UNIVERSIDAD DE GRANADA

ETSIIT INFORMÁTICA Y TELECOMUNICACIÓN

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Metaheurísticas

Guión de Prácticas

Práctica 1.a
Técnicas de Búsqueda de Poblaciones para el Problema de Maximizar Influencia En Redes Sociales (SNIMP)

Curso 2024-25

Tercer en Ingeniería Informática

1 OBJETIVOS

El objetivo de esta práctica es estudiar el funcionamiento de las *Técnicas de Búsqueda Local y de los Algoritmos Greedy* en la resolución del Problema de Maximizar Influencia En Redes Sociales (SNIMP) descrito en las transparencias del Seminario 2. Para ello, se requerirá que el estudiante adapte los siguientes algoritmos a dicho problema:

- Algoritmo Greedy básico.
- Algoritmo Aleatorio.
- Algoritmos de Búsqueda Local (BL).

El estudiante deberá comparar los resultados obtenidos con las estimaciones existentes para el valor de los óptimos de una serie de casos del problema.

La práctica se evalúa sobre un total de **2 puntos**, distribuidos de la siguiente forma:

- BL (**1 puntos**).
- Random **(0.5 puntos).**
- Greedy (**0.5 puntos**).

La fecha límite de entrega será el **el martes 1 de abril de 2025** antes de las 23:55 horas. La entrega de la práctica se realizará por internet a través del espacio de la asignatura en PRADO.

2 TRABAJO A REALIZAR

El estudiante deberá de desarrollar los distintos algoritmos al problema planteado. Los métodos desarrollados serán ejecutados sobre una serie de casos del problema. Se realizará un estudio comparativo de los resultados obtenidos y se analizará el comportamiento de cada algoritmo en base a dichos resultados. **Este análisis influirá decisivamente en la calificación final de la práctica**.

En las secciones siguientes se describen los aspectos relacionados con cada algoritmo a desarrollar y las tablas de resultados a obtener.

3 PROBLEMA Y CASOS CONSIDERADOS

3.1 Introducción al Problema de Problema de Maximizar Influencia En Redes Sociales (SNIMP)

El problema de la (en inglés, *social network influece maximization problem*, SNIMP) es un problema de optimización combinatoria consistente en seleccionar un subconjunto M de m elementos ($|M|=m$) de un conjunto inicial N de n elementos (con $n>m$) de forma que se maximice el conjunto de nodos influídos por ellos. Por eso también se denomina el problema difusión a través de la red *spread information*. El SNIMP se puede formular como:

$S^* \leftarrow_{s \in S} maxICM(G, S, p, ev)$

donde:

- G es el grafo del problema.
- S es el conjunto de todas las posibles soluciones.
- p es la probabilidad de que un usuario sea influído.
- ev es el número de iteraciones usadas en la simulación (usando una simulación de Montecarlo) para ejecutar el ICM, y, por tanto, evaluar las soluciones.

3.2 Representación de una solución

Para resolver los problemas en este guión de prácticas, la solución se representará como un vector o secuencia de números enteros (posiciones seleccionadas como *influencers*), en el que ningún número puede repetirse (es decir, cualquier secuencia de m valores donde $|s| = m$ y $s_i \in [0, n-1]$ con $s_i \neq s_j, \forall i \neq j, i \in [0, m-1], j \in [0, m-1]$). La longitud de la solución será de 10 ($m = 10$).

3.3 Evaluación de una solución

Para evaluar una solución es necesario realizar lo que se denomina un modelo en cascada. Usaremos el modelo más simple, el modelo independiente (en inglés, *Independent Cascade Model*, ICM).

ICM es el más simple, al considerar que la influencia entre cada par de nodos conectados es la misma, con una probabilidad, p. Al ser un modelo probabilístico, lo que se hace para evaluarlo es aplicar simular una transmisión de los nodos elegidos como influyentes múltiples veces, y hacer el promedio de nodos de la red influidos. En esa simulación, por cada conexión saliente de un nodo influído se genera un número aleatorio r, y si es menor que p, se considera al nuevo nodo conectado *influido*. Posteriormente, para cada nodo nuevo influído el proceso se repite con todas sus conexiones de salida. Este proceso se repite hasta que ya no haya nuevas *influencias*.

Visualmente se puede mostrar el seudocódigo:

Algorithm 1 $ICM(G = (V, E), S, p, ev)$

```
1: I ← ∅
2: for i ∈ 1 ... ev do
3:     A* ← S
4:     A ← S
5:     while A ≠ ∅ do
6:         B ← ∅
7:         for v ∈ A do
8:             for (u, v) ∈ E do
9:                 if rnd(0, 1) ≤ p then
10:                    B ← B ∪ {u}
11:                end if
12:            end for
13:        end for
14:        A* ← A* ∪ B
15:        A ← B
16:    end while
17:    I ← I + |A*|
18: end for
19: return I/ev
```

en donde:

- A es el conjunto de nodos infectados sobre el que se inician los *contagios/influencias*.
- A^* es el total de nodos infectados.
- B es el conjunto de nuevos nodos infectados.

Por tanto, el proceso es el siguiente: **Partiendo de los nodos indicados en la solución, se inicia el algoritmo ICM usando *p=0.01* y *env=10*. Dado que es un proceso no determinístico, en la implementación se introducirán en A los nodos de forma ordenada, y se generarán los números aleatorios de la siguiente manera: Al principio se consultará el estado actual (semilla), luego se le asignará un valor concreto, como 35, y al terminar de ejecutar el ICM se volverá a poner el estado de la secuencia seudoaleatoria al valor inicial leído**.

3.4 Casos Considerados

Se utilizarán **4 casos reales** seleccionados de varios de los conjuntos de instancias del *Stanford Large Network Dataset Collection* disponibles en https://snap.stanford.edu/data/index.html.

En particular, se han usado un caso de autores en una revista online (*General Relativity and Quantum Cosmology*), y conexiones en una red social muy poco usada (Gnutella) en tres días distintos (5, 8, y 25 de Agosto de 2002). La tabla 1 muestra sus características.

Tabla 1: Ficheros de redes usadas

Nombre	|Nodo|	|Enlaces|	web
ca-GrQc	5242	14496	https://snap.stanford.edu/data/ca-GrQc.html
p2p-Gnutella05	8846	31839	https://snap.stanford.edu/data/p2p-Gnutella05.html
p2p-Gnutella08	6301	20777	https://snap.stanford.edu/data/p2p-Gnutella08.html
p2p-Gnutella25	22687	54705	https://snap.stanford.edu/data/p2p-Gnutella25.html

Se han elegido estos ficheros para mantener contenido el tiempo de evaluación, ya que el propio proceso de evaluación de la solución puede incrementarse mucho con el tamaño. Aún así el número total de evaluaciones se ha mantenido reducido que en otras prácticas (a 1000).

El formato de los ficheros es el siguiente:

- Un par de líneas con comentarios iniciadas con #.
- Una línea indicando el número de nodos y conexiones: Ejemplo: *# Nodes: 5242 Edges: 28980*
- La línea con el formato, todas en este caso siguen el mismo: *# FromNodeId ToNodeId* por lo que se puede ignorar.
- Líneas con los datos de conexión, indica primero el nodo de salida (numerado desde 0), un tabulador, y luego el nodo de entrada.

Ejemplo (*ca-GrQc.txt*)

```
# Directed graph (each unordered pair of nodes is saved once): CA-GrQc.txt
# Collaboration network of Arxiv General Relativity category (there is an edge
# Nodes: 5242 Edges: 28980
# FromNodeId    ToNodeId
0       1
0       2
0       3
0       4
0       5
0       6
0       7
0       8
4       9
4       0
4       10
...
4410    370
4974    4973
4974    4976
1189    1061
1189    1176
1189    1177
```

4 ALGORITMOS

4.1 Algoritmo Aleatorio

El algoritmo *aleatorio* se basa en generar **1000** soluciones de forma totalmente aleatoria, y devolver la mejor solución encontrada.

4.2 Algoritmo *Greedy*

El algoritmo *greedy* del SNIMP se basa en la heurística de ir seleccionando los nodos que presenten mayores conexiones con el resto, independientemente del resto de nodos ya considerados.

Hay múltiples criterios para identificar el número de conectividad de un nodo, usaremos uno muy sencillo pero que es el que presenta un mejor comportamiento. Conceptualmente es muy sencillo, consiste en sumar el número de conexiones, y el número de conexiones de cada solución vecina.

Formalmente se puede definir la función heurística de un nodo u ($g(u)$) como:

$$g_{ne}(u) = d_u^+ + \sum_{v \in N_u^+} d_u^+$$

en donde $d^+(u) = |N_u^+|$ y $N_u^+ = \{w \in V : (u, w) \in E\}$ siendo E el conjunto de conexiones.

Se realizará una única ejecución sobre cada caso del problema.

4.3 Búsqueda Local

Como algoritmo de BL para el SNIMP consideraremos el esquema del **primero mejor**, tal y como está descrito en las transparencias del Seminario 2. La representación será en forma de un conjunto de elementos seleccionados. Se empleará el movimiento de intercambio *Int*(*Sel*,*i*,*j*) que intercambia el elemento seleccionado i por uno no seleccionado j en la solución actual *Sel*. Una vez realizado el movimiento, se actualiza la solución actual y los valores de contribución de los elementos seleccionados al coste de dicha solución, y se comienza a explorar el nuevo entorno. Se hará el recorrido del entorno de forma totalmente aleatoria. Se aplicarán dos criterios de parada:

- Hasta llegar a 1000 evaluaciones (*LSall*).
- Hasta llegar a 1000 evaluaciones, o sin mejoras en 20 evaluaciones (*BLsmall*).

En cada ejecución de la BL, se partirá de una solución inicial aleatoria y se detendrá la ejecución **cuando no se encuentre mejora en todo el entorno o cuando se cumpla la condición indicada arriba**, es decir, en cuanto se cumpla alguna de las dos condiciones. **Se realizarán cinco ejecuciones sobre cada caso del problema**.

5 DETERMINACIÓN DE LA CALIDAD DE UN ALGORITMO

El modo habitual de determinar la calidad de un algoritmo de resolución aproximada de problemas de optimización es ejecutarlo sobre un conjunto determinado de instancias y comparar los resultados obtenidos con los mejores valores conocidos para dichas instancias, si fuesen conocidos.

Además, los algoritmos pueden tener diversos parámetros o pueden emplear diversas estrategias. Para determinar qué valor es el más adecuado para un parámetro o saber la estrategia más efectiva los algoritmos también se comparan entre sí.

La comparación de los algoritmos se lleva a cabo fundamentalmente usando dos criterios, la *calidad* de las soluciones obtenidas y el *tiempo de ejecución* empleado para conseguirlas. Además, es posible que los algoritmos no se comporten de la misma forma si se ejecutan sobre un conjunto de instancias u otro.

Por otro lado, a diferencia de los algoritmos determinísticos, los algoritmos probabilísticos se caracterizan por la toma de decisiones aleatorias a lo largo de su ejecución. Este hecho implica que un mismo algoritmo probabilístico aplicado al mismo caso de un problema pueda comportarse de forma diferente y por tanto proporcionar resultados distintos en cada ejecución.

Cuando se analiza el comportamiento de un algoritmo probabilístico en un caso de un problema, se desearía que el resultado obtenido no estuviera sesgado por una secuencia aleatoria concreta que pueda influir positiva o negativamente en las decisiones tomadas durante su ejecución. Por tanto, resulta necesario efectuar varias ejecuciones con distintas secuencias probabilísticas y calcular el resultado medio (y a veces la desviación típica) de todas las ejecuciones para representar con mayor fidelidad su comportamiento.

Dada la influencia de la aleatoriedad en el proceso, es recomendable disponer de un generador de secuencia pseudoaleatoria de buena calidad con el que, dado un valor semilla de inicialización, se obtengan números en una secuencia lo suficientemente grande (es decir, que no se repitan los números en un margen razonable) como para considerarse aleatoria. En el espacio de PRADO se

puede encontrar una implementación en lenguaje C++ de un generador aleatorio de buena calidad (*random.hpp*).

Como norma general, el proceso a seguir consiste en realizar un número de ejecuciones diferentes de cada algoritmo probabilístico considerado para cada caso del problema. Es necesario asegurarse de que se realizan diferentes secuencias aleatorias en dichas ejecuciones. Así, el valor de la semilla que determina la inicialización de cada secuencia deberá ser distinto en cada ejecución y estas semillas deben mantenerse en los distintos algoritmos (es decir, la semilla para la primera ejecución de todos los algoritmos debe ser la misma, la de la segunda también debe ser la misma y distinta de la anterior, etc.). Por simplificar y facilitar la reproducibilidad, se usará la misma semilla para todos los casos de uso. Para mostrar los resultados obtenidos con cada algoritmo en el que se hayan realizado varias ejecuciones, se suelen construir tablas que recojan los valores correspondientes a estadísticos como el **mejor** y **peor** resultado para cada caso del problema, así como la **media** y la **desviación típica** de todas las ejecuciones. También se pueden emplear descripciones más representativas como los boxplots, que proporcionan información de todas las ejecuciones realizadas mostrando mínimo, máximo, mediana y primer y tercer cuartil de forma gráfica. Finalmente, se construirán unas tablas globales con los resultados agregados que mostrarán la calidad del algoritmo en la resolución del problema desde un punto de vista general.

Para cada algoritmo no determinístico ejecutado, se ejecutará 5 veces para cada instancia, cada uno con un valor de semilla distinto, y se **indicará la media** tanto en tiempos como en coste final obtenido. S**erá necesario inicializar las semillas del generador aleatorio para poder repetir el experimento** y obtener los mismos resultados si fuera necesario (en caso contrario, los resultados podrían variar en cada ejecución del mismo algoritmo sobre el mismo caso del problema). Dado que es un proceso no determinístico, en la implementación se introducirán en A los nodos de forma ordenada, y se generarán los números aleatorios de la siguiente manera: Al principio se consultará el estado actual (semilla), luego se le asignará un valor concreto, 35, y al terminar de ejecutar el ICM se volverá a poner el estado de la secuencia seudoaleatoria al valor inicial leído.

6 TABLAS DE RESULTADOS A OBTENER

Como algoritmo de BL para el SNIMP consideraremos el esquema del primer mejor, tal y como está descrito en las transparencias del Seminario 2. La representación será en forma de un conjunto de elementos seleccionados. Se diseñará una tabla para cada algoritmo (*Greedy*, *LSall*, *LSsmall*) donde se recojan los resultados de la ejecución de dicho algoritmo al conjunto de casos del problema. Tendrá la misma estructura que la tabla 2. Cada valor será el promedio de las 5 ejecuciones. Se mostrarán los resultados de tiempo con dos decimales. Evaluaciones es el número de evaluaciones, será 1 en greedy, presumiblemente 1000 en *LSall* y un valor menor en *LSsmall*.

Tabla 2: Formato de resultados para el Algoritmo X

Conjunto	Fitness	Tiempo (segs)	Evaluaciones
ca-GrQC	x	x	x
p2p-Gnutella05	x	x	x
p2p-Gnutella08	x	x	x
p2p-Gnutella25	x	x	x

Finalmente, se construirá una tabla de resultados con conjunto de datos que recoja los resultados medios de fitnes, tiempo y evaluaciones para todos los algoritmos considerados, tal como se muestra en la tabla 3. Se incluirá su posición según el fitness (1 para el mejor, 2 para el segundo, ...).

Tabla 3: Formato de resultados para el conjunto XXX

Algoritmo	Posición	Fitness	Tiempo (segs)	Evaluaciones
Random	x	x	x	1000
Greedy	x	x	x	1
LSall	x	x	x	x
LSsmall	x	x	x	x

Finalmente, se usará la tabla 4, indicando por cada algoritmo la posición promedio, el tiempo promedio y el promedio de evaluaciones.

Tabla 4: Tabla final de resultados

Algoritmo	Posición Promedia	Tiempo Promedio (segs)	Total evaluaciones
Random	x	x	1000
Greedy	x	x	1
LSall	x	x	x
LSsmall	x	x	x

A partir de los datos mostrados en estas tablas, el estudiante realizará un análisis de los resultados obtenidos, **que influirá significativamente en la calificación de la práctica**. En dicho análisis se deben comparar los distintos algoritmos en términos de calidad de las soluciones y tiempo requerido para producirlas. Por otro lado, se puede analizar también el comportamiento de los algoritmos en algunos de los casos individuales que presenten un comportamiento más destacado.

7 DOCUMENTACIÓN Y FICHEROS A ENTREGAR

En general, la **documentación** de ésta y de cualquier otra práctica será un fichero pdf que deberá incluir, al menos, el siguiente contenido:

a) Portada con el número y título de la práctica (con el nombre del problema), el curso académico, el nombre, DNI y dirección e-mail del estudiante, y su horario de prácticas.

b) Índice del contenido de la documentación con la numeración de las páginas.

c) **Breve** descripción/formulación del problema (**máximo 1 página**). Podrá incluirse el mismo contenido repetido en todas las prácticas presentadas por el estudiante.

d) Breve descripción de la aplicación de los algoritmos empleados al problema (**máximo 4 páginas**): Todas las consideraciones comunes a los distintos algoritmos se describirán en este apartado, que será previo a la descripción de los algoritmos específicos. Incluirá por ejemplo la descripción del esquema de representación de soluciones y la **descripción en pseudocódigo (no código)** de la función objetivo y los operadores comunes.

e) Descripción en **pseudocódigo** de la **estructura del método de búsqueda** y de todas aquellas **operaciones relevantes** de cada algoritmo. Este contenido, específico a cada algoritmo se detallará en los correspondientes guiones de prácticas. El pseudocódigo **deberá forzosamente reflejar la implementación/ el desarrollo realizados** y no ser una descripción genérica extraída de las transparencias de clase o de cualquier otra fuente. La descripción de cada algoritmo no deberá ocupar más de **2 páginas**.

   Para esta primera práctica, se incluirán al menos las descripciones en pseudocódigo de:

   - Para el algoritmo *greedy*, aparte del esquema general, la función heurística.
   - Para el algoritmo BL, el métodos de exploración del entorno, el operador de generación de vecino, la factorización de la BL si fuese el caso, y la generación de soluciones aleatorias.

f) Breve explicación de la **estructura del código de la práctica**, incluyendo un pequeño **manual de usuario describiendo el proceso para que el profesor de prácticas pueda compilarlo (usando un sistema automático como make o similar) y cómo ejecutarlo, dando algún ejemplo de ejecución**.

g) Experimentos y análisis de resultados:

   - Descripción de los casos del problema empleados y de los valores de los parámetros considerados en las ejecuciones de cada algoritmo (**incluyendo las semillas utilizadas**).
   - Resultados obtenidos según el formato especificado.
   - Análisis de resultados. El análisis deberá estar orientado a **justificar** (según el comportamiento de cada algoritmo) **los resultados** obtenidos en lugar de realizar una mera

"lectura" de las tablas. Se valorará la inclusión de otros elementos de comparación tales como gráficas de convergencia, boxplots, análisis comparativo de las soluciones obtenidas, representación gráfica de las soluciones, etc.

h) Referencias bibliográficas u otro tipo de material distinto del proporcionado en la asignatura que se haya consultado para realizar la práctica (en caso de haberlo hecho).

Aunque lo esencial es el contenido, también debe cuidarse la presentación y la redacción. **La documentación nunca deberá incluir listado total o parcial del código fuente**.

En lo referente al **desarrollo de la práctica**, se entregará una carpeta llamada **software** que contenga una versión ejecutable de los programas desarrollados, así como el código fuente implementado o los ficheros de configuración del framework empleado. El código fuente o los ficheros de configuración se organizarán en la estructura de directorios que sea necesaria y deberán colgar del directorio *src* en el raíz. Junto con el código fuente, hay que incluir los ficheros necesarios para construir los ejecutables según el entorno de desarrollo empleado (tales como *.prj, makefile, *.ide, etc.). En este directorio se adjuntará también un pequeño fichero de texto de nombre LEEME que contendrá breves reseñas sobre cada fichero incluido en el directorio. Es importante que los programas realizados puedan leer los valores de los parámetros de los algoritmos desde fichero, es decir, que no tengan que ser recompilados para cambiar éstos ante una nueva ejecución. Por ejemplo, la semilla que inicializa la secuencia pseudoaleatoria debería poder especificarse como un parámetro más.

En el caso de que en el lenguaje de programación elegido se haya ofrecido un API, deberá de **obligatoriamente implementarlo siguiendo el API ofrecido**. Si no fuese el caso, se adaptará el API recomendado.

El fichero pdf de la documentación y la carpeta software serán comprimidos en un fichero .zip etiquetado con los apellidos y nombre del estudiante (Ej. Pérez Pérez Manuel.zip). Este fichero será entregado por internet a través del espacio de la asignatura en PRADO.

8 MÉTODO DE EVALUACIÓN

Al principio de la práctica se ha indicado la puntuación máxima que se puede obtener por cada algoritmo y su análisis. **La inclusión de trabajo voluntario** (desarrollo de variantes adicionales, experimentación con diferentes parámetros, prueba con otros operadores o versiones adicionales del algoritmo, análisis extendido, etc.) **podrá incrementar la nota final** por encima de la puntuación máxima definida inicialmente, o compensar parcialmente errores en la práctica.

En caso de que el comportamiento del algoritmo en la versión implementada/ desarrollada no coincida con la descripción en pseudocódigo o no incorpore las componentes requeridas, se podría reducir hasta en un 50 % la calificación del algoritmo correspondiente.