Metaheurísticas

Seminario 2. Problemas de optimización con técnicas basadas en búsqueda local

1. Problema de la Mínima dispersión differencial (MDDP)
   - Definición del Problema
   - Ejemplo de Aplicación
   - Análisis del Problema
   - Solución Greedy
   - Búsquedas por Trayectorias Simples
   - Casos del problema.
   - Agradecimientos

Definición del Problema

- El Problema de Dispersión Diferencial, Minimum Differential Dispersion Problem (MDDP) es un problema de optimización combinatoria con una formulación sencilla pero una resolución compleja (**es NP-completo**), que solo con tamaño 50 implica más de 1 hora.

- El problema general consiste en seleccionar un subconjunto *Sel* de m elementos ($|M|=m$) de un conjunto inicial S de n elementos (obviamente, $n > m$) de forma que se **minimize** la dispersión entre los elementos escogidos.

- Además de los n elementos (e_i, $i=1,\ldots,n$) y el número de elementos a seleccionar m, se dispone de una matriz $D=(d_{ij})$ de dimensión $n \times n$ que contiene las distancias entre ellos

Definición del Problema Min-Diff

- Para el problema Min Differential *Dispersion*, **con el que trabajaremos en prácticas**, se busca lo siguiente:

$$\text{Minimize} \quad Max_{i \in M}\{\sum_{j \in M} d_{ij}\} - Min_{i \in M}\{\sum_{j \in M} d_{ij}\}$$

$$\text{Subject to} \quad M \subset N, |M| = m$$

- Las distancias entre pares de elementos se usan para formular el modelo como un problema de optimización binario cuadrático
- Esa formulación es poco eficiente. Se suele resolver como un problema equivalente de programación lineal entera.

Definición del Problema Min-Diff

- Para el problema Min Differential *Dispersion*, **con el que trabajaremos en prácticas**, se calcula la dispersión como:

  1) Para cada punto elegido v se calcula Δ(v) como la suma de las distancias de este punto al resto.

  $$\Delta(v) = \sum_{u \in S} d_{uv}.$$

  2) La dispersión de una solución, denotada como diff(S) se define como la diferencia entre los valores extremos:

  $$\mathit{diff}(S) = \max_{u \in S} \Delta(u) - \min_{v \in S} \Delta(v).$$

  3) El objetivo es minimizar dicha medida de dispersión:

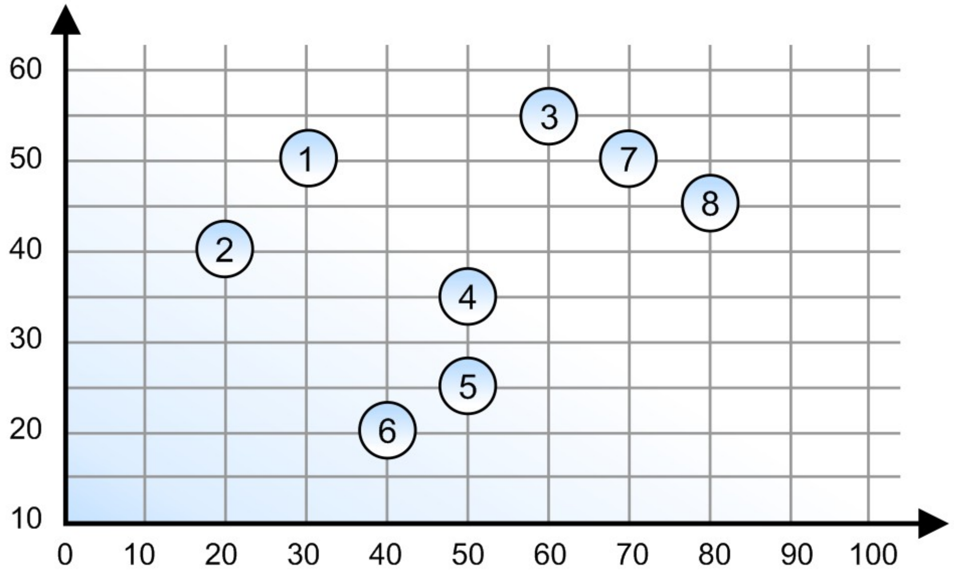
  $$S^{\star} = \arg\min_{S \subseteq V_m} \mathit{diff}(S),$$

  donde *S* es el conjunto solución al problema

Ejemplo de Aplicación: Selección de miembros de un Comité

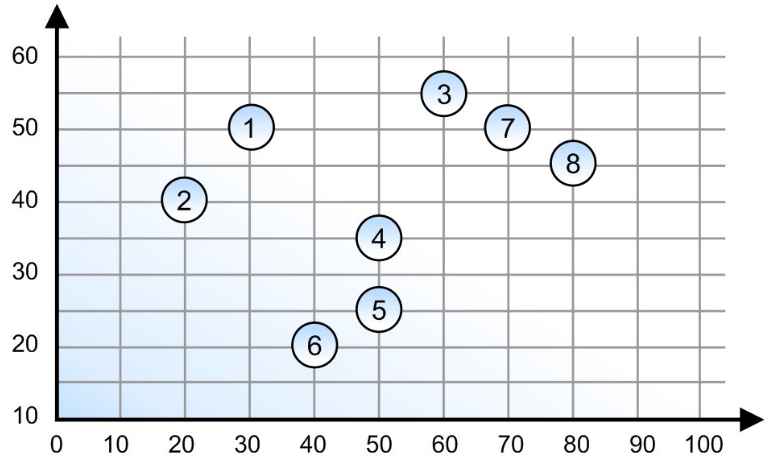
- Tenemos n=8 posibles localizaciones para colocar m=4 farmacias. Queremos situarlas de tal manera que estén separadas entre sí a una distancia parecida (mínima dispersión entre sí):

	Latitud	Longitud
1	50	30
2	40	20
3	55	60
4	35	50
5	25	50
6	20	40
7	50	70
8	45	80



Ejemplo de Aplicación: Selección de miembros de un Comité

- La distancia entre los puntos del gráfico refleja la distancia entre las distintas localizaciones
- La matriz *D* contiene los valores de dichas distancias. En este ejemplo se ha empleado la distancia Euclídea aunque se pueden usar otras métricas



	2	3	4	5	6	7	8
1	14	30	25	32	32	40	50
2		43	30	34	28	51	60
3			22	32	40	30	22
4				10	18	25	32
5					11	32	36
6						42	47
7							11

Matriz de distancias *D*

Ejemplo de Aplicación:
Selección de miembros de un Comité

EJEMPLO DEL MODELO MIN-DIFF (Min-Diff)

- La dispersión entre los elementos escogidos es la **máxima diferencia** de las sumas de las distancias existentes entre ellos:

	2	3	4	5	6	7	8
1	14	30	25	32	32	40	50
2		43	30	34	28	51	60
3			22	32	40	30	22
4				10	18	25	32
5					11	32	36
6						42	47
7							11

Localizaciones seleccionadas:

$x = \{ 3, 4, 6, 8 \}$

$$\left.\begin{array}{l} V(3)=22+40+22=\ 84 \\ V(4)=22+18+32=\ 72 \\ V(6)=40+18+47=105 \\ V(8)=22+32+47=101 \end{array}\right\} \text{Diferencia}$$

$\text{diff}(x) = 105 - 72 = 33$

La solución del modelo **MinDiff** consiste en encontrar el conjunto de 4 empleados con **la menos dispersión** entre ellos

Ejemplo de Aplicación:
Selección de miembros de un Comité

EJEMPLO DEL MODELO MINDIFF (MinDiff) (2/3)

- La dispersión entre los elementos escogidos es la **máxima diferencia** de las sumas de las distancias existentes entre ellos:

	2	3	4	5	6	7	8
1	14	30	25	32	32	40	50
2		43	30	34	28	51	60
3			22	32	40	30	22
4				10	18	25	32
5					11	32	36
6						42	47
7							11

Sol 2: Localizaciones seleccionadas:

$$x = \{ 1, 3, 6, 7 \}$$

$$\left.\begin{array}{l} V(1)=30+32+40=102 \\ V(3)=30+40+30=100 \\ V(6)=32+40+42=114 \\ V(7)=40+30+42=112 \end{array}\right\} \text{Diferencia}$$

$$\text{diff}(x) = 114 - 100 = 14$$

La solución del modelo **MinDiff** consiste en encontrar el conjunto de 4 empleados con **la menos dispersión** entre ellos

Ejemplo de Aplicación:
Selección de miembros de un Comité

EJEMPLO DEL MODELO MAXMIN (MMDP) (3/3)

- La diversidad entre los elementos escogidos es el **mínimo** de las distancias existentes entre ellos:

Sol 1: seleccionadas:

$x = \{ 3, 4, 6, 8 \}$

$$\underbrace{84 \quad 72 \quad 105 \quad 101}_{\text{Diferencia}}$$

$$\Downarrow$$

$diff(x_1) = 33$

Sol 2: seleccionadas:

$x = \{ 1, 3, 6, 7 \}$

$$\underbrace{102 \quad 100 \quad 114 \quad 112}_{\text{Diferencia}}$$

$$\Downarrow$$

$diff(x_2) = 14$

La solución del modelo **MinDiff** consiste en encontrar el conjunto de 4 empleados con **la menor** **dispersión** entre ellos

Aplicaciones del Problema

- Elegir localización de elementos públicos (farmacias, ...)
- Selección de grupos homogéneos
- Identificación de redes densas
- Reparto equitativo
- Problemas de flujo

Duarte, A, Sánchez-Oro, J., Resende, M.G.C, Glover, F, Martí, R (2015). Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. Information Sciences 296, 46-60

Solución Greedy

Duarte, A, Sánchez-Oro, J., Resende, M.G.C, Glover, F, Martí, R (2015). Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. Information Sciences 296, 46-60

- La complejidad del problema ha provocado que se hayan aplicado muchos algoritmos aproximados para su resolución

- Podemos determinar que una buena fórmula heurística para resolver el problema es:

Añadir secuencialmente el elemento no seleccionado que reduzca la dispersión con respecto a los ya seleccionados

Solución Greedy

- El algoritmo valora en cada caso cómo varía la dispersión al seleccionar cada nuevo elemento:

- El primer elemento seleccionado no está definido, puede ser aleatorio.

- Cada vez que se añade un nuevo elemento al conjunto de seleccionados *Sel*, se valora cuál incrementa menos (o reduce) la dispersión respecto a los ya elegidos

- El proceso itera hasta seleccionar los *m* elementos deseados

Solución Greedy

ALGORITMO GREEDY:

$S \leftarrow \emptyset$

$\mathrm{CL} \leftarrow V$

$\boxed{v_0 \leftarrow \texttt{SelectRandom}(\mathrm{CL})}$ **Solución Inicial**

$S \leftarrow S \cup \{v_0\}$

$\mathrm{CL} \leftarrow \mathrm{CL} \setminus \{v_0\}$

while $|S| < m$ **do**

  $\mathrm{RCL} \leftarrow \mathrm{CL}$

  $\boxed{u \leftarrow \arg\min_{v \in \mathrm{RCL}} g(v)}$ **Aplicar heurística**

  $S \leftarrow S \cup \{u\}$

  $\mathrm{CL} \leftarrow \mathrm{CL} \setminus \{u\}$

end while

return S

Solución Greedy

El cálculo de g(u) se aplica de la siguiente manera:

1) Para cada elemento *u* no escogido:

$$\forall\, u \in V - Sel, \partial(u) = \sum_{v \in Sel} d_{uj}$$

2) Luego para cada elemento v existente:

$$\forall\, v \in Sel, \partial(v) = SumaAnterior(v) + d_{uv}$$

3) Una vez actualizado las sumas para cada elemento, se calcula:

$$\partial_{max}(u) = max\left(\partial(u), max_{v \in Sel}\, \partial(v)\right) \qquad \partial_{min}(u) = min\left(\partial(u), min_{v \in Sel}\, \partial(v)\right)$$

4) El cálculo final de g(u) es:

$$g(u) = \partial_{max}(u) - \partial_{min}(u)$$

Búsquedas por Trayectorias Simples: Búsqueda Local del Mejor

- **Representación**: Problema de selección: un conjunto *Sel*=$\{s_1, \ldots, s_m\}$ que almacena los *m* elementos seleccionados de entre los *n* elementos del conjunto *S*

  Para ser una solución candidata válida, tiene que satisfacer las restricciones (ser un conjunto de tamaño *m*):

  - No puede tener elementos repetidos
  - Ha de contener exactamente *m* elementos
  - El orden de los elementos no es relevante

Búsquedas por Trayectorias Simples: Búsqueda Local del Mejor

- **Operador de vecino de intercambio y su entorno**: El entorno de una solución *Sel* está formado por las soluciones accesibles desde ella a través de un movimiento de intercambio

  Dada una solución (conjunto de elementos seleccionados) se escoge un elemento y se intercambia por otro que no estuviera seleccionado (*Int*(*Sel*,*i*,*j*)):

$$Sel = \{s_1, \ldots, i, \ldots, s_m\} \quad \Rightarrow \quad Sel' = \{s_1, \ldots, j, \ldots, s_m\}$$

- <u>*Int*(*Sel*,*i*,*j*) verifica las restricciones</u>: si la solución original *Sel* es factible y el elemento *j* se escoge de los no seleccionados en *Sel*, es decir, del conjunto *S*-*Sel*, siempre genera una solución vecina *Sel'* factible

Búsquedas por Trayectorias Simples: Búsqueda Local del Mejor

- Su aplicación provoca que el tamaño del entorno sea demasiado grande (m!), m=10 => más de 3 millones combinaciones.
- La BL del Mejor del MDP explora todo el vecindario, las soluciones resultantes de los $m \cdot (n-m)$ intercambios posibles, escoge el mejor vecino y se mueve a él siempre que se produzca mejora
- Si no la hay, detiene la ejecución y devuelve la solución actual
- El método funciona bien pero es muy lento incluso para casos no demasiado grandes (n=500) y usando un cálculo factorizado del coste $z_{MS}(Sel)$ para acelerar la ejecución ($O(n)$)
- Es recomendable utilizar una estrategia avanzada más eficiente

Búsqueda Local del Primer Mejor

Duarte, A, Sánchez-Oro, J., Resende, M.G.C, Glover, F, Martí, R (2015). Greedy randomized adaptive search procedure with exterior path relinking for differential dispersion minimization. Information Sciences 296, 46-60

- Algoritmo de búsqueda local del primer mejor: en cuanto se genera una solución vecina que mejora a la actual, se aplica el movimiento y se pasa a la siguiente iteración
  - Se detiene la búsqueda cuando se ha explorado el vecindario completo sin obtener mejora (o tras un número fijo de evaluaciones)
- Se puede explorar el vecindario de forma inteligente:
  - Se calcula la contribución de cada elemento seleccionado al coste de la solución actual (valor de la función objetivo $z_{MS}(Sel)$)
  - Se aplican primero los intercambios de elementos que menos contribuyen
- Se considera una factorización para calcular el coste de *Sel'* a partir del de *Sel* considerando sólo el cambio realizado en la función objetivo por el movimiento aplicado. Además, se "factoriza" también el cálculo de la contribución

Exploración Inteligente del Vecindario

- Técnica que permite focalizar la BL en una zona del espacio de búsqueda en la que potencialmente puede ocurrir algo

- Reduce significativamente el tiempo de ejecución con una reducción muy pequeña de la eficacia de la BL del Mejor (incluso puede mejorarla en algunos problemas)

- Se basa en definir un orden de aplicación de los intercambios (exploración de los vecinos) en una BL del primer mejor

- En cada iteración, al cambiar un nodo *u* por el nodo *v mode(Sel, u, v)* se podría recalcular la medida de dispersión desde cero, pero es mejor calcular la mejora definida como *move_value(Sel, u, v)*

Exploración Inteligente del Vecindario

- En lugar de calcular el valor del movimiento *Int*(*Sel*,*i*,*j*) para todos los intercambios posibles, se escoge el elemento s_{i*} de *Sel* que presenta el menor aporte (es decir, el valor v para el que se *move_value(Sel, u, v)* sea mínimo).

- Tras escoger el elemento a extraer, se prueban sucesivamente los intercambios por los elementos no seleccionados:
  - Si se encuentra un movimiento de mejora, se aplica. Si no, se pasa al siguiente elemento con menor aporte y se repite el proceso
  - Si ningún movimiento del vecindario provoca mejora, se finaliza la ejecución y se devuelve la solución actual

Factorización del Movimiento de Intercambio

- Para generar *Sel'*, el operador de vecino *Int*(*Sel*,*i*,*j*) escoge un elemento seleccionado *i* y lo cambia por uno no seleccionado *j*:
  - *Sel* = {s_1, …, *i*, …, s_m} $\Rightarrow$ *Sel'* $\leftarrow$ Sel - {*i*} + {*j*} $\Rightarrow$ *Sel'* = {s_1, …, *j*, …, s_m}

- No es necesario recalcular todas las distancias de la función objetivo:
  - Al añadir un elemento, las distancias entre los que ya estaban en la solución se mantienen y basta con calcular la dispersión por el nuevo elemento al resto de elementos seleccionados
  - Al eliminar un elemento, las distancias entre elementos que se quedan en la solución se mantienen y basta con restar la distancia del elemento eliminado al resto de elementos en la solución

- Se calcula el nuevo coste de la solución original *Sel como*:

$$Z_{MM}(Sel, u, v) = (\partial_{max} - \partial_{min}) - z_{MM}(Sel)$$

$$\partial_{max} = max\left(\partial(v), max_{w \in Sel}\, \partial(w)\right) \qquad \partial_{min} = min\left(\partial(v), min_{w \in Sel}\, \partial(w)\right)$$

$$\partial(v) = \sum_{w \in Sel} d_{vw} \qquad \forall\, w \in Sel, \partial(w) = anterior(w) - d_{wu} + d_{wv}$$

Factorización del Movimiento de Intercambio

- El coste del movimiento (la diferencia de costes entre las dos soluciones) $\Delta z_{MM}(Sel,i,j) = z_{MM}(Sel') - z_{MM}(Sel)$ se calcula factorizado.
- El cálculo original, implicaba calcular para cada uno de los m elementos la distancia al resto, por tanto era $O(n^2)$. De forma factorizada es sól $O(n)$ considerando las *m*-1 distancias del elemento que ***se elimina*** y las *m*-1 que ***se añaden.***
- Si $\Delta z_{MM}(Sel,i,j)$ es negativo ($\Delta z_{MM}(Sel,i,j)<0$), la solución vecina *Sel'* es mejor que la actual *Sel* (es un problema de minimización) y se acepta. Si no, se descarta y se genera otro vecino.

- Podemos combinar fácilmente la factorización del coste con el cálculo de la contribución de los elementos para mejorar aún más la eficiencia:
  - Las distancias del elemento eliminado equivalen directamente a la contribución de dicho elemento,
  - El cálculo de las aportaciones de los elementos actualmente seleccionados también se puede factorizar. No es necesario recalcularlo completamente, basta con restar la distancia del elemento eliminado y sumar la del añadido:

BL-MDP: Factorización del Movimiento de Intercambio

- El coste $z_{MM}(Sel')$ de la nueva solución vecina es:

$$z_{MM}(Sel') = z_{MM}(Sel) + \Delta z_{MM}(Sel,i,j)$$

- Sólo es necesario calcularlo al final de la ejecución. Durante todo el proceso, basta con trabajar con el coste del movimiento

Repetir

$Sel' \leftarrow$ GENERA_VECINO(Sel);

Hasta ($\Delta z_{\mathrm{MM}}(Sel,i,j) < 0$) **O**

(se ha generado $E(Sel)$ al completo)

Casos del Problema

- Existen distintos grupos de casos del problema para los que se conoce la solución óptima que permiten validar el funcionamiento de los algoritmos de resolución

- Para el MDP, disponemos de cuatro grandes grupos de casos:
  - **Casos GKD** (Glover, Kuo and Dhir, 1998): Entre otras, 20 matrices $n \times n$ con distancias Euclideas calculadas a partir de puntos con r coordenadas ($r \in \{2, \ldots, 21\}$) aleatorias en [0,10]. n=500 elementos y m=50
  - **Casos SOM** (Silva, Ochi y Martins, 2004): Entre otras, 20 matrices $n \times n$ con distancias enteras aleatorias en {0,9} con $n \in \{100, \ldots, 500\}$ elementos y $m \in \{0.1 \cdot n, \ldots, 0.4 \cdot n\}$. P.ej. para n=100 hay 4 casos con m=10, 20, 30, 40
  - **Casos MDG** (Duarte y Martí, 2007):
    - **Tipo a**: 40 matrices $n \times n$ con distancias enteras aleatorias en {0,10}: 20 con n=500 y m=50; y 20 con n=2000 y m=200
    - **Tipo b**: 40 matrices $n \times n$ con distancias reales aleatorias en [0,1000]: 20 con n=500 y m=50; y 20 con n=2000 y m=200
    - **Tipo c**: 20 matrices $n \times n$ con distancias enteras aleatorias en {0,1000}. n=3000 y m={300,400,500,600}

Casos del Problema

- Los casos están recopilados en la **biblioteca MDPLib**, accesible en la Web en la dirección siguiente:

  https://grafo.etsii.urjc.es/optsicom/mindiff/

- En dicha dirección pueden encontrarse tanto los datos como los valores de las mejores soluciones encontradas para 315 casos del problema

- Además, están disponibles los resultados de un ejemplo de una experimentación comparativa de distintos algoritmos con 10 minutos de tiempo de ejecución por caso

La Biblioteca MDPLIB

- El formato de los ficheros de datos es un fichero de texto con la siguiente estructura:

  $n\ m$
  D

  donde n es el número de elementos, m es el número de elementos seleccionados y D es la matriz de distancias entre elementos que está precalculada

- Al ser D una matriz simétrica, sólo se almacena la diagonal superior. El fichero contendrá $n \cdot (n-1)/2$ entradas, una por línea, con el siguiente formato:

  $i\ j\ d_{ij}$

  donde $i, j \in \{0, \ldots, n-1\}$ son respectivamente la fila y la columna de la matriz D, mientras que d_{ij} es el valor de la distancia existente entre los elementos $i+1$ y $j+1$

La Biblioteca MDPLIB

EJEMPLO: FICHERO DEL CASO GKD-c_1_n500_m50:

```
500 50
0 1 11.17945
0 2 12.18565
0 3 15.82056
0 4 7.17287
0 5 12.63171
0 6 10.45706
0 7 12.37497
0 8 12.13219
0 9 13.07364
0 10 10.54751
0 11 9.96995
0 12 12.55428
0 13 12.86351
0 14 7.08237
...
496 497 14.48240
496 498 11.37189
496 499 13.94453
497 498 15.47191
497 499 17.05433
498 499 10.37931
```

Agradecimientos

- Para la preparación de las transparencias de presentación del problema MDPLIB se han usado materiales de los profesores:
  - Rafael Martí. Universidad de Valencia
  - Abraham Duarte. Universidad Rey Juan Carlos
  - Jesús Sánchez-Oro. Universidad Rey Juan Carlos

- Su grupo de investigación ha realizado muchas publicaciones sobre el problema y mantiene la biblioteca MDPLIB. Referencias:

  - Duarte A., Sánchez-Oro J., Resende M.G.C., Glover F., Martí R. Greedy randomized search procedure with exterior path relinking for differential dispersion minimization. Information Sciences, (2016), 46-60.
  - Resende M.G.C., Werneck R.F.A hybrid heuristic for the p-median problemJournal of Heuristics, 10(1) (2016), 59-88
  - Lai X., Hao J-K, Glover, Fred, Yue D. Intensification-driven tabu search for the minimum differential dispersion problem. Knowledge-Based System, 5, January 2019.
  - Aringhieri, R., Cordone R., Grosso A. Construction and improvement algorithms for dispersion problems. European Journal of Operational Research 242 (2015), 21-33.