ENXEÑARÍA DO COÑECEMENTO

4º Grao Enxeñaría Informática

4.4 MH basadas en trayectorias: SA
Curso 2016-17

Alberto J. Bugarín Diz
Departamento de Electrónica e Computación
Universidade de Santiago de Compostela
alberto.bugarin.diz@usc.es

4.4 SA. Introducción

- El **Enfriamiento, Temple** o **Recocido Simulado** (*Simulated Annealing*, SA) es un algoritmo de búsqueda por entornos con un **criterio probabilístico** de **aceptación de soluciones** inspirado en la Termodinámica.

Science 13 May 1983:
Vol. 220 no. 4598 pp. 671-680
DOI: 10.1126/science.220.4598.671

‹ Prev | Table of Contents | Next ›

Optimization by Simulated Annealing

S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi

+ Author Affiliations

ABSTRACT

There is a deep and useful connection between statistical mechanics (the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature) and multivariate or combinatorial optimization (finding the minimum of a given function depending on many parameters). A detailed analogy with annealing in solids provides a framework for optimization of the properties of very large and complex systems. This connection to statistical mechanics exposes new information and provides an unfamiliar perspective on traditional optimization problems and methods.

4.4 SA. Introducción

- Como estrategia para evitar que la búsqueda local finalice en óptimos locales, permiten movimientos hacia **soluciones peores**.
- Si la búsqueda está **avanzando** realmente hacia una buena solución, estos movimientos "**de escape de óptimos locales**" deben realizarse de un **modo controlado**

- En el Temple Simulado esto se realiza controlando la frecuencia de los movimientos de escape mediante una función de probabilidad que hará **disminuir la probabilidad de estos movimientos hacia soluciones peores conforme avanza la búsqueda**

- Se aplica la filosofía habitual de búsqueda de **diversificar al principio** e **intensificar al final**

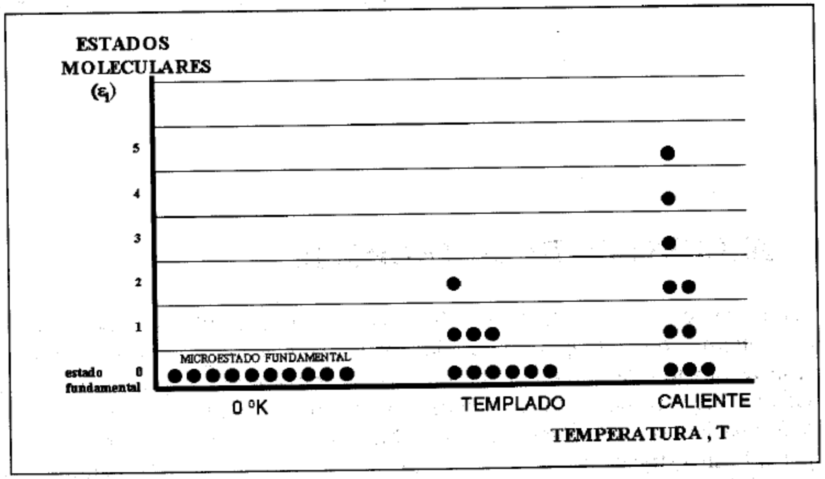
4.4 SA. Introducción

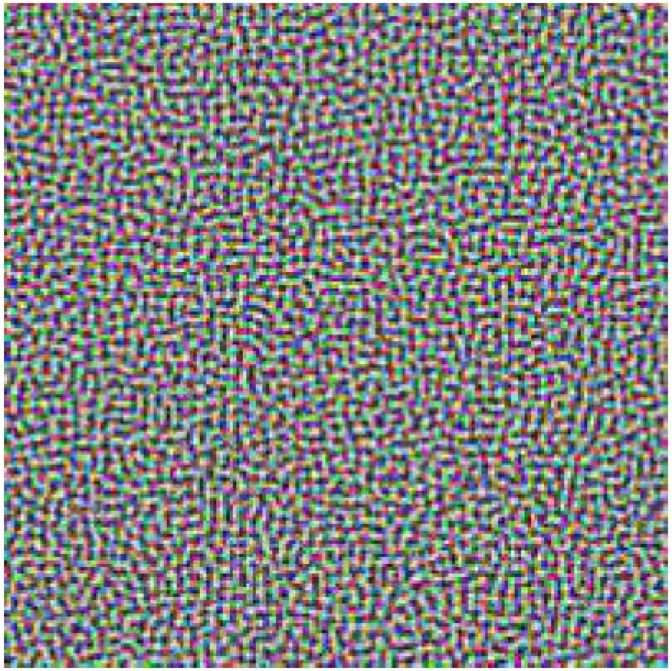
- N. Metropolis (1953): modelado del proceso de enfriamiento simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable (congelado). Las leyes de la termodinámica dicen que a una temperatura **T** la probabilidad de un incremento energético de magnitud **δE** se puede aproximar por:

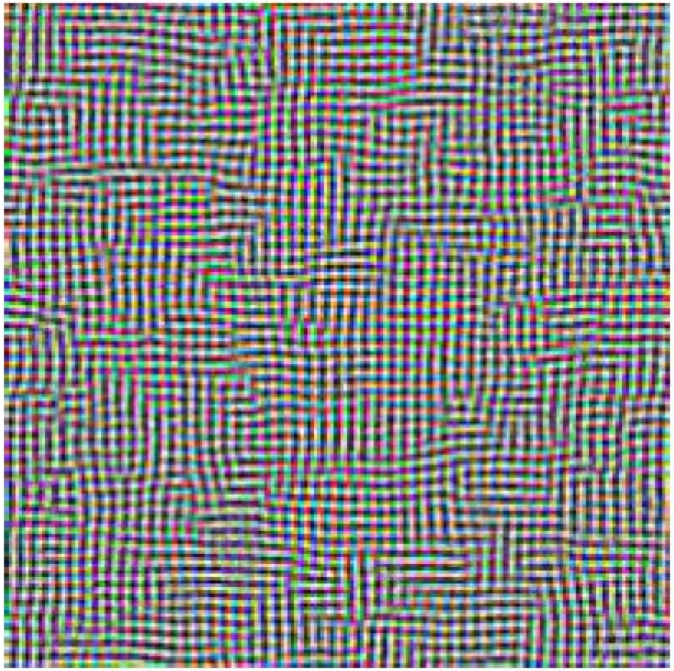
$$[\delta E] = e^{-\frac{\delta E}{kT}},$$ k constante de Boltzmann

- Se genera una **perturbación aleatoria** en el sistema y se calculan los cambios de energía resultantes:
  - si hay una **caída energética**, el **cambio se acepta** automáticamente
  - por el contrario, si se produce un **incremento energético**, el cambio será **aceptado con una probabilidad** indicada por la anterior expresión

4.4 SA. Introducción







4.4 SA. Introducción

- Analogías: termodinámica – optimización

Simulación Termodinámica	Optimización Combinatoria
Estados del sistema	Soluciones factibles
Energía	Coste
Cambio de estado	Solución en el entorno
Temperatura	Parámetro de control
Estado congelado	Solución heurística

S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol 220, Number 4598, pages 671-680, 1983

- Hace uso de una variable llamada **Temperatura**, T, cuyo valor determina en qué medida pueden ser **aceptadas soluciones vecinas peores** que la actual
- La variable Temperatura se **inicializa a un valor alto**, denominado Temperatura inicial, T0, y se **va reduciendo** en cada iteración mediante un mecanismo de enfriamiento de la temperatura, α, hasta alcanzar una **Temperatura final**, Tf

4.4 SA. Algoritmo básico

- En **cada iteración** se genera un **número concreto de vecinos**, $L(T)$, que puede ser fijo para toda la ejecución o depender de la iteración concreta

- Cada vez que se genera un vecino, se aplica el **criterio de aceptación** para ver si sustituye a la solución actual
  - Si la solución vecina es **mejor** que la actual, se **acepta automáticamente**, tal como se haría en la búsqueda local clásica
  - En cambio, si es **peor**, aún existe la **probabilidad de que el vecino sustituya** a la solución actual. Esto permite al algoritmo salir de óptimos locales, en los que la Búsqueda Local clásica quedaría atrapada

- Esta probabilidad depende de la **diferencia de costes** δ entre la solución vecina y actual y de la temperatura *T*:

$$\delta = C(s_{vecina}) - C(s_{actual})$$

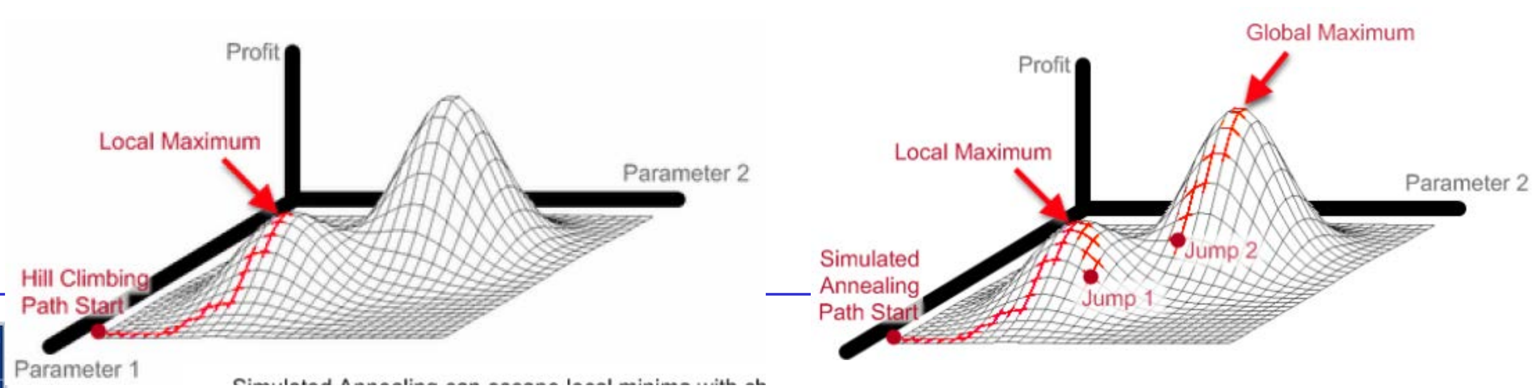
$$P_{aceptación} = e^{\frac{-\delta}{T}}$$

$$\delta < 0 \Leftrightarrow vecino\ mejor$$

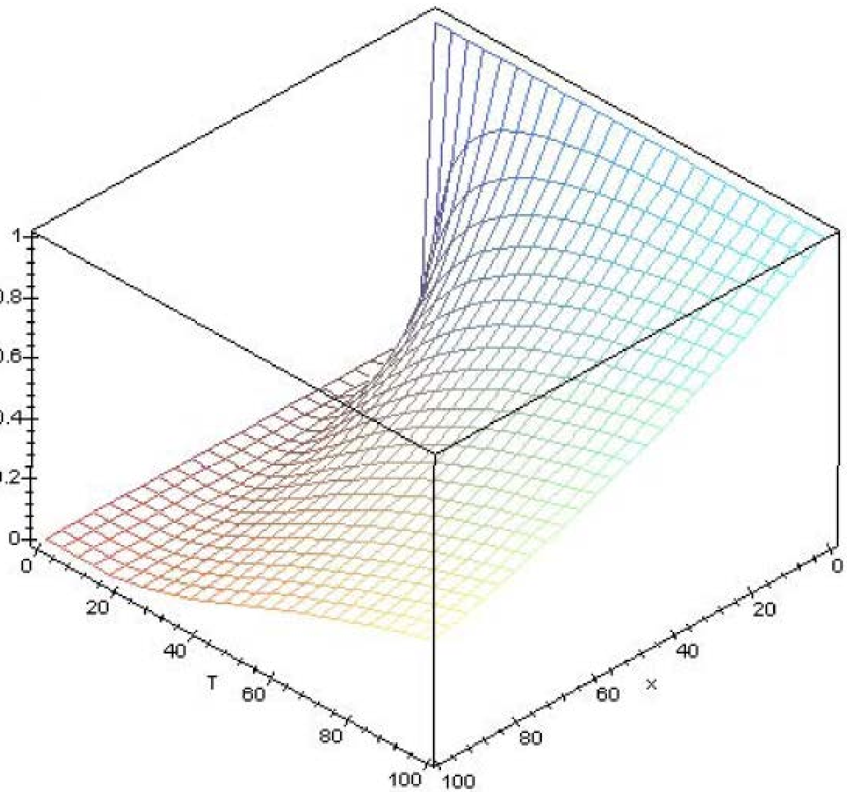
$$\delta > 0 \Leftrightarrow vecino\ peor$$

  - A mayor temperatura, mayor probabilidad de aceptación de soluciones peores. Así, el algoritmo acepta **soluciones mucho peores que la actual al principio de la ejecución** (exploración) pero **no al final** (explotación)
  - A **menor diferencia de costes**, **mayor probabilidad de aceptación** de soluciones peores: soluciones "no mucho peores".

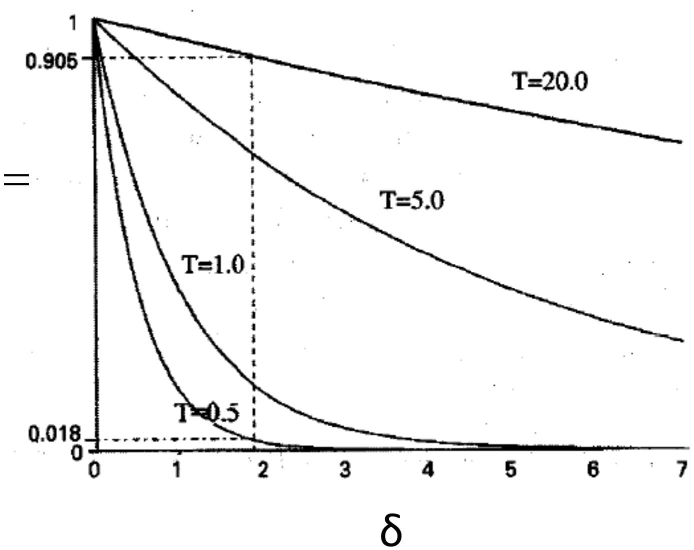
- Una vez finalizada la iteración, es decir, tras generar *L*(*T*) soluciones vecinas, se enfría la temperatura y se pasa a la siguiente iteración



P = exp(-x/t)



$$P_{aceptación} = = e^{\frac{-\delta}{T}}$$



- Caso: Para un valor fijo de diferencia de coste δ=2, la probabilidad de aceptar un movimiento se reduce con T. En un factor 0.905/0.018=50 al pasar de T=20 a T=0.5.

- ALGORITMO BÁSICO (minimizar coste)

$$\delta = C(s_{vecina}) - C(s_{actual})$$

$$\delta < 0 \Leftrightarrow vecino\ mejor$$

$$\delta > 0 \Leftrightarrow vecino\ peor$$

```
INPUT (T_o,α,L,T_f)
T←T_o
S_act←Genera_solucion_inicial
WHILE T≥T_f DO
      FOR cont←1 TO L(T) DO
            S_cand←Selecciona_solucion_N(S_act)
            δ←coste(S_cand)−coste(S_act)
            IF (U(0,1)<e^(-δ/T)) OR
               (δ<0)   THEN S_act←S_cand
            END
      END
      T←α(T)
END
{Escribe como solución, la mejor de las S_act visitadas}
```

4.4 SA. Parámetros y componentes

- REPRESENTACIÓN:
  - Vector ordenado de enteros: TSP (rep. Orden)
  - Vector binario: mochila {0, 1}
  - Vector reales: optimización continua

- SOLUCIÓN INICIAL: técnicas eficientes, conocimiento
- TRANSICIÓN DE SOLUCIONES:
  - 1. Generación de una nueva solución:
    - Definición de N(S)
    - Selección de una S_candidata
  - 2. Cálculo de diferencia de costes
  - 3. Aplicación del criterio de aceptación

- TRANSICIÓN DE SOLUCIONES: movimientos

INPUT (T_o,α,L,T_f)

$T \leftarrow T_o$

$S_{act} \leftarrow$ Genera_solucion_inicial

WHILE $T \geq T_f$ DO

  FOR cont $\leftarrow$ 1 TO L(T) DO

    $S_{cand} \leftarrow$ Selecciona_solucion_N(S_{act})

    $\delta \leftarrow coste(S_{cand}) - coste(S_{act})$

    IF $(U(0,1) < e^{(-\delta/T)})$ OR

      $(\delta < 0)$ THEN $S_{act} \leftarrow S_{cand}$

    END

  END

  $T \leftarrow \alpha(T)$

END

{Escribe como solución, la mejor de las S_{act} visitadas}

Generación de una nueva solución

Cálculo de la diferencia de costos

Aplicación del criterio de aceptación

- SECUENCIA DE ENFRIAMIENTO: 4 parámetros a definir

```
INPUT (To,α,L,Tf)
T←To                                          → Valor inicial del parámetro de control
Sact←Genera_solucion_inicial
WHILE T≥Tf DO                                 → Condición de Parada
      FOR cont←1 TO L(T) DO                   → Velocidad de enfriamiento L(T)
            Scand←Selecciona_solucion_N(Sact)
            δ←coste(Scand)−coste(Sact)
            IF (U(0,1)<e^(−δ/T)) OR
               (δ<0)   THEN Sact←Scand
            END
      END
      T←α(T)                                  → Mecanismo de enfriamiento
END
{Escribe como solución, la mejor de las Sact visitadas}
```

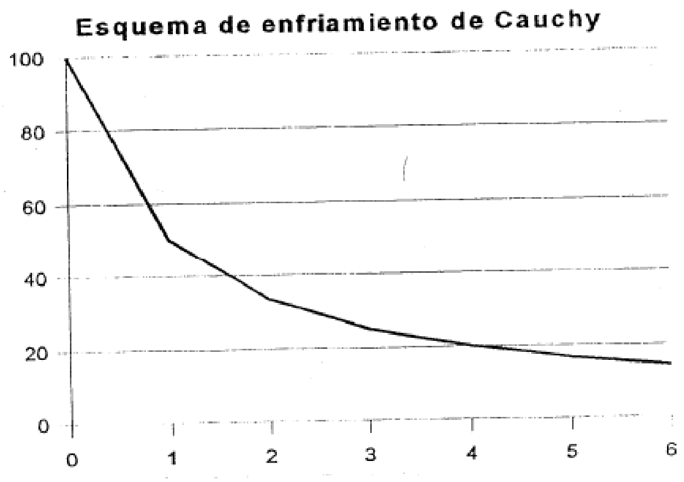
1. **Valor inicial** del parámetro de control (temperatura)
   - No parece conveniente considerar valores fijos independientes del problema:
     - Si es muy baja, no alcanza el óptimo
     - Si es muy alta, el algoritmo tarda en converger
   - Debe permitir que inicialmente sea aceptada casi cualquier transición:
     - Ajuste experimental: valor inicial que permita libertad en los primeros movimientos
     - Proporcional al:
       - Tamaño del problema
       - Tamaño del espacio de soluciones
       - En general, que $\frac{Núm.\,mvtos.\,aceptados\ en\ T_0}{Núm.\,total\ mvtos} \approx 1 \qquad T_0 >> \delta$
       - Depende de la probabilidad ϕ de que una solución sea un μ% peor que S0

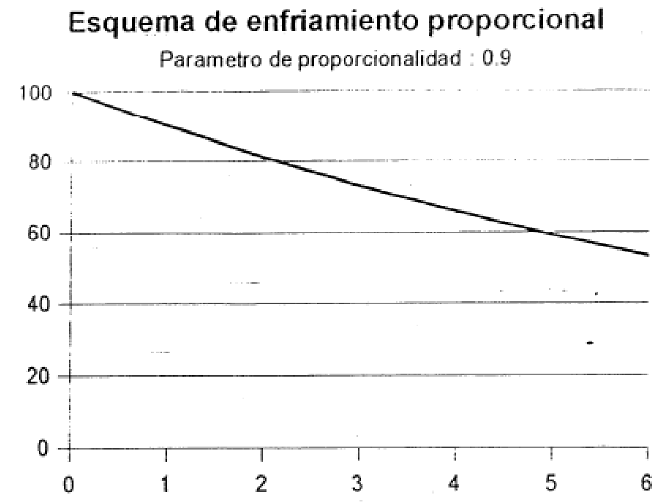
$$T_0 = (\mu / -\ln(\phi)) \cdot C(S_0)$$

4.4 SA. Parámetros y componentes

2. MECANISMO DE ENFRIAMIENTO:
   - Enfriamiento basado en sucesivas temperaturas descendentes fijadas por el usuario
   - Enfriamiento con descenso constante de temperatura
   - Descenso exponencial: $T_{k+1} = \alpha \cdot T_k$ donde k = nº iteración actual, y α cercana a 1 (usualmente [0.8, 0.99])
   - Criterio de Boltzmann: $T_k = T_0 / (1 + \log(k))$
   - Esquema de Cauchy: $T_k = T_0 / (1 + k)$
   - Para controlar el número de iteraciones (Cauchy modificado):
     - $T_{k+1} = T_k / (1 + \beta \cdot T_k)$
     - Para ejecutar exactamente M iteraciones: $\beta = (T_0 - T_f) / (M \cdot T_0 \cdot T_f)$

- MECANISMO DE ENFRIAMIENTO





4.4 SA. Parámetros y componentes

- VELOCIDAD DE ENFRIAMIENTO
  - L(T) debe ser suficientemente grande como para que el sistema llegue a alcanzar su **estado estacionario** para esa temperatura
  - Lo habitual es que sea un valor fijo, pero hay una variante que permite decidir mejor cuando finalizar la iteración actual y enfriar
  - Consiste en enfriar cuando se dé una de las dos situaciones siguientes:
    - Se ha **generado** un número máximo de vecinos (*máx_vecinos).*
    - Se han **aceptado** un número máximo de vecinos (*máx_éxitos).*
  - Lógicamente, *máx_vecinos* tiene que ser mayor que *máx_éxitos.* Una buena proporción puede ser *máx_éxitos=0.1*máx_vecinos*

4.4 SA. Parámetros y componentes

- CONDICIÓN DE PARADA: En teoría, el algoritmo debería finalizar cuando T=0. En la práctica, se para:
  - cuando T alcanza un valor final Tf, fijado previamente
  - después de un número fijo de iteraciones
  - o no hay mejora significativa (o ninguna) tras varias iteraciones
- Como es difícil dar valor de Tf, se suele usar un **número fijo de iteraciones**
- Una buena opción es **parar cuando no se haya aceptado ningún vecino de los L(T) generados en la iteración actual** (*num_éxitos=0)*
- En ese caso, es muy probable que el algoritmo se haya estancado y no vaya a mejorar la solución obtenida
- Combinando este criterio de parada y la condición de enfriamiento de los *máx_vecinos* y *máx_éxitos* se obtiene un equilibrio en la búsqueda que evita malgastar recursos

4.4 Referencias

- F. Herrera (UGR) y col. Algorítmica. Tema 3.
- K.A. Dowsland, B.A. Díaz. Diseño de Heurísticas y Fundamentos del Recocido Simulado. Revista Iberoamericana de Inteligencia Artificial 19 (2003) 93-102. erevista.aepia.org
- J.T. Palma, R. Marín. Inteligencia Artificial: técnicas, métodos y aplicaciones. Ed. McGraw-Hill, 2008. Cap. 9.7.
- D. Henderson, S.H. Jacobson, A.W. Johnson. Chapter 10: The Theory and Practice of Simmulated Annealing. In: F. Glover, G.A. Kochenberber, (Eds.). Handbook of Metaheuristics. Kluwer Academics. (2003) 287-319.