

Centro de Investigación en Cómputo
Instituto Politécnico Nacional
Metaheurísticas
Actividad No. 7

Solución de problemas mediante Recocido Simulado
Curso impartido por: Dra Yenny Villuendas Rey

Adrian González Pardo

21 de octubre de 2020

1. Recocido Simulado (Simulated Annealing, SA)

Ventajas	Desventajas
Permite obtener una solución aproximada al mínimo global	Puede llegar a pasar que en la siguiente iteración a la solución salga del mínimo global y este se alcance un resultado mínimo local
Permite encontrar soluciones razonables (En tiempo)	La solución encontrada puede no ser la mejor
Utiliza poca memoria	No permite vuelta atrás en más de 1 paso

2. Pseudocódigo SA

```
1 s<-Solucion_inicial
2 e_old<-funcion_fitness(s)
3 t<-temp_max
4 while t < temp_min
5   # temp_min es aproximado a 0
6   i=0
7   while i < imax
8     # imax puede ser representativo dependiendo del problema que se aborda
9     # o en todo caso puede ser el numero de iteraciones maxima
10    seleccion_solucion_sucesor_de(s)
11    # seleccion_solucion_sucesor_de es una funcion que explora la vecindad del arbol
12    # elige aleatoriamente
13    e_new<-funcion_fitness(s)
14    delta=e_new-e_old
15    if delta>0
16      if rand(0..1) >= exp(-delta/(K*temp))
17        deshacer_sucesor(s)
18      else
19        e_old=e_new
20      end
21    else
22      e_old=e_new
23    end
24    i+=1
25  end
26  t*=alpha
```

```

28 # alpha es una constante que va al inicio del programa entre [0.88,0.99]
29 end

```

3. SA vs RMHC

La diferencia más notable entre estos dos algoritmos, es notablemente que el RMHC se basa en la mutación aleatoria de un conjunto de datos, generando un camino de solución del arbol de soluciones, mientras que SA parte de una solución donde el mismo algoritmo esta bioinspirado en un proceso físico que busca un ablandamiento para la formación o moldeo de estructuras de metal, entonces si bien ambos son algoritmos heurísticos RMHC se aplica en problemas cuya solución puede ser mutada y en este caso no puede salir de un mínimo local si este cae en ese espacio, mientras que SA permite admitir una respuesta no optima para seguir explorando el espacio de soluciones.

4. Investigación de los ultimos 3 años aplicando el algoritmo de SA

1. Agosto de 2020. Diseño del controlador de regulación automática de voltaje usando SA hibrido con algoritmos de optimización de Forrajeo de mantarraya. [Paper](#)
2. Enero de 2020. Una nueva técnica híbrida de programación genética basada en SA para predecir la capacidad de carga máxima de las pilas. [Paper](#)
3. Junio de 2018. Optimización de SA evolutivo multiobjetivo para el equilibrado de la línea de desmontaje multirrobótico de modelo mixto con tiempo de procesamiento de intervalo. [Paper](#)

5. Aplicaciones SA

5.1. Knapsack

Modelación Matemática

Sean dos funciones

$$f(x) = \sum_{i=1}^N x_i^I h(x_i)$$

$$g(x) = \sum_{i=1}^N x_i^I p(x_i) \leq peso_maximo$$

Donde:

X es un vector de la forma $X = (x_1, x_2, \dots, x_N)$

X^I es un vector el cual es parecido a X pero sus valores $x_i^I \in [0, 1]$ y $x_i^I \in \mathbb{Z}$

$h(x)$ es una función la cual devuelve el beneficio total de los objetos x_i en la mochila

$p(x)$ es una función la cual devuelve el peso total de los objetos x_i en la mochila

En el cual el algoritmo de SA busca iterar sobre cada contenido o casilla del vector X^I de tal forma que se obtiene una solución y sobre esta se busca encontrar una mejor solución.

Por lo cual el test objetivo del algoritmo es encontrar el optimo global sin que este pueda caer en el optimo local o en otra zona.

5.2. Travel Salesman Problem TSP

Modelación Matemática

Sea la función

$$f(x) = \sum_{i=1}^N \sum_{j=1}^N x_{i,j}^I g(x_{i,j})$$

Donde:

X es una matriz de $N \times N$ la cual trabajara para obtener el costo con la función $g(x)$

X^I es una matriz de $N \times N$ la cual contiene valores que permiten saber si el nodo es considerado o no para la solución

La notación (i, j) significa i como nodo origen que va a j

$x_{i,j}^I$ es un valor de la matriz, donde $x_{i,j}^I \in [0, 1]$ y $x_{i,j} \in \mathbb{Z}$

Si $x_{i,j} = -1$ significa que no hay una conexión de i a j

$g(x)$ es una función la cual devuelve el valor costo de ir de i a j

De tal forma que este algoritmo de SA busca iterar sobre el contenido de una solución cualquiera y sobre cada vértice que este modifique el algoritmo buscara realizar una operación de intercambio de la forma $x_{i,j}$ cambia con $x_{j,i}$ de tal forma que el movimiento sea válido y este modifique la solución.

El test objetivo de esta solución es encontrar una solución de minimización de costo de tal forma que la solución recorra todos los nodos.

5.3. Función de Minimización en D dimensiones

Modelación Matemática

Sea la función

$$f(x) = \sum_{i=1}^N x_i^2$$

Donde:

$$x_i \in \mathbb{R}$$

Descrito en los intervalos $x_i \in [-10, 10]$

Donde sabemos que los puntos mínimos de cada x_i los encontramos cuando el valor asignado a él es $x_i = 0$ por lo tanto el ir variando los valores para que el programa se acerque hacia 0

De tal manera que el algoritmo de SA seleccionará 1 índice en el cual se le asignará un valor a x_i .

En el test objetivo es encontrar 1 punto aproximado en el que la función se aproxime a 0 o en el que al menos una coordenada $x_i \sim 0$

6. Ejecución a mano.

6.1. Knapsack

Knapsack Peso-máximo = 10 peso = [5, 1, 10, 4]
 1. Crear una solución valida beneficio = [4, 2, 5, 3]

$$sol = [1, 0, 0, 1] \Rightarrow \begin{array}{l} \text{Peso total} = 9 \\ \text{Beneficio} = 7 \end{array}$$

$$e_{old} = sol. beneficio$$

$$t = 50$$

$$\alpha = 0.88$$

→ Entra en while t_min ~ 0.01

while $i < \text{sol.length}$

$$\text{sol-height}(\text{sol}) \rightarrow [1, 1, 0, 1] \Rightarrow \begin{array}{l} A = 10 \\ B = 9 \end{array}$$

$$e_{new} = sol. \ beneficio$$

$$\Delta = e_{\text{new}} - e_{\text{old}} \rightarrow \Delta = 2$$

it $\delta > 0$ \rightarrow Verdadero
 $(-\delta, \delta) \rightarrow$ Falso

- en while t-min ~ 0.01

white is soft, bright

$$\text{sol-height(sol)} \rightarrow [1, 1, 0, 1] \Rightarrow \begin{matrix} P = 10 \\ B = 9 \end{matrix}$$

$$A = e \cdot \text{new} = e \cdot \text{sol. beneficio}$$

$$\text{if } \delta t > 0 \rightarrow V_{\perp}$$

if ($\text{rand}(0..1) \geq e^{\lambda} \left(\frac{-\delta t \rho}{t} \right)}$) \rightarrow False

elec

e-old-e new

end

et se

$$e^{-\alpha t} = e^{-\lambda \hbar \omega}$$

end

$$f_+ = 1$$

end

$$t = t * 0.88 \rightarrow 40$$

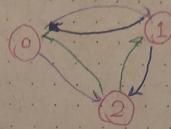
Temps: $\frac{1}{4} [1, 1, 0, 1]$

6.2. TSP

TSP

$$\text{matriz_conexión} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\text{matriz_costo} = \begin{pmatrix} 0 & 3 & 10 \\ 5 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$



$$\begin{aligned} &= 23.8120 \\ &= 78.7090 \\ &78.7090^{\circ} \end{aligned}$$

\downarrow Indices de vertice i a vertice j
solución = [1, 2, 0]

e-old = solución # Costo-total = 16

t = 40

α = 0.9

t-min = 0.001

↳ entra un while

i = 0
while i < sol[0].length

new-sol(solución) → [2, 0, 1] C-total = 6

t = 40

α = 0.9

t-min = 0.001

↳ entra un while

i = 0

while i < sol[0].length

new-sol(solución) → [2, 0, 1] C-total = 6

e-new = solución

Δ = e-new - e-old → Δ = -10

if Δ > 0 → falso

~

else

e-old = e-new

end

i + 1

end

Iteración 1 = [2, 0, 1]

t = t * 0.9 → 36

REDMI NOTE 7
AI DUAL CAMERA

6.3. Función de Minimización en D dimensiones

Función minimización

$$0 = 3 \Rightarrow \text{vector} = [-1, -1, -1]$$

$$\text{solución} = [-7, 5, -4]$$

$$e_{\text{old}} = \text{solución eval} \quad \# \quad 49 + 25 + 16 = 90$$

$$t = 36$$

$$\alpha = 0.95$$

$$t_{\text{min}} = 0.001$$

↳ entra al while

$$i = 0$$

while $i < 0$

$$\text{new_sol}(\text{solución}) \# [-4, 5, -4]$$

$$e_{\text{new}} = \text{solución eval} \quad \# 2 \cdot 16 + 25 = 57$$

$$\Delta = e_{\text{new}} - e_{\text{old}} \# -33$$

if ($\Delta > 0$) → false {

↳ entra al while

$$i = 0$$

while $i < 0$

$$\text{new_sol}(\text{solución}) \# [-4, 5, -4]$$

$$e_{\text{new}} = \text{solución eval} \quad \# 2 \cdot 16 + 25 = 57$$

$$\Delta = e_{\text{new}} - e_{\text{old}} \# -33$$

if ($\Delta > 0$) → false {

{ else }

$$e_{\text{old}} = e_{\text{new}}$$

}

$$i++$$

end

Iteración 1

$$[-4, 5, -4]$$

REDMI NOTE 7
AI DUAL CAMERA end