura de los *clusters* en bosques pequeños, manteniendo la forma pero disminuyendo los nodos que utiliza (por ejemplo utilizar 3 nodos verticales a cada lado en vez de 5, y utilizar 8 nodos horizontales en vez de 12).

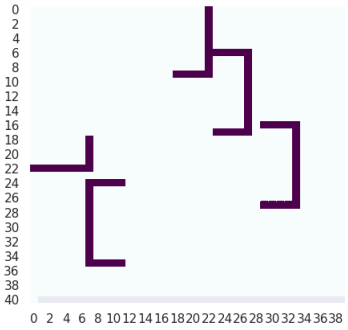
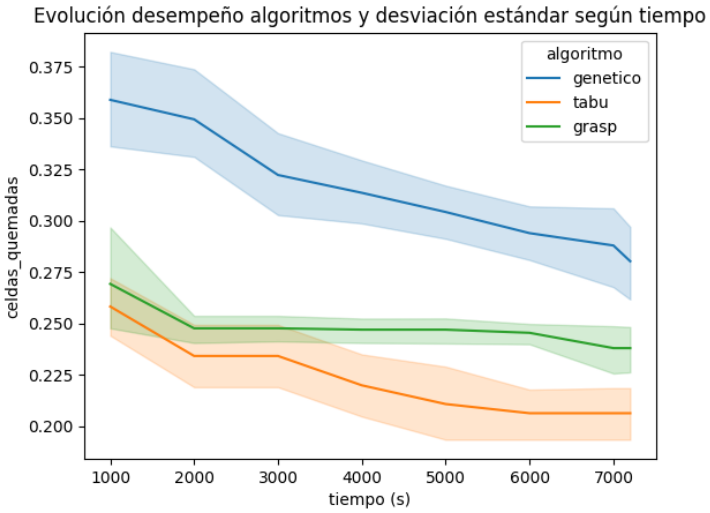
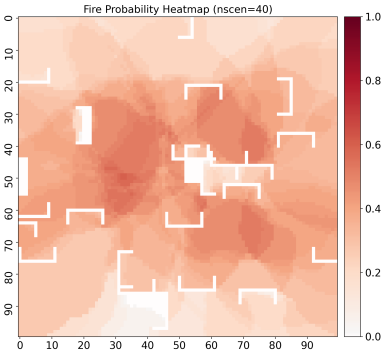



Figura 6.1: Disposición óptima para prueba nº2 en Algoritmo Genético en bosque **b)**

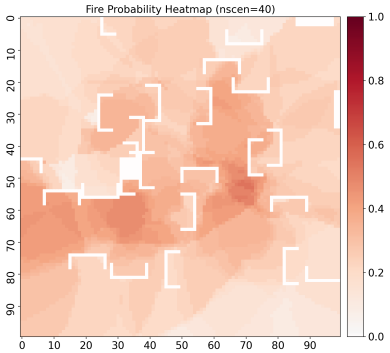
En cuanto a la comparación de las metaheurísticas entre sí, es posible ver que GRASP tiene un desempeño más robusto a lo largo de las variaciones (a excepción del escenario 5.16 en el cual levemente es superada por el Algoritmo Genético pero con un amplio margen de error), lo que demuestra que el enfoque *greedy* permite construir soluciones robustas al enfocarse en cubrir zonas que se han visto más afectadas por los incendios a lo largo de las simulaciones. Por el contrario, Búsqueda Tabú no es capaz de lograr buenos resultados, probablemente debido al bajo número de simulaciones de Monte Carlo lo que no le permite obtener resultados lo suficientemente confiables. Esto se evidencia en la figura 8.1 que representa la evolución promedio y la desviación estándar del desempeño a lo largo del tiempo para cada una de las 5 pruebas realizadas a cada algoritmo en el escenario 6.2a. Esta figura muestra que Búsqueda Tabú converge dentro de su ejecución a soluciones mejores que las de GRASP y bastante superior a las del Algoritmo Genético, pero dichas soluciones no representan de manera fidedigna su valor, ya que al medir su desempeño en un número superior de pruebas no se obtiene el mismo valor o alguno similar (figura 6.2b), a diferencia del Algoritmo Genético (figura 6.2c). Dentro de la ejecución del algoritmo se intenta lidiar con esto realizando un número no demasiado bajo de simulaciones de Monte Carlo (se realizaron 10 simulaciones a cada configuración en las pruebas de 2 horas) pero evidentemente este número no fue suficiente, y el aumentar en demasía dicho valor eleva considerablemente el tiempo de cada iteración.




(a) Evolución en el tiempo desempeño de algoritmos



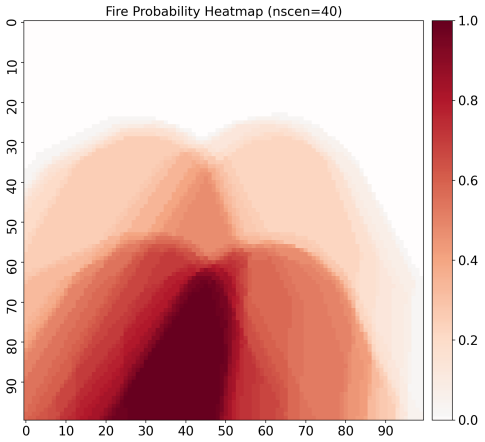

(b) Mapa de calor solución óptima tabú con desempeño de 31% y desempeño en ejecución de 18.68%



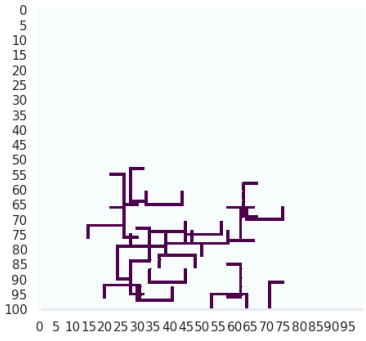

(c) Mapa de calor solución óptima Algoritmo Genético con desempeño de 25% y desempeño en ejecución de 26%

Figura 6.2: Evolución en el tiempo y comparación algoritmos para prueba nº1 en bosque **a)** con 8 vientos aleatorios y punto de ignición en cuadrado

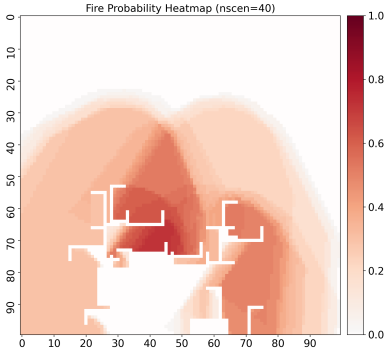
Otro resultado interesante es la topología de las soluciones de GRASP. De acuerdo al ya citado trabajo de [Finney et al, 2008], la forma más efectiva de ubicar cortafuegos es agruparlos más que ubicarlos independientemente en diferentes zonas del bosque. El algoritmo que mejor logró esto es GRASP y esa es la principal explicación de su mayor desempeño. En la figura 6.3 se puede observar tanto la solución óptima de la prueba nº4 en el bosque **c)** con vientos en apertura de 30º y punto de ignición en cuadrado, así como también el mapa de calor de dicha solución óptima y de la solución sin cortafuegos. Estos mapas de calor son generados en base a las simulaciones realizadas por Cell2Fire y representan la probabilidad de quema de cada zona del bosque, en donde zonas más rojas representan áreas que se queman con mayor frecuencia a lo largo de las simulaciones.




(a) Comportamiento incendio libre



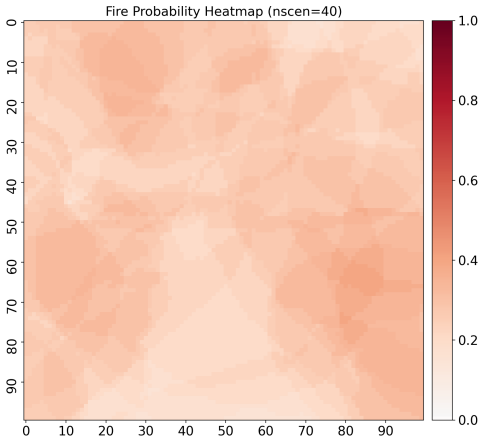

(b) Topología solución Óptima



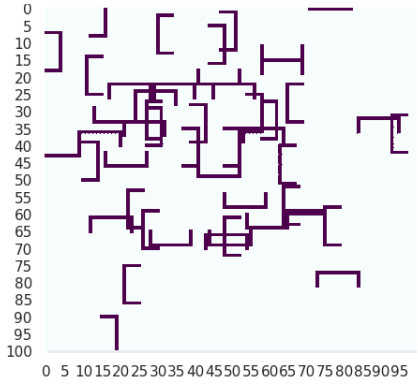

(c) Mapa de calor solución óptima, con desempeño de 15.9%

Figura 6.3: Resultados GRASP en prueba º4 en bosque **c)** con vientos en apertura de 30º y punto de ignición en cuadrado

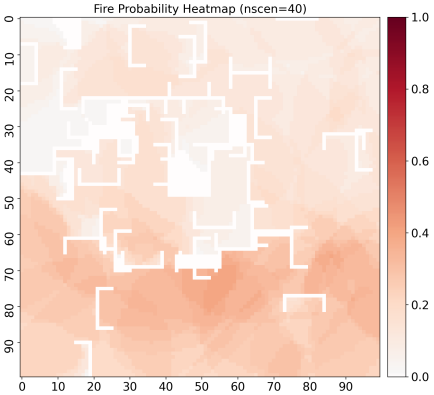
Un resultado similar puede verse en la prueba nº3 del escenario con 10% de superficie total como cortafuegos, esto en el caso del bosque **a)** con vientos en las 8 direcciones a la redonda y punto de ignición aleatorio, escenario en el cual no existe una tendencia clara en el comportamiento del incendio como se puede ver en la figura 6.4a, a diferencia del comportamiento de la figura en la cual GRASP tiende a ubicar cortafuegos en la zona de expansión del incendio. En el caso de la topología de la figura 6.4b, a pesar de ser un escenario con alta aleatoriedad, GRASP busca igualmente agrupar figuras más que situarlas dispersamente por el bosque.




(a) Comportamiento incendio libre



(b) Topología solución Óptima




(c) Mapa de calor solución óptima, con desempeño de 16.8%

Figura 6.4: Resultados GRASP en prueba nº4 en bosque **a)** con vientos en 8 direcciones y punto de ignición aleatorio con 10% de cortafuego

En cuanto a la escalabilidad de los metaheurísticas, todas las metaheurísticas logran mejorar a medida que se aumenta el porcentaje de cortafuegos y el tiempo de ejecución. En el primer caso, el Algoritmo Genético logra un muy buen desempeño con un 10% de superficie total como cortafuego, pero GRASP sigue siendo más constante a lo largo de las variaciones, mientras que si bien Búsqueda Tabú logra mejorar, esta mejora no es suficiente para superar en demasía ni siquiera a la solución aleatoria. Estos resultados tienen sentido con lo esperado, ya que al poseer más cortafuegos, hay una mayor zona incombustible del bosque, por lo que el porcentaje de celdas quemadas debería bajar. Cabe señalar que un resultado medianamente aceptable es que las metaheurísticas utilicen los cortafuegos para al menos tapar las zonas que se incendian y que no cubran zonas que nunca se ven afectadas. Así, al utilizar un 5%, 7% o 10% de superficie total de bosque como cortafuego, se espera que el incendio mejore al menos en esos mismos porcentajes con respecto al configuración sin cortafuegos. Si se analiza el escenario en que se utiliza el bosque **a)** con 8 vientos a la redonda y punto de ignición aleatorio, en promedio se quema un 33% de la superficie del bosque del bosque cuando no se sitúan cortafuegos. Siguiendo la idea recién expuesta, en este mismo escenario al utilizar 5% de cortafuegos, ninguna metaheurística logra superar este umbral, siendo GRASP la más cercana al fallar en un 0.1%. Al 7% sólo GRASP alcanza un desempeño mayor a este umbral, con una mejora de un 8.6%, por lo que se podría decir que los cortafuegos "cubren" un 7% de la

superficie quemada y adicionalmente disminuyen el incendio en un 1.6% de superficie[17]. Finalmente, utilizando un 10%, tanto GRASP como el Algoritmo Genético superan el umbral, con este último aportando un adicional de 4.8% en la disminución del impacto del incendio. La tabla 6.1 resume esta información:

| % máximo de cortafuego | GA | GRASP | TABU |
|---|---|---|---|
| 5% | 2.3% | 4.9% | 3.5% |
| 7% | 6.2% | 8.6% | 5.6% |
| 10% | 14.8% | 12.5% | 9.5% |

Tabla 6.1: Mejora Algoritmos con respecto a escenario libre

Al analizar la escalabilidad en términos temporales, se puede notar una mejora notable en el desempeño de GRASP y una leve mejora tanto en la Búsqueda Tabú como en el Algoritmo Genético, esto tiene 2 razones: el aumento de las simulaciones de Monte Carlo y el aumento del tiempo de ejecución. Al aumentar el número de simulaciones de Monte Carlo, los resultados obtenidos en cada iteración son más robustos y permiten confiar de mejor manera en ellos. De esta forma, tanto en GRASP como en la Búsqueda Tabú se tiene un valor con un error más bajo del desempeño real del óptimo global, con el que se compara el óptimo local de cada iteración. Así, la evolución temporal del desempeño del algoritmo expuesto en la figura 6.5 es mucho más realista que en las pruebas de 2 horas en las que se utiliza un menor número de simulaciones de Monte Carlo. El Algoritmo Genético es mucho más sensible en cuanto al número de simulaciones de Monte Carlo realizadas ya que toda la población depende de la confianza que da el desempeño de los padres, es por esto que ya en las pruebas de 2 horas es necesario tener un número mayor de simulaciones que en las otras metaheurísticas.

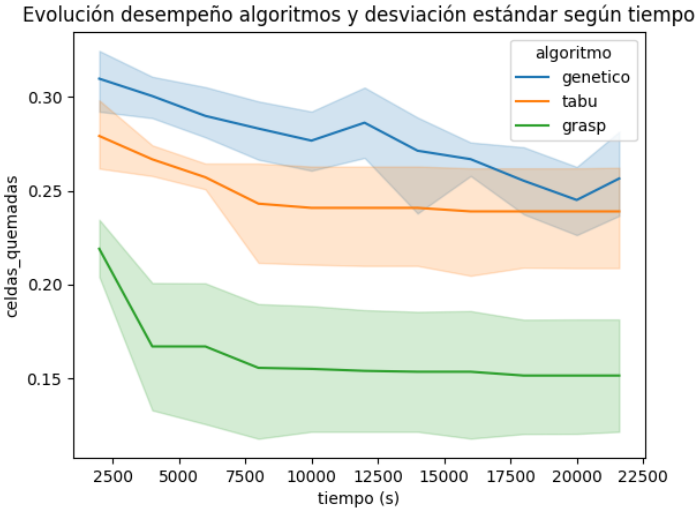



Figura 6.5: Evolución del óptimo global según el tiempo. En el caso del Algoritmo Genético se calcula el mejor individuo de la generación.

[17]Esta frase textualmente no necesariamente es correcta, ya que tal vez se utiliza parte de este porcentaje en zonas que no se ven afectadas por el incendio pero que sí permiten disminuir el impacto del evento. Por lo que en realidad podría cubrirse por ejemplo sólo un 3% de bosque quemado y el resto del porcentaje en una zona que nunca se ha incendiado pero que permite disminuir la expansión del fuego

De la mano de un aumento en el número de simulaciones de Monte Carlo debe aumentarse también el tiempo de ejecución, para así realizar un número suficiente de iteraciones que permitan otorgar convergencia. Al mismo tiempo, este aumento del tiempo permite explorar más soluciones, explotando de esta manera una mayor zona del espacio de soluciones otorgando una mayor probabilidad de encontrar mejores soluciones.

Si bien los algoritmos escalaron positivamente al aumentar el tiempo de ejecución, en la figura 6.5 puede observarse que bastaba aumentar dicho tiempo a aproximadamente 8.000 segundos ($\sim$ 2 horas y 15 minutos) y no a 21.600 segundos (6 horas). Esto se justifica en el hecho de que a partir del momento señalado el desempeño de los algoritmos no aumenta en demasía y se mantiene más o menos constante, a excepción del Algoritmo Genético, cuyo valor varía debido a que se basa en un cálculo realizado en cada generación, mientras que para Búsqueda Tabú y GRASP se observa la evolución del óptimo global que se calcula sólo en el momento en que se encuentra esta solución. Por lo tanto, la única metaheurística que se vio beneficiada por el aumento del tiempo de ejecución a 6 horas es quizás el Algoritmo Genético, ya que para las otras 2 técnicas bastaba aumentar dicho tiempo en tan solo unos 15 minutos.

Finalmente, cabe señalar que como se indicó anteriormente en esta investigación se simplificaron diversas variables. Se propone como una extensión a este trabajo el incluir algunos (no exclusivamente) costos, entre ellos pueden considerarse:

- **Costos Comerciales**: Se pueden considerar costos por pérdidas asociadas a la madera que se ve afectada por el incendio en el caso de plantaciones privadas destinadas a la venta de la misma.
- **Costos Medioambientales**: Al haber un incendio, existe un porcentaje importante de $CO_2$ que se libera al medioambiente que implica mayor contaminación y efectos perjudiciales a la salud de la flora y fauna cercana a los bosques. En cambio, al cortar la madera cuando se generan los cortafuegos, este $CO_2$ no es liberado al entorno ya que el material es utilizado con otros fines. Existe también un impacto a la misma flora y fauna al modificar el paisaje del bosque al ubicar un cortafuego, por lo que dicho impacto igualmente debiese ser considerado como un costo.
- **Costos Operativos**: Puede incluirse también un costo asociado a la gestión operativa de ubicar un cortafuego en el bosque, ya que al ubicar más y más cortafuegos, el costo de realizar la gestión se eleva.

Para incluir estos costos se propone una modificación de la función 3.1 (función objetivo del problema) a una de la forma:

$$\begin{aligned} \min \quad & Costos(c) \\ \text{subject to} \quad & |c| \leq P, \\ & Costos(c) = C_{Comercial}(c) + C_{Medioambiental}(c) + C_{Operativo}(c) \end{aligned}$$

Con,

$$C_{Comercial}(c) = \sum_{i}^{N} \mathbb{E}(IP(quema_i|c)) \cdot ValorComercial_i$$

, donde $N$ es el total de celdas del bosque, $\mathbb{E}(IP(quema_i|c))$ es el valor esperado de la probabilidad de quema de la celda $i$ sujeto a la configuración de cortafuegos $c$ y $ValorComercial_i$ corresponde

al valor comercial de la madera ubicada en la celda $i$. Del mismo modo,

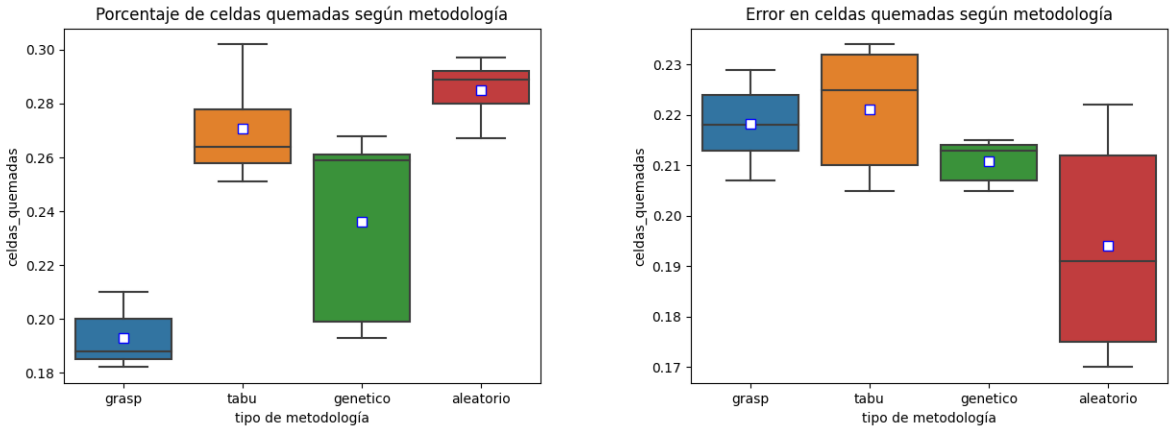
$$C_{Medioambiental}(c) = \sum_{i}^{N} \mathbb{E}(IP(quema_i|c)) \cdot ValorMedioambiental_i$$

, con $N$ el total de celdas del bosque, $\mathbb{E}(IP(quema_i|c))$ es el valor esperado de la probabilidad de quema de la celda $i$ sujeto a la configuración de cortafuegos $c$ y $ValorMedioambiental_i$ corresponde a la evaluación ambiental considerando la flora, fauna y biomasa presente en la celda $i$. Y,

$$C_{Operativo}(c) = \sum_{i}^{N} \mathbb{1}_c(i) \cdot ValorOperativo_i$$

, donde $N$ es el total de celdas del bosque, $\mathbb{1}_c()$ corresponde a la función indicatriz de la configuración de cortafuegos $c$ y $ValorOperativo_i$ corresponde al costo operativo por ubicar un cortafuego en la celda $i$.

Otra posible extensión propuesta proviene del resultado de la figura 6.6.




(a) Promedio desempeño óptimos según metodología (b) Errores valores óptimos según metodología

Figura 6.6: Desempeños y errores de soluciones óptimas en bosque **a)** con vientos en apertura de 30º y punto de ignición aleatorio

El gráfico 6.6b se construye en base a los errores que comete cada solución óptima de cada metaheurística en este caso en el bosque **a)** con vientos en apertura de 30º y punto de ignición aleatorio. Se puede observar en base al promedio de los errores que en general la solución aleatoria se equivoca menos en el porcentaje de celdas que se queman, mientras que las soluciones de la Búsqueda Tabú tienen un error mayor. Destaca que si bien GRASP es la solución que mejor desempeño otorga, en algunas de las simulaciones de Monte Carlo sus mejores soluciones tienen un desempeño peor que la solución aleatoria. Además, se puede observar que a pesar de que la desviación estándar de los desempeños de las soluciones es alta en el Algoritmo Genético, el error que comete cada una de estas soluciones es bastante similar entre ellas. Pero como ya se mencionó, las soluciones de las metaheurísticas pueden llegar a equivocarse más que las soluciones de la solución aleatoria. Con esto, se puede concluir que tanto el mejor como el peor caso de las metaheurísticas tienen valores más extremos que la solución aleatoria. Esto puede ser bueno al existir un error positivo (se queman menos celdas que el promedio), pero podría suceder que efectivamente ocurra

el peor de los casos y el bosque termine quemándose aún más que en la solución aleatoria. Tomando en consideración esto, se propone modificar la función objetivo para no optimizarla en función del promedio de celdas quemadas a lo largo de las simulaciones de Monte Carlo, sino que busque que el peor escenario de estas simulaciones sea el mejor posible. Para esto, la fórmula 3.1 se modifica a:

$$\begin{aligned} &\min && max(BosqueQuemado_i(c)) \\ &\text{subject to} && |c| \leq P, \end{aligned}$$

, con

$$BosqueQuemado_i(c) = \frac{CeldasQuemadas_i(c)}{CeldasTotales}, \quad i \in \{1, 2, 3...., S\}$$

, con $S$ el total de simulaciones en Cell2Fire.

Una de las limitaciones de este enfoque es que efectivamente se minimice el peor de los casos pero en general se obtengan soluciones que no permitan salvar un porcentaje importante del bosque. Debido a esto se proponen también otras extensiones utilizando distintas métricas en el cálculo de la función objetivo como minimizar el error cuadrado a lo largo de las simulaciones, minimizar el valor absoluto del error, optimizar tanto el error como el promedio de celdas quemadas, entre otras.

# 7 Conclusiones

El principal objetivo de este trabajo consistía en lograr comparar una serie de metaheurísticas entre sí y frente a soluciones aleatorias para así determinar si dichos algoritmos eran capaces de asistir al combate de incendios forestales mediante la disminución de la porción de bosque que se quema en estos eventos comparado a una solución aleatoriaa. Considerando los resultados obtenidos se concluye que en determinados escenarios en los que no existe una amplia aleatoriedad en las variables del problema, la versión de las metaheurísticas desarrolladas en este trabajo permiten verificar dicho objetivo, logrando así soluciones que efectivamente cuidan al bosque mediante el salvataje de una porción de bosque que se quemaría en el caso en que este no se protegiera con ningún cortafuego.

En este trabajo se propusieron modificaciones a las metaheurísticas presentadas, por ejemplo realizando movimientos con una cierta distribución de probabilidad dada por los escenarios anteriores, logrando así comprobar que es posible mejorar estos algoritmos al añadirles enfoques más apropiados de acuerdo al problema presentado, demostrando así la flexibilidad de las metaheurísticas al resolver problemas de optimización complejos.

Además, fue posible comprobar algunas de las ideas propuestas en trabajos anteriores, como el hecho de que cortafuegos agrupados otorgan un mejor desempeño al proteger un bosque enfrentado a un incendio forestal, o el hecho de que otorgar una solución inicial sobre-ajustada a una metaheurística puede llegar a impedir su correcta convergencia.

Observando el desempeño y el comportamiento de las soluciones alcanzadas por GRASP y el Algoritmo Genético, se propone el uso de estas metaheurísticas en escenarios geográficos como Chile, en donde ciertos incendios de peligro se generan de acuerdo a determinado comportamiento y no de forma totalmente aleatoria (como en Canadá por ejemplo [Stocks et al, 2002]). De acuerdo a la investigación de [Úbeda y Sarricolea, 2016], los incendios más riesgosos en Chile para la población son aquellos cuyo punto de ignición está en la frontera entre el bosque y la ciudad, por lo que las metaheurísticas propuestas podrían ayudar en disminuir el impacto generado por estos incendios.

Debido al funcionamiento de la interfaz de Cell2Fire y su naturaleza de código abierto, es posible incorporar al simulador alguna de las metaheurísticas aquí expuestas para así asistir al usuario que simula un incendio sugiriéndole estrategias de manejo de combustible para disminuir el impacto de dicho incendio, característica que actualmente el software no posee.

De existir una mayor capacidad computacional u operacional para lograr un mayor número de iteraciones o utilizar una mayor porción de bosque como cortafuego respectivamente, es posible obtener mejores resultados que los aquí expuestos. Del mismo modo, se recomienda preferir el Algoritmo Genético si se tuviese disponible una mayor cantidad de recursos para ubicar más cortafuegos, o escoger GRASP si se tuviese una mayor capacidad computacional para realizar pruebas más largas. Si se contase con recursos reducidos, se recomiendo optar por GRASP que permite buenos resultados a un costo relativamente bajo.

Cabe señalar que una limitación importante de esta investigación y que resulta crucial en el estudio de metaheurísticas es el número de iteraciones realizadas. Como se pudo observar en Búsqueda Tabú, se debería aumentar el número de simulaciones de Monte Carlo para lograr mejores resultados, pero esto conlleva un costo computacional alto. El realizar 10 simulaciones (lo que es equivalente a revisar a un vecino en Búsqueda Tabú o GRASP, o revisar un individuo en el Algoritmo Genético) en Cell2Fire en un bosque de 100x100 celdas para un incendio de 50 horas toma aproximadamente 30 segundos, por lo que elevar este número implica un aumento considerable en el tiempo de ejecución

y por ende un menor número de iteraciones, elemento de suma importancia en las metaheurísticas, que buscan realizar el mayor número posible de iteraciones en poco tiempo. Por lo tanto, el haber encontrado soluciones robustas en pruebas de 2 horas con un número tan bajo de iteraciones (aproximadamente 80 iteraciones factibles en el mejor caso de GRASP) da un buen indicio para la posible escalabilidad del problema, lo que se evidencia en la mejora de todos los algoritmos al aumentar el tiempo de ejecución.

Es necesario considerar que en este trabajo se utilizó una función objetivo que simplifica los escenarios reales, por lo que la aplicación real debiese considerar muchas otras variables que en esta investigación no fueron cubiertas. Del mismo modo, se debe recordar que todos los resultados fueron obtenidos mediante el simulador Cell2Fire que si bien ha tenido un muy buen desempeño en términos comparativos con algunas cicatrices de incendios reales, no es una muestra exacta de la realidad, por lo que los resultados no tan solo cuentan con una distorsión de la realidad asociada al número de simulaciones de Monte Carlo, sino que también existe dicha distorsión al traspasar los resultados desde Cell2Fire a un bosque real.

De la mano con lo anterior, la función objetivo utilizada puede conllevar a que los peores escenarios de las soluciones no sean mucho mejores que una solución aleatoria, por lo que se propone extender dicha función mediante la inclusión de otras métricas. Del mismo modo, como se mencionó anteriormente, la función objetivo simplifica bastantes costos. Es por lo anterior que se propone el extender este trabajo incluyendo dichos costos. Debido a la naturaleza de los problemas de optimización, la inclusión de estos costos no es de complejidad en términos de modelación matemática, pero sí existe una dificultad que proviene de un mayor tiempo de cómputo para calcular la función objetivo en cada simulación y también en el modo en que se determinen los valores de dichos costos, especialmente costos medioambientales asociados a la calidad de vida de la flora y fauna que habita el ecosistema.

# 8 Bibliografía

[Aarts y Van Laarhoven, 1987] E. H. L. Aarts and R. J. M. Van Laarhoven. *Simulated Annealing: Theory and Applications*. Reidel, Dordrecht, 1987.

[Abbas, 2001] H. A. Abbas. *Marriage in honey bees optimization: A haplometrosis polygynous swarming approach*. In CEC'2001 Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 207–214.

[Abt et al, 2015] Abt Karen L., Butry David T., Prestemon Jeffrey P., Scranton Samuel (2015) *Effect of fire prevention programs on accidental and incendiary wildfires on tribal lands in the United States*. International Journal of Wildland Fire 24, 749-762. https://doi.org/10.1071/WF14168

[Acan, 2004] Acan A (2004) *An external memory implementation in ant colony optimization*. In: Dorigo M et al (eds) 4th international workshop on Ant colony optimization and swarm intelligence (ANTS 2004). Lecture notes in computer science, vol 3172. Springer, Heidelberg, pp 73–84

[Agee y Skinner, 2005] J. K. Agee and C. N. Skinner. *Basic principles of forest fuel reduction treatments*. Forest ecology and management, 211(1-2):83–96, 2005.

[Alexandridis et al, 1990] A. Alexandridis, D. Vakalis, C. I. Siettos, and G. V. Bafas. *A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through spetses island in 1990*. Applied Mathematics and Computation, 204(1):191–201, 2008a.

[Ali, Torn y Viitanen, 2002] M. M. Ali, A. Torn, and S. Viitanen. *A direct search variant of the simulated annealing algorithm for optimization involving continuous variables*. Computers and Operations Research, 29:87–102, 2002.

[Applegate et al, 2006] Applegate, D. L., Bixby, R. E., Chvatál, V., & Cook, W. J. (2006). *The Problem. In The Traveling Salesman Problem: A Computational Study* (pp. 1–58). Princeton University Press. http://www.jstor.org/stable/j.ctt7s8xg.4

[Arca et al, 2019] B. Arca, T. Ghisu, M. Casula, M. Salis, and P. Duce. *A web-based wildfire simulator for operational applications*. International Journal of Wildland Fire, 28(2):99–112, 2019.

[Aruga, 2005] Aruga, K. (2005). *Tabu search optimization of horizontal and vertical alignments of forest roads*. Journal of forest research, 10(4), 275-284.

[Bachinger et al, 2016] Laura M Bachinger, Leslie R Brown & Margaretha W van Rooyen (2016) *The effects of fire-breaks on plant diversity and species composition in the grasslands of the Loskop Dam Nature Reserve*, South Africa, African Journal of Range & Forage Science, 33:1, 21-32, DOI: 10.2989/10220119.2015.1088574

[Bailey y Zenteno Torres, 2015] Bailey, Christine & Zenteno Torres, Elizabeth. (2015). *Reflexiones en torno a la vulnerabilidad social y residencial de los asentamiento informales de los cerros de Valparaíso, Chile*. CIDADES, Comunidades e Territórios. 10.15847/citiescommunitiesterritories.dec2015.031.art08.

[Bandopadhyay y Sánchez, 2020] Bandopadhyay, S., & Sánchez, D. A. C. (2020). *Amazonian Fire Events Disturbed the Global Carbon Cycle: A Study from 2019 Amazon Wildfire Using Google Earth Engine*. Environmental Sciences Proceedings, 3(1), 43. MDPI AG. Retrieved from http://dx.doi.org/10.3390/IECF2020-08033

[Bar-Massada, 2011] Bar-Massada, Avi; Radeloff, Volker C.; Stewart, Susan I. 2011. *Allocating fuel breaks to optimally protect structures in the wildland-urban interface*. International Journal of Wildland Fire. 20: 59-68

[Barnett et al, 2016] K. Barnett, S. Parks, C. Miller, and H. Naughton. *Beyond fuel treatment effectiveness: Characterizing interactions between fire and treatments in the us*. Forests, 7(10):237, 2016.

[Barricelli, 1954] Barricelli, N.A. (1954). *Esempi numerici di processi di evoluzione*. Methodos: 45–68.

[Battiti, 2008] Battiti R. (2008) *Maximum Satisfiability Problem*. In: Floudas C., Pardalos P. (eds) Encyclopedia of Optimization. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-74759-0_364

[Bettinger et al, 2007] Bettinger, P., Boston, K., Kim, Y. H., & Zhu, J. (2007). *Landscape-level optimization using tabu search and stand density-related forest management prescriptions*. European Journal of Operational Research, 176(2), 1265-1282.

[Bianchi et al, 2009] Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2009). *A survey on metaheuristic for stochastic combinatorial optimization*. Natural Computing, 8:239 – 287.

[Birge y Louveaux, 1997] Birge, J. R., & Louveaux, F. (1997). *Introduction to stochastic programming*. New York: Springer.

[Bitam, Batouche y Talbi, 2008] S. Bitam, M. Batouche, and E.-G. Talbi. *A taxonomy of artificial honeybee colony optimization*. In E.-G. Talbi and K. Mellouli, editors, META'08 International Con- ference on Metaheuristics and Nature Inspired Computing, Hammamet, Tunisia, Oct 2008.

[Blum, Dan y Seddighin, 2020] Blum, Avrim & Dan, Chen & Seddighin, Saeed. (2020). *Learning Complexity of Simulated Annealing*.

[Bond et al, 2002] Monica L. Bond, R. J. Gutiérrez, Alan B. Franklin, William S. LaHaye, Christopher A. May and Mark E. Seamans. (2002). *Short-Term Effects of Wildfires on Spotted Owl Survival, Site Fidelity, Mate Fidelity, and Reproductive Success* Wildlife Society Bulletin (1973-2006) Vol. 30, No. 4 (Winter, 2002), pp. 1022-1028 (7 pages). https://www.jstor.org/stable/3784267

[Boyd y Vandenberghe, 2004 ] Boyd, Stephen P.; Vandenberghe, Lieven (2004). *Convex Optimization*. Cambridge University Press. p. 129. ISBN 978-0-521-83378-3.

[Brakeall, 2013] Brakeall, John, *Wildfire Assessment Using FARSITE Fire Modeling: A Case Study in the Chihuahua Desert of Mexico* (2013). FIU Electronic Theses and Dissertations. 923. https://digitalcommons.fiu.edu/etd/923

[Branke, Meise y Schmidt, 2008] Branke, J., Meisel, S. & Schmidt, C. *Simulated annealing in the presence of noise*. J Heuristics 14, 627–654 (2008). https://doi.org/10.1007/s10732-007-9058-7

[Brumelle et al, 1998] Brumelle, S., Granot, D., Halme, M., & Vertinsky, I. (1998). *A tabu search algorithm for finding good forest harvest schedules satisfying green-up constraints*. European Journal of Operational Research, 106(2-3), 408-424.

[Bull y Blumton, 1999] Bull, E.L. and Blumton, A.K. (1999). *Effect of fuels reduction on American martens and their prey*. Res. Note PNW-RN-539. Portland, OR: U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station. 9 pp.

[Bullnheimer, Hartl y Strauss, 1999] Bullnheimer B, Hartl RF, Strauss C (1999) *A new rank-based version of the ant system: a computational study*. Cent Eur J Oper Res Econ 7(1):25–38

[Butt et al, 2020] Butt, Edward & Conibear, Luke & Reddington, Carly & Darbyshire, Eoghan & Morgan, William & Coe, Hugh & Artaxo, Paulo & Brito, Joel & Knote, Christoph & Spracklen, Dominick. (2020). *Large air quality and human health impacts due to Amazon forest and vegetation fires*. Environmental Research Communications. 2. 10.1088/2515-7620/abb0db.

[Camaano-Isorna et al, 2011] Caamano-Isorna, F., Figueiras, A., Sastre, I. et al. *Respiratory and mental health effects of wildfires: an ecological study in Galician municipalities (north-west Spain)*. Environ Health 10, 48 (2011). https://doi.org/10.1186/1476-069X-10-48

[Caro et al, 2003] Caro, F., Constantino, M., Martins, I., & Weintraub, A. (2003). *A 2-opt tabu search procedure for the multiperiod forest harvesting problem with adjacency, greenup, old growth, and even flow constraints*. Forest Science, 49(5), 738-751.

[Cary et al, 2009] Cary Geoffrey J., Flannigan Mike D., Keane Robert E., Bradstock Ross A., Davies Ian D., Lenihan James M., Li Chao, Logan Kimberley A., Parsons Russell A. (2009) *Relative importance of fuel management, ignition management and weather for area burned: evidence from five landscape–fire–succession models*. International Journal of Wildland Fire 18, 147-156. https://doi.org/10.1071/WF07085

[Castillo, 1998] Castillo, M. (1997). *Método de Validación para el Simulador de Expansión de Incendios Forestales del Sistema KITRAL*. Memoria de Título. 123p. Facultad de Ciencias Forestales, Universidad de Chile.

[Castillo, 2017] Castillo, J. A. Á. (2017). *Percepción de la resiliencia de un destino turístico afectado por un incendio forestal. Caso destino Vichuquén.*

[Ceperley y Dewing, 1999] Ceperley, D.M., Dewing, M.: *The penalty method for random walks with uncertain energies*. J. Chem. Phys. 110(20), 9812–9820 (1999)

[Cerny, 1985] V. Cerny. *A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm*. Journal of Optimization Theory and Applications, 45:41–51, 1985

[Certini, 2005] Certini, G. (2005). *Effects of fire on properties of forest soils: a review*. Oecologia, 143: 1-10.

[Chan et al, 2020] Chan, Hau, Tran-Thanh, Long and Viswanathan, Vignesh (2020) *Fighting wildfires under uncertainty - a sequential resource allocation approach*. In: Twenty-Ninth International Joint Conference on Artificial Intelligence, Jan 2021. Published in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence pp. 4322-4329. ISBN 9780999241165. doi:10.24963/ijcai.2020/596

[Choromanska y DeLuca, 2001] Choromanska, U. and DeLuca, T.H. (2001). *Prescribed fire alters the impacts of wildfire on soil biochemical properties in a ponderosa pine forest*. Soil Science Society of America Journal, 65: 232-238.

[Chung et al, 2013] Chung W, Jones G, Krueger K, Bramel J, Contreras M. *Optimizing fuel treatments over time and space*. Int J Wildland Fire. 2013;22:1118–33.

[Colorni, Dorigo y Maniezzo, 1992] Colorni A, Dorigo M, Maniezzo V (1992) *Distributed optimization by ant colonies*. In: Varela FJ, Bourgine P (eds) Proceedings of the first European conference on artificial life. MIT Press, Cambridge, pp 134–142

[Cordón et al, 2000] Cordón O, de Viana IF, Herrera F, Moreno L (2000) *A new ACO model integrating evolutionary computation concepts: the best-worst ant system*. In: Dorigo M et al (eds) Abstract proceedings of ANTS 2000 – from ant colonies to artificial ants: second international workshop on ant algorithms. IRIDIA, Université Libre de Bruxelles, Belgium, pp 22–29

[Costa y Silver, 1998] Costa, D., Silver, E.A. *Tabu Search When Noise is Present: An Illustration in the Context of Cause and Effect Analysis*. Journal of Heuristics 4, 5–23 (1998). https://doi.org/10.1023/A:1009636520440

[Cressie, 1993] N. Cressie. *Statistics for Spatial Data*. Wiley, 1993.

[Curran et al, 2018] Curran, T.J.; Perry, G.L.W.; Wyse, S.V.; Alam, M.A. *Managing Fire and Biodiversity in the Wildland-Urban Interface: A Role for Green Firebreaks*. Fire 2018, 1, 3. https://doi.org/10.3390/fire1010003

[Curt et al, 2016] T. Curt, T. Fréjaville, and S. Lahaye. *Modelling the spatial patterns of ignition causes and fire regime features in southern france: implications for fire prevention policy*. International Journal of Wildland Fire, 25(7):785–796, 2016.

[Dantzig, 1963] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.

[Davis, 1958] Davis, Martin (1958). *Computability and Unsolvability*. New York: McGraw-Hill..

[De Sales et al, 2019] De Sales, F., Okin, G.S., Xue, Y. et al. *On the effects of wildfires on precipitation in Southern Africa*. Clim Dyn 52, 951–967 (2019). https://doi.org/10.1007/s00382-018-4174-7

[DeBano, 1990] DeBano, L.F. (1990). *The effect of forest fire on soil properties*. Symposium on management and productivity of Western-Montane forest soil. Boise, ID, USA, pp 151-156.

[Demange et al, 2021] Demange, M., Fonso, A.D., Stefano, G.D., & Vittorini, P. (2021). *A graph theoretical approach to the firebreak locating problem*. ArXiv, abs/2103.10115.

[Deneubourg et al, 1990] Deneubourg JL, Aron S, Goss S, Pasteels JM (1990) *The self-organizing exploratory pattern of the Argentine ant*. J Insect Behav 3(2):159–168

[Depietri y Orenstein, 2021] Yaella Depietri, Daniel E. Orenstein, *Chapter 1.1 - Bridging two cultures of fire risk at the wildland-urban interface: The case of Haifa, Israel*, Editor(s): Pedro Pinto Santos, Ksenia Chmutina, Jason Von Meding, Emmanuel Raju, Understanding Disaster Risk, Elsevier, 2021, Pages 19-36, https://doi.org/10.1016/B978-0-12-819047-0.00001-9.

[Dixon Y Turner, 1991] Dixon, R.K. and Turner, D.P. (1991). *The global carbon cycle and climate change-responses and feedbacks from below-ground systems*. Environmental Pollution, 73: 245-262.

[Díaz-Romero et al, 2013] M. A. Diaz-Romero, V. Alarcon-Aquino and J. A. Diaz-Garcia, *A Hybrid algorithm applied to facility location for forest fire fighting considering budget constraints*, 2013 10th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), 2013, pp. 262-267, doi: 10.1109/ICEEE.2013.6676055.

[Djikstra, 1959] Dijkstra, E.W. *A note on two problems in connexion with graphs*. Numer. Math. 1, 269–271 (1959). https://doi.org/10.1007/BF01386390

[Doerr et al, 2012] Benjamin Doerr, Ashish Hota, and Timo Kötzing. 2012. *Ants easily solve stochastic shortest path problems.* In Proceedings of the 14th annual conference on Genetic and evolutionary computation (GECCO '12). Association for Computing Machinery, New York, NY, USA, 17–24. DOI:https://doi.org/10.1145/2330163.2330167

[Dorigo, 1992] Dorigo M (1992) *Optimization, learning and natural algorithms.* PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy (in Italian)

[Dorigo, 2007] Dorigo M (2007) *Ant colony optimization.* Scholarpedia 2(3):1461

[Dorigo et al, 1991a] Dorigo M, Maniezzo V, Colorni A (1991) *The ant system: an autocatalytic optimizing process.* Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy

[Dorigo, Maniezzo y Colorni, 1991b] Dorigo M, Maniezzo V, Colorni A (1991) *Positive feedback as a search strategy.* Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy

[Dorigo et al, 1996] Dorigo M, Maniezzo V, Colorni A (1996) *Ant system: optimization by a colony of cooperating agents.* IEEE Trans Syst Man Cybern Part B 26(1):29–41

[Dorigo y Di Caro, 1999] Dorigo M, Di Caro GA (1999) *The ant colony optimization meta-heuristic.* In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw Hill, London, pp 11–32

[Dorigo y Gambardella, 1997] Dorigo M, Gambardella LM (1997) *Ant colony system: a cooperative learning approach to the traveling salesman problem.* IEEE Trans Evol Comput 1(1):53–66

[Dorigo y Stützle, 2004] Dorigo M, Stützle T (2004) *Ant colony optimization.* MIT Press, Cambridge

[Dupuy et al, 2020] Dupuy, Jl., Fargeon, H., Martin-StPaul, N. et al. *Climate change impact on future wildfire danger and activity in southern Europe: a review.* Annals of Forest Science 77, 35 (2020). https://doi.org/10.1007/s13595-020-00933-5

[D'Este et al, 2020] D'Este, M., Ganga, A., Elia, M. et al. *Modeling fire ignition probability and frequency using Hurdle models: a cross-regional study in Southern Europe.* Ecol Process 9, 54 (2020). https://doi.org/10.1186/s13717-020-00263-4

[Eberhart y Kennedy, 1995] Eberhart RC, Kennedy J (1995) *A new optimizer using particle swarm theory.* In: Proceedings sixth symposium on micro machine and human science, Piscataway, pp 39–43. IEEE Service Center

[El-Omari, 2021] El-Omari, Nidhal. (2021). *Sea Lion Optimization Algorithm for Solving the Maximum Flow Problem.*

[Elimbi Moudio et al, 2021] Elimbi Moudio, P., Pais, C. & Shen, ZJ.M. *Quantifying the impact of ecosystem services for landscape management under wildfire hazard.* Natural Hazards 106.1, 531–560 (2021). https://doi.org/10.1007/s11069-020-04474-y

[Feo y Resende, 1989] Feo T, Resende M. *Probabilistic algorithm (GRASP) for difficult set covering problems*

[Feo y Resende, 1995] Feo, T.A., Resende, M.G.C. *Greedy Randomized Adaptive Search Procedures.* J Glob Optim 6, 109–133 (1995). https://doi.org/10.1007/BF01096763

[Ferone et al, 2018] Ferone, Daniele & Gruler, Aljoscha & Festa, Paola & Juan, Angel. (2018). *Enhancing and extending the classical GRASP framework with biased randomisation and simulation*. Journal of the Operational Research Society. 70. 1-14. 10.1080/01605682.2018.1494527.

[Finlay et al, 2012] Finlay, S. E., Moffat, A., Gazzard, R., Baker, D., & Murray, V. (2012). *Health impacts of wildfires*. PLoS currents, 4, e4f959951cce2c. https://doi.org/10.1371/4f959951cce2c

[Finney, 1995] Finney, Mark A. 1995. *FARSITE: a fire area simulator for fire managers*. In: Weise, David R.; Martin, Robert E., technical coordinators. The Biswell symposium: fire issues and solutions in urban interface and wildland ecosystems; February 15-17, 1994; Walnut Creek, California. Gen. Tech. Rep. PSW-GTR-158. Albany, CA: Pacific Southwest Research Station, Forest Service, U.S. Department of Agriculture; p. 55-56

[Finney, 2001] Finney, M.A. 2001a. *Design of regular landscape fuel treatment patterns for modifying fire growth and behavior*. For. Sci. 47(2):219-228

[Finney, 2002] Finney, Mark A. (2002) *Fire growth using minimum travel time methods*. Canadian Journal of Forest Research 32, 1420–1424. doi:10.1139/ X02-068

[Finney, 2006] Finney, Mark A. 2006. *A computational method for optimizing fuel treatment locations*. In: Andrews, Patricia L.; Butler, Bret W., comps. 2006. Fuels Management-How to Measure Success: Conference Proceedings. 28-30 March 2006; Portland, OR. Proceedings RMRS-P-41. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. p. 107-123

[Finney et al, 2000] Agee, James K.; Bahro, Bernie; Finney, Mark A.; Omi, Philip N.; Sapsis, David B.; Skinner, Carl N.; van Wagtendonk, Jan W.; Weatherspoon, C. Phillip. 2000. *The use of shaded fuelbreaks in landscape fire managemen*. Forest Ecology and Management 127: 55-66

[Finney et al, 2008] Finney, Mark A.; Seli, Rob C.; McHugh, Charles W.; Ager, Alan A.; Bahro, Bernhard; Agee, James K. 2008. *Simulation of long-term landscape-level fuel treatment effects on large wildfires*. International Journal of Wildland Fire. 16: 712-727

[Fortnow, 2009] Fortnow, Lance (2009). *The status of the P versus NP problem*. Communications of the ACM. 52 (9): 78–86. CiteSeerX 10.1.1.156.767. doi:10.1145/1562164.1562186. S2CID 5969255.

[Garey y Johnson, 1979] Garey, Michael; Johnson, David (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freem

[Gendreau, 2002] M. Gendreau. *An introduction to tabu search*. In Handbook of Metaheuristics. Kluwer, 2002, pp. 37–54.

[Gendreau et al, 1994] Gendreau, Michel & Hertz, Alain & Laporte, Gilbert. (1994). *A Tabu Search Heuristic for the Vehicle Routing Problem*. Mgmt Sci. 40. 1276-1290. 10.1287/mnsc.40.10.1276.

[Glover, 1977] Glover, Fred. (1977). *Heuristics for integer programming using surrogate constraints*. Decision Sciences. 8. 156 - 166. 10.1111/j.1540-5915.1977.tb01074.x.

[Goldberg y Lingle, 1985] Goldberg D, Lingle R (1985) *Alleles, Loci and the traveling salesman problem*. In: Proceed- ings of the international conference on genetic algorithms, pp 154–159

[Goldreich, 2010] Goldreich, Oded (2010). *P, NP, and NP-Completeness*. Cambridge: Cambridge University Press. ISBN 978-0-521-12254-2

[González et al, 2020] González, M., Sapiains, R., Gómez-González, S., Garreaud, R., Mi- randa, A., Galleguillos, M., Jacques, M., Pauchard, A., Hoyos, J., Cordero, L., et al. (2020). *Incendios forestales en chile: Causas, impactos y resiliencia*. Center for Climate and Resilience Research: Santiago, Chile.

[Goss et al, 2020] Goss, M., D.L. Swain, J. Abatzoglou, A. Sarhadi, C. Kolden, A.P. Williams and N.S. Diffenbaugh. *Climate change is increasing the risk of extreme autumn wildfire conditions across California*, Environmental Research Letters, 15(9), 094016, 2020.

[Guo et al, 2016] Futao Guo, Zhangwen Su, Guangyu Wang, Long Sun, Fangfang Lin, Aiqin Liu, *Wildfire ignition in the forests of southeast China: Identifying drivers and spatial distribution to predict wildfire likelihood*, Applied Geography, Volume 66, 2016, Pages 12-21, ISSN 0143-6228, https://doi.org/10.1016/j.apgeog.2015.11.014.

[Heck, 2007] Heck, M. (2007). *Effects of wildfire on growth and demographics of coastal cutthroat trout in headwater streams*. Oregon State University.

[Hirsch et al, 2001] K. Hirsch, V. Kafka, C. Tymstra, R. McAlpine, B. Hawkes, H. Stegehuis, S. Quintilio, S. Gau- thier, and K. Peck. *Fire-smart forest management: A pragmatic approach to sustainable forest management in fire-dominated ecosystems*. The Forestry Chronicle, 77(2):357–363, 2001.

[Holland, 1975] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press. ISBN 978-0-262-08213-6.

[Huang, Romeo y Sangiovanni-Vincentelli, 1986] M. D. Huang, F. Romeo, and A. L. Sangiovanni-Vincentelli. *An efficient general cooling schedule for simulated annealing*. In IEEE International Conference on Computer-Aided Design, Santa Clara, CA, 1986, pp. 381–384.

[Hunt y Cooke, 1995] J. Hunt and D. Cooke. *An adaptive and distributed learning mechanism system based on the immune system*. In IEEE International Conference on Systems, Man and Cybernetics, 1995, pp. 2494–2499.

[Iman y Conover, 1982] Iman R, Conover W (1982) *A distribution-free approach to inducing rank correlation among input variables*. Commun Stat Simul Comput 11(3):311–334

[Ingber, 1996] L. Ingber. *Adaptive simulated annealing*. Control and Cybernetics, 25(1):33–54, 1996.

[Ivanauskas et al, 2003] Ivanauskas, N.M., Monteiro, R. and Rodrigues, R.R. (2003). *Alterations following a fire in a forest community of Alto Rio Xingu*. For. Ecol. Manage., 184: 239-250.

[Jhariya y Raj, 2014] Jhariya, M. K., & Raj, A. (2014). *Effects of wildfires on flora, fauna and physico-chemical properties of soil-An overview*. Journal of Applied and Natural Science, 6(2), 887-897. https://doi.org/10.31018/jans.v6i2.550

[Jhariya, 2014] Jhariya, M.K. (2014). *Effect of forest fire on microbial biomass, storage and sequestration of carbon in a tropical deciduous forest of Chhattisgarh*. Ph.D. Thesis, I.G.K.V., Raipur (C.G.), pp. 259.

[Johnson, 1992] Johnson, D.W. (1992). *Effects of forest management on soil carbon storage*. Water, Air, and Soil Pollution, 64: 83-120.

[Johnson et al, 1989] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. *Optimization by simulated annealing: An experimental evaluation. Part I. Graph partitioning*. Operations Research, 37:865–892, 1989.

[Juan et al, 2021] Juan, A.A., Keenan, P., Martí, R. et al. *A review of the role of heuristics in stochastic optimisation: from metaheuristics to learnheuristics*. Ann Oper Res (2021). https://doi.org/10.1007/s10479-021-04142-9

[Julio y Bosnich 2005] Julio G., Bosnich J. 2005. *Fundamentos del Manejo de Fuego*. Universidad Austral de Chile. Facultad de Ciencias Forestales. Valdivia. Chile. 285p.

[Karaboga, 2005] Karaboga, Dervis. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report - TR06*. Technical Report, Erciyes University.

[Kazimipour, Li y Qin, 2014] Kazimipour B, Li X, Qin A (2014) *A review of population initialization techniques for evolutionary algorithms*. In: Proceedings of the IEEE congress on evolutionary computation, pp 2585–2592

[Kennedy y Eberhart, 1995] Kennedy J, Eberhart RC (1995) *Particle swarm optimization*. In: Proceeding of the IEEE inter- national conference neural networks, Piscataway, vol IV. IEEE Service Center, pp 1942–1948

[Kirkpatrick, Gelatt y Vecchi, 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. *Optimization by simulated annealing*. Science, 220(4598):671–680, 1983.

[Klock y Grier, 1979] Klock, G. O.; Grier, C. C. 1979. *Effects of fire on the long-term maintenance of forest productivity*. In: Gessel, S. P.; Kenady, R. M.; Atkinson, W. A., eds. Proceedings, forest fertilization conference; 1979 September 25-27; Union, WA. Contr. 40. Seattle, WA: University of Washington: 247-250.

[Laguna y Martí, 2002] Laguna M, Martí R (2002) *Scatter search: methodology and implementations in C*. Kluwer Academic, Norwell

[Laroze, 1999] Andre Laroze, *A Linear Programming, Tabu Search Method for Solving Forest-Level Bucking Optimization Problems*, Forest Science, Volume 45, Issue 1, February 1999, Pages 108–116, https://doi.org/10.1093/forestscience/45.1.108

[Legües et al, 2007] Legües, Andres & Ferland, Jacques & Ribeiro, Celso & Vera, Jorge & Weintraub, Andres. (2007). *A tabu search approach for solving a difficult forest harvesting machine location problem*. European Journal of Operational Research. 179. 788-805. 10.1016/j.ejor.2005.03.071.

[Letey, 2001] Letey, J. (2001). *Causes and consequences of fire-induced soil water repellency*. Hydrological Processes, 15(15): 2867-2875.

[Li y Li, 2020] Li, G., & Li, J. (2020). *An Improved Tabu Search Algorithm for the Stochastic Vehicle Routing Problem With Soft Time Windows*. IEEE Access, 8, 158115-158124.

[Lin, Liu y Huang, 2021] Lin Shaorun, Liu Yanhui, Huang Xinyan (2021) *How to build a firebreak to stop smouldering peat fire: insights from a laboratory-scale study*. International Journal of Wildland Fire 30, 454-461. https://doi.org/10.1071/WF20155

[Liu et al, 2007] Liu, W., Xu, W., Han, Y., Wang, C. and Wan, S. (2007). *Responses of microbial biomass and respiration of soil to topography, burning, and nitrogen fertilization in a temperate steppe*. Biology and Fertility of Soil, 44: 259-268.

[Lizundia-Loiola et al, 2020] Lizundia-Loiola, J., Pettinari, M. L., & Chuvieco, E. (2020). *Temporal Anomalies in Burned Area Trends: Satellite Estimations of the Amazonian 2019 Fire Crisis*. Remote Sensing, 12(1), 151. MDPI AG. Retrieved from http://dx.doi.org/10.3390/rs12010151

[Lourenco, Martin y Stutzle, 2002] H. R. Lourenco, O. Martin, and T. Stutzle. *Iterated local search.* In Handbook of Meta-heuristics. Vol. 57 of Operations Research and Management Science. Kluwer Academic Publishers, 2002, pp. 321–353.

[Lyon et al, 2000] Lyon, L.J., Telfer, E.S. and Schreiner, D.S. (2000). *Direct effects of fire and animal responses.* In: Smith, J.K., ed. Wild land fire in ecosystems: effects of fire on fauna. Gen. Tech. Rep. RMRS-GTR-42-vol. 1. Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station; 17-23 pp.

[Mabuhay et al, 2003] Mabuhay, J.A., Nakagoshi, N. and Horikoshi, T. (2003). *Microbial biomass abundance after forest fire in pine forests in Japan.* Ecological Research, 18: 431-441

[Mak y Guo, 2004] Mak, K. and Guo, Z. (2004). *A genetic algorithm for vehicle routing problems with stochastic demand and soft time windows.* In Jones, M., Patek, S., and Tawney, B., editors, Proceedings of the 2004 IEEE systems and information symposium, pages 183–190. IEEE press.

[Maniezzo, 1999] Maniezzo V (1999) *Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem.* INFORMS J Comput 11(4):358–369

[Maniezzo y Carbonaro, 2000] Maniezzo V, Carbonaro A (2000) *An ANTS heuristic for the frequency assignment problem.* Futur Gener Comput Syst 16(8):927–935

[Martí, Pardalos y Resende, 2018] Marti, Rafael & Pardalos, P. & Resende, Mauricio. (2018). *Handbook of Heuristics.* 10.1007/978-3-319-07124-4.

[Martínez, 2021] Martínez M. (2021). *Diseño de patrones espaciales de cortafuegos para prevenir incendios forestales.* (Tesis de Magíster sin publicar). Departamento de Ingeniería Industrial, Universidad de Chile. Santiago.

[Mendoza et al, 2015] Mendoza, Jorge & Rousseau, Louis-Martin & Villegas, Juan. (2015). *A hybrid metaheuristic for the vehicle routing problem with stochastic demand and duration constraints.* Journal of Heuristics. 22. 10.1007/s10732-015-9281-6.

[Merkle y Middendorf, 2001] Merkle D, Middendorf M (2001) *Prospects for dynamic algorithm control: Lessons from the phase structure of ant scheduling algorithms.* In: Heckendorn RB (ed) Proceedings of the 2001 genetic and evolutionary computation conference – workshop program. Workshop "The Next Ten Years of Scheduling Research". Morgan Kaufmann Publishers, San Francisco, pp 121–126

[Metropolis y Ulam, 1949] Metropolis, N., & Ulam, S. (1949). *The Monte Carlo Method.* Journal of the American Statistical Association, 44(247), 335–341. https://doi.org/10.2307/2280232

[Minas et al, 2015] Minas, J., Hearne, J. & Martell, D. *An integrated optimization model for fuel management and fire suppression preparedness planning.* Ann Oper Res 232, 201–215 (2015). https://doi.org/10.1007/s10479-012-1298-8

[Moghaddas y Craggs, 2008] J. J. Moghaddas and L. Craggs. *A fuel treatment reduces fire severity and increases sup- pression efficiency in a mixed conifer forest.* International Journal of Wildland Fire, 16(6): 673–678, 2008.

[Moscato y Cotta, 2003] Moscato P, Cotta C (2003) *A gentle introduction to memetic algorithms.* In: Glover F, Kochenberger G (eds) Handbook of Metaheuristics. Kluwer Academic Publishers, Boston, pp 105–144

[Myers, 2010] Myers, David G. (2010). *Social psychology (Tenth ed.)*. New York, NY: McGraw-Hill. p. 94. ISBN 978-0-07337-066-8. OCLC 667213323.

[Neary et al, 1999] Neary, D.G., Klopatek, C.C., DeBano, L.F. and Ffolliott, P.F. (1999). *Fire effects on belowground sustainability: a review and synthesis*. Forest Ecology and Management, 122: 51-71.

[Orozco y Rivera, 2015] Orozco, Hernán & Rivera, Maite. (2015). *El Desafio de la Gestion Urbana en los cerros de Valparaiso*. 1.

[Pais et al, 2019] Pais, C., Carrasco, J., Martell, D., Weintraub, A., & Woodruff, D.L. (2019). *Cell2Fire: A Cell Based Forest Fire Growth Model to Support Strategic Landscape Management Planning*. Front For Glob. Change 4 (2021): 692706.

[Pais et al, 2021] Cristobal Pais, Jaime Carrasco, Pelagie Elimbi Moudio, Zuo-Jun Max Shen, *Downstream protection value: Detecting critical zones for effective fuel-treatment under wildfire risk*, Computers & Operations Research, Volume 131, 2021, 105252, ISSN 0305-0548, https://doi.org/10.1016/j.cor.2021.105252.

[Palma et al, 2007] Palma Cristian D., Cui Wenbin, Martell David L., Robak Dario, Weintraub Andres (2007) *Assessing the impact of stand-level harvests on the flammability of forest landscapes*. International Journal of Wildland Fire 16, 584-592.

[Parolin, 2020] Parolin, Zachary. (2020) *Monthly Poverty Rates in the United States during the COVID-19 Pandemic*, Poverty and Social Policy Brief 20409, Center on Poverty and Social Policy, Columbia University.

[Paton et al, 2014] Paton, D., Buergelt, P. T., McCaffrey, S., & Tedim, F. (Eds.) (2014). *Wildfire Hazards, Risks, and Disasters*. Elsevier.

[Pinto et al, 2020] G.A.S.J. Pinto, F. Rousseu, M. Niklasson, I. Drobyshev, *Effects of human-related and biotic landscape features on the occurrence and size of modern forest fires in Sweden*, Agricultural and Forest Meteorology, Volume 291, 2020, 108084, ISSN 0168-1923, https://doi.org/10.1016/j.agrformet.2020.108084.

[Pisinger y Toth, 1998] Pisinger D., Toth P. (1998) *Knapsack Problems*. pp. 306. In: Du DZ., Pardalos P.M. (eds) Handbook of Combinatorial Optimization. Springer, Boston, MA. https://doi.org/10.1007/978-1-4613-0303-9_5

[Powell, 1987] M. Powell. *Radial basis functions for multivariable interpolation*. In Algorithms for Approximation. Oxford University Press, 1987, pp. 143–167.

[Prais y Ribeiro, 1999] Prais M, Ribeiro C (1999) *Parameter variation in GRASP implementations*. In: Extended abstracts of the third metaheuristics international conference, Porto, pp 375–380

[Prais y Ribeiro, 2000] Prais M, Ribeiro C (2000) *Parameter variation in GRASP procedures*. Investigación Operativa 9:1–20

[Price, Storn y Lampinen, 2006] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential evolution: A Practical Approach to Global Optimization*. Springer, 2006.

[Pólya, 1945] Pólya, G. (1945). *How to solve it; a new aspect of mathematical method*. Princeton University Press.

[Reed y Marks, 1999] R. D. Reed and R. J. Marks II. *Neural Smithing*. MIT Press, 1999.

[Resende y Riberio, 2016] Mauricio G. C. Resende and Celso C. Ribeiro. 2016. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures* (1st. ed.). Springer Publishing Company, Incorporated.

[Richards, 1990] G. D. Richards. *An elliptical growth model of forest fire fronts and its numerical solution*. Int. J. Numer. Math Eng, 30:1133–1149, 1990.

[Richards y Gunn, 2011] Richards, Evelyn & Gunn, Eldon. (2011). *Tabu search design for difficult forest management optimization problems*. Canadian Journal of Forest Research. 33. 1126-1133. 10.1139/x03-039.

[Rodriguez et al, 2009] Rodriguez, A., Duran, J., Fernandez-Palacios, J.M. and Gallardo, A. (2009). *Short-term wildfire effects on the spatial pattern and scale of labile organic-N and inorganic-N and P pools*. Forest Ecology and Management, 257: 739-746.

[Romanyci y Pelletier, 1985] Marc H. J. Romanycia and F. J. Pelletier. (1985). *What is a heuristic?*. Computational Intelligence. p. 57.

[Running, 2016] S. M. Running. *Is global warming causing more large wildfires?* Science, 313:927–928, 2006. doi: 10.1126/science.

[Russo et al, 2016] Russo L, Russo P, Siettos CI (2016) *A Complex Network Theory Approach for the Spatial Distribution of Fire Breaks in Heterogeneous Forest Landscapes for the Control of Wildland Fires*. PLoS ONE 11(10): e0163226. https://doi.org/10.1371/journal.pone.0163226

[Rutigliano et al, 2007] Rutigliano, Flora & Anna, De & D'Ascoli, Rosaria & Castaldi, Simona & Virzo De Santo, Amalia. (2007). *Impact of fire on fungal abundance and microbial efficiency in C assimilation and mineralisation in a Mediterranean maquis soil*. Biology and Fertility of Soils. 44. 377-381. 10.1007/s00374-007-0214-x.

[Rytwinski y Crowe, 2010] Rytwinski, Adam & Crowe, Kevin. (2010). *A simulation-optimization model for selecting the location of fuel-breaks to minimize expected losses from forest fires*. Forest Ecology and Management. 260. 1-11. 10.1016/j.foreco.2010.03.013.

[Santner et al, 2003] D. T. J. Santner, B. J. Williams, and W. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

[Schumann et al, 2020] Ronald L. Schumann, Miranda Mockrin, Alexandra D. Syphard, Joshua Whittaker, Owen Price, Cassandra Johnson Gaither, Christopher T. Emrich, Van Butsic, *Wildfire recovery as a "hot moment" for creating fire-adapted communities*, International Journal of Disaster Risk Reduction, Volume 42, 2020, 101354, ISSN 2212-4209, https://doi.org/10.1016/j.ijdrr.2019.101354.

[Schöning, 1987] Schöning, Uwe (1987), *Graph isomorphism is in the low hierarchy*, Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science, pp. 114–124; also Journal of Computer and System Sciences 37: 312–323, 1988.

[Shinneman et al, 2019] D. J. Shinneman, M. J. Germino, D. S. Pilliod, C. L. Aldridge, N. M. Vaillant, and P. S. Coates. *The ecological uncertainty of wildfire fuel breaks: examples from the sagebrush steppe*. Frontiers in Ecology and the Environment, 2019.

[Silveira et al, 2020] Silveira, M. V. F., Petri, C. A., Broggio, I. S., Chagas, G. O., Macul, M. S., Leite, C. C. S. S., Ferrari, E. M. M., et al. (2020). *Drivers of Fire Anomalies in the Brazilian Amazon: Lessons Learned from the 2019 Fire Crisis*. Land, 9(12), 516. MDPI AG. Retrieved from http://dx.doi.org/10.3390/land9120516

[Snodgrass, 1953] R. E. Snodgrass. *Anatomy of the Honey Bee*. Comstock Publishing Associates, Ithaca, NY, 1953.

[Stocks et al, 2002] Stocks, B. J., et al., *Large forest fires in Canada, 1959–1997*, J. Geophys. Res., 107, 8149, doi:10.1029/2001JD000484, 2002. [printed 108( D1), 2003]

[Stützle, 1998] Stützle T (1998) *Local search algorithms for combinatorial problems – analysis, improve- ments, and new applications*. PhD thesis, FB Informatik, TU Darmstadt

[Stützle y Hoos, 2000] Stützle T, Hoos HH (2000) *MAX–MIN ant system*. Future Gener Comput Syst 16(8):889–914

[Sumner et al, 2020] Sumner, A., Ortiz-Juarez, E. & Hoy, C. (2020) *Precarity and the pandemic: COVID-19 and poverty incidence, intensity, and severity in developing countries*. WIDER Working Paper 2020/77. Helsinki: UNU-WIDER.

[Taillard y Voß, 2002] Taillard E, Voß S (2002) *POPMUSIC—partial optimization metaheuristic under special intensification conditions*. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer, Boston, pp 613–629

[Talbi, 2009] Talbi, E.-G. (2009). *Metaheuristics: From design to implementation*. Chapter 1: Common Concepts For Metaheuristics. Hoboken, N.J: John Wiley & Sons.

[Torres-Jiménez y Pavón, 2014] Torres-Jiménez, J., Pavón, J. *Applications of metaheuristics in real-life problems*. Prog Artif Intell 2, 175–176 (2014). https://doi.org/10.1007/s13748-014-0051-8 A Discrete Improved Artificial Bee Colony Algorithm for 0-1 Knapsack Problem

[Tsutsui, 2007] Tsutsui S (2007) *Ant colony optimization with cunning ants*. Trans Jpn Soc Artifi Intell 22:29–36

[Tymstra et al, 2010] C. Tymstra, R. W. Bryce, B. M. Wotton, S. W. Taylor, and A. OB. *Development and structure of Prometheus: the Canadian wildland fire growth simulation model*. Informa- tion Report NOR-X-Edmonton (AB): Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre, 417:102, 2010.

[Úbeda y Sarricolea, 2016] Xavier Úbeda, Pablo Sarricolea, *Wildfires in Chile: A review*, Global and Planetary Change, Volume 146, 2016, Pages 152-161, ISSN 0921-8181, https://doi.org/10.1016/j.gloplacha.2016.10.004.

[Vapnik, 1998] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[Vélez, 1982] Vélez, R. (1982). *Fire effects and fuel management in Mediterranean ecosystems in Spain*. In Proceedings of the symposium on Dynamics and Management of Mediterranean-Type Ecosystems. USDA-FOREST SERVICE, Gen. Tech. Report PSW-58.

[Walid, 2004] Ben-Ameur, Walid. (2004). *Computing the Initial Temperature of Simulated Annealing*. Computational Optimization and Applications. 29. 369-385. 10.1023/B:COAP.0000044187.23143.bd.

[Wei, 2012] Wei Y (2012). *Optimize landscape fuel treatment locations to create control opportunities for future fires*. Can J For Res 42(6):1002–1014. https://doi.org/10.1139/x2012-051

[Wei et al, 2008] Wei Y, Rideout D, Kirsch A (2008). *An optimization model for locating fuel treatments across a landscape to reduce expected fire losses*. Can J For Res 38(4): 868–877. https://doi.org/10.1139/X07-162

[Westerling, 2016] A. L. Westerling. *Increasing western us forest wildfire activity: sensitivity to changes in the timing of spring*. Phil. Trans. R. Soc. B, 371(20150178), 2016.

[Whitley, 1989] Whitley D (1989) *The GENITOR algorithm and selection pressure: why rank-based alloca-tion of reproductive trials is best*. In: Proceedings of the international conference on genetic algorithms. Morgan Kaufmann, pp 116–121

[Wiesemann y Stützle, 2006] Wiesemann W, Stützle T (2006) *Iterated ants: an experimental study for the quadratic assignment problem*. In: Dorigo M et al (eds) 5th international workshop on Ant colony optimization and swarm intelligence (ANTS 2006). Lecture notes in computer science, vol 4150. Springer, Heidelberg, pp 179–190

[Wolpert y Macready, 1997] Wolpert DH, Macready WG (1997) *No free lunch theorems for optimization*. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893

[Xinglei et al, 2019] Xinglei Cui, Md Azharul Alam, George LW. Perry, Adrian M. Paterson, Sarah V. Wyse, Timothy J. Curran, *Green firebreaks as a management tool for wildfires: Lessons from China*, Journal of Environmental Management, Volume 233, 2019, Pages 329-336, ISSN 0301-4797, https://doi.org/10.1016/j.jenvman.2018.12.043. (https://www.sciencedirect.com/science/article/pii/S0301479718314658)

[Yang, 2008] Yang, X.-S. (2008). *Nature-Inspired Metaheuristic Algorithm*, 2nd ed.. England: Luniver press.

[Yang, 2021] Xin-She Yang. *Nature-Inspired Optimization Algorithms (Second Edition)*, Academic Press, 2021, Chapter 1 - Introduction to Algorithms, Editor(s): Xin-She Yang, Pages 1-22, ISBN 9780128219867, https://doi.org/10.1016/B978-0-12-821986-7.00008-1

[Zhang, 2019] Zhang, Song & Liu, Sanyang. (2019). *A Discrete Improved Artificial Bee Colony Algorithm for 0–1 Knapsack Problem*. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2930638.

[Zheng, 2010] Zheng R. (2018) *An Optimal Allocation Scheme for City Fire Resources Based on Integer Programming*. In: Li X., Xu X. (eds) Proceedings of the Fifth International Forum on Decision Sciences. Uncertainty and Operations Research. Springer, Singapore. https://doi.org/10.1007/978-981-10-7817-0_11

# Anexos

## Anexo A. Desempeño algoritmos según variación

## Desempeño Algoritmos según tipo de solución

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| Tipo *Cluster* | 0.25 | 0.319 | 0.309 | 0.321 | 0.272 |
| Tipo Nodo | 0.361 | 0.36 | 0.36 | 0.363 | 0.342 |

Tabla 8.1: Desempeño Algoritmo Genético según tipo de solución

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| Tipo *Cluster* | 0.313 | 0.318 | 0.332 | 0.319 | 0.321 |
| Tipo Nodo | 0.405 | 0.355 | 0.369 | 0.367 | 0.416 |

Tabla 8.2: Desempeño Búsqueda Tabú según tipo de solución

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| Tipo *Cluster* | 0.229 | 0.315 | 0.267 | 0.224 | 0.229 |
| Tipo Nodo | 0.407 | 0.337 | 0.405 | 0.37 | 0.369 |

Tabla 8.3: Desempeño GRASP según tipo de solución

## Desempeño Algoritmos según solución inicial

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| Solución Aleatoria | 0.25 | 0.319 | 0.309 | 0.321 | 0.272 |
| Solución Martínez | 0.347 | 0.299 | 0.304 | 0.325 | 0.302 |

Tabla 8.4: Desempeño Algoritmo Genético según solución inicial

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| Solución Aleatoria | 0.313 | 0.318 | 0.332 | 0.319 | 0.321 |
| Solución Martínez | 0.274 | 0.344 | 0.309 | 0.276 | 0.34 |

Tabla 8.5: Desempeño Algoritmo Genético según solución inicial

## Desempeño Algoritmos según sesgo de movimiento

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| $\phi = 0.6$ | 0.347 | 0.299 | 0.304 | 0.325 | 0.302 |
| $\phi = 1$ | 0.34 | 0.315 | 0.3 | 0.341 | 0.359 |

Tabla 8.6: Desempeño Algoritmo Genético según sesgo de movimiento

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| $\phi = 0.6$ | 0.274 | 0.344 | 0.309 | 0.276 | 0.34 |
| $\phi = 1$ | 0.364 | 0.338 | 0.326 | 0.318 | 0.342 |

Tabla 8.7: Desempeño Búsqueda Tabú según sesgo de movimiento

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| $\phi = 0.6$ | 0.229 | 0.315 | 0.267 | 0.224 | 0.229 |
| $\phi = 1$ | 0.328 | 0.241 | 0.235 | 0.255 | 0.237 |

Tabla 8.8: Desempeño GRASP según sesgo de movimiento

# Anexo B. Desempeño algoritmos en cada escenario

## Detalle desempeño algoritmos según variación en cada bosque

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.229 | 0.315 | 0.267 | 0.224 | 0.229 |
| Búsqueda Tabú | 0.313 | 0.318 | 0.332 | 0.319 | 0.321 |
| Algoritmo Genético | 0.25 | 0.319 | 0.309 | 0.321 | 0.272 |
| Aleatorio | 0.353 | 0.376 | 0.353 | 0.336 | 0.316 |
| Libre | 0.386 | 0.363 | 0.389 | 0.401 | 0.368 |
| Martínez | 0.331 | 0.348 | 0.339 | 0.308 | 0.319 |

Tabla 8.9: Desempeño Algoritmos en bosque **a)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.245 | 0.189 | 0.307 | 0.222 | 0.313 |
| Búsqueda Tabú | 0.345 | 0.354 | 0.35 | 0.321 | 0.335 |
| Algoritmo Genético | 0.327 | 0.29 | 0.354 | 0.307 | 0.297 |
| Aleatorio | 0.341 | 0.38 | 0.337 | 0.328 | 0.361 |
| Libre | 0.407 | 0.369 | 0.399 | 0.405 | 0.415 |
| Martínez | 0.358 | 0.31 | 0.316 | 0.374 | 0.346 |

Tabla 8.10: Desempeño Algoritmos en bosque **a)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.249 | 0.268 | 0.297 | 0.287 | 0.302 |
| Búsqueda Tabú | 0.291 | 0.306 | 0.313 | 0.301 | 0.258 |
| Algoritmo Genético | 0.275 | 0.277 | 0.301 | 0.289 | 0.275 |
| Aleatorio | 0.238 | 0.312 | 0.34 | 0.23 | 0.238 |
| Libre | 0.296 | 0.316 | 0.348 | 0.325 | 0.355 |
| Martínez | 0.281 | 0.235 | 0.316 | 0.297 | 0.302 |

Tabla 8.11: Desempeño Algoritmos en bosque **a)** con vientos en 30 direcciones y con punto de ignición aleatorio

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.287 | 0.28 | 0.27 | 0.291 | 0.277 |
| Búsqueda Tabú | 0.306 | 0.313 | 0.287 | 0.278 | 0.291 |
| Algoritmo Genético | 0.308 | 0.313 | 0.287 | 0.278 | 0.291 |
| Aleatorio | 0.217 | 0.279 | 0.287 | 0.309 | 0.327 |
| Libre | 0.322 | 0.343 | 0.339 | 0.271 | 0.341 |
| Martínez | 0.279 | 0.243 | 0.297 | 0.312 | 0.264 |

Tabla 8.12: Desempeño Algoritmos en bosque **a)** con vientos en 8 direcciones y con punto de ignición aleatorio

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.175 | 0.172 | 0.217 | 0.24 | 0.159 |
| Búsqueda Tabú | 0.286 | 0.291 | 0.269 | 0.292 | 0.291 |
| Algoritmo Genético | 0.274 | 0.3 | 0.302 | 0.315 | 0.303 |
| Aleatorio | 0.276 | 0.308 | 0.33 | 0.282 | 0.296 |
| Libre | 0.339 | 0.331 | 0.331 | 0.334 | 0.326 |
| Martínez | 0.296 | 0.283 | 0.288 | 0.31 | 0.282 |

Tabla 8.13: Desempeño Algoritmos en bosque **c)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.21 | 0.182 | 0.188 | 0.2 | 0.185 |
| Búsqueda Tabú | 0.264 | 0.278 | 0.302 | 0.258 | 0.251 |
| Algoritmo Genético | 0.259 | 0.261 | 0.193 | 0.268 | 0.199 |
| Aleatorio | 0.297 | 0.292 | 0.28 | 0.267 | 0.289 |
| Libre | 0.318 | 0.313 | 0.291 | 0.322 | 0.314 |
| Martínez | 0.292 | 0.29 | 0.311 | 0.266 | 0.311 |

Tabla 8.14: Desempeño Algoritmos en bosque **c)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.273 | 0.502 | 0.448 | 0.424 | 0.418 |
| Búsqueda Tabú | 0.409 | 0.476 | 0.435 | 0.423 | 0.4 |
| Algoritmo Genético | 0.336 | 0.435 | 0.351 | 0.504 | 0.426 |
| Aleatorio | 0.489 | 0.488 | 0.401 | 0.418 | 0.468 |
| Libre | 0.486 | 0.557 | 0.516 | 0.484 | 0.492 |

Tabla 8.15: Desempeño Algoritmos en bosque **b)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 0.42 | 0.485 | 0.479 | 0.35 | 0.474 |
| Búsqueda Tabú | 0.562 | 0.421 | 0.439 | 0.545 | 0.538 |
| Algoritmo Genético | 0.418 | 0.554 | 0.459 | 0.407 | 0.489 |
| Aleatorio | 0.457 | 0.513 | 0.53 | 0.457 | 0.495 |
| Libre | 0.54 | 0.527 | 0.56 | 0.556 | 0.48 |

Tabla 8.16: Desempeño Algoritmos en bosque **d)** con vientos en 8 direcciones y con punto de ignición en cuadrado

# Anexo C. Número de iteraciones de algoritmos en cada escenario

## Número de iteraciones algoritmos según variación en cada bosque

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 150 | 204 | 211 | 212 | 201 |
| Búsqueda Tabú | 75 | 44 | 131 | 104 | 120 |
| Algoritmo Genético | 23 | 28 | 31 | 28 | 27 |

Tabla 8.17: Número de iteraciones algoritmos en bosque **a)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 129 | 108 | 194 | 190 | 91 |
| Búsqueda Tabú | 39 | 46 | 37 | 46 | 33 |
| Algoritmo Genético | 30 | 22 | 15 | 28 | 26 |

Tabla 8.18: Número de iteraciones algoritmos en bosque **a)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 80 | 87 | 62 | 83 | 85 |
| Búsqueda Tabú | 28 | 45 | 45 | 31 | 41 |
| Algoritmo Genético | 15 | 26 | 25 | 14 | 17 |

Tabla 8.19: Número de iteraciones algoritmos en bosque **a)** con vientos en 30 direcciones y con punto de ignición aleatorio

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 82 | 86 | 59 | 83 | 86 |
| Búsqueda Tabú | 28 | 45 | 45 | 30 | 40 |
| Algoritmo Genético | 16 | 25 | 23 | 14 | 17 |

Tabla 8.20: Número de iteraciones algoritmos en bosque **a)** con vientos en 8 direcciones y con punto de ignición aleatorio

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 906 | 78 | 96 | 89 | 85 |
| Búsqueda Tabú | 53 | 48 | 35 | 47 | 52 |
| Algoritmo Genético | 40 | 36 | 24 | 26 | 34 |

Tabla 8.21: Número de iteraciones algoritmos en bosque **c)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 118 | 116 | 110 | 56 | 118 |
| Búsqueda Tabú | 58 | 2 | 33 | 58 | 58 |
| Algoritmo Genético | 34 | 31 | 23 | 23 | 40 |

Tabla 8.22: Número de iteraciones algoritmos en bosque **c)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 232 | 249 | 206 | 171 | 236 |
| Búsqueda Tabú | 116 | 116 | 70 | 68 | 118 |
| Algoritmo Genético | 93 | 86 | 56 | 70 | 92 |

Tabla 8.23: Número de iteraciones algoritmos en bosque **b)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 243 | 104 | 263 | 259 | 170 |
| Búsqueda Tabú | 121 | 106 | 101 | 127 | 123 |
| Algoritmo Genético | 96 | 78 | 59 | 93 | 85 |

Tabla 8.24: Número de iteraciones algoritmos en bosque **d)** con vientos en 8 direcciones y con punto de ignición en cuadrado

# Anexo D. Evolución del desempeño en el tiempo de algoritmos en cada escenario

## Detalle evolución desempeño algoritmos en el tiempo según variación en cada bosque

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.33 | 0.37 | 0.403 | 0.36 | 0.33 |
| 2000 | 0.325 | 0.345 | 0.396 | 0.35 | 0.329 |
| 3000 | 0.301 | 0.321 | 0.345 | 0.35 | 0.292 |
| 4000 | 0.293 | 0.319 | 0.315 | 0.343 | 0.295 |
| 5000 | 0.291 | 0.315 | 0.321 | 0.311 | 0.281 |
| 6000 | 0.277 | 0.305 | 0.311 | 0.302 | 0.274 |
| 7000 | 0.268 | 0.302 | 0.305 | 0.308 | 0.254 |
| 7200 | 0.26 | 0.295 | 0.291 | 0.303 | 0.25 |

Tabla 8.25: Desempeño Algoritmo Genético en el tiempo en bosque **a)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.232 | 0.276 | 0.256 | 0.275 | 0.249 |
| 2000 | 0.232 | 0.219 | 0.256 | 0.25 | 0.212 |
| 3000 | 0.232 | 0.219 | 0.256 | 0.25 | 0.212 |
| 4000 | 0.232 | 0.219 | 0.244 | 0.19 | 0.212 |
| 5000 | 0.1868 | 0.219 | 0.244 | 0.19 | 0.212 |
| 6000 | 0.1868 | 0.219 | 0.222 | 0.19 | 0.212 |
| 7000 | 0.1868 | 0.219 | 0.222 | 0.19 | 0.212 |
| 7200 | 0.1868 | 0.219 | 0.222 | 0.19 | 0.212 |

Tabla 8.26: Desempeño Búsqueda Tabú en el tiempo en bosque **a)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.249 | 0.251 | 0.244 | 0.326 | 0.274 |
| 2000 | 0.249 | 0.251 | 0.244 | 0.257 | 0.235 |
| 3000 | 0.249 | 0.251 | 0.244 | 0.257 | 0.235 |
| 4000 | 0.249 | 0.251 | 0.244 | 0.254 | 0.235 |
| 5000 | 0.249 | 0.251 | 0.244 | 0.254 | 0.235 |
| 6000 | 0.249 | 0.251 | 0.244 | 0.246 | 0.235 |
| 7000 | 0.228 | 0.251 | 0.244 | 0.246 | 0.218 |
| 7200 | 0.228 | 0.251 | 0.244 | 0.246 | 0.218 |

Tabla 8.27: Desempeño GRASP en el tiempo en bosque **a)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.317 | 0.341 | 0.351 | 0.343 | 0.359 |
| 2000 | 0.317 | 0.347 | 0.351 | 0.346 | 0.341 |
| 3000 | 0.33 | 0.33 | 0.351 | 0.324 | 0.37 |
| 4000 | 0.328 | 0.323 | 0.351 | 0.303 | 0.346 |
| 5000 | 0.32 | 0.317 | 0.351 | 0.303 | 0.338 |
| 6000 | 0.316 | 0.305 | 0.35 | 0.305 | 0.328 |
| 7000 | 0.314 | 0.281 | 0.335 | 0.3 | 0.29 |
| 7200 | 0.294 | 0.273 | 0.319 | 0.299 | 0.312 |

Tabla 8.28: Desempeño Algoritmo Genético en el tiempo en bosque **a)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.247 | 0.247 | 0.246 | 0.268 | 0.257 |
| 2000 | 0.244 | 0.247 | 0.246 | 0.26 | 0.252 |
| 3000 | 0.244 | 0.247 | 0.239 | 0.207 | 0.251 |
| 4000 | 0.244 | 0.247 | 0.239 | 0.207 | 0.239 |
| 5000 | 0.244 | 0.247 | 0.239 | 0.207 | 0.239 |
| 6000 | 0.235 | 0.247 | 0.231 | 0.207 | 0.239 |
| 7000 | 0.231 | 0.241 | 0.231 | 0.207 | 0.239 |
| 7200 | 0.231 | 0.241 | 0.231 | 0.207 | 0.239 |

Tabla 8.29: Desempeño Búsqueda Tabú en el tiempo en bosque **a)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.241 | 0.265 | 0.275 | 0.25 | 0.297 |
| 2000 | 0.235 | 0.224 | 0.269 | 0.243 | 0.259 |
| 3000 | 0.235 | 0.223 | 0.231 | 0.227 | 0.259 |
| 4000 | 0.233 | 0.223 | 0.214 | 0.198 | 0.183 |
| 5000 | 0.233 | 0.223 | 0.202 | 0.198 | 0.183 |
| 6000 | 0.233 | 0.165 | 0.202 | 0.198 | 0.183 |
| 7000 | 0.219 | 0.165 | 0.202 | 0.198 | 0.183 |
| 7200 | 0.219 | 0.165 | 0.202 | 0.198 | 0.183 |

Tabla 8.30: Desempeño GRASP en el tiempo en bosque **a)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.267 | 0.29 | 0.31 | 0.287 | 0.261 |
| 2000 | 0.281 | 0.29 | 0.3 | 0.287 | 0.261 |
| 3000 | 0.25 | 0.298 | 0.257 | 0.302 | 0.293 |
| 4000 | 0.249 | 0.269 | 0.251 | 0.282 | 0.273 |
| 5000 | 0.236 | 0.283 | 0.234 | 0.292 | 0.27 |
| 6000 | 0.232 | 0.266 | 0.227 | 0.305 | 0.273 |
| 7000 | 0.229 | 0.27 | 0.215 | 0.28 | 0.255 |
| 7200 | 0.221 | 0.261 | 0.206 | 0.262 | 0.246 |

Tabla 8.31: Desempeño Algoritmo Genético en el tiempo en bosque **a)** con vientos en 30 direcciones y con punto de ignición aleatorio

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.064 | 0.151 | 0.131 | 0.148 | 0.158 |
| 2000 | 0.064 | 0.136 | 0.131 | 0.148 | 0.158 |
| 3000 | 0.064 | 0.136 | 0.124 | 0.123 | 0.082 |
| 4000 | 0.064 | 0.136 | 0.124 | 0.123 | 0.082 |
| 5000 | 0.064 | 0.136 | 0.124 | 0.123 | 0.082 |
| 6000 | 0.064 | 0.136 | 0.108 | 0.123 | 0.082 |
| 7000 | 0.064 | 0.115 | 0.108 | 0.123 | 0.082 |
| 7200 | 0.064 | 0.115 | 0.106 | 0.123 | 0.082 |

Tabla 8.32: Desempeño Búsqueda Tabú en el tiempo en bosque **a)** con vientos en 30 direcciones y con punto de ignición aleatorio

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.225 | 0.252 | 0.229 | 0.194 | 0.237 |
| 2000 | 0.218 | 0.239 | 0.191 | 0.194 | 0.194 |
| 3000 | 0.211 | 0.185 | 0.191 | 0.194 | 0.172 |
| 4000 | 0.152 | 0.185 | 0.191 | 0.194 | 0.172 |
| 5000 | 0.152 | 0.176 | 0.18 | 0.194 | 0.172 |
| 6000 | 0.152 | 0.176 | 0.18 | 0.194 | 0.172 |
| 7000 | 0.152 | 0.176 | 0.158 | 0.114 | 0.172 |
| 7200 | 0.152 | 0.176 | 0.158 | 0.114 | 0.172 |

Tabla 8.33: Desempeño GRASP en el tiempo en bosque **a)** con vientos en 30 direcciones y con punto de ignición aleatorio

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.28 | 0.279 | 0.28 | 0.289 | 0.293 |
| 2000 | 0.277 | 0.274 | 0.278 | 0.283 | 0.283 |
| 3000 | 0.275 | 0.283 | 0.271 | 0.283 | 0.275 |
| 4000 | 0.26 | 0.285 | 0.265 | 0.28 | 0.27 |
| 5000 | 0.257 | 0.269 | 0.259 | 0.273 | 0.272 |
| 6000 | 0.251 | 0.256 | 0.249 | 0.269 | 0.256 |
| 7000 | 0.254 | 0.243 | 0.245 | 0.265 | 0.251 |
| 7200 | 0.24 | 0.239 | 0.238 | 0.26 | 0.246 |

Tabla 8.34: Desempeño Algoritmo Genético en el tiempo en bosque **a)** con vientos en 8 direcciones y con punto de ignición aleatorio

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.136 | 0.156 | 0.181 | 0.111 | 0.145 |
| 2000 | 0.112 | 0.106 | 0.177 | 0.111 | 0.128 |
| 3000 | 0.112 | 0.106 | 0.177 | 0.111 | 0.128 |
| 4000 | 0.112 | 0.106 | 0.126 | 0.111 | 0.128 |
| 5000 | 0.112 | 0.106 | 0.126 | 0.111 | 0.11 |
| 6000 | 0.112 | 0.106 | 0.126 | 0.111 | 0.11 |
| 7000 | 0.112 | 0.106 | 0.126 | 0.111 | 0.11 |
| 7200 | 0.112 | 0.106 | 0.126 | 0.111 | 0.11 |

Tabla 8.35: Desempeño Búsqueda Tabú en el tiempo en bosque **a)** con vientos en 8 direcciones y con punto de ignición aleatorio

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.215 | 0.233 | 0.212 | 0.231 | 0.201 |
| 2000 | 0.208 | 0.199 | 0.203 | 0.177 | 0.201 |
| 3000 | 0.208 | 0.199 | 0.203 | 0.177 | 0.199 |
| 4000 | 0.179 | 0.199 | 0.203 | 0.177 | 0.179 |
| 5000 | 0.134 | 0.199 | 0.203 | 0.177 | 0.179 |
| 6000 | 0.134 | 0.199 | 0.198 | 0.177 | 0.179 |
| 7000 | 0.134 | 0.199 | 0.198 | 0.177 | 0.179 |
| 7200 | 0.134 | 0.199 | 0.179 | 0.177 | 0.179 |

Tabla 8.36: Desempeño GRASP en el tiempo en bosque **a)** con vientos en 8 direcciones y con punto de ignición aleatorio

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.29 | 0.298 | 0.293 | 0.318 | 0.309 |
| 2000 | 0.293 | 0.287 | 0.298 | 0.307 | 0.291 |
| 3000 | 0.291 | 0.274 | 0.298 | 0.307 | 0.284 |
| 4000 | 0.285 | 0.276 | 0.298 | 0.287 | 0.27 |
| 5000 | 0.278 | 0.27 | 0.283 | 0.277 | 0.268 |
| 6000 | 0.273 | 0.27 | 0.27 | 0.297 | 0.267 |
| 7000 | 0.256 | 0.259 | 0.268 | 0.295 | 0.257 |
| 7200 | 0.25 | 0.257 | 0.266 | 0.295 | 0.275 |

Tabla 8.37: Desempeño Algoritmo Genético en el tiempo en bosque **c)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.239 | 0.232 | 0.219 | 0.248 | 0.24 |
| 2000 | 0.224 | 0.232 | 0.203 | 0.237 | 0.214 |
| 3000 | 0.224 | 0.228 | 0.203 | 0.237 | 0.214 |
| 4000 | 0.224 | 0.228 | 0.203 | 0.224 | 0.214 |
| 5000 | 0.202 | 0.199 | 0.203 | 0.224 | 0.214 |
| 6000 | 0.202 | 0.199 | 0.203 | 0.224 | 0.214 |
| 7000 | 0.202 | 0.199 | 0.203 | 0.219 | 0.214 |
| 7200 | 0.202 | 0.199 | 0.201 | 0.219 | 0.214 |

Tabla 8.38: Desempeño Búsqueda Tabú en el tiempo en bosque **c)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.226 | 0.11 | 0.203 | 0.185 | 0.17 |
| 2000 | 0.226 | 0.11 | 0.203 | 0.185 | 0.17 |
| 3000 | 0.226 | 0.11 | 0.181 | 0.18 | 0.17 |
| 4000 | 0.226 | 0.11 | 0.149 | 0.18 | 0.17 |
| 5000 | 0.226 | 0.11 | 0.149 | 0.18 | 0.17 |
| 6000 | 0.226 | 0.11 | 0.149 | 0.18 | 0.16 |
| 7000 | 0.226 | 0.11 | 0.149 | 0.18 | 0.151 |
| 7200 | 0.226 | 0.11 | 0.149 | 0.18 | 0.151 |

Tabla 8.39: Desempeño GRASP en el tiempo en bosque **c)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.259 | 0.261 | 0.288 | 0.283 | 0.254 |
| 2000 | 0.264 | 0.253 | 0.257 | 0.28 | 0.223 |
| 3000 | 0.253 | 0.262 | 0.249 | 0.277 | 0.192 |
| 4000 | 0.271 | 0.251 | 0.248 | 0.255 | 0.197 |
| 5000 | 0.265 | 0.254 | 0.245 | 0.254 | 0.19 |
| 6000 | 0.269 | 0.251 | 0.235 | 0.252 | 0.184 |
| 7000 | 0.254 | 0.24 | 0.224 | 0.234 | 0.174 |
| 7200 | 0.243 | 0.235 | 0.222 | 0.233 | 0.162 |

Tabla 8.40: Desempeño Algoritmo Genético en el tiempo en bosque **c)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.217 | 0.254 | 0.196 | 0.218 | 0.198 |
| 2000 | 0.198 | 0.254 | 0.193 | 0.218 | 0.173 |
| 3000 | 0.198 | 0.254 | 0.193 | 0.197 | 0.173 |
| 4000 | 0.196 | 0.254 | 0.193 | 0.197 | 0.173 |
| 5000 | 0.196 | 0.254 | 0.193 | 0.197 | 0.173 |
| 6000 | 0.196 | 0.254 | 0.193 | 0.197 | 0.173 |
| 7000 | 0.196 | 0.254 | 0.193 | 0.197 | 0.173 |
| 7200 | 0.196 | 0.254 | 0.193 | 0.197 | 0.173 |

Tabla 8.41: Desempeño Búsqueda Tabú en el tiempo en bosque **c)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.191 | 0.184 | 0.145 | 0.196 | 0.171 |
| 2000 | 0.179 | 0.151 | 0.145 | 0.196 | 0.17 |
| 3000 | 0.179 | 0.151 | 0.145 | 0.185 | 0.17 |
| 4000 | 0.154 | 0.151 | 0.145 | 0.185 | 0.146 |
| 5000 | 0.154 | 0.151 | 0.145 | 0.183 | 0.146 |
| 6000 | 0.154 | 0.151 | 0.145 | 0.183 | 0.146 |
| 7000 | 0.154 | 0.151 | 0.145 | 0.183 | 0.146 |
| 7200 | 0.154 | 0.151 | 0.145 | 0.183 | 0.146 |

Tabla 8.42: Desempeño GRASP en el tiempo en bosque **c)** con vientos en 30 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.483 | 0.51 | 0.48 | 0.502 | 0.53 |
| 2000 | 0.432 | 0.482 | 0.472 | 0.488 | 0.488 |
| 3000 | 0.41 | 0.48 | 0.46 | 0.482 | 0.47 |
| 4000 | 0.37 | 0.47 | 0.43 | 0.479 | 0.475 |
| 5000 | 0.358 | 0.438 | 0.405 | 0.448 | 0.448 |
| 6000 | 0.343 | 0.414 | 0.388 | 0.414 | 0.412 |
| 7000 | 0.341 | 0.414 | 0.333 | 0.414 | 0.402 |
| 7200 | 0.333 | 0.405 | 0.302 | 0.414 | 0.418 |

Tabla 8.43: Desempeño Algoritmo Genético en el tiempo en bosque **b)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.234 | 0.29 | 0.301 | 0.244 | 0.288 |
| 2000 | 0.234 | 0.244 | 0.301 | 0.244 | 0.223 |
| 3000 | 0.234 | 0.244 | 0.298 | 0.244 | 0.223 |
| 4000 | 0.234 | 0.244 | 0.298 | 0.244 | 0.223 |
| 5000 | 0.234 | 0.244 | 0.298 | 0.244 | 0.223 |
| 6000 | 0.234 | 0.244 | 0.226 | 0.244 | 0.223 |
| 7000 | 0.234 | 0.244 | 0.226 | 0.244 | 0.223 |
| 7200 | 0.234 | 0.244 | 0.226 | 0.244 | 0.223 |

Tabla 8.44: Desempeño Búsqueda Tabú en el tiempo en bosque **b)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.303 | 0.27 | 0.367 | 0.371 | 0.346 |
| 2000 | 0.303 | 0.27 | 0.308 | 0.356 | 0.346 |
| 3000 | 0.303 | 0.27 | 0.308 | 0.356 | 0.346 |
| 4000 | 0.303 | 0.27 | 0.308 | 0.351 | 0.298 |
| 5000 | 0.286 | 0.27 | 0.308 | 0.351 | 0.249 |
| 6000 | 0.286 | 0.27 | 0.308 | 0.331 | 0.249 |
| 7000 | 0.286 | 0.27 | 0.308 | 0.316 | 0.249 |
| 7200 | 0.286 | 0.27 | 0.308 | 0.316 | 0.249 |

Tabla 8.45: Desempeño GRASP en el tiempo en bosque **b)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.493 | 0.51 | 0.518 | 0.516 | 0.53 |
| 2000 | 0.482 | 0.518 | 0.507 | 0.488 | 0.498 |
| 3000 | 0.47 | 0.508 | 0.506 | 0.472 | 0.5 |
| 4000 | 0.437 | 0.5 | 0.493 | 0.459 | 0.482 |
| 5000 | 0.438 | 0.498 | 0.49 | 0.438 | 0.468 |
| 6000 | 0.43 | 0.474 | 0.488 | 0.414 | 0.476 |
| 7000 | 0.42 | 0.474 | 0.463 | 0.394 | 0.462 |
| 7200 | 0.424 | 0.463 | 0.45 | 0.377 | 0.433 |

Tabla 8.46: Desempeño Algoritmo Genético en el tiempo en bosque **d)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.317 | 0.291 | 0.321 | 0.331 | 0.334 |
| 2000 | 0.317 | 0.291 | 0.321 | 0.311 | 0.311 |
| 3000 | 0.297 | 0.291 | 0.321 | 0.311 | 0.292 |
| 4000 | 0.297 | 0.291 | 0.321 | 0.311 | 0.292 |
| 5000 | 0.297 | 0.291 | 0.309 | 0.307 | 0.292 |
| 6000 | 0.297 | 0.291 | 0.296 | 0.307 | 0.292 |
| 7000 | 0.297 | 0.274 | 0.278 | 0.307 | 0.292 |
| 7200 | 0.297 | 0.274 | 0.278 | 0.248 | 0.292 |

Tabla 8.47: Desempeño Búsqueda Tabú en el tiempo en bosque **d)** con vientos en 8 direcciones y con punto de ignición en cuadrado

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 1000 | 0.375 | 0.349 | 0.364 | 0.404 | 0.438 |
| 2000 | 0.375 | 0.349 | 0.35 | 0.404 | 0.374 |
| 3000 | 0.375 | 0.349 | 0.35 | 0.404 | 0.304 |
| 4000 | 0.352 | 0.349 | 0.339 | 0.359 | 0.304 |
| 5000 | 0.352 | 0.349 | 0.339 | 0.359 | 0.304 |
| 6000 | 0.325 | 0.349 | 0.339 | 0.359 | 0.304 |
| 7000 | 0.325 | 0.349 | 0.339 | 0.359 | 0.304 |
| 7200 | 0.325 | 0.349 | 0.339 | 0.359 | 0.304 |

Tabla 8.48: Desempeño GRASP en el tiempo en bosque **d)** con vientos en 8 direcciones y con punto de ignición en cuadrado

# Anexo E. Gráficos de la evolución del desempeño en el tiempo de algoritmos en cada escenario

## Gráficos evolución algoritmos según variación en cada bosque

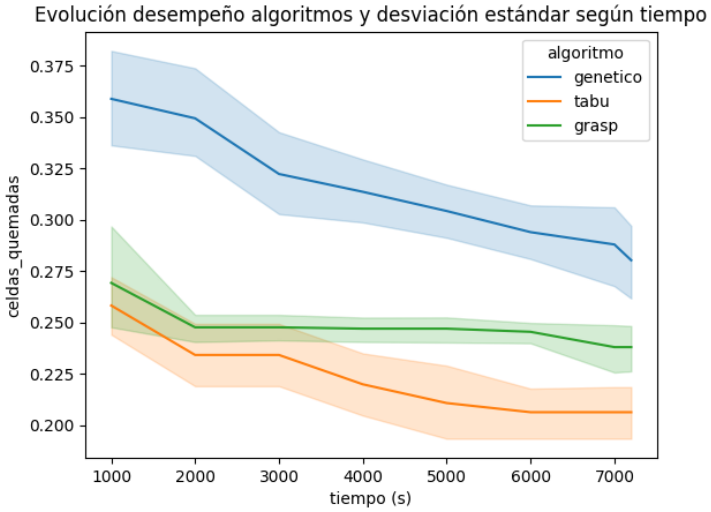



Figura 8.1: Evolución en el tiempo desempeño bosque **a)** con vientos en 8 direcciones y punto de ignición en cuadrado.

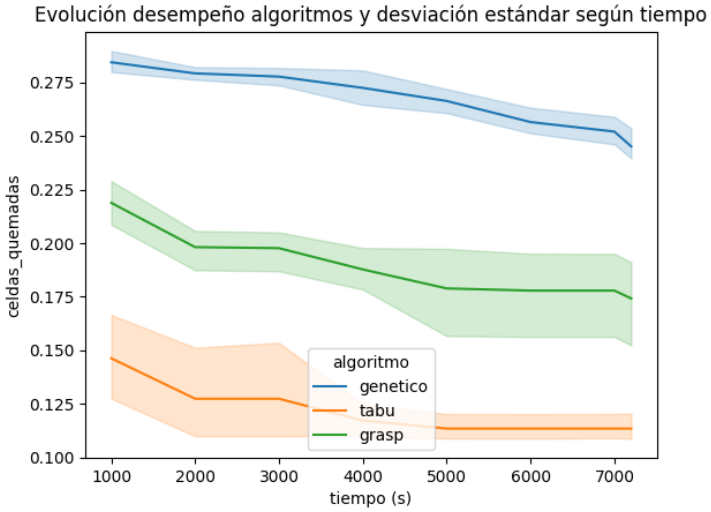



Figura 8.2: Evolución en el tiempo desempeño bosque **a)** con vientos en 8 direcciones y punto de ignición aleatorio.

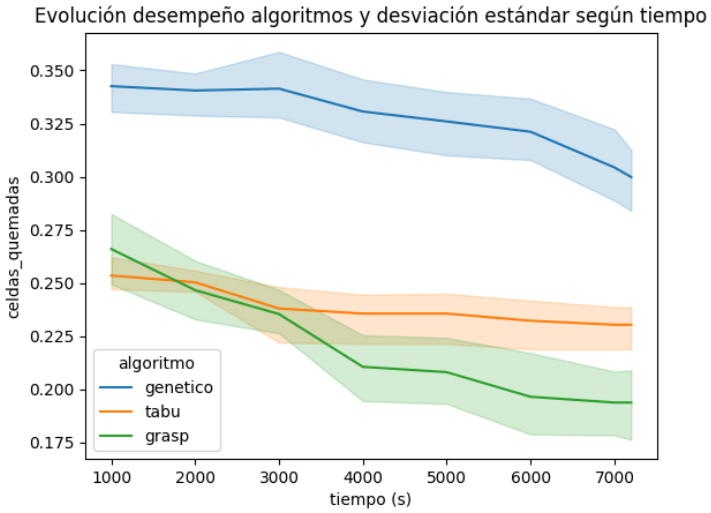



Figura 8.3: Evolución en el tiempo desempeño bosque **a)** con vientos en 30 direcciones y punto de ignición en cuadrado.

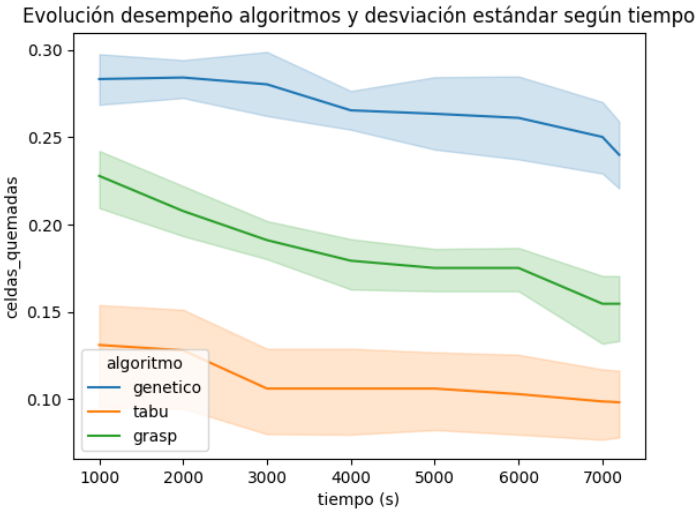



Figura 8.4: Evolución en el tiempo desempeño bosque **a)** con vientos en 30 direcciones y punto de ignición aleatorio.

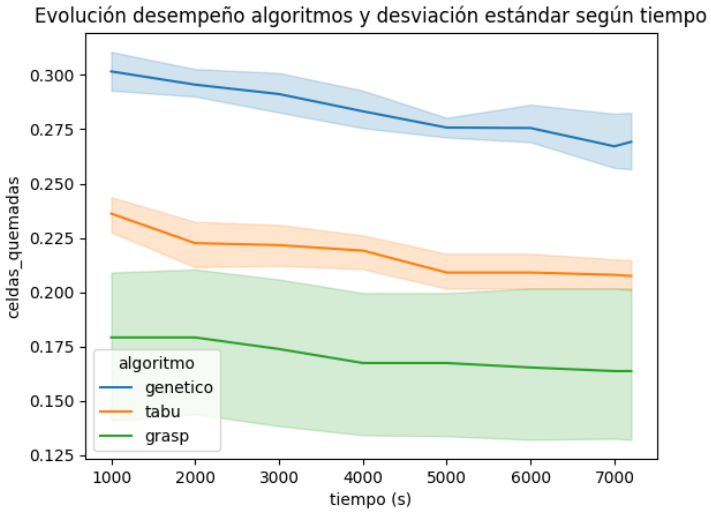



Figura 8.5: Evolución en el tiempo desempeño bosque **c)** con vientos en 8 direcciones y punto de ignición en cuadrado.

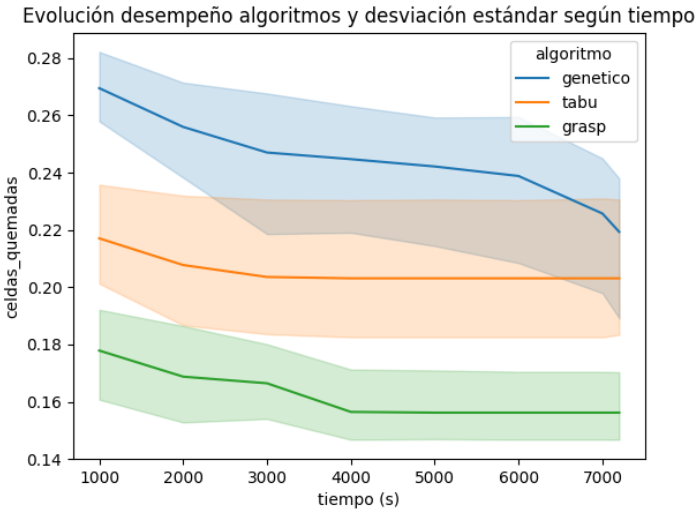



Figura 8.6: Evolución en el tiempo desempeño bosque **c)** con vientos en 30 direcciones y punto de ignición en cuadrado.

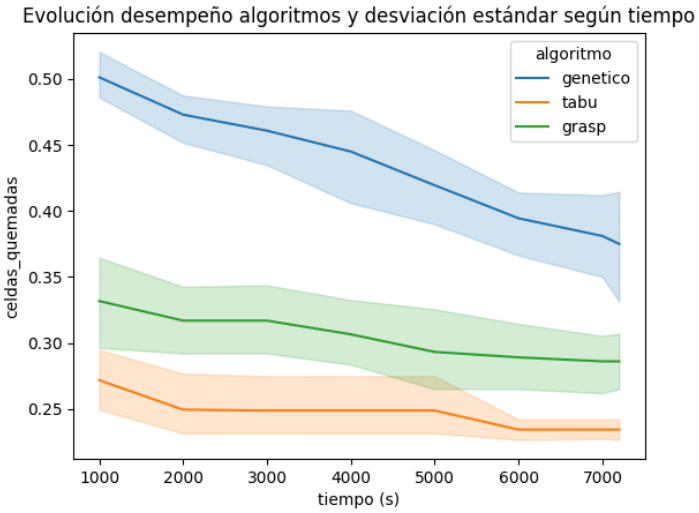



Figura 8.7: Evolución en el tiempo desempeño bosque **b)** con vientos en 8 direcciones y punto de ignición en cuadrado.

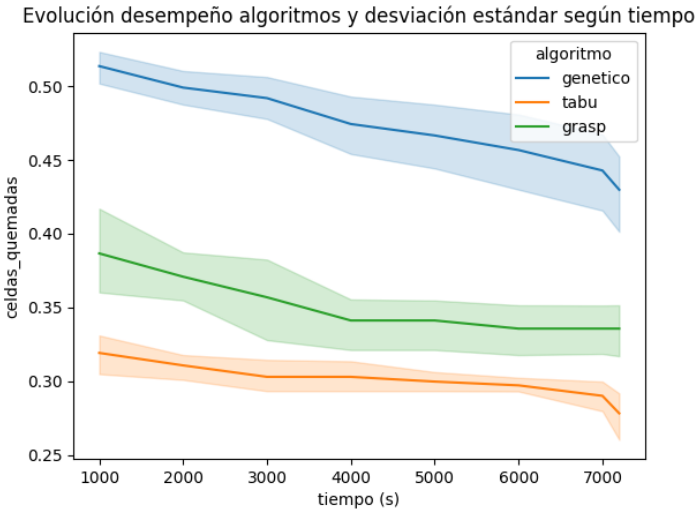



Figura 8.8: Evolución en el tiempo desempeño bosque **d)** con vientos en 8 direcciones y punto de ignición en cuadrado.

# Anexo F. Resultados análisis de sensibilidad

## Desempeño análisis de sensibilidad según porcentaje máximo de cortafuego a ubicar

| Metodología | % Cortafuegos | Promedio | Error | Min | 25% | 50% | 75% | Max |
|---|---|---|---|---|---|---|---|---|
| Aleatorio | 5% | 0.284 | 0.042 | 0.217 | 0.279 | 0.287 | 0.309 | 0.327 |
| | 7% | 0.258 | 0.019 | 0.236 | 0.252 | 0.254 | 0.259 | 0.288 |
| | 10% | 0.241 | 0.033 | 0.209 | 0.216 | 0.243 | 0.243 | 0.294 |
| Algoritmo Genético | 5% | 0.307 | 0.015 | 0.288 | 0.295 | 0.308 | 0.319 | 0.323 |
| | 7% | 0.268 | 0.033 | 0.228 | 0.243 | 0.275 | 0.282 | 0.312 |
| | 10% | 0.182 | 0.011 | 0.166 | 0.177 | 0.183 | 0.186 | 0.197 |
| GRASP | 5% | 0.281 | 0.008 | 0.270 | 0.277 | 0.280 | 0.287 | 0.291 |
| | 7% | 0.244 | 0.017 | 0.226 | 0.233 | 0.237 | 0.256 | 0.266 |
| | 10% | 0.205 | 0.029 | 0.168 | 0.2 | 0.2 | 0.210 | 0.248 |
| Búsqueda Tabú | 5% | 0.295 | 0.014 | 0.278 | 0.287 | 0.291 | 0.306 | 0.313 |
| | 7% | 0.274 | 0.019 | 0.252 | 0.261 | 0.275 | 0.278 | 0.303 |
| | 10% | 0.235 | 0.007 | 0.225 | 0.234 | 0.234 | 0.239 | 0.245 |

Tabla 8.49: Desempeño metodologías en análisis de sensibilidad según porcentaje máximo de cortafuego

## Desempeño análisis de sensibilidad según tiempo máximo de ejecución

| | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| GRASP | 277 | 206 | 195 | 247 | 293 |
| Búsqueda Tabú | 52 | 67 | 69 | 57 | 60 |
| Algoritmo Genético | 57 | 54 | 47 | 47 | 58 |

Tabla 8.50: Número de iteraciones algoritmos en prueba de sensibilidad temporal

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 2000 | 0.302 | 0.278 | 0.319 | 0.332 | 0.317 |
| 4000 | 0.318 | 0.302 | 0.29 | 0.309 | 0.283 |
| 6000 | 0.32 | 0.288 | 0.289 | 0.277 | 0.275 |
| 8000 | 0.297 | 0.273 | 0.306 | 0.285 | 0.254 |
| 10000 | 0.271 | 0.289 | 0.305 | 0.273 | 0.246 |
| 12000 | 0.309 | 0.261 | 0.288 | 0.309 | 0.264 |
| 14000 | 0.289 | 0.286 | 0.289 | 0.289 | 0.204 |
| 16000 | 0.258 | 0.278 | 0.279 | 0.265 | 0.254 |
| 18000 | 0.234 | 0.274 | 0.266 | 0.276 | 0.226 |
| 20000 | 0.227 | 0.256 | 0.264 | 0.265 | 0.214 |
| 21600 | 0.224 | 0.245 | 0.299 | 0.266 | 0.249 |

Tabla 8.51: Desempeño Algoritmo Genético en el tiempo en prueba de sensibilidad temporal

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 2000 | 0.289 | 0.25 | 0.276 | 0.31 | 0.268 |
| 4000 | 0.275 | 0.25 | 0.276 | 0.264 | 0.268 |
| 6000 | 0.25 | 0.25 | 0.253 | 0.264 | 0.268 |
| 8000 | 0.179 | 0.25 | 0.253 | 0.264 | 0.268 |
| 10000 | 0.179 | 0.25 | 0.242 | 0.264 | 0.268 |
| 12000 | 0.179 | 0.25 | 0.242 | 0.264 | 0.268 |
| 14000 | 0.179 | 0.25 | 0.242 | 0.264 | 0.268 |
| 16000 | 0.179 | 0.241 | 0.242 | 0.264 | 0.268 |
| 18000 | 0.179 | 0.241 | 0.242 | 0.264 | 0.268 |
| 20000 | 0.179 | 0.241 | 0.242 | 0.264 | 0.268 |
| 21600 | 0.179 | 0.241 | 0.242 | 0.264 | 0.268 |

Tabla 8.52: Desempeño Búsqueda Tabú en el tiempo en prueba de sensibilidad temporal

| Tiempo (s) | Prueba 1 | Prueba 2 | Prueba 3 | Prueba 4 | Prueba 5 |
|---|---|---|---|---|---|
| 2000 | 0.2 | 0.217 | 0.248 | 0.228 | 0.202 |
| 4000 | 0.2 | 0.203 | 0.098 | 0.136 | 0.199 |
| 6000 | 0.2 | 0.203 | 0.098 | 0.136 | 0.199 |
| 8000 | 0.2 | 0.203 | 0.098 | 0.136 | 0.142 |
| 10000 | 0.2 | 0.2 | 0.098 | 0.136 | 0.142 |
| 12000 | 0.2 | 0.195 | 0.098 | 0.136 | 0.142 |
| 14000 | 0.2 | 0.193 | 0.098 | 0.136 | 0.142 |
| 16000 | 0.2 | 0.193 | 0.098 | 0.136 | 0.142 |
| 18000 | 0.19 | 0.193 | 0.098 | 0.136 | 0.142 |
| 20000 | 0.19 | 0.193 | 0.098 | 0.136 | 0.142 |
| 21600 | 0.19 | 0.193 | 0.098 | 0.136 | 0.142 |

Tabla 8.53: Desempeño GRASP en el tiempo en prueba de sensibilidad temporal