ALGORÍTMICA – II
Grado en Ingeniería Informática en Sistemas de Información
Universidad Pablo de Olavide
EPD02: Búsqueda tabú

Objetivos

1. Aprender diferentes codificaciones para resolver problemas de búsqueda tabú.
2. Aprender a inicializar correctamente los parámetros de entrada.
3. Resolución de problemas reales.

Introducción

En esta práctica se van a aplicar los conceptos aprendidos en las clases de EB (Tema 3) relativos a la estrategia de búsqueda tabú (muy conocido por su acepción inglesa *tabu search*). Se muestra el esquema de búsqueda:

```
1 sBest ← s0
2 tabuList ← []
3 while (not stoppingCondition())
4         candidateList ← []
5         bestCandidate ← null
6         for (sCandidate in sNeighborhood)
7                 if ( (not tabuList.contains(sCandidate)) and (fitness(sCandidate) > fitness(bestCandidate)) )
8                         bestCandidate ← sCandidate
9                 end
10        end
11
12        if (fitness(bestCandidate) > fitness(sBest))
13                sBest ← bestCandidate
14        end
15        tabuList.push(bestCandidate);
16        if (tabuList.size > maxTabuSize)
17                tabuList.removeFirst()
18        end
19 end
20 return sBest
```

Antes de desarrollar la práctica es muy recomendable que el estudiante dedique unos minutos a recordar los conceptos básicos que se vieron en teoría. En concreto, es fundamental recordar:

1. Lista tabú.
2. Tenencia tabú.
3. Criterio de aspiración.
4. Generación de vecinos y selección del mejor candidato.
5. Condición de parada.

Resuelva los ejercicios de esta práctica con la versión básica del algoritmo, esto es, usando sólo memoria a corto plazo. Esto puede dar lugar a soluciones no óptimas, pero se prefiere para afianzar los conceptos.

Ejercicios

Ejercicio 1 (45 minutos). Queremos hallar el máximo de $f(x)=x^3-60x^2+900x+100$ entre x=0 y x=31. Para resolver el problema usando TS, se propone seguir la siguiente estrategia. En primer lugar, discretizamos el rango de valores de x con vectores binarios de 5 componentes entre 00000 y 11111. Estos 32 vectores constituyen S las

soluciones factibles del problema. En la Tabla Auxiliar puede ver cuáles son los valores asociados a cada solución posible.

Suponga una tenencia tabú de 3 unidades de tiempo. Defina una condición de parada que considere adecuada.

En una primera versión, realice el código sin tener en cuenta el criterio de aspiración. Esto es, los elementos que están en la lista tabú no pueden ser accedidos bajo ninguna circunstancia.

A continuación, modifique el código para que sí se considere el criterio de aspiración (aceptar una solución que esté en la lista tabú si es mejor de las que ya tenemos). ¿Mejora la solución?

Tabla auxiliar. Costes para f(x) en función de las soluciones.

Valores de la función $f(x)=x^3-60x^2+900x+100$ entre x=0 y x=31					
0	0	100	10000	16	3236
1	1	941	10001	17	2973
10	2	1668	10010	18	2692
11	3	2287	10011	19	2399
100	4	2804	10100	20	2100
101	5	3225	10101	21	1801
110	6	3556	10110	22	1508
111	7	3803	10111	23	1227
1000	8	3972	11000	24	964
1001	9	4069	11001	25	725
1010	10	4100 óptimo	11010	26	516
1011	11	4071	11011	27	343
1100	12	3988	11100	28	212
1101	13	3857	11101	29	129
1110	14	3684	11110	30	100
1111	15	3475	11111	31	131

Ejercicio 2 (45 minutos). Queremos resolver el problema de la mochila utilizando TS. Para ello, suponga que la capacidad de la mochila es q = 180 y que puede insertar hasta 100 elementos, con pesos aleatorios comprendidos entre [1, 100], esto es, considere el siguiente vector $P = \{p_1, p_2, \ldots, p_{100}\}$, con $p_i = [1,100]$. Además, se dispone de un beneficio asociado a cada elemento, $B = \{b_1, b_2, \ldots, b_{100}\}$, con $p_i = [1,10]$. Se permite que dos elementos pesen lo mismo y que tengan el mismo beneficio.

Resuelva el problema de tal modo que se maximice el beneficio y que no se incumpla la condición del peso máximo permitido.

¿Qué codificación escogería para modelar el problema? ¿Qué temperatura inicial pondría? ¿Cuántas combinaciones reales existen? ¿Cómo generaría la solución inicial?

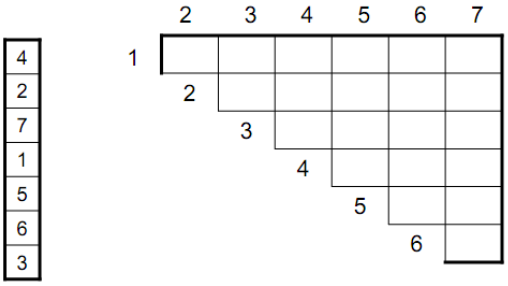
Puede consultar un código ejemplo en:
https://github.com/OmeGak/HybridKnapsack/blob/master/src/heuristics/TabuSearch.java

El proyecto principal se puede encontrar en:
https://github.com/OmeGak/HybridKnapsack

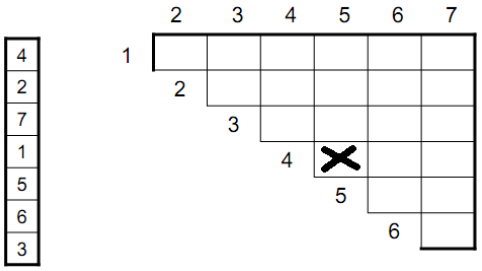
Problemas

Problema 1 (1 hora). Se desea fabricar un material aislante compuesto por siete materiales distintos. Se desea encontrar cuál es el orden en que deben mezclarse dichos materiales para asegurar que sea lo más aislante posible.

Suponga la siguiente situación:



Se puede apreciar un vector en el que aparece el orden en el que se combinan los materiales y, además, una matriz triangular para identificar posibles permutaciones de orden, es decir, posibles soluciones a las que ir.



Así, la posición (4,5) estaría haciendo referencia a que se intercambie el orden de los materiales 4 por el 5.

Para evaluar la calidad aislante de la solución, suponga que ésta se mide por la suma de los tres primeros componentes, esto es, sobre la figura de arriba sería 4+2+7 = 13. Si realizamos la permutación 4 5, entonces, la nueva solución candidata sería [5, 2, 7, 1, 4, 6, 3], siendo su coste 5+2+7 = 14 (>13) y por tanto se aceptaría. De esta condición se deduce que el máximo se alcanzará cuando tengamos en las tres posiciones superiores {5, 6, 7}, en cualquier orden posible o, en otras palabras, que el máximo global sería 5+6+7 = 18.

Puede consultar información adicional en: http://stackoverflow.com/questions/6324062/tabu-search-example-question

Problema 2 (2 horas). Un modelo de coche se configura a partir de n componentes distintos. Cada uno de esos componentes puede tomar m_i, (i = 1, ..., m) posibles valores (v_{ij}). La afinidad de los consumidores para cada posible valor v_{ij} es a_{ij} y el coste c_{ij}. Se desea encontrar una combinación de componentes que alcance la máxima afinidad global con los gustos de los consumidores y cuyo coste no supere un umbral M.

Para poder comprobar que la metaheurística diseñada funciona, suponga que n = m = 4. Los valores a_{ij} y $v_{ij\ i}$ estarán entre [0, 1]. Por último, el coste estará entre [1, 100].

Problema 3 (2 horas). Se dispone de un conjunto de n procesos y un ordenador con m procesadores (de características no necesariamente iguales). Se conoce el tiempo que requiere el procesador j-ésimo para realizar el proceso i-ésimo, t_{ij}. Se desea encontrar un reparto de procesos entre los m procesadores tal que el tiempo de finalización sea lo más corto posible. Tome tantas decisiones como estime conveniente e intente comparar distintas soluciones con distintas configuraciones iniciales. Puede consultar información adicional en: http://dis.um.es/~domingo/apuntes/AlgProPar/0607/CalvoOrtega.pdf