UNIVERSIDAD DE JAÉN
ESCUELA POLITÉCNICA SUPERIOR DE JAÉN

Departamento de Informática. Área de Ciencias de la Computación e Inteligencia Artificial

Metaheurísticas

Guión de Prácticas

Práctica 1:
Metaheurísticas basadas en trayectorias

Curso 2017-2018

Tercer Curso del Grado en Ingeniería Informática

Práctica 1

Metaheurísticas basadas trayectorias para el Problema de Asignación de Frecuencias con Mínimas Interferencias

1. Objetivos

El objetivo de esta práctica es estudiar el funcionamiento de los siguientes algoritmos: Greedy para MI-FAP, Búsqueda Local (BL), Búsqueda Tabú (BT) y GRASP. Para ello, se requerirá que el alumno adapte estos métodos para resolver el problema de Asignación de Frecuencias con Mínimas Interferencias (MI-FAP) descrito en las transparencias del Seminario 2 y que compare los resultados obtenidos en un conjunto de instancias del problema.

La práctica se evalúa sobre un total de **4,5 puntos**, distribuidos de la siguiente forma: Algoritmo Greedy (0,5 punto), BL (1 punto), BT (1,5 punto) y GRASP (1,5 punto) incluyendo análisis de resultados de los algoritmos.

La fecha de explicación es el día 26/09/2017 o 29/09/2017. La fecha límite de entrega será el **23 de Octubre de 2017** antes de las 23:59 horas. La entrega de la práctica se realizará por internet a través de la plataforma ilias. La defensa de la práctica será el día 24/10/2017 o 27/10/2017 durante el horario de práctica.

2. Trabajo a Realizar

El alumno podrá desarrollar los algoritmos de la práctica siguiendo la modalidad que desee: trabajando con cualquiera de los frameworks de metaheurísticas estudiados en el Seminario 1, implementándolos a partir del código C proporcionado en la plataforma ilias de la asignatura o considerando cualquier código disponible en Internet.

Los métodos desarrollados serán ejecutados sobre un conjunto de instancias. 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 el problema y los casos considerados, 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 Asignación de Frecuencias con Mínimas Interferencias (MI-FAP)

El problema del MI-FAP consiste en asignar frecuencias a una serie de trasmisores (cada trasmisor poseen un rango de frecuencias disponibles en las que transmitir). Sea T ={t_1, t_2, … t_n } un conjunto de n trasmisores (TRX) y sea F_i = {f_{i1}, f_{i2},… f_{ik}}} ⊂ N un conjunto de frecuencias válidas que pueden ser asignadas a los transmisores t i ∈T, i=1.. , n. Nótese que el cardinal de k de F_i no es necesariamente el mismo para todos los transmisores. Se define una matriz de interferencias entre las frecuencias MI (incluida esa información en el fichero ctr.txt de cada instancia)

La función objetivo consiste en minimizar la suma de las interferencias existentes en los trasmisores instalados.

$$fitness = \sum_{i=0}^{t} \sum_{j=i+1}^{t} MI(TRXi, TRXj)$$

3.2. Conjuntos de Datos Considerados

El alumno trabajará con los **10 conjuntos de datos** siguientes (obtenidos de la web http://fap.zib.de/problems/CALMA/ y disponibles en la plataforma ilias de la asignatura):

1. CELAR 06 con 200 trasmisores
2. CELAR 07 con 400 trasmisores
3. CELAR 08 con 916 trasmisores
4. CELAR 09 con 680 trasmisores
5. CELAR 10 con 680 trasmisores
6. GRAPH 05 con 200 trasmisores
7. GRAPH 06 con 400 trasmisores
8. GRAPH 07 con 400 trasmisores
9. GRAPH 11 con 680 trasmisores
10. GRAPH 12 con 680 trasmisores

El formato de los ficheros incluidos en cada carpeta es:

- var.txt: la lista de los enlaces de comunicación (trasmisores)
- dom.txt: la lista de dominios de frecuencia
- ctr.txt: la lista de todas las restricciones

3.3. 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.

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 una metaheurística probabilística 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 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 la plataforma ilias de la asignatura se puede encontrar una implementación en lenguaje C de un generador aleatorio de buena calidad (*random.h*).

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.). Para mostrar los resultados obtenidos con cada algoritmo en el que se hayan realizado varias ejecuciones, se deben 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. Alternativamente, 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.

Este será el procedimiento que aplicaremos en las prácticas. Como norma general, el alumno realizará **5 ejecuciones diferentes** de cada algoritmo probabilístico considerado para cada caso del problema. Se considerará como **semilla inicial el número del DNI del alumno**. En la tabla 2 se incluye un ejemplo sobre los posibles valores de la semilla para el generador aleatorio.

Tabla 2: Ejemplo de posibles semillas a utilizar en el generador aleatorio

Ejecución	Semilla
1	12345678
2	23456781
3	34567812
4	45678123
5	56781234

La hoja Excel *Tablas_MI-FAP_2017-18.xls*, disponible en la plataforma ilias de la asignatura, permite generar de forma automática los estadísticos comentados para las tablas de resultados de la práctica.

4. Componentes de los Algoritmos

Los algoritmos de esta práctica tienen en común las siguientes componentes:

- *Esquema de representación*: Se seguirá la representación entera basada en un vector *X* de tamaño *n (trasmisores)*

- *Función objetivo*: Se persigue minimizar las interferencias.

- *Generación de la solución inicial*: La solución inicial se generará usando el algoritmo simple.

- *Esquema de generación de vecinos*: Se cambia el valor de frecuencia incluido en un trasmisor por otra de las frecuencias disponibles.

- *Criterio de aceptación*: Se considera una mejora cuando se reduce el valor global de la función objetivo.

- *Criterio de parada*: Se detendrá la ejecución del algoritmo bien cuando no se encuentre mejora en todo el entorno (BL) o bien cuando se hayan evaluado 10000 soluciones distintas. En los algoritmos BT y GRASP, el criterio de parada será alcanzar un número máximo de iteraciones (es decir, número de soluciones generadas y, por tanto, evaluadas mediante la función objetivo) de 10000 soluciones.

A continuación veremos las particularidades de cada algoritmo.

4.1. Búsqueda Local del primer mejor

Algoritmo

Como algoritmo de BL para el problema MI-FAP consideraremos el esquema del **primer mejor**, tal y como está descrito en las transparencias del ***Seminario 2***. Cada vez que un vecino mejora el coste de la solución actual, se aplicará y se actualizarán el coste. Se continuará la ejecución buscando los vecinos de esa nueva solución. Sera obligatorio emplear la **factorización** de la función objetivo.

4.2. Búsqueda Tabú

Algoritmo

Se trabajará con la versión del algoritmo BT estudiada en clase, la cual utiliza una lista de movimientos tabú:

- *Lista tabú*: Almacena (**TRX, FrecNueva**), donde **TRX** es el trasmisor cambiado y **FrecNueva** es la nueva frecuencia asignada a ese trasmisor. No se permiten estos movimientos durante la tenencia tabú.

- *Operador de Vecino y exploración del entorno para L(T)*: En cada iteración se generarán 20 soluciones vecinas (o menos si no hay tantos vecinos en la solución actual) aplicando el movimiento explicado en la BL. Se seleccionará el mejor vecino de acuerdo a los criterios tabú. Se recomienda emplear iterativamente la **factorización** de la función objetivo para calcular el coste de los vecinos una vez generados con objeto de mejorar la eficiencia.

- *Criterio de aspiración*: Tener mejor coste (menor) que la mejor solución obtenida hasta el momento.

- *Reinicialización:* En el caso de que la búsqueda se estanque (es decir, 2000 iteraciones sin mejora)

Valores de los parámetros y ejecuciones

El número máximo de iteraciones será 10000. El tamaño inicial de cada lista tabú será número de trasmisores dividido entre 6.

4.3. GRASP

Algoritmo

El algoritmo GRASP constará de dos componentes: construcción de soluciones *greedy* aleatorizada, utilizando una lista restringida de candidatos (LRC) durante el

proceso de construcción, y optimización de las mismas mediante el algoritmo de BL. El algoritmo *greedy* aleatorizado a considerar es el explicado en las transparencias del Seminario 2. Una vez generada cada solución *greedy* inicial se aplicará el algoritmo de BL sobre ella. Finalmente, se devolverá la mejor solución encontrada.

La búsqueda local es la BL del primer mejor se detendrá la búsqueda en el momento que se hayan superado las 400 iteraciones de la función objetivo (en esta fase de la BL) o antes si después de examinar todo el vecindario de una solución no se han encontrado soluciones mejores. Continuando una vez que termine la BL con una nueva construcción de soluciones greedy.

Valores de los parámetros y ejecuciones

El número máximo de iteraciones será 10000 (en cada ejecución del algoritmo se generarán al menos 25 soluciones iniciales aleatorias, es decir, 400*25 =10000). La selección del elemento TRX (que pasara a formar parte de la solución) de la LRC se realizara de forma aleatoria.

4.4. Algoritmo de Comparación: Greedy

El algoritmo escogido para ser comparado con las metaheurísticas implementadas es el algoritmo *Greedy*. Se trabajará con la versión del algoritmo estudiada en el seminario 2. Como el resto de algoritmos, el Greedy deberá ser ejecutado 5 veces con inicializaciones aleatorias distintas, resultantes de la especificación de una semilla diferente para el generador de números aleatorios.

5. Tablas de Resultados a Obtener

Se diseñará una tabla para cada algoritmo (BL, BT, GRASP y Greedy) donde se recojan los resultados de la ejecución de dicho algoritmo en los conjuntos de datos considerados. Tendrá la misma estructura que la Tabla 5.1.

Tabla 5.1: Resultados obtenidos por el algoritmo X en el problema del MI-FAP (solo con las 5 primeras instancias, en la práctica incluir 10 instancias)

	CELAR 06		**CELAR 07**		**CELAR 08**		**CELAR 09**		**CELAR 10**	
	Z	**Tiempo**	***Z***	**Tiempo**	***Z***	**Tiempo**	***Z***	**Tiempo**	***Z***	**Tiempo**
Ejec 1	x	x	x	x	x	x	x	x	x	x
Ejec 2	x	x	x	x	x	x	x	x	x	x
Ejec 3	x	x	x	x	x	x	x	x	x	x
Ejec 4	x	x	x	x	x	x	x	x	x	x
Ejec 5	x	x	x	x	x	x	x	x	x	x
Mejor	x	---	x	---	x	---	x	---	x	---
Peor	x	---	x	---	x	---	x	---	x	---
Media	x	x	x	x	x	x	x	x	x	x
Desv. típica	x	x	x	x	x	x	x	x	x	x

Finalmente, se construirá una tabla de resultados global que recoja los resultados medios de calidad y tiempo para todos los algoritmos considerados, tal como se muestra en la tabla 5.2. Para rellenar esta tabla se hará uso de los resultados mostrados en las tablas parciales, siendo m la media y σ la desviación típica. Aunque en la tabla que sirve de ejemplo se han incluido todos los algoritmos considerados en esta práctica, naturalmente sólo se incluirán los que se hayan desarrollado.

Tabla 5.2: Resultados globales en el problema del MI-FAP (ejemplo 5 instancias)

	CELAR 06		**CELAR 07**		**CELAR 08**		**CELAR 09**		**CELAR 10**	
	Z	**Tiempo**	***Z***	**Tiempo**	***Z***	**Tiempo**	***Z***	**Tiempo**	***Z***	**Tiempo**
BL	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)
BT	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)
GRASP	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)
Greedy	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)	m (σ)

A partir de los datos mostrados en estas tablas, el alumno 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 del mejor resultado individual obtenido (capacidad del algoritmo para obtener soluciones de calidad), los resultados medios (robustez del algoritmo) y el tiempo requerido para obtener las soluciones (rapidez del algoritmo). Se comparará el rendimiento de las metaheurísticas entre sí, así como con respecto al algoritmo de referencia, Greedy, tanto a nivel de los casos individuales como desde una perspectiva global.

6. Documentación y Ficheros a Entregar

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

1. Portada con el número y título de la práctica, el curso académico, los algoritmos considerados; el nombre, DNI y dirección e-mail del alumno, y su grupo y horario de prácticas.
2. Índice del contenido de la documentación con la numeración de las páginas.
3. **Breve** descripción/formulación del problema (**máximo 1 página**). Podrá incluirse el mismo contenido repetido en todas las prácticas presentadas por el alumno.
4. Breve descripción de la aplicación de los algoritmos empleados al problema (**máximo 2 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 de la función objetivo y los operadores comunes (en este caso, el de **generación de vecino** y su **factorización**), etc.
5. 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, además de la descripción del esquema de búsqueda, se deberá incluir al menos:

- Para el algoritmo BL, descripción en pseudocódigo del método de exploración del entorno.
- Para el algoritmo BT, descripción en pseudocódigo del manejo de la lista tabú.
- Para el algoritmo GRASP, descripción en pseudocódigo del mecanismo de generación de soluciones greedy probabilísticas, indicando el modo de obtención de la lista restringida de candidatos.

6. **Breve** descripción del algoritmo de comparación, el Greedy (**máximo 1 página**). Podrá incluirse el mismo contenido repetido en el resto de prácticas.
7. Breve explicación del **procedimiento considerado para desarrollar la práctica**: implementación a partir del código proporcionado en prácticas o a partir de cualquier otro, o uso de un framework de metaheurísticas concreto. Inclusión de un pequeño **manual de usuario describiendo el proceso para que el profesor de prácticas pueda replicarlo**.
8. 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.
   - También incluir análisis de tiempos y desviaciones, e influencia del tamaño de la instancia en los resultados obtenidos, etc..
9. 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 caso de haberlo implementado**.

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 los ficheros de datos de los casos del problema y 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 FUENTES 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.). La versión ejecutable de los programas y los ficheros de datos se incluirán en un subdirectorio del raíz de nombre BIN. En este mismo directorio se adjuntará 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.

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

7. Método de Evaluación

Tanto en esta práctica como en las siguientes, se indicará la puntuación máxima que se puede obtener por cada algoritmo y su análisis. La inclusión de trabajo voluntario (experimentación con diferentes parámetros, prueba con otros operadores o versiones adicionales del algoritmo, análisis extendido, etc.) podrá incrementar la nota final.

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.