Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

TESIS FIN DE MÁSTER

ANÁLISIS DE LA EFICIENCIA DE LOS SISTEMAS DE COLONIAS DE HORMIGAS PARA PROBLEMAS DE ACCESIBILIDAD

Alumno: Víctor Sacristán Robles
Directores: Antonio Jiménez Martín
Alfonso Mateos Caballero
Fecha: 17 de Julio de 2015

Agradecimientos

Me gustaría comenzar dando las gracias a aquellos que han hecho posible la realización de esta Tesis Fin de Máster. En primer lugar, y de manera más importante, quiero agradecer a Antonio Jiménez y Alfonso Mateos, directores de este proyecto, quienes han marcado la dirección de esta investigación y cuya guía y sugerencias han facilitado el desarrollo de la tesis. Destacar de ellos su total disponibilidad y dedicación a este proyecto. En segundo lugar, agradecer también al Departamento de Inteligencia Artificial de la Universidad Politécnica de Madrid y al claustro de profesores que lo forma por la calidad de las clases impartidas, y en especial, a Javier Bajo, su coordinador, por su total disponibilidad siempre para resolver los problemas particulares de cada alumno.

De forma más personal, quiero terminar agradeciendo a las personas más importantes de mi vida, a mi familia, en especial a mis padres, que siempre me han apoyado de manera incondicional, que me han brindado todas las oportunidades posibles, que han confiado en mí y que han aconsejado y respetado todas mis decisiones. Por último, agradecer a mi novia, Iti, por escucharme una y otra vez hablar sobre este trabajo, y por todas las demás cosas que no quiero especificar y que son siempre al final las más importantes.

A todos ellos, Gracias.

Resumen

A pesar de los avances en materia de predicción, los desastres naturales siguen teniendo consecuencias devastadoras. Entre los principales problemas a los que se enfrentan los equipos de ayuda y rescate después de un desastre natural o provocado por el hombre se encuentra la planificación de las tareas de reparación de carreteras para conseguir la máxima ventaja de los limitados recursos económicos y humanos.

En la presente Tesis Fin de Máster se intenta dar solución al problema de la accesibilidad, es decir, maximizar el número de supervivientes que consiguen alcanzar el centro regional más cercano en un tiempo mínimo mediante la planificación de qué carreteras rurales deberían ser reparadas dados unos recursos económicos y humanos limitados. Como se puede observar, es un problema combinatorio ya que el número de planes de reparación y conexiones entre las ciudades y los centros regionales crece de forma exponencial con el tamaño del problema.

Para la resolución del problema se comienza analizando una adaptación básica de los sistemas de colonias de hormigas propuesta por otro autor y se proponen múltiples mejoras sobre la misma. Posteriormente, se propone una nueva adaptación más avanzada de los sistemas de colonias de hormiga al problema, el *ACS con doble hormiga*. Este sistema hace uso de dos tipos distintos de hormigas, la exploradora y la trabajadora, para resolver simultáneamente el problema de encontrar los caminos más rápidos desde cada ciudad a su centro regional más cercano (exploradora), y el de obtener el plan óptimo de reparación que maximice la accesibilidad de la red (trabajadora).

El algoritmo propuesto se ilustra por medio de un ejemplo de gran tamaño que simula el desastre natural ocurrido en Haití, y su rendimiento es comparado con la combinación de dos metaheurísticas, GRASP y VNS.

Abstract

In spite of the advances in forecasting, natural disaster continue to ocasionate devastating consequences. One of the main problems relief teams face after a natural or man-made disaster is how to plan rural road repair work to take maximum advantage of the limited available financial and human resources.

In this Master´s Final Project we account for the accesability issue, that is, to maximize the number of survivors that reach the nearest regional center in a minimum time by planning whic rural roads should be repaired given the limited financial and human resources. This is a combinatorial problem since the number of possible repairing solutions and connections between cities and regional centers grows exponentially with the size of the problem.

In order to solve the problem, we analyze the basic ant colony system adaptation proposed by another author and point out multiple improvements on it. Then, we propose a novel and more advance adaptation of the ant colony systems to the problem, the *double-ant ACS*. This system makes use of two different type of ants, the explorer and the worker, to simultaneously solve the problem of finding the shorthest paths from each city to their nearest regional center (explorer), and the problem of identifying the optimal repairing plan that maximize the network accesability (worker).

The proposed algorithm is illustrated by means of a big size example that simulates the natural disaster occurred in Haiti, and its performance is compared with a combination of two metaheuristics, GRASP and VNS.

Índice general

Capítulo 1

Introducción

A pesar de los avances en medidas de predicción, monitorización y seguridad frente a desastres naturales como terremotos, inundaciones, tsunamis, tornados, erupciones volcánicas o huracanes, cada vez que uno tiene lugar, las consecuencias siguen siendo devastadoras, causando un fuerte impacto tanto en vidas humanas, como en la economía y el entorno.

Dentro de los grandes daños causados, los destrozos en las líneas de comunicación e infraestructuras de transporte merecen especial atención debido a que pueden obstruir las labores de rescate y demás actividades de salvamento, incrementando así la posible pérdida de vidas humanas. Esta situación se hace incluso más evidente en zonas en desarrollo, como ha sido el caso de Haiti o Nepal más recientemente, donde las infraestructuras tanto de transporte como de comunicaciones son más vulnerables y sensibles al impacto de desastres naturales.

En este Trabajo Fin de Máster nos concentraremos en la infraestructura de transporte, específicamente en el plan de reparación de carreteras rurales en países en desarrollo tras la ocurrencia de un desastre natural o provocado por el hombre con similares consecuencias.

Las actividades de gestión de desastres se pueden clasificar en cinco fases genéricas (UN *international strategy for disaster reduction*): predicción, alerta, socorro de emergencia, rehabilitación y reconstrucción. Las últimas tres fases están generalmente asociadas con el esfuerzo posterior al desastre y engloban las actividades de respuesta y recuperación. Las actividades de respuesta durante la fase de socorro tienen el fin de proporcionar ayuda durante o inmediatamente después de un desastre para garantizar la preservación de vida y de las necesidades básicas de subsistencia de las víctimas.

En las fases de rehabilitación y reconstrucción se toman las decisiones y se llevan acabo las acciones necesarias después de un desastre natural para restaurar y mejorar las condi-

ciones de vida y la afectación de la comunidad. Estas fases deben también complementar la fase de socorro, garantizando la estabilidad de las infraestructuras de socorro y de ayuda, así como el acceso a las mismas.

Uno de los principales problemas a los que se enfrentan los equipos de socorro después de un desastre natural es la forma de planificar la reparación de caminos rurales para aprovechar al máximo los recursos financieros y humanos disponibles, que son limitados. En concreto, en esta Tesis Fin de Máster estudiaremos el problema de la *accesibilidad*. La *accesibilidad* fue definida por Donnges [16] como el grado de dificultad que las personas o comunidades sufren para acceder a las locaciones que garantizan sus necesidades económicas y sociales básicas.

En este trabajo, modelaremos la *accesibilidad* de un pueblo rural como el tiempo mínimo que se tarda en alcanzar el centro regional (centro de actividad económica, social y ayuda humanitaria) más cercano, siendo la *accesibilidad* máxima cuando el tiempo de viaje al centro regional es mínimo. Nuestro objetivo, por tanto, será maximizar la *accesibilidad* de la toda la red, mediante la minimización de los tiempos de viaje desde cada ciudad rural a su centro regional más cercano, teniendo en consideración el número de habitantes y dando prioridad a la conexión del mayor número de personas.

Este sería un problema muy sencillo si los recursos económicos y humanos disponibles fueran infinitos, no obstante, puesto que los recursos para la reparación de carreteras son limitados nos enfrentamos a un problema de optimización matemática complejo. El problema es además *combinatorio*, puesto que el número de conexiones entre los nodos (ciudades y centros regionales) y por consiguiente, el número de posibles caminos hacia los centros regionales, crece de manera exponencial con el tamaño del problema.

Dada la naturaleza combinatoria del problema de optimización que se quiere resolver, se descarta cualquier tipo de método de resolución exacto de programación lineal (como el algoritmo de ramificación y acotación, o el método de los planos de corte), debido a su poca eficiencia computacional y a la necesidad de que los equipos de planificación actúen con la mayor brevedad posible.

Para la resolución del problema de manera eficiente se propone el uso de metaheurísticas. Estos métodos no aseguran la solución óptima, pero sí consiguen alcanzar soluciones muy próximas a la óptima en un tiempo y con un uso de recursos computacionales mucho menor que el de métodos exactos. Esto es posible gracias a que los algoritmos meteheurísticos analizan solamente un subconjunto de las posibles soluciones, elegido de manera inteligente. En concreto, se proponen los *algoritmos de colonias de hormigas* como metaheurística, debido a su gran uso y eficiencia demostrada en problemas asociados a redes, en este caso, una red de transporte.

La estructura de esta Tesis Fin de Máster (TFM) tiene la siguiente forma: En el Capítulo 2, definiremos los objetivos marcados para este trabajo. Describiremos el problema de la *accesibilidad* e introduciremos la modelización matemática del mismo en el Capítulo 3. Seguidamente, en el Capítulo 4, se hará un estudio completo de las distintas metaheurísticas existentes y se llevará a cabo un estudio del arte de la aplicación de las mismas al problema tratado.

Más detalladamente, en el Capítulo 5, describiremos los algoritmos de optimización con hormigas, especialmente la versión *Ant Conlony System* (ACS) utilizada en este trabajo.

El groso del estudio realizado en esta TFM se presentará en el Capítulo 6, donde comenzaremos describiendo el modelo de ACS utilizado como punto de partida. Posteriormente, se propondrán múltiples mejoras sobre el mismo, y finalmente, se presentará el modelo original *ACS con doble hormiga* propuesto para la resolución del problema tratado.

En el Capítulo 7, se utilizará como ejemplo de aplicación un problema que simula el desastre natural ocurrido en Haití y se compararán los resultados de nuestro algoritmo con los obtenidos por una combinación de GRASP y VNS. Por último, en el Capítulo 8, expondremos nuestras conclusiones y plantearemos posibles líneas de trabajo futuro.

Capítulo 2

Objetivos

La presente Tesis Fin de Máster (TFM) tiene como **objetivo principal** la aplicación de manera eficiente de los sistemas de colonias de hormigas (ACS) al problema de *accesibilidad* considerado. Partiendo del trabajo realizado por Muñoz [48] en su propia TFM, donde propone un adaptación inicial y básica de ACS al problema, se pretende mejorar e innovar sobre esa adaptación inicial para crear un modelo más avanzado, eficaz y eficiente que permita resolver el problema.

Para alcanzar el objetivo principal anteriormente descrito es necesario conseguir una serie de subobjetivos que se describen a continuación.

El primero de ellos es **analizar el problema de la accesibilidad**, es decir, hacer un estudio del estado del arte referente al problema para entender en qué consiste el mismo, cuáles han sido las técnicas de resolución utilizadas por otros autores y cómo han aplicado y adaptado estas técnicas al problema considerado.

Seguidamente, es necesaria la **comprensión y modelización matemática** del problema. En este punto, no basta con conocer el problema de la *accesibilidad*, se busca un entendimiento absoluto del mismo y su modelización. Solo así, mediante la comprensión de los entresijos del problema, será posible el desarrollo de un algoritmo original más eficiente que lo que hoy en día podemos encontrar en la literatura.

Estudiar el **uso de metaheurísticas**, especialmente el conjunto de algoritmos de optimización con colonias de hormigas, ACO. Tras haber analizado el problema de la *accesibilidad* y haber comprendido su modelización matemática, se pretende un estudio de las diferentes metaheurísticas existentes para su resolución. Nos concentraremos especialmente en los algoritmos de hormigas, y así, analizaremos las distintas versiones existentes para la comprensión de sus ventajas e inconvenientes. También será necesario observar cómo los distintos autores aplican estas técnicas en la resolución de problemas.

Principal atención requiere el análisis del método de solución propuesto por Muñoz [48]. Este **modelo inicial** es nuestro punto de partida, por ello, va a ser necesario la implementación del mismo, analizar su funcionamiento y probarlo en varias instancias. Solo así, identificaremos las posibles mejoras sobre el mismo y obtendremos la inspiración necesaria para el desarrollo de una solución original y más eficiente.

Finalmente, tras entender el problema de la *accesibilidad* y su modelización, llevaremos a cabo un estudio del estado del arte y analizaremos el algoritmo base de partida. Se pretende probar y **comparar** nuestra solución propuesta en instancias grandes. Para ello, se utilizará una instancia que simula el desastre natural ocurrido en Haití, y nuestra solución se comparará con otro algoritmo que combina las metaheurísticas GRASP y VNS.

Para realizar satisfactoriamente este último subobjetivo, se debe hacer un **ajuste de parámetros**. Como toda metaheurística, la familia de algoritmos ACO tienen una serie de parámetros donde el rendimiento y eficacia de esta están subordinados a los valores asignados a los parámetros. No existe una regla de oro para el ajuste paramétrico, sino que se suele realizar mediante prueba y error. Una correcta configuración de los parámetros será clave en el correcto funcionamiento del algoritmo propuesto.

Capítulo 3

Descripción de problema y Modelización matemática

El problema a resolver en esta Tesis Fin de Máster se puede describir mediante una red compuesta de un conjunto de nodos, que representan los centros de población y los cruces de carreteras, y la correspondiente red de carreteras que conecta esos nodos. Existen tres clases de nodos: los centros regionales (una minoría de los nodos que representan las ciudades más desarrolladas y habitadas), ciudades o pueblos más pequeños, y un conjunto de puntos de cruce entre carreteras. Como se ha explicado previamente, las aristas del grafo representan las carreteras que conectan esos nodos. Tendremos dos clases de carreteras, aquellas que siguen operativas y las que no están operativas, pues han sido dañadas fruto del desastre natural. En la Figura 3.1 se muestra un ejemplo de una posible red. El tamaño de los círculos alrededor de los centros de población representan su importancia relativa

Debido a la restricción de la disponibilidad de recursos monetarios y humanos suficientes para la reparación de toda la red, se deberá buscar el plan óptimo de reparación que maximice la *accesibilidad* de la red respetando los recursos disponibles. Esta situación da lugar a un problema complejo de optimización con una función objetivo no lineal. En este capítulo, se muestra la formulación matemática basada en el trabajo de [4] y [22] del problema considerado.

Sea $G = (N, \varepsilon)$ un grafo no dirigido, donde $N = \{N_1 \cup N_2 \cup N_3\}$ es un conjunto de nodos y ε es un conjunto de aristas. N está dividido en tres subconjuntos: N_1, centros regionales; N_2, ciudades; N_3, intersecciones. Las aristas en ε representan las diferentes carreteras de la red. Cada una tiene una variable binaria asociada l_e (1 si la carretera está operativa y 0 en caso contrario). En relación con la variable l_e, existe el subconjunto $\varepsilon_r \subset \varepsilon$ compuesto por

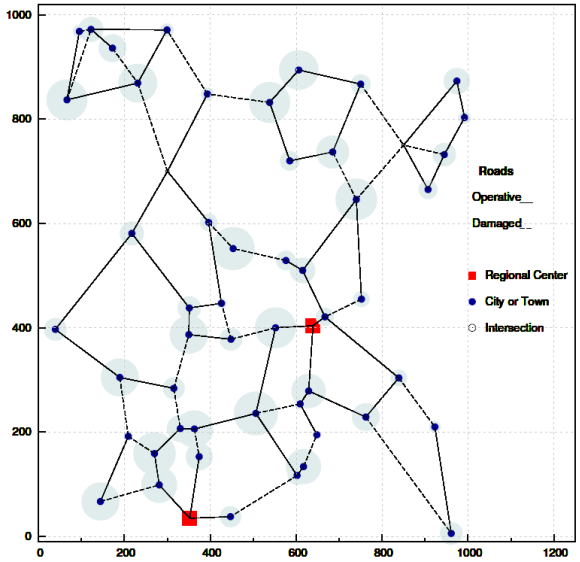

Figura 3.1: Ejemplo de una red de carreteras [22].

todas aquellas carreteras que no están operativas y podrían ser reparadas. Las carreteras que forman parte de este conjunto siempre estarán inicialmente inoperativas y el valor inicial de la variable l_e asociada será cero.

Como se ha comentado previamente, para la reparación de las carreteras existen una serie de recursos limitados. Estos son un presupuesto económico B y un presupuesto de mano de obra H medido en horas de trabajo por persona. Por tanto, tenemos asociados a cada carretera $e \in \varepsilon$ el coste monetario de reparación c_e y el coste de mano de obra m_e.

El objetivo del problema es maximizar la *accesibilidad* de la red mediante la minimización de la suma ponderada del tiempo de viaje desde cada nodo $i \in N_2$ hasta el centro regional más cercano en N_1. Por supuesto, la *accesibilidad* de cada nodo i, es decir, el tiempo de travesía mínimo hacia el centro de regional más cercano, depende de si las carreteras utilizadas en el camino están operativas o no. El tiempo en atravesar una de las aristas es t_e cuando la carretera está operativa y $t_e + M_e$ cuando no lo está. M_e es simplemente un factor de penalización añadido al tiempo de travesía de la carretera cuando esta está operativa y representa el uso de otro tipo de transporte para cruzar la carretera no operativa, por ejemplo, vehículos de tracción animal.

En un país o región no todas las ciudades son igual de importantes. En los países desarrollados el factor de importancia está relacionado con la actividad económica de la ciudad o con la posición estratégica, comercial o militar. Para el problema que nos ocupa, el contexto es más parecido al de un país en desarrollo y, teniendo en cuenta que la resolución del problema tiene fines de salvamento, daremos más importancia al volumen

de habitantes. Así pues, se define un peso w_i para cada nodo $i \in N_2$ para representar la importancia del mismo, siendo este peso w_i una función del número de habitantes de la ciudad asociada a ese nodo i.

A continuación, se muestra la formulación matemática del problema, en primer lugar la función objetivo a optimizar y, seguidamente, el conjunto de restricciones.

La función objetivo se define como:

$$f(x) = min \sum_{i \in \mathcal{N}_2} \left(w_i \; min_{j \in \mathcal{N}_1} \left\{ \sum_{e \in \varepsilon} d_e y_e^{ij} \right\} \right), \tag{3.1}$$

donde

$$d_e = \begin{cases} t_e + (1 - x_e) M_e & \forall e \in \varepsilon_r \\ t_e & \forall e \in \varepsilon \setminus \varepsilon_r \end{cases},$$

y donde la variable x_e se activa ($x_e = 1$) cuando la carretera $e \in \varepsilon_r$ está reparada y se desactiva ($x_e = 0$) cuando no lo está. La variable y_e^{ij} se activa cuando la carretera $e \in \varepsilon_r$ se utiliza en el camino de $i \in N_2$ a $j \in N_1$.

La función objetivo minimiza la suma ponderada de los caminos mínimos para cualquier ciudad $i \in N_2$ al centro regional más cercano $j \in N_1$. El sumatorio interior calcula la longitud en unidades de tiempo de los caminos desde cualquier ciudad $i \in N_2$ a todos los centros regionales, siendo la función de minimización que lo engloba quien selecciona el camino al centro regional más cercano. El sumatorio externo calcula la suma ponderada de los caminos de cada ciudad $i \in N_2$ a su centro regional más cercano, ponderando por la importancia de la ciudad w_i, que es función del número de habitantes. Mediante la minimización de este sumatorio y, por tanto, del tiempo de viaje desde todas las ciudades o pueblos rurales a sus centros regionales más cercanos, se garantiza la máxima *accesibilidad*.

Las restricciones del problema son las siguientes

$$\sum_{e \in \varepsilon(i)} y_e^{ij} = 1 \quad \forall i \in \mathcal{N}_2, \forall j \in \mathcal{N}_1,$$
$$\sum_{e \in \varepsilon(j)} y_e^{ij} = 1 \quad \forall i \in \mathcal{N}_2, \forall j \in \mathcal{N}_1.$$

El primer conjunto de restricciones aseguran, respectivamente, que solo existe una carretera que parte de i en el camino de i a j, y que solo existe una carretera que llega a j en

el camino de i a j, es decir, esta restricción prohíbe que existan dos caminos que partan del mismo nodo y lleguen al mismo destino.

La variable b_k^{ij} se activa cuando el nodo $k \in N$ se visita en el camino de $i \in N_2$ a $j \in N_1$. Entonces, la siguiente restricción se asegura que el camino entre i y j esté conectado

$$\sum_{e \in \varepsilon(k)} y_e^{ij} = 2b_k^{ij}, \ \forall k \in \mathcal{N} \setminus \{i,j\}, \forall i \in \mathcal{N}_2, \forall j \in \mathcal{N}_1.$$

Una de las restricciones más importantes que se definen en el problema tiene que ver con la disponibilidad de recursos limitados. Tanto el presupuesto económico como el presupuesto de mano de obra utilizado en el proceso de reconstrucción de carretera debe ser siempre menor o igual al presupuesto económico y humano inicial, respectivamente.

$$\sum_{e \in \varepsilon_r} c_e x_e \leq \mathcal{B},$$
$$\sum_{e \in \varepsilon_r} m_e x_e \leq \mathcal{H}.$$

Finalmente, se deben incluir la restricciones relativas al dominio de las variables de decisión consideradas, siendo todas ellas binarias.

$$\begin{aligned} x_e &\in \{0,1\} \quad \forall e \in \varepsilon_r, \\ y_e^{ij} &\in \{0,1\} \quad \forall e \in \varepsilon_r, \forall i \in \mathcal{N}_2, \forall j \in \mathcal{N}_1, \\ b_k^{ij} &\in \{0,1\} \quad \forall k \in \mathcal{N} \setminus \{i,j\}, \forall i \in \mathcal{N}_2, \forall j \in \mathcal{N}_1. \end{aligned}$$

Capítulo 4

Optimización

La optimización es una disciplina presente en todos los campos de la ciencia e incluso en algunos ámbitos de la vida, desde los procesos de ingeniería más complicados, la planificación de los procesos de negocio o el enrutamiento de paquetes en redes de transporte, hasta la planificación de las vacaciones o incluso la optimización de la lista de la compra. El proceso de optimización se define como el proceso que trata de conseguir la mejor solución posible a un problema, a la que denotaremos como óptima.

Desde un punto de vista matemático, un problema de optimización o también llamado de *programación matemática* se puede representar del siguiente modo

$$min_{x \in \mathbb{R}^n} \ f_i(x), (i = 1, 2, ..., M)$$

$$\mathbf{x} = (x_1, x_2, ..., x_n)^T$$

Las funciones $f_i(x)$ son las llamadas funciones objetivo o funciones de coste. Los componentes x_i del vector $\mathbf{x}$ se conocen como variables de decisión y están sujetas a una serie de restricciones $h_j(\mathbf{x}) = 0, (j = 1, 2, ..., J)$ y $g_k(\mathbf{x}) \leq 0, (k = 1, 2, ..., K)$ que constituyen la región factible del espacio de búsqueda X, es decir, $x \in X$.

M es el número de funciones objetivo que deseamos optimizar. En nuestro problema particular tenemos $M = 1$, es decir, es un problema uniobjetivo, ya que solo deseamos optimizar una función objetivo, la *accesibilidad*.

Podría ser interesante modelizar este problema con una función multiobjetivo, en donde las restricciones de presupuesto que tenemos se modelizasen como funciones objetivo. Este tipo de configuración permitiría encontrar un buen término medio entre *accesibilidad* y

utilización de recursos. No obstante, puesto que nuestro fin último es el salvamento y la ayuda humanitaria, consideramos más importante la máximización de la *accesibilidad* y utilizaremos para ello todos los recursos disponibles a nuestro alcance.

4.1. Métodos exactos vs Heurísticas vs Metaheurísticas

En optimización, existe una confrontación entre la necesidad de encontrar la solución óptima y hacerlo de manera rápida minimizando el uso de recursos computacionales. En esta sección mostraremos una clasificación de métodos en referencia a cómo afrontan esta confrontazión.

En el campo de la optimización matemática o programación matemática se han desarrollado múltiples técnicas de resolución que garantizan la solución optima del problema, se conocen como **métodos exactos**. Entre los métodos más conocidos destacan el *método del simplex*, para problemas lineales continuos; el *algoritmo de ramificación y acotación* [42] o el de los *planos de corte* [39], para problemas lineales enteros; o los *métodos del gradiente* y *Quasi-Newton* [15], para problemas no lineales.

Todos los métodos exactos citados anteriormente coinciden en que son computacionalmente muy costosos. En problemas combinatorios como el que se presenta en esta TFM, donde el número de soluciones factibles crece exponencialmente con el tamaño del problema, los métodos exactos suelen ser inviables en instancias grandes, ya sea por limitación de tiempo o por falta de memoria.

Con el objetivo de solventar este problema surgen los métodos aproximados, como las **heurísticas**. El término heurística se puede definir como el procedimiento que trata de aportar una solución a un problema de manera eficiente, es decir, intenta encontrar una solución de calidad o próxima a la óptima con un coste computacional razonable. Sin embargo, las heurísticas no son capaces de garantizar la optimalidad de la solución ni de establecer cuán próxima de la solución óptima está la solución encontrada. Las heurísticas se suelen definir para problema específicos, y su funcionamiento fuera de estos problemas no es el adecuado. Esto se debe a que las heurísticas resuelven el problema de manera inteligente usando el conocimiento disponible del mismo. Debido a esta limitación, el principal problema de los métodos heurísticos es su incapacidad para escapar de óptimos locales.

Para superar este problema, aparece una familia de algoritmos más inteligentes conocidos como **metaheurísticas**. Las metaheurísticas, que como su propio nombre indica son heurísticas de alto nivel, son algoritmos potentes normalmente aplicables a un gran número de problemas y con un alta capacidad de escapar de óptimos locales. Se les considera

una estrategia maestra iterativa que guía y optimiza el comportamiento de las heurísticas subordinadas mediante la fusión de técnicas de exploración y explotación del espacio de soluciones, es decir, de manera inteligente mejoran los procedimientos heurísticos muy generales dotándoles de un alto rendimiento y polivalencia.

La evolución de las metaheurísticas y su actividad como área de investigación han tenido un comportamiento exponencial en las últimas décadas, permitiendo resolver problemas que tiempo atrás se consideraban inabordables.

4.2. Clasificación de metaheurísticas y descripción

4.2.1. Metaheurísticas trayectoriales

Las metaheurísticas trayectoriales pueden ser entendidas como un enfoque inteligente de los algoritmos de búsqueda local. Comienzan con una única solución inicial normalmente aleatoria que van mejorando iteración tras iteración durante el proceso de optimización. Su nombre proviene del hecho de que la solución tratada va moviéndose por el espacio de soluciones como si dibujase una trayectoria. Entre estos métodos, destacan el *recocido simulado*, la *búsqueda tabú*, el método *GRASP* o la *búsqueda en entornos variables*, los cuales describiremos más en detalle a continuación.

4.2.1.1. Recocido simulado

El método del Recocido o Enfriamiento Simulado (*Simulated annealing*) está inspirado en la industria metalúrgica. El recocido es una técnica que consiste en calentar el material a muy altas temperaturas y enfriar de manera muy lenta y progresiva para conseguir un estado de mínima entropía. Este estado de mínima entropía da lugar a sólidos con mejores propiedades.

El algoritmo de Recocido o Enfriamiento Simulado fue propuesto inicialmente por Kirkpatrick et al.[40] y Cerny [6] para resolver el famoso problema del viajante. Desde su propuesta se ha demostrado la capacidad del algoritmo para obtener buenos resultados en distintos problemas de optimización.

Apoyándonos en la base teórica de la técnica real, la aplicación del recocido simulado simplemente consiste en sustituir la energía mínima de la técnica real por nuestra función objetivo que queremos minimizar en el algoritmo simulado. La temperatura será simplemente un parámetro, T, que iremos disminuyendo a lo largo de las iteraciones del proceso de optimización.

Algoritmo 1 Recocido simulado uniobjetivo de minimización

```
Elegir aleatoriamente una solución inicial s
Inicializar la temperatura T
while no se satisface el criterio de parada do
    repeat
        Seleccionar aleatoriamente s' ∈ N(s)
        if f(s') ≤ f(s) then
            s ← s'
        else
            s ← s' con una probabilidad p(T, f(s'), f(s))
        end if
    until se alcanza el "equilibrio termodinámico"
    Decrementar T
end while
return mejor solución
```

El algoritmo comienza generando una solución inicial (aleatoria o obtenida por algún tipo de heurística o información disponible sobre el problema) e inicializando el parámetro de temperatura, T, a un valor alto.

En cada iteración se obtiene una solución, s', seleccionada de forma aleatoria de un entorno $N(s)$ de la solución actual. La aceptación de la nueva solución obtenida depende de los valores en la función objetivo para s, s' y el valor del parámetro T. Si $f(s') \leq f(s)$, entonces s' es aceptada y reemplaza a s. En caso contrario, a pesar de que $f(s') > f(s)$, la aceptación de s' se decide de manera estocástica, donde la probabilidad de aceptación depende de la temperatura:

$$P(T, f(s), f(s')) = exp(-\frac{f(s') - f(s)}{T}).$$

Al principio del proceso la temperatura se ha inicializado a un valor bastante alto, por tanto, la probabilidad de poder aceptar una solución que no mejora nuestra solución actual es también alta. En un principio, esto permite explorar el espacio de soluciones para no quedar atrapado en un óptimo local. Según pasan las iteraciones, el parámetro que controla la temperatura desciende paulativamente, disminuyendo la probabilidad de que se acepte una solución peor a nuestra solución actual. Es decir, el algoritmo deja de explorar y comienza a explotar nuestra solución actual para que se produzca la convergencia hacia la solución óptima.

A partir de esta versión original previamente descrita surgen numerosas mejoras o variantes. Destaca por ejemplo, el hecho de guardar la mejor solución encontrada hasta el momento, con el fin de no perderla en el caso de que se acepte probabilisticamente una solución peor. Es también interesante el trabajo de [13] donde se comienza a adaptar el algoritmo para que funcione en dominios continuos y no solo en dominios discretos. Entre las variante más conocidas del algoritmo se encuentran la variante del *enfriamiento microcanónico* (*Microcanonical annealing*) [14], el *método del umbral de aceptación* (*Threshold accepting method*) [21] o el *método ruidoso* (*Noising method*) [7, 8].

4.2.1.2. Búsqueda tabú

Propuesto por Glover en 1986 [28], el algoritmo de *búsqueda tabú* es un procedimiento heurístico que combina el proceso de búsqueda global con técnicas para evitar caer en óptimos locales o caer en ciclos durante el proceso de búsqueda. La habilidad de no caer en ciclos es su principal característica y se consigue mediante el mantenimiento de lo que se conoce como *lista tabú*. La *lista tabú* (LT) es una estructura de memoria donde se almacenan los últimos movimientos de búsqueda realizados, para evitar la repetición de los mismos. Mediante la prohibición de repetir estos movimientos se consigue escapar de los ciclos.

Algoritmo 2 Búsqueda tabú

Elegir aleatoriamente una solución inicial s del espacio de búsquda
$LT \leftarrow \emptyset$
while no criterio de parada **do**
  Seleccionar la mejor solución $s' \in N(s) \backslash LT$
  $s \leftarrow s'$
  Actualizar LT
end while
return mejor solución

El tamaño de la LT es uno de los parámetros más importantes del algoritmo, que controla la memoria del proceso y, por tanto, el comportamiento del algoritmo de búsqueda. Si el tamaño de la lista es excesivamente grande, la posibilidad de caer en un ciclo es ínfima y la explotación del espacio de búsqueda amplia al prohibir revisitar un grán numero de soluciones; sin embargo, al algoritmo le será difícil realizar el proceso de explotación y converger. Por el contrario, si la lista es demasiado pequeña, el algoritmo corre el riesgo de caer en óptimos locales o en ciclos más grandes que los del tamaño de la lista. El tamaño de

lista elegido debe garantizar al algoritmo un término medio entre exploración, explotación y protección frente a ciclos.

Para garantizar la convergencia del algoritmo hacia la solución óptima, la LT debe complementarse con el *criterio de aspiración*. Este criterio es un conjunto de reglas que permiten al algoritmo, en caso de que se cumplan, obviar las restricciones tabú. El casó mas claro es cuando se permite al algoritmo realizar un movimiento tabú, si este nos conduce a una solución mejor que la encontrada hasta el momento.

4.2.1.3. Método GRASP

Propuesto por Feo y Resende en 1995 [25, 26], el método GRASP (*Greedy randomized adaptive search procedure*) es una metaheurística de múltiple comienzo sin memoria con muy buen comportamieto en problemas combinatorios.

En cada iteración del algoritmo se realizan dos pasos: el paso de construcción y el de búsqueda local. El *paso de construcción* en GRASP es similar a la heurística semivoraz propuesta por Hart y Shogan en el año 1987 [34], donde se selecciona una solución factible usando una heurística voraz aleatoria. Seguidamente, tras haber obtenido esta solución inicial, se aplica sobre ella un *proceso de búsqueda local*. Se realizan estos pasos de manera repetitiva durante un número definido de iteraciones y, finalmente, se devuelve la mejor solución encontrada.

Algoritmo 3 Método GRASP

repeat
  Construir una solución factible utilizando una heurística voraz aleatoria
  Aplicar una búsqueda local empezando desde la solución construída
until criterio de parada
return mejor solución

Para el proceso de construcción, dado un problema, se debe definir la solución como un conjunto de elementos, por ejemplo, un conjunto de carreteras a reparar. Una vez definida, se construye de forma iterativa la solución candidata, comenzando con el conjunto vacío y seleccionando de manera probabilística los elementos que al añadirlos a la solución parcial mejoren la función objetivo. La lista de los posibles elementos a añadir se denomina *lista de candidatos restringida* y la selección aleatoria de uno de ellos representa el aspecto probabilístico del algoritmo.

Hay dos enfoques distintos para construir esta lista: por el número de elementos o por su calidad. En el primer caso, la lista se puebla con los p mejores candidatos, donde p es un parámetro del algoritmo. En el segundo caso, se puebla con los candidatos con un valor mayor o igual a $c^{min} + \alpha(c^{max} - c^{min})$, donde c^{min} y c^{max} son el mejor y el peor candidato y α es un parámetro en el intervalo [0,1] que define el compromiso que tendrá el algoritmo entre exploración y explotación.

El parámetro α influye de manera notable en el rendimiento del algoritmo GRASP, por lo que ha habido múltiples propuestas para tratar su ajuste [61]. Se puede inicializar a un valor constante, cambiarlo automáticamente durante el proceso de búsqueda o cambiarlo estocásticamente según alguna distribución de probabilidad.

Una de las ventajas características de GRASP es la facilidad de combinarlo con otras estrategias de búsqueda o metaheurísticas. Esto se consigue reemplazando el algoritmo de búsqueda local por otro tipo de metaheurística, como la búsqueda tabú, el recocido simulado, la búsqueda en entornos variables, o la búsqueda local iterativa, entre otras. También se ha estudiado la adaptación del método a problemas con dominio continuo [36].

4.2.1.4. Búsqueda en entornos variables

En 1997, Hansen y Mladenovic [46] propusieron la *búsqueda en entornos variables* (*Variable neighborhood search*). Esta metaheurística es una estrategia de búsqueda que utiliza un entorno dinámico que va cambiando para escapar de mínimos locales.

Al inicializar, se definen el conjunto de estructuras de entorno, distintas entre sí, que van a generar las soluciones vecinas a partir de una dada. Normalmente, estas estructuras se ordenan aleatoriamente en una lista numerada. A continuación, se genera la solución inicial y comienza el ciclo principal del algoritmo, formado por las fases de *temblor*, *búsqueda local* y *movimiento*.

En la fase de temblor, se genera aleatoriamente una solución s' del entorno $n = 1$ de la solución actual s. Seguidamente, a partir de la nueva solución s' se lanza un proceso de búsqueda local que obtiene una nueva solución s''. Si la nueva solución s'' ha mejorado a la solución inicial s, entonces s'' se convierte en nuestra solución actual y comienza un nuevo ciclo desde $n = 1$ con esta solución. En caso contrario, s'' se descarta, la estructura de entorno se mueve a $n + 1$ y se realizan de nuevo las tres fases previamente descritas.

La eficiencia del método de *búsqueda en entornos variables* depende notablemente de la elección de los entornos. Se considera que para un correcto funcionamiento, los entornos

Algoritmo 4 Búsqueda en entornos variables

```
Seleccionar un conjunto de estructuras de entornos N_n, n = 1, ..., n_max
Elegir aleatoriamente una solución inicial s en el espacio de búsqueda
while no se alcanza el criterio de parada do
    n ← 1
    while n < n_max do
        Temblor: seleccionar aleatoriamente una solución s' en los n entornos N_n(s) de s
        Lanzar búsqueda local empezando desde s' para generar s''
        if s'' es mejor que s then
            s ← s''
            n ← 1
        else
            n ← n + 1
        end if
    end while
end while
return mejor solución
```

elegidos deben ser complementarios. Esto significa que un óptimo local para el entorno N_i no es a su vez un óptimo local para el entorno N_j.

Como en todas las metaheurísticas presentadas hasta el momento, han surgido multiples variaciones a partir del algoritmo original. Cabe destacar la técnica del Descenso en entorno variable (*Variable neighborhood descent*) [32], donde a diferencia del método tradicional el cambio de entorno se hace de forma determinista.

4.2.2. Metaheurísticas evolutivas

Las metaheurísticas evolutivas son un conjunto de algoritmos poblacionales de optimización inspirados en la *teoría de la evolución* del NeoDarwinismo, que hoy en día forman toda una rama de la Inteligencia Artificial conocida como *Computación evolutiva*. El NeoDarwinismo postula que las especies de seres vivos que consiguen sobrevivir son aquellas más adaptadas al entorno. Los algoritmos evolutivos simulan la evolución, mediante procesos de selección y reproducción aplicados a la población de soluciones iniciales para producir mejores soluciones. Los algoritmos con más renombre de la computación evolutiva son las *estrategias evolutivas*, los *algoritmos genéticos* y la *programación genética*.

Cada iteración del algoritmo simula una *generación*, donde a una *población* de soluciones formada por distintas soluciones denominadas *individuos* se les aplica un operador de reproducción. La reproducción combina los *genes* o características de los individuos

Algoritmo 5 Algoritmo de computación evolutiva genérico

```
Inicializar la población con individuos aleatorios
Evaluar cada individuo
repeat
    Seleccionar padres
    Cruzar par de individuos seleccionados
    Mutar la descendencia resultante
    Evaluar los nuevos individuos
    Seleccionar los individuos que pasan a la siguiente generación
until criterio de parada
```

actuales, los *padres*, para producir uno o más nuevos individuos, los *hijos* o *descendencia*. Este operador, se complementa con el operador de *mutación*, que permite la aparición aleatoria de nuevos rasgos en la descendencia. La nueva descendencia es evaluada según su *aptitud* o *fitness*, es decir, se evalúa la calidad de la solución para seleccionar a aquellos individuos que pasarán a la siguiente generación.

4.2.2.1. Estrategias evolutivas

El primer algoritmo evolutivo que se propuso fue el de *estrategias evolutivas*, que surgieron de la mano de Rechenberg en la década de los 60 [52, 53] y fueron desarrolladas más adelante por Schwefel [56]. En comparación con el resto de algoritmos de la misma familia, destacan por su sencillez.

La versión más básica, (1+1)-EE, utilizaba un único padre al que se le aplicaba un operador de mutación del tipo $\vec{x}^{t+1} = \vec{x}^t + N(0, \vec{\sigma})$ para generar un único hijo. Si el hijo era más apto que el padre, entonces sustituía al padre, de lo contrario el hijo era eliminado y el padre volvía a ser mutado.

Fue ya Schwefel quien introdujo el uso de múltiples hijos y padres en las estrategias evolutivas. Lo hizo mediante las versiones $(\mu + \lambda)$-EE y la (μ, λ)-EE, donde μ es el número de padres y λ el número de hijos. En el primer caso, con el fin de mantener constante la población, los μ mejores individuos entre el conjunto de padres e hijos sobreviven y pasan a la siguiente generación. En el segundo caso, todos los padres son eliminados y el total de hijos $\lambda = \mu$ sobreviven.

Al igual que en la versión básica propuesta inicialmente, el operador de mutación en el resto de estrategias evolutivas se realiza mediante la introducción de ruido de valores distribuidos normalmente en los valores originales, es decir, $\vec{x}^{t+1} = \vec{x}^t + N(0, \vec{\sigma})$.

Como es lógico, los parámetros de la distribución de probabilidad normal tienen un papel vital en el rendimiento del algoritmo, por ello, se han propuesto distintos enfoques.

Algoritmo 6 Estrategia evolutiva (μ,λ)

Inicializar la población P, tanto los valores, $P_{problem}$, como las estrategias, $P_{strategy}$
Evaluar la población
$S_{best} \leftarrow best(P)$
repeat
  $children \leftarrow \emptyset$
  for $i = 0$ to λ **do**
    $S_i \leftarrow \emptyset$
    $S_{i_{problem}} \leftarrow mutate(P_{i_{problem}}, P_{i_{strategy}})$
    $S_{i_{strategy}} \leftarrow mutate(P_{i_{strategy}})$
    $children \leftarrow S_i$
  end for
  Evaluar la población
  $S_{best} \leftarrow best(children + S_{best})$
  Seleccionar μ mejores elementos que pasan a la próxima generación
until criterio de parada
return S_{best}

Entre los más clásicos se encuentran *Rechenberg's 1/5 success rule*, *σ-self-adaptation*(σSA) [53] o meta-ES [57]. Recientemente, un estudio propuesto por Hansen et al. [31, 30] con título *Covariance matrix adaptation evolution strategy* (CMA-ES) ha demostrado ser el más eficiente de todos los enfoques y, desde entonces, parece ser el más utilizado tanto para optimizaciones locales como globales.

4.2.2.2. Algoritmos genéticos

A mediados de la década de 1970 John Holland [37] propuso los *Algoritmos Genéticos*. El *algoritmo genético* es, con diferencia, la técnica de computación evolutiva más utilizada. La gran ventaja de los *algoritmos genéticos* reside en que todas las partes principales del algoritmo pueden ser adaptadas a cada problema específico, gracias a la existencia de un sin fin de versiones de cada una de estas. Los algoritmos genéticos difieren unos de otros por la forma en la que representan o codifican la solución, su estrategia de selección, el tipo de operador de cruce y mutación o la estrategia de reemplazo.

Para cada problema se debe elegir la representación de la solución más adecuada, las más comunes son la codificación binaria y la codificación real. El número de técnicas de cruce y mutación existentes en la literatura es abismal. Normalmente, los operadores elegidos dependen de la codificación utilizada.

Algoritmo 7 Algoritmo genético genérico

```
Inicializar la población con individuos aleatorios, P_k
Evaluar la población P_k
repeat
    Seleccionar individuos de P_k
    Cruzar los individuos seleccionados y colocar el resultado en P_{k+1}
    Mutar individuos de P_{k+1} y colocar el resultado en P_{k+1}
    Evaluar la población P_{k+1}
    P_k ← P_{k+1}
    P_{k+1} ← ∅
until criterio de parada
return mejor individuo de P_k
```

El operador de reproducción es el más importante, pues combina los genomas de los distintos padres para generar la nueva descendencia. Entre los operadores de reproducción más tradicionales se encuentran el de punto fijo, el de puntos múltiples o el uniforme.

La estrategia de selección es la que se encarga de elegir a los padres más aptos para la reproducción. Destaca el método de *selección por torneo*, la *selección por ruleta* o el *método del ranking*. Tras haber cruzado a los padres y haber generado la nueva descendencia, se aplica el operador de mutación para modificar aleatoriamente los genomas de algún individuo y dotar al algoritmo de mayor capacidad exploratoria.

Tanto el operador de cruce como el de mutación se aplican de manera estocástica, sin embargo, mientras que el de cruce se aplica con un porcentaje muy alto siempre superior al 70 % y normalmente cercano al 90 %, el operador de mutación es mucho menos frecuente y se aplica con una probabilidad alrededor del 1 %.

Finalmente, mediante la estrategia de reemplazo se seleccionan los individuos, de entre los padres y la nueva descendencia, que poblará la siguiente generación.

Para un correcto funcionamiento del algoritmo, se debe hacer un ajuste de los parámetros de cruce, mutación, elitismo y tamaño de población que garantice un buen equilibrio entre exploración y explotación.

4.2.2.3. Programación genética

La *programación genética* fue introducida por Koza [41] a principios de la década de 1990. Este método no tiene la intención de buscar soluciones sino programas enteros que resuelvan el problema.

Con la programación genética se pretende resolver una de las grandes preguntas de la Inteligencia Artificial: ¿cómo conseguir que un programa realice una acción por sí solo sin

que se le haya dicho o programado específicamente para llevarla a cabo? La estrategia es muy parecida a la de los algoritmos genéticos salvo por pequeñas salvedades. La diferencia más importante frente a los algoritmos genéticos es la longitud de la cadena que representa cada individuo. Mientras que en los algoritmos genéticos la longitud de las cadenas es fija, en los programas genéticos los individuos se codifican mediante *árboles de sintaxis abstracta* (AST) de longitud variable.

Al igual que en los algoritmos genéticos, en programación genética se comienza con una población inicial compuesta por los programas de ordenador elegidos de manera aleatoria, pero guiado por algún método como *grow*, *full*, *ramped half-and-half* [41] o *random branch* [9]. En el caso de la programación genética los estudios existentes revelan que la inicialización de los primeros árboles es sumamente importante en la convergencia del algoritmo hacia el óptimo. Posteriormente, cada programa se evalúa en función de cómo de bien se comporta en el ambiente del problema y en función de esta nota de aptitud se seleccionan los individuos para ser cruzados. De nuevo, igual que en los algoritmos genéticos, se aplican los operadores de cruce y mutación.

Una de las particularidades de la programación es que los operadores evolutivos deben controlar la explosión de código, esto es, el crecimiento incontrolado de los árboles producido al cruzar dos árboles distintos.

Entre los múltiples tipos de operadores presentes en la literatura se encuentran el *subtree crossover* y el *subtree mutation*, también introducidos por Koza [41]. Nótese, que debido a la naturaleza del proceso de evolución en programación genética, se ha sugerido añadir un operador más llamado *operador de reproducción*, que simplemente copia determinados individuos a la siguiente generación, mejorándose sustancialmente el proceso de convergencia. Finalmente, se aplica algún método de reemplazo como el *método generacional* o el del *estado constante*, una comparación entre ambos se puede encontrar en [60].

4.2.3. Métodos de inteligencia de enjambres

Los *métodos de inteligencia de enjambre* son un enfoque relativamente nuevo para la resolución de problemas de optimización que se inspiran en el comportamiento colectivo de colonias de insectos u otros grupos de animales. Estos algoritmos bioinspirados están formados por agentes simples que mediante la interacción entre ellos y el entorno consiguen simular un comportamiento inteligente para resolver determinados tipos de problemas.

A diferencia de otro tipo de métodos, la característica de los métodos de inteligencia de enjambre es que la inteligencia no reside en el individuo, sino que surge de la interacción y cooperación entre ellos, una actividad vital en la supervivencia de estas especies en un

entorno continuamente cambiante. Entre los algoritmos más destacados se encuentran los *algoritmos de enjambre de particulas*, que describiremos a continuación, y los *algoritmos de hormigas* que utilizaremos en esta Tesis Fin de Máster, y que estudiaremos al detalle en el siguiente capítulo de la presente memoria.

4.2.3.1. Optimización por enjambre de partículas

En 1995 James Kennedy y Russell Eberhart [23] proponen la optimización mediante *enjambre de partículas* (*Particle swarm optimization*) como un método de optimización global que se inspira en el comportamiento de las bandadas de pájaros.

En el proceso de inicialización, se generan de manera estocástica múltiples partículas repartidas por el espacio de búsqueda, que representan soluciones candidatas al problema. Cada partícula se define mediante su posición en el espacio y un vector de velocidad. Además, tienen la capacidad de recordar su mejor posición hasta el momento.

El enjambre de partículas se estructura mediante una topología global, descrita por las interconexiones entre partículas, que va a influenciar sobre el movimiento y comportamiento del individuo. El conjunto de partículas con las que una partícula i está conectada son los vecinos de i.

Dos de las topologías que han demostrado mejor comportamiento son *gbest*, que consiste en que el mejor vecino del enjambre influye al resto de partículas, y *lbest*, que consiste en un entramado en forma de anillo donde cada individuo está conectado a sus dos miembros adyacentes en el conjunto de la población y son estos dos vecinos los que influyen en la partícula conectada a ellos.

Después de la fase de inicialización, en cada iteración, la partícula i ajusta su posición $\vec{X}_i$ y su velocidad $\vec{V}_i$ en cada dimensión d del espacio de búsqueda basándose en la mejor posición $\vec{P}_i$ que ha encontrado en su vuelo y la mejor posición $\vec{P}_g$ encontrada por las partículas de su entorno topológico:

$$V_{id}(t+1) = V_{id}(t) + C_1\varphi_1(P_{id}(t) - X_{id}(t)) + C_2\varphi_2(P_{gd}(t) - X_{id}(t)),$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1),$$

donde $i = 1, 2, ..., N$, siendo N el tamaño del enjambre, φ_1 y φ_2 son dos números aleatorios uniformemente distribuidos en el intervalo [0,1], C_1 y C_2 son los *coeficientes de aceleración* que representan la atracción que una partícula tiene hacia su propio éxito (parte cognitiva) o el de sus vecinas (parte social).

Algoritmo 8 Optimización por enjambre de partículas

Inicializar una población P de partículas con posiciones y velocidades aleatorias en las D dimensiones del espacio de búsqueda
while criterio de parada **do**
  for all partícula i **do**
    Adaptar la velocidad de la partícula
    Actualizar la posición de la partícula
    Evaluar la aptitud $f(\vec{X}_i)$
    if $f(\vec{X}_i) < f(\vec{P}_i)$ **then**
      $\vec{P}_i \leftarrow \vec{X}_i$
    end if
    if $f(\vec{X}_i) < f(\vec{P}_g)$ **then**
      $\vec{P}_g \leftarrow \vec{X}_i$
    end if
  end for
end while
return $\vec{P}_g$

Tras la publicación el algoritmo inicial se han propuesto numerosas mejoras. Una de las más exitosas [24] consiste en añadir un parámetro V_{max} limitando la velocidad, forzando así a que la componente $\vec{V}_i$ de cada partícula se mantenga dentro del rango $[-V_{max},+V_{max}]$. En [1] se demuestra que pese a que este parámetro facilita la exploración del espacio de búsqueda, no facilita la convergencia hacia el óptimo. Con la finalidad de encontrar un punto medio entre exploración y explotación, los distintos investigadores han propuesto, en este sentido, diversas estrategias. Las más interesantes son la *inercia* y la *opresión*.

En el *método de inercia* [58] se añade un peso ω que tiene la función de balancear la búsqueda global y la búsqueda local. Un valor grande de ω implica más exploración global (diversifica la búsqueda), mientras que un valor pequeño implica más exploración local (intensifica la búsqueda en la región). La expresión de la velocidad bajo el método de *inercia* queda:

$$V_{id}(t+1) = \omega V_{id}(t) + C_1\varphi_1(P_{id}(t) - X_{id}(t)) + C_2\varphi_2(P_{gd}(t) - X_{id}(t)).$$

Otro método para escapar de óptimos locales es el de *opresión*. Clerk y Kennedy [10] proponen aplicar un factor de opresión, χ, al cálculo de la velocidad de cada partícula. La expresión teniendo en cuenta este factor queda así:

$$V_{id}(t+1) = \chi(V_{id}(t) + C_1\varphi_1(P_{id}(t) - X_{id}(t)) + C_2\varphi_2(P_{gd}(t) - X_{id}(t))),$$

donde χ se define como

$$\chi = \frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}}.$$

4.3. Estado del arte de la aplicación de metaheurísticas al problema

En la presente sección se describen los trabajos científicos encontrados en la literatura, en los que se ha hecho uso de metaheurísticas como medio para la resolución del problema de *accesibilidad* o para otros problemas con similares características.

4.3.1. Metaheurística GRASP para mejorar la accesibilidad después de un desastre

En 2011 Maya y Sörensen [22] proponen la utilización de la metaheurística GRASP (véase la Sección 4.2.1.3) para resolver el mismo problema que se está abordando en esta Tesis Fin de Máster. Recordemos que se pretende resolver el problema de planificar la reconstrucción de una red de carreteras, ambientado en un entorno en vías de desarrollo tras la ocurrencia de un desastre natural, con el objetivo de maximizar la *accesibilidad* de la red. Para esta función de reparación se cuenta con un presupuesto económico y de mano de obra limitado.

El modelo de Maya y Sörensen ha sido una de las referencias más importantes a la hora de abordar este trabajo. Específicamente, uno de los enfoques que nos ha resultado de mayor interés han sido las técnicas utilizadas para almacenar la solución, calcular la función objetivo o calcular la accesibilidad de la red. Pese al alto coste computacional que conlleva las tareas previamente mencionadas, las soluciones propuestas en esta investigación han permitido abordarlas de una manera muy eficiente. Esto se ha conseguido mediante la utilización de dos mátrices: una primera matriz de distancias mínimas, T, que almacena la longitud mínima desde cualquier punto de la red a otro, y una segunda matriz, P, formada por los caminos mínimos desde cada punto de la red a otro.

A diferencia de nosotros, Maya y Sörensen proponen el uso de GRASP, un método constructivo que consta de dos fases, una de construcción y una de mejora. La *fase de construcción* construye una solución inicial por medio de un método voraz que aplica cierta aleatoriedad a la selección de candidato. Seguidamente, se aplica un algoritmo de búsqueda local para mejorar la solución obtenida por el método de construcción. La gran ventaja de GRASP es que la búsqueda local se puede sustituir por algún otro tipo de

heurística más inteligente, Maya y Sörensen proponen el método de *búsqueda en entornos variables* (véase la Sección 4.2.1.4).

Fase de construcción

Para la construcción de la solución inicial los autores definen el *algoritmo de inserción*. En primer lugar, se calculan las matrices de *distancias mínimas* T y de *caminos mínimos* P mediante el *algoritmo Floyd-Warshall* descrito en Hu y Shing [38]. La matriz T almacena para todos los puntos de la red los tiempos mínimos desde uno de ellos a cualquier otro. La matriz P, de manera similar, se encarga de almacenar los caminos. Una vez se han obtenido los valores iniciales de las matrices, iterativamente y hasta agotar el presupuesto, se reparan las carreteras que individualmente mejoren la accesibilidad.

No obstante, recalcular los caminos mínimos para estimar el ahorro en tiempo que supone arreglar una carretera es un proceso muy costoso e inviable. Para solucionarlo, se propone calcular el ahorro en tiempo (*saving*) con la siguiente expresión:

$$saving(e) = \sum_{l \in \mathcal{N}_2} \Big(min_{k \in \mathcal{N}_1}\{T[k,l]\} - \ min_{k \in \mathcal{N}_1}\{T[k,i] + f_t(e,1) + T[j,l]\} \Big), \qquad (4.1)$$

donde $f_t(e,l)$ es el tiempo que se tarda en recorrer la arista e cuando está en el nivel l, siendo l una variable binaria que indica si la carretera e está operativa o dañada.

Algoritmo 9 Algoritmo de inserción

Require: $\mathbf{X}$, B, H, T y P
  repeat
    $saving = 0$
    for todas las carreteras dañadas $e \in \varepsilon_r$ que pueden ser reparadas con el presupuesto restante **do**
      Estimar el ahorro de la inserción para e, $saving(e)$
      if $saving(e) > saving$ **then**
        $saving = saving(e)$
        $candidate = e$
      end if
    end for
    Mejora del candidato: Actualizar X, B, H, T y P
    $saving = 0$
  until no se ha encontrado ningún candidato

Nótese que el algoritmo es completamente determinista, es decir, crea la misma solución inicial cada vez que se ejecuta. Para solucionar este problema se sugiere no seleccionar

siempre la carretera cuya reparación produce el mayor *saving*, si no realizar la selección de manera estocástica.

Fase de mejora

Para la fase de mejora Maya y Sörensen utilizan el método de *búsqueda en entornos variables* (véase la Sección 4.2.1.4), concretamente, el enfoque determinístico *descenso del entorno variable* (VND) propuesto por Hansen y Mladenovic [33].

Algoritmo 10 Algoritmo de mejora

Require: Considerar solución inicial $x \in \mathbf{X}$
  Seleccionar los entornos $N_k(\mathbf{X})|k = 1, 2, ..., k_{max}$
  $k \leftarrow 1$
  repeat
    Explorar entorno k. Encontrar $x^* \in N_k(x)$
    if x^* es mejor que x **then**
      $x \leftarrow x^*$
      $k \leftarrow 1$
    else
      $k \leftarrow k + 1$
    end if
  until $k > k_{max}$

El algoritmo VND es similar a lo presentado en la Sección 4.2.1.3. En este caso particular se utilizan tres entornos, $k_{max} = 3$, que definen los tipos de movimiento: degradación, mejora e intercambio. El movimiento de degradación cambia el estado de la carretera de reparado a dañado; el movimiento de mejora realiza lo opuesto; mientras que el movimiento de intercambio aplica los dos movimientos anteriores de manera simultánea a dos carreteras distintas de la red. El orden o numeración de estas estructuras coincide con lo propuesto en [55], es decir, (1) degradar, (2) intercambiar y (3) mejorar.

4.3.2. Enfoque de maximización de la accesibilidad en la planificación de una red de carreteras

Antunes, Seco y Pinto, [2], proponen en 2003 un enfoque de maximización de la accesibilidad aplicada a largo plazo a la planificación de una red de carreteras interurbanas. El enfoque tradicional utilizado para este tipo de problemas consiste en encontrar la red de coste mínimo capaz de acomodar los flujos origen y destino dados [43].

Sin embargo, este enfoque no se ajusta adecuadamente a los modelos de toma decisiones que caracterizan a los procesos de planificación espacial. Tampoco reconoce la interdependencia entre la calidad de la carretera y el flujo de tráfico, y no toma en cuenta cuestiones de equidad. Por ello, los autores proponen un nuevo enfoque basado en la *accesibilidad* como factor a optimizar en la construcción de la red de carreteras. De este modo, el modelo es mucho más equitativo con todas las ciudades, ya que las más grandes no concentrarán el acceso a la mayor parte de recursos y servicios de la zona sino que la mayoría de ellas tendrán acceso.

En definitiva, el nuevo enfoque intenta obtener la solución con máxima accesibilidad para el desarrollo de una red de carreteras interurbanas a la vez que se garantiza el nivel de equidad para la distribución de la accesibilidad a través de los diferentes centros, dadas unas restricciones de presupuesto. La función objetivo del nuevo enfoque queda del modo:

$$max\ A_\varepsilon = \sum_{i \in a_i} w_i a_i,$$

donde

$$a_i = \sum_{j \in \mathbf{C}_i} \frac{s_j}{c_{ij}(\mathbf{Y})}, \forall i \in \mathbf{C},$$

y c_{ij} es el camino mínimo entre los centros i y j, s_j es el tamaño del centro j e $\mathbf{Y} = \{y_{lm}, l \in \mathbf{L}, m \in \mathbf{M}_i\}$ son las variables de decisión que representan las posibles decisiones a tomar con cada enlace de la red.

Las posibles decisiones son: no hacer nada, construir una nueva conexión de una categoría determinada y mejorar una conexión a una categoría superior.

Para resolver el problema se proponen dos métodos distintos, uno basado en una *búsqueda local* y un segundo la metaheurística de *recocido simulado* (véase la Sección 4.2.1.1). La búsqueda local utilizada es la heurística (A+I), adición e intercambio.

4.3.3. Mejorando la accesibilidad en servicios de salud rurales: El problema de la mejora del cubrimiento máximo en una red

Este proyecto surge bajo la idea de facilitar el acceso del mayor número de ciudadanos en zonas rurales a los servicios sanitarios, enmarcado en un entorno de país en desarrollo. Las agencias gubernamentales e internacionales han subrayado la importancia de la relación entre medios de transporte y salud. Un factor clave para recibir un correcto tratamiento médico reside en la facilidad de acceder a estos servicios en un tiempo, distancia y transporte adecuados. Así pues, en los últimos años se viene estudiando posibles formas

de optimizar la ubicación de los centros sanitarios que garanticen un nivel de accesibilidad alto para los ciudadanos.

Este es un modelo clásico que, ya sea para la ubicación de centros sanitarios o para la ubicación de almacenes en redes de transporte se soluciona con el modelo ubicación-asignación o *location-allocation* (LA) [11]. El problema es que a pesar de ser un modelo eficiente en regiones desarrolladas, ignora en regiones en vías de desarrollo la posible ganancia en términos de accesibilidad que supone mejorar la red de transporte.

Este modelo tradicional supone estáticas las carreteras a lo largo del año, por tanto, como se demuestra en [51], suspende al aplicarse a zonas en desarrollo, donde una simple lluvia puede inhabilitar muchas de las infraestructuras de transporte impidiendo el acceso de muchos ciudadanos a los centros sanitarios.

Para solucionar el problema previamente descrito, se ha propuesto un modelo que incorpora las características específicas de las carreteras y que tiene como objetivo incrementar el acceso a los servicios sanitarios mejorando la red de carreteras y asumiendo que las localizaciones de hospitales y clínicas son fijas.

Se definen de tres tipos de carreteras diferentes, carreteras tipo *trail* (carreteras con una calidad baja siendo imposible circular por ellas en época de lluvias), carreteras tipo *dirt* (se ven afectadas por las lluvias pero pueden ser transitadas con dificultad) y carreteras pavimentadas o *paved* (no se ven afectadas por la época de lluvias y pueden ser utilizadas durante todo el año).

Para modelar el problema se utiliza una estructura tipo grafo G, con un conjunto de nodos, N, que representan las ciudades, y un conjunto de aristas, A, que representan las carreteras que enlazan las ciudades. Nótese que se modela de un modo muy parecido al problema que se intenta resolver en esta Tesis Fin de Máster.

Para la resolución del problema se plantea el uso de técnicas de programación entera. En este caso concreto, se utiliza el software ILOG-CPLEX[1], una suite de resolución de problemas de optimización, orientada a la toma de decisiones.

[1] http://www-03.ibm.com/software/products/es/ibmilogcpleoptistud

Capítulo 5

Sistemas de colonias de hormigas

5.1. La hormiga real

Las hormigas son uno de los insectos más alucinantes que pueblan La Tierra. Estos pequeños insectos superan en número a cualquier otra especie, se estima que hay 1.5 millones de hormigas por cada ser humano. También son uno de los insectos más longevos que existen, pudiendo llegar la reina a 28 años de vida.

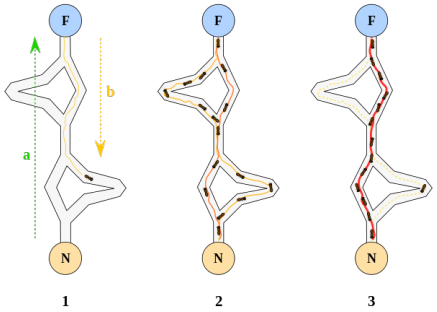

Figura 5.1: Fenómeno del doble puente

A pesar de estas características, lo verdaderamente interesante es su comportamiento en grupo y la cooperación existente entre los millones de individuos que forman una colonia para realizar actividades que escapan a la capacidad y habilidades cognitivas de un solo individuo.

Para nuestro trabajo será particularmente interesante el proceso de exploración de las hormigas o el proceso de búsqueda de alimento. Las hormigas se comunican por un proceso llamado **estigmergia**, es decir, por información local muy simple y utilizando el entorno como vía de comunicación.

La mayoría de las hormigas tiene una capacidad visual muy limitada, siendo ciegas en muchas ocasiones, y para comunicarse utilizan un rastro de material químico llamado *feromona*. Cuando una hormiga encuentra una fuente de alimentación, tras evaluar su calidad,

va dejando un rastro de feromonas proporcional a la calidad del alimento, ocasionando un *feedback* positivo [35] en el resto de las hormigas, para que estas sigan su camino.

Como se puede obervar, es un sistema descentralizado, es decir, que no existe ningún individuo que dirija al resto y emerge de forma espontánea.

El fenómeno del doble puente [20] sirvió para demostrar la optimalidad del proceso de exploración y búsqueda de comida de las hormigas. El mismo se puede observar en la Figura 5.1. Se supone una fuente de alimentación, nodo F de la figura, y una ubicación del nido de las hormigas, nodo A, para comprobar el comportamiento de búsqueda. Inicialmente en cada intersección que se plantea en el camino de una hormiga, esta escoge una dirección de manera aleatoria, pudiendo elegir entre el camino corto o el largo, (*dibujo 1* de la Figura 5.1). Cuando los individuos encuentren la fuente de comida, marcarán con feromona el camino utilizado para volver al nido indicando al resto de hormigas el camino para llegar a la fuente. Con el paso del tiempo, el nivel de feromonas depositado en el camino corto será más abundante que en el camino largo. La razón es que al ser más corto por unidad de tiempo pasarán mayor número de hormigas depositando en total mayor cantidad de feromonas. De este modo, las nuevas hormigas que salen del hormiguero tenderán a seguir con mayor probabilidad el camino con el nivel de feromonas más alto (*dibujo 2*). Finalmente, llegará un momento en que el nivel de feromonas del camino corto sea mucho mayor, pudiendo llegar a evaporarse por completo el rastro de feromonas del camino subóptimo, (*dibujo 3*).

5.2. Optimización por colonias de hormigas

La optimización por colonias de hormigas o *Ant colony optimization* (ACO), es un algoritmo de optimización estocástico que construye una solución de forma iterativa añadiendo componentes a soluciones parciales. ACO es una metaheurística nacida para resolver problemas de optimización combinatoria y por ello ha sido este su mayor área de aplicación.

Aplicar ACO a problemas de optimización combinatoria no es complicado una vez se ha definido un mapeado del problema en forma de grafo que permita construir una solución de forma incremental, una estructura de entorno y una regla estocástica de transición para ser utilizada de forma local en el proceso de construcción de cada solución.

El componente estocástico del algoritmo lo marca la *función de transición* [18], que se utiliza para decidir qué caminos debe seguir la hormiga a la hora de construir las soluciones. Esta función es dependiente de las dos variables más importantes del algoritmo:

- La información **heurística** sobre el problema, que permite clasificar los componentes de la solución global según su calidad a priori.

- La **feromona** artificial, que cambia dinámicamente en tiempo de ejecución y refleja la experiencia de búsqueda adquirida por las hormigas artificiales. Se utilizará para guiar las hormigas hacía las mejores soluciones encontradas por sus predecesoras.

Todo algoritmo de hormigas se compone de una serie de fases generales que se especifican a continuación:

- *Inicialización*: Se inicializan los parámetros del algoritmo, especialmente τ_0, que es la feromona inicial que tendrá cada componente parcial de la solución, es decir, cada camino o arista del grafo.

- *Construcción de soluciones*: El conjunto de hormigas construyen las soluciones para la instancia del problema considerado. Para hacerlo, cada hormiga empieza con una solución inicialmente vacía y va completándola iterativamente. A cada paso, cada hormiga amplía su solución parcial eligiendo un componente de solución factible y añadiéndola a su solución parcial actual. El conjunto de soluciones factibles son los caminos que puede tomar una hormiga desde cada nodo del grafo. Como ya hemos comentado previamente, esta elección se realiza de manera probabilista según la *función de transición.*

- *Aplicación de búsqueda local*: Este paso es totalmente optativo, pero en [19] se demuestra que los algoritmos ACO alcanzan su mejor rendimiento cuando se utilizan conjuntamente con algoritmos de búsqueda local para mejorar cada solución construida por las hormigas.

- *Actualización de feromonas*: Tras haber construido las soluciones, en función de la calidad de las mismas, se actuliza el nivel de feromonas de los caminos o componentes de solución parcial que forman las soluciones globales. De este modo, se pretende guiar a las hormigas de futuras iteraciones a que tomen los componentes parciales de las mejores soluciones globales encontradas por sus hormigas predecesoras.

Algunos de los aspectos del algoritmo general de ACO son dependientes del problema, tales como la información heurística o el algoritmo de búsqueda local. Esta flexibilidad permite adaptar fácilmente ACO a un gran número de problemas con muy buenos resultados en cuanto a eficiencia y rendimiento.

Algoritmo 11 Algoritmo ACO básico

```
Inicialización
while no criterio de parada do
    Hormigas construyen soluciones
    Aplicar búsqueda local (opcional)
    Actualizar feromonas
end while
```

A continuación, se hará una descripción detallada de los algoritmos ACO existentes en la literatura. Se comenzará analizando el algoritmo original, el *Sistema de hormigas*, para pasar a estudiar la gran variedad de variantes y mejoras posteriores, como el *Sistema de colonias de hormigas* o el *Sistema de hormigas Max-Min*, entre otros.

5.2.1. Sistema de hormigas (AS)

En 1996 se propone el primer algoritmo de hormigas, el *Sistema de hormigas* (*Ant System*, AS). Sus autores fueron M. Dorigo y A. Colorni [18], los cuales se bioinspiraron en el comportamiento real de las colonias de hormigas a la hora de buscar su fuente de alimentación. Por sus características, el algoritmo se asocia siempre a problemas en forma de red, donde los nodos son componentes de la solución y las aristas, que contienen las feromonas, representan los caminos a escoger por las hormigas.

El algoritmo AS funciona del modo que se describe a continuación. En la *fase de inicialización* se inicializan los niveles de feromonas. Al comienzo todas las aristas disponen del mismo volumen de feromonas. Seguidamente, partiendo desde cada nodo, las hormigas crean los caminos de manera estocástica guiándose por la función de probabilidad, P_{ij}^k, (véase la Expresión 5.1). Una vez obtenidos los caminos, en función de la calidad de las soluciones obtenidas, se realiza la actualización de feromonas de las aristas pertenecientes a los caminos encontrados (Expresión 5.2). El proceso se repite de manera iterativa hasta que las hormigas converjan hacia una solución o camino óptimo.

Como se ha mencionado previamente, las hormigas construyen los caminos de manera estocástica siguiendo una función de probabilidad o *probabilidad de transición*. Esta probabilidad depende de dos factores: τ_{ij}, la feromona depositada en la arista que incrementa la probabilidad de elegir ese camino cuanto mayor sea su volumen; y η_{ij}, la información heurística del camino que a priori distingue entre caminos mejores y peores. Así pues, la

expresión toma la siguiente forma:

$$P_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}{\sum_{k \in A_k} [\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}, & \text{si } j \in A_k \\ 0, & \text{en caso contrario} \end{cases}, \tag{5.1}$$

donde α y β son parámetros que controlan la importancia relativa de la feromona en relación a la información heurística y A_k es el conjunto de nodos visitados por la hormiga k.

Algoritmo 12 Algoritmo Sistema de Hormigas

```
Inicializar parámetros, τ_ij y Δτ_ij
Colocar las m hormigas en los n nodos
for k = 1 to m do
    Colocar el primer componente de solución de la hormiga k en L_k
    repeat
        Seleccionar el componente j al que moverse con probabilidad P_ij^k(t)
        Mover la hormiga k a la ciudad j
        Insertar el componente j en la lista L_k de la hormiga k
    until L está llena
end for
for k = 1 to m do
    Calcular Δτ_ij^k de la solución
    Actualizar la feromona de cada arista visitada por la hormiga k
end for
```

Se debe distinguir entre iteración y ciclo, siendo una iteración los m movimientos realizados por las m hormigas en el intervalo $(t, t+1)$ y un ciclo las n iteraciones del algoritmo para las cuales cada hormiga ha completado su ruta. En cada iteración, $\tau_{ij}(t)$, la intensidad de feromona en la arista (i, j) en tiempo t, se actualiza de acuerdo a la siguiente expresión:

$$\tau_{ij}(t) = (1 - \rho)\tau_{ij}(t-1) + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \tag{5.2}$$

donde ρ es el *coeficiente de evaporación* del rastro de feromonas, el cual impide la acumulación excesiva de feromonas con el paso de las iteraciones, y $\Delta\tau_{ij}^k$ es la cantidad de feromona depositada en la arista (i, j) por la k-ésima hormiga:

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{L_k}, & \text{si la } k\text{-ésima hormiga utiliza la arista } (i,j) \text{ en su ruta} \\ 0, & \text{en caso contrario} \end{cases}, \tag{5.3}$$

donde L_k es la longitud del camino de la k-ésima hormiga. De este modo, si el camino encontrado por una hormiga es corto, lo que significa una mejor solución, entonces el nivel de feromona depositado es mayor.

En la literatura existente se ha demostrado que el rendimiento de AS empeora con el incremento del número de nodos en el grafo, solo siendo eficiente para estructuras con menos de 30 nodos.

Con el objetivo de superar este inconveniente se han propuesto extensiones o variantes del algoritmo original, que mejoran el balance entre exploración y explotación y con mayor capacidad de escapar de óptimos locales. Entre los más utilizados se encuentran el *Sistema de hormigas elitista* (AS_{elite}), el *Sistema de hormigas con ranking* (AS_{rank}) y *Sistema de hormigas mejor-peor* (AS_{bw}), que se describen a continuación.

5.2.1.1. Sistema de hormigas elitista (AS_{elite})

Una de las problemáticas de la versión original del AS, es la incapacidad de converger hacia una solución óptima para grafos con muchos nodos, es decir, en la última fase del algoritmo no consigue terminar de explotar la mejor solución obtenida.

La idea de la versión AS_{elite} [18] es potenciar la mejor ruta global encontrada después de cada iteración para aumentar la explotación. Para ello, en el proceso de actualización de feromonas se aplica un depósito adicional a los arcos o aristas que pertenecen a las mejores soluciones encontradas de manera global, es decir, desde el principio de la búsqueda:

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{mejorglobal},$$

donde e es el número de hormigas elitistas y

$$\Delta\tau_{ij}^{mejorglobal} = \begin{cases} \frac{Q}{L^*}, & \text{si la arista } (i,j) \text{ pertenece a los mejores caminos} \\ 0, & \text{en caso contrario} \end{cases},$$

siendo L^* la longitud de la mejor ruta encontrada.

5.2.1.2. Sistema de hormigas con ranking (AS_{rank})

La variante AS_{elite} descrita anteriormente es poco eficiente cuando muchas de las hormigas obtienen caminos buenos pero subóptimos. Para corregir esto, en 1997 Bullnheimer et al. proponen el *Sistema de hormigas con ranking* (AS_{rank}) [3].

En este sistema las hormigas son ordenadas en un ranking según la calidad o longitud de su ruta, de modo que la contribución de feromonas por parte de cada hormiga depende de su posición μ en el ranking. Cuanto más avanzada sea su posición, más feromonas depositará. Con este modelo, que suele ser combinado con el modelo elitista, la expresión de la actualización de feromonas queda como

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij} + \sum_{k=1}^{w} -1(w-\mu)\Delta\tau_{ij}^{\mu} + w\Delta\tau_{ij}^{mejorglobal},$$

donde w es el tamaño del ranking, siendo entonces un total de $(w-1)$ hormigas más la mejor hormiga encontrada hasta el momento las que contribuyen al rastro de feromonas, y donde

$$\Delta\tau_{ij}^{\mu} = \begin{cases} \frac{Q}{L_{\mu}}, & \text{si la } \mu\text{-ésima mejor hormiga recorre la arista } (i,j) \\ 0, & \text{en caso contrario} \end{cases},$$

$$\Delta\tau_{ij}^{*} = \begin{cases} \frac{Q}{L^{*}}, & \text{si la arista } (i,j) \text{ es parte de la mejor solución} \\ 0, & \text{en caso contrario} \end{cases},$$

siendo μ el índice del ranking y $\Delta\tau_{ij}^{\mu}$ el incremento del rastro de feromonas en la arista (i,j) causado por la μ-ésima mejor hormiga.

5.2.1.3. Sistema de hormigas mejor-peor (AS_{bw})

Cordón et al. proponen en 1999 la variante *Sistema de hormigas mejor-peor* [12]. El objetivo es el mismo, mejorar la interacción exploración-explotación para mejorar el rendimiento del algoritmo original. La característica particular de este modelo reside en el hecho de que incorpora conceptos y estrategias de la computación evolutiva. Esta versión de AS tiene tres diferencias bien marcadas con respecto al algoritmo original.

En el proceso de actualización de feromonas, además de reforzar positivamente las aristas de la mejor solución global, se penaliza negativamente las aristas de la peor solución encontrada:

$$\tau_{ij} = \begin{cases} \tau_{ij} + \rho \times \Delta\tau_{ij} & \forall (i,j) \in S_{gb} \\ (1-\rho) \times \tau_{ij} & \forall (i,j) \in S_{current-worst} \text{ y } (i,j) \notin S_{gb} \end{cases}.$$

Se utiliza un operador de *mutación* para dotar de mayor diversificación al proceso de búsqueda. De igual modo que en los algoritmos evolutivos, la mutación no se realiza sobre todos los caminos sino que se aplica con cierta probabilidad, P_m. La mutación no modifica directamente los caminos encontrados sino que se aplica en el nivel de feromonas de las aristas:

$$\tau'_{ij} = \begin{cases} \tau_{ij} + mut(it, \tau_{avg}), & \text{si } a = 0 \\ \tau_{ij} - mut(it, \tau_{avg}), & \text{si } a = 1 \end{cases},$$

donde $mut(it, \tau_{avg})$ es el operador de mutación que depende de la media de los rastros de feromonas en la mejor solución global y de la iteración actual it y $a \in [0, 1]$ es un número aleatorio.

El operador de mutación es dependiente de la iteración porque se busca que en las primeras iteraciones la mutación sea mayor para aumentar el proceso de exploración, reduciéndose la mutación con el paso de las iteraciones para darle mayor peso al proceso de explotación.

La tercera gran diferencia de AS_{bw}, aunque es aplicable a otros modelos, es la reinicialización de todos los rastros de feromonas cuando se produce un estancamiento en el proceso de búsqueda (durante un número determinado de iteraciones no se consigue mejorar la mejor solución encontrada), normalmente debido a óptimos locales.

5.2.2. Sistema de colonia de hormigas (ACS)

El *Sistema de colonia de hormigas* (*Ant Colony System*, (ACS)) fue propuesto en 1997 por Marco Dorigo y Luca Maria Gambardella [17]. A diferencia de su predecesor AS, este modelo es especialmente útil para problemas cuyo grafo esta formado por más de 30 nodos, es decir, tiene un tamaño mediano-grande. En comparación con AS se puede decir que ACS:

- Explota en mayor medida la experiencia acumulada en el proceso de búsqueda. Además, la regla de transición incorpora un modo directo para balancear entre exploración y explotación.
- Únicamente la mejor solución o hormiga encontrada hasta el momento participa en el depósito de feromonas, ocasionando una búsqueda mucho más dirigida al entorno de la mejor solución encontrada. La consideración de una sola hormiga hace que el algoritmo sea computacionalmente más eficiente y consiga tratar problemas de mayor tamaño.

- Se incorpora una *regla de evaporación local* que actúa durante el proceso de construcción de caminos y no al final, evaporando feromona en el mismo momento que una hormiga atraviesa una arista. Esto permite una búsqueda mucho más diversificada al aumentar la probabilidad de que las hormigas no sigan todas los mismos caminos.

El algoritmo ACS sigue la siguiente secuencia de pasos. En primer lugar, se inicializa el nivel de feromonas a un nivel idéntico para todos los caminos, y se colocan m hormigas en los n nodos.

Posteriormente, cada hormiga construye un camino siguiendo la regla de transición (Expresión 5.4). Durante la construcción de los caminos se aplica la regla de evaporación local, cada vez que una hormiga atraviese una arista se reduce el nivel de feromona de la misma. De este modo, se reduce la probabilidad de que otra hormiga siga los pasos de la hormiga actual.

Finalmente, una vez que las hormigas han terminado de construir una ruta, estas se evalúan y solo la mejor hormiga o ruta encontrada de manera global deposita feromonas en todas las aristas que pertenecen a esa ruta. Por último, se realiza una evaporación global de feromonas para evitar que el volumen de feromonas sea demasiado alto.

Algoritmo 13 Algoritmo Sistema de colonia de hormigas

```
Inicializar parámetros, τ_ij y Δτ_ij
for k = 1 to m do
    Colocar las m hormigas en los n nodos
    repeat
        La hormiga k aplica P_ij^k
        La hormiga k aplica la regla local de actualización de feromonas
    until todas las hormigas han construido una solución completa
    Aplicar la regla global de actualización de feromonas
end for
```

Como comentabamos en la introducción de este algoritmo, ACS lleva incorporado en la *función de transición* un modo directo de balancear entre explotación y exploración. La probabilidad P_{ij}^k que tiene una hormiga ubicada en el nodo i de elegir el nodo j queda definida mediante:

$$P_{ij}^k = \begin{cases} argmax_{j \in J_k(i)} \left\{ [\tau_{ij}] \times [\eta_{ij}]^\beta \right\}, & \text{si } q \leq q_0 \text{ (explotación)} \\ \frac{[\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}{\sum_{k \in A_k} [\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}, & \text{en caso contrario (exploración sesgada)} \end{cases}, \quad (5.4)$$

donde τ_{ij} es la feromona depositada en la arista, η_{ij} la información heurística del arco y β es un parámetro que controla el peso de la información heurística frente al nivel

de feromonas. q es un número aleatorio uniformemente distribuido en [0,1] y q_0 es un parámetro ($0 \leq q_0 \leq 1$) que determina la importancia relativa de la explotación respecto de la exploración. Cada vez que una hormiga debe elegir una arista para moverse, se genera un número aleatorio q, si $q \leq q_0$ entonces se selecciona la mejor arista (explotación), en caso contrario se elige estocásticamente entre todas las aristas (exploración sesgada).

La *regla de actualización global* es parecida a la de AS. Se aplica al final de la iteración cuando todas las hormigas han construido su camino, todos los rastros se evaporan según el *factor de evaporación* α y solo la mejor solución encontrada hasta el momento deposita feromona en todas las aristas que forman la ruta:

$$\tau_{ij}(t+n) = (1-\alpha)\tau_{ij} + \alpha \times \Delta\tau_{ij},$$

donde

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L_{gb}}, & \text{si } i,j \in T_{best} \\ 0, & \text{en caso contrario} \end{cases},$$

y L_{gb} es la longitud de la mejor ruta global desde el inicio del algoritmo, T_{best}.

La *regla de actualización local*, que se aplica mientras las hormigas construyen cada una de las soluciones, tiene las siguiente forma:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \rho \times \Delta\tau_{ij},$$

donde $\rho (0 < \rho < 1)$ es el parámetro de evaporación local y $\Delta\tau_{ij} = \tau_0$, siendo τ_0 el parámetro de feromona inicial normalmente inicializado según $\tau_0 = (\frac{1}{n \times L_{nn}})$ donde L_{nn}, es la longitud de la ruta producida por la heurística de vecino más próximo o *nearest neighbor heuristic* [54] y n es el número de nodos de la estructura. Recordamos que el objetivo de la regla local de actualización de feromonas es hacer que la preferencia que tienen las hormigas por cada una de las aristas evolucione de manera dinámica durante cada iteración. De este modo, se fuerza a que las hormigas tomen caminos distintos favoreciendo la diversificación.

5.2.2.1. Algoritmo Ant-Q

Un algoritmo que se puede considerar por concepto una variante del ACS es el algoritmo *Ant-Q*, propuesto en 1995 por L.M. Gambardella y M. Dorigo [27]. Este modelo se basa en el *algoritmo Q-learning* [62] y la idea sobre la que se sostenta es la de aplicar el aprendizaje por refuerzo al algoritmo de hormigas.

El algoritmo Q-learning permite al agente ir aprendiendo la política óptima de actuación. Cuando el agente realiza una acción, el nuevo estado al que se llega es evaluado y

el resultado se usa para aprender iterativamente las mejores acciones. Para aplicar esta ideología al algoritmo de hormigas lo que se hace es actualizar las feromonas en función del valor de la evaluación que se realiza en el siguiente estado. Así pues, la fórmula Q-learning adapatada a la actualización de feromonas toma la siguiente forma:

$$\tau_{ij}(t+n) = (1-\alpha)\tau_{ij} + \alpha \times \Delta A\tau_{ij},$$

donde

$$\Delta A\tau_{ij} = \sum_{k=1}^{m} \Delta A\tau_{ij}^{k} + \gamma \times \ max_{z \in J_k(s)}\tau_{ij},$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_{k_{ib}}}, & \text{si } i,j \in J_k \\ 0, & \text{en caso contrario} \end{cases},$$

siendo γ el *factor de aprendizaje* y $J_k(s)$ el conjunto de nodos que ha visitado la hormiga k.

Nótese que en esta nueva versión no se incorpora la regla de actualización local típica en ACS y la actualización de feromonas no se realiza sobre las aristas visitadas por las hormigas sino que se realiza en todas las aristas.

5.2.3. Sistema de hormigas MAX-MIN (MMAS)

T. Stützle y H. Hoos [59] proponen en el año 2000 otra versión de AS que una vez más busca mayor rendimiento en problemas de tamaño grande y balancear el nivel de exploración-explotación para evitar el estancamiento de la búsqueda. El sistema propuesto es el sistema de hormigas MAX-MIN (*MAX-MIN Ant System*), que incorpora principalmente dos mejoras frente al AS:

- Al igual que ACS, solo una hormiga deposita feromona después de cada iteración. Sin embargo, esta hormiga no tiene porqué ser la mejor hormiga global si no que en ocasiones se utiliza la mejor hormiga de la iteración. Esto ayuda a proporcionar diversificación y evitar estancamiento.
- Limita el movimiento de las feromonas hacia extremos, encapsulando el nivel de feromonas dentro del intervalo $[\tau_{min}, \tau_{max}]$.

El estancamiento en los algoritmos de hormigas se produce cuando un posible camino o arista tiene un nivel de feromonas mucho más alto que el resto de los caminos. De este modo, la hormiga siempre escoge el camino con el nivel más alto de feromonas siendo probabilísticamente imposible escoger otro de los caminos.

Para evitarlo se introduce en MMAS el intervalo de feromonas, así, el nivel mínimo de feromonas que cualquier camino puede tener es τ_{min} y el nivel máximo es τ_{max}. Con esta incorporación la hormiga tenderá ha escoger el camino con mayor feromona pero, en ocasiones, con una pequeña probabilidad la hormiga escogerá un camino con nivel de feromonas más bajo ya que la diferencia entre τ_{min} y τ_{max} no es tan grande como para inhabilitar la elección del camino con menor nivel de feromonas. Por supuesto, los valores de estos límites deben ser elegidos de forma adecuada y ajustados, pues condicionan directamente el equilibrio entre la explotación y la exploración en el proceso de búsqueda. Se definen con la siguiente expresión:

$$\tau_{max} = \frac{1}{1-\rho} \times \frac{1}{f(s^{best})} = \frac{1}{\rho \times f(s^{best})},$$

$$\tau_{min} = \frac{\tau_{max} \times (1 - \sqrt[n]{p_{best}})}{(avg-1) \times \sqrt[n]{p_{best}}},$$

donde ρ es el factor de evaporación y $f(s^{best})$ es la heurística relacionada con la mejor solución encontrada.

Así, la convergencia se da cuando para cada posible elección por parte de una hormiga, una de las componentes de solución está en el nivel τ_{max}, mientras que el resto de componentes tienen τ_{min}.

Otra de las características asociadas a este algoritmo es su forma de actulizar feromonas, donde una sola hormiga deposita feromonas pero esta no tiene porque ser la mejor global. La actualización de feromonas se realiza del siguiente modo:

$$\tau_{ij}(t+1) = \rho \times \tau_{ij}(t) + \Delta\tau_{ij}^{best},$$

siendo $\Delta\tau_{ij}^{best} = \frac{1}{f(s^{best})}$ y $f(s^{best})$ el valor de la heurística para la mejor solución de la iteración (s^{ib}) o para la mejor solución global (s^{gb}).

La utilización exclusiva de s^{gb} puede ocasionar que la búsqueda se centre demasiado rápido en el entorno de ésta, limitando el proceso de exploración del algoritmo y pudiendo quedar atrapado en soluciones de baja calidad.

La introducción de s^{ib} en el proceso de actualización de feromonas permite mucha mayor diversificación y dota al algoritmo de potencia para escapar de subóptimos. En la utilización de esta estrategia híbrida se recomienda variar la frecuencia con la que se utiliza s^{ib}. Al comienzo del algoritmo su frecuencia debería ser mayor para favorecer el proceso de exploración. Posteriormente, con el paso de las iteraciones la frecuencia de s^{ib} debería reducirse paulativamente en favor de s^{gb}, dotando al algoritmo de capacidad para converger y explotar la mejor solución obtenida hasta el momento.

5.2.4. La heurística ANTS

V. Maniezzo y A. Carbonaro proponen ANTS [45] en 2000. El algoritmo ANTS se diferencia del algoritmo original AS en tres factores:

- En lugar de utilizar una función heurística para valorar lo atractiva que es una solución, esto se estima en función de valores de coste para completar una solución.
- Se hace uso de una *lista tabú* para dar diversidad a la distribución de probabilidad asociada al movimiento de las hormigas.
- Incorpora una nueva fórmula de actualización de feromonas para evitar el proceso de estancamiento.

Algoritmo 14 Algoritmo ANTS

Inicializar parámetros, variables y el coste mínimo óptimo lineal del problema (LB).
for $k = 1$ to m **do**
  repeat
    Calcular η_{ij} como el coste de completar una solución que contiene j
    Seleccionar el movimiento mediante la fórmula de distribución de probabilidad
    Añadir el movimiento elegido a la lista tabú
  until la hormiga k ha completado su solución
  Aplicar búsqueda local a la solución encontrada
end for
for cada movimiento (ij) de cada hormiga **do**
  Calcular $\Delta\tau_{ij}^k$
  Actualizar los rastros de feromonas
end for

La incorporación de la lista tabú a la *función de transición* para evitar estancamiento y favorecer el proceso explorativo queda de la siguiente forma:

$$P_{ij}^k = \begin{cases} \frac{\alpha\tau_{ij}+(1-\alpha)\eta_{ij}}{\sum_{(iv)\notin tabu_k}(\alpha\tau_{iv}+(1-\alpha)\eta_{iv})}, & \text{si } (ij) \notin tabu_k \\ 0, & \text{en caso contrario} \end{cases}, \tag{5.5}$$

donde α representa la importancia relativa del rastro de feromonas respecto a la calidad de la arista y $tabu_k$ es una lista que indica el conjunto de movimientos no aptos para la hormiga k.

La nueva fórmula de actualización de feromonas tiene la siguiente forma:

$$\tau_{ij}(t) = \tau_{ij}(t-1) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k},$$

donde

$$\Delta\tau_{ij}^{k} = \tau_0\Big(1 - \frac{z_{curr} - LB}{\hat{z} - LB}\Big),$$

siendo k cada hormiga, $\hat{z}$ la media de los costes de los movimientos realizados por cada hormiga, z_{curr} el coste del movimiento actual, LB el coste mínimo de la solución óptima al problema [44] y τ_0 el nivel de feromonas inicial.

5.3. Estado del arte de la aplicación de ACO al problema

5.3.1. Optimización por colonias de hormigas para operaciones de ayuda en desastres

El problema descrito en Yi et al. (2007) también trata con los procesos de planificación tras un desastre natural. Concretamente, se aborda el problema de la coordinación del transporte de productos desde los principales centros de distribución hacia las áreas afectadas, y del transporte de las personas heridas hacia los centros de emergencias médicas. Definen más formalmente la función objetivo como el intento de minimizar el retraso en la provisión de productos de primera necesidad a los afectados y dar servicios médicos a los supervivientes heridos.

Diversas restricciones del problema que no entraremos a detallar lo alejan de ser modelado como un problema clásico VRP. El hecho de que la capacidad del vehículo sea variable a lo largo de la ruta y de que tratemos con distintos tipos de carga, elevan la dificultad del problema por encima del clásico problema *integer multi-commodity flow problem*.

Para su resolución, los autores proponen la división del problema en dos partes, una primera parte donde se construyen las rutas de los vehículos de manera estocástica por medio de la utilización de un algoritmo ACO, y una segunda parte donde se utiliza el algoritmo SMF o *successive maximum flow* [29] para resolver un problema de multireparto basándose en el resultado de los flujos de vehículos. Las dos partes están conectadas de manera implícita, ya que las feromonas del ACO se actualizan en base a los resultados del SMF.

5.3.2. Algoritmo híbrido basado en colonias de hormigas para la reparación de emergencia en el problema de la red tiempo-espacio

A pesar de los avances en medidas de predicción de desastres naturales, cuando estos ocurren los daños causados siguen teniendo un gran impacto en las vías de transporte dificultando los servicios de rescate y la distribución de productos de primera necesidad. Numerosas poblaciones quedan completamente aisladas de estos servicios.

En 2011, Yan et al. [63] proponen un nuevo enfoque para resolver esta situación desde el punto de vista de la aplicación de un sistema de reparación de infraestructuras a corto plazo. Consiste en un modelo de red espacio-tiempo o *time-space network model* con el objetivo de minimizar el tiempo necesario empleado en las tareas de reparación sujetos a un conjunto de restricciones operacionales.

En el modelo, una estación de trabajo es una ubicación donde los equipos de trabajo pueden estacionarse para próximos trabajos de reparación. Un punto de reparación representa la ubicación de un área dañada que no puede ser atravesada. Una ruta de un equipo de trabajo consiste de una secuencia de intersecciones y segmentos contiguos.

El objetivo del trabajo de reparación es, por tanto, la reconexión de la red en el menor tiempo posible, es decir, minimizar el tiempo empleado en realizar todas las tareas de reparación de las infraestructuras. Según el problema descrito, el modelo de red espacio-tiempo se ajusta perfectamente a las necesidades y se formula como un problema de flujo de red entera o *integer network flow problem* con limitaciones.

$$min\ Z = \varepsilon,$$

donde

$$\varepsilon = \ max\ c_{ij} \times m_{ij}, \forall ij \in A^{EC}.$$

donde c_{ij} es el coste del arco (i, j), m_{ij} es una variable binaria que indica si un arco (i, j) tiene flujo, ε es una variable para transformar la función Minimax en una función linear y A^{EC} es el conjunto de todos los arcos.

Para la resolución del problema, los autores proponen el algoritmo ACS junto con algunas técnicas de aceptación de umbral o *threshold accepting* (ACSB), desarrollando así, un enfoque híbrido que resuelve de forma eficiente el problema del flujo en una red tiempo-espacio.

El algoritmo se desarrolla del siguiente modo: comenzando con las soluciones iniciales, cada hormiga construye una solución factible a la cual se le aplica posteriormente un algoritmo de búsqueda local para mejorarla.

Tras las actualización de las feromonas de los arcos que forman parte de las soluciones encontradas, se aplica la regla de aceptación para cambiar el límite de tiempo de trabajo de cada equipo de reparación. El proceso se repite de forma iterativa hasta cumplir condición de parada. Para la generalización de la solución inicial se utiliza el algoritmo de corrección de etiquetas, *label correcting algorithm* (LCA), descrito en [47], y que encuentra los caminos mínimos de cada equipo de trabajo a los puntos de reparación.

Cabe mencionar que en este trabajo la regla de transición típica de los algoritmos de hormigas sirve para ayudar a un equipo de trabajo ubicado en un determinado punto de reparación ya reparado a moverse entre los otros posibles puntos de reparación. Funciona del modo descrito a continuación:

$$j = \begin{cases} argmax\Big\{[\tau(i,u)] \times [\eta(i,u)]^{\beta}\Big\}, & \text{si } q \leq q_0 \\ S, & \text{en caso contrario} \end{cases},$$

$$P_k(i,j) = \begin{cases} j, & \text{si } j \in J_k(i) \\ 0, & \text{en caso contrario} \end{cases},$$

donde q es una variable aleatoria uniformemente distribuida en el rango [0,1], q_0 es un parámetro $0 \leq q_0 \leq 1$, S es una variable aleatoria que representa la probabilidad de que una hormiga en el nodo i seleccione el nodo j para moverse, $\tau(i,j)$ es el valor de la feromona en el arco que va desde el nodo i al nodo j, $\eta(i,j)$ es el coste inverso del arco (i,j), $J_k(i)$ es el conjunto de nodos que puede visitar una hormiga k desde el nodo i, β es un parámetro que determina la importancia relativa de la feromona respecto al coste del arco y $P_k(i,j)$ es la probabilidad que una hormiga k elija moverse desde el nodo i al nodo j.

5.3.3. Un algoritmo ACO invertido aplicada al tráfico

En [5] se propone un ACO invertido para la resolución del problema de la congestión del tráfico. Este problema no se asemeja mucho al que estamos abordando en esta TFM, sin embargo, la aplicación original de un ACO nos ha parecido muy interesante e influyente a la hora de desarrollar nuestro propio algoritmo ACO.

En la mayoría de los estudios expuestos hasta el momento, la aplicación del algoritmo ACO solo se utilizaba para resolver el problema clásico de encontrar los caminos óptimos, y se utilizaban otro tipo de algoritmos para el resto de decisiones que se debían tomar. Tanto en este trabajo como en esta TFM, se intenta aplicar el algoritmo ACO de manera original para resolver decisiones que no sean la clásica del camino óptimo.

El algoritmo ACO invertido desarrollado en este paper cambia la lógica básica del algoritmo original, donde las hormigas tienden a seguir el rastro con mayor volumen de feromonas, por una lógica donde las hormigas huyen de los caminos con muchas feromonas evitando así los caminos más congestionados, es decir, los cáminos por donde han pasado mayor número de coches (hormigas) poblados de un mayor volumen de feromonas. Se demuestra que mediante el uso de este sistema se mejora la congestión general de la red, y tanto los conductores que siguen las instrucciones del ACO como los que no mejoran sus tiempos de viaje.

Capítulo 6

Adaptación de los Sistemas de Colonias de Hormigas al problema

En este capítulo explicaremos de forma detallada los pasos seguidos para el desarrollo del algoritmo de SCH de colonias de hormigas que hemos propuesto y la forma en la que lo hemos adaptado al problema. Comenzaremos haciendo una descripción del modelo inicial, a continuación, pasaremos a comentar las mejoras propuestas sobre el mismo, para finalmente, describir en detalle el modelo original que se propone en esta Tesis Fin de Máster.

6.1. Modelo inicial

Muñoz [48], en su propia TFM, propone un adaptación inicial y muy básica del ACS al problema. Este modelo inicial será nuestro punto de partida para desarrollar un modelo más eficiente y eficaz para resolver el problema de *accesibilidad* tratado. En esta sección explicaremos el modelo de partida y las distintas funciones y características del mismo.

El modelo se compone de cuatro fases importantes, la mayoría de ellas heredadas del algoritmo general ACS. En la *fase de inicialización*, se dan los valores iniciales a los parámetros, es especialmente importante la feromona inicial. Posteriormente, se lanzan m hormigas desde cada ciudad $n_2 \in N_2$ formando m caminos desde cada ciudad a su centro regional más cercano. Tras haber construido las hormigas el total de los caminos, se identifican los caminos más cortos y se procede a la fase de reparación.

En la *fase de reparación* se reparan los caminos teniendo en consideración el número de hormigas que han atravesado los mismos y el tiempo ahorrado si son reparadas.

Finalmente, se lleva a cabo la actualización de las feromonas. El proceso descrito se repite de manera iterativa hasta alcanzar la condición de parada. El esquema algorítmico de este proceso tiene la siguiente forma:

Algoritmo 15 Modelo inicial básico

```
Inicializar parámetros, I_b, G_b, τ_0, τ_ij y Δτ_ij
Calcular matriz de distancias T
repeat
    repeat
        Inicializar I_b
        repeat
            Colocar m hormigas en los n nodos
            for k = 1 to m do
                La hormiga k aplica la regla de transción
                La hormiga k aplica la regla local de actualización de feromonas
            end for
        until todas las hormigas han construido una solución completa
        Se aplica la regla de reparación de carreteras
        Se aplica la regla global de actualización de feromonas
        if I_b < best_it then
            I_b = best_it
        end if
    until Se alcanza el número máximo de iteraciones
    if G_b < I_b then
        G_b = I_b
    end if
until se alcanza la condición de parada
```

En dicho algoritmo, I_b y G_b son variables de control para mantener la mejor solución por iteración y la mejor solución global, respectivamente.

A continuación, describiremos en mayor detalle cada una de las fases que conforman el algoritmo.

Fase de inicialización

En la *fase de inicialización* tiene lugar dos procesos de importancia, el cálculo de la *matriz de distancias mínimas* y la elección de la feromona incial, τ_0.

La **matriz de distancias mínimas**, T, es una matriz cuadrada que almacena las distancia más cortas desde un punto a cualquier otro de la red. Así, el elemento T_{ij} representa la menor distancia en unidades de tiempo que se tarda en recorrer el camino desde la ciudad i hasta la j.

Son Maya y Sörensen en [22] quienes proponen la utilización de la matriz de distancias mínimas para el cálculo del ahorro que proporciona arreglar una carretera o para el cálculo de la accesibilidad total de la red, cálculos que sin la ayuda de la matriz T serían computacionalmente muy costosos y poco eficientes. El cálculo de esta matriz se realiza con el algoritmo más eficiente conocido para obtener las distancias mínimas en un grafo, el algoritmo *Floyd-Warshall* [38].

Algoritmo 16 Algoritmo Floyd-Warshall

```
Inicializar path = ∞
for k = 0 to n − 1 do
    for all (i, j) ∈ [0, n − 1] do
        path[i, j] = min(path[i, j], path[i, k] + path[k, j])
    end for
end for
```

El algoritmo *Floyd-Warshall* comienza inicializando a infinito las distancias de cada par de nodos conectados en el grafo. Después, itera sobre el listado completo de nodos escogiendo en cada iteración un nodo k que compara con todas las combinaciones de nodos (i, j) existentes en el grafo comprobando si la distancia de i a k añadiéndole la distancia de k a j es menor que la distancia existente de i a j, en cuyo caso actualiza dicha distancia y modifica el camino haciendo que pase por k.

Una definición clásica de los ACS para la feromona inicial es la que proporciona Dorigo en [17]. Se define el nivel de feromona inicial como $\tau_0 = (\frac{1}{n \times L_{nn}})$, donde n es el número de nodos contenidos en la representación del problema y L_{nn} es la distancia resultante de aplicar la heurística del vecino más próximo.

Sin embargo, en Muñoz [48] se utiliza la definición propuesta por Maya y Sörensen en [22], siendo $\tau_0 = (\frac{1}{n \times IA})$, donde IA es simplemente la accesibilidad de una solución inicial rápida obtenida por el *algoritmo de inserción (IA)*.

Algoritmo 17 Algoritmo de inserción

Inicializar x, B, H y T
repeat
  $saving = 0$
  for $e \in \varepsilon_r | l_e \neq 1 \wedge c_e \leq \hat{B} \wedge m_e \leq \hat{H}$ **do**
    Estimar el ahorro de la inserción para e, $saving(e)$
    if $saving(e) > saving$ **then**
      $saving = saving(e)$
      $candidate = e$
    end if
  end for
  Mejorar *candidate*: Actualizar x, $\hat{\text{B}}$, $\hat{\text{H}}$ y T; $saving = 0$
until no se encuentran más candidatos a insertar

Como se puede observar, el algoritmo empieza con una solución vacía x, donde cada carretera en ε_r se mantiene en su estado dañado y se inicializan los presupuestos B y H. Al iniciar la inserción, la matriz de distancias T se calcula mediante el algoritmo *Floyd-Warshall* descrito en [38]. Después, iterativamente se reparan algunas carreteras para mejorar la accesibilidad total y se continúa hasta que no se pueden realizar más mejoras con el presupuesto restante.

En el momento de reparar es necesario considerar la mejora que produce reparar una carretera en términos de accesibilidad total en la red. El cálculo de la accesibilidad de la red es un proceso bastante costoso computacionalmente, por lo que en Maya y Sörensen [22] se propone una fórmula para calcular el ahorro conseguido en términos de accesibilidad total por reparar una carretera específica e:

$$saving(e) = \sum_{l \in N_2} w_l \Big(min_{k \in N_1}\{T_{kl}\} - \ min_{k \in N_1}\{T_{ki} + f_t(e,1) + T_{jl}\} \Big), \tag{6.1}$$

donde $T[i,j]$ representa el tiempo más corto de viajar de i a j, y la función $f_t(e,1)$ es el tiempo para atravesar la carretera e cuando esta está reparada.

La función de ahorro tiene la siguiente lógica: Para cada población de N_2, se obtiene la distancia desde la población a su centro regional más cercano. A esta distancia, se le resta la distancia desde la población a su centro regional más cercano haciendo pasar el camino por el nuevo tramo arreglado. Si la diferencia es positiva, significa que para esa ciudad el camino hacia el centro regional más cercano ha mejorado.

La diferencia multiplicado por el peso de la ciudad (el número de habitantes) es la ganancia en accesibilidad total para esa ciudad. Más adelante, demostraremos que el modo en que se ha interpretado la función *saving* no es el más indicado. Propondremos por tanto, una fórmula alternativa que representa en mejor medida el ahorro por reparar una carretera expresado en términos de accesibilidad total.

Fase de construcción

La *fase de construcción* tiene lugar del mismo modo que en cualquier algoritmo ACS (veáse la Sección 5.2.2). Las hormigas construyen los caminos de manera estocástica según la *función de transición*, que recordamos tiene la siguiente expresión:

$$P_{ij}^{k} = \begin{cases} argmax_{j \in J_k(i)} \left\{ [\tau_{ij}] \times [\eta_{ij}]^{\beta} \right\}, & \text{si } q \leq q_0 \text{ (explotación)} \\ \frac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum_{k \in A_k} [\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}, & \text{en caso contrario (exploración sesgada)} \end{cases}, \quad (6.2)$$

donde τ_{ij} es la feromona depositada en la arista, η_{ij} la información heurística del camino, β es un parámetro que controla el peso de la información heurística frente al nivel de feromonas, q es un número aleatorio uniformemente distribuido en [0,1] y q_0 es un parámetro ($0 \leq q_0 \leq 1$) que determina la importancia relativa de la explotación respecto de la exploración.

P_{ij}^{k} es, por tanto, la probabilidad de escoger un camino entre los posibles caminos disponibles que tiene una hormiga situada en un nodo k.

La heurística, η_{ij}, es la información a priori que tenemos acerca del problema, por tanto, su definición debe adecuarse al problema considerado. Muñoz [48], para este modelo inicial propone utilizar como medida heurística el *saving* (6.1), es decir, el ahorro en términos de accesibilidad por reparar una carretera. Más adelante en el documento, explicaremos porque el *saving* no es una medida adecuada para esta función y expondremos nuestra solución alternativa.

Fase de reparación

La *fase de reparación* tiene lugar después de la fase de construcción, de manera independiente, y totalmente fuera del algoritmo de hormigas que está teniendo lugar en la fase de construcción.

Una vez las hormigas han construido los caminos, se seleccionan los mejores caminos desde cada ciudad. Las carreteras que pueden ser reparadas, y que están presentes en los caminos de las mejores hormigas, se ordenan en un ranking. Con un enfoque determinista,

se empieza reparando la carretera situada en primera posición y se continúa en orden hasta agotar el presupuesto disponible B y H.

Para establecer el ranking se tiene en consideración los siguientes aspectos: el número de hormigas que han pasado por la carretera en cuestión, el ahorro de tiempo individual M_e y el coste económico c_e y humano m_e que conlleva reparar la carretera.

Nótese que la solución en la que estamos interesados es la solución de reparación, es decir, en el conjunto de carreteras que debemos reparar para maximizar la accesibilidad. Sin embargo, esta solución se obtiene de forma independiente, determinista y fuera del algoritmo de hormigas, por lo que difícilmente el algoritmo convergerá hacia la solución óptima de reparación. Más adelante, mostraremos las formas y modelos propuestos para corregir este inconveniente.

Actualización de feromonas

La actualización de feromonas funciona de manera idéntica al ACS genérico, (véase la Sección 5.2.2). Durante el proceso tienen lugar dos tipos de actualizaciones de feromonas distintas: una *actualización global*, donde solo las mejores hormigas encontradas de manera global contribuyen al rastro de feromonas; y una *actualización local*, que se aplica de manera dinámica en el proceso de construcción de soluciones, de modo que cuando una hormiga atraviesa una carretera se produce una evaporación instantánea de feromona en esa carretera para disminuir la probabilidad de que el resto de hormigas de esa iteración sigan el mismo camino.

Recordamos que la función de actualización local sigue la expresión

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho \times \Delta\tau_{ij},$$

donde $\rho(0 < \rho < 1)$ es un parámetro y $\Delta\tau_{ij} = \tau_0$.

La función de actualización global es:

$$\tau_{ij}(t) = (1-\alpha)\tau_{ij}(t-1) + \alpha \times \Delta\tau_{ij},$$

donde $0 < \alpha < 1$ es el parámetro de evaporación y

$$\Delta\tau_{ij} = min \sum_{i \in \mathcal{N}_2} \left(w_i \times min_{j \in \mathcal{N}_1} \left\{ \sum_{e \in \varepsilon} d_e y_e^{ij} \right\} \right),$$

es decir, $\Delta\tau_{ij}$ es la accesibilidad de la solución encontrada. Recordemos de la modelización matemática del problema (Sección 3), que la *accesibilidad* se definía como la suma ponderada de la longitud de los caminos desde cada ciudad a su centro regional más cercano.

6.2. Mejoras sobre el modelo inicial

Tras haber estudiado el modelo inicial propuesto por Muñoz [48], identificamos posibles mejoras sobre el mismo. En esta sección del capítulo, describiremos las mejoras propuestas y comentaremos las razones que nos han llevado a realizar tales modificaciones.

6.2.1. Mejora de la fórmula del *Saving*

El *saving* (Expresión 6.1), o función de ahorro por reparar un carretera medido en términos de accesibilidad total, es posiblemente la medida que contiene mayor información acerca del problema que queremos resolver. Si el ahorro por reparar una carretera es muy alto, nos indica que posiblemente en nuestra solución final será conveniente reparar esa carretera para maximizar la accesibilidad total. Sin embargo, la definición e interpretación que se realiza de esta función en Muñoz [48] no es la más adecuada.

Recordamos que la fórmula de ahorro o *saving* se define en Muñoz del modo propuesto por Maya y Sörensesn [22], es decir,

$$saving(e) = \sum_{l \in N_2} w_l \Big(min_{k \in N_1}\{T_{kl}\} - \ min_{k \in N_1}\{T_{ki} + f_t(e,1) + T_{jl}\} \Big). \tag{6.3}$$

Del modo definido, se puede observar que si el término a la derecha de la diferencia es mayor al término de la izquierda, entonces el valor que entra en el sumatorio es negativo y tiende a reducir el *saving*. De este modo, el ahorro total en términos de accesibilidad que se produce por arreglar una carretera podría ser negativo. Sin duda, ésta ha sido la manera de interpretar la fórmula que se realiza en Muñoz [48], donde se expresa que en caso de no haber carreteras con *saving* positivo se considerará como mejor carretera aquella con el *saving* negativo más pequeño.

Desde nuestro punto de vista esto es un error de interpretación. Nunca reparar una carretera puede ocasionar una pérdida en términos de accesibilidad total, además, tampoco llevará nunca a que el camino de una ciudad cualquiera a su centro regional más cercano empeore y sea más largo. De este modo, no debería existir nunca un *saving* negativo, el ahorro será positivo si reparar una carretera mejora la accesibilidad de la red y será cero en caso contrario.

Teniendo en consideración lo anteriormente indicado, la fórmula de ahorro o *saving* que nosotros proponemos es la siguiente:

$$saving_{total}(e) = \sum_{l \in N_2} w_l \left(saving_l \right), \tag{6.4}$$

siendo $saving_l$ el ahorro en el camino desde el nodo l a su centro regional más cercano que se produce por reparar la carretera e, y que se puede expresar como

$$saving_l = \begin{cases} Z, & \text{si } Z > 0 \\ 0, & \text{si } Z \leq 0 \end{cases} \tag{6.5}$$

donde

$$Z = min_{k \in N_1}\{T_{kl}\} - \ min_{k \in N_1}\{T_{ki} + f_t(e,1) + T_{jl}\}. \tag{6.6}$$

De este modo, el *saving* total por reparar una carretera se define como la suma de los ahorros que se producen si se mejora el camino de alguna ciudad hasta su centro regional más cercano. En caso de que alguna ciudad no mejore su camino, no significa que se produzca un empeoramiento y, por tanto, su ahorro será nulo.

6.2.2. Mejora del algoritmo de inicialización

Para el cálculo de la feromona inicial, en el modelo de Muñoz [48] se utiliza el *algoritmo de inserción* (IA), propuesto también por Maya y Sörensen [22], el cual obtiene una solución inicial de manera muy rápida. El algoritmo funciona de manera correcta, no obstante, consideramos interesante la mejora del algoritmo de inicialización para obtener una solución inicial rápida de mayor calidad.

Pese a que en el modelo inicial esta solución solo se utiliza para calcular el nivel de feromona inicial, para un modelo más avanzado que se pruebe en problemas de mayor tamaño podría ser interesante obtener una solución de calidad inicial para guiar desde el comienzo la búsqueda del algoritmo de hormigas. Con ese objetivo en mente, se propone un algoritmo de búsqueda secuencial hacia delante, *sequential forward selection* (SFS).

En el algoritmo IA (véase la Expresión 6.1) las carreteras se ordenaban en un ranking según su *saving*, es decir, según el ahorro que se produce en tiempo total por reparar la carretera en cuestión. Una vez realizado el ranking, se comenzaba reparando la carretera situada en primera posición (la de mayor *saving*) y se iba reparando progresivamente hasta agotar el presupuesto disponible. La diferencia básica entre el algoritmo IA y nuestro SFS,

es que nosotros en cada iteración recalcularemos los *saving* para todas las carreteras que no se han reparado. La razón de esta decisión es que el *saving* de una carretera depende del momento exacto en el que se encuentra la red. Cuando se repara una carretera, la red ha cambiado y por tanto los saving por arreglar el resto de carreteras que siguen inoperativas han podido cambiar también. Recalculando los *saving* cada vez que se arregla una carretera se consigue una solución mucho más precisa.

Algoritmo 18 Algoritmo SFS

Inicializar $solution = \emptyset$, B y H
repeat
  Calcular $saving(e)$ para $\forall e \in \varepsilon_r$
  Reparar e con el mayor *saving*
  Añadir e to *solution*
  Actualizar presupuesto restante
until fin de presupuesto

El lector puede pensar que la ganancia en precisión se consigue a costa de un mayor esfuerzo computacional. Pese a que esto es cierto, el esfuerzo computacional de ambos métodos es insignificante, devolviéndose una solución en menos de un segundo.

6.2.3. Mejora de la heurística utilizada

La *fase de construcción* de caminos del modelo inicial (véase la Sección 6.1) se realiza del modo genérico en cualquier ACS, es decir, las hormigas construyen los caminos de manera estocástica siguiendo la *función de transición*.

Como se puede ver en la Expresión (6.2), la hormiga hace uso del nivel de feromonas y de las heurísticas de las carreteras disponibles para tomar la decisión de hacia donde desplazarse. En Muñoz [48] se propone el uso de la medida del *saving* como información heurística, sin embargo, en esta ocasión el *saving* no es una medida adecuada puesto que no se está decidiendo que carretera reparar sino por cual carretera viajar. El *saving* es además una medida que no tiene lógica para una carretera que está en perfectas condiciones y no necesita repararse.

Sin embargo, cuando las hormigas deciden las carreteras por las que desplazarse, en muchas ocasiones (en la mayoría) deben decidir entre carreteras que están operativas y han estado así desde el principio del problema.

Por lo tanto, se debe buscar una heurística que represente la calidad de una carretera en términos de tiempo de viaje, y que además tenga sentido tanto para carreteras operativas como para carreteras dañadas.

En esta Tesis Fin de Máster se propone como medida heurística la inversa de la distancia en unidades de tiempo t_e que se tarda en recorrer una carretera. Utilizar esa medida en la heurística es muy común dentro de los algoritmos ACO. No obstante, la utilización de esta medida plantea un problema cuando se trata de carreteras que están dañadas y pueden por tanto ser reparadas. En este caso, la distancia en unidades de tiempo sería t_e si la carretera se repara y $t_e + M_e$ si no se repara. Recordamos que M_e es un factor de penalización por utilizar una carretera dañada.

Puesto que es una heurística, bastaría con tomar el valor medio de entre t_e y $t_e + M_e$ y asignárselo a esa carretera. No obstante, con la intención de conseguir una medida heurística más precisa se propone la siguiente solución: Las carreteras dañadas recibirán un valor comprendido en el intervalo $[t_e, t_e + M_e]$. Este valor dependerá de la probabilidad que tiene una carretera de ser reparada. Aquellas carreteras con una probabilidad de reparación alta recibirán un valor cercano al lado izquierdo del intervalo, mientras que las carreteras con poca probabilidad de ser reparadas se les asignará un valor cercano al lado derecho del intervalo, es decir, cercano a $t_e + M_e$.

Para calcular la probabilidad a priori de que una carretera sea reparada se utilizará la fórmula de ahorro o *saving*. Aquellas carreteras cuyo *saving* sea alto en comparación con el resto de carreteras de manera proporcional, tendrá entonces mayor probabilidad a priori de ser reparada, y se le asignará un valor de heurística cercano a t_e.

6.2.4. Conexión de la fase de reparación con la fase de construcción

El principal problema del modelo inicial básico propuesto por Muñoz [48] es la desconexión existente entre la fase de reparación y la fase de construcción. Recordemos que en la fase de construcción se utiliza el algoritmo de hormigas para encontrar los caminos óptimos desde cada ciudad a los centros regionales. No debemos olvidar que está solución no es la que nos interesa.

En la fase de reparación, en función del número de hormigas que pasan por las carreteras y otra serie de elementos, se decide qué conjunto de carreteras reparar dado el presupuesto disponible. Es este conjunto de carreteras, el plan de reparación, la solución del problema en la que estamos interesados.

Sin embargo, las soluciones de reparación obtenidas durante el transcurso del algoritmo no afectan, ni influyen, el algoritmo de hormigas en ningún momento, si no que el algoritmo

de hormigas es totalmente independiente de las soluciones de reparación obtenidas y solo se preocupa de encontrar los caminos óptimos.

De este modo, el algoritmo de hormigas jamás convergerá hacia el mejor plan de reparación, como mucho, encontrará caminos rápidos desde las ciudades hacia los centros regionales.

En este punto, proponemos una solución que busca conectar la fase de reparación con la fase de construcción para que las soluciones de reparación encontradas influyan en la dirección que toma algoritmo de hormigas.

Inicialmente, se propone una modificación sencilla que consiste en lo siguiente: Si tras realizar el proceso de reparación de carreteras, el plan de reparación que obtenemos es mejor que la mejor solución obtenida globalmente, entonces, los tiempos de los caminos de las mejores hormigas globales se recalcularán en función de las carreteras reparadas en este nuevo plan. De este modo, se consigue un feedback indirecto desde la fase de reparación hacia el algoritmo de hormigas con el fin de dirigir este hacia el mejor plan de reparación. Nótese en rojo la pequeña diferencia en el Algoritmo 19.

Algoritmo 19 Modelo inicial mejorado

Inicializar parámetros: G_b, τ_0.
Calcular matriz de distancias T
repeat
  repeat
    Colocar las m hormigas en los n nodos
    for $k = 1$ to m **do**
      La hormiga k aplica la regla de transción
      La hormiga k aplica la regla local de actualización de feromonas
    end for
  until todas las hormigas han construido una solución completa
  Se seleccionan las mejores hormigas de cada ciudad
  Se aplica la regla de reparación de carreteras
  if $newSolution$ mejor que G_b **then**
    $G_b = newSolution$
    (Recalcular los caminos de las mejores hormigas globales en función de G_b!)
  end if
  Se aplica la regla global de actualización de feromonas
until se alcanza el número máximo de iteraciones

Más adelante, demostraremos que pese a que esta mejora sirve para resolver instancias pequeñas, no será suficiente para resolver instancias de mayor tamaño. Para conseguir la resolución de instancias grandes se propone en la siguiente sección un modelo original más avanzado que fusiona completamente las labores de construcción y reparación y que, al contrario que este modelo inicial, utilizará todo el potencial del algoritmo de hormigas en la decisión de reparación.

6.2.5. Reparación estocástica

La decisión de reparación en el modelo inicial sobre el que estamos trabajando se realiza con un enfoque determinista. Cuando las hormigas han construido los caminos y se han seleccionado los mejores, se procede a ordenar las carreteras que pueden ser reparadas en un ranking. La posición de una carretera dentro del ranking es función del número de hormigas que pasan por esa carretera en los mejores cáminos encontrados y del ahorro o *saving* en términos de accesibilidad que se produce por arreglar esa carretera. Una vez construido el ranking, de manera determinista, se comienza reparando la carretera situada en primera posición y se continúa reparando paulatinamente hasta agotar el presupuesto disponible.

En este apartado, se propone utilizar un enfoque estocástico que permita obtener soluciones de reparación más diversificadas. Para ello, aparte del número de hormigas que pasan por las carreteras se considera la siguiente *función de probabilidad*:

$$P(e) = \frac{C_e^T}{C_e^T *},$$

donde C_e^T es el beneficio por unidad de coste de arreglar la carretera en cuestión y C_e^T* es el máximo beneficio por unidad de coste de entre todas las carreteras no operativas.

El beneficio unitario o beneficio por unidad de coste viene expresado de la siguiente forma:

$$C_e^T = \frac{saving(e)}{c_e \times \frac{H}{B+H} + m_e \times \frac{B}{B+H}},$$

donde B es el presupuesto económico total, H el presupuesto humano total, c_e el coste monetario de reparar la carretera e en cuestión y m_e el coste humano medido en horas por trabajador.

Nótese que a la hora de calcular el beneficio o ahorro por unidad de coste, se realiza una suma ponderada de los costes de reparación en función de la cantidad de los distintos tipos de presupuesto que tenemos disponibles.

Los costes de reparación de la carretera se multiplican por el porcentaje disponible de presupuesto (del tipo contrario al del coste) frente al total. La lógica bajo esta fórmula reside en que si por ejemplo tenemos mucho más presupuesto humano que económico, entonces, el coste económico c_e deberá tener mucho más peso en la función de coste, puesto que lo más probable es que se agote el presupuesto económico antes que el humano.

Una de las ventajas añadidas de la reparación con el enfoque estocástico es la generación de tantas soluciones de reparación (solución que nos interesa) por iteración como deseemos. Con el enfoque determinista se utilizaban m hormigas para generar m caminos desde cada ciudad, luego se elige el mejor camino desde cada ciudad y se genera una única solución de reparación, un único conjunto de carreteras a arreglar. De este modo, todo el esfuerzo computacional se utiliza en generar m soluciones de caminos distintos (solución que no nos interesa) y solo un plan de reparación. Sin embargo, con el nuevo enfoque estocástico, dado los mejores caminos encontrados se pueden generar estocásticamente tantos planes de reparación como queramos.

Adelantamos, brevemente, que el enfoque estocástico planteado de este modo no consigue buenos resultados. Su problema es que la probabilidad de reparar una carretera es estática y no evoluciona con el algoritmo. Para conseguir converger hacia la solución óptima de reparación es deseable que si al reparar una carretera se obtiene una buena solución, entonces, en la siguiente iteración la probabilidad de reparar esa carretera debería aumentar. En la Sección 6.3, donde se presenta el modelo original propuesto en esta TFM, se mostrará el modo de incorporar una probabilidad de reparación dinámica que guiará el algoritmo hacia el plan óptimo de reparación.

6.2.6. Resultados

Para comprobar la eficiencia de nuestro algoritmo ACS mejorado se utiliza una instancia pequeña del problema de accesibilidad que se encuentra disponible en `https://raw.githubusercontent.com/ekth0r/ndereba/master/resources/OR_SpectrumData/Data/2-40-2-55-25.txt` y la solución obtenida es comparada con la solución obtenida por Muñoz [48] en la misma instancia.

La instancia del problema considerado dispone de 44 nodos, de los cuales 2 son centros regionales (nodos blancos en la Figura 6.1), 40 son ciudades o pueblos rurales (nodos azules) y 2 de ellos son puntos de cruce (nodos verdes). Los nodos están conectados entre si por 55 carreteras, de las cuales 25 están dañadas y pueden ser reparadas (líneas rojas discontinuas).

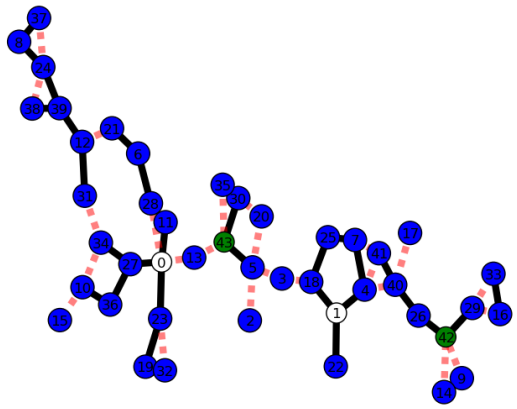

Figura 6.1: Instancia del problema de accesibilidad [48]

La Tabla 6.1 muestra los tiempos que se tardan en atravesar las carreteras (t_e), el coste económico (c_e) y de mano de obra (m_e) asociado a la reparación de cada carretera $e \in \epsilon_r$. Los presupuestos disponibles son $B = 74$ unidades monetarias y $M = 83$ horas de trabajo. El factor de penalización por usar una carretera dañada, M_e, es 10.

Los parámetros que hemos utilizado para el ACS mejorado son los mismos que se utilizan para el modelo inicial de Muñoz y son comunes dentro de los ACS: $\beta = 2$, $\alpha = \varphi = 0'1$ (parámetros de evaporación global y local). El número de hormigas es 10 y el número de iteraciones es 80.

En una primera prueba, se utiliza nuestro algoritmo mejorado pero con el enfoque determinista. En la solución óptima encontrada por nuestro algoritmo mejorado las carreteras reparadas son [17, 25, 37, 11, 45, 6, 12, 9, 34, 38, 39, 44, 1, 4, 5, 32] y el tiempo total de travesía desde cada ciudad al centro regional más cercano es 711304'7 unidades de tiempo (véase el mapa reparado en la Figura 6.2). Nuestra solución mejora notablemente a la solución que obtiene el modelo inicial en Muñoz [48], donde las carreteras reparadas son [17, 37, 6, 46, 23, 45, 32, 12, 11, 38, 39, 48, 24, 34, 4, 44, 9] y el tiempo total de travesía es de 715554'7 unidades de tiempo.

Puesto que nuestra solución supera a la obtenida por Muñoz, con el objetivo de ver cuán buena es la solución encontrada, se resuelve la instancia de manera exacta. Para ello, se aborda el problema a fuerza bruta comprobando cada solución factible, proceso que nos lleva 28 horas. La resolución exacta del problema nos revela que la solución óptima del problema es la misma que nosotros hemos encontrado.

Cuadro 6.1: Tiempos de travesía y costes financieros y de mano de obra

e	t_e	c_e	m_e	e	t_e	c_e	m_e	e	t_e	c_e	m_e
(1)	243.3	10.5	2.0	(20)	31.3	4.8	6.8	(38)	270.9	2.0	2.5
(2)	99.0	6.5	3.1	(21)	145.7	-	-	(39)	254.3	2.3	2.9
(3)	202.6	6.4	7.2	(22)	112.6	-	-	(40)	173.7	-	-
(4)	135.2	2.2	4.8	(23)	29.1	6.1	10.3	(41)	11.7	-	-
(5)	53.4	8.1	7.4	(24)	88.1	4.5	7.0	(42)	115.4	-	-
(6)	176.8	4.9	3.4	(25)	227.7	7.0	11.9	(43)	228.7	-	-
(7)	214.1	-	-	(26)	181.9	-	-	(44)	132.3	3.3	3.6
(8)	79.1	-	-	(27)	91.0	-	-	(45)	94.1	0.0	4.0
(9)	112.1	4.0	5.5	(28)	64.2	-	-	(46)	192.0	4.2	1.0
(10)	111.4	-	-	(29)	105.7	-	-	(47)	85.4	-	-
(11)	123.0	13.3	12.4	(30)	70.7	11.2	6.8	(48)	46.3	1.6	3.8
(12)	191.3	3.1	5.5	(31)	75.3	4.4	0.6	(49)	151.2	-	-
(13)	135.4	-	-	(32)	107.1	3.7	4.7	(50)	117.7	-	-
(14)	116.6	-	-	(33)	166.3	-	-	(51)	225.6	-	-
(15)	82.8	-	-	(34)	53.9	1.2	4.7	(52)	51.6	-	-
(16)	215.0	-	-	(35)	36.9	-	-	(53)	67.7	-	-
(17)	72.0	3.3	3.1	(36)	71.6	-	-	(54)	137.5	-	-
(18)	102.6	-	-	(37)	128.8	4.8	4.2	(55)	177.4	-	-
(19)	88.9	-	-								

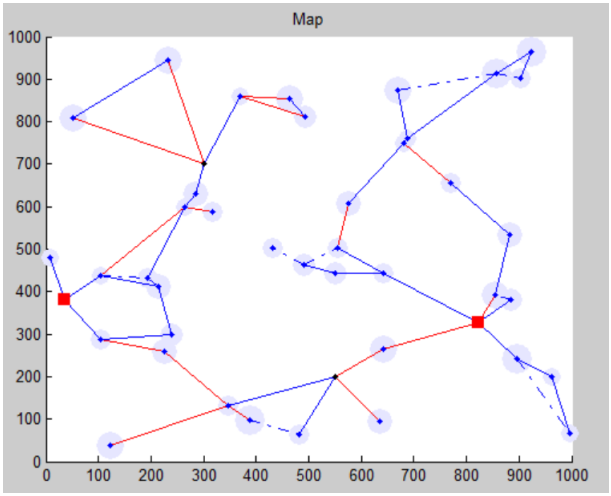

Figura 6.2: Instancia resuelta con las carreteras reparadas en rojo

En una segunda prueba, se utiliza el enfoque estocástico (Sección 6.2.5), y como ya se adelantaba, los resultados obtenidos no son buenos. El problema bajo este enfoque, es que la probabilidad de reparar una carretera es estática y no evoluciona con el algoritmo. Para conseguir converger hacia la solución óptima de reparación es deseable que si al reparar

una carretera se obtiene una buena solución, entonces, en la siguiente iteración la probabilidad de reparar esa carretera debería aumentar. En la siguiente sección, se presenta el modelo original propuesto en esta TFM, donde se muestra el modo de incorporar una probabilidad de reparación dinámica que guiará el algoritmo hacia el plan óptimo de reparación.

6.3. Modelo original propuesto: ACS con doble hormiga

En la sección anterior vimos que el modelo inicial con nuestras mejoras implementadas era capaz de resolver una instancia pequeña del problema de accesibilidad. Sin embargo, en instancias grandes los resultados obtenidos no son buenos. Destacábamos dos grandes problemas por los que el modelo anterior fracasaba:

- La decisión de reparación y la de obtener los caminos óptimos eran independientes. El potencial de búsqueda del algoritmo de hormigas se utilizaba para la búsqueda de caminos y no para la búsqueda del plan óptimo de reparación.
- Las mejoras propuestas y el enfoque estocástico planteado con el fin de guiar la búsqueda hacia la mejor solución de reparación no son suficientes. El principal problema del enfoque estocástico anterior es que la probabilidad de reparar una carretera no evoluciona dinámicamente con el algoritmo.

Llegados a este punto se decide romper con el modelo inicial propuesto en Muñoz [48] y plantear un nuevo modelo que resuelva la problemática del modelo anterior, donde el potencial de los algoritmos de hormigas se utilice para encontrar el plan óptimo de reparación y no exclusivamente para buscar caminos. En esta Tesis Fin de Máster se propone una nueva adaptación de los ACS al problema, un modelo al que hemos llamado *ACS con doble hormiga*.

6.3.1. Idea general del modelo

La propiedad inherente del algoritmo *ACS con doble hormiga* es la utilización de dos tipos de hormigas distintos. Una primera hormiga, la *exploradora*, tiene la labor de encontrar los caminos más rápidos desde las distintas ciudades a los centros regionales más cercanos. La segunda hormiga, la *trabajadora*, tomará la decisión de qué carreteras reparar y buscará el plan óptimo de reparación que maximice la accesibilidad de la red.

A diferencia del modelo inicial, donde ambas labores se realizaban de forma independiente, en este nuevo modelo ambas hormigas trabajarán a la vez y siempre en pareja para construir la solución de caminos y de reparación al mismo tiempo.

El modelo está bioinspirado en las colonias de hormiga *Atta*, conocida comúnmente como la hormiga *cortadora de hojas* y procedente de Costa Rica. Estas hormigas trabajan en pareja, mientras una de las hormigas porta la hoja, otra más pequeña se sube a lomos de la hoja para inspeccionar la calidad de la misma y proteger a la hormiga portadora de un tipo de mosca parasitaria. Para el caso que nos ocupa debe quedar claro que puesto que las hormigas son de distinto tipo, las feromonas que guían su comportamiento también son distintas y utilizaremos variables diferentes para almacenarlas.

La idea básica del algoritmo es la siguiente: En primer lugar, se inicializan los valores de las feromonas, que serán distintas para cada clase de hormiga. Posteriormente, en la *fase de construcción* de soluciones se lanzan m parejas de hormigas que encontrarán los caminos más rápidos desde cada ciudad a los centros regionales y decidirán el plan óptimo de reparación para la red. De este modo, se construyen m caminos desde cada ciudad a su centro regional más cercano y m planes de reparación, uno por cada pareja de hormigas. Dentro de la pareja, la hormiga *exploradora* comienza construyendo los caminos del modo común, es decir, siguiendo la *regla de transición*, en el momento en que la *exploradora* decide avanzar por una carretera dañada, la *trabajadora* entra en juego y decide si se repara la carretera o no se repara.

Por supuesto, la hormiga *trabajadora* tendrá en consideración el presupuesto disponible, y si este se agota no se reparará ninguna carretera más.

Para que el algoritmo tenga el funcionamiento deseado y se consiga diversidad de soluciones, el orden de las ciudades donde comienza la pareja de hormigas es totalmente aleatorio. De este modo, se evita que las primeras carreteras reparadas sean siempre las mismas por el simple hecho de ser las primeras que se visitan.

Una vez que las m parejas de hormigas han construido las soluciones, se seleccionan los mejores caminos encontrados y el mejor plan de reparación y se comparan con las mejores soluciones globales. Si las nuevas soluciones son mejores que las anteriores, se actualiza las mejores soluciones.

Como se propuso en la Sección 6.2.4, la longitud de los mejores caminos se calcula en función de la carreteras reparadas en la mejor solución de reparación encontrada hasta el momento. Esto aumenta la relación entre la solución de los mejores caminos y la solución del mejor plan de reparación. Por último, se procede a la actualización de feromonas, donde solo las mejores hormigas depositan feromonas. La *exploradora* actualizará sus feromonas según los mejores caminos encontrados y la *trabajadora* lo hará según el mejor plan de

reparación encontrado, es decir, el que maximice la accesibilidad de la red. Nótese que al igual que en los ACS típicos, en la fase de construcción de soluciones tiene lugar una actualización local y dinámica de las feromonas.

Algoritmo 20 ACS con doble hormiga

Inicializar parámetros: $\tau_0^{explorer}$, τ_0^{worker}, T.
repeat
  for m parejas de hormigas **do**
    Se inicializa el presupuesto B y H
    for $n \in N_2$ ciudades, elegido el orden al azar **do**
      repeat
        La hormiga $k^{explorer}$ aplica la regla de transición
        La hormiga $k^{explorer}$ aplica la regla local de actualización de feromonas
        if se encuentra una carretera dañada **then**
          La hormiga k^{worker} aplica la regla de reparación
          La hormiga k^{worker} aplica la regla local de actualización de feromonas
          La hormiga k^{worker} actualiza el presupuesto
        end if
      until se ha construido el camino para la ciudad n elegida aleatoriamente
    end for
  end for
  Se seleccionan la mejores hormigas *explorer* de cada ciudad
  Se selecciona la mejor hormiga *worker*
  if $bestIterationWorker$ mejor que $bestGlobalWorker$ **then**
    Actualizar $bestGlobalWorker$
    Recalcular caminos de $bestGlobalExplorer$ en función de $bestGlobalWorker$
  end if
  if $bestIterationExplorer$ mejor que $bestGlobalExplorer$ **then**
    Actualizar $bestGlobalExplorer$
  end if
  Se aplica la regla global de actualización de feromonas para la hormiga *explorer*
  Se aplica la regla global de actualización de feromonas para la hormiga *worker*
until se alcanza el número máximo de iteraciones

A continuación, comentaremos cada una de las fases del algoritmo en mayor detalle.

6.3.2. Fase de inicialización

En este apartado explicaremos como se calcula el nivel inicial de feromona para cada tipo de hormiga $\tau_0^{explorer}$ y τ_0^{worker}. Es especialmente interesante la proposición de una *matriz de feromonas múltiple* para el caso de las hormigas exploradoras.

Cálculo de la feromona inicial para la hormiga exploradora

En el modelo inicial que se propuso en Muñoz [48] se utilizaba una única matriz de feromonas para crear los caminos desde cada una de las ciudades a sus centros regionales más cercanos. La realidad es que lo que parece un problema único, es un conjunto de problemas independientes entre sí.

El problema de encontrar el camino más rápido desde una ciudad i a su centro regional más cercano es totalmente independiente al problema de encontrar el camino más rápido desde una ciudad $j \neq i$ a su centro regional más cercano. Utilizar una carretera a podría ser una buena decisión para ir de i al centro regional, pero ser una mala decisión si partiésemos de la ciudad j. Teniendo esto en consideración no es conveniente la utilización de una única matriz de feromonas para resolver el problema conjunto.

Nosotros proponemos la utilización de una **matriz de feromonas múltiple** para la resolución del problema conjunto. Esto quiere decir que utilizaremos tantas matrices de feromonas como ciudades haya en la red, y estas serán independientes unas de otras. De este modo, si hay n ciudades o pueblos en la red, resolveremos n problemas independientes con n matrices de feromonas distintas, donde un problema independiente es encontrar el camino óptimo de una ciudad n a su centro regional más cercano.

Cada una de las matrices de feromonas independientes, que forman en conjunto lo que hemos llamado *matriz de feromonas múltiple*, tendrá por tanto su propio valor de feromona inicial.

El nivel de feromona inicial para cada una de las matrices se calcula según la siguiente fórmula:

$$\tau_0^{explorer} = \left(\frac{1}{m \times L_n} \right), \tag{6.7}$$

donde m es el número de hormigas y L_n es la longitud del camino de la ciudad n a su centro regional más cercano, calculado con el algoritmo *Floyd-Warshall* y suponiendo que ninguna de las carreteras ha sido reparada.

Cálculo de la feromona inicial para la hormiga trabajadora

Para el cálculo de la feromona inicial de la hormiga trabajadora se utiliza una expresión parecida a la anterior. No obstante, las variables utilizadas no son relativas a distancia sino que están relacionadas con la accesibiliad de la red. De este modo, conseguimos que el cálculo de las feromonas y las variables que se utilizan para ello estén relacionadas con el tipo de decisión o solución que se está buscando. Sin duda, esto será muy conveniente a la hora de resolver el problema, pues las feromonas fluirán en concordancia con el tipo de problema que se está resolviendo.

La expresión para el cálculo del nivel de feromona inicial para la hormiga trabajadora queda

$$\tau_0^{worker} = \left(\frac{1}{m \times A}\right), \tag{6.8}$$

donde A es la accesibilidad de una solución inicial subóptima calculada de manera rápida con el algoritmo SFS que propusimos en la Sección 6.2.2.

6.3.3. Hormiga exploradora

Como ya hemos indicado previamente, la función de la hormiga exploradora es obtener los caminos óptimos desde cada ciudad o pueblo hacia su centro regional más cercano. El funcionamiento para esto es el genérico en los algoritmos ACS, donde la hormiga construye los caminos siguiendo la *función de transición*:

$$P_{ij}^k = \begin{cases} argmax_{j \in J_k(i)} \left\{ [\tau_{ij}] \times [\eta_{ij}]^\beta \right\}, & \text{si } q \leq q_0 \text{ (explotación)} \\ \frac{[\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}{\sum_{k \in A_k} [\tau_{ij}(t)]^\alpha \times [\eta_{ij}]^\beta}, & \text{en caso contrario (exploración sesgada)} \end{cases}.$$

En esta expresión, que hemos visto en múltiples ocasiones a lo largo del documento, lo interesante es la definición que se utiliza para la información heurística η.

Para la hormiga exploradora la información heurística utilizada es la misma que se propone en la Sección 6.2.3, es decir, la inversa de la distancia en unidades de tiempo t_e que se tarda en cruzar la carretera si está operativa. Si la carretera está dañada, entonces se le asigna un valor comprendido en el intervalo $[t_e, t_e + M_e]$ según la probabilidad de ser reparada, que se estimaba según la fórmula de ahorro o *saving*. No se debe olvidar que en la fórmula de transición, la feromona τ depende del nodo o ciudad del que hemos partido al comienzo del camino, puesto que cada ciudad tiene su propia matriz de feromonas.

6.3.4. Hormiga trabajadora

La hormiga trabajadora es la estrella del algoritmo, se encarga de encontrar el plan óptimo de reparación de carreteras que maximice la accesibilidad de la red. En cualquier ACO, las hormigas construyen las soluciones eligiendo entre los distintos caminos disponibles según la *fórmula de transición*, pero la decisión de reparar una carretera o no, no supone la elección de caminos en sí.

Para que el funcionamiento de la hormiga trabajadora sea equivalente a una hormiga de un ACO común, se propone modelar la decisión de reparación de modo que tome forma de caminos. Uno de los caminos conllevará la decisión de reparar y el otro camino codifica la decisión de no reparar. El funcionamiento se hace evidente en la Figura 6.3.

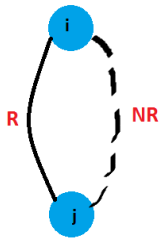

Figura 6.3: Modelización de la decisión de reparación

Por tanto, cuando la hormiga trabajadora debe decidir si reparar una carretera que va desde un nodo i a un nodo j, lo que hace es decidir entre dos posibles caminos: Un camino donde se repara la carretera (linea continua en la figura) o un camino donde no se repara (linea discontinua). Con esta forma de modelizar la decisión podemos entonces hacer uso de la regla de transición típica de un ACO, que para este caso la llamaremos *regla de reparación*:

$$P_{ij}^{k} = \begin{cases} argmax_{j \in J_k(i)} \left\{ [\tau_{ij}] \times [\eta_{ij}]^{\beta} \right\}, & \text{si } q \leq q_0 \\ \frac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum_{k \in A_k} [\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}, & \text{en caso contrario} \end{cases} .$$

En esta ocasión, τ_{ij} en la fórmula es τ_{ij}^{worker} y η_{ij} es η_{ij}^{worker}. La información heurística para la hormiga trabajadora se define de forma distinta a la información heurística de la hormiga exploradora. Nos interesa que esté expresada en términos de accesibilidad y no en términos de longitud de caminos, puesto que la decisión que se está tomando tiene como objetivo maximizar la accesibilidad de la red.

El camino que codifica la decisión de no reparar tendrá como valor heurístico la accesibilidad de la red cuando ninguna carretera ha sido reparada. Por ende, el valor heurístico del camino que representa la decisión de reparar será la accesibilidad de la red cuando solo la carretera en cuestión es reparada, es decir, la accesibilidad de la red sin reparar más la accesibilidad extra que genera reparar la carretera calculado con la fórmula del *saving* (Expresión 6.4).

6.3.5. Actualización de feromonas

La actualización de feromonas funciona de manera similar a un ACO genérico (véase la Sección 5.2.2). En nuestro algoritmo, utilizamos dos tipos distintos de hormiga, la *exploradora* y la *trabajadora* y, por tanto, debemos actualizar ambas feromonas por separado.

Recordamos la expresión genérica de actualización global de feromonas:

$$\tau_{ij}(t) = (1-\alpha)\tau_{ij}(t-1) + \alpha \times \Delta\tau_{ij},$$

donde $0 < \alpha < 1$ es el parámetro de evaporación global y

$$\Delta\tau_{ij} = \frac{1}{C(S_{bestGlobal})}.$$

Para el caso de la hormiga exploradora $\tau_n^{explorer}$, $C(S_{bestGlobal})$ es la longitud del camino más rápido encontrado hasta el momento entre la ciudad n y su centro regional más cercano. Recordamos en este punto que para la búsqueda de caminos se utilizaban múltiples matrices de feromona, una para cada ciudad n. Cuando se trata de la hormiga trabajadora τ^{worker}, $C(S_{bestGlobal})$ es la accesibilidad del mejor plan de reparación encontrado hasta el momento.

Para la expresión de la actualización local de feromonas, que se aplicada de manera dinámica durante el proceso de construcción de soluciones, se utiliza la expresión genérica:

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t-1) + \rho \times \Delta\tau_{ij},$$

donde $(0 < \rho < 1)$ es el parámetro de evaporación local y $\Delta\tau = \tau_0^{explorer}$ si se trata de la hormiga exploradora y $\Delta\tau = \tau_0^{worker}$ si actualizamos las feromonas de la hormiga trabajadora.

6.4. Implementación

El lenguage de programación utilizado en la presente Tesis Fin de Master ha sido el lenguaje M, propio de Matlab, en concreto hemos usado la versión R2013b. Nos hemos decantando por este entorno de programación debido a las facilidades que incorpora para el manejo de matrices y sus respectivos cálculos. Sumado a esto último, la extensa experiencia que tenemos en el uso de este lenguaje ha inclinado la balanza más a su favor.

La arquitectura general del sistema está formada por tres partes diferenciadas: un *acceso a datos*, que se compone de ficheros de texto que son parseados y transformados en estructuras de datos que la aplicación comprende; un módulo *Main*, que es el punto de entrada al sistema y; finalmente, un objecto *ACS*, que ejecuta la metaheurística para resolver el problema tratado.

El módulo *Main*, como ya se ha comentando, es el punto de entrada del sistema. En este módulo se define la instancia del problema de accesibilidad que se desea resolver, al igual que el conjunto de parámetros del algoritmo de resolución que queramos utilizar. De momento, los únicos métodos de resolución disponibles son la metaheurística ACS propuesta en este trabajo y un método que resuelve el problema por fuerza bruta.

El módulo *ACS* es el responsable de ejecutar la lógica de la metaheurística para resolver la instancia del problema de accesibilidad que se ha definido en el modulo *Main*. Cuando ACS obtiene una solución, devolverá la misma al modulo *Main* para posterior análisis, obtención de estadísticas o para mostrar salidas gráficas.

El módulo *acceso a datos* se encarga de transformar los datos originales que definen el problema y que son un conjunto de ficheros de texto, en un formato o estructura de datos que el módulo ACS pueda entender. Cada fichero de texto almacena el conjunto de ciudades o pueblos, centros regionales, carreteras, presupuestos disponibles y demás variables que definen el problema. La sintaxis utilizada en los ficheros de texto es la siguiente:

```
{número de centros regionales} //Number of Regional Centers
{número de ciudades} //Number of intermediate cities
{número de intersecciones} //Number of cross points
{número de carreteras} //Number of arcs
{presupuesto económico} //Budget
{presupuesto humano} //Budget team hours
// NODES: Matrix [Nodes][3] (id, type, weight, Xcoord, Ycoord)
{id nodo 1} {id tipo nodo 1} {peso nodo 1} {coordenada X1} {coordenada Y1}
{id nodo 2} {id tipo nodo 2} {peso nodo 2} {coordenada X2} {coordenada Y2}
{id nodo 3} {id tipo nodo 3} {peso nodo 3} {coordenada X3} {coordenada Y3}
```

```
...

// ROADS: Matrix [nRoads][5] (id, head, tail, lenght, status)
{id carretera 1} {nodo inicial 1} {nodo final 1} {tiempo} {estado}
{id carretera 2} {nodo inicial 2} {nodo final 2} {tiempo} {estado}
{id carretera 3} {nodo inicial 3} {nodo final 3} {tiempo} {estado}
...

// REPAIR: Matrix [DamagedRoads, 3] (id, cost, Hman_req).//
{id carretera 1} {coste económico} {coste humano}
{id carretera 2} {coste económico} {coste humano}
{id carretera 3} {coste económico} {coste humano}
...
```

Capítulo 7

Ejemplo de aplicación

Para comprobar la eficiencia de nuestro algoritmo *ACS con doble hormiga* bajo condiciones reales hemos utilizado una instancia de gran tamaño del problema de accesibilidad que simula el desastre natural ocurrido en Haití. Nuestra solución es comparada con la obtenida en la misma instancia por un algoritmo que mezcla las metaheurísticas GRASP y VNS propuesto por Maya y Sörensen [22] (véase la Sección 4.3.1 para una descripción del algoritmo).

En unas pocas semanas, 4 huracanes y tormentas tropicales golpearon a principios de Septiembre 2008 el país de Haití. De acuerdo con los informes oficiales presentados en OCHAN-UN [49] y OMP [50], 800.000 personas se vieron afectadas entre otros motivos, por las numerosas vías principales bloqueadas o puentes destruidos, que dificultaron las labores de rescate y los procesos logísticos.

En [22] se crea esta instancia basada en los datos de multitud de fuentes. Se usaron los datos de *GISDataDepot* (http://data.geocomm.com) para definir la red de carreteras. El estado de la red después del desastre natural se obtuvo de información pública en *Mapaction* (www.mapaction.org) y *Reliefweb* (www.reliefweb.com). La información demográfica se tomó de *Falling Rain Genomics Inc.* (www.fallingrain.com). Finalmente, por la incapacidad de encontrar los costes y tiempos necesarios para reparar cada carretera, estos se definieron de manera lineal en proporción a la longitud de la carretera.

La instancia generada de esta forma se compone de 216 nodos, de los cuales 103 corresponden con ciudades o pueblos rurales. La red esta formada por 281 carreteras, de las cuales 30 fueron dañadas. La Figura 7.1 muestra la red aquí descrita. Las ciudades que se han elegido como centros regionales aparecen referenciadas en el mapa, son Port Au-Prince, Les Cayes y Cap Haitien.

Se definen tres escenarios distintos para el presupuesto disponible, representando el 25 %, 50 % y 75 % del total necesario para reparar todas las carreteras dañadas de la red. La penalización M_e por utilizar una carretera dañada es idéntica para todas las carreteras y su valor se define como la suma de la longitud del conjunto de carreteras dañadas. En el Apéndice A se pueden encontrar las tablas correspondientes a los tiempos de travesía y los costes financieros y de mano de obra por reparar cada carretera.

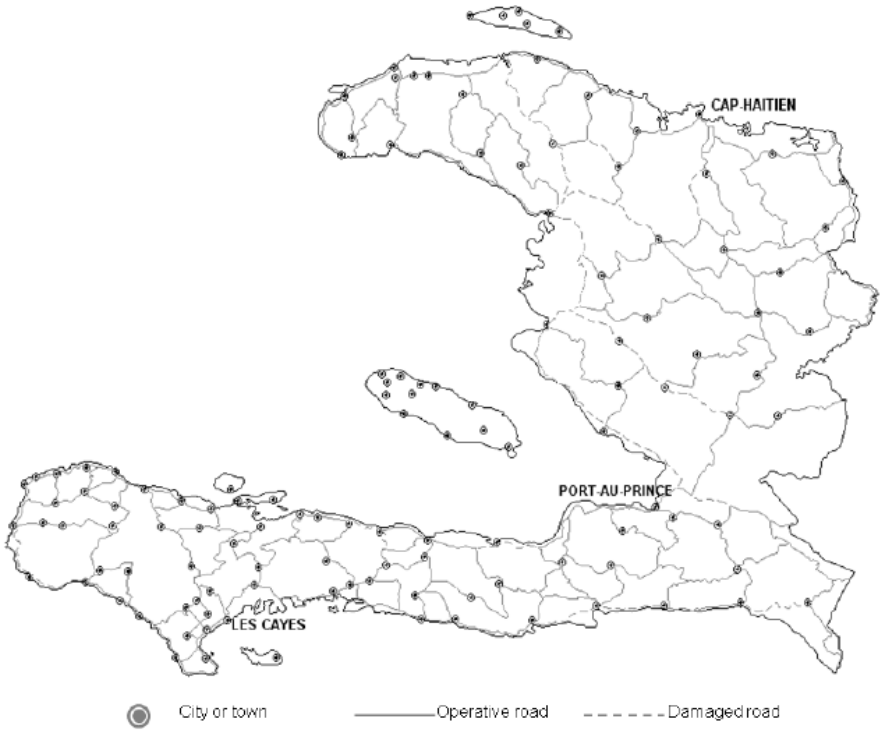

Figura 7.1: Red de carreteras para la instancia de Haití [48]

Los parámetros utilizados en nuestro algoritmo *ACS con doble hormiga* para la resolución del problema son: $\beta = 2$, $\alpha = \varphi = 0'1$ (parámetros de evaporación global y local), $q = 0'1$ (parámetro de balance entre explotación y exploración), el número de hormigas es 10 y el número de iteraciones es 100.

Las soluciones que obtuvimos se describen a continuación. Para el escenario en el que se dispone del 25 % del presupuesto (B=30; H=32) el plan óptimo de reparación conlleva reparar las carreteras [23, 5, 3, 22, 25, 7, 24] y el tiempo de travesía total desde cada ciudad a su centro regional más cercano es de 125.449.501'62 unidades de tiempo.

Para el escenario en el que tenemos el 50 % del presupuesto (B=60; H=64) se reparan las carreteras [5, 24, 20, 25, 22, 3, 27, 7, 23, 1, 2, 9, 11] y el tiempo total de travesía es de 121.055.243'5 unidades de tiempo.

Finalmente, para el 75 % del presupuesto (B=90; H=96) se repara el conjunto de carreteras [25, 22, 23, 24, 5, 27, 11, 1, 7, 29, 3, 2, 20, 29] y la accesibilidad en unidades de tiempo es de 120.995.327'08.

Puesto que en Maya y Sörensen [22] no se facilita el plan de reparación obtenido, con la finalidad de poder comparar nuestras soluciones con las obtenidas por ellos, expresamos nuestras soluciones en términos de ciudades y personas afectadas tras la reparación, del mismo modo que lo hacen ellos. Las soluciones que nosotros hemos obtenido se indican en la Tabla 7.1.

Cuadro 7.1: Ciudades y personas afectadas

	Sin reparar	25 %	50 %	75 %
Ciudades afectadas	31	14	1	0
Personas afectadas	774.432	398.631	8.163	0

Tras comparar los resultados con los de la metaheurística GRASP y VNS de Maya y Sörensen podemos concluir que los resultados obtenidos con nuestro algoritmo *ACO de doble hormiga* son exactamente los mismos.

Finalmente, para averiguar cuan buena es la solución que obtenemos, hacemos un esfuerzo por resolver el problema de Haití de manera exacta (fuerza bruta) en su instancia más sencilla (25 % del presupuesto), proceso que acarrea más de 5 días de computación.

La resolución exacta que obtenemos nos revela que la solución que se obtiene por medio del algoritmo de hormigas es la solución óptima del problema, y por tanto no se puede mejorar. Del mismo modo, se supone que para los otros escenarios de presupuesto la solución que estamos obteniendo es la solución óptima del problema.

Una vez más, se hace evidente la necesidad de seguir desarrollando e investigando metaheurísticas para la resolución de problemas de este tipo. Nuestro algoritmo consigue devolver la solución óptima del problema en menos de 10 minutos para la instancia más compleja.

Capítulo 8

Conclusiones y Líneas futuras de investigación

En este último capítulo, echamos la vista atrás para analizar el nivel de logro de los distintos objetivos marcados para esta Tesis Fin de Máster (TFM). Además, en base a los resultados obtenidos y los problemas encontrados, proponemos una serie de potenciales líneas futuras de investigación para seguir mejorando el funcionamiento del algoritmo propuesto y su aplicación en el problema de la *accesibilidad* o similares.

El objetivo principal de esta TFM era el de aplicar de manera eficiente los sistemas de colonias de hormigas (ACS) al problema de la *accesibilidad*, mejorando el modelo inicial propuesto en Muñoz [48]. El objetivo se ha alcanzado con mejores resultados de los esperados. No solo se ha mejorado el modelo inicial de Muñoz, sino que se ha presentado un nuevo modelo original, el *ACS con doble hormiga*, que utiliza todo el potencial de los algoritmos de hormigas, y que como se demuestra en el Capítulo 7 consigue resolver eficientemente el problema de la *accesibilidad* para instancias grandes.

A continuación, se repasa la consecución de los distintos subojetivos marcados en el Capítulo 2 (analizar y comprender el problema de accesibilidad y su modelización matemática, estudiar el uso de metaheurísticas, analizar el modelo inicial de Muñoz, realizar el ajuste paramétrico apropiado y comparar nuestro algoritmo con el de otros autores en instancias grandes).

El primer subobjetivo era analizar y comprender el problema de accesibilidad y su modelización matemática. En la realización de esta TFM se ha obtenido un nivel de maestría en el problema de la accesibilidad. La compresión de los entresijos más internos del problema ha permitido proponer expresiones matemáticas que representen el problema de manera fiel, es el caso de la fórmula de ahorro o *saving* introducida en la Sección 6.2.1.

En el Capítulo 4, se realiza una descripción amplia de las metaheurísticas presentes en la literatura y se resume el estado del arte de la aplicación de las mismas al problema considerado. Seguidamente, en el Capítulo 5, se describe más detalladamente los algoritmos de hormigas y su uso en la resolución de problemas similares al de accesibilidad. Por tanto, se da por conseguido el segundo subojetivo, estudiar el uso de metaheurísticas.

El tercer subobjetivo consistía en el análisis del modelo inicial de Muñoz. En la Sección 6.1 se describe de manera detallada el mismo. En su estudio se identifican los inconvenientes del modelo y, se proponen alternativas y mejoras sobre el mismo en la Sección 6.2. La comprensión de las limitaciones de este modelo incluso en su versión mejorada, nos sirven de guía para desarrollar nuestro nuevo modelo, el *ACS con doble hormiga*.

En cuanto a la configuración de parámetros, pese a no haber desarrollado un sistema automático de configuración de los mismos, estos se han ajustado de manera manual por prueba y error sobre los valores que se recomiendan en el artículo original de ACS [17]. No obstante, pese a que los resultados obtenidos son buenos, se considera que existe margen de mejora en este aspecto.

Finalmente, en el Capítulo 7, se prueba nuestro algoritmo *ACS con doble hormiga* en una instancia real y de gran tamaño que simula el desastre natural ocurrido en Haití, y se muestran los resultados obtenidos en comparación con los obtenidos por una metaheurística basada en GRASP y VNS [22]. Los resultados que obtenemos igualan con los obtenidos por la metaheurística GRASP/VNS de Maya y Sörensen, habiéndose alcanzado la optimalidad en los mismos.

A pesar de los excelentes resultados obtenidos en esta TFM, que han superado las expectativas iniciales, todavía existe margen de mejora en algunos aspectos del algoritmo y su aplicación. A continuación, se proponen posibles líneas de trabajo futuro para seguir desarrollando el modelo y mejorando su comportamiento.

Sin duda el punto menos desarrollado del algoritmo es el de ajuste de parámetros. La mejora en este aspecto se podría dar en dos líneas distintas. La primera, sería el desarrollo de un sistema automático de configuración paramétrico que potenciara el comportamiento del algoritmo. La segunda, tiene que ver con la mejora de la interacción entre la hormiga *exploradora* y la *trabajadora*. Actualmente, los parámetros de evaporación de feromonas local y global, así como el parámetro q del balance entre explotación y exploración, son compartidos para ambas hormigas. Sin embargo, podría ser interesante que cada una de las hormigas tuviese sus parámetros de forma independiente. Esto permitiría, por ejemplo, potenciar la explotación en el proceso de búsqueda de caminos rápidos (hormiga exploradora), ya que no estamos interesados en obtener los caminos óptimos y una so-

lución subóptima buena sería satisfaciente, y sin embargo, favorecer la exploración y la diversificación en la búsqueda del plan óptimo de reparación (hormiga trabajadora) para asegurarnos de no quedar atrapados en óptimos locales.

Para un mayor control del nivel de feromonas se propone la implementación de un sistema de hormigas MAX-MIN (MMAS) y su comparación contra el actual sistema de colonia de hormigas (ACS). Durante las pruebas realizadas con ACS, se ha observado que en ocasiones los niveles de feromonas de las distintas carreteras se diferencian con demasiada velocidad, haciendo que sea probabilísticamente imposible la reparación de algunas carreteras. En el sistema MMAS el nivel de feromonas se mantiene dentro del intervalo $[\tau_{min}, \tau_{max}]$ favoreciendo la diversificación de soluciones y potenciando el proceso de exploración.

La línea de investigación que nos resulta más interesante es la aplicación del sistema *ACS con doble hormiga* en otro tipo de problemas distinto al de la accesibilidad. En esta TFM se ha demostrado la posibilidad de utilizar algoritmos de colonias de hormigas para decisiones que no solo conllevan la elección de los caminos más rápidos. Por ende, parece razonable su aplicación en otros problemas.

Otra importante línea de investigación será la comparación de manera más concienzuda de nuestro *ACS con doble hormiga* frente a la metaheurística GRASP y VNS de Maya y Sörensen [22]. Como se ha demostrado en el Capítulo 7, ambos algoritmos han obtenido la solución óptima en la instancia que simula el desastre de Haití. Se pretende, por tanto, utilizar una pila de problemas de gran tamaño para comparar el rendimiento de ambas metaheurísticas.

Por último, existe la posibilidad de transformar el problema resuelto con un enfoque uniobjetivo a un enfoque multiobjetivo. De este modo, se podría considerar la interacción coste-accesibilidad.

Capítulo 9

Bibliografía

[1] P. J. Angeline. Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In *Evolutionary Programming VII*, pages 601–610. Springer, Berlin, 1998.

[2] A. Antunes, Á. Seco, and N. Pinto. An accessibility–maximization approach to road network planning. *Computer-Aided Civil and Infrastructure Engineering*, 18(3):224–240, 2003.

[3] B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank based version of the ant system: A computational study. *SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business*, 1997.

[4] A. M. Campbell, T. J. Lowe, and L. Zhang. Upgrading arcs to minimize the maximum travel time in a network. *Networks*, 47(2):72–80, 2006.

[5] J. Capela, P. Machado, D. Castro, and P. Henriques Abreu. An inverted ant colony optimization approach to traffic. *Engineering Applications of Artificial Intelligence*, 36:122–133, 2014.

[6] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[7] I. Charon and O. Hudry. The noising method: A new method for combinatorial optimization. *Operations Research Letters*, 14(3):133–137, 1993.

[8] I. Charon and O. Hudry. The noising methods: A generalization of some metaheuristics. *European Journal of Operational Research*, 135(1):86–101, 2001.

[9] K. Chellapilla. Evolving computer programs without subtree crossover. *IEEE Transactions on Evolutionary Computation*, 1(3):209–216, 1997.

[10] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[11] L. Cooper. Location-allocation problems. *Operations Research*, 11(3):331–343, 1963.

[12] O. Cordon, I. F. de Viana, F. Herrera, and L. Moreno. *A new ACO model integrating evolutionary computation concepts: The best-worst Ant System.* Citeseer: Princeton, 2000.

[13] J.-P. Courat, G. Raynaud, I. Mrad, and P. Siarry. Electronic component model minimization based on log simulated annealing. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 41(12):790–795, 1994.

[14] M. Creutz. Microcanonical monte carlo simulation. *Physical Review Letters*, 50(19):1411–1414, 1983.

[15] J. E. Dennis, Jr and J. J. Moré. Quasi-newton methods, motivation and theory. *SIAM Review*, 19(1):46–89, 1977.

[16] C. Donnges and V. Foley. *Improving Access in Rural Areas-Guidelines for Integrated Rural Accessibility Planning.* Transport Research Laboratory: Berkshire, 2003.

[17] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[18] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Management, and Cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.

[19] M. Dorigo and T. Stützle. *Ant colony optimization.* MIT Press: Cambridge, 2004.

[20] F. Ducatelle. Swarmanoid robots find shortest path over double bridge? https://www.youtube.com/watch?v=oBhv4pKksgU, 2009.

[21] G. Dueck and T. Scheuer. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161–175, 1990.

[22] P. M. Duque and K. Sörensen. A grasp metaheuristic to improve accessibility after a disaster. *OR Spectrum*, 33(3):525–542, 2011.

[23] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. IEEE: Piscataway, 1995.

[24] R. Eberhart, P. Simpson, and R. Dobbins. *Computational intelligence PC tools*. Academic Press Professional, Inc: San Diego, 1996.

[25] T. A. Feo and M. G. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71, 1989.

[26] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[27] L. M. Gambardella, M. Dorigo, et al. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *International Conference on Machine Learning*, pages 252–260, 1995.

[28] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.

[29] A. Goldberg and R. Tarjan. Solving minimum-cost flow problems by successive approximation. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 7–18. ACM, 1987.

[30] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[31] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In *Sixth International Conference on Genetic Algorithms*, pages 57–64. Citeseer, 1995.

[32] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[33] P. Hansen and N. Mladenović. *Variable neighborhood search*. Springer: Berlin, 2005.

[34] J. P. Hart and A. W. Shogan. Semi-greedy heuristics: An empirical study. *Operations Research Letters*, 6(3):107–114, 1987.

[35] C. Haskins. How ants communicate? https://www.youtube.com/watch?v=gcHt5n3NGKO, 2011.

[36] M. J. Hirsch, C. Meneses, P. M. Pardalos, and M. G. Resende. Global optimization by continuous GRASP. *Optimization Letters*, 1(2):201–212, 2007.

[37] J. H. Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press: Ann Arbor, 1975.

[38] T. C. Hu and M.-t. Shing. *Combinatorial Algorithms: Enlarged Second Edition.* Courier Corporation: Toronto, 2012.

[39] J. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8:703–712, 1960.

[40] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[41] J. R. Koza. *Genetic Programming: vol. 1, On the programming of computers by means of natural selection.* MIT press: Cambridge, 1992.

[42] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[43] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55, 1984.

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

[45] V. Maniezzo and A. Carbonaro. An ants heuristic for the frequency assignment problem. *Future Generation Computer Systems*, 16(8):927–935, 2000.

[46] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.

[47] E. F. Moore. *The shortest path through a maze.* Bell Telephone System, USA, 1959.

[48] H. Muñoz. Sistema de colonias de hormigas para problemas de accesibilidad tras un desastre natural. Master's thesis, Universidad Politécnica de Madrid, 2014.

[49] OCHA-UN. Haiti flash appeal 2008. Technical report, Office for the cordination of humanitarian affairs, United Nations, 2008.

[50] OMP. Wfp-assisted development projects, protracted relief and emergency operations. Technical report, The Regional Bureau for Latin America and the Caribbean, United Nations, 2009.

[51] J. R. Oppong. Accommodating the rainy season in third world location-allocation applications. *Socio-Economic Planning Sciences*, 30(2):121–137, 1996.

[52] I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Ministry of Aviation, Royal Aircraft Establishment, United Kingdom, 1965.

[53] I. Rechenberg. *Evolutionsstrategie: Optimierung technisher Systeme nach Prinzipien der biologischen Evolution*. Springer, Berlin, 1973.

[54] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *Society for Industrial and Applied Mathematics Journal on Computing*, 6(3):563–581, 1977.

[55] M. P. Scaparra and R. L. Church. A grasp and path relinking heuristic for rural road network development. *Journal of Heuristics*, 11(1):89–108, 2005.

[56] H.-P. Schwefel. *Numerical optimization of computer models*. John Wiley & Sons: Hoboken, 1981.

[57] H.-P. Schwefel and G. Rudolph. *Contemporary evolution strategies*. Springer: Berlin, 1995.

[58] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings in IEEE World Congress on Computational Intelligence*, pages 69–73. IEEE, Piscataway, 1998.

[59] T. Stützle and H. H. Hoos. Max–min ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[60] G. Syswerda. A study of reproduction in generational and steady state genetic algorithms. *Foundations of genetic algorithms*, 2:94–101, 1991.

[61] E.-G. Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons: Hoboken, 2009.

[62] C. J. C. H. Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[63] S. Yan and Y.-L. Shih. A time-space network model for work team scheduling after a major disaster. *Journal of the Chinese Institute of Engineers*, 30(1):63–75, 2007.

Apéndice A

Datos del problema de Haití

Cuadro A.1: Tiempos de travesía y costes financieros y de mano de obra

e	t_e	c_e	m_e	e	t_e	c_e	m_e	e	t_e	c_e	m_e
(1)	16.99	5.6	5.0	(95)	5.65	-	-	(189)	6.36	-	-
(2)	29.3	6.5	9.3	(96)	12.24	-	-	(190)	9.52	-	-
(3)	7.49	0.9	3.6	(97)	9.01	-	-	(191)	5.63	-	-
(4)	5.4	2.5	5.1	(98)	10.62	-	-	(192)	8.24	-	-
(5)	22.67	7.1	7.7	(99)	12.63	-	-	(193)	7.22	-	-
(6)	15.3	7.1	3.7	(100)	7.14	-	-	(194)	9.3	-	-
(7)	9.13	2.8	3.7	(101)	15.95	-	-	(195)	16.69	-	-
(8)	7.36	4.0	1.0	(102)	19.32	-	-	(196)	15.13	-	-
(9)	4.47	3.8	1.4	(103)	6.06	-	-	(197)	15.03	-	-
(10)	15.64	7.0	5.8	(104)	12.72	-	-	(198)	9.86	-	-
(11)	18.16	4.9	5.8	(105)	12.6	-	-	(199)	15.12	-	-
(12)	8.69	4.2	2.6	(106)	1.29	-	-	(200)	2.18	-	-
(13)	16.14	2.8	0.9	(107)	3.04	-	-	(201)	0.44	-	-
(14)	13.42	4.0	3.8	(108)	5.92	-	-	(202)	0.32	-	-
(15)	6.9	3.2	2.7	(109)	10.6	-	-	(203)	9.21	-	-
(16)	7.39	3.8	4.2	(110)	9.8	-	-	(204)	9.46	-	-
(17)	12.08	3.6	2.4	(111)	1.13	-	-	(205)	1.15	-	-
(18)	13.34	2.8	3.4	(112)	5.95	-	-	(206)	6.47	-	-
(19)	1.61	2.5	3.6	(113)	2.73	-	-	(207)	1.93	-	-
(20)	9.8	3.0	4.2	(114)	9.27	-	-	(208)	3.9	-	-
(21)	13.16	6.5	6.2	(115)	10.24	-	-	(209)	10.38	-	-
(22)	12.19	4.4	5.1	(116)	4.75	-	-	(210)	28.59	-	-

Cuadro A.2: Continuación

e	t_e	c_e	m_e	e	t_e	c_e	m_e	e	t_e	c_e	m_e
(23)	12.75	4.6	4.3	(117)	5.58	-	-	(211)	12.52	-	-
(24)	1.19	3.3	2.1	(118)	4.06	-	-	(212)	6.33	-	-
(25)	2.4	4.4	4.2	(119)	2.14	-	-	(213)	2.79	-	-
(26)	9.14	2.7	5.5	(120)	4.52	-	-	(214)	6.23	-	-
(27)	12.7	4.4	6.5	(121)	10.66	-	-	(215)	6.67	-	-
(28)	3.81	1.7	5.1	(122)	10.08	-	-	(216)	13	-	-
(29)	14.26	3.7	3.3	(123)	6.34	-	-	(217)	15.72	-	-
(30)	13.7	2.7	5.6	(124)	1.41	-	-	(218)	7.4	-	-
(31)	17.57	-	-	(125)	3.18	-	-	(219)	13.69	-	-
(32)	4.1	-	-	(126)	0.23	-	-	(220)	6.3	-	-
(33)	10.83	-	-	(127)	12.16	-	-	(221)	8.93	-	-
(34)	10.71	-	-	(128)	10.45	-	-	(222)	12.8	-	-
(35)	5.06	-	-	(129)	8.97	-	-	(223)	7.69	-	-
(36)	12.25	-	-	(130)	7.51	-	-	(224)	8.05	-	-
(37)	3.26	-	-	(131)	2.86	-	-	(225)	0.38	-	-
(38)	8.68	-	-	(132)	10.32	-	-	(226)	1.47	-	-
(39)	4.25	-	-	(133)	1	-	-	(227)	3.53	-	-
(40)	4.1	-	-	(134)	15.07	-	-	(228)	8.14	-	-
(41)	11.75	-	-	(135)	10.35	-	-	(229)	4.81	-	-
(42)	7.4	-	-	(136)	12.5	-	-	(230)	5.99	-	-
(43)	7.08	-	-	(137)	10.8	-	-	(231)	10.41	-	-
(44)	7.09	-	-	(138)	8.68	-	-	(232)	4.34	-	-
(45)	11.63	-	-	(139)	19.91	-	-	(233)	3.47	-	-
(46)	18.5	-	-	(140)	16.35	-	-	(234)	7.4	-	-
(47)	4.66	-	-	(141)	9.47	-	-	(235)	20.01	-	-
(48)	7.3	-	-	(142)	21.55	-	-	(236)	5.33	-	-
(49)	11.6	-	-	(143)	9.67	-	-	(237)	6.75	-	-
(50)	3.38	-	-	(144)	10.92	-	-	(238)	15.88	-	-
(51)	2.23	-	-	(145)	10.87	-	-	(239)	0.99	-	-
(52)	16.69	-	-	(146)	10.07	-	-	(240)	0.4	-	-
(53)	4.33	-	-	(147)	2.2	-	-	(241)	0.28	-	-
(54)	7.33	-	-	(148)	4.8	-	-	(242)	16.91	-	-
(55)	7.59	-	-	(149)	0.86	-	-	(243)	4.27	-	-
(56)	7.58	-	-	(150)	5.08	-	-	(244)	8.02	-	-
(57)	6.14	-	-	(151)	0.67	-	-	(245)	5.17	-	-
(58)	8.83	-	-	(152)	3.99	-	-	(246)	6.09	-	-
(59)	12.7	-	-	(153)	5.53	-	-	(247)	10.82	-	-

Cuadro A.3: Continuación

e	t_e	c_e	m_e	e	t_e	c_e	m_e	e	t_e	c_e	m_e
(60)	7.66	-	-	(154)	5.67	-	-	(248)	0.75	-	-
(61)	6.23	-	-	(155)	7.72	-	-	(249)	6.76	-	-
(62)	7.19	-	-	(156)	14.76	-	-	(250)	15.03	-	-
(63)	2.89	-	-	(157)	3.53	-	-	(251)	6.88	-	-
(64)	5.2	-	-	(158)	2.35	-	-	(252)	6.9	-	-
(65)	2.93	-	-	(159)	7.54	-	-	(253)	8.18	-	-
(66)	5.31	-	-	(160)	8.72	-	-	(254)	3.25	-	-
(67)	11.16	-	-	(161)	4.81	-	-	(255)	8.7	-	-
(68)	5.14	-	-	(162)	22.52	-	-	(256)	3.04	-	-
(69)	12.35	-	-	(163)	17.9	-	-	(257)	10.13	-	-
(70)	3.94	-	-	(164)	8.35	-	-	(258)	3.22	-	-
(71)	4.28	-	-	(165)	10.2	-	-	(259)	5.47	-	-
(72)	5.14	-	-	(166)	7.32	-	-	(260)	3.03	-	-
(73)	3.24	-	-	(167)	10.79	-	-	(261)	3.67	-	-
(74)	1.69	-	-	(168)	0.87	-	-	(262)	12.72	-	-
(75)	2.19	-	-	(169)	3.55	-	-	(263)	3.7	-	-
(76)	3.7	-	-	(170)	10.17	-	-	(264)	5.82	-	-
(77)	3.58	-	-	(171)	5.27	-	-	(265)	5.72	-	-
(78)	2.18	-	-	(172)	2.96	-	-	(266)	7.51	-	-
(79)	2.83	-	-	(173)	13.14	-	-	(267)	8.32	-	-
(80)	4.57	-	-	(174)	8.17	-	-	(268)	8.54	-	-
(81)	5.25	-	-	(175)	14.7	-	-	(269)	19.77	-	-
(82)	10.83	-	-	(176)	8.83	-	-	(270)	1.54	-	-
(83)	5.22	-	-	(177)	6.02	-	-	(271)	3.28	-	-
(84)	2.75	-	-	(178)	4.11	-	-	(272)	28.43	-	-
(85)	17.33	-	-	(179)	4.18	-	-	(273)	21.86	-	-
(86)	6.56	-	-	(180)	2.53	-	-	(274)	13.16	-	-
(87)	6.08	-	-	(181)	3.54	-	-	(275)	13.5	-	-
(88)	0.81	-	-	(182)	7.67	-	-	(276)	9.87	-	-
(89)	5.83	-	-	(183)	1.78	-	-	(277)	2.73	-	-
(90)	11.95	-	-	(184)	23.1	-	-	(278)	2.5	-	-
(91)	3.48	-	-	(185)	14.15	-	-	(279)	6.01	-	-
(92)	9.66	-	-	(186)	4.25	-	-	(280)	14.84	-	-
(93)	12.14	-	-	(187)	7.45	-	-	(281)	17.38	-	-
(94)	5.32	-	-	(188)	0.59	-	-				