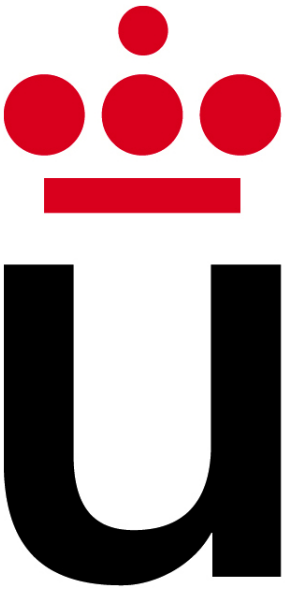
# Práctica III

## Simulación y Metaheurísticas

### Metaheurísticas

*José Ignacio Escribano*



Móstoles, 15 de mayo de 2016

# Índice de tablas

# Índice de figuras

# Índice

# 1. Introducción

En esta práctica implementaremos una de las metaheurísticas planteadas para el problema del p-hubs.

# 2. Resolución de la práctica

A continuación mostraremos la implementación del recocido simulado para el problema del p-hub. El código se encuentra en el método estático `recocidoSimulado` de la clase `Práctica3`.

```
1  static Solución recocidoSimulado(Enfriamiento.TIPOS_ENFRIAMIENTO tipo,
   ↪ Solución sol_ini, InstanciaPHub instancia,
2  int temp_ini, int nrep, double alpha, long tiempo) {
3
4      double tk = 0;
5
6      Solución x = sol_ini;
7
8      int i = 1;
9
10     switch (tipo) {
11         case BOLTZMANN: tk = Enfriamiento.crierioBoltzmann(temp_ini,
   ↪ alpha, i); break;
12         case CAUCHY: tk = Enfriamiento.esquemaCauchy(temp_ini, alpha, i);
   ↪ break;
13         case LUNDYyMEES: tk = Enfriamiento.esquemaCauchy(temp_ini, alpha,
   ↪ i); break;
14         case DESCENSO_GEOMETRICO: tk =
   ↪ Enfriamiento.crierioBoltzmann(temp_ini, alpha, i);
15     }
16
17     long fin = System.currentTimeMillis() + tiempo;
18
19     double delta;
20
21     do {
22         int m = 0;
23         do {
24             // Generamos número aleatorio entre 0 y número de soluciones
   ↪ de la vecindad - 1
25             Random r = new Random();
26             List<Solución> vecindad = PHub.generarVecindad(x, instancia);
```

```
27              int n = r.nextInt(vecindad.size());
28
29              Solución y = vecindad.get(n);
30
31              delta = y.getObjetivo() - x.getObjetivo();
32
33              if (delta < 0) {
34                  x = y;
35              } else {
36                  double u = r.nextDouble();
37                  if (u <= Math.exp(-delta / tk)) {
38                      x = y;
39                  }
40              }
41              m++;
42          } while (m != nrep);
43
44          switch (tipo) {
45              case BOLTZMANN: tk = Enfriamiento.crierioBoltzmann(temp_ini,
    ↪ alpha, i); break;
46              case CAUCHY: tk = Enfriamiento.esquemaCauchy(temp_ini, alpha,
    ↪ i); break;
47              case LUNDYyMEES: tk = Enfriamiento.esquemaCauchy(temp_ini,
    ↪ alpha, i); break;
48              case DESCENSO_GEOMETRICO: tk =
    ↪ Enfriamiento.crierioBoltzmann(temp_ini, alpha, i);
49          }
50          i++;
51      } while (System.currentTimeMillis() <= fin);
52
53      return x;
54  }
```

Esta función recibe un tipo de enfriamiento, una solución inicial, una instancia, una temperatura inicial, un número de repeticiones, un valor (parámetro de la función de enfriamiento) y el tiempo de ejecución del programa, en milisegundos.

En la línea 6 se establece que `x` es la solución inicial, y desde la línea 10 a la 16 se establece qué función de enfriamiento se utilizará para el recocido simulado.

Se elige una solución al azar de la vecindad de `x`, que llamaremos `y`, y se calcula la diferencia entre la solución nueva y la inicial. A este valor lo llamemos `delta`. Si `delta` es menor que cero, es decir $\delta = f(y) - f(x) < 0 \iff f(y) < f(x)$, es decir, se mejora la solución actual y se guarda. En caso contrario (si no se mejora la solución) se genera un número aleatorio en

el intervalo $(0, 1)$. Si este número aleatorio es menor que $e^{-\delta/t_k}$, se guarda la solución `y`. Notar que $t_k$ es el valor de la temperatura en la iteración $k$. Este proceso se repite `nrep` veces, y todo lo anterior se repite durante el tiempo especificado. Éste es nuestro criterio de parada.

En cuanto la funciones de enfriamiento se ha definido una clase `Enfriamiento` dentro de la clase `Práctica3`. Se han implementado cuatro funciones de enfriamiento, que vienen dadas por las siguientes ecuaciones:

- Descenso geométrico: $t_{i+1} = \alpha t_i,\ \alpha \in [0.8, 0.99]$
- Criterio de Boltzmann: $t_i = \dfrac{t_0}{1 + \log i}$
- Esquema de Cauchy $t_i = \dfrac{t_0}{1 + i}$
- Lundy y Mees: $t_{i+1} = \dfrac{t_i}{1 + \beta t_i}$, con $\beta$ muy pequeño

El código del criterio de Boltzmann se muestra a continuación:

```
1 public static double crierioBoltzmann(double t0, double alpha, int i) {
2     return t0 / (1 + Math.log(i));
3 }
```

Esta función recibe tres parámetros: la temperatura inicial, el valor del parámetro $\alpha$ y la iteración i. Se devuelve directamente el valor dado por la fórmula indicada anteriormente.

De forma similar se implementan los demás métodos.

## 2.1. Análisis de los resultados

Para evaluar este método ejecutamos durante 1 minuto (60 milisegundos) cada una de las instancias con cada uno de las funciones de enfriamiento.

Los parámetros usados han sido los siguientes:

- Temperatura inicial = 10000
- $\alpha = \begin{cases} 0.99, \text{ si es descenso geométrico} \\ 0.001, \text{ si es Lundi y Mees} \end{cases}$
- Número de repeticiones = 250
- Tiempo de ejecución (condición de parada) = 60000 (1 minuto por instancia)

Tabla 1: Resultados de la ejecución del recocido simulado

| Instancia | Función de enfriamiento | | | | Media |
|---|---|---|---|---|---|
| | Descenso geométrico | Criterio de Boltzmann | Esquema de Cauchy | Lundi y Mees | |
| 1 | 818.9541 | 883.8607 | 939.1282 | 891.3631 | 883.3266 |
| 2 | 1186.6325 | 1438.0429 | 1000.0160 | 974.8233 | 1149.8787 |
| 3 | 1152.2130 | 996.1984 | 1103.6910 | 1198.2770 | 1112.5949 |
| 4 | 1026.4867 | 1009.7156 | 1042.9046 | 972.5211 | 1012.9071 |
| 5 | 1018.6530 | 1033.4533 | 889.6963 | 922.5584 | 966.0903 |
| 6 | 1191.6062 | 1049.2819 | 905.0196 | 1179.4797 | 1081.3469 |
| 7 | 1142.4889 | 1140.0351 | 1472.7656 | 914.2835 | 1167.3933 |
| 8 | 1077.5733 | 1278.6855 | 1149.6770 | 890.9676 | 1099.2259 |
| 9 | 1126.6379 | 920.1679 | 971.2918 | 922.4024 | 985.1250 |
| 10 | 1158.27774 | 1130.5921 | 1121.0358 | 935.7108 | 1086.4042 |
| Media | 1089.9524 | 1088.0034 | 1059.5226 | 980.2387 | |

Los resultados de la ejecución se pueden ver en la Tabla 1. En amarillo se muestra el mejor valor de la función objetivo de cada instancia.

Se puede observar que en la primera instancia se obtiene mejor resultado con el método del descenso geométrico, en las instancias 3 y 9, el mejor método es el criterio de Boltzmann, en las instancias 5 y 6 se obtiene la mejor solución con el esquema de Cauchy, en las instancias 2, 4, 7,8 y 10 se obtiene la mejor solución con el método de Landi y Mees. Es decir, con el método del descenso geométrico se obtiene sólo 1 mejor instancia, con el criterio de Boltzmann y el esquema de Cauchy se obtienen 2 instancias y con el método de Landi y Mees se obtienen 5 instancias. Además, este método tiene la mejor media de tiempos. Parece que este método es el mejor de los cuatro (con los parámetros definidos anteriormente).

Necesitamos conocer cuál es mejor valor de $\beta$ que minimiza el valor de las instancias. Para ello, repetimos el proceso anterior para distintos parámetros de $\beta$, desde 1 hasta $10^{-10}$. Los resultados se pueden ver en la Tabla

# 3. Conclusiones