Un ejemplo de Temple simulado (Simulated Annealing, SA)

Dada la función:

$$f(x) = x^3 - 60x^2 + 900x + 100$$

queremos hallar el máximo de dicha función entre los puntos $x = 0$ y $x = 31$.

A continuación se describe cómo se puede resolver este problema utilizando un algoritmo de temple simulado. Antes de empezar, debemos tener en cuenta que existen diversas versiones sobre este algoritmo, pero todas se basan en el mismo principio: buscar un esquema de "temperatura" que haga que el descenso sea lo suficientemente significativo al principio para permitir "saltos" (siempre gobernados por procesos aleatorios) en el espacio de estados que impidan que nos estanquemos en un mínimo (o máximo) local y poco a poco ir disminuyendo dicho descenso para que podamos encontrar el extremo relativo global.

1) Discretizamos el rango de valores de x con vectores binarios de 5 componentes entre 00000 y 11111. Estos 32 vectores constituyen el conjunto S de las soluciones factibles del problema. Para cada

$i \in S$ definimos $N(i) = \{ j / j$ resulta de cambiar una componente de $i \}$

es decir, $N(i)$ tiene 5 elementos.

Por ejemplo, si i = 10100 (20), los cinco componentes serán:

$N(j = 5)$ = 00100 (4)

$N(j = 4)$ = 11100 (28)

$N(j = 3)$ = 10000 (16)

$N(j = 2)$ = 10110 (22)

$N(j = 1)$ = 10101 (21)

(Nótese que cuanto más significativo es el bit que se cambia, mayor es el salto en el valor de x que figura entre paréntesis)

2) Asignamos **un valor inicial a T** intuitivamente, por ejemplo, T_0 = 100 o 500 y en cada iteración del algoritmo lo reduciremos en 10%[1], es decir, $T_k = 0.9\ T_{k-1}$ (**esquema de enfriado**).

A continuación se presenta una versión general del algoritmo:

---

[1] Puede probarse con otros esquemas de reducción de temperatura todavía más suaves y comprobar la convergencia al óptimo. Se ha encontrado que las reducciones que son más eficaces se encuentran entre el 0.4 y 0.6.

1. i = GeneraUnEstadoInicial()
2. $T = T_0$; $g := 0$
4. **mientras** (CondicionesParoNoActivas(g,T)) **hacer**
3. j = TomaUnVecinoAleatorioDe(i)
4. **si** $f(j) > f(i)$
5. $i = j$
6. **si no**
7. **si** (Aleatorio($0, 1$) < $\exp(f(j) - f(i)/T)$)
8. $i = j$
9. **fin si**
10. **fin si**
11. $g = g + 1$
12. T = Actualiza(g,T)
13. **fin mientras**

El algoritmo de temple simulado trabajará de la siguiente forma:

Elegimos una solución i

Establecemos al azar un valor de j de $N(i)$ y reemplazamos j por i si $f(j) > f(i)$

En caso contrario rechazar j si $f(j) \leq f(i)$ y $\xi > e^{\frac{f(j)-f(i)}{T_k}}$, siendo ξ un número al azar en el intervalo (0,1)

A continuación se muestran los valores de la función según los valores de x, tanto en binario como en decimal:

Binario	Decimal	$f(x)$		Binario	Decimal	$f(x)$
00000	0	100		10000	16	3236
00001	1	941		10001	17	2973
00010	2	1668		10010	18	2692
00011	3	2287		10011	19	2399
00100	4	2804		10100	20	2100
00101	5	3225		10101	21	1801
00110	6	3556		10110	22	1508
00111	7	3803		10111	23	1227
01000	8	3972		11000	24	964
01001	9	4069		11001	25	725
01010	10	4100	(óptimo)	11010	26	516
01011	11	4071		11011	27	343
01100	12	3988		11100	28	212
01101	13	3857		11101	29	129
01110	14	3684		11110	30	100
01111	15	3475		11111	31	131

En la siguiente tabla mostramos cinco iteraciones partiendo de un *T* inicial de 100.

El algoritmo llega a 10000 y no puede salir de la atracción de este valor en las siguientes 50 iteraciones. El valor 10000 (x=16) es un máximo relativo dentro del entorno N(10000). Esto muestra que el valor inicial de *T* es muy bajo para sacarlo de la trampa en que cae partiendo de 10011.

T	**bit**	**vecino(x)**	***f*(vecino)**	**delta *f***	**¿cambiar?[2]**	***x***
						10011
100	1	00011	2287	112	no	10011
90	3	10111	1227	1172	no	10011
81	5	10010	2692	<0	si	10010
73	2	11010	516	2176	no	10010
66	4	10000	3236	<0	si	10000
59	3	10100	2100	1136	no	10000

Y en la siguiente tabla se muestran 9 iteraciones partiendo de un *T* inicial de 500 y se llega al optimo 01010 (x=10) en la iteración 7. En las 150 iteraciones restantes el valor visitado fue el optimo o uno de sus vecinos. Esto muestra la importancia de fijar acertadamente T_0 y el esquema de trabajo.

T	**bit**	**vecino(x)**	***f*(vecino)**	**delta *f***	**¿cambiar?[2]**	***x***
						10011
500	1	00011	2287	112	si	00011
450	3	00111	3803	<0	si	00111
405	5	00110	3556	247	si	00110
364	2	01110	3684	<0	si	01110
328	4	01100	3988	<0	si	01100
295	3	01000	3972	16	si	01000
266	4	01010	4100*	<0	si	01010
239	5	01011	4071	29	si	01011
215	1	11011	343	3728	no	01011

[2] Téngase presente que la decisión de intercambiar valores no sólo depende del valor del incremento entre *f*(*j*) y *f*(*i*) sino también del valor aleatorio de ξ y por tanto, los valores que se obtienen si se reproduce el ejemplo, pueden ser distintos a los mostrados en la tabla.