eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

MINERÍA DE GRAFOS SOCIALES USANDO ALGORITMOS BIOINSPIRADOS

Alumna: Fernández León, Alicia
Director: del Ser Lorente, Javier

Curso: 2018-2019

Fecha: 22, 07, 2019

Resumen Laburpena Abstract

El crecimiento del uso de Internet, en concreto de las redes sociales, ha supuesto una complicación en la gestión de las mismas. En el campo de la minería de grafos, el cual nos permite visualizar redes con usuarios e interacciones entre los mismos, se ha tratado de buscar una forma de simplificación para ese análisis de datos. En este contexto, se plantea la búsqueda de comunidades que comparten atributos mediante algoritmos de optimización que alivien el coste computacional.
En este trabajo, se presenta un algoritmo capaz de particionar una red de alta dimensionalidad de forma óptima, dando lugar a un avance en el campo de la optimización a gran escala, subrama de la heurística surgida recientemente que carece de avances relacionados con la minería de grafos.

Palabras Clave: red social, minería de grafos, comunidad, optimización.

Interneteko erabileraren hazkundea, bereziki sare sozialak, konplikazio bat izan da horren kudeaketan. Grafikoen meatzaritzaren esparruan, erabiltzaileei eta horien arteko elkarrekintzak sarea bistaratzeko aukera ematen duena, datu horiek aztertzeko sinplifikazioa bilatzen hasi da. Testuinguru horretan, konputazionalaren kostua arintzen duten optimizazio algoritmoen bidez atributuak partekatzen dituzten komunitateen bilaketa sortzen da.
Lan honetan, tamaina handiko dimentsionaltasun sare bat partizionatzeko gai den algoritmoa aurkezten da, eskala handiko optimizazioaren alorrean aurrerapena sortuz, duela gutxi grafoen meatzaritzarekin lotutako aurrerapenik ez duen heuristikaren azpialdean.

Gako-hitzak: sare soziala, grafikoen meatzaritza, komunitatea, optimizazioa.

The increase in Internet use, in particular social networks, has been a complication in the management of the same. In the field of graph mining, which allows us to visualize a network with users and interactions between them, we have begun to look for a simplification of them for the analysis of the data. In this context, the search for communities that share attributes is raised using optimization algorithms that alleviate the computational cost.
This paper presents an algorithm capable of optimally dividing a high-dimensional network, leading to an advance in the field of large-scale optimization, a sub-branch of the recently emerged heuristic that lacks advances related to graph mining.

Keywords: social network, graph mining, community, optimization.

Índice

Lista de figuras

Lista de tablas

Lista de acrónimos

CEC Congress on Evolutionary Computation

DE Differential Evolution

ES Evolution Strategy

GA Genetic Algorithm

GP Genetic Programming

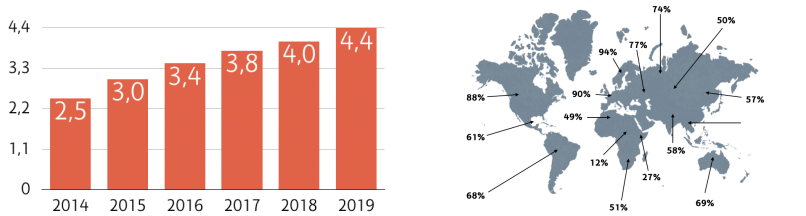
LSGO Large-Scale Global Optimization

RC Relaxed Caveman

LFR Lancichinetti–Fortunato–Radicchi

1. Introducción

Las redes sociales son una tendencia en alza que ha generado un gran impacto en la sociedad. Actualmente, según el estudio 2019 Digital Yearbook publicado por Hootsuite y We Are Social [1] , 4.38 billones de personas se conectan a Internet, de las cuales 3.48 billones son usuarios activos de redes sociales.



Usuarios de Internet, en billones. Inserción de Internet por regiones.

Figura 1: 2019 Digital Yearbook

Este incremento del uso de Internet, concretamente de las redes sociales, supone dificultad en la gestión y el análisis de las mismas. En consecuencia, se buscan nuevas maneras de automatizar estos procesos, logrando así proporcionar nuevos servicios. En este caso, se trata de modular y se plantea como un problema de aprendizaje computacional que nos va a permitir predicción de contactos, recomendaciones, detecciones de comunidades, etc. Este trabajo se centra en la detección de comunidades, la cual es muy útil a la hora de descubrir círculos de influencia o de células terroristas.

Pero, ¿qué es la detección de comunidades? Una red puede ser modelada como un grafo G=(V, E), donde V representa M nodos o vértices y E representa las aristas entre elementos V. En lenguaje adaptado a las redes sociales, un nodo representa a un usuario de la red, mientras que las aristas representan la interacción entre los usuarios. La detección de comunidades se basa en encontrar grupos de nodos similares entre sí y disimilares con los nodos de otros grupos. Esa similaridad está basada en las aristas, las cuales tienen ciertas características como el peso o la direccionalidad que influyen en la fuerza de esa unión.

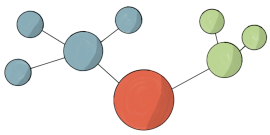


Figura 2: Ejemplo simplificado de grafo.

Este trabajo tratar de resolver problemas de detección de comunidades con nuevos algoritmos evolutivos apropiados para manejar grandes dimensionalidades.

2. Contexto

El término comunidad, proveniente del latín, puede ser definido como una asociación o grupo de individuos que comparten atributos, propiedades o relaciones.Un individuo puede formar parte de varias comunidades si comparte atributos con ellas, por lo que las comunidades pueden solaparse. De forma matemática, es mucho más complejo encontrar una definición exacta, ya que existen muchos inconvenientes a la hora de particionar o dividir las redes en comunidades. Por ello, se han desarrollado diferentes métodos de detección, los cuales se pueden dividir en cuatro grandes tipos, tal y como se puede observar en la figura 3. El trabajo va a centrar su estudio en los dos primeros.

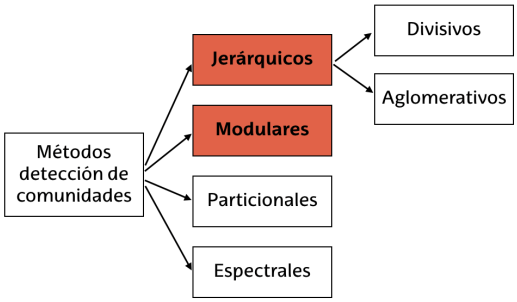


Figura 3: Tipos de métodos de detección de comunidades.

2.1. Métodos jerárquicos

Los métodos jerárquicos tienen como objetivo la búsqueda de divisiones naturales de la red, esto se consigue agrupando clusters para formar uno nuevo o separando alguno existente para dar lugar a otros nuevos. Esto se basa en que las comunidades pueden estructurarse de forma jerárquica y ser representadas mediante un dendograma o árbol de clasificación, tal y como se puede observar en el ejemplo de la siguiente figura. A medida que se va construyendo el árbol, se van juntando los nodos representados por puntos, hasta llegar a un criterio de parada, quedando así una red formada por cuatro agrupamientos.

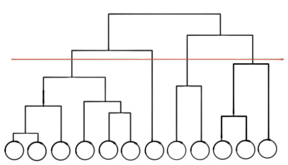


Figura 4: Ejemplo simplificado de dendograma.

Los métodos jerárquicos están a su vez divididos en dos tipos. Por un lado, los métodos aglomerativos o ascendentes empiezan en análisis con el mismo número de nodos que de agrupaciones. A partir de ese punto, se empiezan a crear comunidades, de forma ascendente tal y como su nombre indica, hasta que todos los nodos quedan en el mismo conglomerado. Por otro lado, los métodos divisivos o descendentes, siguen el proceso inverso al anterior. Partiendo de que todos los nodos forman parte del mismo conglomerado, se van realizando divisiones, de manera que se van formando grupos cada vez más pequeños.

Dentro de esta metodología, se pueden encontrar notables algoritmos como el algoritmo de Radicchi o el algoritmo de Girvan y Newman, el cual se analiza más en profundidad en la sección de alternativas. El problema de los métodos jerárquicos es la inexistencia de una variable que permita valorar las diferentes soluciones propuestas. Ante esto, Girvan y Newman proponen la modularidad.

2.2. Métodos modulares

La modularidad es uno de los primeros intentos de conseguir una función de calidad que atendiera a todos los elementos esenciales a la hora de definir una comunidad, desde la selección de las comunidades hasta la comprobación de la fortaleza de las mismas. Esta función es la que se utiliza en el trabajo, por lo que más adelante, en la sección de solución, se hará un análisis más en profundidad de su funcionamiento. Debido a que es una estrategia de maximixación, este se convierte en un problema de optimización numérica, por lo que resulta natural pensar en la heurística para resolver este tipo de problemas. El objetivo es optimizar el valor de modularidad y resolver el problema en tiempos razonables.

Algunos de los algoritmos más conocidos en esta metodología son el algoritmo de Newman, porteriormente mejorado por Clauset, y por otro lado, el algoritmo de Blondel, conocido como algoritmo Louvain, el cual presenta mejores resultados en términos de modularidad que el anterior.

Estas cuatro técnicas de detección de comunidades no son utilizadas únicamente para el estudio de redes sociales, sino que han supuesto también un gran avance en múltiples campos de investigación como el análisis de Redes de Interacción de Proteínas.

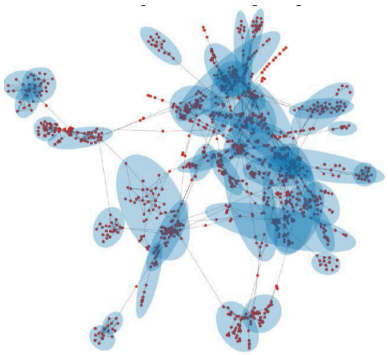


Figura 5: Ejemplo de comunidades en una proteina.

3. Estado del arte

El crecimiento de las redes sociales ha supuesto un gran problema en el área de la minería de grafos, el cual se ha centrado en estudiar diferentes campos como Ego Networks, cadenas de confianza o detección de comunidades. Este trabajo se centra en este último, el cual se trata de un problema de optimización.

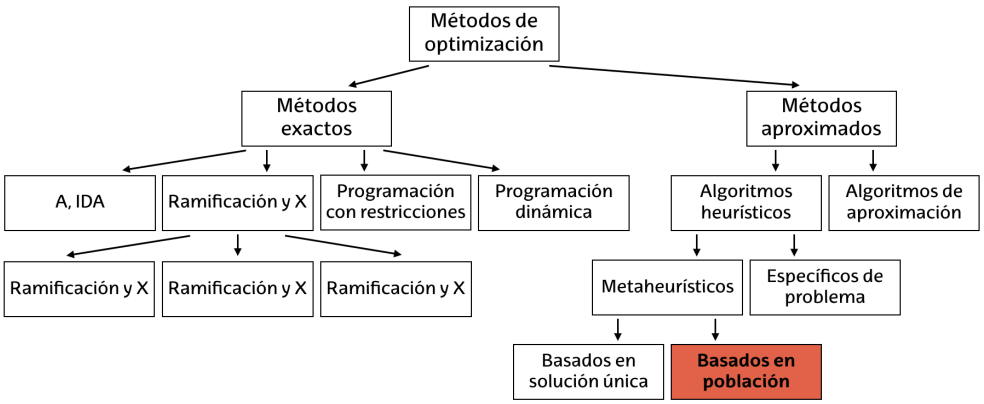


Figura 6: Tipos de métodos de optimización.

La figura 6 muestra un esquema organizativo de los distintos tipos de métodos de optimización. Este proyecto está basado en los algoritmos bioinspirados o heurística basada en la población, los cuales son uno de los campos más relevantes en el diseño de algoritmos. Los algoritmos bioinspirados replican la forma que tiene la naturaleza de enfrentarse a un problema o desafío. Son de carácter adaptativo, ya que se realimentan, y ofrecen poblaciones de soluciones, no respuestas únicas. Estos algoritmos no sólo están presentes en la minería de grafos o en el reconocimiento de patrones, se utilizan en diversos campos como el diseño de fármacos, diseño de tácticas militares, el diseño de estructuras como antenas de telecomunicación, etc.

Dentro de los algoritmos bioinspirados podemos encontrar varios sistemas biológicos como las redes neuronales, inteligencia de enjambre o computación evolutiva. En la categoría de computación evolutiva, los métodos más utilizados son los algoritmos genéticos GA, la evolución diferenecial DE, las estrategias evolutivas ES o y la programación genética GP. Este trabajo se centra en los algoritmos de tipo computación evolutiva, en concreto en los genéticos, los cuales son, dentro del campo de la optimización, unos de los que mayor utilidad presentan.

Los algoritmos genéticos modifican de forma repetida durante una serie de generaciones una población de soluciones. En cada generación, el algoritmo escoge soluciones de la población y genera con ellas un hijo, el cual combina las mejores características de ambos. Este proceso se divide en varias fases, tal y como muestra en la figura 7.

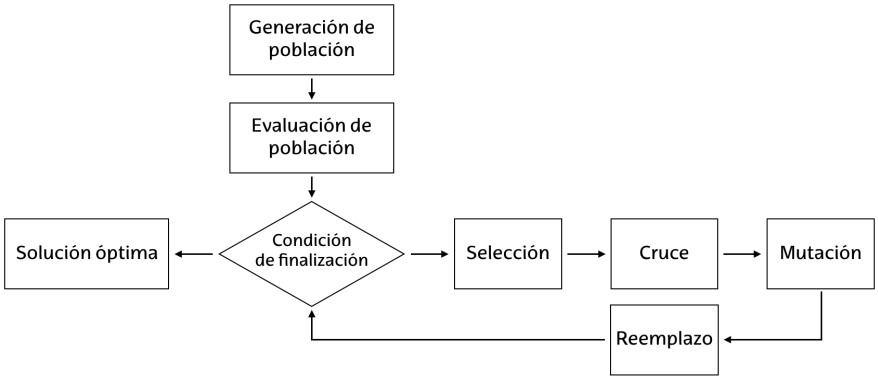


Figura 7: Diagrama básico de algoritmo genético.

- **Generación de población**
  Se genera una población inicial aleatoria formada por diferentes candidatos, todos ellos posibles soluciones del problema.
- **Evaluación de la población**
  Se evalua cada uno de los candidatos para conocer cuán óptima es la solución.
- **Condición de finalización**
  Se establece un número de evaluaciones o un límite de margen de mejora que marca el final del algoritmo y la obtención de la solución más óptima.
- **Selección**
  Se escogen los candidatos que van a ser cruzados y mutados de cara a la siguiente generación. Los candidatos que obtienen una mejor respuesta ante la función de calidad tienen mayor probabilidad de ser escogidos.
- **Cruce**
  Es el principal operador, el cual combina dos soluciones de la población para generar descendientes con mejores características.
- **Mutación**
  Con el objetivo de disminuir el espacio de búsqueda se realiza una modificación aleatoria de la población.
- **Reemplazo**
  Selección de los mejores individuos de la nueva población.

Estos algoritmos se pueden aplicar para la detección de comunidades, pero convergen lentamente ya que no están adaptados a grandes escalas. Esta es la hipótesis de investigación del proyecto. En los últimos tiempos, ha surgido una subrama en la heurística bioinspirada, LSGO, Large Scale Global Optimization. En LSGO, los algoritmos se modifican para tratar con diferentes problemas: la existencia de muchas variables, muchos objetos y una gran cantidad de restricciones. En esta rama existen varias opciones para tratar problemas genéricos, pero no existe nada en el ámbito de la minería de grafos. La propuesta de este proyecto es LSGO para partición de comunidades en grandes redes.

4. Objetivos y alcance

4.1. Objetivo

El objetivo principal de este trabajo es, basándose en la estructura que sigue el algoritmo SHADEILS premiado en la CEC 2013, obtener un algoritmo bioinspirado que permita formar comunidades en un grafo social de alta dimensionalidad. Este algoritmo será un nuevo paso en la rama de la heurística LSGO, en la cual apenas se han realizado estudios relacionados con la minería de grafos.

4.2. Alcance

Para definir el alcance del proyecto se establecen una serie de objetivos los cuales se distribuyen en un determinado plazo de tiempo. En este caso, el proyecto se va a dividir en tres fases dependientes una de la otra, es decir, es necesario el correcto funcionamiento de todas ellas para cumplir con el objetivo principal. El orden de realización va a ser el que se plantea a continuación:

- **Fase 1: Minería de grafos**
  Simular redes de alta dimensionalidad, es decir, un alto número de nodos y variaciones en las conexiones intercomunidad e intracomunidad. Para ello, se han escogido dos benchmark conocidos, los cuales se comentan en profundidad en el apartado de solución.
- **Fase 2: Clusterización**
  Obtener, mediante el algoritmo de optimización creado, soluciones óptimas de particiones de comunidades en la red de alta dimensionalidad creada en la fase anterior. El algoritmo ha de obtener buenos resultados en un número razonable de evaluaciones y generaciones, para cumplir con su objetivo de aliviar coste computacional.
- **Fase 3: Análisis de resultados**
  Realizar una comparativa de los resultados obtenidos en la fase anterior, respecto a las particiones que propone el benchmark y los resultados obtenidos con el algoritmo de referencia en el campo, Girvan y Newman.

5. Beneficios

De cara a conocer la viabilidad del proyecto, resulta interesante realizar un análisis de los beneficios que presenta. Este punto se centra en los beneficios técnicos, sociales y económicos, de los cuales el primero es el principal.

5.1. Técnicos

Tal y como se ha visto en el estado del arte, los algoritmos actuales dentro de la computación evolutiva no se encuentran adaptados a altas dimensionalidades, se pueden aplicar pero convergen lentamente. Para ello, ha surgido la subrama de la heurística LSGO. Este algoritmo se presenta como un avance en el campo de la minería de grafos, la cual no ha sido tratada en LSGO. Por otro lado, en el contexto de las redes sociales, será un avance de cara a las empresas creadoras, ya que permitirá mejorar la gestión de las mismas y adaptar las funcionalidades que ofrecen en base al estudio de los usuarios y relaciones entre ellos.

5.2. Sociales

Este proyecto tiene como objetivo principal el análisis y agrupación de los usuarios según sus atributos en común. Esto supondrá una mejora de las redes sociales, no sólo de cara a la programación de esta, tal y como se ha mencionado anteriormente, sino de cara a sus usuarios. Gracias al algoritmo se podrán obtener redes sociales más inteligentes, más sociales y más personalizadas, es decir, el conocimiento de la estructura de los usuarios permite adaptar la red social para un mejor uso de estas: desde recomendaciones adaptadas al usuario, hasta reconocimiento de círculos sociales, recomendación de perfiles, etc.

5.3. Económicos

Este proyecto no está orientado al beneficio económico, ya que su principal objetivo, tal y como se ha mencionado anteriormente, es la investigación y el estudio en la nueva subrama heurística para facilitar la gestión de grandes redes.

6. Análisis de alternativas

6.1. Lenguaje

El lenguaje escogido para realizar el proyecto es Python, ya que es el lenguaje por excelencia para trabajar en el campo de la optimización y de la minería de grafos. Con Python se dispone de todas las herramientas de forma más sencilla que con otros lenguajes de programación, incluida la librería Networkx [21], la cual permite trabajar con los diferentes benchmark, la función de calidad y el algoritmo alternativo escogidos.

	Java	**Python**
Sencillez (30 %)	9.0	10.0
Rendimiento (30 %)	7.0	7.0
Soporte (30 %)	5.0	10.0
Coste de económico (10 %)	10.0	10.0
TOTAL	7.3	**9.1**

Tabla 1: Comparativa de lenguajes.

6.2. Solución

En Networkx se encuentran disponibles una gran cantidad de generadores para estudiar las redes sociales. Se han seleccionado varios y realizado una comparativa de lo que ofrecen. LFR se ha escogido principalmente porque proporciona un atributo comunidad para cada nodo que permite conocer la partición teórica de la red. Por otro lado, se ha escogido Relaxed Caveman sobre los otros restantes porque los parámetros que incluye se adaptan más a las necesidades del proyecto.

	Caveman	**Relaxed Caveman**	**Gaussian**	**LFR**
Sencillez (30 %)	10.0	10.0	10.0	6.0
Parámetros (30 %)	6.0	9.0	8.0	8.0
Comunidades (30 %)	0.0	0.0	0.0	10.0
Soporte (10 %)	10.0	10.0	10.0	10.0
TOTAL	5.8	**6.7**	6.4	**8.2**

Tabla 2: Comparativa de generadores.

Para la creación del algoritmo no se plantea ninguna alternativa ya que no se hace uso de códigos externos. La estructura base a seguir va a ser la de SHADEILS, el cual fue premiado en la CEC 2013 por los resultados obtenidos.

7. Análisis de riesgos

En este apartado se evalua los eventos que se pueden ocurrir e impactar negativamente en el proyecto. Para la evaluación de estos riesgos se va a utilizar una metodología basada tres pasos: identificación del riesgo, evaluación del mismo y elaboración de un plan de contingencia para un menor impacto en el caso de que suceda.

7.1. Identificación de los riesgos

Se ha realizado un estudio para identificar los posibles riesgos que pueden surgir en el proyecto. Los riesgos son los siguientes:

- Error en el diseño
- Error en el presupuesto
- Superación de plazo planificado
- Mala gestión de los recursos

7.2. Evaluación de los riesgos

Para evaluar ese riesgo hay que tener en cuenta dos factores importantes, la probabilidad de que suceda y el impacto que tendría en nuestro proyecto. Esto se puede mostrar de forma más visual en la matriz probabilidad-impacto.

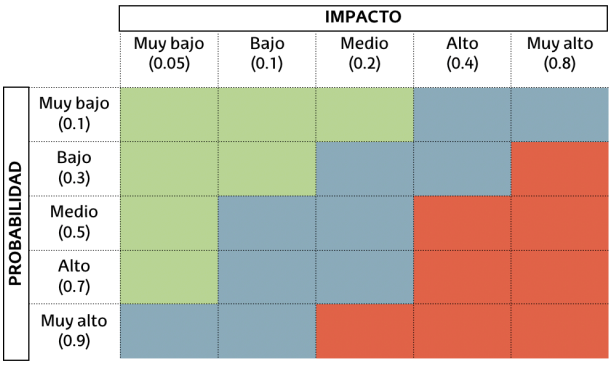


Figura 8: Matriz probabilidad impacto.

- **Error en el diseño (A)**:
  En este caso un mal diseño puede llevar a graves consecuencias en el caso de no ser detectado a tiempo, ya que daría lugar a resultados inferiores. El impacto es alto, pero la probabilidad de que ocurra es medio, ya que se puede plantear algo teórico que no funcione como estaba planeado.

- **Error en el presupuesto (B)**:
  Se ha calculado erróneamente bien porque no se ha tenido en cuenta algún factor o bien porque alguno de los factores se ha supuesto de forma errónea. En este caso, el impacto será medio, aunque dependerá de si el presupuesto se ha calculado por encima o por debajo del final. Por otro lado, se ha considerado una probabilidad media de aparición.

- **Superación de plazo planificado (C)**:
  Esto se debe a que alguna de las tareas no ha cumplicado con su tiempo asignado y puede suponer la pérdida de calidad del proyecto. En este caso, el impacto será medio, ya que, como anteriormente se ha comentado, el proyecto puede disminuir considerablemente su calidad. En cuanto a la probabilidad, esta se puede considerar baja, ya que se realiza un complejo estudio inicial para realizar la planificación.

- **Mala gestión de los recursos (D)**:
  Una mala gestión de los recursos puede dar lugar a pérdidas y por tanto dar lugar a un problema en la planificación o en el presupuesto. El impacto es medio y la probabilidad de que ocurra es baja, ya que, al igual que en la planificación, se realiza un complejo estudio anterior.

Riesgo	Probabilidad	Impacto	Resultado
A	0.5	0.4	0.2
B	0.5	0.2	0.1
C	0.3	0.2	0.06
D	0.1	0.2	0.02

Tabla 3: Tabla de riesgos.

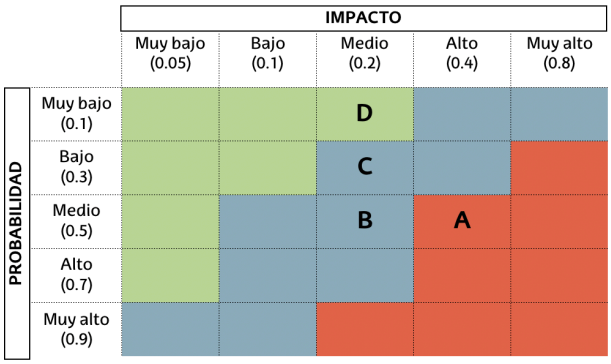


Figura 9: Resultado matriz probabilidad impacto.

7.3. Plan de contingencia

Para realizar el plan de contingencia en primer lugar hay que ordenar los riesgos en función del resultado de probabilidad-impacto obtenido en el paso anterior. De esta forma, es más sencillo evaluar el nivel de importancia que tienen y la necesidad de actuación sobre cada uno de ellos.

Figura 10: Orden de riesgos según su probabilidad-impacto.

Los planes de contingencia se realizan para cada riesgo de forma individual y se centra en prevenirlo o en caso de que ocurra, tener soluciones para ellos. Los planes de contingencia individuales en este proyecto son los siguientes.

- **Error en el diseño (A)**:
  Para evitar fallos, lo más óptimo es la revisión ocasional del funcionamiento de los bloques de código mediante fases de prueba de software para cada uno de ellos y pruebas globales una vez terminado el código. Las pruebas de bloques nos permitirán analizar la eficiencia de los mismos de cara a la mejora del proyecto final.

- **Error en el presupuesto (B)**:
  La forma más óptima de prevenir errores en el presupuesto es tener en cuenta los posibles imprevistos añadiendo una partida en el presupuesto para ellos. Esta es necesaria debido a que es difícil conocer o predecir con exactitud las necesidades futuras que se van a plantear en el proyecto.

- **Superación de plazo planificado (C)**:
  La mejor forma es prevenir esto es realizar un estudio de completo del proyecto antes de la creación de la planificación y hacer revisiones semanales del estado de la fase en la que se encuentra el proyecto en ese momento mediante entregables puntuales. Si se da el caso, se ha de presentar un mínimo de datos válidos que muestren el funcionamiento del algoritmo, pero se deberá ampliar la cantidad de cara a la presentación.

- **Mala gestión de los recursos (D)**:
  Para evitar la mala gestión de los recursos y gastos injustificados, lo más óptimo es realizar una revisión mediante un libro de cuentas que nos permita llevar un control de gastos.

8. Descripción de la solución

La solución se divide en tres fases, las cuales se irán analizando en profundidad en las siguientes secciones. En primer lugar, se crea una red de alta dimensionalidad, se aplica el algoritmo para esa red y por último, se analiza la solución obtenida.

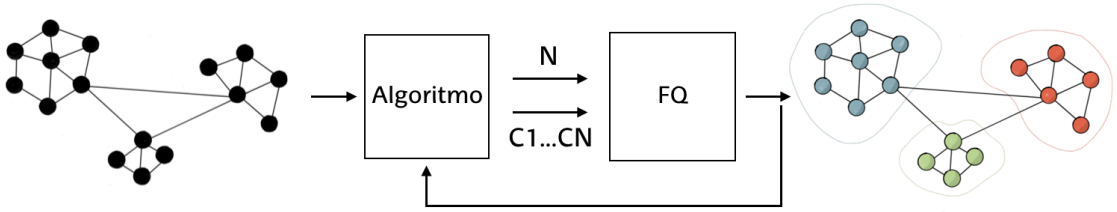


Figura 11: Esquema general de la solución planteada.

8.1. Creación de la red

Para poder evaluar el funcionamiento del parámetro modularidad, es necesario tanto el propio algoritmo como un algoritmo generador de redes. En este trabajo, se va a hacer uso de dos benchmark diferentes, los cuales se caracterizan por generar redes en las cuales los tamaños de las comunidades siguen leyes de potencia.

8.1.1. Relaxed Caveman

El benchmark RC es sencillo de usar y permite configurar la probabilidad de existencia de aristas mediante el parámetro p, cuanto mayor sea este valor, más estarán conectados los nodos de la red. Para este proyecto se ha escogido el valor 0 para obtener una red limpia y 0.1 para el uso del algoritmo. El parámetro l indica el número de comunidades presentes en el grafo, mientras que k indica el número de nodos en cada comunidad definida, por lo tanto, el número total de nodos en la red será el producto de ambos.

```
1 G=relaxed_caveman_graph(l, k, p)
```

Figura 12: Código Python generador RC.

8.1.2. Lancichinetti–Fortunato–Radicchi

LFR es un algoritmo generador que devuelve un grafo G con una estructura conocida, esto se debe a que para cada nodo tiene un atributo "community" que almacena en qué comunidad se encuentra en la solución ideal. La modularidad que se obtiene con esas comunidades va a ser considerada como modularidad teórica.

```
1 G=LFR_benchmark_graph(n, tau1, tau2, mu, average_degree=None)
```

Figura 13: Código Python generador LFR.

Para el proyecto se ha escogido $tau1$, $tau2$ y $average\_degree$ basandose en estudios del funcionamiento del algoritmo [12], siendo $tau1$ y $tau2$ constantes con valor 2 y $average\_degree$ $0,1 * n$. En cuanto al valor de μ este corresponde con $1 - p$, siendo p el valor explicado en el benchmark anterior, por lo que tomará un valor de 0,9. Por último, el número de nodos de la red o n, el cual será la variable principal del proyecto, ya que servirá para analizar la eficacia del algoritmo para diferentes tamaños de redes.

8.2. Algoritmo

El objetivo del algoritmo es buscar una solución óptima en una nube de posibles soluciones. De forma matemática el problema se presenta como un espacio S, en el cual existe una solución óptima o factible $x_o \epsilon S$, la cual cumple $f(x) \leq f(x_o)$ para $x \epsilon S$, donde f es la función fitness o de calidad. Las soluciones alternativas se representan como el conjunto finito de variables $\{X_i : i = 1, 2, ..., n\}$ en un dominio de $U = \{x = (x_i : i = 1, 2, ..., n) : x_i \epsilon U_i\}$. La siguiente figura muestra un ejemplo simplificado de las diferentes soluciones, o candidatos para este proyecto, que se pueden dar.

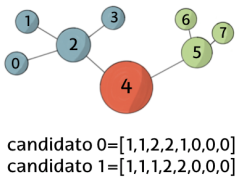


Figura 14: Ejemplo simplificado de dos soluciones.

Para ello, basándose en la estructura de SHADEILS [16], el algoritmo está formado por dos bloques de optimización, una búsqueda global y una búsqueda local. La búsqueda local es un proceso que, dada la solución actual en la que se encuentra la generación, selecciona una solución de su entorno. Esto es un inconveniente, ya que sólo puede obtener una solución localmente óptima y por lo tanto, la solución actual queda atrapada en ese entorno. Por ello, se utiliza en combinación con la búsqueda global, la cual ofrece una perspectiva más completa de la nube de soluciones.

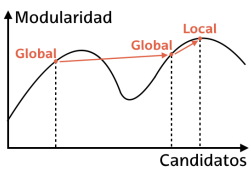


Figura 15: Gráfica ejemplo simplificada de la función de calidad-iteraciones.

Para cada evaluación, se propone una población $population$ formada por una serie de candidatos, es decir, soluciones. En primer lugar, se realiza una búsqueda global que devuelve una población nueva del mismo tamaño. En segundo lugar, se coge el mejor candidato de esa nueva población y se le aplica búsqueda local. Si el el candidato obtenido es mejor que la solución más óptima de la evaluación $bestsolution$, se sustituye y se guarda su valor fitness. A continuación se presenta un pseudocódigo con el funcionamiento explicado.

```
1  for _ in range(run) do
2      population<-random_populations()
3      actualsolution<-population(best)
4      bestsolution<-actualsolution
5      bestmod=modularity(G,bestsolution)
6      while totalgen<maxgen do
7          population<-busqueda_global(population,actualsolution)
8          actualsolution<-population(best)
9          actualsolution=busqueda_local(actualsolution,ils,G)
10         population(best)<-actualsolution
11         mod=modularity(G,actualsolution)
12         if mod>bestmod then
13             bestsolution<-actualsolution
14             bestmod=mod
15         end
16     end
17 end
```

Figura 16: Pseudocódigo del algoritmo.

8.2.1. Búsqueda global

La búsqueda global sigue la idea del esquema planteado para algoritmos evolutivos, pero se divide en dos fases diferentes. Ésta devuelve una población formada por los mejores candidatos de la población antigua y la población obtenida en la búsqueda.

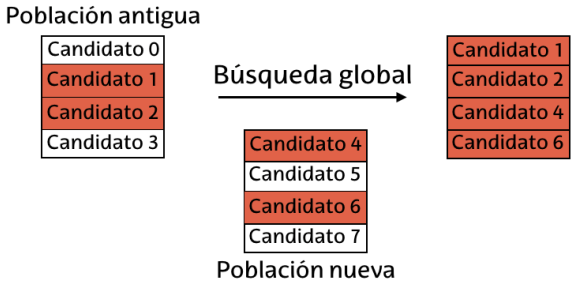


Figura 17: Población en la búsqueda global.

De cara al pseudo código es importante destacar que ambas fases se encuentran en el mismo bucle for que recorre la población. Se ha realizado la división del código para facilitar la comprensión del mismo.

1. **Fase 1**
   Mezcla de la situación actual con la diferencia entre dos soluciones de la población cogidas al azar, detectanto los nodos ubicados en comunidades diferentes y variando en función de una probabilidad frc la comunidad a la que pertenecen esos nodos sumando o restando una unidad al valor que tienen asignados dichos nodos.

```
for (p,xp) enumerate (population) do
    auxiliar<-xp
    a<-population(random)
    b<-population(random)
    for z in range (len(a)) do
        arrayA<-nodosVecinos()
        arrayB<-nodosVecinos()
        for (k,xk) in enumerate(arrayA) do
            for (l,xl) in enumerate(arrayB) do
                if xk==xl then
                    coincidencias+=1
                end
            end
        end
        tanto=coincidencias/len(array) ;
        if tanto<0.5 then
            if random()<frc then
                auxiliar[z]+=1 ;
            else
                auxiliar[z]-=1 ;
            end
        end
    end
end
```

Figura 18: Código búsqueda global fase 1.

2. **Fase 2**
   Acercamiento de la solución actual a la mejor solución de la población en el instante actual: se coge la solución con mayor modularidad y la solución a mutar. Con una probabilidad frc, se copia cada comunidad de la mejor solución en la segunda.

```
for i in range(1,np.max(best_solution)+1) do
    arrayC<-nodosVecinos()
    if random()>=frc then
        for (j,xj) in enumerate(arrayC) do
            auxiliar[xj]=i;
        end
    end
end
solutions.append(auxiliar)
newpopulation<-solutions(bests)
```

Figura 19: Código búsqueda global fase 2.

8.2.2. Búsqueda local

La búsqueda local consiste en analizar las aristas que conectan a un nodo e ir variando su pertenencia a una comunidad en función de una probabilidad definida frc. Para ello, se escoge al mejor candidato obtenido en la búsqueda global y se recorre elemento a elementos realizando las siguientes operaciones.

Figura 20: Búsqueda local.

```
1  for z in range(ils) do
2  |  auxiliar<-actualsolution
3  |  for (k,xk) in enumerate(population): do
4  |  |  lines=listaAristas()
5  |  |  if random<frc then
6  |  |  |  auxiliar[k]=lines[random]
7  |  |  end
8  |  end
9  |  if modularity(auxiliar)>modularity(actualsolution) then
10 |  |  actualsolution<-auxiliar
11 |  else
12 |  |  auxiliar<-actualsolution
13 |  end
14 end
```

Figura 21: Código búsqueda local.

8.3. Función de calidad

La modularidad (Q) es una función que mide la calidad de una partición concreta de una red en comunidades. Se trata, según su definición, de la diferencia entre el número de enlaces que existen en los grupos y el número de enlaces esperado en una red aleatoria equivalente:

$$Q = \frac{1}{2L} \sum_{ij} [A_{ij} - \frac{k_i - k_j}{2L}] \delta(c_i, c_j) \tag{8.1}$$

Puede tomar valores entre -1 y 1. Cuanto mayor es su valor, más óptima es la partición, es decir, el número de enlaces internos de las comunidades es mayor de lo que cabría esperar de forma aleatoria y las comunidades se encuentran dispersamente conectadas. Una característica muy útil de la modularidad es que no solamente indica qué tan buena es una partición mediante valores positivos, sino que también puede describir qué tan malo es un agrupamiento mediante valores negativos.

9. Descripción de los resultados

9.1. Girvan y Newman

Girvan-Newman se encuentra dentro de los métodos de detección de comunidades jerárquicos, concretamente, los métodos divisivos mencionados anteriormente en el estado de arte. Esto implica que la idea que sigue es, partiendo de una comunidad de gran tamaño, ir dividiéndola eliminando los enlaces de baja similitud.

La idea que sigue el algoritmo es que las aristas que conectan módulos separados tienen un "Betweenness" mayor, ya que todas las rutas más cortas de un módulo tienen que atravesarlas. Si vamos eliminando las aristas con valor "Betweenness mayor", obtendremos un mapa jerárquico o dendograma de ese grafo.

El valor de "Betweenness" se calcula como:

$$\sum_{s,t \neq v} = \frac{\sigma_{st}(e)}{\sigma_{st}} \tag{9.1}$$

donde:
σ es el número total de rutas más cortas desde el nodo s hasta el nodo t,
$\sigma_{st}(e)$ es el número de las rutas que pasan a través de e,

El algoritmo se basa en tres pasos:

1. Encontrar la o las aristas con mayor "Betweenness" y eliminarlas del grafo. Esto puede hacer que la gráfica se separe en múltiples modulos. Si es así, este es el primer nivel de regiones en la partición del gráfico.
2. Se vuelven a calcular los "Betweenness" y se vuelven a eliminar las aristas con mayores valores. Esto puede romper alguno de los módulos existentes en módulos menores, estos son regiones anidadas.
3. Se continúa con el proceso mientras haya aristas en el grafo, volviendo a calcular en cada paso el "Betweenness" y eliminando las aristas con valores más altos.

Este método, tal y como se menciona en el contexto, proporciona una serie de divisiones de la red en comunidades más pequeñas, pero no proporciona un análisis de esas divisiones, para ello se utilizará la modularidad.

9.2. Métrica NMI

NMI es la normalización de la puntuación de la información mutua que nos permite escalar los resultados entre 0 y 1, donde 0 significa la inexistencia de información mutua y 1 la perfecta correlación.

$$NMI(Y;C) = \frac{2 * I(Y;C)}{[H(Y) + H(C)]} \tag{9.2}$$

donde:
Y=etiquetas de clase,
C=etiquetas de cluster,
I(Y;C)=información mutua entre Y y C,
H( )=entropía
Los logaritmos presentes en esta ecuación son en base 2.

9.3. Valoración del resultado

Para la valoración del resultado se realizarán dos procesos. Por un lado se realizará una comparativa de las modularidades obtenidas en cada caso. Para el caso de RC, se compararán la modularidad según el resultado de Girvan-Newman, según el resultado del algoritmo y la modularidad limpia, es decir, la modularidad cuando en ese grafo la p es 0. Para el caso de LFR, se comparán los dos primeros anteriores y la modularidad teórica que ofrece la red LFR. Por otro lado, se utilizará la métrica NMI, la cual compara los arrays de las mejores soluciones de los casos.

Se han realizado una serie de pruebas acerca del funcionamiento, todas ellas adaptadas a la capacidad de ejecución del dispositivo adquirido. Las redes menores de 500 nodos trabajan con los parámetros de 200 generaciones durante 10 evaluaciones diferentes y un número total de 5 iteraciones para cada búsqueda local. Por otro lado, para redes mayores de 500 nodos el número de iteraciones se mantiene, mientras que, en este caso, se realizan 30 generaciones durante 5 evaluaciones diferentes, ya que el coste computacional es mayor. El tamaño de población de soluciones escogido para ambos casos es de 50. Todos los datos obtenidos se guardan en un CSV que sigue la forma de las siguientes figuras.

1817	8	199	0.18278462148235136	1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 1 1 0 3 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 3 0 1
1818			0.18278462148235136	1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 3 3 1 1 0 3 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 3 0 1
1819	Run	Generacion	Modularidad	Solucion
1820	9	0	0.05147639054036193	2 0 2 2 3 0 2 3 0 1 0 2 3 1 0 4 1 3 2 2 3 2 3 0 1 2 0 3 3 2 0 1 1 0 1 2 1 0 1 1 0 2 1 0 0 1 1 4
1821	9	1	0.05277643017571664	4 0 3 4 3 0 4 3 0 1 0 4 3 1 0 4 1 3 1 1 3 4 3 0 1 3 0 3 3 1 0 1 1 0 1 2 1 0 1 1 0 2 1 0 0 1 1 4
1822	9	2	0.0702859558726395	4 3 3 4 3 0 4 3 2 0 2 4 3 0 0 4 0 3 4 3 3 4 3 3 4 3 4 3 3 1 1 4 3 0 2 2 1 4 2 2 1 2 4 1 0 1 2 4
1823	9	3	0.08396395825076192	2 3 3 2 3 0 2 3 0 0 0 2 3 0 0 4 0 3 2 3 3 2 3 3 1 3 0 3 3 1 1 1 3 0 1 2 1 0 1 1 1 2 1 1 0 1 1 4

Figura 22: Muestra del fichero a lo largo de generaciones.

2021	Limpia	GN	Algoritmo	NMI Limpia-GN	NMI Limpia-Algoritmo	NMI GN-Algoritmo
2022	0.254104161	0.29167435592	0.7078763376932433	0.5526044356745	0.36056955254089057	0.2959059321737013

Figura 23: Muestra datos finales fichero.

A continuación, se muestra un par de tablas de resumen con algunos de los datos obtenidos durante las pruebas. Se han omitido decimales respecto al CSV, ya que no son tan relevantes de cara al análisis.

RC	Comunidades	Modularidad limpia Modularidad algoritmo	NMI limpia-algoritmo Modularidad GN	NMI limpia-GN NMI algoritmo-GN
50	10	-0.1044 0.1043	0.2247 -4.16e-06	0.0198 0.0346
150	5	0.7078 0.2541	0.3605 0.2916	0.5526 0.2959
150	15	0.8230 0.5218	0.6969 0.1129	0.2995 0.1671
300	5	0.8079 0.2116	0.2947 0.1651	0.3745 0.0821
300	15	0.8368 0.3263	0.4983 0.1140	0.3000 0.0870
600	10	0.8094 0.1439	0.2171 0.1623	0.3750 0.0698
600	15	0.8420 0.1697	0.3003 0.1138	0.3383 0.0839

Tabla 4: Tabla de resultados RC.

LFR	Modularidad teórica Modularidad algoritmo	NMI teórica-algoritmo Modularidad GN	NMI teórica-GN NMI algoritmo-GN
200	-0.1447 0.0872	0.094 -1.6369	0.0201 0.0344
400	-0.0379 0.0952	0.3288 -7.14e-07	0.0193 0.0266
600	-0.0930 0.0854	0.2546 0.0005	0.0111 0.0204
1000	-0.0124 0.0729	0.3339 -1.4464	0.0298 0.0142

Tabla 5: Tabla de resultados LFR.

Si bien es cierto que en ocasiones el algoritmo no mejora los resultados obtenidos con Girvan-Newman, sólo lo consigue igualar o aproximarse, en general presenta mejores resultados. Esto sucede igual con las comunidades que ofrece el benchmark LFR, el cual devuelve resultados de menor modularidad. En cuanto a las redes limpias, esto no se da en situaciones reales. En esas situaciones, donde realmente las redes no son limpias, el algoritmo realizado en el proyecto puede encontrar comunidades más modulares que el resto de las propuestas. También se ha de tener en cuenta el factor de número del evaluaciones y generaciones realizados, los cuales no son equivalentes a los utilizados en otros algoritmos a comparar. Por lo tanto, se ha concluido que el funcionamiento es bueno. El algoritmo a lo largo de las evaluaciones mejora de forma correcta, para ello se muestran una serie de gráficas a continuación, cuatro con redes pequeñas y tres con redes mayores, pero menor número de generaciones.

En las siguientes figuras, por un lado se muestra cómo van mejorando las generaciones en cada una de las evaluaciones, representadas con un color diferente en base a la función de calidad. Por otro lado, se representa mediante diferentes boxplots los valores de modularidad que toman las diferentes evaluaciones en cada generación y se cruza el mismo con una curva que pasa por los valores medios de ellos.

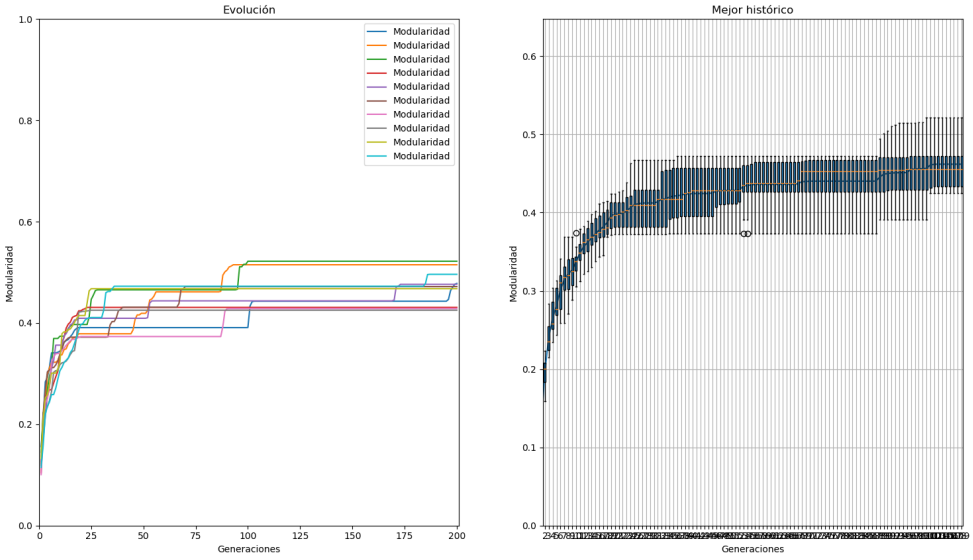


Figura 24: Gráficas evolución RC de 150 nodos y 15 comunidades.

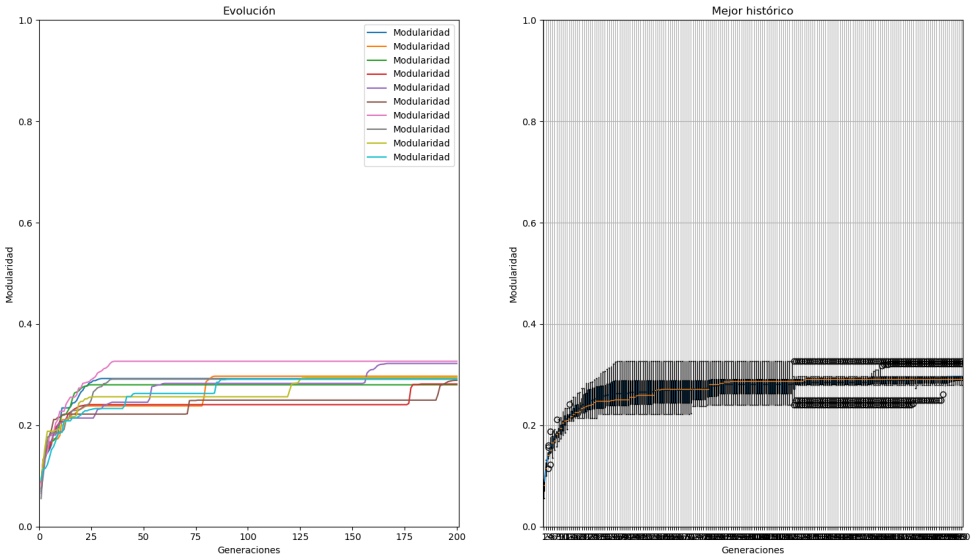


Figura 25: Gráficas evolución RC de 300 nodos y 15 comunidades.

A pesar de ser un análisis de 30 generaciones, en las siguiente figura, se puede observar que el funcionamiento del algoritmo es bueno y va mejorando generación a generación, lo cual es lo que se esperaba conseguir.

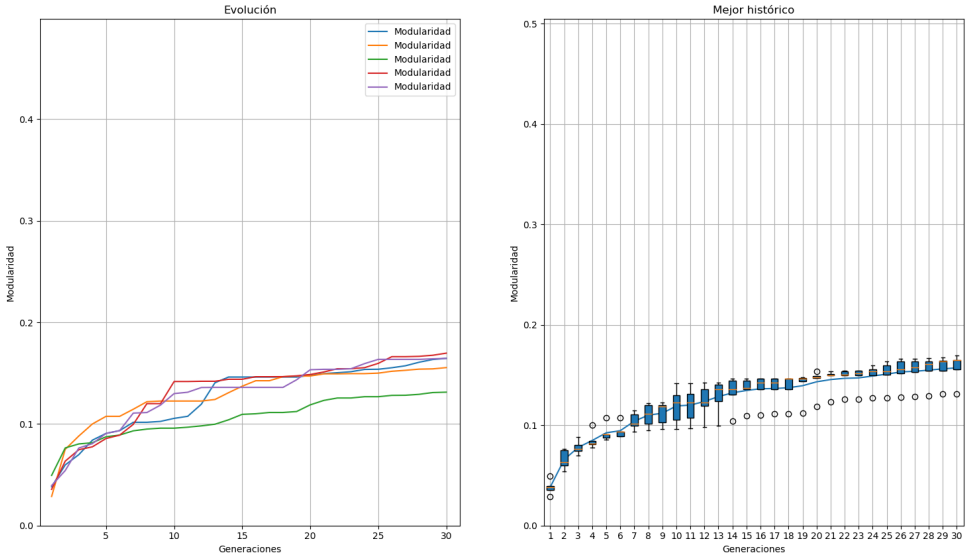


Figura 26: Gráficas evolución RC de 600 nodos y 15 comunidades.

Para las siguientes figuras, en redes LFR, se puede observar que el funcionamiento del algoritmo sigue el patrón de mejora gradual de las redes anteriores.

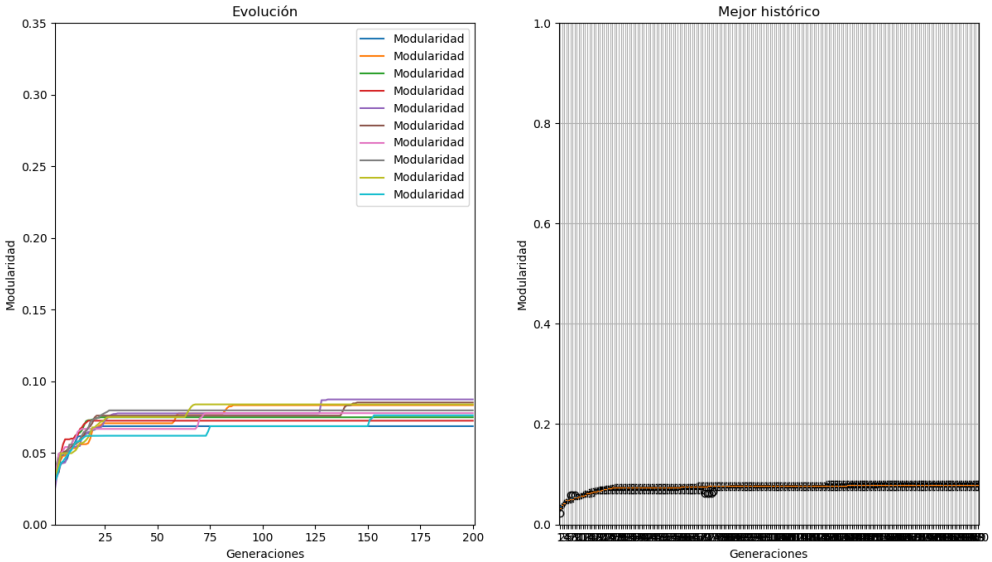


Figura 27: Gráficas evolución LFR de 200 nodos.

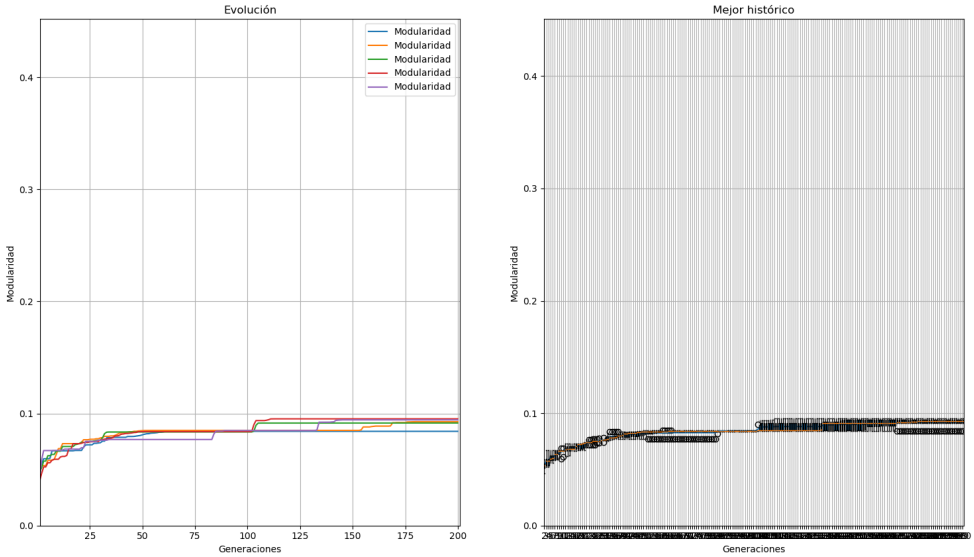


Figura 28: Gráficas evolución LFR de 400 nodos.

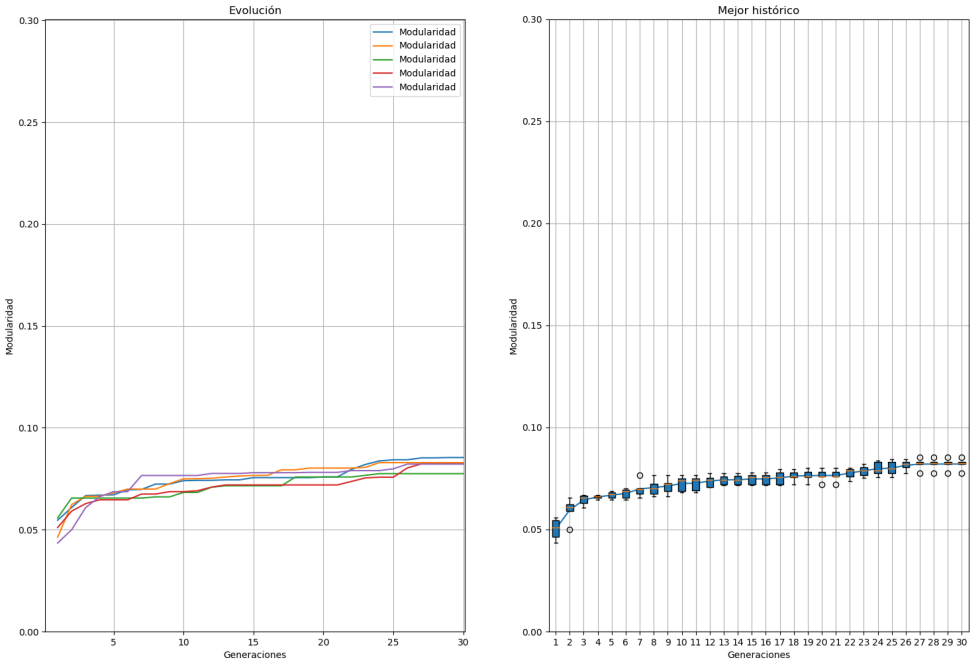


Figura 29: Gráficas evolución LFR de 600 nodos.

Por último, la red más grande tratada para esta ronda de pruebas. Aunque la mejora es más lenta que en los casos anteriores, el funcionamiento sigue siendo bueno. Esto se debe a que el número de nodos presentes en la red es mayor, entra en la categoría de alta dimensionalidad. Además, el número de generaciones es menor, por lo que el crecimiento está limitado.

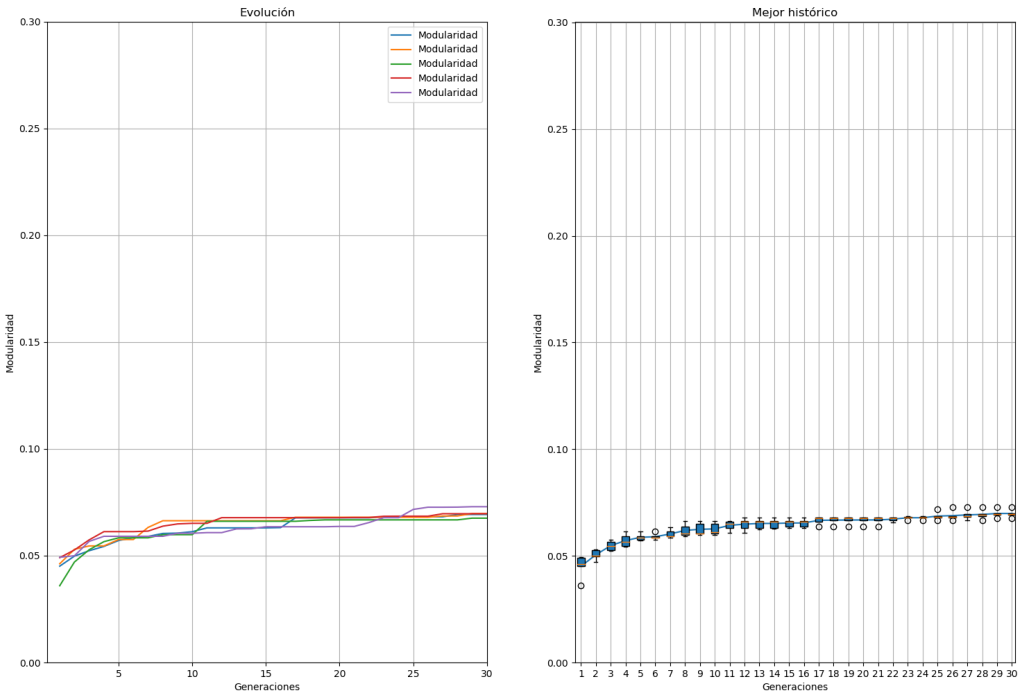


Figura 30: Gráficas evolución LFR de 1000 nodos.

10. Descripción de tareas

En este apartado se va a realizar un análisis de la planificación del proyecto. El trabajo se ha dividido en cuatro fases, las cuales están formadas por diferentes tareas.

- **Investigación** En esta fase se escoge al director de proyecto y tras una serie de reuniones, se asigna un tema para el trabajo de fin de grado. Se incluye la preparación del software necesario para el proyecto y la fase inicial de lectura acerca del tema a abordar y definición de objetivos a lograr.
- **Creación del algoritmo** En esta fase se aprende el lenguaje de trabajo, se trabaja con la minería de grafos y se desarrolla el algoritmo bioinspirado para el problema de optimización, tanto la búsqueda local como la búsqueda global.
- **Pruebas** En esta fase se realizan las pruebas de rendidimiento del algoritmo comparándolos con otros y realizando las mejoras pertinentes.
- **Documentación** En esta fase se redacta el documento final y se prepara una presentación para éste.

10.1. Diagrama de Gantt

Para organizar las fases y sus respectivas tareas en el tiempo la mejor forma de hacerlo es un diagrama de Gantt. El proyecto abarca el curso académico 2018-2019, pensado principalmente para el segundo cuatrimestre. Como días no laborables se establecerán los días festivos y los días de exámenes oficiales.

10.1.1. Investigación

EDT	Descripción	Fecha inicial	Fecha final	Duración
F1	Investigación	29/11/18	20/12/18	22 días
F1.1	Elección del TFG	29/11/18	03/12/18	5 días
F1.2	Preparación del software	04/12/18	10/12/18	7 días
F1.3	Búsqueda de información	11/12/18	13/12/18	3 días
F1.4	Definición de objetivos	18/12/18	20/12/18	3 días

Tabla 6: Tareas de la fase de investigación.

10.1.2. Creación del algoritmo

EDT	Descripción	Fecha inicial	Fecha final	Duración
F2	Creación del algoritmo	01/02/19	31/05/19	120 días
F2.1	Familiarización con Python	01/02/19	08/02/19	8 días
F2.2	Generación de la red	8/02/19	28/02/19	20 días
F2.3	Creación de búsqueda global	01/03/19	31/03/19	31 días
F2.4	Creación de búsqueda local	01/04/19	30/04/19	30 días
F2.5	Creación del algoritmo final	01/05/19	31/05/19	31 días

Tabla 7: Tareas de la fase de creación del algoritmo.

10.1.3. Pruebas

EDT	Descripción	Fecha inicial	Fecha final	Duración
F3	Pruebas	01/06/19	20/06/19	20 días
F3.1	Pruebas baja dimensionalidad	01/06/19	04/06/19	4 días
F3.2	Pruebas alta dimensionalidad	05/06/19	20/06/19	16 días

Tabla 8: Tareas de la fase de pruebas.

10.1.4. Documentación

EDT	Descripción	Fecha inicial	Fecha final	Duración
F3	Documentación	21/06/19	22/07/19	31 días
F3.1	Preparación de la memoria	21/06/19	23/06/19	3 días
F3.2	Redacción de la memoria	24/06/19	22/07/19	23 días

Tabla 9: Tareas de la fase de documentación.

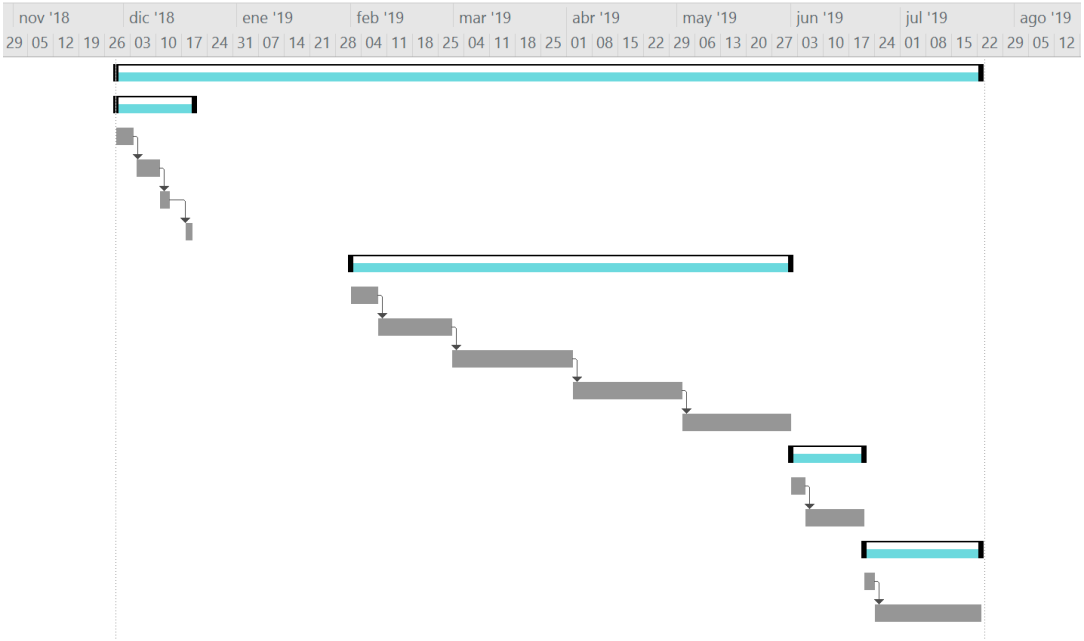


Figura 31: Diagrama de Gantt

11. Descripción del presupuesto

Se va a realizar un estudio ecónomico del proyecto mediante un descargo de datos. Como se trata de una investigación, el presupuesto a realizar es simple con una pequeña cantidad de conceptos a tener en cuenta. El análisis se basará en las horas internas, amortizaciones y gastos.

En primer lugar, se ha trabajado con un ordenador personal de un valor de 1600 €. A este equipo se le estima una 20000 horas de vida útil, es decir, casi 2 años y medio. El software disponible en este equipo para el proyecto es libre, por lo que no supone ningún gasto extra. En segundo lugar, se ha dispuesto de un servidor que permite realizar extras no asignadas al ordenador personal del ingeniero. Por último, se han adquirido una serie de materiales de oficina y documentación necesaria para el proyecto como libros, ...

11.1. Horas internas

Para que este proyecto sea viable serán necesarios un ingeniero técnico que desarrolle la solución y el documento, y un director que se encarge de la coordinación.

Concepto	Uso (h)	Coste horario (€/h)	Coste (€)
Ingeniero Técnico	300	25	7500.0
Director	50	35	1750.0
TOTAL			9250.0

Tabla 10: Horas internas

11.2. Amortizaciones

Como material amortizable sólo se necesitarán dos equipos: un ordenador Macbook Pro y un servidor. No serán necesarias licencias, ya que la programación se realizará en software libre: Anaconda para Python y TexStudio para este documento.

Concepto	Coste adquisición (€)	Vida útil (h)	Uso (h)	Coste (€)
Macbook Pro	1600.0	20000	300	24.0
TOTAL				24.0

Tabla 11: Amortizaciones

11.3. Gastos

Por último, los gastos no amortizables serán de material de oficina y de documentación. En este trabajo no se ha requerido de ningún software o equipo específico no reutilizable en otros proyectos.

Concepto	Coste (€)
Material oficina	40.0
Documentación	60.0
TOTAL	100.0

Tabla 12: Gastos

11.4. Gastos totales

Para obtener los gastos totales se realiza la suma de las horas internas, las amortizaciones y los gastos. Los gastos totales calculados para el proyecto son de 9374 €. Finalmente, se ha añadido una partida de imprevistos del 10 % para prevenir los riesgos comentados en su apartado corresponiente, ya que no se puede predecir con exactitud las futuras necesidades. El coste final tras incluir los imprevistos es de 10311.4 €.

Concepto	Coste (€)
Horas internas	9250.0
Amortizaciones	24.0
Gastos	100.0
Subtotal	9374.0
Imprevistos (10 %)	937.4
TOTAL	10311.4

Tabla 13: Gastos totales

12. Conclusiones

Este proyecto comenzó debido al interés generado por la influencia de las redes sociales a nivel mundial. Una vez realizado el proceso de documentación, investigación y programación, en este documento, se ha presentado un algoritmo optimizador capaz de particionar una red de alta dimensionalidad. Tras realizar un análisis de los resultados, se ha llegado a la conclusión de que el uso de este tipo de algoritmos tiene mucho futuro en la minería de grafos, ya que reducen de forma considerable el análisis de ésta, gracias a particiones lógicas basadas en las características concretas de la red. Además, el hecho de adaptarlo a redes de alta dimensionalidad supone un avance en la subrama de la heurística LSGO.

Entre los posibles escenarios futuros del algoritmo, el más destacado es la continuación del desarrollo del mismo. Actualmente, el algoritmo parte una deducción inicial pensada por el programador del número máximo de comunidades en la red. En un futuro, este valor se podría obtener mediante el propio algoritmo. Por otro lado, otra posible mejora sería adaptar el algoritmo de cara a cada evaluación, es decir, que el criterio de parada de las iteraciones se decida en función del progreso de los resultados obtenidos. Otro posible escenario futuro es la implementación del algoritmo en un proyecto personal recién iniciado consistente en una aplicación red social, lo cual permitirá adaptar las funcionalidades disponibles de cara a los usuarios. El último escenario futuro planteado es la extrapolación del funcionamiento de este algoritmo a otros campos mediante modificaciones, es decir, tomarlo como referencia para el análisis en otros campos.

Como conclusión final, tras el proceso de elaboración del proyecto, se ha comprendido que, a pesar de su complejidad, el uso de este tipo de algoritmos optimizadores tiene un gran futuro debido a la funcionalidad que posee en diferentes ámbitos de la vida, incluso los no tecnológicos.

Bibliografía

[1] Hootsuite, WeAreSocial. *DIGITAL 2019*.
https://wearesocial.com/uk/digital-2019

[2] Universidad de Granada. *Detección de comunidades*.
https://elvex.ugr.es/idbis/dm/slides/53%20Graph%20Mining%20-%20Community%20Detection.pdf

[3] Universidad de Granada. *Detección de comunidades en redes: Algoritmos y aplicaciones*.
http://digibug.ugr.es/bitstream/handle/10481/55384/TFM%20-%20Julio%20Omar%20Palacio.pdf?sequence=1

[4] Universidad de Granada. *Métodos Jerárquicos de Análisis Cluster.*.
https://www.ugr.es/~gallardo/pdf/cluster-3.pdf

[5] Universidad católica de Valparaíso, Faculta de Ingeniería. *Detección de comunidades en redes complejas o grafos usando metaheurísticas*.
http://opac.pucv.cl/pucv_txt/txt-5000/UCD5101_01.pdf

[6] Rodrigo Aldecoa, Universidad Politécnica de Valencia . *Detección de comunidades en redes complejas*.
https://riunet.upv.es/bitstream/handle/10251/15337/TFM_RodrigoAldecoa.pdf?sequence=1

[7] El-Ghazali Talbi, University of Lille – CNRS – INRIA. *Metaheuristics form design to implementation*.
https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwjI-cPKvJ3jAhWJsBQKHX_ODdoQFjAAegQIBBAC&url=https%3A%2F%2Fwww.researchgate.net%2Ffile.PostFileLoader.html%3Fid%3D5703bd8e4048546d0943c349%26assetKey%3DAS%253A347504454455296%2540459862926298&usg=AOvVaw3QAlDYxSql8OzETSFHDU9H

[8] Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria. *Algoritmos Bioinspirados.*.
http://paginaspersonales.deusto.es/cruz.borges/Papers/10APIAXXI.pdf

[9] Universidad de Granada. *Algoritmos genéticos: Conceptos básicos*.
https://sci2s.ugr.es/sites/default/files/files/Teaching/GraduatesCourses/Bioinformatica/Tema%2006%20-%20AGs%20I.pdf

[10] Ignacio Riquelme Medina. *Revisión de los Algoritmos Bioinspirados*.
https://www.researchgate.net/publication/303803004_Revision_de_los_Algoritmos_Bioinspirados

[11] Donglei Du, University of New Brunswick. *Social network analysis: community detection*.
http://www2.unb.ca/~ddu/6634/Lecture_notes/Lect10_community_R.pdf

[12] Andrea Lancichinetti, Santo Fortunato, Filippo Radicchi. *Benchmark graphs for testing community detection algorithms*.
https://pdfs.semanticscholar.org/2af4/a96f88ec630c57a28461751af3659ec98dd4.pdf?_ga=2.152116094.693891952.1562511421-627850451.1555349976

[13] Santofortunato. *Benchmark graphs to test community detection algorithms*.
https://sites.google.com/site/santofortunato/inthepress2

[14] Fernando Sancho. *Metaheurísticas para Búsqueda y Optimización*.
http://www.cs.us.es/~fsancho/?e=208

[15] Northeastern University. *Normalized Mutual Information, Estimating Clustering Quality*.
https://course.ccs.neu.edu/cs6140sp15/7_locality_cluster/Assignment-6/NMI.pdf

[16] Daniel Molina, Antonio LaTorre, Francisco Herrera. *SHADE with Iterative Local Search for Large-Scale Global Optimization*.
https://sci2s.ugr.es/sites/default/files/ficherosPublicaciones/2478_WCCI2018_shadeils.pdf

[17] Daniel Molina, Antonio LaTorre. *WCCI'2018 Large-Scale Global Optimization Competition Results*.
http://www.tflsgo.org/download/comp2018_slides.pdf

[18] Swagatam Das, Sankha Subhra Mullick, P. N. Suganthan. *Recent Advances in Differential Evolution – An Updated Survey*.
https://www.researchgate.net/publication/292616241_Recent_Advances_in_Differential_Evolution_-_An_Updated_Survey

[19] Minhazul Islam Sk, Swagatam Das, Subhrajit Roy. *Adaptive Differential Evolution with p-Best Crossover for Continuous Global Optimization*.
https://www.researchgate.net/publication/226384925_Adaptive_Differential_Evolution_with_p-Best_Crossover_for_Continuous_Global_Optimization

[20] Minhazul Islam Sk, Swagatam Das, Subhrajit Roy. *Adaptive Differential Evolution with p-Best Crossover for Continuous Global Optimization*.
https://www.researchgate.net/publication/226384925_Adaptive_Differential_Evolution_with_p-Best_Crossover_for_Continuous_Global_Optimization

[21] *Networkx*.
https://networkx.github.io/documentation/stable/index.html