Universidad Autónoma de Madrid
Escuela Politécnica Superior

Proyecto fin de carrera

DESARROLLO DE UN ALGORITMO GENÉTICO PARA LA CREACIÓN DE REDES WIFI EN ENTORNOS URBANOS

Grado en Ingeniería Informática

Luis Cristian Mateos Carrera
Junio 2015

DESARROLLO DE UN ALGORITMO GENÉTICO PARA LA CREACIÓN DE REDES WIFI EN ENTORNOS URBANOS

AUTOR: Luis Cristian Mateos Carrera
TUTOR: Antonio González Pardo
Co-Tutor: David Camacho Fernández

Grupo de la EPS: AIDA
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2015

Resumen

Resumen

El uso de **Internet** se ha vuelto una herramienta de uso **imprescindible** en el día a día. En un principio su uso estaba restringido a dispositivos sin movilidad, pero gracias a la tecnología **Wi-Fi y redes móviles**, se ha extendido su aplicación a dispositivos portátiles, permitiendo su uso sin conexión física.

El uso de las redes Wi-Fi tiene limitaciones como el número máximo de conexiones o alcance, por lo que para conectarse en lugares en los que no está disponible es necesario contratar un servicio de Internet móvil. En consecuencia, surge la **idea y necesidad** de crear redes Wi-Fi que cubran a la mayor población posible en entornos urbanos.

En este trabajo se estudia la **eficiencia** del despliegue y uso de los recursos disponibles, con el fin de satisfacer dicha necesidad. Para ello se ha desarrollado un **algoritmo genético** debido a su demostrada eficacia en problemas del mismo tipo. El algoritmo recibe los datos de la población (posibles lugares en los que colocar puntos de acceso y localización de población), y de los tipos de routers posibles, de modo que minimiza los costes y maximiza la cantidad de gente conectada eligiendo las localizaciones en las que colocar los tipos de routers con mayor rendimiento.

En el Trabajo de Fin de Grado se trata el desarrollo y análisis de un algoritmo genético para resolver dicho problema.

Palabras Clave

Redes Wi-Fi, entornos urbanos, Algoritmo Genético.

Abstract

With the passage of time, the usage of the Internet has evolved into an essential tool in todays life. At the onset, its usage was limited to inmobile devices, but thanks to Wi-fi technology, its usege has been expanded to portable devices, allowing its usage without the need of a fixed wire connectivity.

However, the use of Wi-fi networks have certain constraints such as a maximum number of connections or its reach, thereby making it necessary to have a mobile Internet Service Provider. As a consequence, the idea and need arises for the emergence of Wi-Fi networks that can reach a greater number of population in complete urban setting.

In this report, we examine the efficiency of the deployment and usage of the available resources with the aim of meeting such demand. With this in mind, we have developed a Genetic Algorithm due to its proven effectiveness in identical type of problems. The algorithm obtains data from the population (possible locations where to install access points, positioning of population), and the types of possible routers with the aim of minimizing costs and maximizing the number of people connected by selecting the locations where to lay the types of routers with highest efficiency.

The goal of this final degree study deals with the development and analysis of the Genetic Algorithm to solve the above mentioned problem.

Key words

Wi-Fi, urban enviroments, genetic algorithms.

Agradecimientos

A todas las personas que me han ayudado a llegar hasta aquí. Mis padres, mi hermano y Adri.

Gracias.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

Los algoritmos bio-inspirados son aquellos que simulan el comportamiento y el modo de procesar dificultades de sistemas biológicos, para responder con una salida adecuada y óptima a problemas. Estos algoritmos, en concreto los algoritmos estocásticos, han demostrado ser eficientes frente a problemas como el de las N-Reinas ([3],[11]) o el problema de planificación de proyectos ([5],[9],[10]) (Project Scheduling Problem).

Los algoritmos bio-inspirados más populares son las redes neuronales artificiales[6], la lógica difusa[6] y la computación evolutiva[7], siendo el último el colectivo al que pertenece el algoritmo en el que se centra este trabajo, los algoritmos genéticos (GA), con el fin de comprobar su capacidad de resolver un problema de gestión de recursos de forma eficiente y ofreciendo una salida igual o cercana a la más óptima posible.

Los algoritmos genéticos trabajan sobre una población compuesta de individuos, que a su vez se componen de genes. Cada gen representa un atributo, por lo que cada individuo esta formado por un conjunto de atributos que juntos proponen una solución al problema. La característica principal de estos algoritmos es que buscan elegir a los individuos más destacados de una población para la fase de reproducción, teniendo cada individuo una probabilidad determinada de ser escogido dependiendo de sus propiedades. De esta forma, combinando los genes de los individuos seleccionados y modificando de forma aleatoria algunos de los genes, se da paso a la siguiente población de individuos, sobre la que se repite el proceso. En consecuencia de esto, se deduce la segunda característica: es necesaria una función de evaluación llamada fitness, a partir de la cual se evalúan los individuos.

En el problema que trata resolver este TFG, se busca distribuir de forma eficiente una serie de recursos limitados para desplegar una red Wi-Fi en una población urbana. Los atributos y recursos con los que se trabajan son:

- Dinero disponible
- Tipos de routers
  - Coste
  - Capacidad
  - Rango

- Mapa
  - Personas
  - Puntos libres

La población será representada en una matriz, posicionándose los individuos y los lugares en los que es posible colocar los routers sobre ella.

Teniendo en cuenta estos datos en este documento se tratará el estado del arte de los algoritmos bio-inspirados en el capítulo 2 y la adaptación de los algoritmos genéticos a nuestro problema junto con su implementación en el capítulo 3. Los resultados de la fase de experimentación se muestran en el capítulo 4 y por último las conclusiones extraidas de este TFG se explicarán en el capítulo 5.

2

Estado del arte

Este capítulo tiene como objetivo contextualizar el estado del arte para las áreas tratadas en este trabajo. En el apartado 2.1 se habla sobre los algoritmos bioinspirados, haciendo énfasis en el apartado 2.1.1 sobre los algoritmos de arrecifes de coral y de colonia de hormigas, y en el apartado 2.1.2 en los algoritmos genéticos.

En el apartado 2.2 se describe el problema que tratamos de resolver usando algoritmos géneticos y el por que los usamos para resolver dicho problema.

2.1. Algoritmos bioinspirados

La computación bioinspirada es un campo de estudio que a menudo se relaciona con el campo de la inteligencia artificial, ya que muchos de sus objetivos están vinculados con la capacidad de aprender de las máquinas. La computación bioinspirada trata temas como la conectividad y el comportamiento social.

Generalmente la inteligencia artificial ha tratado de reproducir la capacidad del cerebro humano, pero existen otros enfoques basados en otros conjuntos biológicos organizados. De esta forma se utilizan y analizan ejemplos de vida reales como modelo para mejorar el funcionamiento de los ordenadores.

De entre todos los campos que han servido de inspiración para los algoritmos bioinspirados, los más usados, y que trataremos a continuación, se basan concretamente en el comportamiento colectivo de las hormigas, la forma de expandirse de los arrecifes de coral y la teoría de la evolución de Darwin.

2.1.1. Algoritmos de Optimización basados en Arrecifes de Coral (CRO)

La optimización de arrecifes de coral se trata de un algoritmo evolucional bioinspirado, que simula los procesos de los arrecifes de coral. Ha sido probado que este algoritmo es eficaz en problemas de optimización con un solo objetivo, obteniendo mejores resultados que otros algorimos de optimización. Este algoritmo empieza presentando diferentes soluciones para el problema de optimización representadas por los corales. Las soluciones, representadas por los

corales, se colocan en una matriz, donde quedan espacios vacíos al principio del algoritmo. El algoritmo continúa aplicando la reproducción del arrecife, tras lo cual empieza una lucha por el espacio. En cada paso del algoritmo se produce una formación de larvas, y cada larva intenta ocupar el espacio libre en el arrecife. Si encuentra un espacio depende de su fuerza o salud, o de una probabilidad aleatoria. Después de algunas generaciones, el espacio en el arrecife se vuelve escaso, pero un proceso de depredación del coral asegura que siempre exista espacio libre en el arrecife. El pseudocódigo del CRO es el mostrado en el algoritmo 1.

inicializaciónDelArrecife
$g \leftarrow 0$
while $g < G_{max}$ **do**
    Fase de reproducción
    Evaluación de los nuevos corales
    Colocación de las larvas
    Proceso de depredación
    Procedimiento de control de réplicas
    $g \leftarrow g + 1$
end
end
Devolver el mejor coral del arrecife

Algorithm 1: Algoritmo de Optimización basado en Arrecifes de Coral

La fuerza o salud de cada coral viene dada por una función de evaluación, que permite comparar los diferentes corales en la selección y reproducción. En la fase de inicialización los corales se representan sobre la matriz que simula el arrecife, y son colocados aleatoriamente. El número de corales depende del tamaño de la matriz y de un parámetro, con el cual se configura la densidad de la población.

En la fase de reproducción el arrecife de coral genera nuevos corales. La reproducción puede ser de tres tipos:

- Sexual externa: Se produce un crossover entre dos corales seleccionados según su evaluación.
- Sexual interna: Se produce una copia de los corales para mutarlos posteriormente de manera aleatoria.
- Asexual: se seleccionan los corales con mayor evaluación y se copian, de manera que intenten buscar un espacio en el arrecife.

Utilizando el procedimiento asexual podría darse el caso de que un arrecife estuviese ocupado en su mayor parte por un mismo coral con una evaluación muy alta. Para evitar que esto ocurra se emplea un algoritmo de control de réplicas, que cuenta el número de muestras iguales en el arrecife y elimina la mitad de estos ejemplares.

El siguiente proceso es la colocación de larvas, en la que los nuevos corales encuentran un espacio vacío en el que colocarse. Cuando se crea una nueva larva, ésta permanece a la espera hasta que terminen de crearse todas las larvas. Una vez terminado este proceso, todas las larvas creadas buscarán su sitio. El lugar en el que cada larva intentará asentarse es elegido aleatoriamente. Si el lugar está vacío se colocará en esa posición. En caso contrario comparará su salud con el coral que se encuentre en ese sitio y, si es mayor, le reemplazará, quedando eliminado el coral antiguo del arrecife. En caso de que su salud sea menor intentará colocarse en otra posición. Si la larva no es capaz de encontrar un sitio después de un número de intentos definido, muere.

Por último se incorpora un proceso de depredación, que simula la muerte de corales por factores externos. De esta manera se libera espacio en el arrecife entre generaciones, y se asegura que siempre exista espacio disponible en el arrecife. Los corales seleccionados son los que menos salud tengan del arrecife. Cada uno de los corales seleccionados tiene una probabilidad baja de ser eliminados.

2.1.2. Algoritmos de Optimización basado en Colonias de Hormigas

Este algoritmo imita el comportamiento de las colonias de hormigas. Individualmente las hormigas siguen unas reglas sencillas, pero las interacciones que realizan entre ellas y el entorno dan lugar a un comportamiento que podríamos etiquetar como inteligente. El propósito de las hormigas será encontrar el camino más corto recorriendo los nodos de un grafo. En cada periodo de tiempo las hormigas elegirán el camino que seguirá en el siguiente movimiento según su probabilidad. Esta probabilidad dependerá de la heurística asociada al problema y las feromonas que las hormigas desprenden.

Cuando una hormiga encuentra el objetivo, recorre el camino realizado de vuelta, dejando a su paso feromonas. Estas feromonas contienen información de la solución alcanzada y de la calidad de dicha solución, y serán almacenadas en cada vértice, de tal forma que el vértice almacene el total de feromonas depositadas por todas las hormigas que hayan pasado por él. El número de feromonas depositadas dependerá de la calidad del camino recorrido. En consecuencia, los mejores caminos almacenarán mayor número de feromonas y tendrán mas posibilidades de ser elegidos por el resto de hormigas.

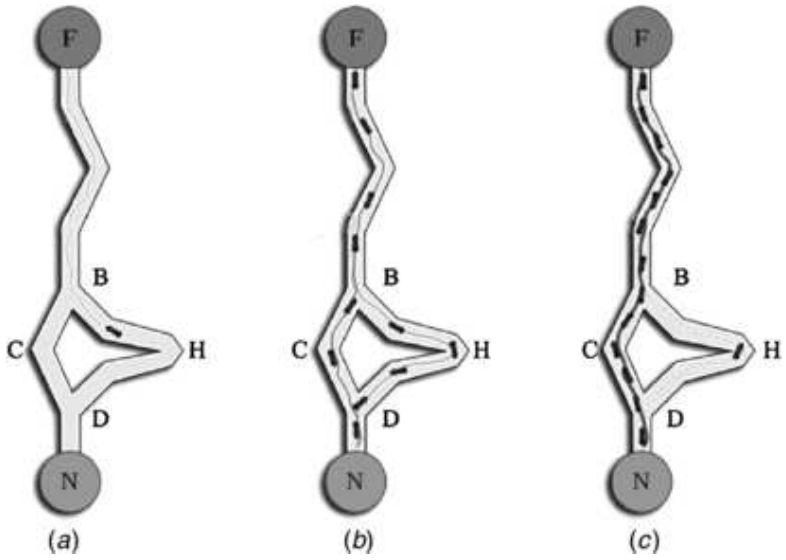

Figura 2.1: Ejemplo algoritmo colonia de hormigas

En el momento t, dada una hormiga k en el nodo i, la probabilidad de moverse al nodo j se define por la ecuación de la figura 2.2.

Donde N_i^k representa el conjunto de nodos viables conectados al nodo i, mientras que η_{ij} representa el valor de la heurística al moverse del nodo i al nodo j. α es un parámetro que controla la influencia de las feromona, β controla la influencia de la heurística y τ devuelve el valor de las feromonas. El dividendo devuelve el total de feromonas depositadas en todos los vértices que conectan el nodo i con el resto de nodos.

Por último, para evitar la explotación de caminos con muchas feromonas y propiciar la aparición de nuevas posibilidades en cada iteración se produce una evaporación de feromonas.

$$p_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{u \in \mathcal{N}_i^k} \tau_{iu}^{\alpha}(t)\eta_{iu}^{\beta}(t)} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{if } j \notin \mathcal{N}_i^k \end{cases}$$

Figura 2.2: Ecuación probabilidad de movimiento ACO

2.1.3. Algoritmos Genéticos (GA)

Los Algoritmos Genéticos son un tipo de computación evolutiva. Los algoritmos de computación evolutiva son aquellos que se basan en la teoría de la evolución de Darwin para ofrecer un espacio de soluciones. Estos algoritmos se basan en la selección natural y en la supervivencia del más fuerte.

El psudocódigo genérico para un algoritmo evolutivo es el mostrado en algoritmo 2. Una población se encuentra compuesta de individuos aleatoriamente inicializados. Estos individuos representan una posible solución al problema, estando formada cada una de las soluciones por genes que en conjunto forman el genotipo de un individuo. Cada gen representa una característica del individuo.

Para el funcionamiento del algoritmo es necesario una función que los evalúe, llamada fitness. Una vez evaluados se seleccionarán individuos de la población teniendo en cuenta su fitness. Este fitness se utilizará de una manera u otra dependiendo del modo de selección elegido entre uniforme, torneo o proporcional. Una vez seleccionados los padres, se crea una nueva generación de individuos en la fase de reproducción, volviendo a repetirse el proceso con los nuevos individuos.

```
t ← 0
P_t ← n initialized individuals
while termination criteria is not satisfied do
    EvaluateFitness(I_i) ∀I_i ∈ P_t
    P* ← new empty population
    while P* is not full do
        parents ← selectParent(P_t)
        offspring ← reproductionProcess(parents)
        includeOffspring(P*,offspring)
    end
    t ← t + 1
    P_t ← P*
end
```

Algorithm 2: Algoritmo Evolutivo Genérico

El proceso se repite hasta que se cumple la condición de parada que puede suceder en dos casos:

- El número de generaciones ha llegado al número máximo de generaciones permitidas.
- El valor del fitness de una población no difiere significativamente del fitness de la generación anterior.

A la hora de aplicar un algoritmo evolutivo hay que tener en cuenta:

- Codificación del genotipo: la codificación del genotipo tiene influencia sobre los procedimientos reproductivos.
- Operador de selección: las siguientes generaciones dependerán del operador con el que se seleccionen a los padres que las producirán.
- Fase de reproducción: ésta fase se compone del crossover y de la mutación.

Los operadores más comunes en la fase de selección son: uniforme, proporcional y torneo. En el primero todos los individuos tienen la misma probabilidad de ser seleccionados, independientemente de su fitness, por lo que el mejor individuo tiene las mismas probabilidades de ser seleccionado que el peor individuo. El segundo operador es el más usado. La probabilidad de ser seleccionado se reparte entre todos los individuos teniendo en cuenta su fitness. De éste se puede tener en cuenta el fitness tanto positivamente como negativamente, de forma que podamos buscar maximizarlo o minimizarlo. El último operador es la selección por torneo. Utilizando este modo se seleccionan aleatoriamente un número de individuos determinado por parámetro a los que posteriormente se les compara el fitness. El individuo con mejor fitness es seleccionado.

En la reproducción se producen dos fases, crossover y mutación, que dan como resultado una nueva generación de individuos.

En la fase de crossover se mezcla el material genético de los padres en el hijo. Aunque existen varias técnicas de crossover, la más común y simple es el One-Point crossover. En éste crossover se selecciona un punto del genotipo, hasta el cual se copiarán los genes del primer padre y desde el cual se copiarán los genes del segundo padre. Se puede observar un ejemplo en la figura 2.3.

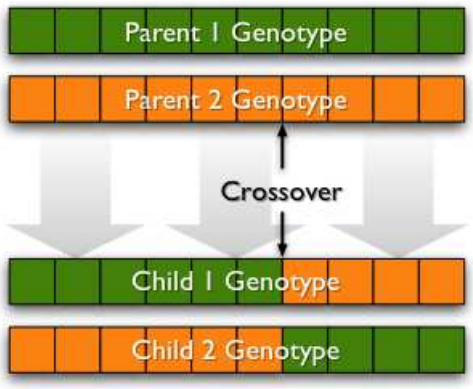

Figura 2.3: Ejemplo crossover

Por último, en la fase de mutación, se modifican aleatoriamente valores en los genotipos de la nueva generación. De esta forma se consigue diversidad genética.

2.2. Problema de implantación de redes wifi en entornos urbanos

La popularidad de las redes inalámbricas WiFi es un hecho innegable, debido a su estandarización, su bajo coste y su facilidad de uso. Hoy en día, cualquier portátil o dispositivo personal dispone de acceso WiFi y esto hace que diversos gobiernos y ayuntamientos estén invierto en el desarrollo de redes WiFi de acceso abierto en diversas ciudades del mundo. El objetivo es que la mayor parte de la población pueda hacer uso de esta red gratuita.

Sin embargo, el desarrollo de esta red no es sencillo, ya que existen diversos factores a tener en cuenta como pueden ser el coste de desarrollo, el número de habitantes que podrían hacer uso de la red, o la definición de los puntos de acceso de la red.

Para resolver el problema utilizamos algoritmos genéticos, ya que las características del problema son altamente compatibles con la definición de individuo y genotipo. Existen otros trabajos que han abordado este problema utilizando algoritmos genéticos como [1] en el que define el genotipo como los puntos de acceso disponibles, tanto habilitados como deshabilitados, o [2] que lo hace utilizando un sistema de *Harmony Search*[8]. En ambos trabajos el coste de cada punto de acceso se encuentra ligado de manera proporcional con su capacidad.

En éste trabajo definimos el gen como la relación entre la posición y tipo de router (conteniendo el tipo de router capacidad, coste y distancia a la que es capaz de ofrecer servicio) y los usuarios conectados a dicho punto de acceso. Por último, el tamaño del genotipo puede variar dentro de un límite definido por el número de posiciones para puntos de acceso disponibles, por lo que a diferencia de otros trabajos su tamaño no es estático. De este modo el genotipo de un individuo se encuentra formado por todos los puntos de acceso habilitados en sus respectivas posiciones y los usuarios conectados a ellos.

Los datos desde los que partiremos serán la población y su colocación, las posiciones disponibles para los puntos de acceso y los datos de los tipos de routers. Inicialmente colocaremos la población y las localizaciones disponibles para los puntos de acceso sobre una matriz que simulará el mapa. En la fase de inicialización se crea un número definido por el usuario de individuos, y se rellenarán de forma aleatoria con los routers, posiciones y usuarios a los que dará servicio. Para simplificar el trabajo los usuarios se conectarán al primer punto de acceso que tengan disponible y no se repartirán para aprovechar las capacidades de todos los routers.

La parte más importante en los algoritmos genéticos es definir la funcion *fitness*. Para nuestro problema la función evalúa a cada individuo teniendo en cuenta el número de personas conectadas y el precio de todos sus routers. En consecuencia para buscar la solución óptima se busca maximizar el fitness del individuo mediante la fórmula de la figura 2.4.

Para la fase de selección utilizaremos la selección proporcional al fitness, cuanto más fitness mayores serán las probabilidades de ser elegido.

$$fitness(k) = \alpha * usuariosConectados + \beta * (\text{precioMáximo}() - precio(k))$$

Figura 2.4: Fórmula del fitness

Para la fase de reproducción se seleccionan aleatoriamente dos individuos a partir del criterio anterior. Se realiza un crossover sobre los dos individuos seleccionados, a partir del cual se crea un nuevo individuo. Este nuevo individuo tiene una probabilidad definida por parámetro de ser mutado, por lo que su router puede variar. A la hora de realizar el crossover y añadir los genes de los padres en el hijo es necesario realizar una comprobación cada vez que se añade un gen del segundo padre, ya que es probable que el primer padre ya haya añadido un punto de acceso en una localización concreta, independientemente de si es el mismo tipo de router o no. En caso de que esto suceda el hijo mantiene el primer gen que se le haya añadido, desechando el segundo. Esta fase se repetirá tantas veces como sea necesario para llegar al número de individuos definido por parámetro en la siguiente generación.

Siguiendo este proceso, se producirán generaciones sucesivas hasta que se cumpla la condición de parada.

3

Implementación del Algoritmo Genético

Este capítulo detalla la implementación del Algoritmo Genético usado en este TFG. En primer lugar se explicarán la implementación de los apartados del algoritmo, de tal modo que se concluya explicando el algoritmo en su conjunto. Se explicarán los parámetros de entrada y de ajuste, siendo estos últimos necesarios para el correcto funcionamiento de la aplicación y por último se detallará la salida.

3.1. Implementación del individuo

El individuo contiene la información de una posible solución al problema. Para este TFG el individuo se ha codificado de forma que los datos que contiene son las personas conectadas, precio del despliegue y su respectivo genotipo. A su vez el genotipo guardará la información del conjunto de genes. En nuestro caso cada gen almacena la información del despliegue de un solo router, siendo esta la posición en los ejes X e Y de la matriz que representa el mapa y el tipo de router.

3.2. Inicialización de los datos

El punto de inicio del algoritmo es la inicialización. La primera acción en este punto es cargar los datos necesarios para el funcionamiento del algoritmo de sus respectivos ficheros. Estos datos se pueden categorizar de dos formas, datos del problema y datos de configuración. Los datos del problema se cargan de tres ficheros distintos:

- Fichero con la posición de las personas
- Fichero con la información de los routers
  - Coste
  - Capacidad
  - Rango de la señal

$$probabilidad(k) = \frac{fitness(k) * 100}{fitnessTotal()}$$

Figura 3.1: Fórmula de cálculo de probabilidad

- Fichero con los puntos de acceso disponibles

Los ficheros con la posición de las personas y los puntos de acceso almacenan la información en forma de coordenadas sobre los ejes X e Y.

Los datos de configuración son aquellos que se usan para definir el comportamiento del programa y se encuentran explicados en la sección 3.3.

Como último paso en el proceso de inicialización se crea una primera generación aleatoria utilizando los datos cargados.

3.3. Fase de evolución

En esta fase de evolución del algoritmo se iterarán una serie de operaciones sobre los datos que darán lugar a las siguientes generaciones. El algoritmo continuará hasta que se cumpla una condición de parada. En el caso de nuestro algoritmo la condición de paradas se cumple cuando se realizan un número determinado de iteraciones. De este modo evitaremos detectar malos resultados por máximos locales.

A continuación se explica en detalle la implementación de cada una de las partes de la fase de evolución.

3.3.1. Fase de selección

Cada individuo se encuentra evaluado por una función fitness, por lo que los individuos con mejor fitness supondrán una solución mejor al problema. Dicha función, representada en la figura 2.4, tiene en cuenta el gasto económico que representaría dicho individuo y el número de personas que es capaz de acoger a partir de la disposición de los individuos en la matriz.

Una vez evaluados los individuos se les asigna una probabilidad de ser seleccionados relacionando su fitness con el fitness total de la generación. La fórmula a partir de la cual se deduce la probabilidad se puede observar en la figura 3.1.

De esta forma cuanto mayor sea su fitness y más contribuya al fitness global, mayor será su probabilidad.

3.3.2. Fase de reproducción

Los individuos seleccionados en la fase de reproducción son los encargados de generar los nuevos individuos para la siguiente generación. Para ello se utiliza el proceso llamado crossover, mediante el cual el hijo recibe un número de genes de uno de los padres y otro número de genes del otro padre. Como se ha mencionado anteriormente, uno de los puntos diferenciadores de este TFG respecto a otros trabajos, es que el número de genes que debe tener un individuo no está establecido, por lo que del crossover puede ser generado un individuo con un número de genes menor o mayor que el de la media de sus padres (siempre debe tener al menos un gen). De este modo garantizamos más variedad, y que si un inviduo con un fitness muy alto es seleccionado

varias veces en la fase de selección no conlleve que los individuos mejor valorados de la siguiente generación sean todos sus hijos.

El proceso de crossover también ha sido simplificado para garantizar más variedad. Como se puede apreciar en la figura 2.3, de un crossover de dos padres se generan dos hijos, sin desechar ninguna parte del genotipo de los padres. En nuestro trabajo el crossover de dos individuos solo generará un hijo. Esto provocará más variedad, ya que serán necesarias el doble de selecciones de individuos para completar el cupo de individuos necesario para la siguiente generación.

El algoritmo que sigue el crossover se encuentra representado en el algoritmo 3. El número de genes seleccionado del padre y de la madre es aleatorio. De este modo el algoritmo primero añade al hijo los primeros N genes del padre, siendo N un número aleatorio entre '1' y el número de genes del padre. En segundo lugar se seleccionan los M últimos genes del segundo padre siendo M un número aleatorio entre '1' y el número de genes del segundo padre. Los genes del segundo padre se añadirán desde el final hasta el principio, de modo que si un gen está repetido en el hijo se añada el siguiente gen del segundo padre, prevaleciendo el gen del primer individuo seleccionado. La condición para que un gen se repita es que el router de ambos padres esté colocado en el mismo punto de acceso, independientemente del número de usuarios conectados y el tipo de router. Por último existe una probabilidad parametrizable de que este crossover no se produzca.

```
N ← random(1, firstFather.size())
i ← 0
while i < N do
    hijo.add(firstFather.get(i))
    i++
end
i ← 0
M ← random(1, secondFather.size())
while i < M do
    if !hijo.add(secondFather.get(secondFather.size()-i)) then
        if secondFather.size()< M then
            M++
        end
    end
    i++
end
```

Algorithm 3: Algoritmo de crossover

3.3.3. Mutación

Cada vez que se produce un crossover entre dos padres existe una probabilidad parametrizable de que se produzca una mutación en el nuevo individuo.

Para este TFG la mutación en el individuo se produce sobre la localización de un gen escogido al azar. Una vez escogida aleatoriamente la nueva localización del gen, es necesario comprobar que no existe ningún otro gen en el individuo que la comparta. En caso de que esto sea así se escogerá una nueva localización sucesivamente hasta que no se repita o no existan más localizaciones posibles.

3.4. Parámetros de entrada

El programa desarrollado para este TFG acepta varios parámetros para condicionar la salida. Estos datos son leidos de su respectivo fichero, el cual debe contener los siguientes campos definidos:

- Número de generaciones
- Tamaño de la población
- Tasa de crossover
- Tasa de mutación
- Repeticiones

El número de generaciones indica cuantas repeticiones de la fase de evolución se realizarán. El tamaño de la población indica el número de individuos que deben generarse en cada población. Las tasas de crossover y mutación indican la probabilidad de que se produzca el crossover entre dos individuos seleccionados y la probabilidad de que se produzca una mutación en el hijo respectivamente. Las repeticiones indican cuantas veces se ejecutará el programa. Todos estos parámetros son leidos de un fichero XML adjunto al programa.

3.5. Parámetros de ajuste

Debido a que los valores con los que trabaja el fitness pueden ser influidas por el usuario en la entrada (precio de los routers, posiciones de las personas y de los puntos de acceso), puede darse el caso de que los precios sean demasiado elevados o el número de personas conectadas mucho mayor que el precio. Por esta razón es necesario ajustar la importancia de ambos valores multiplicándolos por parámetros de ajuste.

En este trabajo se le ha dado mayor importancia al número de personas conectadas (α * numeroPersonasConectadas), mientras que el papel del precio (β * (precioMáximo()-precio(k))) será de elemento diferenciador entre individuos con un número de usuarios conectados similar o parecido.

3.6. Salida

La salida del programa consistirá en dos ficheros de texto. El primero contendrá la información media de los valores de cada generación, siendo estos:

- Fitness medio de la población
- Precio medio de la población
- Número medio de personas conectadas de la población
- Longitud media de la población

El segundo fichero contará con toda la información de cada individuo de cada generación:

- Probabilidad
- Fitness
- Número de genes
- Usuarios conectados
- Usuarios desconectados
- Precio

4

Fase experimental

En esta sección se analizan los resultados obtenidos del programa desarrollado en el TFG dependiendo de los parámetros de entrada descritos en la sección anterior.

Los parámetros usados para estos resultados son los descritos en la tabla 4.2.

Parámetro	Valor
Número de generaciones	100
Tamaño de la población	100
Crossover rate	0.9
Mutation rate	0.1
Repeticiones	10

Cuadro 4.1: Tabla de parámetros

Por otro lado, debido a los datos utilizados, el fitness se ha ponderado multiplicando el número de personas conectadas en cada individuo por '100', de modo que el precio queda como elemento diferenciador entre individuos con un número de personas conectadas similar.

Los datos de entrada para la fase de experimentación cuentan con 1000 usuarios y 1000 puntos de acceso posibles. Los valores de los ejes varían entre 0 y 200, por lo que el mapa cuenta con 40000 posiciones posibles.

Los datos de los routers utilizados se encuentran el la siguiente tabla:

Identificador	Capacidad	Radio	Precio
0	5	200	100
1	10	150	50
2	8	180	75
3	3	400	200

Cuadro 4.2: Tabla de routers

Al ser un algoritmo estocástico y tener componentes aleatorios es necesario repetir el algoritmo varias veces, de modo que utilicemos la media de los resultados como dato. En nuestro caso se ha establecido el parámetro de repeticiones a 10, por lo que nuestros datos se basarán en la media de 10 ejecuciones.

En primer lugar analizaremos el avance temporal del fitness medio de los individuos en las 30 primeras generaciones.

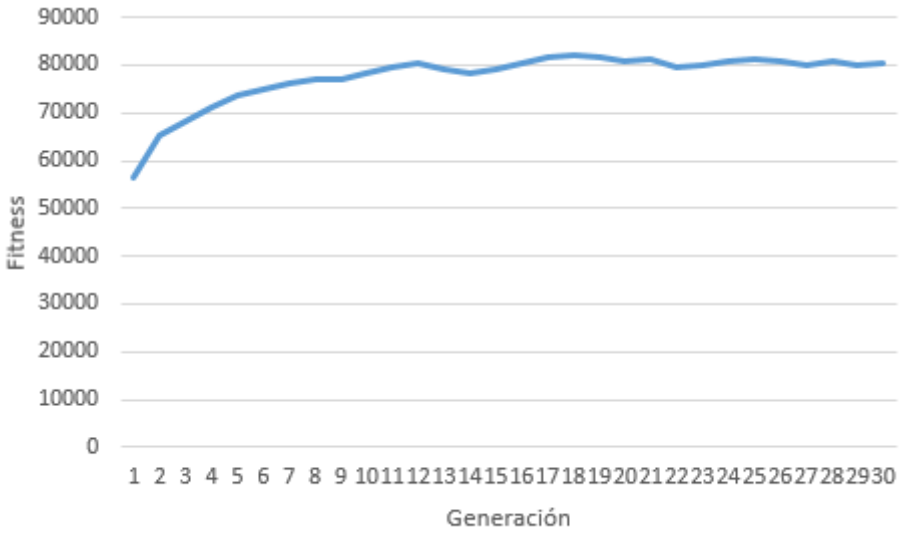

Figura 4.1: Fitness en 30 generaciones

Generación	Fitness	Precio	Conectados
0	56644,83	5433,475	524,383
10	78390,96	8438,224	727,7419
20	80924,49	8966,712	755,537
30	80608,46	8733,352	750,9931
40	79407,61	8573,032	738,0565
50	80208,72	8757,03	748,6825
60	80844,99	8827,346	755,7234
70	78847,64	8500,186	733,4533
80	81504,83	8870,453	763,8028
90	80256,26	8639,408	748,5317
100	80853,53	8678,169	756,017

Cuadro 4.3: Tabla de fitness medio en 100 generaciones

Como se puede observar en la figura 4.1 y en el cuadro 4.3, en las primeras 10 generaciones el fitness medio crece rápidamente, desde 56644 en la generación inicial a 78390. Esto es debido a que el margen de mejora en las primeras generaciones es muy amplio, y el número de personas conectadas pasa de 524 en la primera generación a 727 en la décima. A partir de la décima generación el fitness crece muy lentamente, ya que el margen de mejora pasa a ser menor.

Cuando ampliamos el número de muestras a 100, podemos observar en la figura 4.2 como la influencia de las mutaciones, con una probabilidad de ocurrir en 1 de cada 10 individuos, el tamaño variable de los individuos, y el hecho de que incluso los peores individuos tengan posibilidades de ser seleccionados producen variedad entre los individuos, consiguiendo que el fitness no sea estable y existan picos.

Por otro lado en la figura 4.3, si enfrentamos el precio con los usuarios conectados, observamos como es necesaria una alta inversión económica para que el número de usuarios conectados aumente.

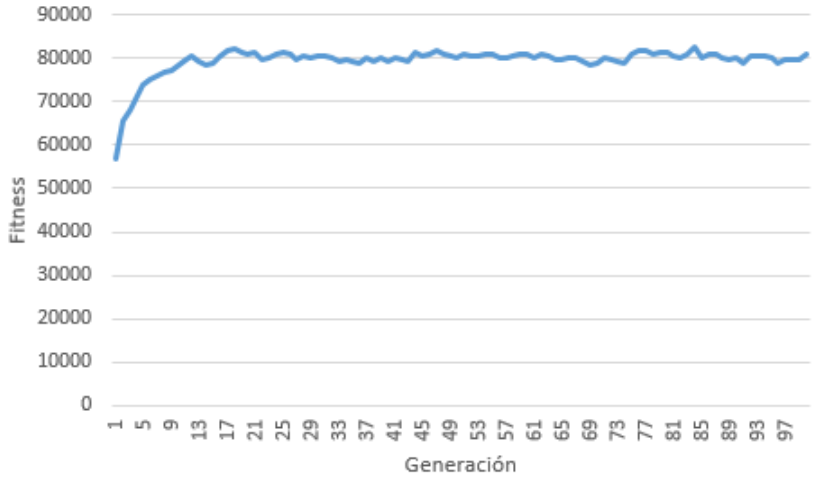

Figura 4.2: Fitness en 100 generaciones

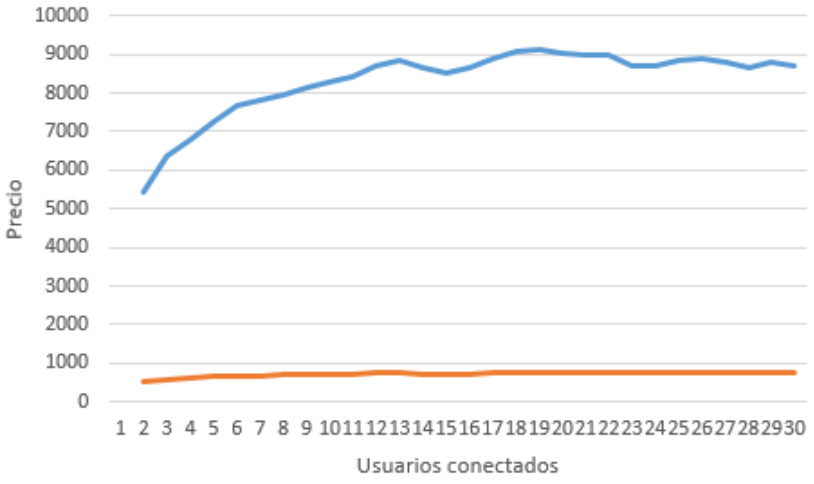

Figura 4.3: Precio VS usuarios conectados

Por último, en la tabla 4.4, comparamos dos individuos de la última generación con un resultado casi inmejorable si tenemos en cuenta el número de personas conectadas:

Fitness	Número de genes	Conectados	Desconectados	Precio
99400.0	191	992	8	13850
99550.0	177	986	14	13100

Cuadro 4.4: Comparación de individuos

Al compararlos podemos observar la influencia del precio en el fitness, haciendo que un individuo con un número de personas inferior tenga un fitness mayor, y por lo tanto mayores probabilidades de ser seleccionado. Esto es debido principalmente a que el segundo individuo es capaz de conectar a tan solo 6 personas menos con una diferencia de 14 puntos de acceso.

5

Conclusiones y trabajo futuro

Como conclusión, podemos afirmar que el problema se ajusta perfectamente a la forma de trabajar de los algoritmos genéticos, siendo capaz de dar una salida que puede ser considerada de óptima y fácilmente configurable, de tal modo que se adapte a las exigencias y necesidades del problema.

Por otro lado, hemos comprobado que los algoritmos genéticos son capaces de funcionar con individuos de tamaño variable siempre y cuando el algoritmo esté correctamente calibrado. La ponderación en el fitness es la parte más importante en esta característica. En nuestro caso se ha tenido que darle más importancia al número de usuarios conectados, ya que, generalmente, cuanto mayor es el tamaño del genotipo más posibilidades tienen los usuarios de conectarse. Como es obvio, por norma general, el número de personas conectadas es directamente proporcional al precio, por lo que si le diésemos más importancia a disminuir el precio que a aumentar el número de personas conectadas acabaríamos con individuos de un solo gen en pocas iteraciones en las que se el precio sería muy bajo pero también el número de usuarios.

Como trabajo futuro queda comprobar su eficacia con datos reales y analizar los resultados dependiendo de las posibles configuraciones con el fin de encontrar la que mejor se adapte.

Glosario de acrónimos

- **CRO**: Coral Reef Optimization
- **ACO**: Ant Colony Optimization
- **GA**: Genetic Algorithm

Bibliografía

[1] Diana Manjarres, Itziar Landa-Torres, Sergio Gil-Lopez, Javier Del Ser, Sancho Salcedo-Sanz, *A heuristically-driven multi-criteria tool for the design of efficient open WiFi access networks*, CAMAD 2012: 80-84.

[2] Itziar Landa-Torres, Sergio Gil-Lopez, Javier Del Ser, Sancho Salcedo-Sanz, Diana Manjarres, José Antonio Portilla-Figueras, *Efficient citywide planning of open WiFi access networks using novel grouping harmony searchheuristics*, Eng. Appl. of AI 26(3): 1124-1130 (2013).

[3] Homaifar, A. ; Dept. of Electr. Eng., North Carolina A&T State Univ., Greensboro, NC, USA ; Turner, J. ; Ali, S., *The N-queens problem and genetic algorithms*, ISBN: 0-7803-0494-2

[4] Andries P. Engelbrecht, *Computational Inteligence*, ISBN: 978-0-470-03561-0

[5] Hartmann, S, *A competitive genetic algorithm for resource-constrained project scheduling*, Naval Research Logistics, 45: 733–750. doi: 10.1002/(SICI)1520-6750(199810)45:7¡733::AID-NAV5¿3.0.CO;2-C

[6] Kosko, B, *Neural networks and fuzzy systems*, Prentice hall.

[7] A. Eiben, P.-E Raué, Z.Ruttkay, Solving constraint satisfaction problems using genetic algorithms *Evolutionary COmputation*, IEEE Wold Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, pages 542-547. IEEE.

[8] Zong Woo Geem, Joong Hoon Kim, G.V. Loganathan, *A New Heuristic Optimization Algorithm: Harmony Search*, doi: 10.1177/003754970107600201

[9] A. Gonzalez-Pardo, D. Camacho, *Solving Resource-Constraint Project Scheduling Problems based on ACO algorithms*, pages 290-291

[10] A. Gonzalez-Pardo, D. Camacho, *Solving Resource-Constraint Project Scheduling Problems based on ACO algorithms*, pages 290-291

[11] A. Gonzalez-Pardo, D. Camacho, *A new CSP graph-based representation for Ant Colony Optimization*, pages 689–696