Recocido Simulado

S. Ivvan Valdez

Centro de Investigación en Matemáticas, CIMAT AC.
Ciencias de la computación

Introducción conceptual

Algoritmos de optimización global que veremos

1. “Algunos” algoritmos para optimización global de funciones no, necesariamente, convexas, con un solo mínimo, etc.
2. Generalmente, de búsqueda directa.
3. No deterministas.

Consideraremos el problema de:

$$min/max \quad f(x), \quad \text{para} \quad x \in \Omega$$

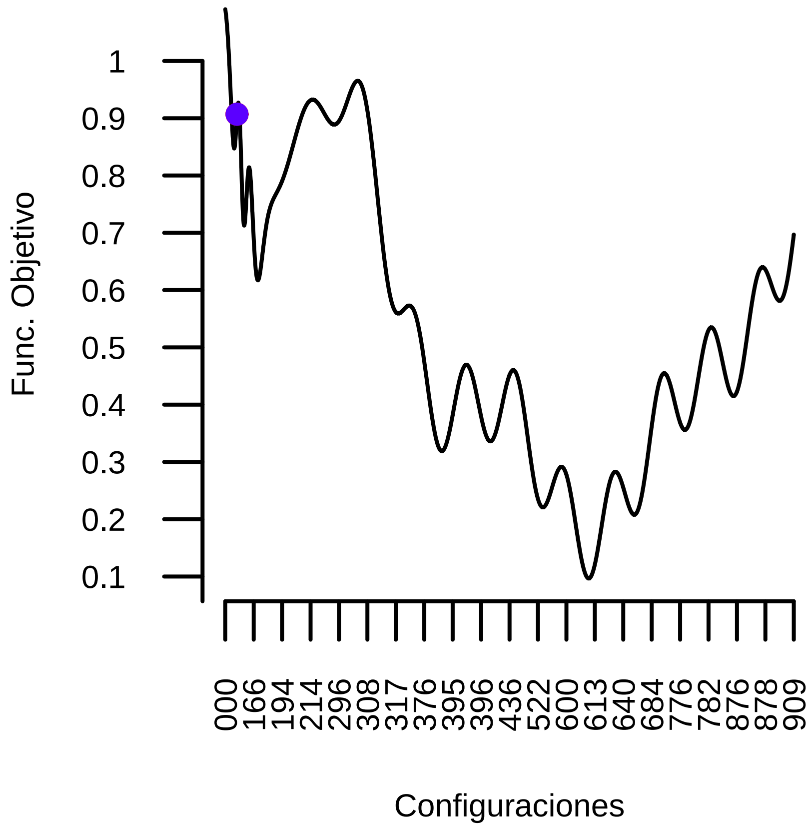
A Ω le denominaremos espacio de búsqueda. Los algoritmos (de esta plática) solo requieren el valor de la evaluación de la función $f(x_i)$, para una posible solución candidata x_i. Es decir, no se requiere que $f(x)$ sea convexa, derivable, y Ω puede ser discreto o continuo, tomaremos de ejemplo (gráfico) un caso discretos.

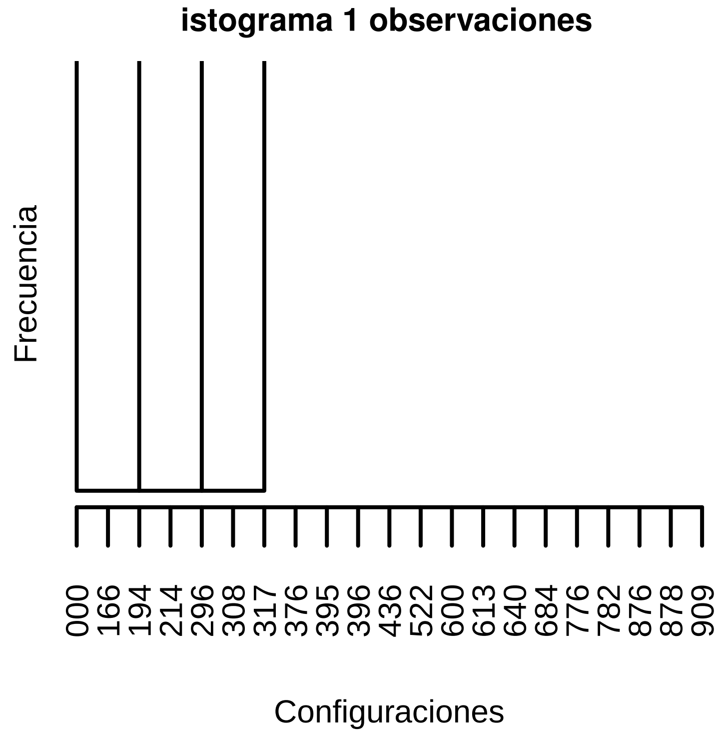
Supongamos que una posible solución al problema es una permutación de N dígitos, ej. $x_i = (0, 9, 2)$ para $N = 3$. Denominaremos a cada posible solución como una *configuración*.
El recocido simulado favorece el muestre de configuraciones con bajo valor de función objetivo con probabilidad:

$$p(x_i) = \frac{e^{-f(x_i)/T}}{Z}, \tag{1}$$

que está relacionada con la distribución de Boltzmann. Evidentemente los estados de menor energía $f(x)$ (función objetivo) les corresponde un mayor valor de probabilidad.

Inspirado por el enfriamiento en los materiales que los lleva a configuraciones estables o de baja energía, Kirkpatrick propone un método para aproximar el mínimo de una función mediante muestreo de una distribución de probabilidad.

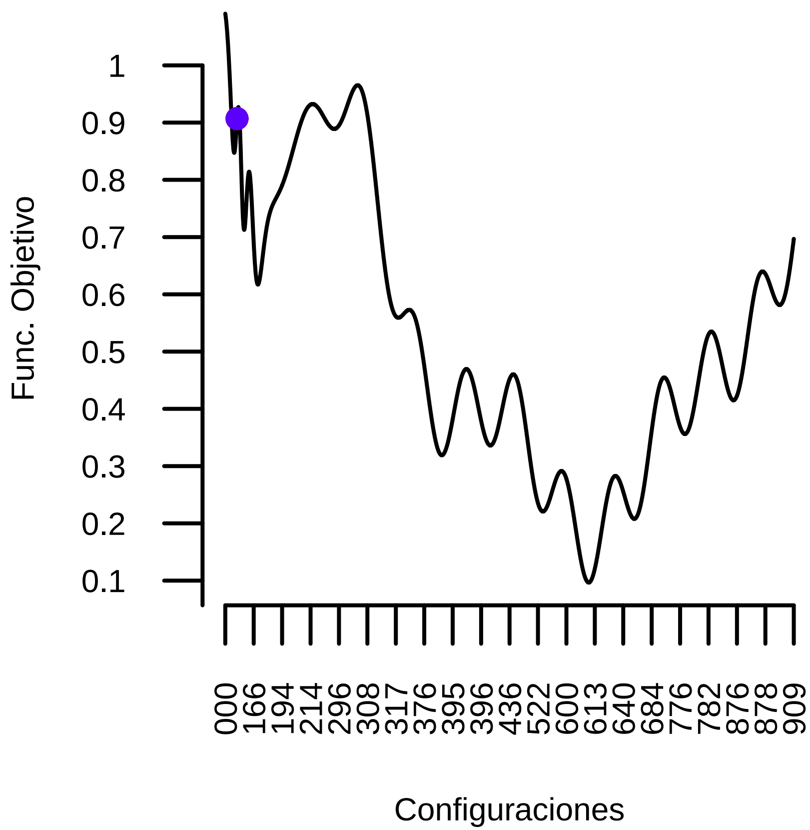


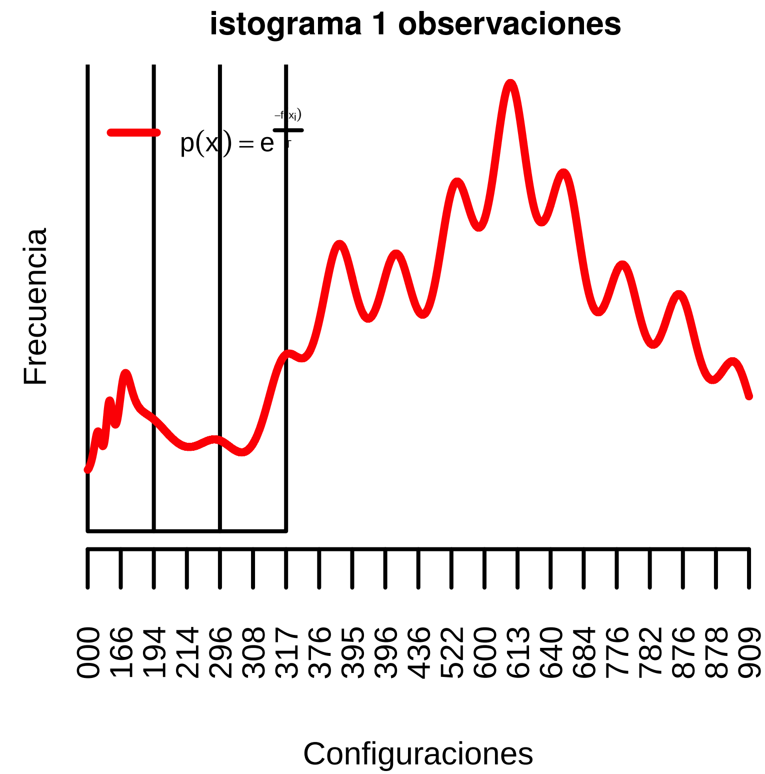


RS-Recocido Simulado (Kirkpatrick 1983)

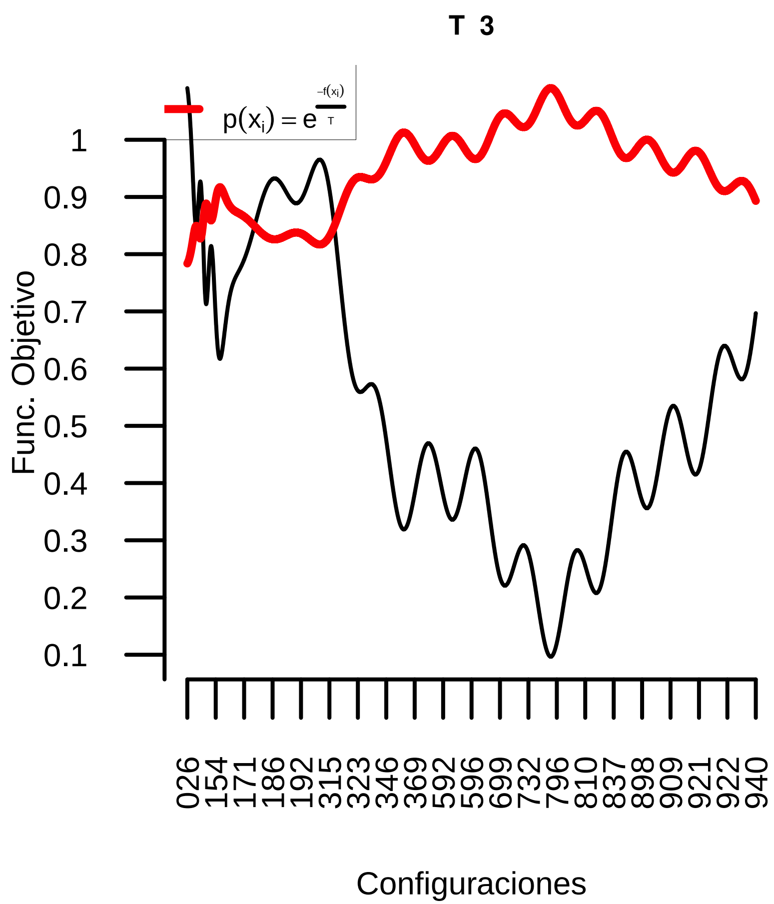
Introducción conceptual
Algoritmos de optimización global que veremos
Problema general
RS-Recocido Simulado (Kirkpatrick 1983)
El Algoritmo
Parámetros y tamaño de paso
Paralelización
Referencias
Tarea

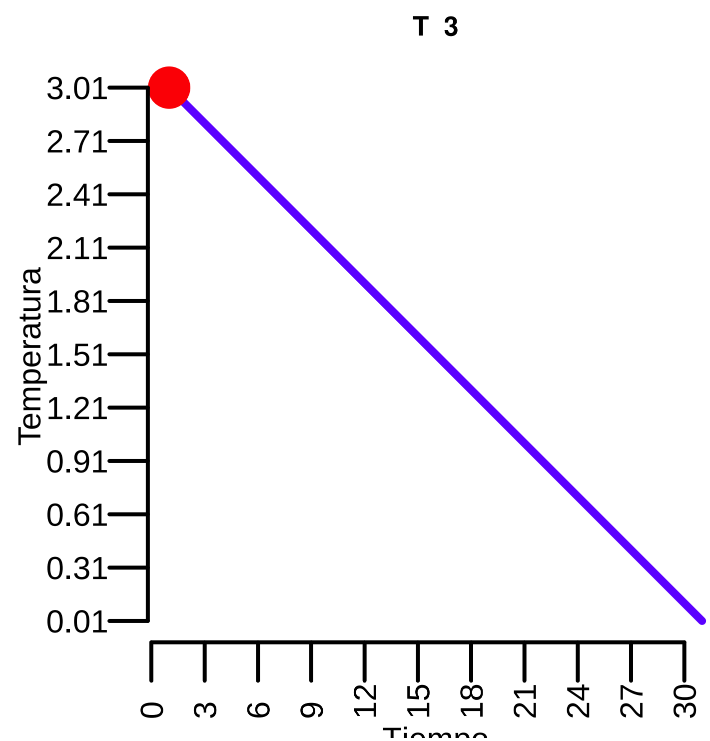
El RS se basa en obtener muestras de $p(x_i) = \frac{e^{-f(x_i)/T}}{Z}$ mediante el algoritmo de Metropolis, el cual solo requiere un valor inicial x_0 y un valor de prueba x_j generado por una distribución conocida, mediante un criterio la nueva configuración x_j pueden ser aceptadas o no, las soluciones aceptadas provienen de $p(x_i) = \frac{e^{-f(x_i)/T}}{Z}$.





Convergencia.
Hasta aquí solamente se hace uso del algoritmo de Metropolis para obtener configuraciones aleatorias de $p(x_i) = \frac{e^{-f(x_i)/T}}{Z}$. La aportación del RS es utilizar un *esquema de enfriamiento.* Es decir, ir modificando el valor de la constante de temperatura para aumentar paulatinamente la probabilidad en la configuración óptima x^*.





- El RS básicamente es un algoritmo que genera observaciones de una distribución de probabilidad $p()$, la cual tiene ciertas características:
  - Si el valor de función objetivo $f(x_i)$ es mejor (menor para minimización) que el de $f(x_j)$ entonces $p(x_i) \geq p(x_j)$.

- El RS básicamente es un algoritmo que genera observaciones de una distribución de probabilidad $p()$, la cual tiene ciertas características:
  - Si el valor de función objetivo $f(x_i)$ es mejor (menor para minimización) que el de $f(x_j)$ entonces $p(x_i) \geq p(x_j)$.
  - Por lo anterior la probabilidad $p(x^*)$ del (los) óptimo x^* es mayor o igual a la de cualquier otra configuración.

- El RS básicamente es un algoritmo que genera observaciones de una distribución de probabilidad $p()$, la cual tiene ciertas características:
  - Si el valor de función objetivo $f(x_i)$ es mejor (menor para minimización) que el de $f(x_j)$ entonces $p(x_i) \geq p(x_j)$.
  - Por lo anterior la probabilidad $p(x^*)$ del (los) óptimo x^* es mayor o igual a la de cualquier otra configuración.
  - La distribución puede ser controlada para que después de cierto tiempo (iteraciones) la masa de probabilidad se concentre en el óptimo global.

- El RS básicamente es un algoritmo que genera observaciones de una distribución de probabilidad $p()$, la cual tiene ciertas características:
  - Si el valor de función objetivo $f(x_i)$ es mejor (menor para minimización) que el de $f(x_j)$ entonces $p(x_i) \geq p(x_j)$.
  - Por lo anterior la probabilidad $p(x^*)$ del (los) óptimo x^* es mayor o igual a la de cualquier otra configuración.
  - La distribución puede ser controlada para que después de cierto tiempo (iteraciones) la masa de probabilidad se concentre en el óptimo global.
- Una configuración de prueba x_j se genera a partir de la configuración actual x^t, las configuraciones aceptadas o “seleccionadas” tienen como distribución subyacente $p(x_i) = \frac{e^{-f(x_i)/T}}{Z}$

- El RS básicamente es un algoritmo que genera observaciones de una distribución de probabilidad $p()$, la cual tiene ciertas características:
  - Si el valor de función objetivo $f(x_i)$ es mejor (menor para minimización) que el de $f(x_j)$ entonces $p(x_i) \geq p(x_j)$.
  - Por lo anterior la probabilidad $p(x^*)$ del (los) óptimo x^* es mayor o igual a la de cualquier otra configuración.
  - La distribución puede ser controlada para que después de cierto tiempo (iteraciones) la masa de probabilidad se concentre en el óptimo global.
- Una configuración de prueba x_j se genera a partir de la configuración actual x^t, las configuraciones aceptadas o “seleccionadas” tienen como distribución subyacente $p(x_i) = \frac{e^{-f(x_i)/T}}{Z}$
- **Es evidente que podemos construir otras distribuciones que tengan un comportamiento similar a la que usa el RS, y también podrían construirse otros algoritmos de búsqueda global utilizando estas distribuciones.**

El Algoritmo

- Se propone un punto: x_0
- Se perturba el punto en una vecindad $x_0 \rightarrow \hat{x}_0$
- Con cierta probabilidad (dependiendo de la evaluación de las funciones objetivo) el nuevo punto reemplaza al anterior.

$\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, ..., x_n)$.
Se desea *minimizar* $f(x)$
$x_i^{inf} < x_i < x_i^{sup}$, para $i = 1..n$.

El criterio de aceptación de Metropolis:

1. Sea $\mathbf{x}_t$ la solución actual y $\hat{\mathbf{x}}$ la solución perturbada.

$\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, ..., x_n)$.

Se desea *minimizar* $f(x)$

$x_i^{inf} < x_i < x_i^{sup}$, para $i = 1..n$.

El criterio de aceptación de Metropolis:

1. Sea $\mathbf{x}_t$ la solución actual y $\hat{\mathbf{x}}$ la solución perturbada.
2. $\Delta f = f(\hat{\mathbf{x}}) - f(\mathbf{x}_t)$.

$\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, ..., x_n)$.
Se desea *minimizar* $f(x)$
$x_i^{inf} < x_i < x_i^{sup}$, para $i = 1..n$.

El criterio de aceptación de Metropolis:

1. Sea $\mathbf{x}_t$ la solución actual y $\hat{\mathbf{x}}$ la solución perturbada.
2. $\Delta f = f(\hat{\mathbf{x}}) - f(\mathbf{x}_t)$.
3. **If** $(\Delta f(\mathbf{x}) \leq 0$) **then** $\mathbf{x}_{t+1} = \hat{\mathbf{x}}$.

$\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, ..., x_n)$.
Se desea *minimizar* $f(x)$
$x_i^{inf} < x_i < x_i^{sup}$, para $i = 1..n$.

El criterio de aceptación de Metropolis:

1. Sea $\mathbf{x}_t$ la solución actual y $\hat{\mathbf{x}}$ la solución perturbada.
2. $\Delta f = f(\hat{\mathbf{x}}) - f(\mathbf{x}_t)$.
3. **If** $(\Delta f(\mathbf{x}) \leq 0\,)$ **then** $\mathbf{x}_{t+1} = \hat{\mathbf{x}}$.
4. **else** El nuevo punto se acepta con probabilidad $\exp(\frac{-\Delta f}{T})$.

$\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, ..., x_n)$.
Se desea *minimizar* $f(x)$
$x_i^{inf} < x_i < x_i^{sup}$, para $i = 1..n$.

El criterio de aceptación de Metropolis:

1. Sea $\mathbf{x}_t$ la solución actual y $\hat{\mathbf{x}}$ la solución perturbada.
2. $\Delta f = f(\hat{\mathbf{x}}) - f(\mathbf{x}_t)$.
3. **If** $(\Delta f(\mathbf{x}) \leq 0$) **then** $\mathbf{x}_{t+1} = \hat{\mathbf{x}}$.
4. **else** El nuevo punto se acepta con probabilidad $\exp(\frac{-\Delta f}{T})$.

Una forma de realizar el paso en 4 es: $u \sim U(0, 1)$, **if** $u < \exp(\frac{-\Delta f}{T})$ se acepta.

$\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, ..., x_n)$.
Se desea *minimizar* $f(x)$
$x_i^{inf} < x_i < x_i^{sup}$, para $i = 1..n$.

El criterio de aceptación de Metropolis:

1. Sea $\mathbf{x}_t$ la solución actual y $\hat{\mathbf{x}}$ la solución perturbada.
2. $\Delta f = f(\hat{\mathbf{x}}) - f(\mathbf{x}_t)$.
3. **If** $(\Delta f(\mathbf{x}) \leq 0)$ **then** $\mathbf{x}_{t+1} = \hat{\mathbf{x}}$.
4. **else** El nuevo punto se acepta con probabilidad $\exp(\frac{-\Delta f}{T})$.

Una forma de realizar el paso en 4 es: $u \sim U(0,1)$, **if** $u < \exp(\frac{-\Delta f}{T})$ se acepta.

T es un parámetro denominado temperatura. Con un valor fijo de T la sucesión $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_i, ...$ no es decreciente. Para valores grandes de T (comparados con las diferencias de la función objetivo), los puntos aceptados son practicamente aleatorios.

$\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, ..., x_n)$.
Se desea *minimizar* $f(x)$
$x_i^{inf} < x_i < x_i^{sup}$, para $i = 1..n$.

El criterio de aceptación de Metropolis:

1. Sea $\mathbf{x}_t$ la solución actual y $\hat{\mathbf{x}}$ la solución perturbada.
2. $\Delta f = f(\hat{\mathbf{x}}) - f(\mathbf{x}_t)$.
3. **If** ($\Delta f(\mathbf{x}) \leq 0$) **then** $\mathbf{x}_{t+1} = \hat{\mathbf{x}}$.
4. **else** El nuevo punto se acepta con probabilidad $\exp(\frac{-\Delta f}{T})$.

Una forma de realizar el paso en 4 es: $u \sim U(0,1)$, **if** $u < \exp(\frac{-\Delta f}{T})$ se acepta.

T es un parámetro denominado temperatura. Con un valor fijo de T la sucesión $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_i, ...$ no es decreciente. Para valores grandes de T (comparados con las diferencias de la función objetivo), los puntos aceptados son practicamente aleatorios.

El recocido simulado comienza con una temperatura alta definida por el usuario y se genera una secuencia de puntos hasta que se alcanza cierto equilibrio.

Algorithm 1: Recocido simulado.

Input: $T_0 =$ Temperatura (muy alta inicialmente).

$t = 0$ Se propone un punto inicial $\mathbf{x}_{best} = \mathbf{x}_0 = [x_1, x_2, ..., x_n]$;
Se evalua e inicializa el mejor $f_{best} = f(\mathbf{x}_t)$;
$\hat{\mathbf{x}} = \mathbf{x}_t$;
while *Mientras no se satisfaga el criterio de paro* **do**
    for $Ajustes\_de\_T = 1...N_T$ **do**
        for $No.Ciclos = 1..N_s$ **do**
            for $i = 1..n$ **do**
                $e_i \sim U(-v_i, v_i)$;
                while $(\hat{x}_i + e_i) < x_i^{inf} \vee (\hat{x}_i + e_i) > x_i^{sup}$ **do**
                    $e_i \sim U(-v_i, v_i)$;
                Evaluar $\hat{\mathbf{x}} = (\mathbf{x}_t + [0, 0, ...e_i, ..0])$;
                if $f(\hat{\mathbf{x}}) < f_{best}$ **then**
                    $f_{best} = f(\hat{\mathbf{x}})$;
                    $\mathbf{x}_{best} = f(\hat{\mathbf{x}})$;
                Aceptar el nuevo punto $\hat{\mathbf{x}}$ de acuerdo con el criterio de Metropolis.;
        Ajustar el tamaño del vector de paso v;
    Reducir la temperatura T;
    Actualizar $\mathbf{x}_t = \mathbf{x}_{best}$ y $f(\mathbf{x}_t) = f_{best}$;

Parámetros y tamaño de paso

Cálculo del tamaño de paso

N_s =Número de pasos antes de recalcular.

n_i =Número de veces que un vector fue aceptado (en el paso del criterio de metropolis).

c_i =Constante de usuario (se recomienda igual a 2.0).

if$(n_i > 0.6N_s)$

$$v_i = v_i \left(1.0 + c_i \frac{n_i/N_s - 0.6}{0.4}\right)$$

if$(n_i < 0.4N_s)$

$$v_i = v_i / \left(1.0 + c_i \frac{0.4 - n_i/N_s}{0.4}\right)$$

De otra forma: No se cambia v_i

$$T = r_t T$$

Existen otros *métodos de enfriamiento*, este solo es el mas común.

Valores sugeridos de los parámetros:

$N_s = 20$ Número de pasos antes de recalcular el tamaño de paso.

Para el paralelo: $N_s = integer\left(N_s/(no\ de\ procesos)\right)$ o 1 si $no\ de\ procesos > N_s^{serial}$.

$N_T = max(100, 5n)$ siendo n el número de variables.

$c_i = 2$ para $i = 1..n$

$N_\epsilon = 4$ Para criterio de paro. (número de veces que el valor local $|f(x_t) - f_{best}| \leq \epsilon$

$r_T = 0.85$ Para reducción de la temperatura

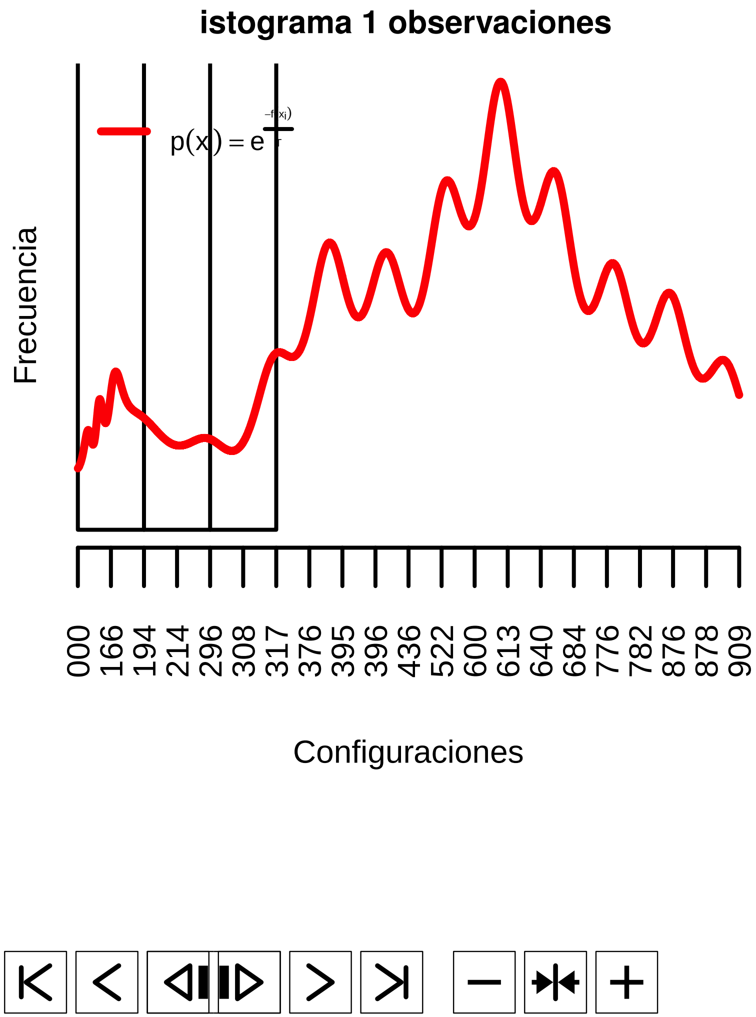
Paralelización

- Se paraleliza el ciclo de N_s y el algoritmo paralelo hace $N_s/(numero - de - procesos)$ de ciclos.
- Cada proceso tiene su propio $\hat{\mathbf{x}}$, $\mathbf{x}_{best}$ y $\mathbf{x}_t$ y su contador n_i.
- El $\mathbf{x}_{best}$ global se actualiza cuando se actualizan los tamaños de paso, y el $\mathbf{x}_t$ de cada proceso se actualiza también.

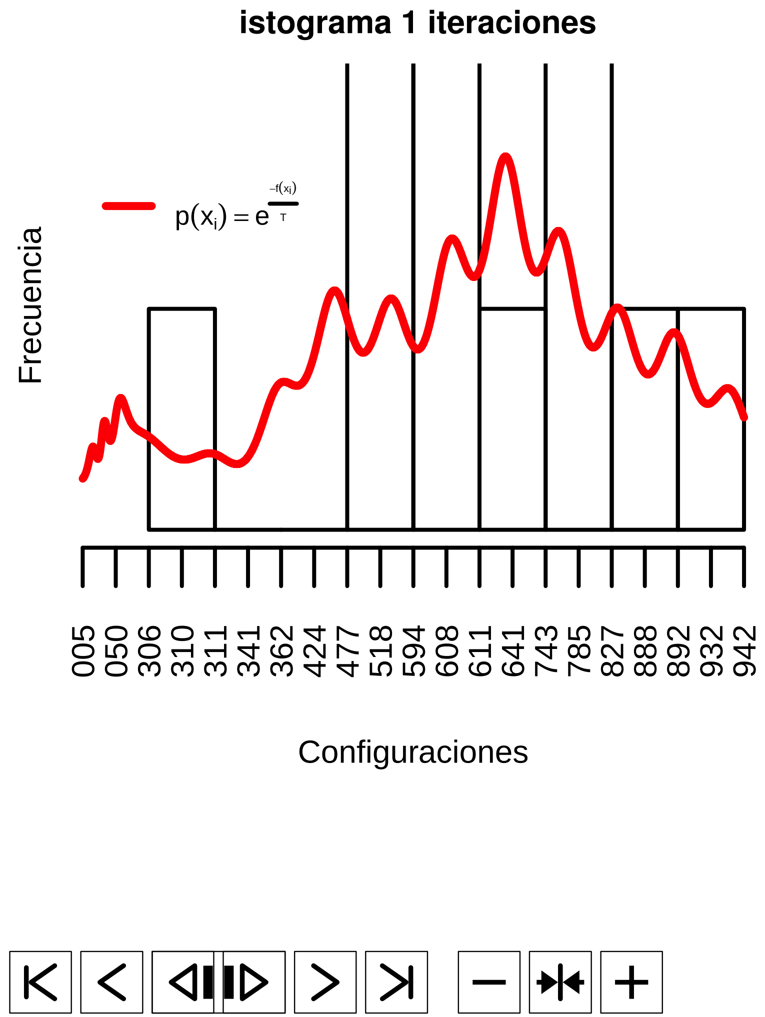
Referencias

Tarea

- Programar el generador hibrido de aletorios (o copiar y pegar las funciones prácticamente), de: Efficient pseudo-random number generation for monte-carlo simulations using graphic processors.
- Programar algoritmo serial y paralelo. Del paralelo programar dos versiones.
  - Usando critical/atomic y barriers donde tenga que sincronizar.
  - Asíncrono usando locks.
- Los problemas de prueba:
  - dos funciones unimodales del CEC2008: Shifted Sphere y Shifted Schwefel. Ejecutar con 10 variables y con 0 corrimiento la función de esfera (hasta que vean que entrega algo cercano al óptimo) y cuando funcione ejecutar para 30 variables y reportar esos resultados en su tarea.
  - El problema de minimización del potencial de Lenard-Jones del CEC2011. (Agregar una grafica de mejor solución de este problema en su reporte).
- Para 15 ejecuciones calcular:
  - Media y desviación estándar del valor de funcion objetivo de la aproximación al óptimo, pot todas las versiones.
  - Media y desviación estándar del tiempo.
  - Media y desviación estándar del SpeedUp.



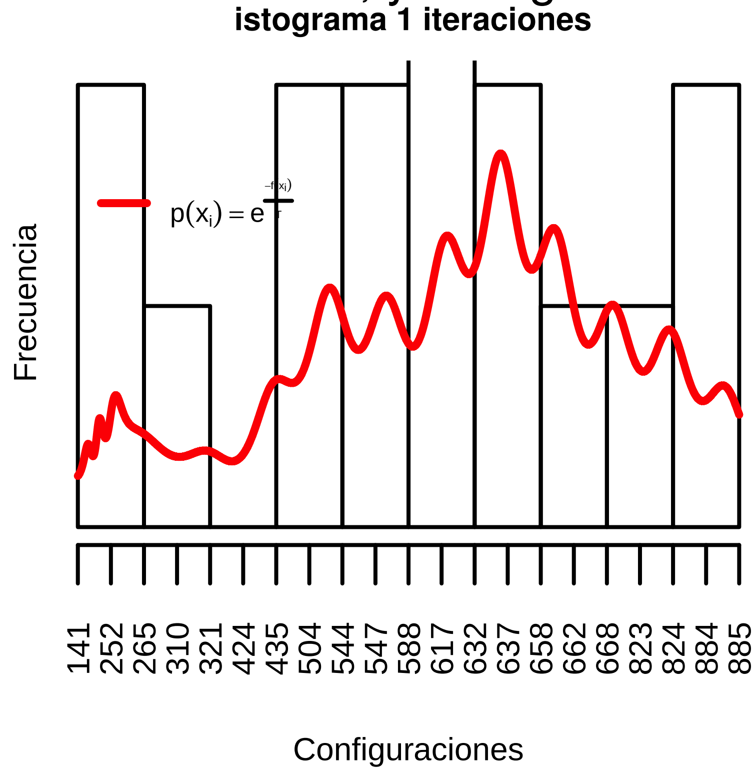
Serial



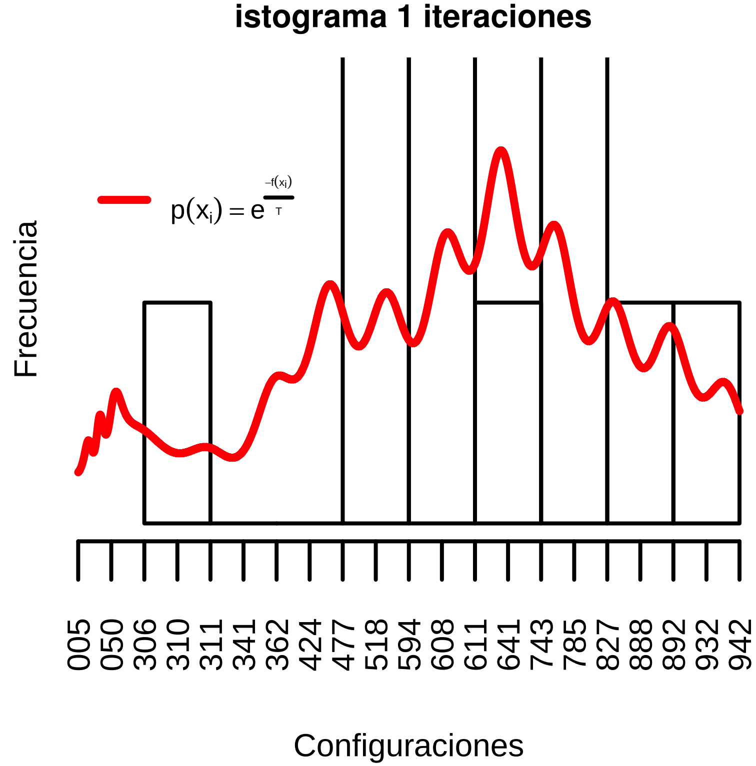
Paralelo

paralelo 10 vs paralelo 20

La primera version (paralelo 10 comparte información de la mejor solución cada 10 iteraciones, y la segunda cada 20.



Paralelo 10



Paralelo 20

[1] **Algoritmo Serial**
A. Corana, M. Marchesi and S. Ridella Minimizing Multimodal Functions of Continuous Variables with the “Simulated Annealing” Algorithm. *ACM Transactions on Mathematical Software*, 13(3):262 – 280,1987.

[2] **Algoritmo Paralelo con MPI**
J.S. Higginson, R.R. Neptune and F.C. Anderson Simulated parallel annealing withind the neighborhood for optimization of biomechanical systems. *Journal of Biomechanics*, 38(2005):1938 – 1942, 2005.

[3] **Aqui viene el potencial de Lenard-Jones**
Das, Swagatam, and P. N. Suganthan. Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems. Jadavpur Univ., Nanyang Technol. Univ., Kolkata, India, 2010.

[4] **Aqui vienen el problema de la esfera y el otro**
Tang, Ke, Xn Yo, Ponnuthurai Nagaratnam Suganthan, Cara MacNish, Ying-Ping Chen, Chih-Ming Chen, and Zhenyu Yang. Benchmark functions for the CEC2008 special session and competition on large scale global optimization. Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.

[5] **Generador de aletorios, acuerdense que cada hilo accede a sus propias semillas y variables**
Mohanty, Siddhant, A. K. Mohanty, and F. Carminati. Efficient pseudo-random number generation