	ALGORÍTMICA – II Grado en Ingeniería Informática en Sistemas de Información Universidad Pablo de Olavide **EPD01: Enfriamiento simulado**

Objetivos

1. Aprender diferentes codificaciones para resolver problemas de enfriamiento simulado.
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 2) relativos a la estrategia de búsqueda enfriamiento simulado (muy conocido por su acepción inglesa *simulated annealing*). 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 cualquier caso, se recuerda el esquema de búsqueda:

```
T ← T_INICIAL; solActual ← generarSolucionInicial(); solMejor ← solActual;
vecinos ← Ø;

do
     nVecinos = generarVecinos(solActual, vecinos, T);
      for i = 1 to nVecinos do
      costeActual = calcularCoste(solActual);
      costeVecino = calcularCoste(vecino[i]);
      delta = costeActual – costeVecino;
            if (delta < 0) or (probAceptacion(delta, T)==true)
                  solActual ← vecino[i];
                  costeActual = calcularCoste(solActual);
                  if (costeActual > costeMejor)
                        solMejor ← solActual;
                        costeMejor = costeActual;
                  end if
            end if
      end for
      actualizarT(T);
while (T > T_FINAL)

return solMejor;
```

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 SA, 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.

Le damos un valor inicial a T intuitivamente, por ejemplo, T_0 =100 o 500 y en cada iteración del algoritmo lo reduciremos en 10%, es decir utilizando la estrategia de descenso geométrico: $T_k = 0.9\ T_{k-1}$.

Cada iteración consiste en lo siguiente:

1. El número de vecinos queda fijado a 5, siendo éstos variaciones de un bit de la solución. Por ejemplo, si partimos de 00011, los 5 posibles vecinos resultantes serían: 10011, 01011, 00111, 00001, 00010.
2. Para aplicar el criterio de aceptación, escogemos un vecino, buscamos su coste asociado en la Tabla 1 que se proporciona y calculamos la diferencia con la solución actual. Si está más cerca del óptimo se acepta, si no, se aplica P_a (δ, T) = $e^{-\delta/T}$. Siendo T la temperatura actual y δ la diferencia de costes entre la solución candidata y la actual.
3. Concluya la búsqueda cuando el proceso se enfríe o cuando no se acepte ninguna solución de su vecindad.

Tabla 1. 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

Considere T_INICIAL = 100 y T_FINAL = 1.

Ejercicio 2 (45 minutos). Queremos resolver el problema de la mochila utilizando SA. Para ello, suponga que la capacidad de la mochila es MAX_PESO = 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, …, p_{100}}, con p_i = [1,100]. Se permite que dos elementos pesen lo mismo.

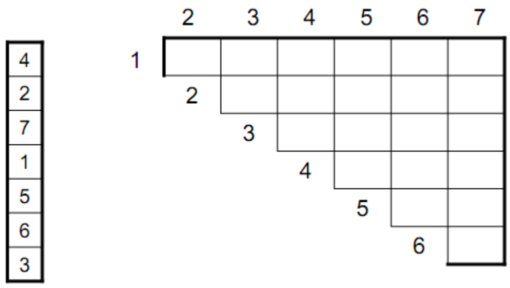
¿Qué codificación escogería para modelar el problema?
¿Qué temperatura inicial pondría?
¿Cuántas combinaciones reales existen?

Considere T_INICIAL = 200 y T_FINAL = 5.

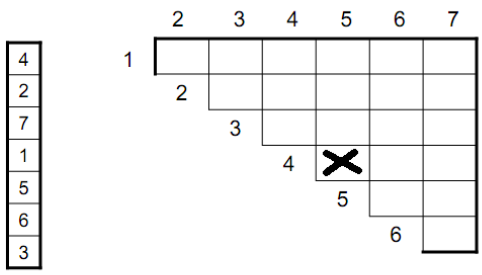
Ejercicio 3 (15 minutos). Modifique el ejercicio anterior para que, además de considerar el peso máximo de la mochila, considere un beneficio asociado a cada elemento insertado. Esto es, defina un vector de beneficios adicional, B = {b_1, b_2, …, b_{100}}, con b_i = [1,100], tal que cada elemento p_i tenga asociado un beneficio b_i. El objetivo ahora será maximizar el beneficio de los elementos insertados en la mochila, sin haber exceder el peso máximo permitido.

Problemas

Problema 1 (3 horas). 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 por 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.

Problema 2 (3 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.

Problema 3 (2 horas). Un modelo de coche se configura a partir de N componentes distintos. Cada uno de esos componentes puede tomar M posibles valores, siendo v_{ij} el valor que toma el componente i (i=1, …, N) para la posibilidad j (j=1, …, M) La afinidad de los consumidores para cada posible valor v_{ij} es a_{ij} y el coste c_{ij}. Se entiende que el coste es el valor por la afinidad, esto es, $c_{ij} = v_{ij} * a_{ij}$. 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 MAX_COSTE.

Para poder comprobar que la metaheurística diseñada funciona, suponga que N = M = 4. Los valores a_{ij} y v_{ij} estarán comprendidos entre [1, 10] y, por tanto, c_{ij} entre [1, 100]. Por último, MAX_COSTE < 120.