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 la Mínima Dispersión Diferencial (MDD)

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 la Mínima Dispersión Diferencial (MDD) 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 la Mínima Dispersión

El problema de la mínima dispersión diferencial (en inglés, *minimum differential dispersion problem*, MDD) 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 minimice la dispersión entre los elementos escogidos. El MDD se puede formular como:

$$\text{Minimizar } Max_{x_i \in M}\left(\sum_{j \in M} d_{ij}\right) - Min_{x_i \in M}\left(\sum_{j \in M} d_{ij}\right) \text{ con } M \subset N, |M| = m$$

donde:

- M es una solución al problema que consiste en un vector binario que indica los m elementos seleccionados
- d_{ij} es la distancia existente entre los elementos i y j.

3.2 Casos del MDD Considerados

Se utilizarán **50 casos** seleccionados de varios de los conjuntos de instancias disponibles en la MDPLIB (*https://grafo.etsii.urjc.es/optsicom/mdp/*), todas pertenecientes al grupo **GDK** con distancias aleatorias:

- 50 del grupo **GDK-b** con distancias reales con, n entre {25, 50, 75, 100, 125, 150}, y m entre *2* y *45* (GDK-bGKD-b_1_n25_m2.txt a *GKD-b_50_n150_m45.txt*),

El fichero *Tablas_MDD_2024-25.xls*, disponible en el espacio de PRADO y la web de la asignatura, incluye las características (nombre, tamaño y coste de la mejor solución conocida) de las instancias seleccionadas. Aunque pueden ser descargados de la MDPLib, los ficheros de los 50 casos se han incluido también en las dos localizaciones del espacio de la asignatura (PRADO y web externa) para facilitar el trabajo al estudiantado.

El formato de los ficheros es el siguiente:

- Una primera línea donde se indica el número de elementos n y el número de elementos a seleccionar m del problema.
- $n \cdot (n\text{-}1)/2$ líneas con el formato $i\ j\ d_{ij}$ ($i, j\{0, \ldots, n\text{-}1\}$) que recogen el contenido de la matriz de distancias entre los elementos. Se almacena sólo la diagonal superior.

Ejemplo (*GKD-b_50_n150_m45*):

```
150 45
0 1 130.46434
0 2 170.38187
0 3 155.07985
0 4 146.31268
```

...

```
146 148 137.12430
146 149 109.40894
147 148 110.07581
147 149 144.91512
148 149 157.33194
```

4 ALGORITMOS

4.1 Algoritmo Aleatorio

El algoritmo *aleatorio* se basa en generar **100000** soluciones de forma totalmente aleatoria, y devolver la mejor solución encontrada.

4.2 Algoritmo *Greedy*-MDD

El algoritmo *greedy* del MDD se basa en la heurística de ir seleccionando los elementos que reduzcan o que aumenten lo mínimo la dispersión entre los distintos elementos elegidos. El primer elemento se escoge aleatoriamente. En los m-1 pasos siguientes se va escogiendo el elemento que implique una menor dispersión de los ya seleccionados hasta el momento y el nuevo elemento considerado.

Se realizará una única ejecución sobre cada caso del problema.

4.3 Búsqueda Local

Como algoritmo de BL para el MDD 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*. **Será obligatorio emplear la factorización** de la función objetivo en todos los casos. 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. La exploración del entorno se realizará de forma distinta según dos opciones:

- En orden totalmente aleatorio (**randLS**).
- En función de la heurística de los elementos seleccionados, de forma que aquellos que según la heurística reducirían más la dispersión, o la aumentarían menos, se exploren antes (**heurLS**)

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 hayan realizado 100000**

evaluaciones de la función objetivo, es decir, en cuanto se cumpla alguna de las dos condiciones. **Se realizará 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 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. **Será 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).

Para facilitar la comparación de algoritmos en las prácticas del MDP se considerarán dos estadísticos distintos denominados *Desv* y *Tiempo:*

- *Desv* se calcula como la media de las desviaciones, en porcentaje, del valor obtenido por cada método en cada instancia respecto al mejor valor conocido para ese caso. Es decir, para un problema de minimización como el MDD:

$$Desv = 100 \cdot \sum_{i \in Casos} \frac{ValorAlgoritmo_i - MejorValor_i}{ValorAlgoritmo_i}$$

De esta forma, no se tiene en cuenta el valor concreto de una solución para una instancia, sino que se determina lo cerca que se está de obtener la mejor solución conocida.

- *Tiempo* se calcula como la media del tiempo de ejecución empleado por el algoritmo para resolver cada caso del problema.

Cuanto menor es el valor de *Desv* para un algoritmo, mejor calidad tiene dicho algoritmo, porque en media obtiene soluciones más cercanas al mejor valor conocido. Por otro lado, si dos métodos obtienen soluciones de la misma calidad (tienen valores de *Desv* similares), uno será mejor que el otro si emplea menos tiempo en media. En la web de la asignatura hay también disponible un código en C (*timer*) para un cálculo adecuado del tiempo de ejecución de los algoritmos metaheurísticos.

La hoja Excel *Tablas_MDD_2024-25.xls* ya mencionada permite generar de forma automática los estadísticos comentados para las tablas de resultados de la práctica.

6 TABLAS DE RESULTADOS A OBTENER

Como algoritmo de BL para el MDD 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*, LSrandom, LSheur) donde se recojan los resultados de la ejecución de dicho algoritmo al conjunto de casos del problema. Tendrá la misma estructura que la tabla 1.

Tabla 1: Resultados obtenidos por el Algoritmo X en el MDD

Caso	Desv	Tiempo
GKD-b_1_n25_m2	x	x
GKD-b_2_n25_m2	x	x
GKD-b_3_n25_m2	x	x
GKD-b_4_n25_m2	x	x
GKD-b_5_n25_m2	x	x
GKD-b_6_n25_m7	x	x
GKD-b_7_n25_m7	x	x
GKD-b_8_n25_m7	x	x
GKD-b_9_n25_m7	x	x
GKD-b_10_n25_m7	x	x
GKD-b_11_n50_m5	x	x
GKD-b_12_n50_m5	x	x
GKD-b_13_n50_m5	x	x
GKD-b_14_n50_m5	x	x
GKD-b_15_n50_m5	x	x
GKD-b_16_n50_m15	x	x
GKD-b_17_n50_m15	x	x
GKD-b_18_n50_m15	x	x
GKD-b_19_n50_m15	x	x
GKD-b_20_n50_m15	x	x
GKD-b_21_n100_m10	x	x
GKD-b_22_n100_m10	x	x
GKD-b_23_n100_m10	x	x
GKD-b_24_n100_m10	x	x
GKD-b_25_n100_m10	x	x
GKD-b_26_n100_m30	x	x
GKD-b_27_n100_m30	x	x
GKD-b_28_n100_m30	x	x
GKD-b_29_n100_m30	x	x
GKD-b_30_n100_m30	x	x
GKD-b_31_n125_m12	x	x
GKD-b_32_n125_m12	x	x
GKD-b_33_n125_m12	x	x
GKD-b_34_n125_m12	x	x
GKD-b_35_n125_m12	x	x
GKD-b_36_n125_m37	x	x
GKD-b_37_n125_m37	x	x
GKD-b_38_n125_m37	x	x
GKD-b_39_n125_m37	x	x
GKD-b_40_n125_m37	x	x
GKD-b_41_n150_m15	x	x
GKD-b_42_n150_m15	x	x
GKD-b_43_n150_m15	x	x
GKD-b_44_n150_m15	x	x
GKD-b_45_n150_m15	x	x
GKD-b_46_n150_m45	x	x
GKD-b_47_n150_m45	x	x
GKD-b_48_n150_m45	x	x
GKD-b_49_n150_m45	x	x
GKD-b_50_n150_m45	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 2 agrupados por tamaño, y en la tabla 3 en total.

Tabla 2: Resultados globales por Tamaño en el MDD

Algoritmo	Tamaño	Desv	Tiempo
Greedy	50	x	x
LSrandom	50	x	x
LSheur	50	x	x
Greedy	100	x	x
LSrandom	100	x	x
LSheur	100	x	x
Greedy	150	x	x
LSrandom	150	x	x
LSheur	150	x	X

Tabla 3: Resultados globales totales en el MDD

Algoritmo	Desv	Tiempo
Greedy	x	x
LSrandom	x	x
LSheur	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.