Problema QAP

El QAP, o Quadratic Assignment Problem, pertany a la classe NP-Complets i és un problema que que tracta la localització. Es tracta d'assignar N instalacions a una quantitat N de llocs o localitzacions on es considera un cost associat a cadascuna de les assignacions. Aquest cost dependrà de les distàncies i el flux entre les instalacions.

Branch and Bound

Aquest algorisme és el que hem triat per afrontar el problema QAP que se'ns planteja i consisteix en un recorregut seguint diverses estratègies marcades.
El Branch & Bound es divideix en dos parts molt marcades:

- La primera part és el recorregut sistemàtic de l'arbre d'estats d'un problema (Lazy o Eager). Per fer aquest recorregut utilitzarem una estratègia de ramificació, o Branching.
- Després tenim les tècniques de poda per a eliminar tots aquells nodes que no ens portin a solucion òptimes. Això ho farem calculant la funció de cota o Bounding. En el nostre cas utilitzarem el Gilmore-Lawler per resoldre aquesta part del problema.

L'algoritme recorrerà tot l'arbre d'estat (encara que no l'arbre sencer ja que alguns nodes poden haver-se podat durant l'execució del programa) i finalment retornarà la millor solució de entre totes les nostres solucions parcials.
Les dues estratègies que hem utilitzat per a fer el Branching en el nostre arbre d'estat han sigut:

- Lazy: La evaluació de tipus lazy es podria resumir en què el bound només serà realitzat quan és realment necessari, és a dir, calcularem els boundings par les solucions parcials lo més tard possible. Amb aquest branch, seleccionarem un dels nodes de l'arbre d'estats i calcularem la seva cota. Si aquesta és més gran que la millor solució trobada la descartem, sinó anirem calculant les cotes i comprovant-les amb la millor solució fins a aconseguir una solució parcial.

- Eager: La evaluació de tipus eager farà el bound tan abans com sigui possible. Amb aquest branch, calcularem les cotes de tots els elements fills d'un pare de l'arbre d'estats del problema. Seguidament explorarem el node que té la menor

cota de tots els fills calculats, mentre que els altres nodes els guardarem en una estructura que els mantindrà ordenats (com per exemple una cua de prioritats). A l'hora en que el primer dels elements ordenats tingui una cota major que la millor solució, haurem acabat d'explorar l'arbre d'estats.

- Temps:

El temps emprat per executar el Branch and Bound Lazy amb un problema amb 20 Planetes i 14 recursos ha estat de 2890 ms.
El temps emprat per executar el Branch and Bound Eager amb un problema amb X Planetes i 14 recursos ha estat de 395 ms.

Gilmore-Lawler (GLB)

L'algorisme Gilmore-Lawler o GLB és el bound que farem servir per el nostre projecte. Aquest bound és fàcil de computar però té un gran cost ja que incrementa la mida n del problema.
El que fem per obtenir el lower bound és resoldre el LSAP amb la matriu de cost L:

$$A = \begin{matrix} 7 & 2 & 1 & 7 \\ 0 & 0 & 4 & 2 \\ 3 & 0 & 5 & 6 \\ 2 & 3 & 4 & 0 \end{matrix}$$

$$B = \begin{matrix} 1 & 7 & 3 & 1 \\ 2 & 1 & 0 & 6 \\ 0 & 4 & 2 & 6 \\ 0 & 0 & 3 & 7 \end{matrix}$$

$C = 0$.

Ara aconseguim:

$a1 = (2, 1, 7), b1 = (7, 3, 1)$;

$a2 = (0, 4, 2), b2 = (2, 0, 6)$;

$a3 = (3, 0, 6), b3 = (0, 4, 6)$;

$a4 = (2, 3, 4), b4 = (0, 0, 3)$.

Després sabem que la matriu és:

20 10 14 3
10 4 8 0
15 6 12 0
27 18 24 6

Junt amb la matriu C obtenim la matriu de costos L:

$$L = \begin{matrix} 27 & 17 & 28 & 52 \\ 10 & 4 & 8 & 0 \\ 20 & 11 & 22 & 35 \\ 27 & 18 & 24 & 6 \end{matrix}$$

La solució òptima és la permutació (2, 3, 1 ,4) amb valor òptim z = 51.

Hungarian Algorithm

L'algoritme Hungarian ens servirà per a trobar un assignament òptim per al problema del Linear Assignment Problem (LAP).
Per entendre els passos a seguir per resoldre el Hungarian, utilitzarem un exemple a mesura que expliquem cada fase d'aquest algoritme:
Aquesta és la matriu de costos original:

90	75	75	80
35	85	55	65
125	95	90	105
45	110	95	115

1. Extreure la fila mínima: Per a cada fila, busquem l'element més petit i el restem de cada element d'aquesta fila.

15	0	0	5	(-75)
0	50	20	30	(-35)
35	5	0	15	(-90)
0	65	50	70	(-45)

2. Extreure columna mínima: Per a cada columna, busquem l'element més petit i el restem de cada element d'aquesta columna.

15	0	0	0
0	50	20	25
35	5	0	10
0	65	50	65
			(-5)

3. Cobrir tots els zeros amb el mínim nombre de línies: Cobrim tots els zeros en la matriu resultant utilitzant el mínim nombre de línies horitzontals i verticals. Si es necessite n línies, existeix una assignació òptima entre els zeros. El algorisme para.

15	0	0	0	**x**
0	50	20	25	
35	5	0	10	**x**
0	65	50	65	
x				

4. Crear zeros addicionals: Busca l'element més petit que no està cobert per cap línea del pas 3. Extreu aquest element de tots els elements descoberts, i afegeix l'element a tots els elements que han estat coberts 2 vegades.

35	0	0	0
0	30	0	5
55	5	0	10
0	45	30	45

3. Cobrir tots els zeros amb el mínim nombre de línies:

35	0	0	0	**x**
0	30	0	5	
55	5	0	10	
0	45	30	45	
x		**x**		

4. Crear zeros adicionals:

40	0	5	0
0	25	0	0
55	0	0	5
0	40	30	40

3. Cobrir tots els zeros amb el mínim nombre de línies:

40	0	5	0	**x**
0	25	0	0	**x**
55	0	0	5	**x**
0	40	30	40	**x**

5. L'assignament òptim:
Com que es necessiten 4 línies per cobrir tots els zeros, aquests cobreixen un assignament òptim.

40	0	5	0
0	25	0	0
55	0	0	5
0	40	30	40

Aquest correspon al següent assignament òptim de la matriu de cost original:

90	75	75	80
35	85	55	65
125	95	90	105
45	110	95	115

El valor òptim és 275.

Tabu Search

El Tabu Search és un algoritme meta-heurístic que pot utilitzar-se per a resoldre problemes del tipus TSP o, en el nostre cas, del QAP. Aquest algorisme consisteix en anar visitant veïns (o cerca local) per tal d'arribar a una solució iterativament. La particularitat d'aquest algorisme és que cada vegada que explorem un veï, modificarà l'estructura de tots els altres veïns per tal de fer una cerca progressiva i trobar així la millor solució pel problema en qüestió.

- Temps:

El temps emprat per executar el Tabu Search amb un problema amb 20 Planetes i 14 recursos ha estat de 1930 ms.

Bibliografia

Webs:

- http://ingenierias.uanl.mx/48/48_Uso_de_busqueda.pdf
- www.seas.upenn.edu/qaplib/codes.html
- http://hungarianalgorithm.com/

Llibres:

- Burkard, Dell'Amico, Martello. Assignment problems (SIAM, 2009)(ISBN 0898716632)