Lista de Algoritmos Heurísticos

Carlos Antonio Macías Duarte

29 de marzo 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 ← 1 : Contador de iteraciones;
2  g ← generarSolucionInicial() : Solución inicial;
3  mientras i ≤ t hacer
4      p ← generarVecindad(n,g) : Generar nueva vecindad;
5      j ← 1 : Contador de la vecindad;
6      mientras j ≤ n hacer
7          si p_j mejor que g entonces
8              g ← p_j;
9          si no, si exp((f(g) − f(p_j))/c) > rand(0,1) entonces
10             g ← p_j;
11     i ← i + 1;
12     n ← generarNuevoTamanoVecindad() : Genera el nuevo tamaño
         de la vecindad siguiente;
13     c ← 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 celulas a clonar
t: Número de generaciones
Salida:
g : Mejor celula

$i \leftarrow 1$: Contador de generaciones;
$p \leftarrow generarPoblacionInicial(N)$: Población inicial;
mientras $i \leq t$ **hacer**
  $q \leftarrow seleccionarMejoresCelulas(p, n)$: Mejores celulas de la generación;
  $c \leftarrow$ *clonar(q)* : Generar clones;
  $m \leftarrow$ *mutar(c)* : Hipermutación;
  $p \leftarrow generarNuevaPoblacion(p, m, N)$: Genera la nueva población;
  $actualizarParametros()$: Actualiza los parametros de clonación y mutación;
  $g \leftarrow seleccionarMejorCelula(p)$: Mejor celula de la generación;
  $i \leftarrow i + 1$;
devolver g

3. Algoritmo Genético

Algoritmo 3: Algoritmo Génetico (GA)

Entrada:
n: Tamaño de la población
t: Número de generaciones
Salida:
g : Mejor individuo

$p \leftarrow$ *inicializarPoblacion(n)* : Inicialización la primera generación;
$i \leftarrow 1$: Contador de iteraciones;
$g \leftarrow seleccionarMejorIndividuo(p)$: Obtener el mejor individuo;
$u_c \leftarrow generarUmbralCruza(0, 1)$: Umbral de cruza;
$u_m \leftarrow generarUmbralMutacion(0, 1)$: Umbral de mutación;
mientras $i \leq t$ **hacer**
    $q \leftarrow seleccionarParejas(p)$: Selección de las parejas;
    $p \leftarrow cruzar(q, u_c)$: Proceso de cruza de parejas;
    $p \leftarrow mutar(p, u_m)$: Proceso de Mutación de individuos;
    $p \leftarrow generarNuevaPoblacion(p)$: Proceso de elitismo;
    $g \leftarrow seleccionarMejorIndividuo(p)$;
    $i \leftarrow i + 1$;
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

$p \leftarrow$ *inicializarEnjambre(n)* : Inicialización del enjambre;
$i \leftarrow 1$: Contador de iteraciones;
$g \leftarrow seleccionarMejorParticula(p)$: Mejor partícula;
mientras $i \leq t$ **hacer**
    $generarNuevaVelocidad(p, g)$: Actualiza la velocidad de las partículas;
    $generarNuevaPosicion(p)$: Calcular nuevas posiciones;
    $g \leftarrow seleccionarMejorParticula(p)$: Obtener la Mejor partícula;
    $i \leftarrow i + 1$;
devolver g