Algorítmica (2016-2017)

Grado en Ingeniería Informática

Universidad de Granada

Práctica 3

Algoritmos Greedy:
Problema de Asignación Cuadrática

Gregorio Carvajal Expósito
Gema Correa Fernández
Jonathan Fernández Mertanen
Eila Gómez Hidalgo
Elías Méndez García
Alex Enrique Tipán Párraga

Contenidos

Análisis del Problema

En el **problema de asignación cuadrática** (QAP) disponemos de n unidades y n localizaciones, con el fin de encontrar la asignación óptima de cada unidad a una localización. Además, el número de unidades coincide con el número de localizaciones.

Las localizaciones tendrán diferentes distancias entre sí, y una distancia entre un lugar A a un lugar B no tiene porqué ser la misma que de B a A. Llamaremos *distancia potencial* a la suma de las distancias desde un lugar dado hacia las demás.

Las unidades tendrán diferentes flujos de interacción entre sí, donde un flujo del objeto A con el objeto B no tiene porqué ser el mismo que el del objeto B con el objeto A. Llamaremos *flujo potencial* a la suma de flujos de un objeto dado con el resto.

El QAP se puede formular como:

$$p^* = min_p \left\{ \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{p(i)p(j)} \; d_{ij} \right\}$$

- p: es una solución al problema que consiste en una permutación que representa la asignación de la unidad i a la localización p(i)
- $f_{p(i)p(j)}$: es el flujo que circula entre la unidad i y la j
- d_{ij}: es la distancia existente entre la localización i y la j

En nuestro problema las unidades serán los oficinistas y las localizaciones serán las habitaciones donde los ubicaremos.

Diseño de la Solución

Lista de Candidatos

La lista de candidatos viene dada por todas las parejas "unidad$_j$ en localización$_i$" posibles. En nuestro caso particular sería una lista formada por parejas de oficinista y habitación.

Conjunto Solución

Habremos obtenido nuestro conjunto solución cuando todas las unidades hayan sido asignadas a una localización.

Función Selección

Nuestra función de selección dependerá del beneficio de los candidatos, que viene dado por el valor absoluto de la diferencia entre el flujo potencial de la unidad y la distancia potencial de la localización . Para así, seleccionar el candidato que mayor beneficio tenga, que en nuestro caso es el mayor de las diferencias calculadas anteriormente. De esta forma intentamos que las unidades con menor flujo potencial queden emparejadas con las localizaciones con mayor distancia potencial y viceversa.

Función de Factibilidad

Un candidato será factible de ser introducido en la solución, si aún no se ha asignado esa unidad a una localización, y la localización no se ha asignado a ninguna otra unidad. Solamente puede haber una unidad por localización.

Estructura del archivo que contiene el problema a resolver

Para mayor comodidad, vamos a almacenar los posibles problemas en ficheros de texto, los cuales nuestro programa será capaz de cargar. El número de líneas de este fichero será de 2n + 1, siendo n el tamaño del problema (número de localización y unidades). La estructura será la siguiente:

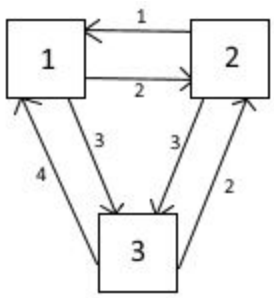
- En la primera línea, tendremos un número indicando el tamaño del problema (n).
- En las siguientes n líneas se describen cada una de las localizaciones. Cada una de estas líneas tendrán números separados por espacios indicando la distancia a cada uno de los otros lugares.
- Las siguientes n líneas describen cada uno de las unidades. Cada una de estas líneas tendrán números separados por espacios indicando el flujo de interacción con cada uno de los otros objetos.

Ejemplo de archivo contenedor de problema:

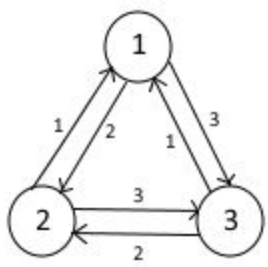
Contenido del fichero:

```
3
0 2 3
1 0 3
4 2 0
0 2 3
1 0 3
1 2 0
```

Distancias de los lugares:



Flujos de los objetos:



¿Cómo obtener la Solución?

Partiendo de las distancias y flujos potenciales, los cuales están relacionados con las localizaciones(H) y unidades(O) respectivamente, obtenemos una matriz de beneficios de cada una de las asignaciones y elegimos el que sea mayor. Realizamos este procedimiento hasta completar todas y cada uno de las localizaciones.

La implementación de esta idea es muy simple. Hemos optado por almacenar las distancias y flujos potenciales en dos vectores tales que:

H1	H2	H3	H4	H5
dpH1	*dpH2*	*dpH3*	*dpH4*	*dpH5*

O1	O2	O3	O4	O5
fpO1	*fpO2*	*fpO3*	*fpO4*	*fpO5*

A partir de estos vectores hacemos la selección antes explicada y obtenemos un vector nuevo con las asignaciones de cada unidad a cada localización, donde el índice del vector hace referencia a la habitación, y el valor del vector en esa posición al oficinista asociado. Más adelante se explica a fondo el funcionamiento del algoritmo.

H1	H2	H3	H4	H5
O?	*O?*	*O?*	*O?*	*O?*

Esqueleto del Algoritmo

```
S:Solucion greedyQAP (L:localizaciones, U:unidades)

      // Construir lista de candidatos
      para cada localizacion l1
            para cada localizacion l2
                  distancia_potencial += distancia_entre(l1, l2)

      para cada unidad u1
            para cada unidad u2
                  flujo_potencial_de_u += flujo_entre(u1, u2)

      LC = ∅

      para cada localizacion l
            para cada unidad u
                  b = calcular_beneficio(l, u)
                  añadir(LC, l, u, b)
      // Fin de creación de la lista de candidatos
```

```
// Seleccionamos los candidatos para la solución
mientras (LC ≠ ∅) Y NO solucion(S) {

      x = seleccionar(LC)
      insertar(S, x)
      eliminar(LC, x)
      eliminarNoFactibles(LC, x)
}

devolver S
```

Cabe destacar que en este problema no disponemos de una función **Factibilidad** que compruebe si un candidato es factible o no. Esto es porque, dada la naturaleza del problema, en el momento en el que seleccionamos un candidato, sabemos de antemano qué candidatos dejan de ser factibles y podemos eliminarlos de la lista de candidatos directamente.

Funcionamiento del Algoritmo

A partir de la matriz de distancias de las localizaciones, calculamos la distancia potencial de cada una de ellas como la suma de las distancias desde la localización en cuestión al resto de ellas. De la misma forma, calculamos el flujo potencial de cada unidad. Por ejemplo, dadas las siguientes matrices de distancias y flujos, obtenemos las distancias y flujos potenciales siguientes:

	0	1	2		Distancia potencial
Localizacion 0	0	2	3	→	5
Localizacion 1	1	0	3	→	4
Localizacion 2	4	2	0	→	6

	0	1	2		Flujo potencial
Unidad 0	0	2	3	→	5
Unidad 1	1	0	3	→	4
Unidad 2	1	2	0	→	3

Una vez que tenemos los vectores de distancias y flujos potenciales, pasamos a calcular el beneficio individual de cada candidato (parejas localización-unidad). Para ello, se construye una matriz de beneficios, que representa a nuestra lista de candidatos. El cálculo del beneficio de cada candidato ya ha sido detallado en el apartado **Función selección**. En nuestro ejemplo, la lista de candidatos se quedaría así:

	Unidad 0	Unidad 1	Unidad 2
Localizacion 0	0	1	2
Localizacion 1	1	0	1
Localizacion 2	1	2	3

Una vez creada la lista de candidatos con los beneficios de cada uno calculados, pasamos a seleccionar los candidatos. Para ello, buscamos el candidato que mayor beneficio tenga y lo insertamos en la solución. En caso de empate, se coge el primero encontrado. En nuestro ejemplo, el candidato seleccionado es *Unidad 2 en Localización 2*.

	Unidad 0	Unidad 1	Unidad 2
Localizacion 0	0	1	2
Localizacion 1	1	0	1
Localizacion 2	1	2	3

	0	1	2
SOLUCION			2

Posteriormente, eliminamos todos los candidatos de la misma fila y misma columna del candidato seleccionado.

	Unidad 0	Unidad 1	Unidad 2
Localizacion 0	0	1	-1
Localizacion 1	1	0	-1
Localizacion 2	-1	-1	-1

Repetimos hasta que la lista de candidatos se quede vacía y nuestra solución quede completa:

El siguiente candidato es *Unidad 0 en Localización 1*, que empata con otro candidato, pero el algoritmo lo encontraría antes:

	Unidad 0	Unidad 1	Unidad 2
Localizacion 0	0	1	-1
Localizacion 1	1	0	-1
Localizacion 2	-1	-1	-1

	0	1	2
SOLUCION		0	2

Eliminamos los candidatos necesarios:

	Unidad 0	Unidad 1	Unidad 2
Localizacion 0	-1	-1	-1
Localizacion 1	1	-1	-1
Localizacion 2	-1	-1	-1

Y repetimos otra vez hasta que no queden candidatos:

	Unidad 0	Unidad 1	Unidad 2
Localizacion 0	-1	-1	-1
Localizacion 1	1	-1	-1
Localizacion 2	-1	-1	-1

	0	1	2
SOLUCION	1	0	2

Eliminamos los candidatos necesarios:

	Unidad 0	Unidad 1	Unidad 2
Localizacion 0	-1	-1	-1
Localizacion 1	-1	-1	-1
Localizacion 2	-1	-1	-1

Como ya no quedan candidatos, ya tenemos nuestra solución construida:

	0	1	2
SOLUCION	1	0	2

En la solución queda representado que en la Localización 0 colocamos la Unidad 1, en la Localización 1 colocamos la Unidad 0 y en la Localización 2 colocamos la Unidad 2.

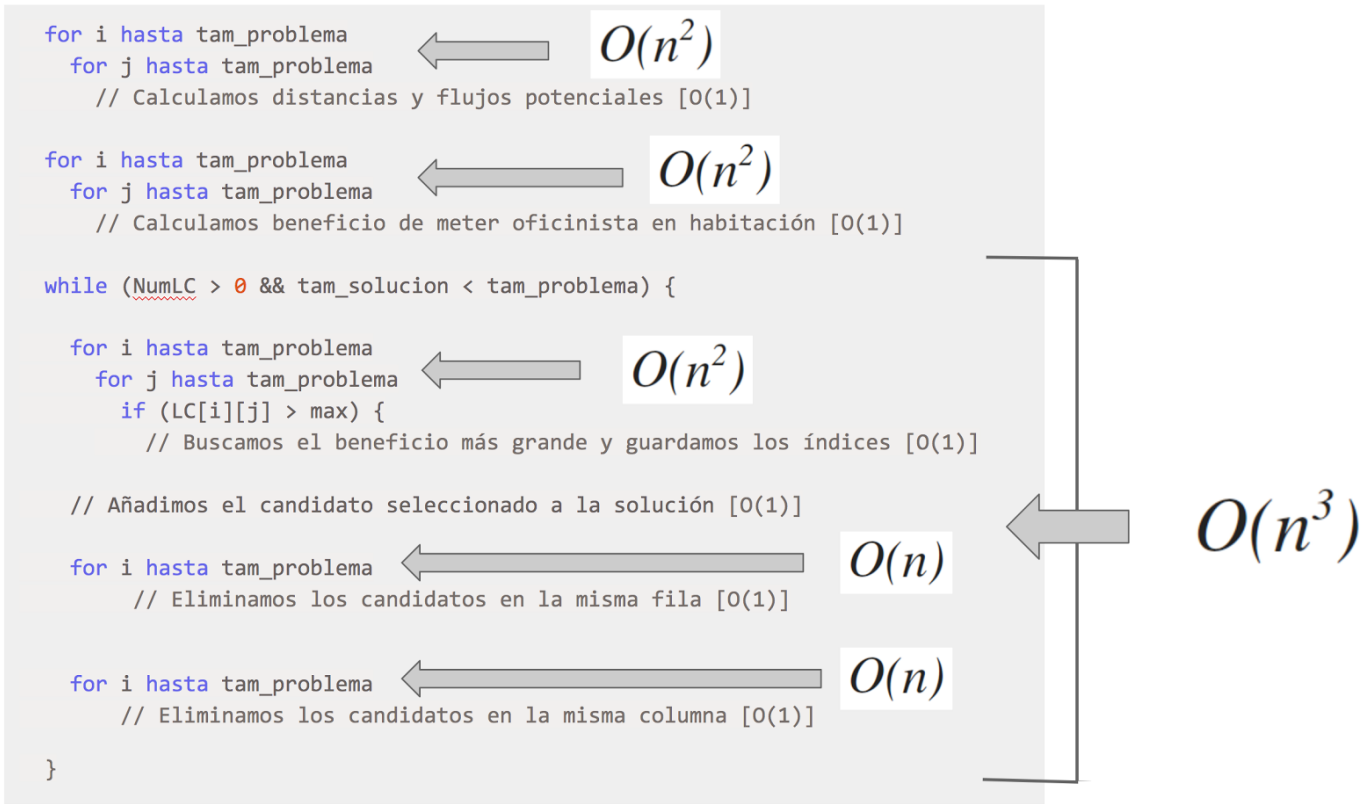
Enunciado de un Problema Real

El problema se puede describir de la siguiente forma:

Queremos decidir dónde construir n instalaciones (p.ej. fábricas) y tenemos n posibles localizaciones en las que podemos construir dichas instalaciones. Conocemos las distancias que hay entre cada par de instalaciones y también el flujo de materiales que ha de existir entre las distintas instalaciones (p.ej. la cantidad de suministros que deben transportarse de una fábrica a otra). El problema consiste en decidir dónde construir cada instalación de forma que se minimice el coste de transporte de materiales.

Cálculo de la Eficiencia Teórica

Vamos a analizar el código del Greedy con el fin de obtener su eficiencia teórica. Se puede apreciar a simple vista que el código son varios bucles anidados, por lo que nuestra eficiencia será el de orden mayor. En este caso, el bucle *while* será quien determine la eficiencia del código al encontrarnos 3 bucles anidados. Por tanto por la regla de la suma nuestra eficiencia es $O(n^3)$, que se obtiene escogiendo de la parte interna del while los bucles de orden mayor $O(n^2)$ y aplicamos la regla de la multiplicación con el bucle while, obteniendo así $O(n^3)$. Para el cálculo de la eficiencia se han simplificado todas las sentencias O(1) por un comentario de la acción que se realiza.



Instrucciones para Compilar y Ejecutar

Hemos creado un Makefile para que sea lo más sencillo posible compilar y ejecutar el algoritmo. Tan solo hay que ejecutar la orden make y se creará el ejecutable QAP_main. Para ejecutarlo, hay que introducir como parámetro un nombre de fichero, que será el fichero que contiene el problema. La estructura de este fichero se ha explicado en la sección **Análisis del Problema**. Por ejemplo, si el fichero se llamara *problema.dat*, la instrucción a ejecutar sería:

```
./QAP_main problema.dat
```