

# Lista de Algoritmos Heurísticos

Carlos Antonio Macías Duarte

4 de abril de 2019

## 1. Recocido Simulado

---

**Algoritmo 1:** Recocido Simulado (SA)

---

**Entrada:** $n$ : Tamaño de vecindad $c$ : Control de temperatura $t$ : Número de iteraciones**Salida:** $g$  : Mejor solución

```
1  $i \leftarrow 1$  : Contador de iteraciones;
2  $g \leftarrow generarSolucionInicial()$  : Solución inicial;
3 mientras  $i \leq t$  hacer
4    $p \leftarrow generarVecindad(n, g)$  : Generar nueva vecindad;
5    $j \leftarrow 1$  : Contador de la vecindad;
6   mientras  $j \leq n$  hacer
7     si  $p_j$  mejor que  $g$  entonces
8        $g \leftarrow p_j$ ;
9     si no, si  $exp(\frac{f(g) - f(p_j)}{c}) > rand(0, 1)$  entonces
10       $g \leftarrow p_j$ ;
11    $i \leftarrow i + 1$ ;
12    $n \leftarrow generarNuevoTamanoVecindad()$  : Genera el nuevo tamaño
    de la vecindad siguiente;
13    $c \leftarrow generarNuevaTemperatura()$  : Genera el nuevo valor de
    control de temperatura;
14 devolver  $g$ 
```

---

## 2. Algoritmo de Selección Clonal

---

**Algoritmo 2:** Algoritmo de Selección Clonal (CLONALG)

---

**Entrada:**

$N$ : Tamaño de población

$n$ : Número de células a clonar

$t$ : Número de generaciones

**Salida:**

$g$  : Mejor célula

```
1  $i \leftarrow 1$  : Contador de generaciones;
2  $p \leftarrow generarPoblacionInicial(N)$  : Población inicial;
3 mientras  $i \leq t$  hacer
4    $q \leftarrow seleccionarMejoresCélulas(p, n)$  : Mejores células de la
     generación;
5    $c \leftarrow clonar(q)$  : Generar clones;
6    $m \leftarrow mutar(c)$  : Hipermutación;
7    $p \leftarrow generarNuevaPoblacion(p, m, N)$  : Genera la nueva población;
8    $actualizarParametros()$  : Actualiza los parámetros de clonación y
     mutación;
9    $g \leftarrow seleccionarMejorCélula(p)$  : Mejor célula de la generación;
10   $i \leftarrow i + 1$ ;
11 devolver  $g$ 
```

---

### 3. Algoritmo Genético

---

**Algoritmo 3:** Algoritmo Genético (GA)

---

**Entrada:**

$n$ : Tamaño de la población

$t$ : Número de generaciones

**Salida:**

$g$  : Mejor individuo

1  $p \leftarrow inicializarPoblacion(n)$  : Inicialización la primera generación;

2  $i \leftarrow 1$  : Contador de iteraciones;

3  $g \leftarrow seleccionarMejorIndividuo(p)$  : Obtener el mejor individuo;

4  $u_c \leftarrow generarUmbralCruza(0, 1)$  : Umbral de cruza;

5  $u_m \leftarrow generarUmbralMutacion(0, 1)$  : Umbral de mutación;

6 **mientras**  $i \leq t$  **hacer**

7      $q \leftarrow seleccionarParejas(p)$  : Selección de las parejas;

8      $p \leftarrow cruzar(q, u_c)$  : Proceso de cruza de parejas;

9      $p \leftarrow mutar(p, u_m)$  : Proceso de Mutación de individuos;

10     $p \leftarrow generarNuevaPoblacion(p)$  : Proceso de elitismo;

11     $g \leftarrow seleccionarMejorIndividuo(p)$ ;

12     $i \leftarrow i + 1$ ;

13 **devolver**  $g$

---

## 4. Optimización por Enjambre de Partículas

---

**Algoritmo 4:** Optimización por Enjambre de Partículas (PSO)

---

**Entrada:**

$n$ : Tamaño del enjambre

$t$ : Número de iteraciones

**Salida:**

$g$  : Partícula con la mejor posición

1  $p \leftarrow inicializarEnjambre(n)$  : Inicialización del enjambre;

2  $i \leftarrow 1$  : Contador de iteraciones;

3  $g \leftarrow seleccionarMejorParticula(p)$  : Mejor partícula;

4 **mientras**  $i \leq t$  **hacer**

5      $generarNuevaVelocidad(p, g)$  : Actualiza la velocidad de las partículas;

6      $generarNuevaPosicion(p)$  : Calcular nuevas posiciones;

7      $g \leftarrow seleccionarMejorParticula(p)$  : Obtener la Mejor partícula;

8      $i \leftarrow i + 1$ ;

9 **devolver**  $g$

---

## 5. Búsqueda Armónica

---

**Algoritmo 5:** Búsqueda Armónica (HS)

---

**Entrada:***HMS*: Tamaño de la memoria*HMCR*: Razón de exploración*PAR*: Razón de ajuste de tono*BW*: Ancho de desplazamiento*NI*: Número de improvisaciones $l_i$ : Límites inferiores $l_s$ : Límites superiores**Salida:** $g$  : Mejor armonía

```
1  $HM \leftarrow inicializarMemoria(HMS, l_i, l_s)$  : Inicialización de la memoria;  
2  $i \leftarrow 1$  : Contador de iteraciones;  
3 mientras  $i \leq NI$  hacer  
4    $s \leftarrow improvisarArmonia(HM, HMCR, l_i, l_s)$  : Creación de nueva  
   armonía;  
5    $s \leftarrow ajustarTono(s, PAR, BW, l_i, l_s)$  : Ajuste al tono de la armonía;  
6    $HM \leftarrow actualizarMemoria(HM, s)$  : Actualización de la memoria;  
7    $i \leftarrow i + 1$ ;  
8  $g \leftarrow seleccionarMejorArmonia(HM)$  : Mejor armonía de la memoria;  
9 devolver  $g$ 
```

---