Universidad Carlos iii de Madrid

Escuela Politécnica Superior

Departamento de
Teoría de la Señal y Comunicaciones

PROYECTO FIN DE CARRERA

OPTIMIZACIÓN ESTOCÁSTICA MEDIANTE MÉTODOS DE MONTE CARLO

Alberto Serrano Cádiz

Abril de 2011

Proyecto Fin de Carrera

Optimización estocástica mediante métodos de Monte Carlo

Autor

Alberto Serrano Cádiz

Tutor

Luca Martino

La defensa del presente Proyecto Fin de Carrera se realizó el día 15 de Abril de 2011, siendo evaluada por el siguiente tribunal:

Presidente: David Luengo García

Secretario: Efrain Tito Mayhua López

Vocal: Juan Carlos Torres Zafra

y habiendo obtenido la siguiente Calificación:

Leganés, a 15 de Abril de 2011

Agradecimientos

En primer lugar me gustaría agradecer a mi GRAN tutor y amigo **Luca Martino** todo el apoyo mostrado. La ilusión y ganas que ha puesto en este trabajo han sido inigualables.

Me gustaría dar las gracias a todas aquellas personas que me han dado fuerzas para finalizar este proyecto. Solamente mis padres saben el trabajo que ha costado confeccionar este escrito, y es por ello que les doy las gracias de todo corazón.

También quiero agradecer su apoyo a todos mis amigos. Sin sus ánimos, habría tirado la toalla hace mucho tiempo. Mi más sincera gratitud para Verónica, Cristina, Jose, Carlos, Jaime, Sete, Patricio, Paco, Jorge, Rebeca, Dary, Suraya, Torres, Farruco, Merce, Diego y muchos otros.

GRACIAS

Índice general

Lista de Figuras

Lista de Tablas

Nunca consideres el estudio como un deber, sino como una oportunidad para penetrar en el maravilloso mundo del saber.

Albert Einstein

Capítulo 1

Notación

1.1. Vectores, matrices, puntos e intervalos

Las magnitudes escalares se denotarán usando letras regulares (por ejemplo x o X), mientras que para los vectores $\mathbf{x}$ y las matrices $\mathbf{X}$ se utilizarán letras resaltadas en negrita. Las componentes de un vector n-dimensional se denotarán con corchetes en la forma $\mathbf{x} = [x_1, ..., x_n]$. En ocasiones, resulta conveniente interpretar $\mathbf{x}$, como un punto del espacio, por lo que, cuando sea necesario, se enfatizará esta representación con la notación alternativa $\mathbf{x} = (x_1, ..., x_n)$.

Se usa una notación similar para los intervalos en tiempo real. De forma específica, para dos valores frontera $a \leq b$, se denotará $[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}$ para un intervalo cerrado, mientras que

$$(a, b] = \{x \in \mathbb{R} : a < x \leq b\}, \quad [a, b) = \{x \in \mathbb{R} : a \leq x < b\},$$

representan intervalos semiabiertos, y finalmente $(a, b) = \{x \in \mathbb{R} : a < x < b\}$ denota un intervalo abierto.

1.2. Variables aleatorias, distribuciones y densidades

Se indicarán las variables aleatorias (v.a.) con letras mayúsculas (X, $\mathbf{X}$), mientras que se utilizarán letras minúsculas para denotar las correspondientes realizaciones (x, $\mathbf{x}$). Nótese que cuando las letras están en negrita puede confundirse la notación con la utilizada para matrices y vectores, y es por eso, que será el propio texto el que indique si es una v.a o no. A menudo, cuando se muestrea una colección de variables aleatorias, se utilizará la notación de superíndices $x^{(i)}$, $\mathbf{x}^{(i)}$, donde i hace referencia a la muestra i-ésima.

Se usarán letras minúsculas, por ejemplo $q(\cdot)$, para denotar la función densidad de probabilidad de una variable aleatoria. Así, la densidad de probabilidad de Y se expresará como $q(y)$, y la densidad de probabilidad de X dado $Y = y$ se escribirá $p(x|y)$. La función de distribución de una v.a. se indicará como $F_X(\cdot)$ y la probabilidad de que se de un evento, por ejemplo $X \leq x$, se denotará como $\text{Prob}\{X \leq x\}$. Así que podemos escribir $F_X(a) = \text{Prob}\{X \leq a\}$.

La función densidad objetivo de la que se desea generar muestras se denota como $p_o(x)$ mientras que $p(x)$ es una función proporcional a $p_o(x)$, y se podrá indicar como $p(x) \propto p_o(x)$.

Con $\pi(x)$ se denotará la densidad tentativa a partir de la cual se generarán nuevos puntos candidatos. Es más sencillo generar muestras de la densidad $\pi(x)$ que de la función densidad objetivo $p_o(x)$, y suele servir de apoyo para muestrear esta segunda.

Se indicará indistintamente una densidad o una distribución uniforme en un intervalo $[a, b]$ como $\mathcal{U}([a, b])$. De igual forma, la densidad o distribución Gaussiana con media μ y varianza σ^2 se denotará como $\mathcal{N}(\mu, \sigma^2)$. Será el propio texto el que diferencie si se está trabajando con una densidad o con una distribución. Con el símbolo $\sim$ se indicará que una v.a. X o una muestra aleatoria x' posee la densidad indicada, por ejemplo, $X \sim \mathcal{U}([a, b])$ ó $x' \sim \mathcal{N}(\mu, \sigma^2)$. La expresión $X \stackrel{d}{=} Z$ denota que dos variables aleatorias X y Z son "iguales en distribución", es decir, tienen la misma función de distribución de probabilidad.

1.3. Conjuntos

Los conjuntos se denotarán mediante letras mayúsculas caligráficas, por ejemplo, $\mathcal{R}$. El soporte de la V.A. de interés X se denotará como $\mathcal{S} \subseteq \mathbb{R}$ (por ejemplo, $\mathcal{S}$ es el dominio de la densidad objetivo $p_o(x)$). En algunos casos, sin perdida de la generalidad se puede considerar $\mathcal{D} = \mathbb{R}$ por conveniencia.

La función indicadora del conjunto $\mathcal{S}$ se escribe como $\mathbb{I}_{\mathcal{S}}(x)$. Dicha función toma el valor 1 si $x \in \mathcal{S}$ y 0 en caso contrario.

$$\mathbb{I}_{\mathcal{S}}(x) = \begin{cases} 1 & \text{if } x \in \mathcal{S} \\ 0 & \text{if } x \notin \mathcal{S} \end{cases}. \tag{1.1}$$

1.4. Notación y funciones más utilizadas

- $\pi(x)$: densidad tentativa ("*proposal density*") utilizada para generar puntos candidatos.
- $p_o(x)$: densidad objetivo ("target density") que se desea muestrear. Es decir, aquella densi-

dad de la que se desea generar muestras aleatorias.

- $p(x)$: función proporcional a la densidad objetivo, es decir, $p(\mathbf{x}) \propto p_o(\mathbf{x})$.
- $f(x)$: función de coste que se quiere optimizar (en concreto, minimizar).
- $\mathcal{S} \subseteq \mathbb{R}^m$: conjunto de definición de la variable x de interés.
- $\mathcal{U}([a,b])$: densidad o distribución uniforme en el intervalo cerrado de límites a y b.
- $\mathcal{N}(\mu, \sigma^2)$: densidad o distribución gaussiana de media μ y varianza σ^2.
- $\delta(x-\mu)$: delta de Dirac definida como

$$\delta(x-\mu) = \begin{cases} +\infty & \text{if } x = \mu \\ 0 & \text{if } x \neq \mu \end{cases}. \tag{1.2}$$

- $\mathcal{H}(\mathbf{x}_c, r)$: se trata de un conjunto particular, una hiperesfera (que también llamaremos "vecindario") según la norma generica $||\cdot||$ centrada en $\mathbf{x}_c$ y de radio r, es decir

$$\mathcal{H}(\mathbf{x}_c, r) = \{\mathbf{x} \in \mathbb{R}^m : \ ||\mathbf{x} - \mathbf{x}_c|| \leq r\}.$$

1.5. Abreviaciones

v.a.: **V**ariable **A**leatoria.

PRS: **P**ure **R**andom **S**earch.

PAS: **P**ure **A**daptive **S**earch.

HAS: **H**esitant **A**daptive **S**earch.

ARS: **A**celerated **R**andom **S**earch.

ML: **M**aximum **L**ikelihood.

TSP: **T**ravelling **S**alesMan **P**roblem.

SA: **S**imulated **A**nnealing.

MPSA: **M**ultiple **P**article **S**imulated **A**nnealing.

TA: **T**hreshold **A**ccepting.

PSO: **P**article **S**warm **O**ptimization.

ACO: **A**nt **C**olony **O**ptimization.

ST: **S**imulated **T**empering.

MH: **M**etropolis **H**astings.

Capítulo 2

Objetivos del proyecto

Este proyecto se propone investigar las potencialidades de los métodos de optimización estocástica, y su estrecha relación con los métodos de muestreo aleatorio (también conocidos como métodos de Monte Carlo).

Las principales fases del proyecto han sido cuatro:

- Una primera fase de estudio bibliográfico profundo y exhaustivo, en la que se buscó y examinó una cantidad considerable de información sobre algoritmos de optimización estocástica, con el fin de asimilar el mayor conocimiento posible. Una vez hecho esto, se clasificaron los algoritmos más interesantes en cuanto a facilidad de implementación, flexibilidad y robustez. Se han buscado métodos potentes que no necesiten de excesiva información del problema para ser aplicados. Fáciles tanto en implementación como en concepto, y por último, flexibles o generales, ya que son capaces de resolver un abanico de problemas muy amplio.

- La segunda fase del proyecto se concentró en investigar la relación existente entre los métodos estocásticos de optimización y los métodos de muestreo de variables aleatorias. En mucho casos, los métodos de optimización hallan la explicación teórica de su óptimo funcionamiento en los métodos de muestreo. Fue nuestra misión en esta fase del trabajo encontrar bases teóricas consistentes, que evidenciaran la relación entre estos dos tipos de técnicas, y explicarán como se llega a estos resultados satisfactorios analíticamente.

- En el transcurso de la tercera fase se ha intentado identificar los pasos claves que unen todos los algoritmos de optimización estudiados, de modo que se pudieran establecer las semejanzas entre los mismos, así como los rasgos distintivos. Esta fase tenía como objetivo,

el poder diseñar nuevos algoritmos; intentando fusionar las características positivas de distintas filosofías de optimización, con el fin de mejorar las técnicas ya existentes.

- En la cuarta y última fase se implementaron algunos de los algoritmos seleccionados como más interesantes y se comprobó su funcionamiento para un problema real. Esta implementación sirvió para ver de forma práctica los conceptos estudiados y sacar las conclusiones pertinentes. Es importante resaltar que aunque las pruebas se realizaron para resolver un problema especifico (y teórico), el código puede usarse en cualquier tipo de problema distinto.

El proyecto está estructurado de modo que en el Capítulo 3 se introducirá el problema de optimización a resolver y se analizarán los diferentes algoritmos de muestreo existentes. Además, se evidenciarán las importantes relaciones entre optimización y muestreo. En el Capítulo 4 se explicará el funcionamiento de las cadenas de Markov y se tratará en profundidad el algoritmo Metropolis-Hastings. Esta técnica de muestreo es la más útil en el sentido de que ha sido el método que ha llevado a los investigadores a desarrollar mayor cantidad de técnicas de optimización y de mayor éxito. El Capítulo 5 está dedicado a los métodos estocásticos de optimización, intentando focalizar la atención en las técnicas, a nuestro juicio, más interesantes. A su vez, dada la jungla de métodos que se pueden considerar estocásticos, se realizará una clasificación de los mismos lo más clara y sencilla posible. En el Capítulo 6 se implementarán los algoritmos Simulated Annealing y Acelerated Random Search para la optimización de una función de coste complicada. Se realizarán pruebas con todos los parámetros que intervienen en los mismos con el fin de analizar el comportamiento de estas técnicas. Por último, en el Capítulo 7 se mostrarán las conclusiones obtenidas en este proyecto y se darán ideas sobre las posibles lineas futuras de investigación.

Capítulo 3

Introducción a la optimización y el muestreo

3.1. Problema de optimización

Este proyecto se centrará en resolver problemas de optimización del tipo

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}}[f(\mathbf{x})] \quad \text{con} \quad \mathbf{x} \in \mathcal{S} \subseteq \mathbb{R}^m \tag{3.1}$$

donde $f(\mathbf{x})$ representa una función objetivo genérica (a la que nos referiremos también como función de coste, en el resto del trabajo) y $\mathcal{S}$ representa el conjunto de valores que puede asumir la variable $\mathbf{x} = [x_1, ..., x_m]$. En este trabajo, se estudiará y analizará la utilización de técnicas de optimización estocástica.

Con $\hat{\mathbf{x}}$ se indica la posición del mínimo *global* óptimo. Además, es importante notar que en general puede haber más de un punto donde se logre el valor del mínimo global así que una escritura más adecuada sería

$$\hat{\mathcal{X}} = \left\{ \mathbf{x} \in \mathcal{S} \subseteq \mathbb{R}^m : \quad \mathbf{x} = \arg\min_{\mathbf{x}}[f(\mathbf{x})] \right\}, \tag{3.2}$$

donde el conjunto $\hat{\mathcal{X}} = \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, ..., \hat{\mathbf{x}}_q\}$ puede contener más de un elemento.

Para asegurar la existencia de un mínimo global óptimo $\hat{\mathbf{x}}$, se necesita asumir algunas condiciones de regularidad. Si el espacio de búsqueda $\mathcal{S}$ es no vacío y compacto, entonces el teorema de Weierstrass del análisis clásico nos garantiza la existencia de una solución global para el problema expresado en la Ecuación (3.1).

3.1.1. Métodos deterministas

Los métodos más utilizados para la resolución de problemas de optimización son tradicionalmente deterministas, basados sobre todo en técnicas y resultados del análisis numérico. Se

trata de métodos iterativos, que en ciertas circunstancias, convergen a los puntos estacionarios (gradiente nulo) de la función $f(\mathbf{x})$.

El principal defecto de este enfoque es la necesidad de conocer información analítica sobre la función de coste $f(\mathbf{x})$. Por esto, en muchos casos, estas técnicas asumen ciertas condiciones de regularidad sobre $f(\mathbf{x})$.

Además, la posible aplicación de un método determinista depende mucho de la dimensión m, la complejidad del espacio de búsqueda $\mathcal{S}$ y del número de parámetros en juego.

Pero quizás el mayor inconveniente es la dificultad en garantizar la convergencia al mínimo global. La casi totalidad de los algoritmos suelen quedar atrapados en mínimos *locales* dependiendo de la complejidad de la función de coste o de las condiciones iniciales. Así que, en muchos casos, no se puede asegurar que la solución obtenida con algoritmos deterministas sea la correcta, y además, los recursos computacionales necesarios para lograr una solución válida suelen ser demasiado altos.

3.1.2. Optimización estocástica

Por estas múltiples razones se han estudiado otros tipos de enfoques. Por ejemplo, en 1991, Dyer y Frieze [Dyer et al., 1991] demostraron que la estimación del volumen de un cuerpo convexo conlleva evaluar una función en un número exponencial de ocasiones. Sin embargo, si se acepta una solución menos precisa, pero con alta probabilidad de encontrarse cerca de la solución correcta, entonces un algoritmo estocástico puede proporcionar resultados en tiempo polinómico.

En este proyecto se estudiarán unos algoritmos de optimización que contienen un cierto grado de aleatoriedad en su pasos. Por eso, se hablará de algoritmos de optimización estocástica. Además, se analizará y comentará la estrecha relación entre métodos de generación de números aleatorios y métodos de optimización estocástica (vease, por ejemplo, la sección (3.3)).

Utilizando la clasificación realizada por Geyer en [Geyer, 1996], y que aparece recogida también en [Robert y Casella, 1999], es útil distinguir ahora entre dos clases de algoritmos:

1. *aproximación estocástica*,
2. y *exploración estocástica*.

Este proyecto se centrará en técnicas que pertenecen a esta última clase.

Las técnicas de aproximación estocástica construyen aproximaciones locales de la función objetivo $f(\mathbf{x})$ para luego optimizar. Dos ejemplos de métodos de aproximación estocástica son el algoritmo EM y el algoritmo de Robbins-Monroe [Robert y Casella, 1999].

Por otro lado, los algoritmos de exploración estocástica intentan buscar el mínimo global en sub-regiones del espacio completo de búsqueda $\mathcal{S}$, donde con mayor probabilidad se encuentra la solución. Estas técnicas suelen ser denominadas algoritmos de *búsqueda aleatoria* ("random search algorithms").

A continuación, se enumerarán las razones de porqué estas técnicas han despertado tanto interés en la literatura y, además, se indicarán unas importantes limitaciones.

3.1.3. Ventajas de la exploración estocástica

- Se trata de algoritmos muy potentes, porque necesitan sólo *saber evaluar la función de coste* $f(\mathbf{x})$, en cualquier punto.

  Así que pueden tratar problemas con función objetivo $f(\mathbf{x})$ no convexa, no diferenciable y posiblemente no continua; cuyo dominio puede ser continuo, discreto, o incluso una mezcla de ambos. Obviamente, esto significa que la variable $\mathbf{x}$ puede ser tanto continua como discreta, o una mezcla de variables continuas y discretas. Además, el espacio de búsqueda $\mathcal{S}$ puede ser arbitrariamente complicado.

  Incluso, estas técnicas pueden ser utilizadas en problemas donde la misma función de coste $f(\mathbf{x})$ y la región $\mathcal{S}$ estén afectadas por un cierto grado de aleatoriedad. Por ejemplo, cuando $f(\mathbf{x})$ y $\mathcal{S}$ pueden cambiar con el tiempo, o cuando son conocidos con un cierto grado de incertidumbre.

- La convergencia al mínimo global está garantizada (por lo menos, para los algoritmos más importantes), aunque se trate de una *convergencia en probabilidad*. Esto significa que nuestro algoritmo llegará seguramente al mínimo global, pero en general no se sabe "cuando". Es decir, no se conoce cuanto tiempo necesitará nuestro algoritmo para hallar la solución óptima.

- La principal virtud es la de ser capaces de encontrar soluciones razonablemente buenas de forma muy rápida. Aunque la solución proporcionada puede que no sea la óptima (dependiendo de la condición de parada del algoritmo), ésta ha sido conseguida con esfuerzo computacional relativamente bajo y en poco tiempo.

- Otra ventaja de los algoritmos de exploración estocástica es la fácil implementación (programables en una FPGA, por ejemplo) para diferentes aplicaciones.

  El hecho de que estos métodos tan solo necesiten de la evaluación de la función objetivo

$f(\mathbf{x})$, los convierte en métodos sumamente flexibles que pueden funcionar "bien", para muchos problemas distintos. A efectos prácticos esto se traduce en que estas técnicas puede ser implementadas/programadas una sola vez y pueden estar ya disponibles para resolver una amplia gama de problemas de optimización.

3.1.4. Carencias de la exploración estocástica

- Un defecto de estas técnicas es que las prestaciones dependen mucho de la elección de los parámetros que rigen el algoritmo. Estos parámetros se suelen ajustar para la resolución del problema específico, y no existen actualmente reglas establecidas sobre como elegirlos. Por ello, la forma más corriente de ajuste consiste en experimentos previos de prueba/error.

- Otra importante limitación, ya citada anteriormente, es que, aunque la convergencia está garantizada (en probabilidad), no se sabe en que iteración el algoritmo convergerá. Por esto, no es fácil diseñar condiciones adecuadas de finalización del algoritmo. En general, se suelen fijar un número máximo de iteraciones admitidas, o bien un número máximo de iteraciones de espera sin que el algoritmo haya mejorado la solución anteriormente proporcionada.

- Ciertos métodos, o incluso ciertas clases de estas técnicas de exploración estocástica, no se basan directamente en un desarrollo teórico robusto, sino más bien se rigen en similitudes con otros fenómenos físicos y en resultados de simulaciones/experimentaciones.

De todas formas, se puede afirmar con rotundidad, que a pesar de las desventajas comentadas, los métodos de búsqueda aleatoria son métodos muy populares, capaces de generar soluciones *razonablemente buenas* de forma *rápida y sencilla*, por lo que son frecuentemente utilizados en la práctica.

3.2. Clasificación

Como hemos evidenciado anteriormente, no siempre se ha desarrollado una teoría profunda y robusta que certifique el buen funcionamiento de un algoritmo de exploración estocástica. Esto se debe a que la atención de los científicos se ha concentrado en encontrar algoritmos lo más rápidos y más precisos posibles, con un método, en general, de prueba y error.

Esto ha generado una jungla de técnicas algunas más lejanas, otras más parecidas, pero, en muchos casos, difíciles de relacionar entre sí. De hecho, muchísimas clasificaciones y de distinta naturaleza han sido sugeridas. Es por ello que una primera fase de este proyecto ha sido dedicada

a un largo trabajo de búsqueda bibliográfica y de comparación de metodología. Parte de este trabajo, se ve reflejado en el Capítulo 5.

Además, es importante recordar que en la sección previa, ya se hizo una primera distinción entre técnicas de aproximación estocástica y de exploración estocástica.

A esta última clase (exploración estocástica) se suele asociar comúnmente el término *metaheurístico*. En ocasiones, el término metaheurístico también se utiliza para designar a aquellos algoritmos que *simulan* comportamientos físicos reales existentes en la naturaleza, como por ejemplo la evolución de las especies o la organización de ciertos animales (abejas u hormigas principalmente).

Entre las muchas observaciones y distinciones encontradas en la literatura sobre estos métodos, nos han parecido particularmente relevantes las dos siguientes:

- Schoen [Schoen, 2002] divide los métodos de búsqueda aleatoria en dos fases:

  1. *búsqueda local*,
  2. *búsqueda global*.

  La búsqueda local cumple una búsqueda especifica (por ejemplo en la cuenca de atracción de un solo mínimo) y precisa del mínimo, mientras la segunda fase se refiere a una búsqueda más amplia pero menos exhaustiva del mínimo (se busca en espacio más amplio, pero con alta probabilidad de albergar un mínimo).

  Esta división permite clasificar las distintas técnicas según el tipo de búsqueda en local o global.

- Zlochin [Zlochin et al., 2004] siguieren distinguir los algoritmos de exploración estocástica en dos clases:

  1. "model based methods", e
  2. "instance based methods".

  Los primeros consideran un modelo probabilístico conocido explícitamente y asociado al problema de optimización, desde donde generar números aleatorios (puntos en la búsqueda aleatoria). Algoritmos de este tipo son "Ant colony optimization" y el "Cross-entropy method".

  Al contrario, los segundos no asumen ningún modelo específico, y la generación de nuevos puntos candidatos depende del estado anterior en el cual se encuentra el algoritmo. Ejemplos

de estos últimos son "Simulated annealing" o los algoritmos genéticos.

Además, nos parece importante la distinción entre

- algoritmos de un *solo punto*,
- algoritmos basados con un conjunto de puntos a la vez, es decir con *poblaciones*.

Claramente, los segundos garantizan mejores prestaciones pero con coste computacional mayor. Además, aumenta también el coste de diseño dado que hay que establecer como los distintos puntos interactúan entre si, garantizando una rápida convergencia.

En este proyecto analizaremos sobre todo aquellos algoritmos de más amplia aplicación, que pueden ser utilizados con la misma facilidad en problemas discretos como continuos (o mezcla de los dos). También, se han tratado algunos que sólo pueden trabajar en dominios discretos, debido a su especial relevancia.

3.3. Relación entre optimización y muestreo

Todos los algoritmos de búsqueda aleatoria utilizan internamente generadores de números aleatorios. Además, las prestaciones son fuertemente dependientes de la elección de este generador.

En general, se suelen llamar *métodos de muestreo* a aquellas técnicas que generan "muestras" (números aleatorios) que se distribuyen como una determinada densidad objetivo $p_o(\mathbf{x})$. No todas las densidades de probabilidad son fácilmente "muestreables", así que existen diferentes enfoques y técnicas de muestreo.

La optimización y el muestreo son dos problemas íntimamente relacionados. Se puede decir que "*el muestreo es un problema más fácil respecto a la optimización; se puede considerar una primera fase*" [Liang et al., 2010]. Esta frase es cierta, aunque se puede afirmar también que un problema de optimización se puede reducir, casi por completo, a un problema de generación de números aleatorios con una adecuada densidad de probabilidad (por lo menos conceptualmente). Esta sección está dedicada a explicar y aclarar esta estricta relación.

Dado un dominio $\mathcal{S}$ finito y acotado, si se supone que somos capaces de muestrear una variable uniforme definida en $\mathcal{S}$, un método trivial de exploración estocástica podría ser:

1. Muestrear N puntos $\mathbf{x}^{(i)}$, $i = 1, ..., N$, $\mathbf{x}^{(i)} \in \mathbb{R}^m$, distribuidos uniformemente en $\mathcal{S}$.
2. La salida del algoritmo será $\tilde{\mathbf{x}} = \arg\max q_T(\mathbf{x}^{(i)}) = \arg\min f(\mathbf{x}^{(i)})$, siendo $f(\mathbf{x})$ nuestra función de coste a minimizar.

Es decir, se evalúa N veces la función objetivo $f(\mathbf{x})$ en los puntos elegidos uniformemente en $\mathcal{S}$. Seguramente si $N \to \infty$, se conseguirá *exactamente* el mínimo global óptimo $\hat{\mathbf{x}}$, pero el esfuerzo computacional también tenderá a infinito.

Este algoritmo también se puede escribir de forma iterativa en la forma:

1. Se elige $\mathbf{x}^{(0)}$ uniformemente en $\mathcal{S}$, y se inicializa $i = 0$.
2. Se genera una nueva muestra $\mathbf{x}'$ uniformemente en $\mathcal{S}$.
3. Si $f(\mathbf{x}^{(i)}) \leq f(\mathbf{x}') \to \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)}$.
4. Si $f(\mathbf{x}^{(i)}) > f(\mathbf{x}') \to \mathbf{x}^{(i+1)} = \mathbf{x}'$.
5. Actualizar $i = i + 1$. Si $i \neq N$ volver al paso 2.
6. En caso contrario el algoritmo termina con salida $\tilde{\mathbf{x}} = \mathbf{x}^{(N)}$.

Es evidente que se debe esperar un tiempo infinito para asegurarse la convergencia exacta al mínimo global óptimo.

Sin embargo, es muy interesante notar que a cualquier función de coste $f(\mathbf{x})$ se puede asociar a una densidad de probabilidad de la forma

$$q_T(\mathbf{x}) \propto H \circ f(\mathbf{x})/T = H(f(\mathbf{x})/T), \tag{3.3}$$

Donde $H(\vartheta) : \mathbb{R} \to \mathbb{R}$ es una función positiva decreciente y T es un parámetro de escala. Claramente se tiene que cumplir

$$\int_{\mathbf{x}\in\mathcal{S}} H(f(\mathbf{x})/T)d\mathbf{x} \leq +\infty. \tag{3.4}$$

En caso de que $H(\vartheta) = \exp(-\vartheta)$, tenemos $q_T(\mathbf{x}) = \exp(-f(\mathbf{x})/T)$, que se corresponde con la distribución de Boltzmann que veremos con detalle en el Capítulo 5.

Se deben realizar 3 observaciones de suma importancia:

- Los valores $\hat{\mathbf{x}}$ que minimizan la función objetivo $f(\mathbf{x})$ son los mismos que maximizan la función $q_T(\mathbf{x})$, es decir, sus modas. Por lo tanto se puede escribir que

$$\hat{\mathbf{x}} = \arg\min_x f(\mathbf{x}) = \arg\max_x q_T(\mathbf{x}). \tag{3.5}$$

- Muestrear desde una densidad $q_T(\mathbf{x})$ significa generar números aleatorios con una frecuencia de aparición proporcional al área por debajo de $q_T(\mathbf{x})$. Es decir, si somos capaz de muestrear

$q_T(\mathbf{x})$, se obtendrá muestras $\mathbf{x}'$ cercanas a las modas de $q_T(x)$ con probabilidad alta, o lo que es lo mismo, cercanas al mínimo global de la función objetivo $f(x)$. De esta forma se puede proponer un método más refinado que el trivial visto anteriormente:

1. Muestrear N puntos $\mathbf{x}^{(i)} \sim q_T(\mathbf{x}), i = 1, ..., N$.
2. La salida del algoritmo será $\tilde{\mathbf{x}} = \arg\min f(\mathbf{x}^{(i)})$.

Seguramente la convergencia será más rápida que la vista en el método trivial, porque se seleccionan puntos $\mathbf{x}^{(i)}$ cerca del mínimo global de $f(\mathbf{x})$. Es decir, se están explorando zonas del espacio de búsqueda $\mathcal{S}$ donde con mayor probabilidad se encuentran los mínimos de la función de coste $f(\mathbf{x})$; mientras anteriormente, dado que se trabajaba con una distribución uniforme, no se hacia ninguna diferenciación entre unas regiones del espacio $\mathcal{S}$ y otras.

- El parámetro de escala T actúa a modo de varianza. La densidad $q_T(\mathbf{x})$ al variar dicho parámetro T mantiene la posición de las modas pero varía la dispersión de $q_T(\mathbf{x})$. Cuanto más pequeño es el valor de T, más estrecha se hará la función $q_T(\mathbf{x})$ en torno a su máximo global (mínimo global para $f(\mathbf{x})$). De este modo, si se sabe muestrear desde $q_T(\mathbf{x})$ cuando T es pequeño ($T \to 0$), la convergencia del método de optimización sería más rápida. Para el caso límite de $T = 0$, la función $q_T(\mathbf{x})$ se convierte en una delta de Dirac (o un tren de deltas) y la convergencia del método de optimización sería prácticamente instantánea. En este caso, se necesitaría muestrear sólo un punto que coincidiría con el mínimo global de $f(\mathbf{x})$ (máximo global de $q_T(\mathbf{x})$). En este sentido, se puede afirmar que un problema de optimización se reduce a un problema de muestreo de una densidad.

A continuación, se presentará, en forma muy sintética, las principales estrategias utilizadas para la generación de números aleatorios distribuidos con una densidad de probabilidad no uniforme genérica.

3.4. Métodos de muestreo

Los métodos de muestreo proporcionan números aleatorios de acuerdo a una *densidad objetivo* $p_o(\mathbf{x})$ con $\mathbf{x} \in \mathbb{R}^m$. Estos métodos se diferencian de los métodos de generación de números pseudo-aleatorios porque asumen a su disposición una fuente aleatoria con densidad conocida. Es decir, mientras que los métodos de generación de números pseudo-aleatorios son realmente algoritmos deterministas (típicamente caóticos) que simulan la aleatoriedad de la secuencia generada, los algoritmos de muestreo elaboran las muestras proporcionadas por otra fuente, logrando nuevas muestras distribuidas como la densidad objetivo $p_o(\mathbf{x})$.

Introduciremos las distintas clases de métodos de muestreo existentes, proporcionando algún ejemplo significativo en cada clase. Los métodos de muestreo se pueden dividir de la siguiente forma:

1. Métodos basados en la *Transformación* de una variable aleatoria.

2. Métodos basados en un test de *aceptación-rechazo*.

3. Métodos que utilizan *cadenas de Markov* ("Markov Chain Monte Carlo", MCMC).

4. Métodos que asignan *pesos* a las muestras ("Importance Sampling").

Es importante evidenciar que esta última categoría no puede realmente considerarse como una clase de generadores de números aleatorios. De hecho, mientras los tres primeros grupos pueden considerarse generadores de números aleatorios, el "Importance Sampling", y sus variantes, aproximan la medida de probabilidad definida por la densidad objetivo $p_o(\mathbf{x})$, asignando pesos a las muestras generadas por la fuente aleatoria. Se suele utilizar sobre todo para el cálculo de momentos, o de forma más general para calcular integrales.

La primera clase está compuesta por los métodos más eficaces dado que todas las muestras generadas resultan ser independientes entra ellas, y se logran tan solo aplicando una transformación adecuada. El único inconveniente de este método es que en muy pocas ocasiones puede ser utilizado, dado que no es fácil encontrar la transformación apropiada.

Los métodos de Aceptación-Rechazo también son universales (en teoría, aplicables a cualquier densidad) y generan muestras independientes. Sin embargo, no todas las muestras generadas son aceptadas. Además, en ocasiones la probabilidad de rechazo puede ser muy elevada, haciendo el método inviable desde el punto de vista computacional. Otro inconveniente es que se necesita el cálculo analítico de cotas superiores, que no siempre es posible.

El tercer grupo está compuesto por generadores aleatorios basados en cadenas de Markov. Estos métodos tienen la gran ventaja que pueden ser aplicados fácilmente en cualquier ocasión. Como desventaja se tiene el hecho de que las muestras generadas están correlacionadas entre si, y que se necesita de un periodo transitorio de convergencia de la cadena a la densidad objetivo $p_o(\mathbf{x})$ ("*burn in period*").

3.4.1. Métodos basados en transformación

Para facilitar la comprensión, en esta sección vamos a considerar el caso unidimensional. Se consideran dos variables aleatorias $Y \sim q(y)$, $Z \sim h(z)$ tales que $Z = g(Y)$ donde $g(y) : \mathbb{R} \to \mathbb{R}$ es una función monótona, $Y \in \mathbb{R}$ y $Z \in \mathbb{R}$. Es bien conocido que las dos densidades están relacionadas entre sí, por la siguiente expresión

$$h(z) = q(g^{-1}(z)) \left| \frac{dg^{-1}}{dz} \right|. \tag{3.6}$$

Si sabemos generar muestras y' desde $q(y)$ podemos lograr muestras z' con densidad $h(z)$, utilizando la relación $z' = g(y')$.

Recordamos que por simplicidad hemos tratado el caso escalar $Y \in \mathbb{R}$ y $Z \in \mathbb{R}$, pero las mismas consideraciones pueden ser extendidas al caso multidimensional. El método de inversión que se tratará a continuación nos proporciona una transformación una variable uniforme con otra variable aleatoria (v.a.) de interés.

Método de Inversión

Dada una v.a. uniforme $U \sim \mathcal{U}([0, 1])$, luego otra v.a. $X \sim p_o(x)$ y su función de distribución acumulativa

$$F_X(x) = \int_{-\infty}^{x} p_o(y)dy, \tag{3.7}$$

es fácil demostrar que la siguiente relación entre variables aleatorias

$$U = F_X(X), \tag{3.8}$$

se verifica. Además, si se indica con F_X^{-1} la inversa de la función acumulativa F_X, podemos escribir

$$X = F_X^{-1}(U). \tag{3.9}$$

La ecuación nos sugiere un algoritmo para generar muestras de $p_o(x)$ [Robert y Casella, 1999]. De hecho, asumiendo que se sabe muestrear una densidad uniforme en $[0, 1]$, los siguientes dos pasos

1. generar $u' \sim \mathcal{U}[0,1]$,
2. transformar u' utilizando F_X^{-1}, es decir, $x' = F_X^{-1}(u')$,

producen muestras x' distribuidas como $p_o(x)$.

Este método genera muestras independientes y con el mínimo esfuerzo computacional posible: tan sólo una transformación.

El problema es que, en general:

1. es un método valido sólo para variables escalares,
2. no siempre es posible calcular analíticamente la función acumulativa $F_X(x)$,
3. y, en muchos casos, aunque es posible conseguir F_X, la función acumulativa no puede ser invertida analíticamente para hallar F_X^{-1}.

Así, que en la practica, su aplicación es muy limitada. Por esta razón, otras técnicas han sido desarrolladas.

3.4.2. Método de aceptación-rechazo

El método de aceptación-rechazo fue sugerido por John Von Neumann a mediados del año 1951 [Von Neumann, 1951]. Se trata de una técnica de muestreo "universal", dado que puede ser utilizada para muestrear cualquier densidad objetivo $p_o(\mathbf{x}) \propto p(\mathbf{x})$ $(x \in \mathcal{S} \subset \mathbb{R}^m)$. Su aplicación está limitada sólo por la habilidad de encontrar una pareja adecuada $\pi(\mathbf{x})$ y M, donde $\pi(\mathbf{x})$ es una densidad fácil de muestrear (*densidad tentativa*) y M es una constante tal que

$$M \geq \frac{p(\mathbf{x})}{\pi(\mathbf{x})}. \tag{3.10}$$

El algoritmo consiste en generar muestras desde $\pi(\mathbf{x})$, y aceptarla con probabilidad

$$p_A = \frac{p(\mathbf{x})}{M\pi(\mathbf{x})}, \tag{3.11}$$

donde se recuerda que $p(\mathbf{x}) \propto p_o(\mathbf{x})$. Es decir, el algoritmo aplica un test a cada muestra que puede ser aceptada o rechazada. Claramente, es posible demostrar que las muestras aceptadas se distribuyen de acuerdo a la densidad objetivo $p_o(\mathbf{x})$.

Más especificamente, los pasos del algoritmo son los siguientes:

1. Considerar $i = 0$. Se genera una muestra $\mathbf{x}' \sim \pi(\mathbf{x})$ y $u' \sim \mathcal{U}([0,1])$.
2. Si $u' \leq \frac{p(\mathbf{x}')}{M\pi(\mathbf{x}')} \rightarrow x^{(i)} = \mathbf{x}'$. Actualizar $i = i + 1$.

3. Si $u' > \frac{p(\mathbf{x}')}{M\pi(\mathbf{x}')}$, descartar $\mathbf{x}'$.

4. Si $i < N$ volver al paso 1. En caso contrario, se finaliza el algoritmo.

Es importante notar que el método de aceptación-rechazo realmente produce puntos aleatorios distribuidos uniformemente en el área por debajo de la función $p(\mathbf{x}) \propto p_o(\mathbf{x})$.

Para facilitar la comprensión, vamos a considerar el caso escalar $x \in \mathbb{R}$. Como se muestra en la Figura 3.1, el algoritmo genera la pareja (x', v') donde $x' \sim \pi(x)$ y v' es una muestra uniforme en el intervalo $[0, M\pi(x')]$, es decir, $v' \sim \mathcal{U}([0, M\pi(x')])$. La Figura 3.1 muestra como lo puntos (x', v') pertenecientes al área por debajo de $p(x) \propto p_o(x)$ son aceptados (puntos en verde).

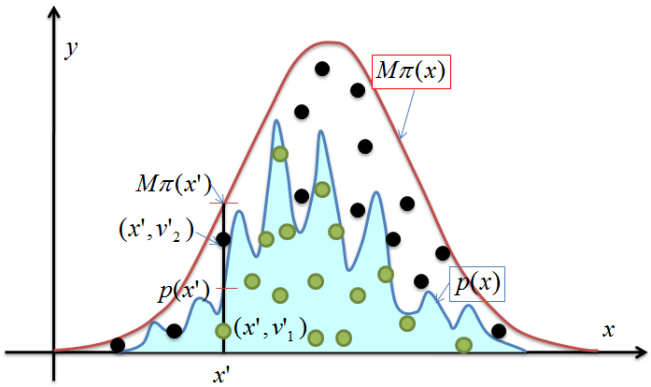

Figura 3.1. Funcionamento del método de aceptación-rechazo. Dado una pareja (x', v') con $x' \sim \pi(x)$ y $v' \sim \mathcal{U}([0, M\pi(x')])$, el punto (x', v') es aceptado si cae dentro del área en azul por debajo de la función objetivo $p(x) \propto p_o(x)$. En este caso, la muestra x' es aceptada. Los puntos aceptados están representados con el color verde mientras que los rechazados en color negro.

Si la condición $M \geq \frac{p(\mathbf{x})}{\pi(\mathbf{x})}$ en la Ecuación (3.10) no está garantizada $\forall \mathbf{x} \in \mathcal{S}$, las muestras generadas por el algoritmo de aceptación-rechazo se distribuyen de acuerdo a la densidad

$$q(\mathbf{x}) \propto \text{mín}\,[p(\mathbf{x}), M\pi(\mathbf{x})].$$

De hecho, es fácil ver que las muestras generadas $\mathbf{x}' \sim \pi(\mathbf{x})$ en las regiones donde $M\pi(\mathbf{x}') < p(\mathbf{x}')$ serán aceptadas con probabilidad uno, y claramente se distribuirán como la densidad tentativa $\pi(\mathbf{x})$. Al contrario, si para las muestras generadas $\mathbf{x}' \sim \pi(\mathbf{x})$ se cumple que $M\pi(\mathbf{x}') \leq p(\mathbf{x}')$ en la Ecuación (3.10), entonces el algoritmo de aceptación-rechazo funcionará correctamente aceptando solo las muestras que se distribuyen de acuerdo a $p_o(\mathbf{x}) \propto p(\mathbf{x})$.

Hallar una buena cota superior M es fundamental para limitar la tasa de rechazo. La cota óptima es obviamente

$$M^* = \sup p(\mathbf{x})/\pi(\mathbf{x}),$$

que es el valor mínimo que garantiza la desigualdad (3.10).

Pero, aunque sea posible hallar M^*, las prestaciones del algoritmo pueden no ser buenas, es decir, la probabilidad p_A en (3.11) puede ser lejana de 1. Esto, que constituye una limitación estructural del método de aceptación-rechazo, ocurre porque, en general, la forma de la densidad tentativa $\pi(\mathbf{x})$ difiere de la forma densidad objetivo $p_o(\mathbf{x}) \propto p(\mathbf{x})$. Así que para mejorar la tasa de aceptación es necesario utilizar una densidad $\pi(\mathbf{x})$ lo más parecida posible a $p(\mathbf{x})$.

Se puede afirmar de este modo que la eficiencia del algoritmo depende de la elección de la pareja M y $\pi(\mathbf{x})$.

Como hemos dicho el algoritmo de aceptación-rechazo es un método universal que produce muestras independientes, pero el precio a pagar por esto, es que muchas muestras generadas por la densidad tentativa serán descartadas. Para generar N muestras desde $p_o(\mathbf{x})$ vamos a necesitar $N + R$ muestras desde $\pi(\mathbf{x})$ (R es el numero de muestras rechazadas), así que en general el algoritmo puede resultar muy lento. Otro defecto es la necesidad de hallar analíticamente una cota superior M lo más fina posible.

3.4.3. Métodos MCMC

Los métodos "Markov Chain Monte Carlo" (MCMC) se basan en la utilización de una cadena de Markov. Las cadenas de Markov, bajo ciertas condiciones, suelen converger a una densidad estacionaria, invariante con el tiempo, que nosotros denotaremos $p_e(\mathbf{x})$.

Así que la idea de fondo de los métodos MCMC es diseñar una cadena de Markov cuya densidad estacionaria $p_e(x)$ coincida con la densidad objetivo $p_o(\mathbf{x})$. En esta caso, la función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ será una probabilidad condicional dependiente del instante anterior y la muestra generada en el paso $t-1$.

La principal ventaja de los métodos MCMC es que pueden ser aplicados a la casi totalidad de densidades que se encuentran en la práctica con extrema facilidad. Esta característica los hace particularmente interesantes desde el punto de vista de la optimización. De hecho, muchos algoritmos de búsqueda aleatoria son variantes de métodos MCMC.

Por otro lado, los principales inconvenientes son los siguientes:

1. Las muestras producidas son correlacionadas, dado que provienen de una cadena de Markov. Esto a parte de producir una perdida en términos de información, en ciertos casos, llega a limitar la aplicación de métodos MCMC. Por ejemplo, si la densidad objetivo tiene mucha masa de probabilidad concentrada en una moda, es posible que la cadena se quede atrapada en la misma, ralentizando mucho la convergencia a la densidad estacionaria.

2. Existe un transitorio, es decir, un periodo de convergencia a la densidad estacionaria (que coincide la densidad objetivo $p_o(\mathbf{x})$), denominado *burn in period*. Por tanto, suponiendo periodo de convergencia de M muestras, para generar N muestras válidas de la densidad objetivo $p_o(\mathbf{x})$, se necesitará generar $N + M$ muestras.

El Capítulo 4 se dedica a esta metodología. Se concentrará sobre todo en el estudio del algoritmo MCMC más conocido y utilizado: el *algoritmo Metropolis-Hasting*.

3.4.4. Importance Sampling

Dada una fuente aleatoria con densidad $\pi(\mathbf{x})$ ($x \in \mathcal{S} \subset \mathbb{R}^m$), hasta ahora se ha visto como modificar las muestras producidas por $\pi(\mathbf{x})$ a través

- ó de una adecuada transformación,
- ó de un test apropiado,
- ó de la creación de una cadena de Markov,

logrando otras muestras distribuidas de acuerdo a $p_o(\mathbf{x})$.

Otro enfoque, conocido como "importance sampling", consiste en asociar *pesos* a las muestras producidas por la densidad tentativa $\pi(\mathbf{x})$. El peso de una muestra representa la "importancia" de la misma. Es decir, pesos más altos serán asociados a muestras de mayor "valor", en términos de muestreo.

Esto significa que, "importance sampling" no es un generador de números aleatorios. Realmente es un método que aproxima la medida de probabilidad definida por la densidad objetivo $p_o(\mathbf{x})$. Es decir, "importance sampling" proporciona una densidad con *soporte aleatorio* (discreto) $h(\mathbf{x})$ que aproxima $p_o(\mathbf{x})$.

El método de "importance sampling" está compuesto por los siguientes pasos:

1. Generar N muestras $\mathbf{x}^{(i)}$, $i = 1, ..., N$, desde la densidad tentativa $\pi(\mathbf{x})$.

2. A cada muestra $\mathbf{x}^{(i)}$, $i = 1, ..., N$, se asocia un peso

$$\omega_i = \frac{p_o(\mathbf{x}^{(i)})}{\pi(\mathbf{x}^{(i)})}. \tag{3.12}$$

3. Finamente se normalizan los pesos

$$\bar{\omega}_i = \frac{\omega_i}{\sum_{i=1}^{N} \omega_i}, \tag{3.13}$$

de manera que $\sum_{i=1}^{N} \bar{\omega}_i = 1$.

La densidad $h(\mathbf{x})$ está compuesta por un tren de deltas, es decir,

$$h_N(\mathbf{x}) = \sum_{i=1}^{N} \bar{\omega}_i \delta(\mathbf{x} - \mathbf{x}^{(i)}), \tag{3.14}$$

donde con $\delta(\mathbf{x} - \mathbf{x}^{(i)})$ se indica la función delta de Dirac ($\delta(\mathbf{x} - \mathbf{x}^{(i)}) = +\infty$ si $\mathbf{x} = \mathbf{x}^{(i)}$ ó $\delta(\mathbf{x} - \mathbf{x}^{(i)}) = 0$ si $\mathbf{x} \neq \mathbf{x}^{(i)}$).

Claramente, $h_N(\mathbf{x})$ está definida para todo $\mathbb{R}^m$ así que su dominio es en efecto continuo. Pero, dado que toma valores no nulos solo en un conjunto discreto de puntos $\mathbf{x}^{(i)}$, se suele hablar de soporte discreto. También hay que evidenciar que este soporte es aleatorio, formado por las *partículas* (muestras) $\mathbf{x}^{(i)}$ generadas por la densidad tentativa $\pi(\mathbf{x})$.

Es necesario notar que, dado que se utilizan deltas como funciones base, no es del todo correcto afirmar que $h_N(\mathbf{x}) \to p_o(\mathbf{x})$ cuando $N \to +\infty$. Esta convergencia no puede ser garantizada en este caso. Sin embargo, se puede utilizar $h_N(\mathbf{x})$ para aproximar momentos de $p_o(\mathbf{x})$, o más en general, para el cálculo de integrales que suponen la utilización de $p_o(\mathbf{x})$. Por ejemplo, dada una variable aleatoria $\mathbf{X} = [X_1, ..., X_m] \in \mathbb{R}^m$ con densidad $p_o(\mathbf{x})$, se considera en el cálculo de la esperanza $E[g(\mathbf{X})]$ (se trata de una integral múltiple sobre $\mathcal{S} \subset \mathbb{R}^m$),

$$E[g(\mathbf{X})] = \int_{\mathcal{S}} g(\mathbf{x}) p_o(\mathbf{x}) d\mathbf{x}. \tag{3.15}$$

que esta esperanza puede aproximarse como

$$E[g(\mathbf{X})] \approx \sum_{i}^{N} \bar{\omega}_i g(\mathbf{x}^{(i)}), \tag{3.16}$$

donde $\mathbf{x}^{(i)} \sim \pi(\mathbf{x})$, $i = 1, ..., N$, y los pesos $\bar{\omega}_i$ están hallados como en las ecuaciones (3.12) y (3.13).

La Figura 3.2 muestra el funcionamiento del algoritmo "Importance Sampling".

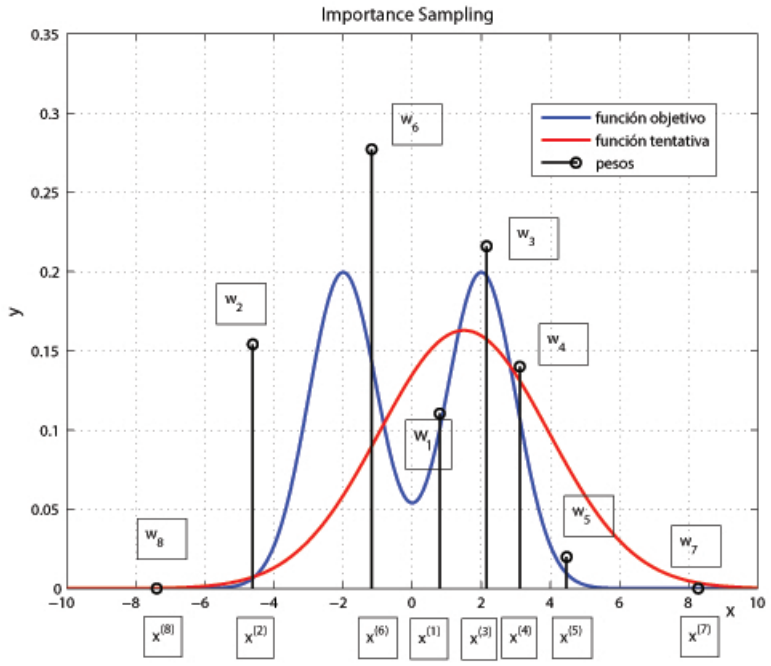

Figura 3.2. Funcionamiento de importance sampling. Se observa como cada muestra $x^{(i)}$ tomada tiene asignado un peso en función de la Ecuación vista en 3.13.

Es importante evidenciar que el peso asociado a la muestra $\mathbf{x}^{(i)}$ es proporcional al valor $p_o(\mathbf{x}^{(i)})$ e inversamente inversamente proporcional a $\pi(\mathbf{x}^{(i)})$. Es decir, claramente una muestra $\mathbf{x}'$ suele tener una mayor "importancia" si la probabilidad $p_o(\mathbf{x}^{(i)})$ es alta, y también si la probabilidad que la densidad tentativa $\pi(\mathbf{x}^{(i)})$ es baja (las muestras proporcionadas poco frecuentemente tienen mayor importancia porque conllevan mayor información).

3.4.5. Resampling

El "resampling" (remuestreo) es una tecnica (necesaria) utilizada sobre todo el en muestreo secuencial de variables aleatorias con importance sampling. Aquí nos interesa introducirlo para su conexión con los metodos de exploración estocásticas que utilizan varios puntos (varias partículas). Hay diferentes tipos de remuestreo. Nosotros nos concentraremos solo en el remuestreo multinomial.

Multinomial Resampling

Sustancialmente, se trata de muestrear (suele ser N veces) la variable aleatoria discreta W con masa de probabilidad los pesos

$$\text{Prob}\{W = i\} = \bar{\omega}_i,$$

o también se puede ver también como muestrear (N veces) una variable continua X con densidad un tren de deltas,

$$h_N(\mathbf{x}) = \sum_{i=1}^{N} \bar{\omega}_i \delta(\mathbf{x} - \mathbf{x}^{(i)}).$$

Claramente, las partículas con menor peso $\bar{\omega}_i$ tiene poca probabilidad de "*sobrevivir*" al resampling, mientras las partículas con mayor peso $\bar{\omega}_i$ serán "*replicadas/duplicadas*" con alta probabilidad después de un paso de resampling.

Como ejemplo consideremos $N = 3$ partículas, $x^{(1)} = 0{,}2$, $x^{(2)} = -2$ y $x^{(3)} = 7{,}6$ con pesos $\bar{\omega}_1 = 0{,}3$, $\bar{\omega}_1 = 0{,}5$, $\bar{\omega}_1 = 0{,}2$. Si muestreamos 3 veces la variable W logramos, por ejemplo, la secuencia

$$2, 2, 1...$$

es decir, las partículas después del resampling serán las siguientes

$$x_r^{(1)} \equiv x^{(2)} = -2, \quad x_r^{(2)} \equiv x^{(2)} = -2, \quad x_r^{(3)} \equiv x^{(1)} = 0{,}2.$$

La partícula $x^{(3)} = 7{,}6$ ha desaparecido mientras $x^{(2)}$ aparece 2 veces.

Capítulo 4

Algoritmo Metropolis-Hastings (MH)

4.1. Historia de las técnicas de Monte Carlo y algoritmos MCMC

Muchos investigadores consideran el algoritmo Metropolis-Hastings como uno de los 10 algoritmos más influyentes en el desarrollo de la ciencia del siglo XX [Beichl y Sullivan, 2000]. Este algoritmo forma parte de una amplia clase de algoritmos conocidos como Markov Chain Monte Carlo (MCMC). Esta clase de técnicas son métodos de Monte Carlo que utilizan cadenas de Markov, que como se sabe, han jugado y juegan un papel muy relevante en estadística, física e ingeniería durante los tres últimas décadas. Existen problemas en dimensiones muy altas, como por ejemplo, el cálculo del volumen de un cuerpo convexo, para los cuales la resolución mediante métodos MCMC es una de las mejores opciones para obtener una solución en un tiempo razonable (polinómico en función de la dimensión d). Los métodos MCMC nos permiten muestrear cualquier tipo de densidad por compleja que ésta sea, sin mas que saber evaluar la densidad objetivo $p_o(\mathbf{x})$ en un determinado punto. Por otro lado, se tiene el inconveniente de que las muestras generadas no son totalmente independientes entre ellas, sino que están correlacionadas.

Los métodos de Monte Carlo tuvieron su origen durante la II Guerra Mundial, a principios de los años 50 [Robert y Casella, 2008]. A pesar de que los métodos de Monte Carlo se utilizaban a menudo, no fue hasta 1970 con Hastings cuando se empezaron a usar de forma corriente en la práctica. El primer algoritmo MCMC más conocido es sin duda el algoritmo Metropolis, publicado por Metropolis en 1953. El descubrimiento de este algoritmo fue fruto del trabajo del mismo grupo de científicos que habían definido los métodos de Monte Carlo. La mayoría de ellos eran físicos que trabajaban en la bomba atómica.[1]

[1] La construcción de la bomba atómica no conllevo el uso de técnicas de simulación. Estas técnicas por el contrario si fueron utilizadas abundantemente para desarrollar la bomba de hidrogeno.

La historia cuenta que Stan Ulam, en 1946, se encontraba recuperándose de una larga enfermedad, cuando decidió explorar las posibilidades que ofrecía el juego del solitario (52 cartas), en cuanto a probabilidad de aparición de unas cartas u otras. Después de abordar el tema de las combinaciones de cartas de forma exhaustiva, decidió adoptar una postura más práctica, realizando simultáneamente varios "solitarios", y anotando el número de solitarios que finalizaban satisfactoriamente. Esta simple idea de seleccionar muestras estadística para resolver un problema combinacional duro, representa el punto de partida de los métodos de simulación de Monte Carlo actuales.

Stan Ulam pronto se dio cuenta de la potencia que podría llegar a tener este método usando ordenadores. Se podrían usar este tipo de métodos para resolver preguntas procedentes de la física, como por ejemplo, el problema de la difusión de los neutrones, o el cálculo de las configuraciones de menor energía para diversos compuestos químicos. Stan se puso en contacto con John Von Neumann (que también se dio cuenta rápidamente del potencial de la idea), y durante los años siguientes desarrollaron muchos métodos de Monte Carlo. A principios de los años 50, Enrico Fermi utilizó los algoritmos de Monte Carlo en el cálculo de la difusión de los neutrones, y más tarde diseño el FERMIAC, que era un dispositivo electrónico para realizar los cálculos rápidamente [Anderson, 1986].

Durante mucho tiempo, matemáticos y físicos acudieron a Stan Ulam con diferentes problemas en busca de solución (Fermi, Von Neumann, Ulam, Teller, Richtmyer, Bethe, Feynman, Gamow). En 1949, Nick Metropolis junto con Stan Ulam publicaron el primer documento accesible a todos sobre simulación, donde se usaban los métodos de Monte Carlo [Metropolis y Ulam, 1949].

A partir del año 1953, aparecieron muchos documentos sobre simulación por Monte Carlo. Desde el punto de vista de la inferencia estadística, la contribución más significativa fue la generalización del algoritmo Metropolis realizada por Hastings en 1970. Hastings y su estudiante Peskun demostraron que el algoritmo Metropolis, y de forma más general el algoritmo Metropolis-Hastings (MH), son instancias particulares de una gran familia de algoritmos (MCMC). En los años 80, dos documentos importantes sobre MCMC aparecieron en el campo de la computación y la inteligencia artificial [Geman y Geman, 1984] y [Pearl, 1984].

Este capítulo se centrará en describir en el algoritmo MH. Para ello, se introducirán primero las ideas básicas sobre cadenas de Markov. Además, es importante subrayar que aunque todas las técnicas que se tratarán a continuación puede ser utilizadas en un espacio de dimensión genérica $\mathbb{R}^m$, se ha preferido considerar el caso escalar $x \in \mathbb{R}$ para simplificar la notación y las gráficas. Esto sin perdida de generalidad, porque la extensión al caso multidimensional es inmediata.

4.2. Cadenas de Markov

Una cadena de Markov $\{X_t\}_{t=0}^{+\infty}$, $t \in \mathbb{N}$, es un proceso estocástico discreto $\{X_0, X_1, ...\}$, con la propiedad de que la densidad de la variable aleatoria $X_t \in \mathbb{R}$, dados todos los valores previos $\{X_1, X_2, ..., X_{t-1}\}$, depende solamente de la variable aleatoria X_{t-1} en el paso anterior. Así que se puede escribir

$$\text{Prob}[X_t \in \mathcal{A}|X_0, X_1, ..., X_{t-1}] = \text{Prob}[X_t \in A|X_{t-1}] \tag{4.1}$$

para cualquier conjunto $\mathcal{A} \subseteq \mathbb{R}$, y donde $\text{Prob}(\cdot|\cdot)$ denota una probabilidad condicional.

En esta sección, se introducirán brevemente los conceptos básicos sobre cadenas de Markov necesarios para la comprensión de los algoritmos MCMC (y más en concreto del algoritmo MH). Para ello, primero se tratará el caso de las cadenas de Markov con un número finito de estados (cadenas discretas), y a continuación, se estudiará el caso de número infinito de estados con valores continuos (cadenas continuas). En ambos casos, la transición temporal la consideraremos discreta $t \in \mathbb{N}$. Esto es, los tiempos evaluados serán el tiempo $t = 1, 2, 3...$, y no de forma continua.

4.2.1. Cadenas discretas

Dada una variable aleatoria X_t que puede tomar valores en un conjunto de estados finito, por ejemplo $\mathcal{S} = \{1, 2, ..., m\}$, se llama cadena de Markov al proceso estocástico X_t tal que

$$p(x_t|x_{t-1}, ..., x_1) = p(x_t|x_{t-1}), \tag{4.2}$$

es decir el valor que puede tomar X_t depende solo de la variable aleatoria X_{t-1} al paso $t-1$. Dado que la cadena de Markov está perfectamente identificada por la probabilidad condicional $p(x_t|x_{t-1})$, a continuación se definirá el *kernel* de una cadena de Markov como

$$K(x_t|x_{t-1}) \triangleq p(x_t|x_{t-1}).$$

Claramente se verifica que $\sum_{i=1}^{m} K(x_t = i|x_{t-1}) = 1$ para cualquier t. Una cadena se define *homogenea* si la $K(x_t|x_{t-1})$ permanece invariante para todo t.

En el caso que el numero de estado es finito, el kernel $K(x_t|x_{t-1})$ realmente define una *matriz de de transición*. Como ejemplo, se considera una cadena de Markov con tres estados con un gráfico asociado mostrado en Figura 4.1. Se considera la siguiente matriz de transición

$$\mathbf{K} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0{,}1 & 0{,}9 \\ 0{,}6 & 0{,}4 & 0 \end{bmatrix}. \tag{4.3}$$

Cada valor indica una probabilidad de transición. Por ejemplo dada la posición (i, j), el valor correspondiente es la probabailidad de pasar de estado i al estado j $(i \to j)$. Notese que todas las lineas suman 1.

Esta matriz puede visualizarse a través del grafo mostrado en Figura 4.1, donde cada rama representa las posibles transiciones que puede ocurrir dado que la variable aleatoria $X_{t-1} = i$ con $i = 1, ..., m$.

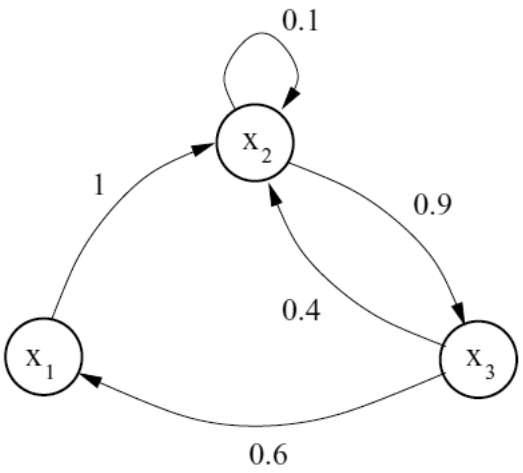

Figura 4.1. Cadenas de Markov. Ejemplo sobre el uso de cadenas de Markov usando un el grafo de transiciones.

Si por ejemplo, el vector de probabilidades para el estado inicial es $\mathbf{p_0} = [0.5, 0.2, 0.3]$, se puede escribir el vector de probabilidades en $t = 1$ como

$$\mathbf{p_1} = \mathbf{p_0}\mathbf{K} \text{ obteniendo } \mathbf{p_1} = [0.2,0.6,0.2]. \tag{4.4}$$

Siguiendo el mismo mecanismo para un instante de tiempo genérico t se obtiene

$$\mathbf{p_t} = \mathbf{p_{t-1}}\mathbf{K} = (\mathbf{p_{t-2}}\mathbf{K})\mathbf{K} = ... = \mathbf{p_0}\mathbf{K}^t. \tag{4.5}$$

Cuando $t \to \infty$, se obtiene que en ciertos casos, $\mathbf{p_t} \to \mathbf{p_e}$. Es decir,

$$\lim_{t\to\infty} \mathbf{p_t} = \mathbf{p_e}. \tag{4.6}$$

Este vector $\mathbf{p_e}$, denominado distribución invariante o estacionaria juega un papel fundamental en los algoritmos MCMC. Para cualquier punto (estado) de comienzo, la cadena convergerá a la distribución invariante $\mathbf{p_e}$, siempre y cuando la matriz de transición $\mathbf{K}$ sea ergódica

Es decir, que cumpla las siguientes propiedades:

1. **Irreductibilidad**. Para cualquier estado de la cadena de Markov, existe una probabilidad positiva de visitar todos los demás estados. Esto se traduce en que la matriz de transición **K** no puede ser separada en matrices más pequeñas, que es lo mismo que afirmar que el grafo de transiciónes es conexo.

2. **Aperiodicidad**. La cadena no debe quedar atrapada en ciclos.

Además si consideramos el problema de autovalores y autovectores

$$\mathbf{x} = \lambda \mathbf{x}\mathbf{K}, \tag{4.7}$$

Es notorio que existen N (dimensión de la matriz $\mathbf{K}$) valores en el campo complejo $\lambda_i \in \mathbb{C}$, $i = 1, ..., N$, que verifican la Ecuación (4.7). Desde el punto de vista espectral se puede notar que $\mathbf{p_e}$ es un autovector de la matriz $\mathbf{K}$ asociado al autovalor unitario ($\lambda_j = 1$ con $j \in \{1, ..., N\}$). De hecho, el teorema de Perron-Frobenius [Brualdi, 1991] de álgebra lineal básica dice que el resto de autovalores tienen un valor absoluto menor que uno. El segundo autovalor con mayor valor absoluto determinará la velocidad de convergencia de la cadena. Específicamente, a menor valor absoluto de este segundo autovalor, mayor será la velocidad de convergencia.

4.2.2. Cadenas a valores continuos

Las variables aleatorias $\{X_t, t = 0, 1..., +\infty\}$ que componen la cadena de Markov pueden tomar valores continuos (es decir, $X_t \in \mathbb{R}$). Al igual que en el caso discreto, se deben cumplir las condiciones de irreductibilidad y aperiodicidad para que la cadena converja a una densidad invariante $p_e(x)$. En este caso, la densidad invariante se define como

$$p_e(x_t) = \int_{\mathcal{S}} K(x_t|x_{t-1}) p_e(x_{t-1}) dx_{t-1} \tag{4.8}$$

donde $K(x_t|x_{t-1})$ es el kernel de la cadena y representa siempre la densidad condicional de x_t dado x_{t-1}.

Una condición necesaria pero no suficiente para asegurar que una determinada $p_e(x)$ es la función de distribución invariante, se denomina *condición de reversibilidad o de balance* y se expresa como

$$K(x_t|x_{t-1}) p_e(x_{t-1}) = K(x_{t-1}|x_t) p_e(x_t). \tag{4.9}$$

De hecho si integramos ambos lados por x_{t-1}

$$\int_{\mathcal{S}} K(x_t|x_{t-1}) p_e(x_{t-1}) dx_{t-1} = \int_{\mathcal{S}} K(x_{t-1}|x_t) p_e(x_t) dx_{t-1}, \tag{4.10}$$

podemos sacar $p_e(x_t)$ a segundo miembro porque no depende de x_{t-1}

$$\int_{\mathcal{S}} K(x_t|x_{t-1})p_e(x_{t-1})dx_{t-1} = p_e(x_t)\int_{\mathcal{S}} K(x_{t-1}|x_t)dx_{t-1}, \tag{4.11}$$

y dado que $\int_{\mathcal{S}} K(x_{t-1}|x_t)dx_{t-1} = 1$ llegamos

$$\int_{\mathcal{S}} K(x_t|x_{t-1})p_e(x_{t-1})dx_{t-1} = p_e(x_t). \tag{4.12}$$

Las Ecuaciones (4.8) y (4.12) son identicas.

4.3. Algoritmos MCMC

Se considere ahora una densidad objetivo $p_o(x)$ que no se puede muestrear directamente con otros métodos (como el método de inversión descrito en el Capitulo 3).

Los métodos MCMC son métodos de generación de números aleatorios que producen una cadena de Markov con densidad invariante $p_e(x)$ igual a la densidad objetivo $p_o(x)$. En otras palabras, MCMC genera una secuencia de muestras $\{x_t\}_{t=0}^{N}$, correlacionadas entre ellas, cuya distribución converje a la densidad deseada $p_o(x)$.

Los métodos MCMC tienen una estructura común que podría resumirse en dos pasos:

1. **Generación:** se propone un punto candidato x' como posible nuevo paso x_t. En general, este paso depende de la muestra generada en el instante anterior x_{t-1}.

2. **Decisión:** se acepta el punto candidato x' con una determinada probabilidad. Dicha probabilidad de aceptación $\mathcal{A}(x_{t-1}, x')$ puede ser siempre uno, como en los algoritmos *Gibbs Sampling* y *Slice sampling*, o distinta de uno. Si el punto es aceptado $x_t = x'$ sino, con probabilidad $1 - \mathcal{A}(x_{t-1}, x')$, se considera $x_t = x_{t-1}$.

Un esquema de lo expuesto se muestra en la Figura 4.2.

El resto de este capitulo se tratará el método MCMC sin duda más relevante: el algoritmo Metropolis-Hastings.

4.4. El algoritmo Metropolis-Hastings

El algoritmo Metropolis-Hastings (MH) es el algoritmo MCMC más popular [Hastings, 1970] y [Metropolis et al., 1953]. Una razón es que otros algoritmos MCMC pueden ser interpretados como casos especiales de esta técnica.

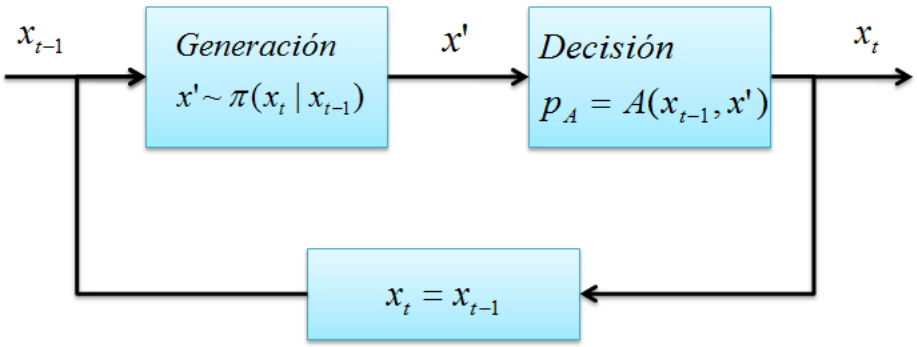

Figura 4.2. Esquema de un algoritmo MCMC. Dada una muestra x_{t-1}, se propone un punto candidato $x' \sim \pi(x_t|x_{t-1})$. A continuación, el movimiento $x_{t-1} \to x_t = x'$ se acepta con una probabilidad dada $\mathcal{A}(x_{t-1}, x')$. En caso contrario se considera $x_t = x_{t-1}$.

Se asume que se sabe muestrear fácilmente la densidad tentativa $\pi(x_t|x_{t-1})$ (que en este caso es una densidad condicional dependiente del estado anterior), que va a proporcionar los puntos candidatos y se recuerda que la probabilidad de aceptación de un punto candidato vendrá indicada como $\mathcal{A}(x_{t-1}, x')$.

El algoritmo Metropolis-Hastings está formado por los siguientes pasos:

1. A partir de la densidad tentativa $\pi(x_t|x_{t-1})$, se genera un candidato $x' \sim \pi(x_t|x_{t-1})$ de acuerdo a dicha densidad.

2. Se calcula
$$\mathcal{A}(x_{t-1}, x') = \text{mín}\left[1, \frac{\pi(x_{t-1}|x')p(x')}{\pi(x'|x_{t-1})p(x_{t-1})}\right], \tag{4.13}$$
donde $p(x) \propto p_o(x)$ (recordamos que $p_o(x)$ es nuestra densidad objetivo).

3. Se genera una muestra uniformemente en el intervalo $[0, 1]$, es decir $u' \sim \mathcal{U}([0, 1])$.

4. El estado siguiente se define como
$$x_t = \begin{cases} x' & \text{si } u' \leq \mathcal{A}(x_{t-1}, x') \\ x_{t-1} & \text{si } u' > \mathcal{A}(x_{t-1}, x') \end{cases}. \tag{4.14}$$

La estructura en pseudo-código para generar N muestras de la cadena se encuentra en el Cuadro 4.1.

Es importante añadir que el algoritmo necesita de un periodo transitorio o "*burn in period*" para converger. Es decir, se necesitan de una serie de pasos temporales para obtener muestras

Algoritmo Metropolis-Hastings
1: Se elige de forma aleatoria el punto de comienzo x_0, y se inicializa $i = 1$. Sea N el número total de muestras deseadas y $p(x) \propto p_o(x)$.
2: Se muestrea $u' \sim \mathcal{U}[0,1]$ y se genera un punto candidato $x' \sim \pi(x_t|x_{t-1})$.
3: Se calcula $\mathcal{A}(x_{t-1}, x') = \text{mín}\left[1, \frac{\pi(x_{t-1}|x')p(x')}{\pi(x'|x_{t-1})p(x_{t-1})}\right]$.
4: Si $u' \leq \mathcal{A}(x_{t-1}, x')$, se considera $x_t = x'$ y $x^{(i)} = x_t$, $i = i + 1$.
5: Si $u' > \mathcal{A}(x_{t-1}, x')$, se considera $x_t = x_{t-1}$ y $x^{(i)} = x_t$, $i = i + 1$.
6: Si $i = N$, finalizar. En caso contrario volver al paso 2.

Cuadro 4.1

válidas (cuya densidad coincida con la densidad estacionaria $p_e(x)$), por lo cual habrá que descartar varias muestras al comienzo del algoritmo. Suponiendo un "*burn-in period*" de M muestras, para conseguir N muestras realmente válidas se deberán generar $N + M$ muestras y desechar las M primeras. No es una tarea fácil decidir cuanta muestras hay que descartar, para ello hay muchos estudios en la literatura ([Liu, 2004] y [Fan et al., 2009]).

En la Figura 4.3 se observa el funcionamiento del algoritmo Metropolis-Hastings en diferentes instantes de tiempo. En cada paso hay dos posibilidades: quedarse en el mismo estado o aceptar el nuevo movimiento.

Claramente el algoritmo MH produce una cadena de Markov. La *probabilidad de transición* que define esta cadena, es decir su *kernel*, es

$$K(x_t|x_{t-1}) = \pi(x_t|x_{t-1})\mathcal{A}(x_{t-1}, x_t) + \delta(x_t - x_{t-1})(1 - r(x_{t-1})). \tag{4.15}$$

El primer termino a la derecha del signo igual $\pi(x_t|x_{t-1})\mathcal{A}(x_{t-1}, x_t)$, se refiere a la probabilidad de seleccionar $x_t = x' \sim \pi(x_t|x_{t-1})$ y aceptar el movimiento propuesto (de x_{t-1} a x_t) con probabilidad $\mathcal{A}(x_{t-1}, x_t)$.

El segundo termino $\delta(x_t - x_{t-1})r(x_{t-1})$, se refiere a la probabilidad de descartar el movimiento seleccionado y quedarse en el estado x_{t-1}, es decir, que $x_t = x_{t-1}$. La función $\delta(x_t - x_{t-1})$ representa una delta de Dirac en x_{t-1} (es 1 si $x_t = x_{t-1}$ y 0 si $x_t \neq x_{t-1}$) y $(1 - r(x_{t-1}))$ es la probabilidad de rechazo de un candidato genérico x' (indicamos con $r(x_{t-1})$ la probabilidad de aceptación de un candidato genérico x'), seleccionado cuando el estado anterior es x_{t-1}. Es decir, dado $x' \sim \pi(x'|x_{t-1})$, la probabilidad de aceptar esta muestra candidata será $\mathcal{A}(x_{t-1}, x')$,

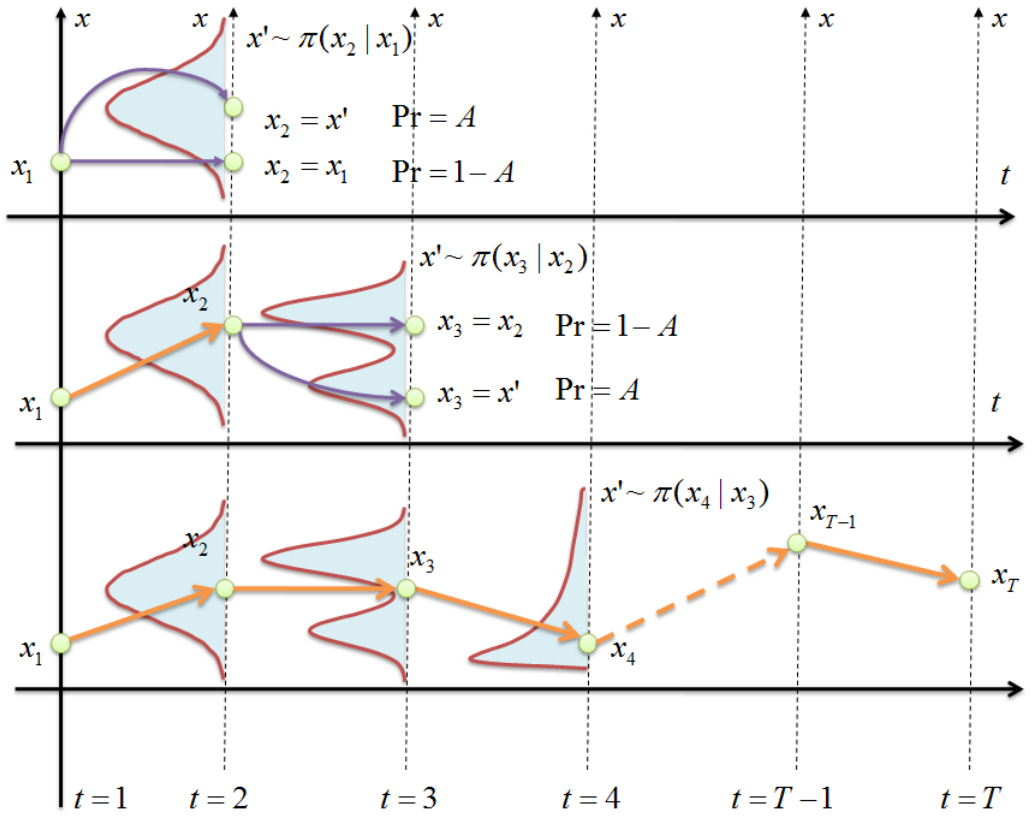

Figura 4.3. Funcionamiento del algoritmo MH. Hay que notar que en cada instante se muestrea de una densidad tentativa $\pi(x_t|x_{t-1})$ diferente que depende del estado anterior x_{t-1}. Luego se decide si se acepta o rechaza la muestra x' con una probabilidad $\mathcal{A}(x_{t-1}, x')$. Si el movimiento a x' es descartado la cadena se queda en el mismo estado $x_t = x_{t-1}$.

y $r(x_{t-1})$ será definido como

$$r(x_{t-1}) = \int_{\mathcal{S}} \pi(x'|x_{t-1})\mathcal{A}(x_{t-1}, x')dx'. \tag{4.16}$$

Dicho de otra forma, $r(x_{t-1})$ es el valor medio de la probabilidad de aceptación cuando la cadena se encuentra en el estado x_{t-1}. Un ejemplo sobre lo expuesto puede verse en la Figura 4.4.

Se puede demostrar que las muestras generadas por MH convergen (después del transitorio) a las que se generarían con la densidad objetivo $p_o(x)$. De hecho, el kernel del MH visto en la Ecuación de reversibilidad (4.9), se reescribe por comodidad abajo

$$K(x_t|x_{t-1})p_o(x_{t-1}) = K(x_{t-1}|x_t)p_o(x_t).$$

Y se puede notar que

$$\delta(x_t - x_{t-1})(1 - r(x_{t-1})) = \delta(x_{t-1} - x_t)(1 - r(x_t)), \tag{4.17}$$

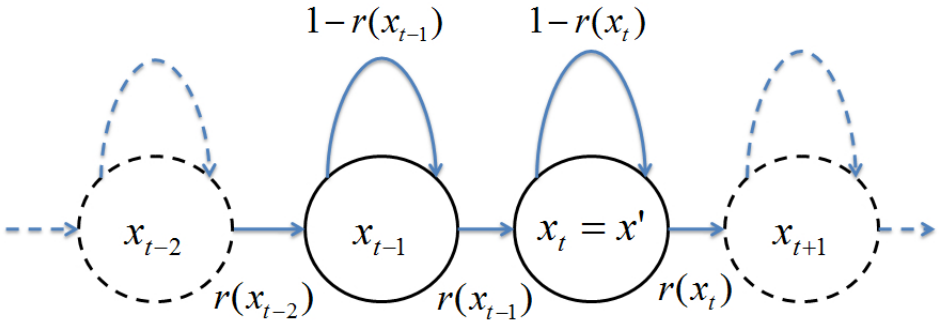

Figura 4.4. Transición entre estados. Dado x_{t-1}, la probabilidad de aceptar un punto candidato viene dada por $r(x_{t-1})$, mientras que la probabilidad media de mantenerse en el mismo estado es la complementaria $1 - r(x_{t-1})$.

dado que las dos deltas de Dirac son no nulas solo si $x_{t-1} = x_t$. Así, considerando las Ecuaciones (4.15) y (4.17) sólo tiene que verificarse la ecuación de balance *parcial*

$$\pi(x_t|x_{t-1})\mathcal{A}(x_{t-1}, x_t)p_o(x_{t-1}) = \pi(x_{t-1}|x_t)\mathcal{A}(x_t, x_{t-1})p_o(x_t), \tag{4.18}$$

donde se considera sólo el primer termino del kernel $K(x_t|x_{t-1})$ del MH. En este caso (si se cumple la Ecuación (4.18)), el kernel $K(x_t|x_{t-1})$ es irreducible y aperiódico, así que $p_o(x)$ es la unica densidad invariante para $K(x_t|x_{t-1})$. Para ello, se debe demostrar el siguiente resultado.

Teorema: La ecuación de balance parcial

$$\pi(x_t|x_{t-1})\mathcal{A}(x_{t-1}, x_t)p_o(x_{t-1}) = \pi(x_{t-1}|x_t)\mathcal{A}(x_t, x_{t-1})p_o(x_t),$$

es verificada cuando la función $\mathcal{A}(x_{t-1}, x_t)$ es la del algoritmo MH.

Demostración: A continuación, vamos a considerar $p(x) = p_o(x)$ por mayor claridad. Se recuerda que las expresiones de $\mathcal{A}(x_{t-1}, x_t)$ y $\mathcal{A}(x_t, x_{t-1})$ se definen como

$$\mathcal{A}(x_{t-1}, x_t) = \text{mín}\left[1, \frac{\pi(x_{t-1}|x_t)p_o(x_t)}{\pi(x_t|x_{t-1})p_o(x_{t-1})}\right],$$
$$\mathcal{A}(x_t, x_{t-1}) = \text{mín}\left[1, \frac{\pi(x_t|x_{t-1})p_o(x_{t-1})}{\pi(x_{t-1}|x_t)p_o(x_t)}\right].$$

Para realizar la demostración completa es necesario distinguir ente los siguientes casos:

1. cuando $\pi(x_{t-1}|x_t)p_o(x_t) < \pi(x_t|x_{t-1})p_o(x_{t-1})$,
2. y cuando $\pi(x_{t-1}|x_t)p_o(x_t) \geq \pi(x_t|x_{t-1})p_o(x_{t-1})$.

En el primer caso ($\pi(x_{t-1}|x_t)p_o(x_t) < \pi(x_t|x_{t-1})p_o(x_{t-1})$) las funciones de aceptación quedan como

$$\mathcal{A}(x_{t-1}, x_t) = \text{mín}\left[1, \frac{\pi(x_{t-1}|x_t)p_o(x_t)}{\pi(x_t|x_{t-1})p_o(x_{t-1})}\right] = \frac{\pi(x_{t-1}|x_t)p_o(x_t)}{\pi(x_t|x_{t-1})p_o(x_{t-1})},$$
$$\mathcal{A}(x_t, x_{t-1}) = \text{mín}\left[1, \frac{\pi(x_t|x_{t-1})p_o(x_{t-1})}{\pi(x_{t-1}|x_t)p_o(x_t)}\right] = 1.$$

Si se sustituyen estas funciones de aceptación en la Ecuación (4.18), se obtiene

$$\pi(x_t|x_{t-1})\frac{\pi(x_{t-1}|x_t)p_o(x_t)}{\pi(x_t|x_{t-1})p_o(x_{t-1})}p_o(x_{t-1}) = \pi(x_{t-1}|x_t)\cdot 1\cdot p_o(x_t), \tag{4.19}$$

y con cálculos triviales se llega a

$$\pi(x_{t-1}|x_t)p_o(x_t) = \pi(x_{t-1}|x_t)p_o(x_t). \tag{4.20}$$

Esto significa que el kernel del algoritmo MH cumple la ecuación de balance cuando se verifica que $\pi(x_{t-1}|x_t)p_o(x_t) < \pi(x_t|x_{t-1})p_o(x_{t-1})$. Resta comprobar el caso en el que se verifica que $\pi(x_{t-1}|x_t)p_o(x_t) \geq \pi(x_t|x_{t-1})p_o(x_{t-1})$. En este segundo caso, las funciones de aceptación quedan como

$$\mathcal{A}(x_{t-1}, x_t) = \text{mín}\left[1, \tfrac{\pi(x_{t-1}|x_t)p_o(x_t)}{\pi(x_t|x_{t-1})p_o(x_{t-1})}\right] = 1 \tag{4.21}$$

$$\mathcal{A}(x_t, x_{t-1}) = \text{mín}\left[1, \tfrac{\pi(x_t|x_{t-1})p_o(x_{t-1})}{\pi(x_{t-1}|x_t)p_o(x_t)}\right] = \tfrac{\pi(x_t|x_{t-1})p_o(x_{t-1})}{\pi(x_{t-1}|x_t)p_o(x_t)} \tag{4.22}$$

Se sustituyen estas funciones de aceptación en la Ecuación (4.18) obteniéndose

$$\pi(x_t|x_{t-1})\cdot 1\cdot p_o(x_{t-1}) = \pi(x_{t-1}|x_t)\tfrac{\pi(x_t|x_{t-1})p_o(x_{t-1})}{\pi(x_{t-1}|x_t)p_o(x_t)}p_o(x_t), \tag{4.23}$$

$$\pi(x_t|x_{t-1})p_o(x_{t-1}) = \pi(x_t|x_{t-1})1p_o(x_{t-1}). \tag{4.24}$$

Es decir, también en este segundo caso (cuando $\pi(x_{t-1}|x_t)p_o(x_t) \geq \pi(x_t|x_{t-1})p_o(x_{t-1})$), se cumple la ecuación de balance. □

Dado que la densidad $p_o(x)$ satisface la ecuación de balance respecto al kernel $K(x_t|x_{t-1})$ definido en la Ecuación (4.15), se puede afirmar que la densidad $p_o(x)$ es estacionaria. Es decir, $p_e(x) = p_o(x)$, estando las muestras x_t generadas con el algoritmo MH (después de un periodo transitorio o "*burn in period*") distribuidas de acuerdo a la densidad objetivo $p_o(x)$.

Llegados a este punto, es importante resaltar que algunos autores utilizan probabilidades de aceptación distintas a la que se utiliza en el MH.

4.5. Otras funciones de aceptación

La función de aceptación dada por el algoritmo Metropolis-Hastings no es la única posible. De hecho, diversos autores han diseñado funciones de aceptación $A(x, y) = \{(x, y) \in \mathcal{S}\times\mathcal{S} \to [0, 1]\}$ que cumplen la condición de balance de la Ecuación (4.9). A continuación, vamos a considerar $p(x) = p_o(x)$ sin perdida de generalidad.

Se recuerda que la probabilidad de aceptación de Metropolis-Hastings es de la forma

$$\mathcal{A}_1(x, y) = \text{mín}\left[1, \frac{\pi(x|y)p_o(y)}{\pi(y|x)p_o(x)}\right],$$

donde se utiliza en subíndice 1 para diferenciarla de las siguientes funciones. Primero se observa que, en el caso de que la función tentativa $\pi(y|x)$ sea simétrica, es decir $\pi(y|x) = \pi(x|y)$, la función de aceptación queda como

$$\mathcal{A}_{1s}(x, y) = \text{mín}\left[1, \frac{p_o(y)}{p_o(x)}\right]. \tag{4.25}$$

Por otro lado, otra función de aceptación que respeta la ecuación de balance es, por ejemplo,

$$\mathcal{A}_2(x, y) = \frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x) + p_o(y)\pi(x|y)}. \tag{4.26}$$

Esta función fue propuesta por Barker [Barker, 1965] en el 1965. Si la función tentativa es simétrica, $\pi(y|x) = \pi(x|y)$, la función de aceptación de Barker queda como

$$\mathcal{A}_{2s}(x, y) = \frac{p_o(y)}{p_o(x) + p_o(y)}. \tag{4.27}$$

Es importante remarcar otra vez que, tratándose de una probabilidad (de aceptación), siempre se tiene $0 \leq \mathcal{A}(x, y) \leq 1$, para todas las parejas $\{(x, y) \in \mathcal{S} \times \mathcal{S}\}$.

Hastings logró una fórmula que engloba ambas probabilidades de aceptación descritas anteriormente,

$$\mathcal{A}_3(x, y) = \frac{c(x, y)}{1 + \frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}} \tag{4.28}$$

donde la función $c(x, y)$ tiene que ser

1. simétrica $c(x, y) = c(y, x)$,
2. no negativa, $c(x, y) \geq 0$,
3. y $c(x, y) \leq 1 + \frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}$, de manera que $0 \leq \mathcal{A}(x, y) \leq 1$.

Si, por ejemplo, $c(x, y) = 1$

$$\mathcal{A}_3(x, y) = \frac{1}{1 + \frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}} = \frac{p_o(y)\pi(x|y)}{p_o(y)\pi(x|y) + p_o(x)\pi(y|x)}, \tag{4.29}$$

que es la función de aceptación de Barker en la Ecuación (4.26), es decir $\mathcal{A}_3(x, y) \equiv \mathcal{A}_2(x, y)$. Además, si

$$c(x, y) = \text{mín}\left[1 + \frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}, 1 + \frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)}\right], \tag{4.30}$$

se obtiene la función de aceptación de Metropolis-Hastings vista en la Ecuación (4.13), es decir $\mathcal{A}_3(x, y) \equiv \mathcal{A}_1(x, y)$. De hecho, si por ejemplo $p_o(x)\pi(y|x) < p_o(y)\pi(x|y)$ se tiene que

$$c(x, y) = \text{mín}\left[1 + \frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}, 1 + \frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)}\right] = 1 + \frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)},$$

y la función de aceptación queda

$$\mathcal{A}_3(x,y) = \frac{1+\frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}}{1+\frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}} = 1.$$

Si, por otro lado, $p_o(x)\pi(y|x) \geq p_o(y)\pi(x|y)$,

$$c(x,y) = \text{mín}\left[1+\frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}, 1+\frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)}\right] = 1+\frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)},$$

y la función de aceptación puede expresarse como

$$\mathcal{A}_3(x,y) = \frac{1+\frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)}}{1+\frac{p_o(x)\pi(y|x)}{p_o(y)\pi(x|y)}} = \frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)} \cdot \underbrace{\frac{p_o(x)\pi(y|x)+p_o(y)\pi(x|y)}{p_o(y)\pi(x|y)+p_o(x)\pi(y|x)}}_{1},$$

$$\mathcal{A}_3(x,y) = \frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)}.$$

De modo que finalmente si $c(x,y)$ tiene la forma de la Ecuación (4.30) la función $\mathcal{A}_3(x,y)$ se puede escribir como

$$\mathcal{A}_3(x,y) \equiv \mathcal{A}_1(x,y) = \text{mín}\left[1, \frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)}\right]. \tag{4.31}$$

Otra opción fue dada por Charles Stein (en una comunicación privada)

$$\mathcal{A}_4(x,y) = \frac{c(x,y)}{p_o(x)\pi(y|x)}, \tag{4.32}$$

donde $c(x,y)$ es una la función simétrica con las características descritas anteriormente. Además, otra opción sería también

$$\mathcal{A}_5(x,y) = \frac{p_o(y)c(x,y)}{\pi(y|x)}. \tag{4.33}$$

Finalmente, es interesante conocer como se pueden hallar otras funciones que cumplan la condición de reversibilidad. Para ello, se considera la función

$$R(x,y) \triangleq \frac{p_o(y)\pi(x|y)}{p_o(x)\pi(y|x)}, \tag{4.34}$$

y una transformación genérica $F(z) : [0,+\infty) \to [0,1]$ que verifica la condición

$$F(z) = zF(1/z). \tag{4.35}$$

Una posible (genérica) función de aceptación $\mathcal{A}_g(x,y)$ es definida como

$$\mathcal{A}_g(x,y) \triangleq (F \circ R)(x,y) = F(R(x,y)). \tag{4.36}$$

Se puede demostrar que $\mathcal{A}_g(x,y)$ cumple la ecuación de balance. La Ecuación (4.36) define una *clase* de funciones de aceptación, es decir, no todas las posibles funciones $\mathcal{A}(x,y)$ tienen que poderse expresar como en la Ecuación (4.36).

Nombre	Función	Notas
Metropolis	$\mathcal{A}_{1s}(x,y) = \text{mín}\left[1, \frac{p_o(y)}{p_o(x)}\right]$	Función tentativa simétrica
Metropolis-Hastings	$\mathcal{A}_1(x,y) = \text{mín}\left[1, \frac{\pi(x\|y)p_o(y)}{\pi(y\|x)p_o(x)}\right]$	
Barker	$\mathcal{A}_2(x,y) = \frac{p_o(y)\pi(x\|y)}{p_o(x)\pi(y\|x)+p_o(y)\pi(x\|y)}$	
Hastings Extendida	$\mathcal{A}_3(x,y) = \frac{c(x,y)}{1+\frac{p_o(x)\pi(y\|x)}{p_o(y)\pi(x\|y)}}$	$c(x,y)$ simétrica
Stein (1)	$\mathcal{A}_4(x,y) = \frac{c(x,y)}{p_o(x)\pi(y\|x)}$	$c(x,y)$ simétrica
Stein (2)	$\mathcal{A}_5(x,y) = \frac{p_o(y)c(x,y)}{\pi(y\|x)}$	$c(x,y)$ simétrica
Una clase genérica	$\mathcal{A}_g(x,y) \triangleq F(R(x,y))$	$R(x,y) \triangleq \frac{p_o(y)\pi(x\|y)}{p_o(x)\pi(y\|x)}, F(z) = zF(1/z)$

Cuadro 4.2. Tabla resumen. Funciones de aceptación estudiadas.

Podemos hallar la función de Barker $\mathcal{A}_2(x,y)$ eligiendo $F(z) = \frac{z}{1+z}$ mientras que utilizando $F(z) = \text{mín}[1,z]$, volvemos a la probabilidad de aceptación $\mathcal{A}_1(x,y)$ del algoritmo MH estándar.

Por ultimo, es interesante observar, que para el caso de estados discretos, ha sido demostrado en [Peskun, 1973] que la función de aceptación original del MH $\mathcal{A}_1(x,y)$ resulta ser la elección óptima en términos de eficiencia estadística.

En el Cuadro 4.2 se hace un resumen de las funciones de aceptación vistas hasta el momento.

4.6. Elección de la función tentativa

Las prestaciones del algoritmo MH dependen considerablemente de la función tentativa $\pi(x_t|x_{t-1})$. Existen un número infinito de posibilidades para elegir la densidad tentativa. Una elección muy común es utilizar una distribución Gaussiana centrada en el valor anterior, o bien una distribución de Cauchy, la cual tiene colas más pesadas, que facilita saltos muy largos ocasionalmente. Ambas distribuciones son simétricas, de manera que calcular la probabilidad de aceptación, como se verá, es menos costoso.

La Figura 4.5 muestra como varían las prestaciones de MH cambiando la densidad $\pi(x_t|x_{t-1})$. En el ejemplo mostrado en figura se considera una densidad tentativa Gaussiana de la forma $\pi(x_t|x_{t-1}) \propto \exp\{-(x_t - x_{t-1})^2/(2\sigma^2)\}$. Modificando tan solo la varianza, la probabilidad de rechazo puede hacerse muy elevada dando lugar a correlaciones muy altas.

Se puede ver que cuando la varianza toma el valor $\sigma^2 = 100$ el porcentaje de rechazo es muy elevado (la cadena repite muchas muestras). Por otro lado, con un valor de la desviación típica

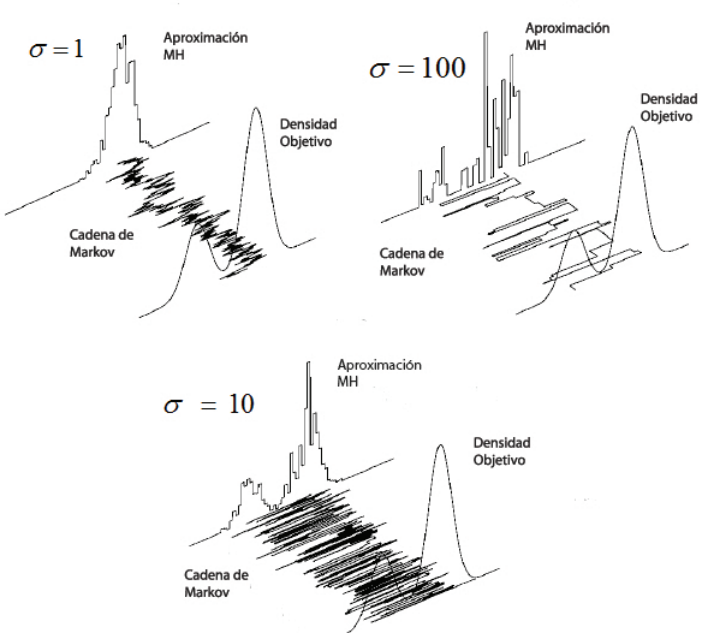

Figura 4.5. MH utilizando diferentes varianzas. Aproximación mediante MH con densidad tentativa gaussiana. Se muestran los resultados para 3 gaussianas de varianzas 1, 10 y 100.

muy bajo ($\sigma = 1$) la cadena se queda enganchada en una moda de la densidad objetivo $p_o(x)$ debida la excesiva correlación entre las muestras. Esto claramente ralentiza la convergencia a la densidad estacionaria (es decir, el transitorio es más largo). Sin embargo, con un valor intermedio de la desviación típica ($\sigma = 10$), el histograma formado por las muestras producidas por el MH consigue una estimación relativamente buena de la densidad objetivo $p_o(x)$.

A continuación, se considerarán unos casos particulares interesantes.

4.6.1. Función tentativa independiente

Un caso particular del algoritmo MH muy interesante, es cuando la densidad tentativa es independiente del estado anterior, es decir,

$$\pi(x_t|x_{t-1}) = \pi(x_t). \tag{4.37}$$

La probabilidad de aceptación, en este caso, tiene la forma

$$\mathcal{A}(x_{t-1}, x_t) = \text{mín}\left[1, \frac{p(x_t)\pi(x_{t-1})}{p(x_{t-1})\pi(x_t)}\right] = \text{mín}\left[1, \frac{w(x_t)}{w(x_{t-1})}\right], \tag{4.38}$$

donde hemos definido los pesos

$$w(x) \triangleq \frac{p(x)}{\pi(x)}, \tag{4.39}$$

con $p(x) \propto p_o(x)$. Esta situación es particularmente interesante porque puede compararse con el *método aceptación-rechazo* visto en la sección (3.4.2), y con el *importance sampling* que se introdujeron en el Capítulo 3. Notese, de hecho, que los pesos en la Ecuación (4.39) son los mismos que en el importance sampling. La diferencia radica en que aquí se añade un paso de test (aceptación o rechazo del movimiento) dependiente del peso actual y del anterior, pero finalmente las muestras de la cadena producida por el MH tendrán todas un "mismo peso". Es decir, aparte del transitorio, todas las muestras se distribuyen como $p_o(x)$ y son igual de "buenas".

Comparando con el método de aceptación-rechazo hay que decir que éste puede ser aplicado sólo cuando conocemos (analíticamente) una constante M tal que se cumpla la condición $M\pi(x) \geq p(x)$ $(p(x) \propto p_o(x))$, mientras el MH puede utilizarse siempre. La Figura 4.6 muestra los diferentes escenarios donde pueden ser utilizados los dos métodos. Además, con la técnica de aceptación-rechazo en general muchas muestras serán descartadas, cosa que no ocurre con el MH (aparte de las muestras del transitorio, ó "burn in period"). El precio a pagar es que con el algoritmo MH las muestras producidas están correlacionadas (y a veces incluso repetidas), mientras el método de aceptación-rechazo genera muestras independientes. Se puede afirmar que evitamos el rechazo de las muestras, pagándolo con correlación entre ellas.

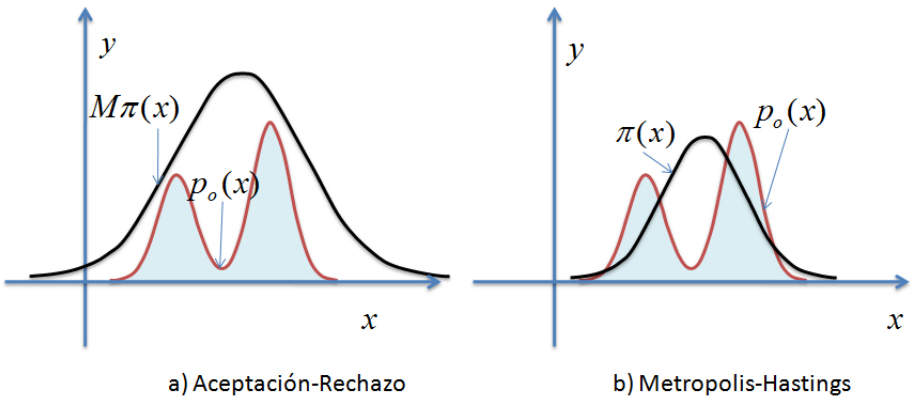

Figura 4.6. Metropolis-Hastings vs aceptación rechazo. **(a)** El método de aceptación-rechazo puede aplicarse solo cuando una constante M, tal que $M\pi(x) \geq p(x)$, es conocida. **(b)** El algoritmo MH no necesita que se cumplan desigualdades entre $\pi(x)$ y $p(x) \propto p_o(x)$.

4.6.2. Función tentativa simétrica

Otro posible caso especial del algoritmo MH asume una densidad tentativa simétrica de manera que

$$\pi(x_t|x_{t-1}) = \pi(x_{t-1}|x_t), \tag{4.40}$$

obteniendo una probabilidad de aceptación de la forma

$$\mathcal{A}(x_{t-1}, x_t) = \text{mín}\left[1, \frac{p(x_t)}{p(x_{t-1})}\right], \tag{4.41}$$

con $p(x) \propto p_o(x)$. Además es importante notar que

- si $p_o(x') \geq p_o(x_{t-1})$, entonces $\mathcal{A}(x_{t-1}, x_t) = 1$,
- mientras si $p_o(x') < p_o(x_{t-1})$, se tiene $\mathcal{A}(x_{t-1}, x_t) = \frac{p(x_t)}{p(x_{t-1})} < 1$.

Es decir, cuando el punto candidato x' presenta un valor de $p(x)$ mayor que el anterior aceptamos el movimiento siempre, mientras que si el movimiento se realiza hacia un punto con valor de la función objetivo $p(x) \propto p_o(x)$ menor, la probabilidad de aceptar es $\frac{p(x_t)}{p(x_{t-1})}$. Claramente, esta propiedad es muy interesante desde el punto de vista de la optimización (maximización en este caso). Observamos este funcionamiento en la Figura 4.7.

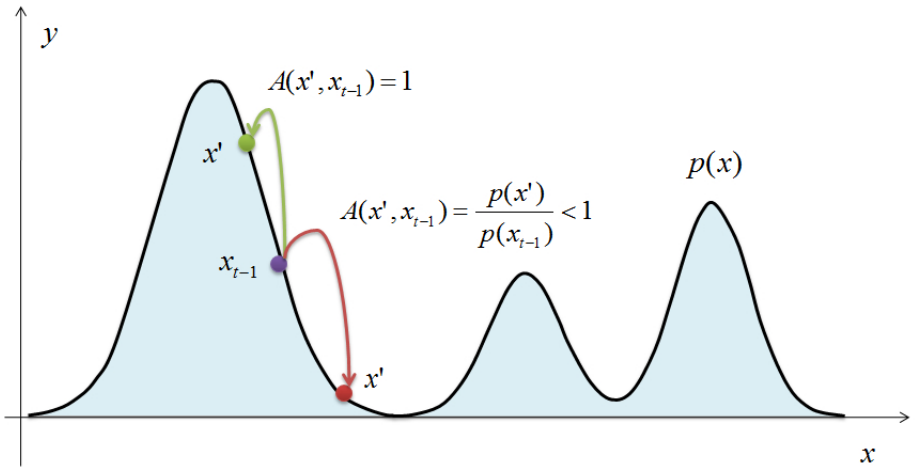

Figura 4.7. Funcionamiento con función tentativa simétrica. Los movimientos hacia "arriba" se aceptan siempre, mientras hacia "abajo" se aceptan con probabilidad $\frac{p(x')}{p(x_{t-1})}$.

4.6.3. Función tentativa como camino aleatorio

Se considera una variable aleatoria X_t del proceso estocástico (la cadena de Markov) asociado al algoritmo MH al tiempo t. En ocasiones la expresión

$$X_t \sim \pi(x_t|x_{t-1}), \tag{4.42}$$

se puede expresar de forma equivalente con la siguiente Ecuación entre variables aleatorias:

$$X_t = X_{t-1} + E, \tag{4.43}$$

donde E es una variable aleatoria con densidad genérica $q(\epsilon)$. Por ejemplo, E podría ser una variable aleatoria Gaussiana $E \sim q(\epsilon) \propto \exp\{-(\epsilon-\mu)^2/(2\sigma^2)\}$. La Ecuación (4.43) define un camino aleatorio ("random walk").

Cuando esto ocurre (es decir, la Ecuación (4.42) es equivalente a la (4.43)), la densidad tentativa se suele denotar también como

$$\pi(x_t|x_{t-1}) = \pi(x_t - x_{t-1}), \tag{4.44}$$

aunque esta notación no sea la más apropiada (se pierde la noción de probabilidad condicional).

En la Figura 4.8 se puede ver el funcionamiento del algoritmo MH utilizando una función tentativa como "random walk".

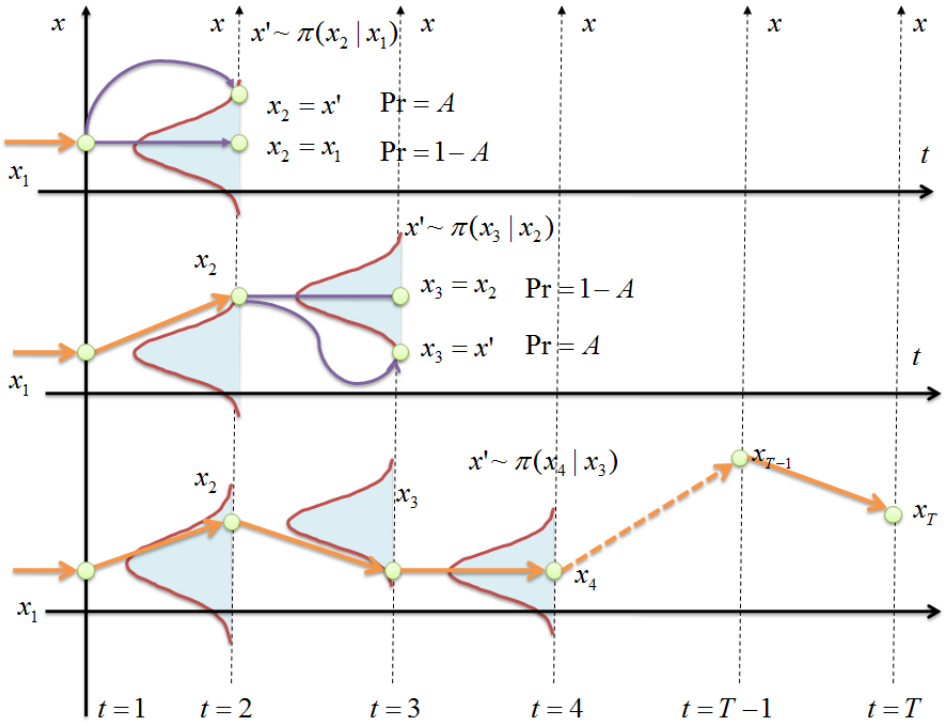

Figura 4.8. Funcionamiento del MH con una función tentativa como camino aleatorio. Se puede apreciar que la media de la densidad tentativa $\pi(x_t|x_{t-1})$ coincide con la muestra aceptada anteriormente.

Se aprecia claramente en la Figura 4.8 como la media de la densidad tentativa $\pi(x_t|x_{t-1})$ se desplaza en función de las muestras aceptadas anteriormente.

4.7. Conclusiones y consideraciones finales

El algoritmo MH es sin duda el más importante dentro de la categoría de los MCMC. La característica más positiva es sin duda que puede ser aplicado prácticamente en cualquier circunstancia, para muestrear cualquier tipo de densidad objetivo. Esto lo hace muy atractivo también desde el punto de vista de la optimización. Por otro lado, el principal defecto es que las muestras generadas no son independientes sino correlacionadas, e incluso a veces *repetidas*. Esto ralentiza la convergencia de la cadena a la densidad invariante.

Capítulo 5

Métodos estocásticos para la optimización

5.1. Introducción

Por comodidad, vamos a reescribir el problema de optimización planteado al principio de este proyecto, es decir la Ecuación (3.1), que tenía la forma:

$$\hat{\mathbf{x}} = \arg\min_{x}[f(\mathbf{x})] \text{ con } \mathbf{x} \in \mathcal{S}$$

Bajo ciertas condiciones de regularidad y considerando el espacio de búsqueda $\mathcal{S}$ no vacío y compacto, el teorema de Weierstrass del análisis clásico garantiza la existencia de una solución global óptima.

Los métodos de optimización estocástica son esencialmente computacionales, y por ello, se han introducido de forma progresiva en el mundo de la optimización a medida que ordenadores más y más potentes han sido diseñados. Las primeras contribuciones de estos métodos se remontan al año 1958 con Bock y Croes, que desarrollaron diferentes métodos para resolver el problema del viaje del hombre de negocios, más conocido como "*travelling salesman problem*": dada una serie de ciudades y de distancias entre ellas, el problema consiste en diseñar el camino más corto que pase por todas las ciudades, desde una de comienzo. Se puede representar con un conjunto de nodos (ciudades) y de aristas con pesos asignados (distancias entre ciudades) formando un grafo.

Es difícil clasificar los algoritmos de optimización estocástica. En este capítulo se dividirán estas técnicas en dos macro-categorias. Por un lado la correspondiente a los métodos de *aproximación estocástica* y por otra los algoritmos "*metaheurísticos*", que tal y como se definen en este trabajo, se corresponden con las técnicas de *exploración estocástica*.

Generalmente, cuando se utiliza el término metaheurístico en un algoritmo, suele referirse

a métodos que imitan algún comportamiento que se da en la naturaleza, como por ejemplo, el principio de evolución mediante selección natural (algoritmos genéticos), el proceso de recocido de metales como el hierro ("*Simulated Annealing*"), o la autoorganización de las colonias de hormigas ("*Ant Colony Optimization*").

Sin embargo, este proyecto tratará el termino metaheurístico, con una definición más general, basada en las propiedades de los algoritmos tal y como se hace en [Winker y Maringer, 2007]. Las características comunes a todas las técnicas que trataremos a continuación son las siguientes:

1. que contengan un cierto grado de aleatoriedad,
2. que sean capaces de proporcionar "buenas" aproximaciones del mínimo global óptimo $\hat{\mathbf{x}}$, y que dicha aproximación vaya mejorando a medida que los recursos computacionales atribuidos o el tiempo empleado aumente,
3. que sean robustos frente a cambios en las características del problema. Es decir, que tengan buenas prestaciones no para un problema específico, sino para diferentes clases de problemas,
4. que puedan ser implementados de forma sencilla,
5. y que puedan ser utilizados por aplicaciones en el campo continuo o discreto (casi) sin diferencias.

Se recuerda que se puede demostrar que la mayoría de estas técnicas presentan una convergencia en probabilidad al mínimo global óptimo. Otra ventaja importante de estos algoritmos es que no imponen restricciones sobre la función de coste $f(\mathbf{x})$. De hecho, para la mayoría de los algoritmos que se discutirán en este Capítulo, es suficiente con saber evaluar la función de coste $f(\mathbf{x})$ para un valor dado del espacio de búsqueda $\mathcal{S}$. No es necesario asumir propiedades globales de la función de coste, y no es necesario poder calcular derivadas (que en ocasiones es imposible).

Existen innumerables funciones de coste que no cumplen las condiciones necesarias de convergencia para poder aplicar los algoritmos deterministas de optimización tradicionales. Es por ello, que no se podrá asegurar en una cantidad considerable de casos, si la solución obtenida con dichos métodos es válida. O bien, los recursos necesarios para llegar a una solución valida son demasiado altos (tiempo, dinero, etc).

Un problema clásico en el que es posible aplicar métodos metaheurísticos es el de maximizar la función de verosimilitud. Cuando la función de verosimilitud es globalmente convexa, existen numerosos métodos deterministas que obtienen resultados satisfactorios. Sin embargo, cuando

esta función presenta dificultades (multitud de mínimos locales, discontinuidades, etc), puede ser útil utilizar una técnica estocástica en lugar de una determinista.

Se tratarán a continuación los métodos de aproximación estocástica y las técnicas de exploración estocástica. Éstas últimas se dividen en algoritmos de trayectoria (5.3) y basadas en población (5.7).

5.2. Clasificación de los métodos estocásticos

El objetivo de está sección es presentar una muestra bastante amplia, pero a la vez concisa, de los métodos estocásticos más importantes desarrollados en las dos últimas décadas. Se debe notar que no existe una forma única de clasificar estos algoritmos, sino que la clasificación de este proyecto se ha basado en otras ya confeccionadas intentando agrupar las técnicas con características parecidas en la misma clase. De este modo, será más fácil estudiar los puntos fuertes y débiles de cada método (o grupo de métodos) por separado.

Como ya se comentó anteriormente la optimización estocástica se divide en dos grandes clases. Estas dos grandes clases:

- métodos de aproximación estocástica,
- y métodos de exploración estocástica.

Este ultimo grupo suele también llamarse o, más precisamente, tiene un solape considerable con otras clases como los algoritmos metaheurísticos, los algoritmos de *búsqueda local*[1] y los algoritmos de *búsqueda aleatoria* ("random search"). Estas definiciones, que se encuentran con frecuencia en literatura, definen en gran parte la misma categoría de técnicas. Por esta razón, en este proyecto utilizaremos estos términos (exploración estocástica - métodos metaheurísticos - métodos de búsqueda local - métodos de búsqueda aleatoria) casi como sinónimos.

Respecto a las otras clasificaciones, por ejemplo las de [Zlochin et al., 2004] y [Schoen, 2002] expuestas en el Capitulo 3, se debe entender que no son en ningún caso clasificaciones contrapuestas entre sí, sino que todas se podrían fusionar y complementarse.

En la Figura 5.1 se puede ver un posible esquema de clasificación junto con las técnicas más características de cada clase y subclase.

[1] No se refiere a búsqueda de mínimos locales, como podría parecer a la vista del nombre, sino que adquiere el calificativo local debido la búsqueda en vecindarios locales a un punto considerado.

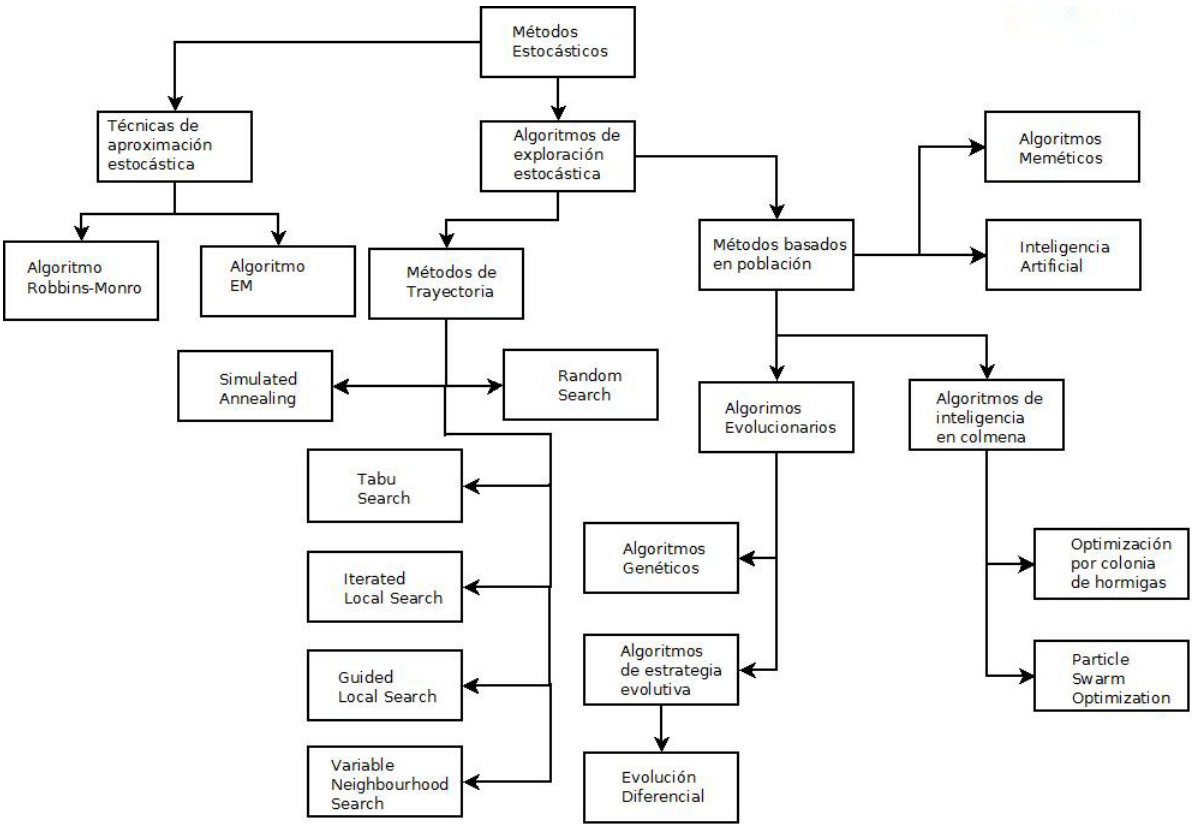

Figura 5.1. Clasificación de los distintos métodos de optimización estocásticas.

Metodos de aproximación estocástica

Están basados en la creación de aproximaciones locales de la función de coste en el punto considerado [Robert y Casella, 1999]. Los pasos seguidos por un algoritmo de aproximación son esencialmente dos:

- Se genera una aproximación $\hat{f}(\mathbf{x})$ de la función de coste $f(\mathbf{x})$ en $\mathbf{x}_{t-1}$.
- Se minimiza la aproximación $\hat{f}(\mathbf{x})$ encontrando un punto $\mathbf{x}^*$. Al siguiente paso se considera $\mathbf{x}_t = \mathbf{x}^*$.

Se podría afirmar y demostrar que también las técnicas aproximativas explotan las propiedades probabilísticas de la función de coste, que en ocasiones pueden ser difíciles de interpretar. Suelen relacionarse con problemas de estimación de modelos donde no se suelen conocer todos los datos.

La técnica de aproximación estocástica más conocida es el algoritmo EM (*Expectation Maximization Algorithm*), propuesta en [Dempster et al., 1977], y otras como el método de Robbins-Monro descrito en 1951 [Robbins y Monro, 1951]. Este algoritmo se utiliza habitualmente en

problemas de agrupamiento o aprendizaje máquina [Mackay, 2003], en reconstrucción de imágenes médicas tomográficas [Mostafa et al., 2001], así como en problemas de redes neuronales como los expuestos por [Neal, 1999], [Ripley, 1994] y [Ripley, 1996].

Algoritmos de exploración estocástica

Estos métodos, dada la salida anterior $\mathbf{x}_{t-1} \in \mathbb{R}^m$, seleccionan en el paso siguiente otro punto (uniformemente, por ejemplo) en un vecindario $\mathcal{H}(\mathbf{x}_{t-1}, \eta)$ definido como

$$\mathcal{H}(\mathbf{x_{t-1}}, \eta) \triangleq \{\mathbf{x} \in \mathbb{R}^{\mathbf{m}} : \|\mathbf{x} - \mathbf{x_{t-1}}\| \leq \eta\}, \tag{5.1}$$

donde $\|\cdot\|$ es una norma en dimensión m y η una constante arbitraria. En este caso la probabilidad de seleccionar un punto fuera de $\mathcal{H}(\mathbf{x}_{t-1}, \eta)$ es nula.

En general, se puede considerar una versión más "blanda" donde la probabilidad de transición es

$$\mathbf{x}_t \sim \pi(\mathbf{x}_t|\mathbf{x}_{t-1}) \propto H(\|\mathbf{x} - \mathbf{x_{t-1}}\|), \tag{5.2}$$

donde $H(\vartheta)$ es una función tal que

$$\int_{\mathbb{R}} H(\vartheta) d\vartheta \leq +\infty,$$

como por ejemplo $H(\vartheta) = \exp(-\vartheta^2)$ (en este caso $\pi(\mathbf{x}_t|\mathbf{x}_{t-1}) \propto \exp\left(-\|\mathbf{x} - \mathbf{x_{t-1}}\|^2\right)$). En este caso, puntos fuera del vecindario $\mathcal{H}(\mathbf{x}_{t-1}, \eta)$ pueden ser seleccionados aunque con baja probabilidad.

Además, las técnicas de exploración estocástica se dividen en:

- métodos de un solo punto $\mathbf{x}_t \in \mathbb{R}^m$ (denominados también métodos de *trayectoria*), y
- algoritmos que utilizan varios puntos, es decir donde se trabaja con un conjunto de M puntos, que denominaremos población

  $$\mathbf{X}_t = [\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, ..., \mathbf{x}_{M,t}].$$

  La matriz $\mathbf{X}_t$ tiene dimensión $m \times M$.

Entre los del primer tipo figuran Simulated Annealing o los algoritmos de búsqueda aleatoria ("Random Search") , mientras que entre los segundos se podrían nombrar los algoritmos genéticos o los métodos de optimización por colonia de hormigas.

5.3. Métodos de un solo punto (de trayectoria)

Los métodos de un solo punto son algoritmos de búsqueda local que trabajan sobre un sola trayectoria $\{\mathbf{x}_t\}_{t=0}^{T}$ siguiendo el esquema mostrado en la Figura 5.2.

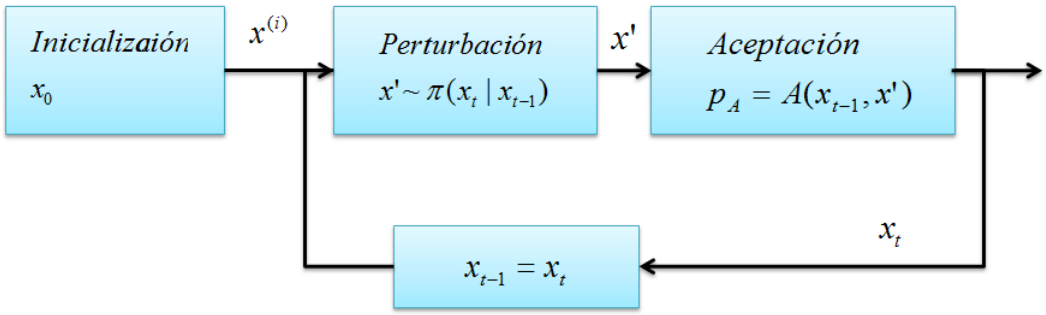

Figura 5.2. Diagrama de una técnica de trayectoria. Se inicializa un punto $\mathbf{x}_0$ y se propone un punto candidato $\mathbf{x}'$, que se aceptará con una probabilidad $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}')$. Si se acepta el punto se actualiza $\mathbf{x}_t = \mathbf{x}'$, y en caso contrario $\mathbf{x}_t = \mathbf{x}_{t-1}$.

El punto de inicialización $\mathbf{x}_0$, suele ser generado de forma uniforme en $\mathcal{S}$. Para ello, $\mathcal{S}$ debe ser finito (o se tiene conocimiento de la zona donde se sitúa el mínimo global). El punto candidato $\mathbf{x}'$ es generado usando (en la mayoría de los casos) una función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$, que depende del punto en la iteración anterior $\mathbf{x}_{t-1}$ del algoritmo. Un método muy común para generar un punto candidato $\mathbf{x}_t$, es definir un paso de una desviación tipica $\mathbf{r}_{t-1}$ en una determinada muestra aleatoria $\mathbf{e}_{t-1}$. Un algoritmo que utilice esta técnica generaría el punto candidato $\mathbf{x}'$ en la forma

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{r}_{t-1}\mathbf{e}_{t-1}. \tag{5.3}$$

En el Cuadro 5.1 se muestra el clásico algoritmo de búsqueda local para minimizar una función de coste $f(\mathbf{x})$. Por simplicidad se explica la técnica para una sola partícula $\mathbf{x}_{t-1} \in \mathbb{R}^m$, pero se opera de la misma forma si en lugar de un solo punto se tuviera una población con varios puntos $\mathbf{X}_t$.

La salida de nuestro algoritmo de optimización será un $\boldsymbol{x}_t$ que cumple el criterio de parada. Con probabilidad 1 se puede demostrar que $\mathbf{x}_t$ converge al mínimo global $\hat{\mathbf{x}}$ de $f(\mathbf{x})$, $\mathbf{x}_t \to \hat{\mathbf{x}}$ cuando $t \to \infty$.

Se aprecia claramente el parecido entre el bucle interno del algoritmo y el método Metropolis-Hastings visto en el Capítulo 4, en cuanto a la generación de un punto candidato $\mathbf{x}'$ a través de una función tentativa $\pi(\cdot)$ y su posterior aceptación con una probabilidad $\mathcal{A}(\cdot)$. Sin embargo, los objetivos de los métodos de exploración estocásticas y la técnica Metropolis-Hastings son

Método de exploración estocástica genérico
1. Inicialización: Generar la solución inicial $\mathbf{x}_0$, $t = 0$.
2. Perturbación: Generar $\mathbf{x}' \sim \pi(\mathbf{x}_t|\mathbf{x}_{t-1})$.
3. Aceptación:
3.1 Generar $u' \sim \mathcal{U}([0,1])$.
Siendo $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}')$ es una función de aceptación del punto candidato.
3.2 si $u' < \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') \rightarrow \mathbf{x}_t = \mathbf{x}'$, ó
3.3 si $u' \geq \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') \rightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$.
6. Finalización: Actualizar $t = t + 1$. Finalizar si se cumple el criterio de parada, sino repetir desde el paso 2.

Cuadro 5.1

distintos. Mientras que en el algoritmo Metropolis-Hastings el objetivo era muestrear puntos de una densidad objetivo $p_o(\mathbf{x})$, mediante la generación de una cadena de Markov (cuya densidad estacionaria $p_e(\mathbf{x})$ era igual a la densidad objetivo $p_o(\mathbf{x})$); en esta sección los puntos de la cadena de Markov generados se utilizan para calcular el mínimo global óptimo $\hat{\mathbf{x}}$ de la función de coste objetivo $f(\mathbf{x})$.

Claramente, los algoritmos de un solo punto difieren en la elección de las funciones $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ y $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}')$.

5.3.1. Pure Random Search (PRS)

El algoritmo "*Pure Random Search*", también denominado "*Blind Random Search*", fue descrito por primera vez en [Brooks, 1958], y es la técnica de búsqueda aleatoria más sencilla. Este método genera una secuencia de puntos independientes $\mathbf{x}_{t-1}$ uniformemente distribuidos en la región de búsqueda $\mathcal{S}$ hasta satisfacer el criterio de parada (que suele ser un número fijo de iteraciones N). Se recuerda que esta generación uniforme es sólo posible si $\mathcal{S}$ es acotado. El funcionamiento de la técnica se resume en el Cuadro 5.2.

La función tentativa utilizada en la definición del método es una distribución uniforme en $\mathcal{S}$. Es decir, el método así como descrito en el Cuadro 5.2 puede ser utilizado solo cuando $\mathcal{S}$ es acotado. Es por ello, que una versión más general del algoritmo debería utilizar una función tentativa genérica $\pi(\mathbf{x_t}|\mathbf{x_{t-1}})$ con colas infinitas.

En 1981, Rubinstein probó que para una función de coste objetivo $f(\mathbf{x})$ continua, el algorit-

Pure Random Search
1. Inicialización: Elegir $\mathbf{x}_0 \sim \mathcal{U}(\mathcal{S})$ y fijar el número de iteraciones $t = 0$.
2. Perturbación: Generar $\mathbf{x}' \sim \mathcal{U}(\mathcal{S})$.
3. Aceptación:
4. Si $f(\mathbf{x}') < f(\mathbf{x}_{t-1}) \rightarrow \mathbf{x}_t = \mathbf{x}'$.
5. Mientras si $f(\mathbf{x}') < f(\mathbf{x}_{t-1}) \rightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$.
6. Finalización: Parar si se cumple la condición de parada. En caso contrario $t = t + 1$ y volver a 2.

Cuadro 5.2

mo converge al mínimo global óptimo $\hat{\mathbf{x}}$ con probabilidad uno. El problema radica en que para converger a esta solución, puede que el algoritmo necesite una cantidad de tiempo (iteraciones) no asumible. En 1992, Zabinsky y Smith probaron que el algoritmo era ineficiente, ya que simplemente restringiendo la técnica a valores monotónicos, se conseguía una mejora exponencial en el número de iteraciones requeridas [Zabinsky y Smith, 1992]. De este modo, el algoritmo no trabaja bien cuando la dimensión m del problema tratado es muy alta. El proceso de aceptación de nuevos puntos candidatos se muestra en la Figura 5.3 (para el caso unidimensional).

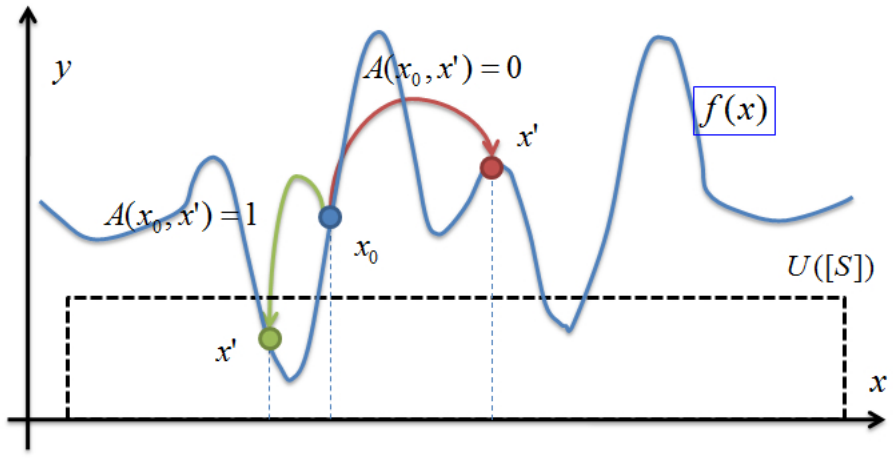

Figura 5.3. Funcionamiento Pure Random Search. Se elige un punto candidato $\mathbf{x}'$ aleatoriamente en todo el espacio de definición de la función de coste. En caso de que $\mathbf{x}'$ cumpla que $f(\mathbf{x}') < f(\mathbf{x}_{t-1})$, se acepta el punto con probabilidad uno, rechazandose en caso contrario.

Se observa que sólo los movimientos hacia puntos con menor valor de la función de coste son

aceptados. La función de aceptación $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}')$ según el Cuadro genérico 5.1 es en este caso

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') = \begin{cases} 1 & \text{si} \quad f(\mathbf{x}') < f(\mathbf{x}_{t-1}), \\ 0 & \text{si} \quad f(\mathbf{x}') \geq f(\mathbf{x}_{t-1}). \end{cases} \tag{5.4}$$

Además repetimos que una versión más general del algoritmo consiste en utilizar una función tentativa genérica $\pi(\mathbf{x_t}|\mathbf{x_{t-1}})$ con colas infinitas, en lugar de una uniforme $\mathcal{U}(\mathcal{S})$.

5.3.2. Pure Adaptive Search (PAS)

Pure adaptive search [Patel et al., 1988, Zabinsky y Smith, 1992] es una variante del pure random search (PRS) que realmente es de difícil aplicación dado que requiere un conocimiento analítico de la función de coste más profundo (no sólo la evaluación como para el PRS).

De hecho, el PAS genera una secuencia de puntos uniformemente distribuidos en el subespacio de búsqueda mejorado $\mathcal{S}_i$, definido como $\mathcal{S}_i = \{\mathbf{x} \in \mathcal{S} : f(\mathbf{x}) < \mathbf{y}_{t-1}\}$. Notar que para generar esta secuencia el subepacio $\mathcal{S}_i$, al igual que en el caso anterior, dicho espacio debe estar acotado. Si esto no es así, habrá que utilizar una densidad tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ no uniforme sobre este subespacio. El algoritmo funciona del modo mostrado en el Cuadro 5.3.

Pure Adaptive Search
1. Inicialización: Fijar el número de iteraciones $t = 0$, inicializar $\mathbf{x}_0 \sim \mathcal{U}([\mathcal{S}])$ y fijar $\mathbf{y}_0 = f(\mathbf{x}_0)$.
2. Perturbación: Generar $\mathbf{x}'$, de acuerdo con una distribución uniforme en el subconjunto mejorado dado por $\{\mathcal{S}_i = \mathbf{x} \in \mathcal{S} : f(\mathbf{x}) < \mathbf{y}_{t-1}\}$.
3. Aceptación: Fijar $\mathbf{x}_t = \mathbf{x}'$ e $\mathbf{y}_t = f(\mathbf{x}_t)$.
4. Finalización: Parar si se cumple la condición de parada. En caso contrario $t = t + 1$ y volver a 2.

Cuadro 5.3

Hay que hacer tres consideraciones importantes:

1. respecto al PRS, el PAS utiliza una densidad para proponer candidatos (tentativa) más eficiente, dado que propone puntos solo en subespacio $\mathcal{S}_i$ de interés.

2. Por esto realmente tenemos que la función de aceptación es

$$\mathcal{A}(x', x_{t-1}) = 1,$$

siempre.

3. Por otro lado, el gran problema de esta técnica radica en que en general el subespacio mejorado $\mathcal{S}_i$, es muy difícil de hallar analíticamente. Esto hace prácticamente imposible la aplicación de este método a problemas reales.

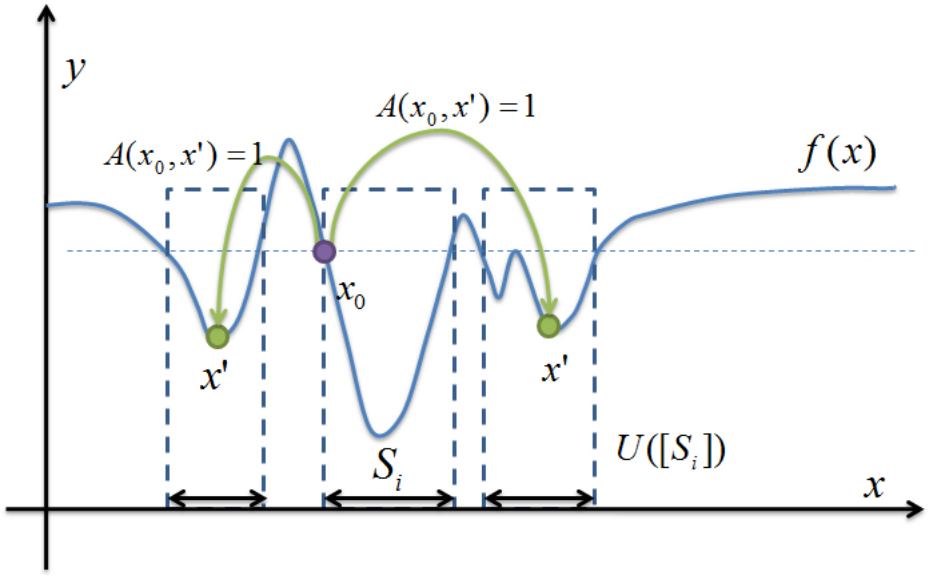

Figura 5.4. Funcionamiento Pure Adaptive Search. Se propone un punto candidato $\mathbf{x}'$, muestreando en el subespacio mejorado $\mathcal{S}_i$, que se acepta con probabilidad uno.

5.3.3. Accelerated Random Search (ARS)

Este algoritmo apareció por primera vez en [Appel y Radulovic, 2000]. Al ser la notación un poco diferente a la que se viene utilizando hasta ahora se empezará definiendo las variables más importantes que aparecen. Siendo el espacio de definición de la función de coste $\mathcal{S}$ y siendo éste acotado; se denotará la norma en $\mathcal{S}$ como $\|\cdot\|$. Se utilizará como función tentativa una densidad uniforme sobre una hiperesfera (vencidario) de radio r centrada en $\mathbf{x}$ y se denotará como $\mathcal{H}(\mathbf{x}, r)$. Por último, se definen un factor de contracción $c > 1$ (ó constante de reducción) y un umbral de precisión $\rho > 0$. El método opera tal y como se muestra en el Cuadro 5.4.

En la versión básica del algoritmo, se genera la solución inicial $\mathbf{x}_0$ de forma uniforme en $\mathcal{S}$, y se inicializa el valor del radio de la hiperesfera $r_0 = a$; siendo a un número entero positivo que se podrá modificar para mejorar el comportamiento del algoritmo. Se utiliza como función tentativa una densidad uniforme en una hiperesfera centrada en $\mathbf{x}_{t-1}$ y de radio $r_{t-1} \in (0, a]$ a

Accelerated Random Search
1. Inicialización: Inicializar $\mathbf{x}_0 \sim \mathcal{U}([\mathcal{S}])$. Fijo el número de iteraciones $t = 0$ y $r_0 = a$.
2. Perturbación: Dado $\mathbf{x}_{t-1} \in \mathcal{S}$ y r_{t-1}, se genera $\mathbf{x}'$ uniformemente en la hiperesfera $\mathcal{H}(\mathbf{x}_{t-1}, r_{t-1})$.
3. Aceptación: Si :
4. $f(\mathbf{x}') > f(\mathbf{x}_{t-1}) \rightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$ y $r_t = a$.
5. $f(\mathbf{x}') \leq f(\mathbf{x}_{t-1}) \rightarrow \mathbf{x}_t = \mathbf{x}'$ y $r_t = r_{t-1}/c$.
Cuando $r_t < \rho$ entonces vamos $r_t = a$.
6. Finalización: Parar si se cumple la condición de parada. En caso contrario $t = t + 1$ y volver a 2.

Cuadro 5.4

partir de la cual se generarán nuevos puntos candidatos $\mathbf{x}'$.

En la versión extendida la generación de puntos candidatos se realiza a partir de una función tentativa genérica $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$. Por ejemplo, se podría utilizar una densidad gaussiana centrada en x_{t-1} con una varianza σ_{t-1} (que tiene relación directa con el radio r_{t-1}).

La parte más importante de Accelerated Random Search está en el paso de aceptación dado que también se modifica un parametro (la varianza) de la densidad tentativa. La función de aceptación es exactamente igual al PRS, es decir

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') = \begin{cases} 1 & \text{si} \quad f(\mathbf{x}') < f(\mathbf{x}_{t-1}), \\ 0 & \text{si} \quad f(\mathbf{x}') \geq f(\mathbf{x}_{t-1}). \end{cases} \tag{5.5}$$

Pero en este caso de que la función de coste en el punto candidato sea menor o igual que la función de coste en la solución anterior ($f(\mathbf{x}') \leq f(\mathbf{x}_{t-1})$), también se disminuye la varianza de la función tentativa en un factor c ($\sigma_t \rightarrow \frac{\sigma_t}{c}$). Es decir, al encontrar un punto mejor que el anterior, se disminuye la varianza de la función tentativa. Esto es, porque asumiendo que se está cerca del mínimo global, se intenta mejorar la precisión de la solución proporcionada. La constante c es un número entero positivo, modificable por el implementador para mejorar las prestaciones del algoritmo. Un ejemplo de lo expuesto puede verse en la Figura 5.5.

Dado $\mathbf{x}_0$, se muestrea la función tentativa obteniendo $\mathbf{x}_1$. Al ser $f(\mathbf{x}_1) < f(\mathbf{x}_0)$, se acepta el punto candidato y se disminuye la varianza de la función tentativa en un factor c ($\sigma_0 \rightarrow \sigma_1$).

Por otro lado, si la función de coste en el punto candidato es mayor que la función de coste en

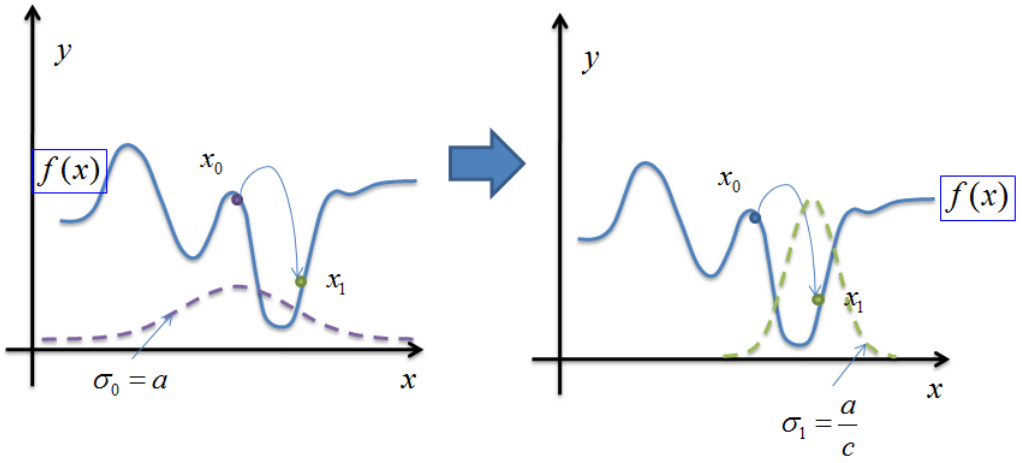

Figura 5.5. Funcionamiento Accelerated Random Search (Caso 1). Se realiza un movimiento descendente y se disminuye el valor de la varianza de la función tentativa en un factor c. Se debe notar que $\sigma_0/c > \rho$.

la solución anterior ($f(\mathbf{x}') > f(\mathbf{x}_{t-1})$), se hace todo lo contrario. Es decir, se aumenta la varianza de la función tentativa a un valor $\sigma_t = a$ (máximo valor de la varianza permitido). Se puede observar el funcionamiento de esta técnica en la Figura 5.6.

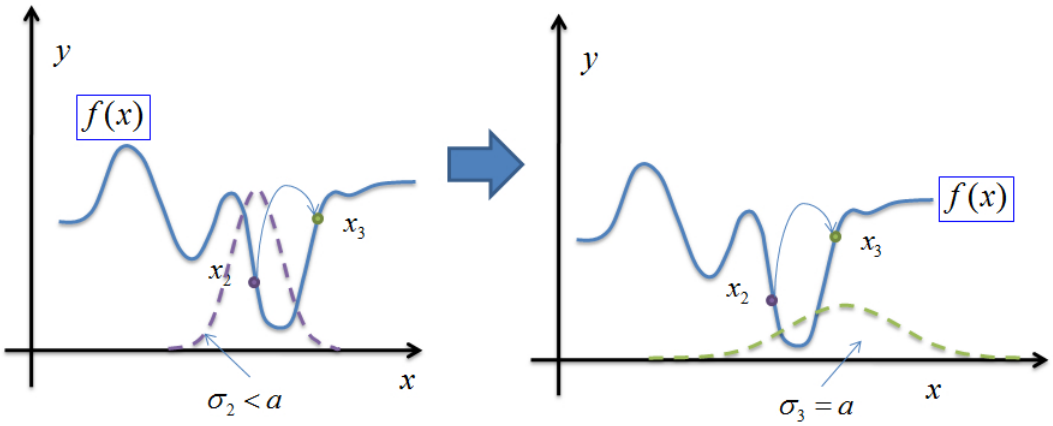

Figura 5.6. Funcionamiento Accelerated Random Search (Caso 2). Se realiza un movimiento ascendente y se aumenta el valor de la varianza de la función tentativa ($\sigma_2 = a$).

Se muestrea un nuevo punto $\mathbf{x}_2$, que cumple que $f(\mathbf{x}_2) > f(\mathbf{x}_1)$, de manera que se rechaza el punto y se aumenta la varianza de la función tentativa ($\sigma_2 = a$).

Esta técnica de aumento y disminución del radio intenta, que cuando el algoritmo se acerque al mínimo global óptimo, no escape de su cuenca de atracción. Esto lo hace disminuyendo la varianza de la función tentativa, de manera que se generen muchos puntos candidatos en el

entorno de la solución óptima. Por el contrario, cuando el punto muestreado es peor que el anterior, lo más probable es que todavía se esté lejos de la solución óptima y el algoritmo se deba concentrar en otras regiones del espacio. Es por ello, que se aumenta la varianza de la función tentativa, posibilitando el movimiento a puntos lejanos de $\mathbf{x}_t$.

Existe un comportamiento adicional a los ya expuestos para combatir que el método no quede estancado en mínimos locales $\mathbf{x}_{local}$. Se basa en hacr uso de un entero $\rho > 0$, que recibe el nombre de umbral de precisión. El funcionamiento consiste en aumentar la varianza de la función tentativa cuando dicha varianza tenga un valor por debajo de este umbral ($\sigma_t < \rho$). Es decir, se aumenta la varianza de la función tentativa cuando se han aceptado *sucesivamente* una serie de puntos candidatos. Este "reinicio" permite al algoritmo escapar de los mínimos locales, ya que si el método se encuentra en la cuenca de atracción de la solución global óptima, la convergencia seguirá estando asegurada. Cuando la varianza es muy pequeña seguramente se habrá logrado una muy buena precisión, pero también es posible que no sea la solución global y debamos aumentar la varianza para encontrar dicho mínimo global. El funcionamiento del umbral de precisión se muestra en la Figura 5.7.

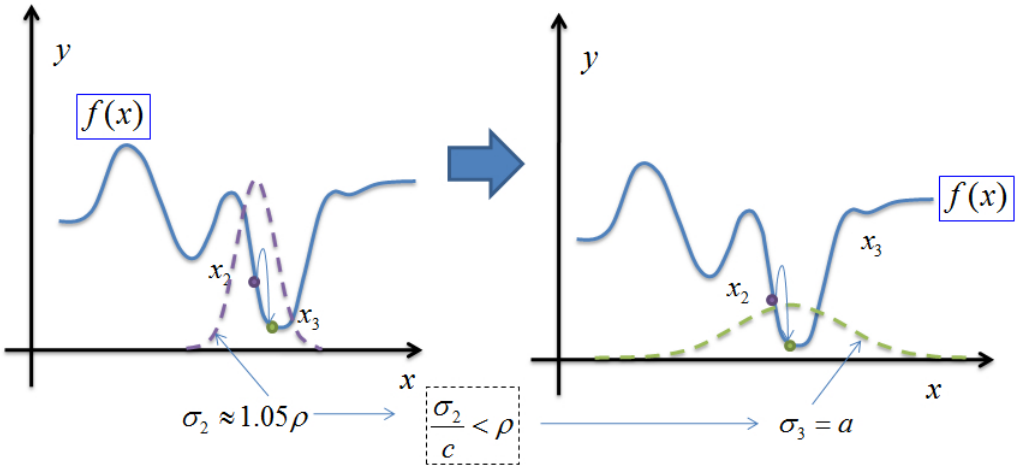

Figura 5.7. Funcionamiento del umbral de precisión en ARS. Tras la aceptación de sucesivos puntos se tiene que $\frac{\sigma_2}{c} < \rho$, por lo que se aumenta la varianza haciendo $\sigma_3 = a$.

Se observa como al aceptar consecutivamente dos puntos ($\mathbf{x}_1$ y $\mathbf{x}_2$), el radio de la función tentativa se sitúa por debajo del umbral de precisión, provocando que se aumente la varianza de la función tentativa (a un valor a) en la siguiente iteración.

5.3.4. Simulated Annealing (SA)

El algoritmo "Simulated Annealing" recibe su nombre de la analogía con el fenómeno físico de recocido ("*annealing*") del metal. Mientras que a muy altas temperaturas las moléculas del metal en estado líquido se mueven libremente, si la temperatura va disminuyendo progresivamente, la movilidad de estas moléculas se va perdiendo; formándose un cristal puro que se corresponde con el estado de mínima energía posible del metal. Si la temperatura baja demasiado rápido el cristal generado no es un cristal puro sino policristalino, el cual posee un estado amorfo y un nivel de energía mayor que el del cristal puro. La función de energía $E(\mathbf{x})$ correspondiente al fenómeno físico representa el papel de la función de coste $f(\mathbf{x})$ en los problemas de optimización, y el estado de cristal puro se corresponde con el mínimo global óptimo $\hat{\mathbf{x}}$. La temperatura en los problemas de optimización, aunque sigue referenciándose con el mismo nombre, ya no es una temperatura real, sino simplemente un parámetro del algoritmo (temperatura ficticia) y se denotará como T_t. Esta temperatura T_t tiende a cero a medida que aumentan las iteraciones $t \to +\infty$.

El algoritmo Simulated Annealing funciona tal y como se muestra en el Cuadro 5.5.

Simulated Annealing
1. Inicialización: Inicializar $\mathbf{x}_0 \in \mathcal{S}$, $t = 0$, y inicializar también la temperatura inicial T_0.
2. Perturbación: Generar un punto candidato $\mathbf{x}' \sim \pi(\mathbf{x}_t \vert \mathbf{x}_{t-1})$.
3. Aceptación: Generar $u' \sim \mathcal{U}([0,1])$.
4. Si $u' > \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t) \to \mathbf{x}_t = \mathbf{x}_{t-1}$.
5. Si $u' \leq \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t) \to \mathbf{x}_t = \mathbf{x}'$.
6. Actualizar $t = t + 1$. Si $t \neq N$ volver a 2.
8. Actualizar la temperatura $T_{t+1} = \phi(T_t)$ (ϕ función decreciente $\to 0$ cuando $t \to +\infty$).
9. Finalización: Parar si se cumple la condición de parada. En caso contrario volver a 2.

Cuadro 5.5

La forma *más típica* de la probabilidad de aceptación es

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_k) = \begin{cases} 1 & \text{si} \quad f(\mathbf{x}') < f(\mathbf{x}_{t-1}), \\ h(\mathbf{x}_{t-1}, \mathbf{x}', T_t) & \text{si} \quad f(\mathbf{x}') \geq f(\mathbf{x}_{t-1}), \end{cases} \tag{5.6}$$

donde claramente $0 \leq h(\mathbf{x}_{t-1}, \mathbf{x}', T_t) \leq 1$.

La probabilidad de transición $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t)$ depende del parametro temperatura T_t. Además, podemos afirmar que en el SA la probabilidad $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t)$ ($h(\mathbf{x}_{t-1}, \mathbf{x}', T_t)$) es directamente

proporcional al valor de la temperatura T_t (mayor temperatura mayor probabilidad de aceptar el movimiento).

En este sección veremos como elegir la función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$, la función de aceptación $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t)$, la función de enfriado ("*cooling schedule*") o de actualización de la temperatura $\phi(T_t)$ (donde ϕ es una transformación adecuadamente escogida que $\to 0$ cuando $t \to +\infty$) y el criterio de parada. Todos estos parámetros deben ser elegidos apropiadamente para conseguir un algoritmo Simulated Annealing eficiente. En las siguientes secciones se estudiará como afectan estos parámetros al funcionamiento del algoritmo.

La función de aceptación

Existen varias funciones de aceptación utilizadas en la literatura para algoritmos Simulated Annealing de optimización en espacio continuo. Sin duda, la más referenciada es

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t) = \text{mín}\left[1, \exp\left(-\frac{f(\mathbf{x}') - f(\mathbf{x}_{t-1})}{T_t}\right)\right] \tag{5.7}$$

Esta función de aceptación, que algunos textos denominan de Boltzmann, puede expresarse como

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_k) = \begin{cases} 1 & \text{si} \quad f(\mathbf{x}') < f(\mathbf{x}_{t-1}), \\ \exp\left(-\dfrac{f(\mathbf{x}') - f(\mathbf{x}_{t-1})}{T_t}\right) & \text{si} \quad f(\mathbf{x}') \geq f(\mathbf{x}_{t-1}), \end{cases} \tag{5.8}$$

es decir, tenemos

$$h(\mathbf{x}_{t-1}, \mathbf{x}', T_t) \triangleq \exp\left(-\frac{f(\mathbf{x}') - f(\mathbf{x}_{t-1})}{T_t}\right) \quad \text{con} \quad f(\mathbf{x}') \geq f(\mathbf{x}_{t-1}).$$

Entonces, siempre se aceptan movimientos descendentes hacia un punto con función de coste objetivo menor ($f(\mathbf{x}') < f(\mathbf{x}_{t-1})$). Sin embargo, para que el algoritmo no quede atrapado en mínimos locales, también se aceptan movimientos ascendentes hacia puntos con función de coste objetivo $f(\mathbf{x})$ mayor ($f(\mathbf{x}') > f(\mathbf{x}_{t-1})$) con probabilidad $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t)$ como puede verse en la Figura 5.8.

De la Ecuación (5.7) se deriva que a medida que la temperatura T_t converge a cero, la probabilidad de aceptar movimientos ascendentes también se hace más pequeña, haciendo más difícil que el algoritmo escape de los mínimos locales.

Relación entre Simulated Annealing y el algoritmo Metropolis-Hastings

Se vio en el Capítulo 3, que a la función de coste $f(x)$ se le podía asociar una densidad

$$q(\mathbf{x}) \propto H(f(\mathbf{x})).$$

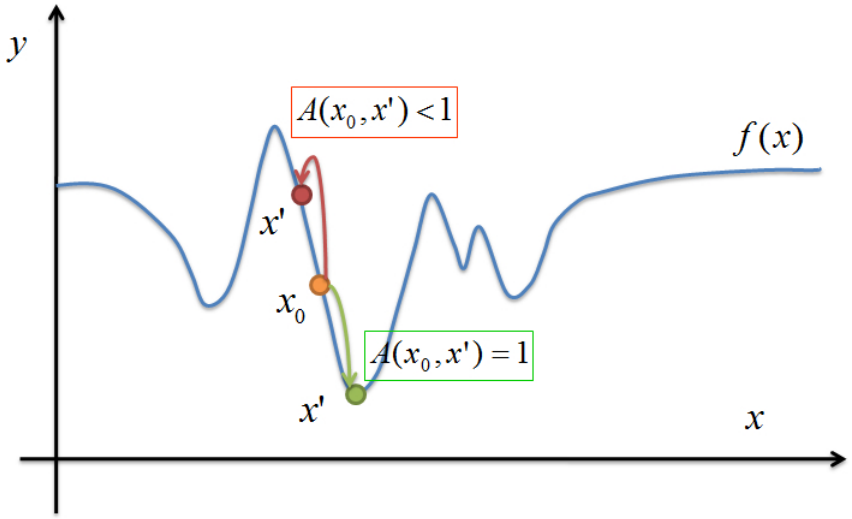

Figura 5.8. Aceptación en Simulated Annealing. Los movimientos ascendentes son aceptados con probabilidad $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t) < 1$, mientras que en los descendentes se aceptan con probabilidad 1.

Si ahora se elige $H(\theta) = \exp(-\theta)$ se puede escribir

$$q(\mathbf{x}) = \exp(-f(\mathbf{x})). \tag{5.9}$$

Tal y como se comentó en el Capítulo 3 los valores $\hat{\mathbf{x}}$ que minimizan la función objetivo $f(\mathbf{x})$ son los mismos que maximizan la función $q(\mathbf{x})$, es decir, sus modas. Por otro lado, muestrear desde una densidad $q(\mathbf{x})$ significa generar números aleatorios con una frecuencia de aparición proporcional al área por debajo de $q(\mathbf{x})$. Es decir, si somos capaz de muestrear $q(\mathbf{x})$, se obtendrá muestras $\mathbf{x}'$ cercanas a las modas de $q(\mathbf{x})$ con probabilidad alta, o lo que es lo mismo, cercanas al mínimo global de la función objetivo $f(\mathbf{x})$.

La idea que maneja Simulated Anneling visto como variación de Metropolis-Hastings hace referencia a la simulación de una cadena de Markov no homogenea (generación de muestras por Metropolis-Hastings), cuya distribución invariante $p_e(\mathbf{x})$ en la iteración k-ésima coincida no con $q(\mathbf{x})$, sino que

$$p_e(\mathbf{x}) \propto q^{(1/T_t)}(\mathbf{x}), \tag{5.10}$$

donde T_t es la temperatura en la iteración t-ésima y cumple que $\lim_{t\to\infty}(T_t) = 0$. Utilizando la notación del algoritmo Metropolis-Hastings, se podría reescribir el algoritmo Simulated Annealing de la siguiente forma:

1. Inicializar $\mathbf{x}_0$ uniformemente en $\mathcal{S}$ y fijar la temperatura inicial T_0. Inicializar $t = 0$.

2. Muestrear $\mathbf{x}' \sim \pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ y $u' \sim \mathcal{U}([0,1])$.

3. Se reescribe la función de aceptación vista en la Ecuación (5.7) con la notación del algoritmo Metropolis-Hastings como

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') = \text{mín}\left[1, \frac{q^{\frac{1}{T_t}}(\mathbf{x}')\pi(\mathbf{x}_{t-1}|\mathbf{x}')}{q^{\frac{1}{T_t}}(\mathbf{x}_{t-1})\pi(\mathbf{x}'|\mathbf{x}_{t-1})}\right]. \tag{5.11}$$

4. Si $u' \leq \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') \longrightarrow \mathbf{x}_t = \mathbf{x}'$.

5. En caso contrario, si $u' > \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}')$ tenemos $\longrightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$.

6. Actualizar la temperatura $T_{k+1} = \phi(T_t)$ de acuerdo a la ley de enfriamiento o "*cooling schedule*".

La técnica de Simulated Annealing puede verse como un "*random walk*" generado a partir de una cadena de Markov no homogenea controlada por la temperatura.

Si se considera una función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ simétrica [Dekkers y Aarts, 1996], es decir, que cumpla que

$$\pi(\mathbf{x}_t|\mathbf{x}_{t-1}) = \pi(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad \forall \mathbf{x}_{t-1}, \mathbf{x}_t \in \mathcal{S}, \tag{5.12}$$

entonces la Ecuación (5.11) queda

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') = \text{mín}\left[1, \frac{q^{\frac{1}{T_t}}(\mathbf{x}')}{q^{\frac{1}{T_t}}(\mathbf{x}_{t-1})}\right]. \tag{5.13}$$

Como consecuencia podemos afirmar que, si la densidad tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ *es simétrica*, la función de aceptación Metropolis vista en la Ecuación (5.7) trabaja sobre una densidad objetivo $(p_o(\mathbf{x}) = p_e(\mathbf{x}) = q^{\frac{1}{T_t}}(\mathbf{x}))$ del tipo

$$p_o(\mathbf{x}) \propto \exp\left(-\frac{f(\mathbf{x}) - f(\mathbf{x}_{t-1})}{T_t}\right), \tag{5.14}$$

que es denominada *distribución de Boltzmann* asociada a la función de coste $f(\mathbf{x})$. De hecho, sustituyendo $q(\mathbf{x}) \propto \exp\{-(f(\mathbf{x}) - f(\mathbf{x}_{t-1}))\}$ en la Ecuación (5.13) se logra la Ecuación (5.7). Es importante notar que la distribución de Boltzmann tiende a concentrarse en la región del mínimo global óptimo $\hat{\mathbf{x}}$ a medida que $T_t \to 0$.

Otras funciones de aceptación

Considerando siempre una función tentativa simétrica $\pi(\mathbf{x}_t|\mathbf{x}_{t-1}) = \pi(\mathbf{x}_{t-1}|\mathbf{x}_t)$, otra posible función de aceptación (utilizada en la literatura) es la dada por Barker que se define como

$$\begin{aligned}\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t) &= \frac{\exp\left(\frac{f(\mathbf{x}_{t-1})}{T_t}\right)}{\exp\left(\frac{f(\mathbf{x}_{t-1})}{T_t}\right) + \exp\left(\frac{f(\mathbf{x}')}{T_t}\right)}, \\ \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t) &= \frac{1}{1 + \exp\left(\frac{f(\mathbf{x}') - f(\mathbf{x}_{t-1})}{T_t}\right)}.\end{aligned} \tag{5.15}$$

Usando la función de aceptación de Barker *también los movimientos descendentes pueden ser rechazados*. En particular, aquellos que no mejoran significativamente el valor de la función de coste $f(\mathbf{x})$. Sin embargo, a medida que la temperatura T_t se decrementa, estos movimientos descendentes tienen una mayor probabilidad de aceptación. Por otro lado, los movimientos ascendentes son aceptados con una probabilidad menor (mayor rechazo), que cuando se usa la función de aceptación vista en la Ecuación (5.7).

Esta función $\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t)$ en (5.15) tiene cierta relación con el *estadístico de Bose-Einstein* utilizados en procesos físicos de partículas. Otra función de aceptación diferente a la proporcionan Tsallis y Cole [Cole, 2009, Klos y Kobe, 2001] (basada en el estadístico de Tsallis) que se define como

$$\mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}', T_t) = \text{mín}\left[1, \left(\frac{1 + (1 - w(T_t))f(\mathbf{x}_{t-1})}{1 + (1 - w(T_t))f(\mathbf{x}')}\right)^{\frac{w(T_t)}{1-w(T_t)}}\right], \tag{5.16}$$

y $\lim_{t\to+\infty} w(T_t) = 1$ (se recuerda que $\lim_{t\to+\infty} T_t = 0$).

Expuestas varias de las funciones de aceptación más referenciadas en la bibliografía, en la siguiente sección se estudiará en profundidad el comportamiento de los algoritmos Simulated Annealing con diferentes leyes de enfriado $\phi(T_t)$ y con diferentes funciones tentativas $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$.

Elección de la función de enfriado

La elección de la función o ley de enfriado $\phi(T_t)$ (junto con la elección de la función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$) son los puntos más críticos en la definición de un algoritmo de Simulated Annealing. En esta sección, se verán las elecciones que se han considerado más relevantes o innovadoras y se explicará el porqué de estas elecciones.

La más sencilla "cooling schedule" es

$$T_{t+1} = \alpha T_t, \tag{5.17}$$

donde α es una constante positiva entre menor que 1. Otra ley de enfriado muy típica es

$$T_t = T_0 \exp(-(1-c)t), \tag{5.18}$$

con $c \in [0,1)$ en la cual la temperatura decae de forma exponencial, o la que utiliza logaritmos como

$$T_t = \frac{T_0}{\log t}. \tag{5.19}$$

En [Bohachevsky y Johnson, 1986] la función de enfriado tiene la forma

$$T_{t+1} = \beta[f(\mathbf{x}_t) - \hat{\mu}]^c, \tag{5.20}$$

donde $\beta, c > 0$ son constantes. Donde, $\hat{\mu}$ suele ser una estimación del mínimo global (puede ser hallada por el mismo algoritmo). Cuando el valor estimado de $\hat{\mu}$ es demasiado alto (se consigue fácilmente llegar a un punto que cumple que $f(\mathbf{x}_{t-1}) < \hat{\mu}$), este valor se debe actualzar. Bohachevsky recomienda utilizar valores de $c \approx 1$, pero advierte que la mejor forma de calcular el valor de β es mediante un procedimiento de ensayo-error aplicado al problema en consideración.

En [Marinari y Parisi, 1992] se propone una técnica llamada *Simulated Tempering* donde en lugar de utilizar una función determinista ϕ para actualizar la temperatura T_t, se utiliza una cadena de Markov con densidad invariante una delta en 0 (en manera que $T_t \to 0$ cuando $t \to +\infty$). Es decir, la temperatura al paso t se actualiza muestreando una densidad

$$T_t \sim g(T_t|T_{t-1}),$$

donde con $g(T_t|T_{t-1})$ indicamos una adecuada densidad condicional.

5.3.5. Threshold accepting (TA)

Este método es la analogía determinista de SA, donde la secuencia de temperaturas

$$T_1, T_2, ..., T_t ...,$$

es reemplazada por una secuencia de umbrales $\tau_1, \tau_2, ..., \tau_t .....$ Estos umbrales, al igual que la temperatura, decrecen a medida que aumenta el número de iteraciones. El funcionamiento del algoritmo se explica en el Cuadro 5.6.

En este caso, aceptar un movimiento ascendente no depende de una probabilidad (métodos estocásticos), sino de un umbral dado (métodos deterministas). Si el punto está por debajo del umbral será aceptado, siendo rechazado en caso contrario; tal y como se muestra en la Figura 5.9. Todos las figuras sobre aceptación de puntos candidatos $\mathbf{x}'$ se han realizado por simplicidad para el caso unidimensional, es decir, para $\mathbf{x}' = x \in \mathbb{R}$ escalar.

Threshold accepting
1. Inicialización: Generar la solución inicial $\mathbf{x}_0$ uniformemente en $\mathcal{S}$. Inicializar $i = 0$ como el número de iteraciones y el umbral inicial τ_0.
2. Perturbación: Calcular $\mathbf{x}' \sim \pi(\mathbf{x}_t \vert \mathbf{x}_{t-1})$.
3. Aceptación: Si:
4. $\Delta = f(\mathbf{x}_{t-1}) - f(\mathbf{x}') < \tau_{t-1} \rightarrow \mathbf{x}_t = \mathbf{x}'$.
5. En caso contrario $\rightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$.
Actualizar $\tau_t = \phi(\tau_{t-1})$, donde ϕ es una función de actualización del umbral.
Actualizar $t = t + 1$.
6. Finalización: Parar si se cumple la condición de parada. En caso contrario volver a 2.

Cuadro 5.6

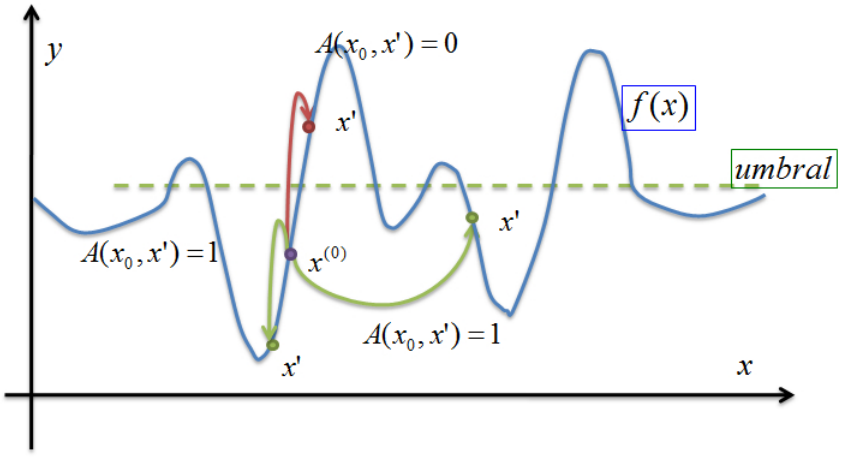

Figura 5.9. Funcionamiento de Threshold Accepting. La probabilidad de aceptar puntos candidatos por debajo de un umbral establecido es 1, mientras que la probabilidad de aceptar puntos candidatos $\mathbf{x}'$ por encima de dicho umbral es 0. Se observa claramente la naturaleza determinista del algoritmo.

5.4. Condición de parada

En esta sección se abordará el problema de cuando parar la ejecución de un algoritmo. Dadas la complicadas características de algunas funciones de coste, es difícil definir un criterio de parada que garantice que el método parará justamente cuando el mínimo global óptimo $\hat{\mathbf{x}}$ sea detectado. De este modo, las condiciones de parada propuestas en la literatura son en su totalidad de

naturaleza probabilística.

En [Bohachevsky y Johnson, 1986] y [Brooks y Verdini, 1988] la técnica de Simulated Annealing se para cuando no se produce aceptación de puntos candidatos $\mathbf{x}'$ durante un número fijo de iteraciones (entre 20-50 en [Bohachevsky y Johnson, 1986] y 50 en [Brooks y Verdini, 1988]). En [Vanderbilt y Louie, 1984] la condición de parada viene determinada por

$$\bar{f} - \hat{\mu} \leq \eta \bar{f} \tag{5.21}$$

donde $\bar{f}$ es la media de las últimas M salidas del algoritmo estocástico, $\hat{\mu}$ es el valor de la función de coste más bajo encontrado hasta la iteración actual y η se refiere a la precisión (*accuracy*) y toma valores del orden de 10^{-3} tanto en [Vanderbilt y Louie, 1984] como en los estudios aportados por [Bohachevsky y Johnson, 1986].

Otras condiciones de parada como la propuesta en [Jones y Forbes, 1995], sugieren detener el algoritmo cuando la medida de la varianza σ_f^2 caiga por debajo de un valor dado (a determinar en función del problema considerado). Esta varianza viene fijada por los puntos visitados por el algoritmo y se actualiza de la forma

$$\sigma_f^2 = c''[f(\mathbf{x}_{t-1}) - \bar{f}] + (1 - c'')\sigma_f^2. \tag{5.22}$$

De esta manera el impacto de las primeras iteraciones se decrementa iteración a iteración, y σ_f^2 sobre todo representa la varianza de la función en las últimas iteraciones.

5.5. ARS y SA: dos filosofías de optimización distintas

Una vez hemos se han expuesto los métodos de trayectoria más significativos es importante parar un momento y reflexionar sobre las filosofías de optimización analizadas. Más en concreto, las técnicas que nos han parecido más relevantes son las utilizadas por Accelerated Random Search y por Simulated Annealing.

Cuando se utiliza Accelerated Random Search el parámetro que cambia a lo largo de la ejecución del método es la *varianza de la función tentativa*. Esta varianza se reducía o ampliaba en función de los puntos candidatos muestreados. Si los puntos candidatos no mejoraban la solución encontrada en la iteración anterior, la varianza se aumentaba; mientras que si los puntos propuestos mejoraban al punto anterior, la varianza se reducía en un factor c ajustable.

Sin embargo, la técnica propuesta por Simulated Annealing es conceptualmente distinta. En Simulated Annealing la densidad tentativa permanecía constante durante todo el proceso. Es el parámetro temperatura el que cambia durante el proceso de optimización. De forma genérica,

esta temperatura tiende a cero a medida que aumenta el número de iteraciones, salvo que se de un reinicio. Pues bien, este cambio en la temperatura se relaciona con un cambio en la densidad objetivo $p_o(\mathbf{x})$ asociada a la función de coste $f(\mathbf{x})$. Es decir, en Simulated Annealing no es la función tentativa la que se ve modificada, sino la *varianza de densidad objetivo* $p_o(\mathbf{x})$.

Este es el punto principal de diferenciación de las dos filosofías. Sin embargo, estas formas de optimizar no son independientes una de otra y podrían fusionarse entre sí. Se podría sugerir, por ejemplo, implementar un algoritmo Simulated Annealing en el que la temperatura controlara de alguna manera la varianza de la función tentativa utilizada para generar puntos candidatos.

5.6. Exploración exhaustiva y precisión: Exigencias contrapuestas

En el análisis de toda técnica de optimización se plantea la resolución de dos objetivos opuestos entre sí. Por un lado, se desea que el método explore todo el espacio $\mathcal{S}$ donde se define la función de coste, y consiga encontrar la zona del espacio donde se encuentra el mínimo global óptimo. Y por otro lado, se quiere generar muchas muestras en la zona del mínimo global, de manera que la solución hallada al final sea una solución exacta, es decir, que sea una solución tan cercana al mínimo global óptimo como sea posible.

Lamentablemente estas dos condiciones no se combinan fácilmente, dado que una exploración exhaustiva del espacio $\mathcal{S}$ suele retardar la búsqueda de la precisión, dado que aunque estemos cerca del mínimo global, proponemos puntos lejanos para asegurarnos de explorar todo el espacio $\mathcal{S}$. Es por ello, que debe encontrarse un compromiso ("*trade-off*") entre estas dos exigencias, que dependerá del problema (aplicación) que se esté resolviendo. Por ejemplo, si se desea simplemente conocer la región del espacio donde se encuentra el mínimo global óptimo sin importarnos la precisión, dedicaremos la mayoría de los recursos a analizar el espacio completo $\mathcal{S}$. Esta necesidad de compromiso entre exhaustividad y precisión se ve de forma muy clara por ejemplo en el algoritmo Accelerated Random Search. Dicho método comienza la búsqueda utilizando varianzas de la función tentativa muy altas (para de este modo explorar todo $\mathcal{S}$), y una vez ha llegado a la región del espacio que contiene el mínimo global óptimo (se aceptan sucesivamente todos los puntos candidatos), comienza a disminuir la varianza de la función tentativa. Esto disminución produce que se generen muchos puntos en la región *interesante*, de forma que se mejore la precisión de la técnica. Pero dado que no se tiene la seguridad de que la solución encontrada sea el mínimo global, después de un cierto número de iteraciones se aumenta otra vez la varianza al

valor original.

5.7. Métodos basados en poblaciones

En contraste con los métodos basados en un solo punto, se trabaja ahora de forma simultánea con un conjunto de puntos (*partículas*) denominado población de M elementos $\mathbf{x} \in \mathbb{R}^m$ (al paso t).

$$\mathbf{X}_t = [\mathbf{x}_{1,t}, \mathbf{x}_{2,t}, ..., \mathbf{x}_{M,t}].$$

Entonces, la matriz $\mathbf{X}_t$ tiene dimensión $m \times M$.

Claramente, estos métodos basados en población suelen ser más eficientes que los métodos que trabajan con un solo punto con respecto a la exploración global de todo el espacio de búsqueda $\mathcal{S}$, pero con una *mayor carga computacional*. El esquema genérico de funcionamiento de estas técnicas se muestra en la Figura 5.10.

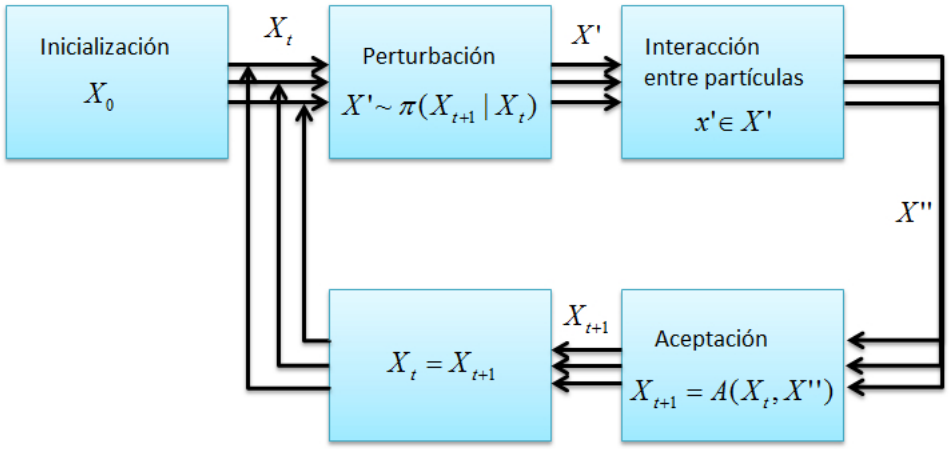

Figura 5.10. Esquema de un método basado en poblaciones. Se inicializa un conjunto de puntos o población $\mathbf{X}_0$. Se proponen un conjunto de soluciones candidatas $x' \in \mathbf{X}'$, que interactúan entre ellos para generar unos nuevos puntos $x'' \in \mathbf{X}''$. Seguidamente estos puntos se aceptan o rechazan con una probabilidad dada $\mathcal{A}(\mathbf{x}', \mathbf{x}'')$, realimentando al bloque de perturbación.

Como podemos notar, en este caso hay un bloque nuevo, que da más "variabilidad" (y capacidad interpretativa) a estas técnicas: se trata de la fase de iteracción entre las diferentes partículas. En realidad, para los algoritmos poblaciones, los bloques de perturbación, interacción entre partículas y aceptación se pueden encontrar fusionados entre sí. Es decir, distintas tareas se pueden hacer conjuntamente. Por ejemplo, un algoritmo puede realizar la perturbación de cada individuo, teniendo ya en cuenta las posiciones de los demás individuos que forman la población,

aglutinando de este modo los bloques de interacción entre partículas y perturbación. Entonces, las diferencia más significativa entre los métodos de una sola partícula y los basados en poblaciones está en la fase de interacción. En los métodos poblacionales, la partículas cooperan entre si, intercambiándose información, mientras que en los métodos de un sólo punto este intercambio no existe.

Desde el punto de vista del muestreo de variable aleatoria, un ejemplo entre interacción (intercambio de información) entre partículas es el *resampling* (ver la Sección 3.4.5).

Los algoritmos metaheurísticos basados en poblaciones usan colecciones de puntos para generar otra colección de puntos distinta. Muchos de estos algoritmos están motivados por procesos biológicos e incluyen desde los algoritmos genéticos o la programación evolucionaria, hasta la optimización por colonia de hormigas. El concepto subyacente en la programación evolucionaria es el de producir descendencia ("*off-spring*") a partir de una población $\mathbf{X}_t$. Esta descendencia, a su vez, será el resultado de la interacción entre las distintas partículas $\mathbf{x}_{i,t} \in \mathbf{X}_t$, $i = 1, ..., M$. La forma de la interacción vendrá fijada por el algoritmo en particular que se esté utilizando. Por ejemplo, en algoritmos genéticos, la interacción viene fijada a través de dos procesos. Uno de combinación o "*crossover*" y otro de mutación.

En algunos de los algoritmos poblaciones, como por ejemplo los algoritmos genéticos, el proceso de aceptación recibe el nombre de "criterio de selección" o "supervivencia". Anteriormente se vio que algunos métodos de trayectoria solo aceptaban movimientos descendentes, es decir, aquellos que cumplieran que $f(\mathbf{x}') < f(\mathbf{x}_{i,t})$. Esto tendría su traducción para algoritmos poblacionales, en un algoritmo que realizara un ranking de la población en cada iteración, basándose en el valor de función de coste $f(\mathbf{x})$ (denominada de idoneidad o "*fitness*") evaluada para cada individuo $\mathbf{x}_{i,t}$ de la población. Sin embargo, en cuanto a algoritmos de poblaciones se refiere, la experiencia ha demostrado que la realización de estos rankings no produce buenos resultados. Se ha verificado que cierto grado de diversidad en la población $\mathbf{X}_t$ debe ser mantenido ("*variety*") para que el algoritmo no converja prematuramente a un mínimo local.

Especialmente interesantes en cuanto a la función de supervivencia o aceptación se refiere son las técnicas basados en la élite poblacional como las estudiadas en [Rardin, 1998], que subdividen la población en tres categorías. Se tendrá por un lado la élite (los mejores puntos), por otro lado los inmigrantes (puntos añadidos para mantener la diversidad), y por último las soluciones de combinación o cruce entre los dos primeros grupos. Claramente, en [Rardin, 1998] se afirma que el tamaño de la población M influye en las prestaciones del algoritmo. Si tenemos poblaciones muy pequeñas, el algoritmo presentará problemas para encontrar el mínimo global óptimo $\hat{\mathbf{x}}$; y

sin embargo, si la población es muy grande entonces el algoritmo será muy ineficiente en cuanto a recursos consumidos. Generalmente, el tamaño de la población M se suele fijar de forma experimental para cada problema.

Si se considera ahora que se trabaja con varios puntos iniciales $\{\mathbf{x}_{0,0}, \mathbf{x}_{1,0}, ..., \mathbf{x}_{M,0} \in \mathbf{X}_0\}$, iterando un mismo algoritmo en paralelo para cada uno de ellos; es posible que, a partir de distintos puntos de comienzo y tras varias iteraciones, se llegue a la misma solución por separado. En este caso, cobra sentido el concepto de agrupamiento o clustering, mediante el cual se podría formar un grupo con todos los puntos que han obtenido el mismo resultado; y asignar unos pesos a las soluciones obtenidas, tal como se hacía en la técnica de "Importance Sampling" vista en 3.4.4. De este modo, cuantos más puntos se acumulen en un mismo grupo o "*cluster*", mayor será la probabilidad de que el mínimo global óptimo $\hat{\mathbf{x}}$ se encuentre cerca. Se podrían establecer así una serie de puntos prometedores o "*Promising Points*", en los que resultaría interesante iniciar una búsqueda local tal y como se realiza en los estudios Schoen [Schoen, 2002] o Torn [Torn y Zilinskas, 1989]).

Los ejemplos más importantes de esta clase de técnicas son sin duda los 3 siguientes: *algoritmos genéticos*, *algoritmos de colonia de hormigas* (Ant Colony Optimization, ACO) y *algoritmos particle swarm* (se observe la Figura 5.1). En literatura, se puede encontrar una variedad enorme de metodos metaheurísticos, pero mucho de poco interés teórico y practico. Sin embargo, después de una atento estudio, nos ha parecido también interesante una variante del SA que se presenta a continuación.

5.7.1. Simulated Annealing para varías partículas (MPSA)

Una primera y trivial posibilidad que siempre puede ser explotada en los algoritmos SA (simulated annealing) es la de adoptar una estrategia de comienzo múltiple, es decir varios SA en paralelo independientes entre sí. Estas técnicas de comienzo múltiple son una manera de solventar los problemas de convergencia demasiado lenta o de exploración poco exhaustiva de $\mathcal{S}$.

Una técnica más sofisticada está descrita en [Molvalioglu et al., 2007], donde se define el algoritmo "*Multiple Particle Simulated Annealing*" que implementa M algoritmos SA trabajando en paralelo pero con "comunicación" entre ellos. Se comienza con M puntos iniciales diferentes

$$\mathbf{X}_0 = \{\mathbf{x}_{1,0}, \mathbf{x}_{2,0}, ..., \mathbf{x}_{M,0}\},$$

elegidos aleatoriamente en $\mathcal{S}$. Se denotará $\mathbf{X}_t = \{\mathbf{x}_{0,t}, \mathbf{x}_{1,t}, ..., \mathbf{x}_{M,t}\}$ al conjunto de puntos de

búsqueda en la iteración t-ésima, En cada iteración se usará un conjunto de funciones tentativas

$$\pi_1(\mathbf{x}_{1,t+1}|\mathbf{x}_{1,t}), \pi_2(\mathbf{x}_{2,t+1}|\mathbf{x}_{2,t}), ..., \pi_M(\mathbf{x}_{M,t+1}|\mathbf{x}_{M,t}), \tag{5.23}$$

para generar un conjunto de soluciones candidatas $\mathbf{X}' = \{\mathbf{x}'_1, \mathbf{x}'_2, ..., \mathbf{x}'_M\}$ y confeccionar el nuevo conjunto de puntos $\mathbf{X}_{t+1}$. También vamos a considerar M funciones de aceptación, con diferentes parámetros temperatura $T_{1,t}, ..., T_{M,t}$,

$$\mathcal{A}_1(\mathbf{x}_{1,t+1}, \mathbf{x}_{1,t}, T_{1,t}), \mathcal{A}_2(\mathbf{x}_{2,t+1}, \mathbf{x}_{2,t}, T_{2,t}), ..., \mathcal{A}_M(\mathbf{x}_{M,t+1}, \mathbf{x}_{M,t}, T_{M,t}). \tag{5.24}$$

El algoritmo funciona del siguiente modo:

1. Fijar $i = 0$ y generar uniformemente M puntos en $\mathcal{S}$. Definir $\mathbf{X}_0 = \{\mathbf{x}_{1,0}, \mathbf{x}_{2,0}, ..., \mathbf{x}_{M,0}\}$. Por ultima fijar la temperaturas iniciales $T_1, ..., T_M$.

2. Generar M puntos candidatos de acuerdo a las funciones tentativas

   - $\mathbf{x}'_1 \sim \pi_1(\mathbf{x}_{1,t+1}|\mathbf{x}_{1,t})$,
   - $\mathbf{x}'_2 \sim \pi_2(\mathbf{x}_{2,t+1}|\mathbf{x}_{2,t})$,
   - ...
   - $\mathbf{x}'_M \sim \pi_M(\mathbf{x}_{M,t+1}|\mathbf{x}_{M,t})$.

3. Un punto $\mathbf{x}'_i$, $i = 1, ..., M$ es aceptado $\mathbf{x}_{i,t+1} = \mathbf{x}'_i$ con probabilidad $\mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i)$.

4. Si $\mathbf{x}'_i$ es rechazado, se definen unos pesos con $j \in \{1, ..., M\}$ y $j \neq i$

   $$w_j = \frac{\mathcal{A}_j(\mathbf{x}_{j,t+1}, \mathbf{x}_{j,t}, T_{j,t})}{\sum_{i=1}^{M} \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_{i,t})}. \tag{5.25}$$

   entonces con probabilidad w_j elegimos $\mathbf{x}_{i,t+1} = \mathbf{x}'_j$.

5. La salida de nuestro algoritmo será $\hat{\mu} = \text{mín}[\mathbf{x}_{1,t+1}, ..., \mathbf{x}_{M,t+1}]$ y si se cumple el criterio de parada se termina.

6. Si no se cumple el criterio de parada, se incrementa $t = t+1$ se actualizan las temperaturas $T_{i,t+1} = \phi_i(T_{i,t})$, $i = 1, ..., M$ y se repete desde el paso 2.

La elección del punto candidato puede resumirse como sigue

$$\mathbf{x}_{i,t+1} = \begin{cases} \mathbf{x}'_i & \text{con probabilidad} \quad \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i), \\ \mathbf{x}'_j & \text{con probabilidad} \quad (1 - \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i))w_j, \end{cases} \tag{5.26}$$

para todos los $j \in \{1, ...M\} \neq i$, y con $\mathbf{x}'_k \sim \pi_2(\mathbf{x}_{k,t+1}|\mathbf{x}_{k,t})$, $k = 1, ..., M$. Se puede escribir más explícitamente de esta forma (si $i \neq 1$ y $i \neq M$)

$$\mathbf{x}_{i,t+1} = \begin{cases} \mathbf{x}'_i & \text{con probabilidad} \quad \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i), \\ \mathbf{x}'_1 & \text{con probabilidad} \quad (1 - \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i))w_1, \\ ... & \\ \mathbf{x}'_{i-1} & \text{con probabilidad} \quad (1 - \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i))w_{i-1}, \\ \mathbf{x}'_{i+1} & \text{con probabilidad} \quad (1 - \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i))w_{i+1}, \\ ... & \\ \mathbf{x}'_M & \text{con probabilidad} \quad (1 - \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i))w_M, \end{cases} \tag{5.27}$$

El modo más sencillo de elegir la función de aceptación $\mathcal{A}$ es haciendo uso de la función de aceptación de Boltzmann vista en la Ecuación 5.7. Obviamente, hay un intercambio de información entre las posiciones $\mathbf{x}_{i,t}$. Concretamente, la probabilidad que las trayectorias se intercambien puntos es $(1 - \mathcal{A}_i(\mathbf{x}_{i,t+1}, \mathbf{x}_{i,t}, T_i))w_j$, con $i = 1, ..., M$ y $j \in \{1, ...M\} \neq i$.

5.7.2. Paralelización

Los algoritmos estocásticos han sabido progresar con el avance de la tecnología, y la paralelización del trabajo utilizando varios procesadores también ha influido en el desarrollo de nuevas técnicas. En [Hamma et al., 1993] se utilizan diferentes procesadores para la generación de puntos candidatos; implementando funciones tentativas $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ distintas en cada procesador. En particular, se implementa la misma función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ con diferentes varianzas de manera que aquellas funciones tentativas con varianza pequeña exploren en entorno local alrededor del punto actual, y las funciones tentativas con varianza grande realicen una exploración global en el entorno del punto anterior $\mathbf{x}_{t-1}$.

5.8. Esquemas de posible hibridación

Por hibridación se entiende la combinación de dos o más algoritmos para la consecución de un fin común, que en este caso es la consecución del mínimo global óptimo. Pueden darse relaciones de muchos tipos entre las que podemos destacar mecanismos de cooperación, sustitución o secuenciación.

En esta sección se reproduce la clasificación expuesta por Talbi en [Talbi, 2002]. Esta clasificación es una combinación de agrupación jerárquica y plana. Esquema jerárquico en cuanto

que se distingue entre nivel bajo de hibridación y nivel alto, y esquema plano o al mismo nivel en cuanto que se puede distinguir entre hibridación de "*relay*" (relevo en equipo) e hibridación co-evolucionaria.

Para obtener hibridación de bajo nivel se reemplaza una parte de un método metaheurístico por la misma parte de otro método metaheurístico distinto (por ejemplo, se cambia la fase de perturbación de un método por la de otro). Una hibridación de alto nivel se caracteriza sin embargo, porque los distintos métodos metaheurísticos de optimización son auto-contenidos.

En cuanto a hibridación al mismo nivel existe hibridación de relevo, en la que los diferentes métodos son aplicados secuencialmente, e hibridación co-evolucionaria, en la que los distintos algoritmos cooperan para la consecución del resultado final.

Es importante también distinguir las hibridaciones no ya en cuanto a interacción entre los distintos métodos sino a la morfología de los mismos. Esta clasificación podría establecerse de la siguiente manera:

- **Hibridación homogénea y heterogénea.** Los métodos híbridos homogéneos combinan algoritmos poblacionales o de trayectoria entre ellos mismos; mientras que en los heterogéneos se combinan algoritmos de trayectoria (basados en un solo punto) con algoritmos poblacionales.

- **Hibridación global y parcial** En los algoritmos globales todos los algoritmos exploran la misma solución mientras que en los parciales el espacio de exploración se encuentra particionado, ocupándose diferentes algoritmos de diferentes particiones.

- **Especializados y generales.** Los algoritmos especializados combinan métodos que permiten solucionar diferentes subproblemas del problema principal, mientras que las técnicas híbridas intentan solucionar el problema principal directamente. Por ejemplo, una técnica que intente optimizar los parámetros de otro método (que resuelva el problema principal) representa un algoritmo especializado.

La siguiente Figura 5.11 ilustra las relaciones posibles entre algoritmos híbridos.

5.9. Aplicaciones

Los métodos metaheurísticos son utilizados para resolver problemas en multitud de disciplinas. Algunos de los entornos de trabajo más característicos son los siguientes:

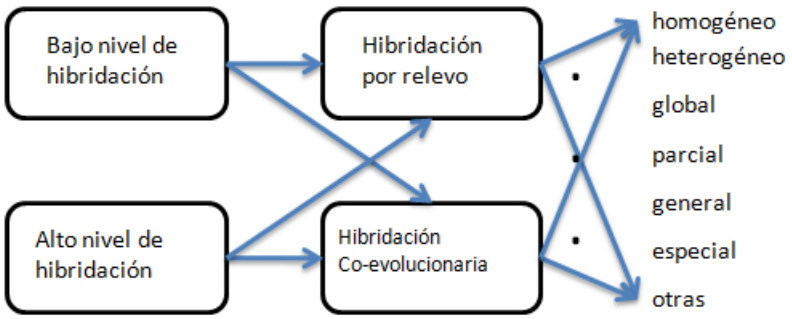

Figura 5.11. Combinaciones posibles entre algoritmos.

Diseño de circuitos Los algoritmos estocásticos se utilizan en el diseño de circuitería compleja de múltiples capas. Cuando el número de dispositivos a integrar en un microchip asciende a decenas de cientos de componentes, la colocación de cada componente y las conexiones necesarias supone un problema de optimización del espacio, para que este sea lo menor posible.

Matemáticas El diseño de árboles de decisión y clasificación eficientes constituye un problema NP-completo para el que suelen utilizarse técnicas metaheurísticas. Otro tipo de problemas en este ámbito son los relacionados con la diagonalización de matrices de elevadas dimensiones.

Procesado de imágenes La reconstrucción de imágenes y el reconocimiento de formas son problemas en los que habitualmente se utilizan técnicas de búsqueda local. En la reconstrucción de imágenes se necesita del diseño de filtros, que vienen definidos a través de un conjunto de parámetros que será necesario optimizar, para que se ajusten a los datos dados en la imagen. De esta manera, el filtro será capaz de reconstruir la imagen correctamente.

Biología Casi todos los problemas en los que se utiliza algoritmos metaheurísticos se encuadran en el campo de la biología molecular. Un problema complicado para el que estos métodos resultan muy eficientes es el de definir la estructuras de las cadenas de proteínas (por ejemplo, la conformación de los péptidos). Las técnicas de búsqueda local también se utilizan para determinar los alineamientos existentes en cadenas largas de ADN (ácido desoxirribonucleico) y ARN (ácido ribonucleico).

Geo-física Los algoritmos estocásticos se ha probado que son útiles para modelar formas de onda sísmicas. Se hace notar que la naturaleza estocástica de Simulated Annealing, es de hecho la ventaja principal de este tipo de técnicas, ya que la naturaleza imprevisible de los terremotos

es una de sus características más relevantes.

Aplicaciones militares El despliegue óptimo de recursos es un problema importante que se convierte en crucial cuando se habla de seguridad nacional o política de defensa. Se han desarrollado aplicaciones militares basadas en métodos de búsqueda local sobre como deben desplegarse las defensas anti-misiles a lo largo de una ciudad o territorio, así como aplicaciones de rastreo de enemigos en mar y aire.

Finanzas Existen aplicaciones financieras de cálculo de intereses entre mercados, cambio de divisas o de estimación de la productividad de un determinado producto en la que se busca optimizar una serie de parámetros, de manera que el beneficio obtenido o la rentabilidad (a corto, medio o largo plazo) sea máxima.

5.10. Conclusiones

A lo largo de este capítulo se ha intentado realizar una clasificación de los métodos estocásticos existentes en la actualidad que más nos han interesado. Se dividieron estas técnicas en algoritmos exploratorios y de aproximación estocástica. Entre los denominados métodos exploratorios o de búsqueda local podemos distinguir los algoritmos de trayectoria y los basados en poblaciones. Destacan entre los del primer grupo Simulated Annealing, Accelerated Random Search; y en el segundo los algoritmos genéticos o el algoritmo Simulated Annealing para varias partículas.

Se ha hecho especial énfasis en la relación entre Simulated Annealing y el algoritmo Metropolis-Hastings visto en el Capítulo 4. Éste último proporciona la base teórica que explica el porqué del buen funcionamiento de Simulated Annealing.

También se ha hablado largo y tendido durante este capítulo sobre las filosofías de optimización que plantean Simulated Annealing y Accelerated Random Search. La filosofía del primero se basa en la modificación de la densidad objetivo $p_o(\mathbf{x})$, mientras que el segundo trabaja modificando la varianza de la función tentativa $\pi(\mathbf{x_t}|\mathbf{x_{t-1}})$.

Se ha visto como diferentes modos de encarar la optimización pueden dar buenos resultados, siempre y cuando se alcance un compromiso aceptable entre exploración rigurosa del espacio $\mathcal{S}$ y precisión (*accuracy*) de los resultados.

Capítulo 6

Implementando Simulated Annealing y Acelerated Random Search

En este capítulo se implementarán en Matlab los algoritmos Simulated Annealing y Acelerated Random Search para optimizar la función de coste

$$f(x,y) = (x\sin(20y) + y\sin(20x))^2\cosh(\sin(10x)x) + (x\cos(10y) + y\sin(10x))^2\cosh(\cos(20y)y). \tag{6.1}$$

que tiene la forma mostrada en la Figura 6.1.

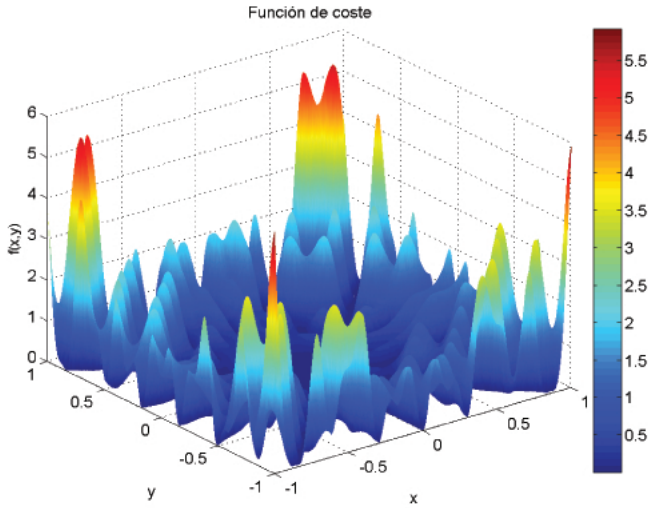

Figura 6.1. Función de coste.

Se observa que dicha función de coste presenta numerosos mínimos locales, situándose el mínimo global óptimo en el punto $(x,y) = (0,0)$. Para poder distinguir mejor el valor de los distintos

mínimos locales en ocasiones se trabajará con $\log(f(x,y))$. La representación de $\log(f(x,y))$ se muestra en la Figura 6.2. A lo largo de este capítulo también se utilizará la representación de las

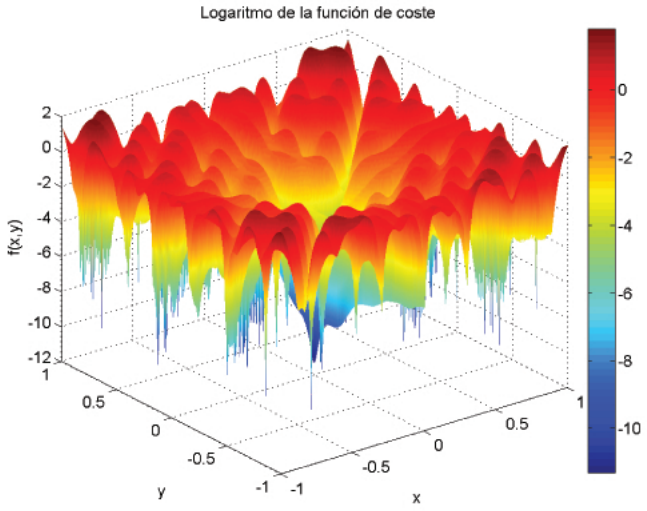

Figura 6.2. Logaritmo de la función de coste. Se representa la función $\log f(x,y)$, donde se distingue mejor entre dos mínimos locales distintos.

curvas de nivel de $\log(f(x,y))$ mostradas en la Figura 6.3. Estas curvas de nivel se utilizarán so-

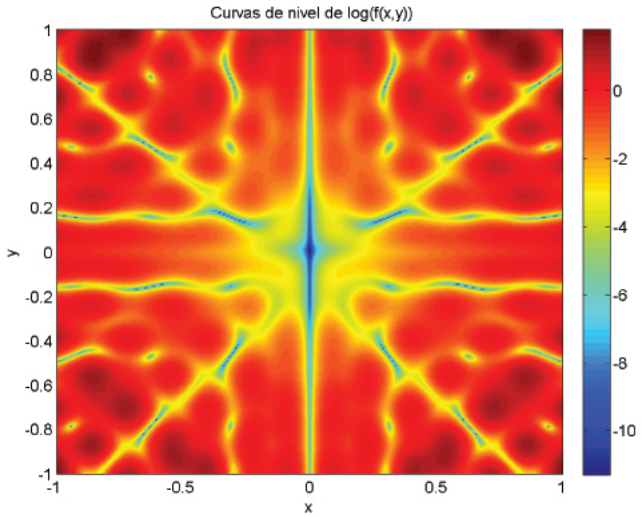

Figura 6.3. Curvas de nivel de $\log(f(\mathbf{x}))$. Se puede apreciar la existencia de numerosos mínimos locales alrededor del mínimo global óptimo situado en $(x,y) = (0,0)$.

bre todo para mostrar la evolución de los algoritmos Simulated Annealing y Acelerated Random Search a medida que aumenta el número de iteraciones.

Se empezará implementando el algoritmo Simulated Annealing, realizando variaciones en sus parámetros y analizando su funcionamiento. Seguidamente, se implementarán Acelerated Random Search y Simulated Annealing para varias partículas operando de forma similar. Por último, se compararán y discutirán los resultados obtenidos, comentando las ventajas e inconvenientes de usar un algoritmo u otro.

6.1. Implementación de Simulated Annealing

Tal y como se expuso en el Capítulo 5 Simulated Annealing opera de la siguiente forma:

1. Inicializar $\mathbf{x}_0 \in S$, el número de iteraciones $t = 0$ y la temperatura inicial T_0.
2. Generar un punto candidato $\mathbf{x}' \sim \pi(\mathbf{x}_t|\mathbf{x}_{t-1})$.
3. Generar $u' \sim \mathcal{U}([0,1])$.
4. Si $u' > \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') \rightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$.
5. Si $u' \leq \mathcal{A}(\mathbf{x}_{t-1}, \mathbf{x}') \rightarrow \mathbf{x}_t = \mathbf{x}'$.
6. Actualizar la tempera $T_t = \phi(T_{t-1})$. Actualizar $t = t + 1$.
7. En caso de cumplirse el criterio de parada terminar. En caso contrario volver a 2.

Se observa que los parámetros temperatura inicial T_0, función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$, función de enfriamiento ($T_t = \phi(T_{t-1})$) y condición de parada no están definidos explícitamente. En función de estos parámetros el algoritmo se comportará de manera distinta. Nuestro cometido en las siguientes secciones será evaluar el comportamiento del algoritmo para distintos valores de los parámetros, y diferenciar que valores nos interesan. Es decir, calcular cuales son los parámetros óptimos.

6.1.1. Primera Prueba

En esta primera prueba se verá como cambian las prestaciones del algoritmo en función de la temperatura inicial T_0 escogida. Para ello, se elegirá una temperatura inicial T_0 variante y se dejarán fijos los siguientes parámetros:

- El punto de comienzo $\mathbf{x}_0$ se restringirá a sólo 4 valores situados a distancia 4 del mínimo global, elegiendose aleatoriamente en el conjunto $\{(4,4);(-4,-4);(4,-4);(-4,4)\}$.

- Como condición de parada se impuso un número de iteraciones fijas $N_{Max} = 60000$.

- Como ley de enfriamiento se va a utilizar la expresión $T_t = \alpha T_{t-1}$, donde la constante α se fijó al valor de 0.99.

- Como función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ se utilizó una distribución normal centrada en el punto anterior de búsqueda $\mathbf{x}_{t-1}$ con matriz de covarianzas dada por:

$$\mathbf{\Sigma}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.2}$$

  donde se observa claramente que la varianza de las v.a. X, Y es 1, estando incorreladas entre sí.

- En caso de que el algoritmo quedará estancado durante 12 iteraciones en un punto cuyo valor de la función de coste sea mayor que el *mejor* valor $\mathbf{x}_{best}$ encontrado hasta el momento se efectuará un reinicio. El reinicio consistirá en fijar $\mathbf{x}_{t-1} = \mathbf{x}_{best}$ y $T = T_0$.

La temperatura inicial T_0 se hará variar entre los valores $T_0 = 0.5$ y $T_0 = 100$, repitiendo 10000 veces la medida para cada valor, de manera que se puedan establecer unos estadísticos de media y varianza de los resultados.

En la Figura 6.4 se puede ver el valor medio de la función de coste $f(x, y)$ conseguido para cada valor de la temperatura inicial T_0 simulado.

Para valores de la temperatura inicial $T_0 < 10$ el algoritmo parece que queda enganchado en mínimos locales de la función de coste. Esto es provocado porque un valor demasiado bajo de la temperatura repercute en una probabilidad de aceptación de puntos candidatos $\mathbf{x}'$ demasiado baja. Se aprecia que la variabilidad de los resultados en el intervalo $[0, 10]$ es elevada (aproximadamente $6e-4$) debido a que el algoritmo converge cada vez en un mínimo local distinto (y por lo tanto, con distinto valor de la función de coste). Por otro lado, para valores de $T_0 > 80$ la convergencia del algoritmo se hace demasiado lenta, haciendo que se llegue a valores medios de la función de coste $f(x, y)$ más altos. Una temperatura alta provoca probabilidades de aceptación demasiado altas, que hacen que el algoritmo tarde más en converger. Las temperaturas iniciales con las que se llega a mejores resultados se encuentran en el intervalo $T_0 \in [10, 80]$. Por ejemplo, para $T_0 = 40$, el valor medio de la función de coste medio obtenido es 2.2e-7.

A continuación, en lugar de medir el valor medio de la función de coste obtenido para cada valor de la temperatura inicial T_0, se medirá la distancia al mínimo global óptimo situado en

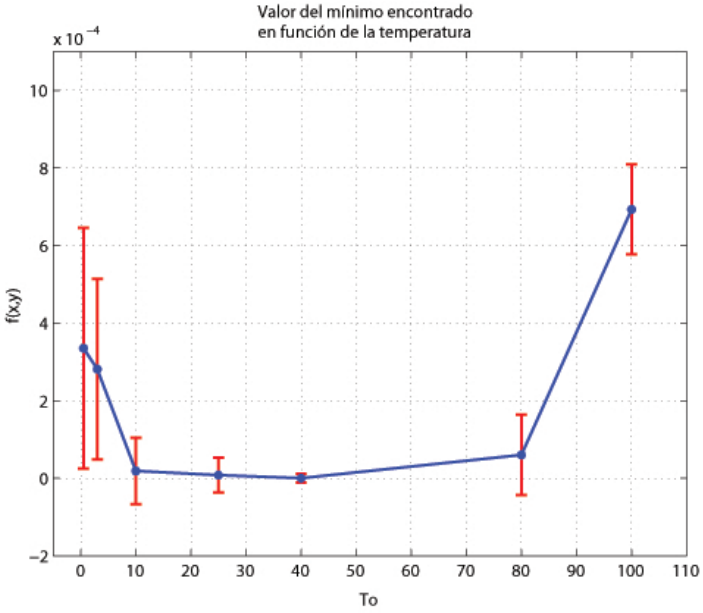

Figura 6.4. $f(x, y)$ vs T_0. Se representa el valor medio de la función de coste $f(x, y)$ en función de la temperatura inicial T_0 utilizada.

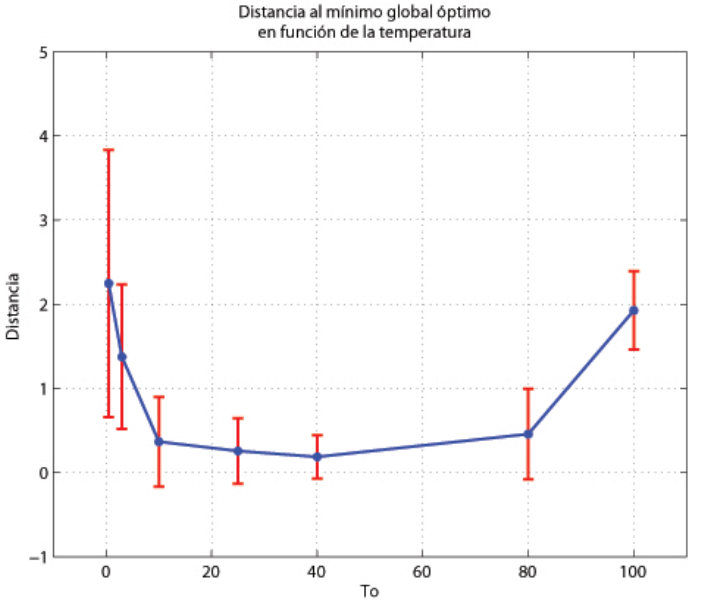

Figura 6.5. Distancia en función de T_0. Se representa la distancia media al mínimo global óptimo en función de la temperatura inicial T_0.

el punto $(0, 0)$. La distancia al mínimo se calculó, a partir de las coordenadas del punto, como $\sqrt{x^2 + y^2}$, y los resultados obtenidos se muestran en la Figura 6.5.

Se verifica que para valores de la temperatura bajos el algoritmo queda anclado en mínimos locales lejanos al mínimo global óptimo. Sin embargo, para $T_0 \in [10, 80]$ el algoritmo conver-

ge adecuadamente, obteniendo valores muy cercanos al mínimo global óptimo (a distancia 0.2 aproximadamente) en todas las ocasiones. De ahí que la variabilidad en este intervalo sea la más reducida.

Una vez analizada la distancia al mínimo global óptimo se decidió realizar una prueba para analizar la velocidad de convergencia del algoritmo. Para ello, en lugar de dejar fijo el número de iteraciones del método, se fijó como objetivo para la finalización del algoritmo la consecución de un valor de la función de coste menor que $1e-6$. Cuando el algoritmo llegará a este valor la técnica finalizaría, guardando el número de iteraciones necesarias. La evolución en el número de iteraciones necesarias en función de la temperatura inicial T_0 se muestra en la Figura 6.6.

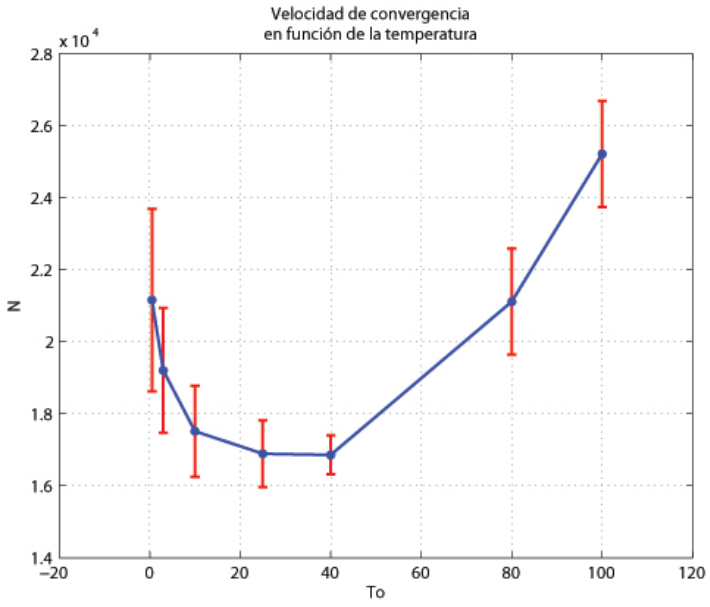

Figura 6.6. Velocidad de convergencia en función de T_0. Se representan las iteraciones medias necesarias para conseguir el objetivo fijado en función de la temperatura inicial T_0.

Para valores de la temperatura inicial muy bajos la variabilidad es muy alta. En ocasiones, el algoritmo convergía muy rápido y en otras ocasiones se necesitaban muchas iteraciones para obtener la convergencia. Los valores óptimos de la temperatura inicial se encuentran en el intervalo $T_0 \in [20, 40]$, donde tan solo se necesitan una media 17000 iteraciones para converger. Es por ello que para las siguientes pruebas (en las que se variarán otros parámetros) se utilizará un valor de la temperatura inicial de $T_0 = 40$.

También se calculó la probabilidad de aceptación media de puntos candidatos en función de la temperatura inicial T_0. Los resultados se muestran en la Figura 6.7.

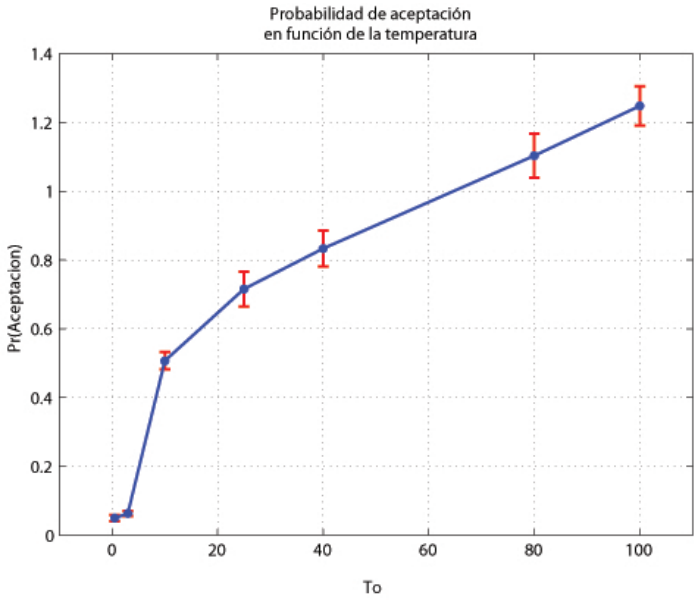

Figura 6.7. Evolución de la probabilidad de aceptación. Se representa el valor de la probabilidad de aceptación con respecto a la temperatura inicial T_0 utilizada.

Como es lógico, a medida que aumenta la temperatura inicial, la probabilidad de aceptación se va haciendo mayor. Se recuerda que la probabilidad de aceptación de un punto candidato $\mathbf{x}'$ es 1 si $f(\mathbf{x}') < f(\mathbf{x}_{t-1})$ y menor que uno en caso contrario. En concreto, esta probabilidad de aceptar un punto peor es de la forma $\exp(-(\frac{f(\mathbf{x}')-f(\mathbf{x}_{t-1})}{T}))$, siendo la probabilidad de aceptar mayor, cuanto mayor es la temperatura. La probabilidad de aceptación mostrada es la probabilidad de aceptación media durante todo el proceso. Lógicamente para las primeras iteraciones (temperatura alta) la probabilidad de aceptación es alta, reduciéndose a medida que el algoritmo se encamina hacia el mínimo. Para el caso concreto de $T_0 = 40$, la probabilidad de aceptación media es de aproximadamente el 1 %. Sin embargo, no es constante durante todo el proceso, sino que para las primeras iteraciones la probabilidad de aceptación es casi del 45 %, siendo de tan solo el 0,2 % para las últimas iteraciones.

Para finalizar esta primera prueba se muestra un ejemplo de la evolución (sobre las curvas de nivel de $\log f(\mathbf{x})$) del algoritmo para un valor de la temperatura inicial $T_0 = 40$ en la Figura 6.8.

Se observa como el algoritmo explora todo el espacio de búsqueda hasta aproximarse a la cuenca de atracción del mínimo global óptimo. A partir de ese momento, se concentra en esa región del espacio.

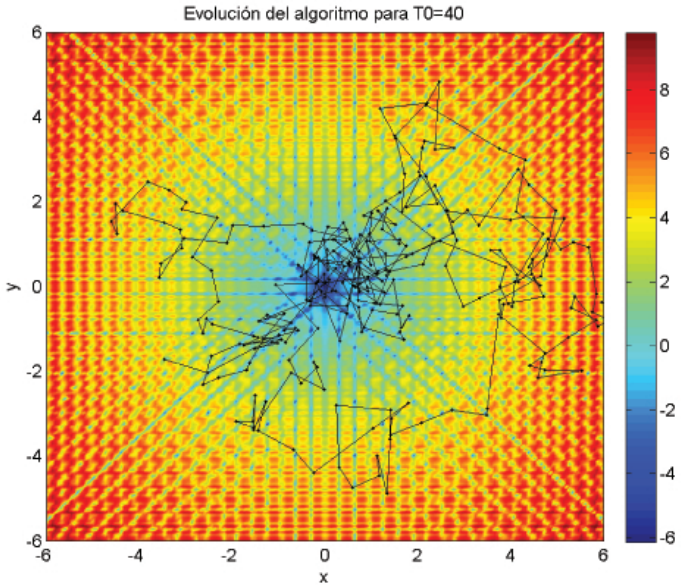

Figura 6.8. Se representa la evolución del algoritmo a medida que aumentan las iteraciones.

6.1.2. Segunda prueba

A continuación, se variará el número de iteraciones N y se calculará el valor medio de la función de coste $f(x, y)$ conseguido. Por supuesto, cuanto mayor sea el número de iteraciones mejores serán los resultados obtenidos. Sin embargo, se deberá encontrar un compromiso entre número de iteraciones y valor del mínimo encontrado, para que de este modo las simulaciones consuman un tiempo razonable. La temperatura inicial utilizada será $T_0 = 40$, con un valor de la constante de enfriamiento $\alpha = 0{,}99$. Se modificará el número de iteraciones desde $N = 2000$ hasta $N = 200000$. Se muestran los valores medios de la función de coste conseguidos en la Figura 6.9.

Se observa como el valor medio de $f(x, y)$ es menor a medida que aumenta el número de iteraciones, tal y como era previsible. Para $N < 5000$, el algoritmo no converge adecuadamente por lo que la variabilidad en los resultados es elevada. En la Figura 6.10 se muestra la distancia media al mínimo global óptimo de las soluciones encontradas en función del número de iteraciones.

Se aprecia como a medida que se aumentan el número de iteraciones la distancia media al mínimo global disminuye. Para $N < 10000$, el número de iteraciones es insuficiente e influye bastante en los resultados, mientras que para $N > 20000$, los cambios en los resultados son mínimos. Para guardar cierto compromiso entre calidad de los resultados y tiempo de simulación

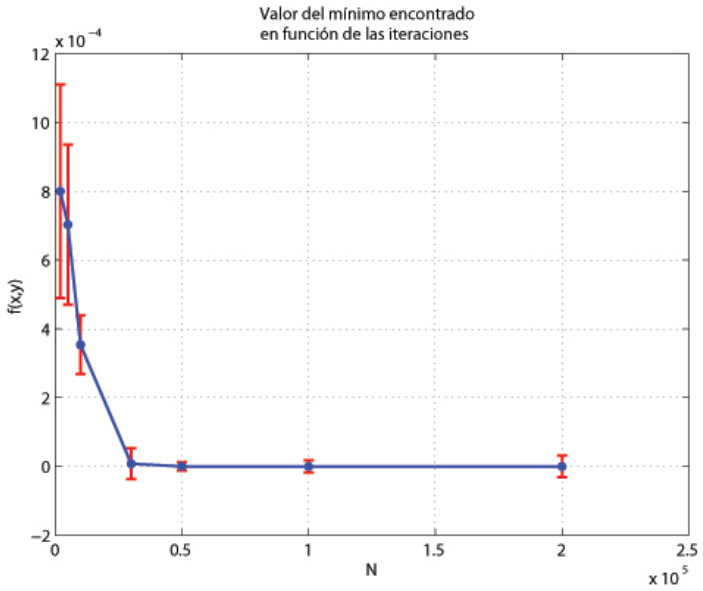

Figura 6.9. $f(x, y)$ vs N. Se representan los valores de la función de coste $f(x, y)$ en función del número de iteraciones N, así como la varianza de los resultados obtenidos.

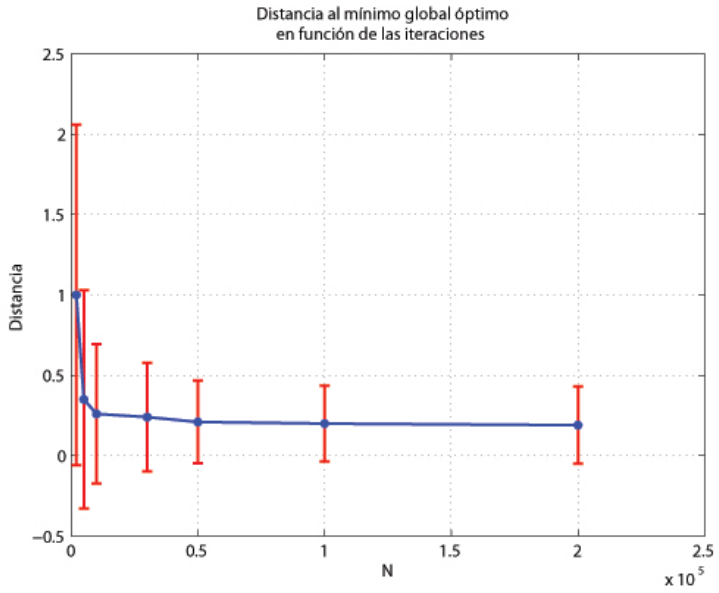

Figura 6.10. Distancia al mínimo en función de N. Se representa como varía la distancia al mínimo global óptimo a medida que el número de iteraciones crece.

a partir de esta prueba se utilizará un valor de máximo de $N = 30000$.

6.1.3. Tercera prueba

En esta prueba se probará el comportamiento del algoritmo para distintos valores de la constante de enfriamiento α (se recuerda que $T_t = \alpha T_{t-1}$). En concreto, se simulará con valores comprendidos entre $\alpha = 0.2$ y $\alpha = 1$, para un número de iteraciones fija $N = 30000$, y con valor de la temperatura inicial $T_0 = 40$. El valor medio de la función de coste para los mínimos encontrados puede verse en la Figura 6.11, junto con la variabilidad de los resultados.

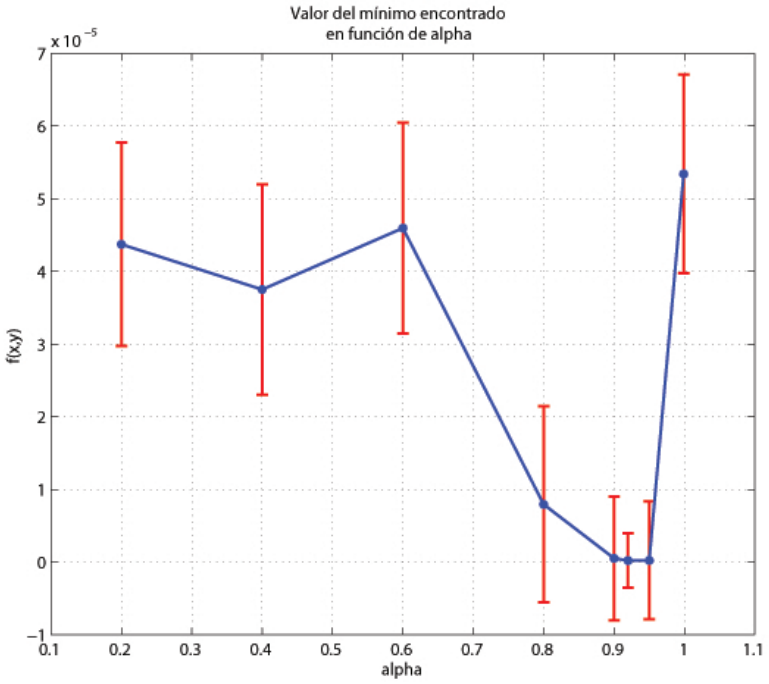

Figura 6.11. Valor de la función de coste $f(x, y)$ para distintos valores de la constante de enfriamiento α.

Para valores de α por debajo de 0.8, el algoritmo converge muy rápido pero no al mínimo deseado, por lo que los valores de la función de coste conseguidos no son muy buenos. Por otro lado, para valores de $\alpha > 0.96$, la convergencia es muy lenta. En la Figura 6.12 se muestra como de lejos se encuentran en media los mínimos encontrados respecto del óptimo, en función de la constante de enfriamiento.

Parece claro que intervalo para el cual el algoritmo no queda enganchado en mínimos locales es el correspondiente a $\alpha \in [0.8, 0.96]$. Seguidamente se intento obtener datos sobre la velocidad de convergencia para estos valores *óptimos* de α, obteniendo los resultados que se muestran en la Figura 6.13 (como condición de parada se impuso la consecución de un mínimo con valor de $f(x, y) \leq 1e - 6$).

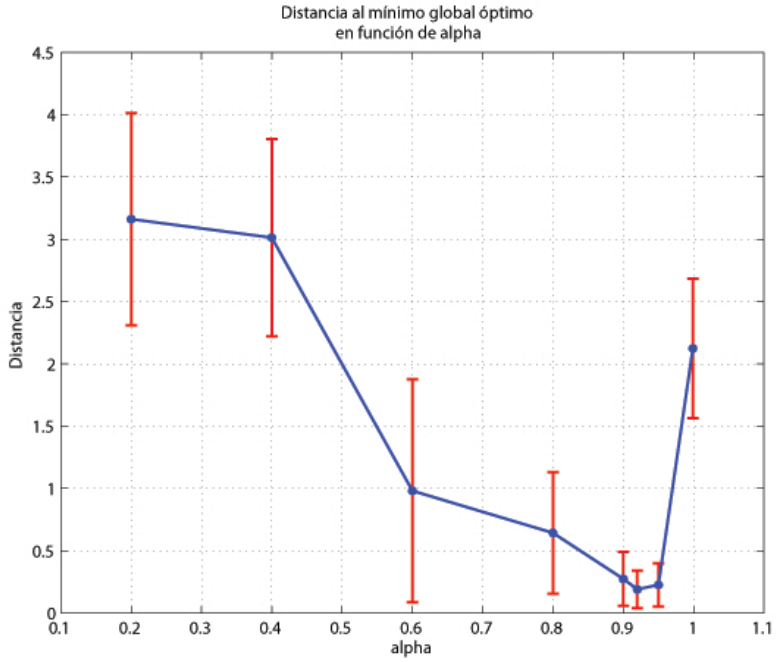

Figura 6.12. Distancia al mínimo global óptimo en función de α.

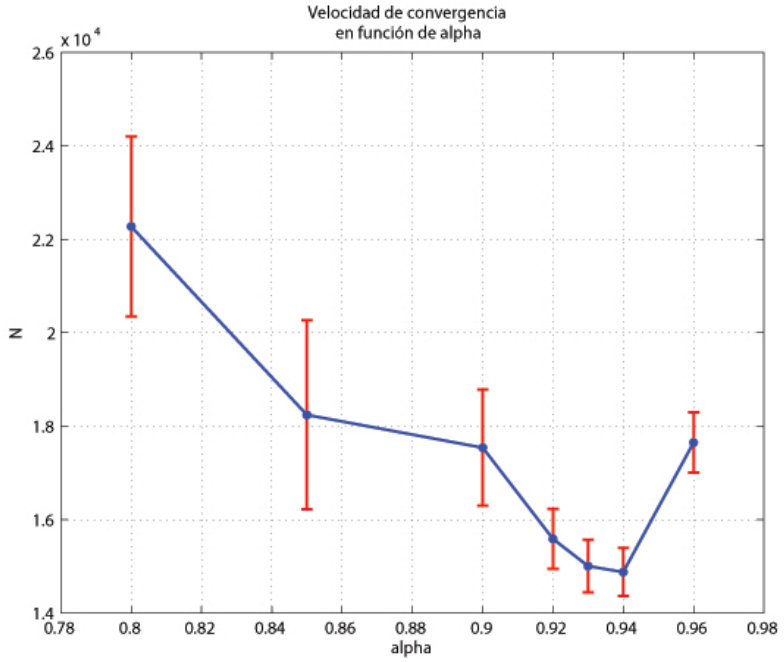

Figura 6.13. Velocidad de convergencia en función de α. Se representa el número de iteraciones hasta la obtención del criterio de parada del método en función de α

Para el intervalo $\alpha \in [0.92, 0.96]$ la técnica de Simulated Annealing converge muy rápido y con poca variabilidad en los resultados. Es por ello, que para las siguientes pruebas se tomará un valor de $\alpha = 0.94$.

6.1.4. Cuarta prueba

Se modificará ahora la función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ para generar nuevos puntos candidatos $\mathbf{x}'$. Se decidió seguir utilizando una normal bidimensional para generar nuevos puntos pero se impuso como parámetro variable la varianza tanto en el eje x como en el eje y. Para un número fijo de $N = 30000$ iteraciones, $\alpha = 0.94$ y $T_0 = 40$ se calculó el valor medio de la función de coste para distintas varianzas. Los resultados obtenidos se muestran en la Figura 6.14.

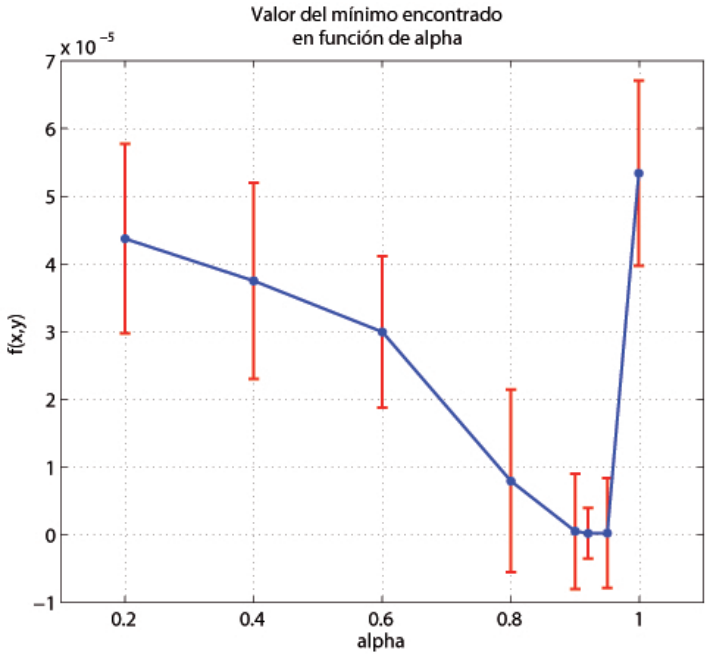

Figura 6.14. Variando la función tentativa. Valor medio de la función de coste para distintos valores de la varianza de la función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$.

Se observa como el mejor funcionamiento del algoritmo se da para el intervalo de la varianza $[0.5, 2]$. Para valores de la varianza por debajo de 0.5, los movimientos a puntos muy lejanos son muy improbables, lo que hace que el algoritmo converja despacio y obtenga peores resultados que para varianzas mayores. Por el contrario, para valores de la varianza mayores de 2, los saltos a puntos muy lejanos tienen mayor probabilidad de producirse, haciendo que el algoritmo se comporte de forma casi aleatoria y no converja adecuadamente. De ahí, la gran variabilidad de los resultados para varianzas elevadas.

A continuación se calculó la distancia media al mínimo global óptimo para distintos valores de la varianza, obteniendo los resultados que se muestran en la Figura 6.15.

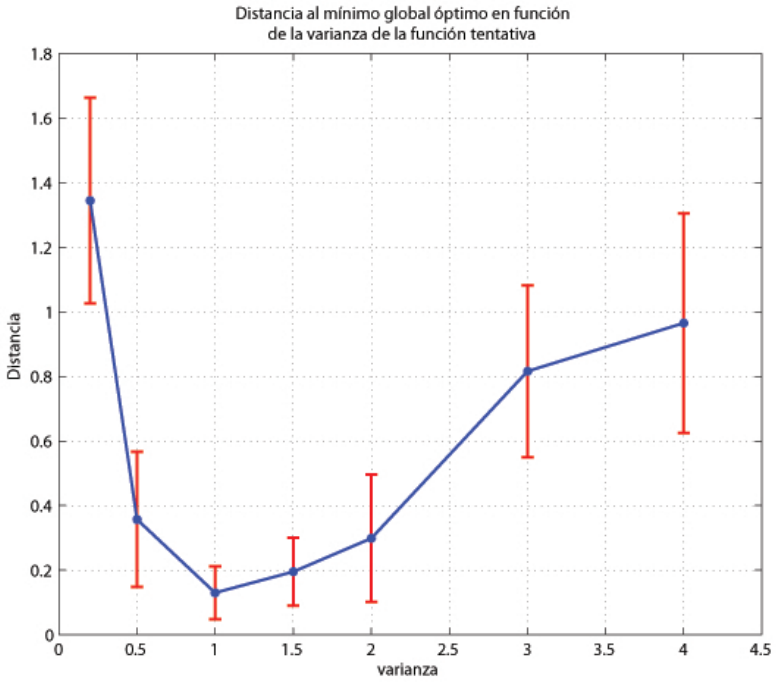

Figura 6.15. Distancia media al mínimo global óptimo en función de la varianza de la función tentativa.

Para valores de la varianza demasiado reducidos, el algoritmo converge lentamente, quedando alejado del mínimo global óptimo. Sin embargo para valores de la varianza demasiado altos, el algoritmo no converge adecuadamente y los resultados obtenidos presentan una variabilidad muy alta. De la Figura 6.15 se concluye que el intervalo óptimo de valores para la varianza es el $[0.5, 2]$.

Llegados a este punto puede resultar interesante mostrar la evolución del método (sobre las curvas de nivel de $\log(f(\mathbf{x}))$) para un valor de la varianza alto y para uno bajo. En concreto se muestra la evolución de la técnica para unos valores de $\sigma^2 = 0.4$ y $\sigma^2 = 5$ en las Figuras 6.16 y 6.17 respectivamente.

Se puede apreciar fácilmente como para valore de σ^2 pequeños la longitud media de los movimientos es muy pequeña, desembocando en una convergencia lenta del algoritmo. Sin embargo para valores de la varianza muy altos el comportamiento de la técnica casi podría calificarse de aleatorio, eligiendo puntos muy distantes entre sí (de una iteración a otra).

A continuación, se realizaron pruebas para analizar la velocidad de convergencia en función de la varianza. Para ello se puso como condición de parada la consecución de un mínimo cuya función de coste fuera menor o igual que $1e{-}6$. En ese momento se pararía la ejecución del método

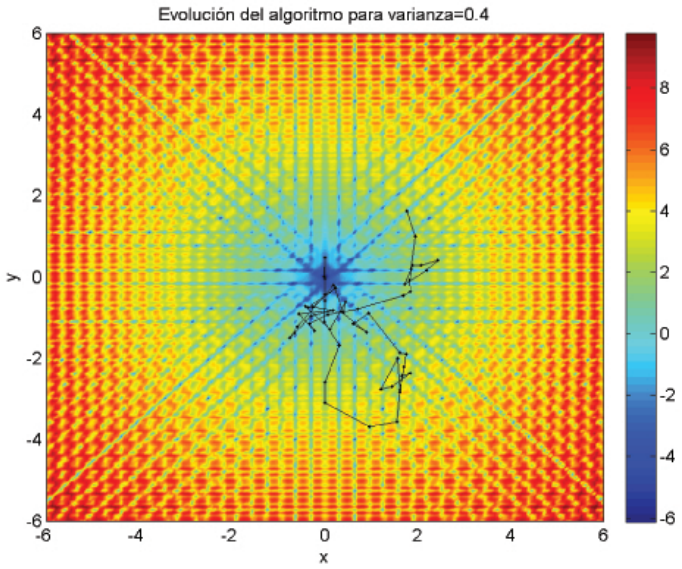

Figura 6.16. Evolución del algoritmo para un valor de $\sigma^2 = 0.4$.

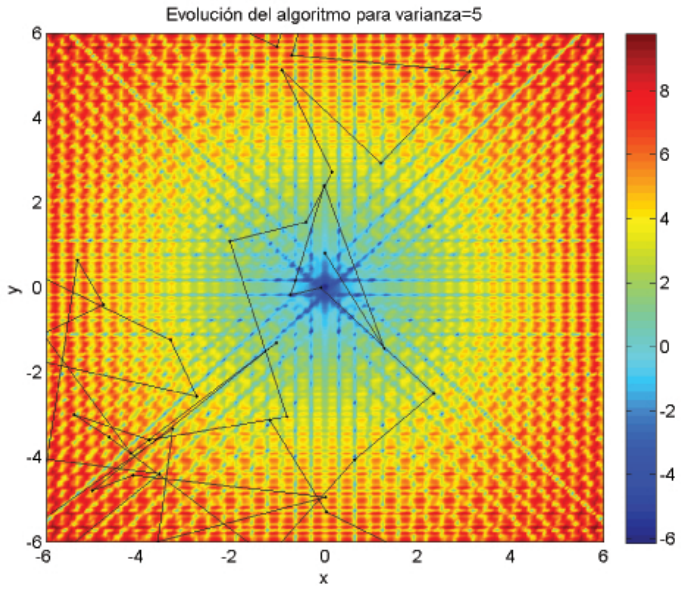

Figura 6.17. Evolución del algoritmo para un valor de $\sigma^2 = 5$.

y se tomaría nota de las iteraciones realizadas hasta el momento. Los resultados obtenidos se muestran en la Figura 6.18.

Los mejores resultados parecen obtenerse para un valor de $\sigma^2 = 1.2$. Es por ello que a partir de esta prueba, se utilizó este valor óptimo para la varianza.

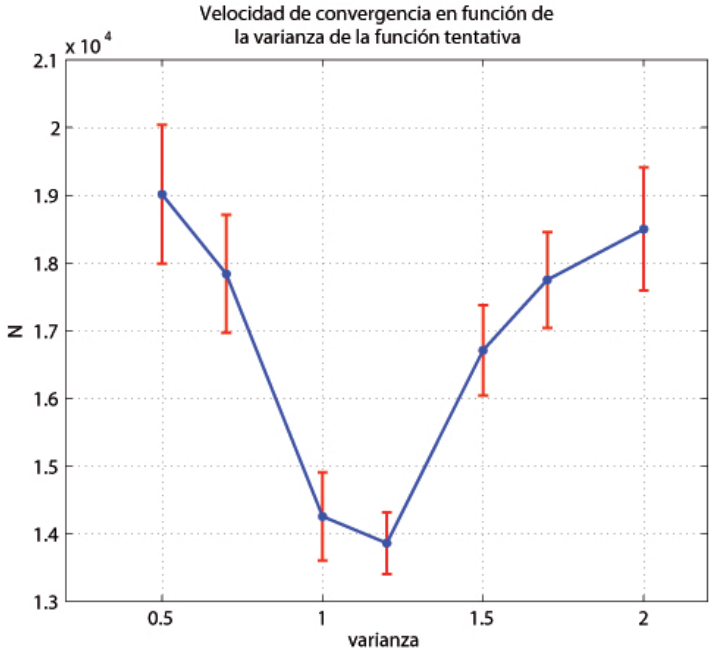

Figura 6.18. Evaluación de la velocidad del algoritmo para distintos valores de σ^2.

6.1.5. Quinta Prueba

En esta prueba se verá como se comporta el método ante la exploración anisótropa del espacio. Ese decir, ante aquella exploración que beneficia unas direcciones en detrimento de otras. Para implementar este hecho se utilizará como función tentativa una gaussiana bidimensional con cierto grado de dependencia entre las variables aleatorias X e Y. Las matriz de covarianzas genérica utilizada será:

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & b \\ b & 1 \end{bmatrix} \tag{6.3}$$

donde $b = \mathbb{E}[(X - \mu_x)(Y - \mu_y)]$ tomará los valores [0, 0.35, -0.35, 0.75, -0.75]. Las curvas de nivel de la función tentativa para $b = 0$ se muestran en la Figura 6.19.

Por otro lado, las curvas de nivel de la función tentativa para $b = 0.35$ y $b =$ -0.35 se muestran en la Figura 6.20.

Por último, las curvas de nivel de la función tentativa para $b = 0.85$ y $b =$ -0.85 se muestran en la Figura 6.21.

Para un número fijo de $N = 30000$ iteraciones y con un valor de la temperatura inicial de $T_0 = 40$ se simuló la técnica con el fin de calcular el valor medio de la función de coste para las distintas funciones tentativas vistas (en función de b). Los resultados se muestran en la Figura

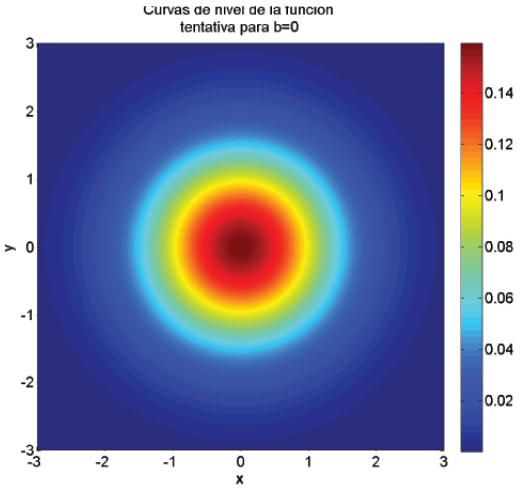

Figura 6.19. Curvas de nivel de la función tentativa para X, Y incorreladas.

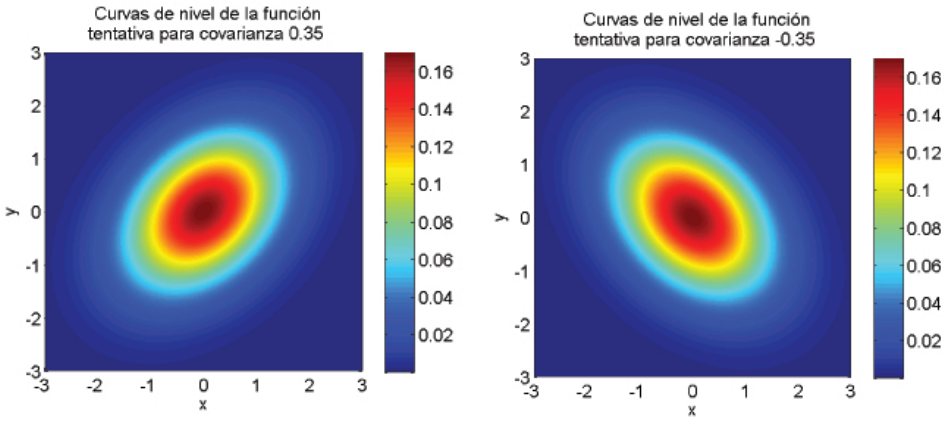

Figura 6.20. Curvas de nivel de la función tentativa para $b = \pm 0.35$. A la izquierda para $b = 0.35$ y a la derecha para $b =$ -0.35.

6.22.

Los mejores resultados se obtienen cuando la función tentativa es simétrica respecto a los ejes x, y, sin favorecer la exploración en una dirección determinada. En la Figura 6.23 se muestra la distancia al mínimo global óptimo obtenida para cada uno de los valores de b simulados.

En esta Figura se certifica que la técnica da sus mejores prestaciones para una exploración isótropa del espacio. Para valores de b muy altos, el algoritmo no optimizaba correctamente porque la exploración del espacio no era la adecuada, provocando una alta variabilidad en los resultados.

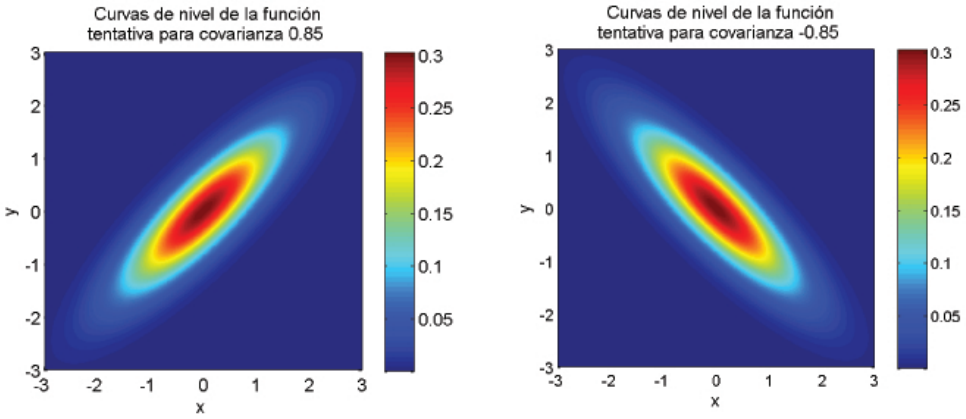

Figura 6.21. Curvas de nivel de la función tentativa para $b = \pm 0.85$. A la izquierda para $b = 0.85$ y a la derecha para $b =$ -0.85

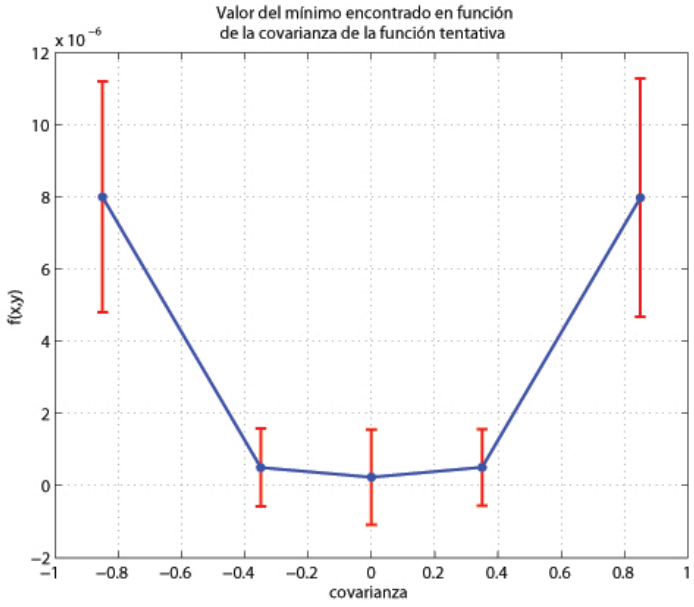

Figura 6.22. Valor medio de la función de coste $f(x, y)$ en función del valor de b.

6.1.6. Sexta Prueba

En esta sexta prueba se analizará el comportamiento del método con una ley de enfriamiento ("*cooling schedule*") distinta a la que se viene utilizando hasta ahora. En concreto se cambiará la ley ($T_t = \alpha T_{t-1}$) por ($T_t = T_0 \exp(-(1-c)t)$) con constante $c \in [0, 1]$. La ley de enfriamiento presenta la forma mostrada en la Figura 6.24 para un valor de $T_0 = 40$.

Claramente, cuanto mayor sea el valor de c, la temperatura descenderá más lentamente. El valor de c es de especial importancia en el algoritmo porque determinará la velocidad de

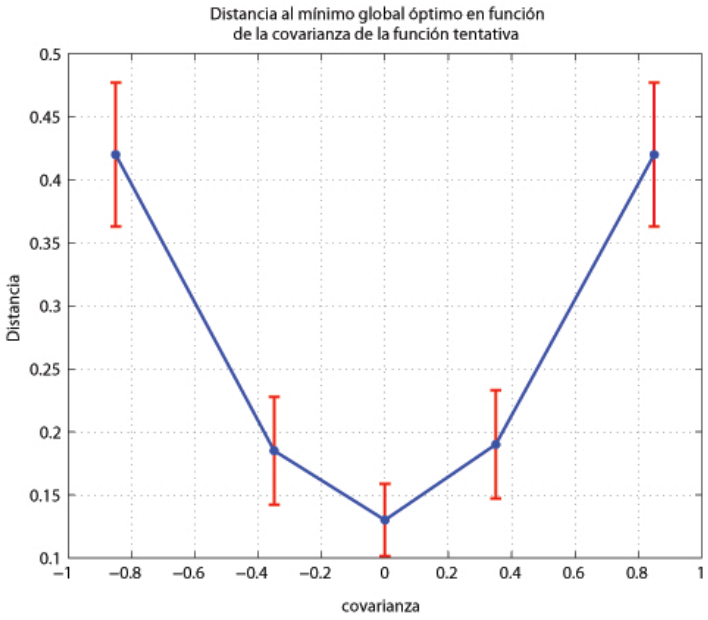

Figura 6.23. Distancia media al mínimo global óptimos en función del valor de b en la función tentativa.

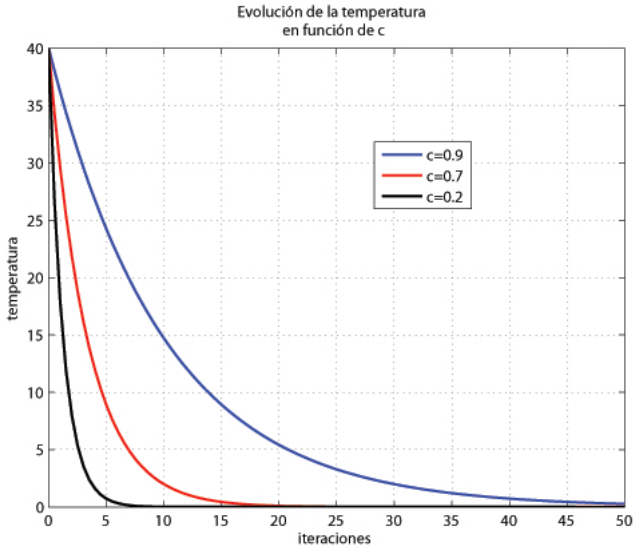

Figura 6.24. Evolución de la temperatura. Se muestra la evolución de la temperatura para distintos valores del parámetro c.

convergencia del mismo y si esa convergencia es al mínimo global óptimo $\hat{\mathbf{x}}$, o por el contrario, el algoritmo queda anclado en la cuenca de atracción de mínimos locales.

Fijando un valor de $T_0 = 40$, se simulará el algoritmo para distintos valores de la constante c. Como función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ se seguirá utilizando una normal bidimensional, y se tomarán 30000 iteraciones del algoritmo. Al finalizar el algoritmo se analizará el mínimo al que ha conver-

gido el algoritmo en cuanto a posición y profundidad. Llegados a este punto se tuvo un problema, ya que la temperatura inicial T_0 escogida funcionaba bien con la ley de enfriamiento anterior, pero no con la actual. Se presenta aquí el mayor problema del algoritmo Simulated Annealing, que se basa en la dificultad de establecer los parámetros de optimización, ya que generalmente todos son dependientes entre sí y de la función de coste a optimizar.

Se sospechaba que el algoritmo funcionaba mal porque la ley de enfriamiento era demasiado abrupta en sus primeras iteraciones. Fue por ello que se probó aumentando dicha temperatura inicial. Con un valor de $T_0 = 250$, el algoritmo funcionaba bien para intervalo muy amplio de valores de la constante c. La prueba de lo expuesto se muestra en la Figura 6.25.

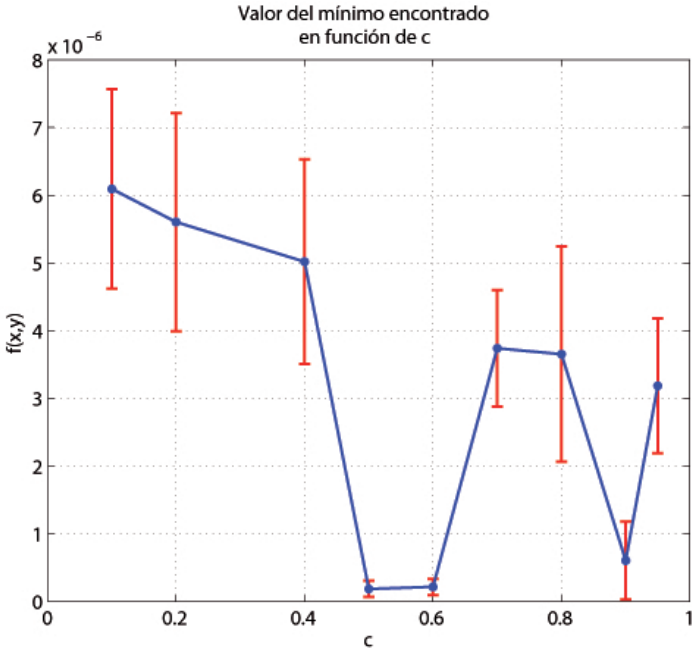

Figura 6.25. Análisis del algoritmo. Valor medio de la función de coste en función del valor de c.

Se observa que el intervalo que mejores prestaciones presenta es $[0.45, 0.65]$. Si se compara, en esta ocasión, el valor medio de la función de coste en los con los valores obtenidos en la Figura 6.4 (mismo número de iteraciones N, pero distinta ley de enfriamiento) se ve claramente que utilizando esta nueva ley de enfriamiento la profundidad media de los mínimos es un orden de magnitud menor. Esto parece indicar, que esta nueva "*cooling schedule*" es más apropiada que la utilizada anteriormente. A continuación, se hará una prueba referente a la velocidad de convergencia con esta nueva ley de enfriamiento. El requisito impuesto para parar el algoritmo es que se consiguiera un valor de la función de coste menor que $1e-6$. Los resultados pueden

verse en la Figura 6.26.

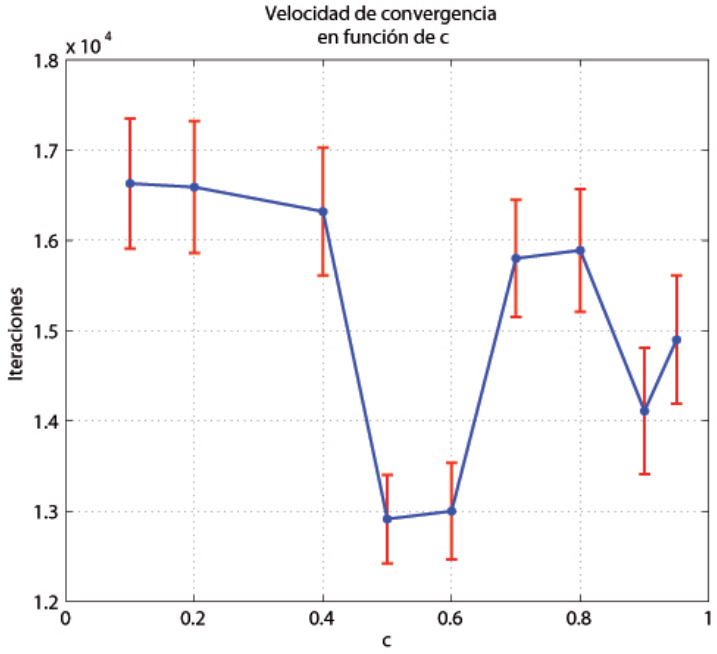

Figura 6.26. Se muestra la velocidad de convergencia en iteraciones en función del valor de la constante de enfriamiento c.

Se aprecia claramente que con esta nueva ley de enfriamiento para un valor de $c = 0.5$ se necesitan menos iteraciones que las que se necesitaban con la ley de enfriamiento anterior. Alrededor de un 25 % menos. Utilizaremos pues la implementación de Simulated Annealing para una ley de enfriamiento ($T_t = T_0 \exp(-(1-c)t)$), $c = 0.5$ y $T_0 = 250$ como modelo de referencia para comparar esta técnica con Acelerated Random Search, implementada en la siguiente sección.

6.2. Implementación de Acelerated Random Search

En esta sección se implementará el algoritmo Acelerated Random Search en su versión más general, vista en el Capítulo 5. Se recuerda que los pasos seguidos por esta técnica son los siguientes:

1. Inicializar $\mathbf{x}_0 \in S$, el número de iteraciones $t = 0$ y la varianza inicial σ_0^2. Definir un valor máximo a para la varianza, un umbral de precisión ρ y una constante de contracción de la varianza c.

2. Generar un punto candidato $\mathbf{x}' \sim \pi(\mathbf{x}_t|\mathbf{x}_{t-1})$.

3. Si $f(\mathbf{x}') \leq f(\mathbf{x}_{t-1}) \rightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$. Reducir $\sigma_t^2 \rightarrow \sigma_{t-1}^2/c$.

- Si $\sigma_i^2 < \rho \rightarrow \sigma_t^2 = a$. Ir al paso 5.

4. Si $f(\mathbf{x}') > f(\mathbf{x}_{t-1}) \rightarrow \mathbf{x}_t = \mathbf{x}_{t-1}$.

   - Aumentar $\sigma_t^2 = a$.

5. Actualizar $t = t + 1$.

6. En caso de cumplirse el criterio de parada terminar. En caso contrario volver a 2.

Se procederá de forma similar a la utilizada cuando se estudió Simulated Annealing. La condición de parada será un número fijo de 30000 iteraciones tal y como se hizo anteriormente de manera que la comparación de resultados sea lo más consistente posible. Los parámetros modificables del algoritmo son la forma de la función tentativa (uniforme, gaussiana, etc), la varianza inicial σ_0^2 de dicha función tentativa, la constante c de reducción de la varianza y la constante de precisión ρ. A continuación se muestran las pruebas derivadas de la variación de estos parámetros con sus respectivos resultados.

6.2.1. Primera prueba sobre Acelerated Random Search

En esta primera prueba se estudiará el comportamiento de Acelerated Random Search para distintos anchuras de la función tentativa. Como función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$ se utilizará una distribución uniforme en el intervalo $\mathbf{x} \in [\mathbf{x}_{t-1} - \frac{\sigma_0}{2}\ , \mathbf{x}_{t-1} + \frac{\sigma_0}{2}]$. Se seguirá utilizando σ (que antes se utilizaba como desviación típica de la densidad tentativa gaussiana) para hacer referencia a la anchura de la densidad uniforme. Se utilizarán también unos valores de las constantes $c = 1.3$ (constante de reducción de la varianza) y $\rho = 0.5$ (umbral o constante de precisión). Se tomaron valores de la distancia al mínimo global óptimo en función de la anchura inicial σ_0 obteniendo los resultados mostrados en la Figura 6.27.

Se puede apreciar como el algoritmo funciona peor para valores de la anchura de la función tentativa muy pequeños o muy altos. Cuando σ_0 es pequeño (menor que 3) la exploración del espacio está muy restringida y la probabilidad de saltos largos es muy pequeña. Además, cuando el algoritmo encuentra un punto peor que el anterior, y abre de nuevo la anchura, esta apertura no es muy grande, lo cual en ocasiones no permite al método salir de la cuenca de atracción del mínimo local, obteniendo resultados insatisfactorios. Por otro lado, cuando σ_0 es muy alta (mayor que 30), se explora adecuadamente todo el espacio pero al ser la apertura de la anchura tan elevada, la técnica se hace más lenta y se obtienen peores medidas. Se puede ver un zoom del intervalo óptimo (respecto a σ_0) en la Figura 6.28.

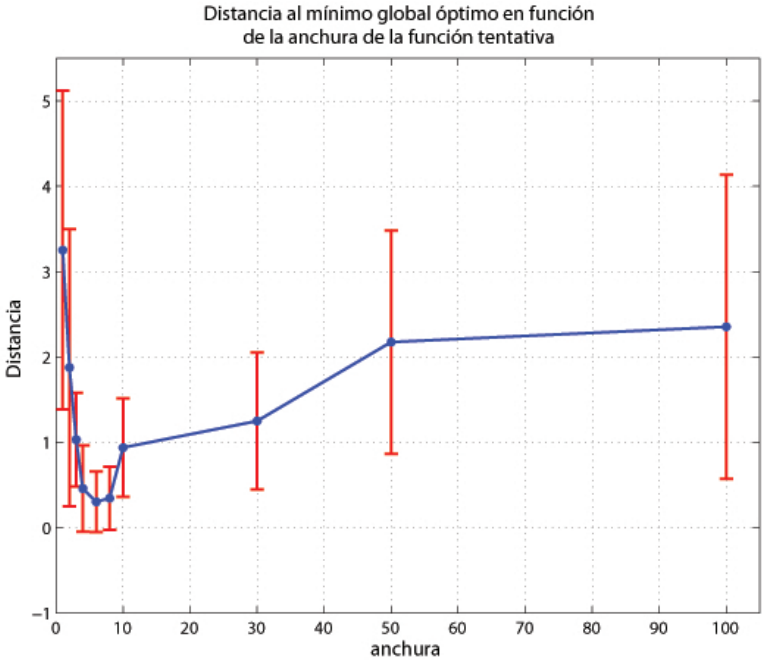

Figura 6.27. Distancia media al mínimo global en función de la anchura inicial σ_0 de la función tentativa.

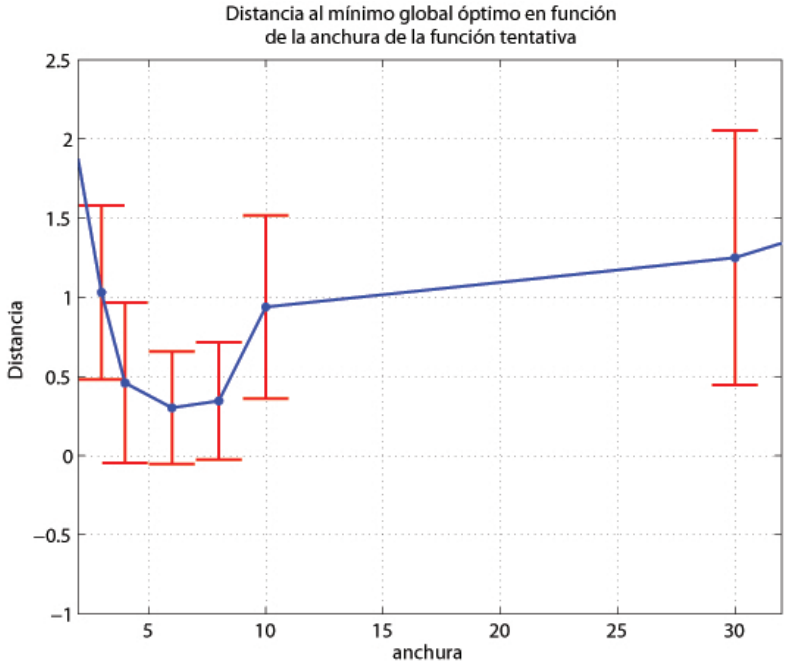

Figura 6.28. Distancia media al mínimo global en función de σ_0 en el intervalo óptimo.

Se tomará a partir de este momento un valor de $\sigma_0 = 6$. Se observa que utilizando este valor la distancia media al mínimo global óptimo se sitúa por debajo de 0.5 con una variabilidad en los resultados también muy baja.

6.2.2. Segunda prueba sobre Acelerated Random Search

Para esta segunda prueba se eligió un valor de la anchura inicial $\sigma_0 = 6$ y $\rho = 0{,}5$ (constante de precisión). Se hizo variar c (constante de reducción de la varianza) entre los valores de $c = 1.5$ y $c = 3$, calculando para cada caso la distancia media al mínimo global óptimo. Los resultados obtenidos se muestran en la Figura 6.29.

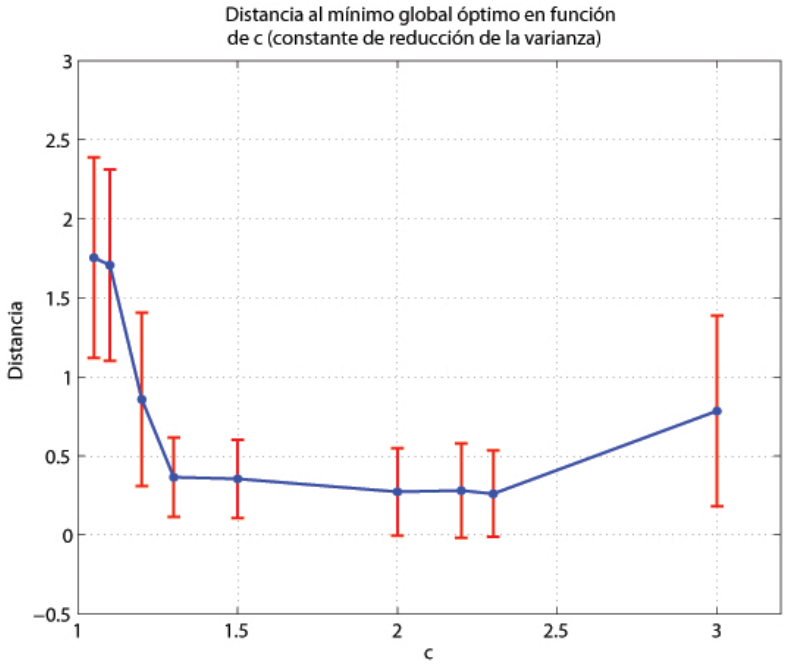

Figura 6.29. Distancia al mínimo global en función del valor de la constante c de reducción de la anchura.

Se observa un intervalo óptimo muy amplio para los valores de c situado en $[1.3, 2.5]$. En concreto, para $c = 2.2$ se obtiene una distancia media al mínimo global de 0.25 con un dispersión en la medida de 0.276.

Para valores de la constante c pequeños, la anchura de la función tentativa disminuye muy lentamente. Esto contribuye a una exploración espacial más exhaustiva pero repercute en la velocidad del algoritmo. Es por ello que para valores de $c < 1.2$ la distancia media de la solución encontrada al mínimo global es elevada. De este modo, se podría pensar que aumentando mucho el valor de esta constante, la contracción de la anchura se haría muy abrupta y la técnica convergería

de forma muy rápida y muy precisa al mínimo global. Sin embargo, se puede ver en la Figura 6.29 que para valores de $c > 2.3$ los resultados son peores que en el intervalo óptimo. Esto es porque el algoritmo llega a un mínimo local de la función de coste, y reduce mucho su varianza hasta el valor límite ρ. Llegado a este punto, vuelve a aumentar la varianza, pudiéndose dar dos opciones en este momento:

- que el método muestree un punto cercano al mínimo global y se de cuenta de que está en un mínimo local.
- que vuelva a cerrar la varianza concentrándose de nuevo en el mínimo local (desperdiciándose tiempo e iteraciones).

Es por eso que se necesita guardar un compromiso entre exploración completa (cuanto más exhaustiva es la exploración más iteraciones necesita) y precisión. Diseñada esta prueba se eligió un valor óptimo de $c = 2.2$.

6.2.3. Tercera prueba sobre Acelerated Random Search

En esta prueba se fijaron los valores de $\sigma_0 = 6$ y $c = 2.2$ y se variará el valor de ρ (umbral de precisión). La constante de precisión indica cuando aumentar la varianza (una vez ésta se ha reducido hasta un valor muy bajo). En concreto, se probó con valores de ρ en el intervalo $[0, 6]$, y se calculó la distancia media al mínimo global óptimo obteniendo los resultados mostrados en la Figura 6.30.

El valor que tome la constante de precisión ρ es vital en el funcionamiento del algoritmo. Se podría pensar que lo más lógico sería elegir un valor de ρ muy cercano a 0. Sin embargo vemos en la Figura 6.30 que los valores más pequeños no son los más adecuados. Si el método se encuentra en la cuenca de atracción del mínimo global, lo mejor sería tener un valor de ρ lo más pequeño posible, pero si estamos en la cuenca de otro mínimo, las iteraciones que se malgastan hasta llegar a un valor muy pequeño de σ, se traducirán en tiempo perdido por la técnica. Esto es así, porque al abrir la anchura seguramente encontremos otro mínimo mejor (porque estamos analizando un mínimo local). Dicho esto, tampoco será productivo tener un valor de ρ muy alto. Un valor muy alto de ρ valdrá para no malgastar iteraciones en los mínimo locales, pero cuando realmente se llegue cerca del mínimo global óptimo, al disminuir la anchura de la función tentativa, se llegará muy rápido al límite fijado por el umbral de precisión, produciéndose muchos "reinicios" en la anchura de la función tentativa, lo que también es contraproducente. En la Figura 6.30 se hace un zoom sobre los mejores valores de ρ obtenidos.

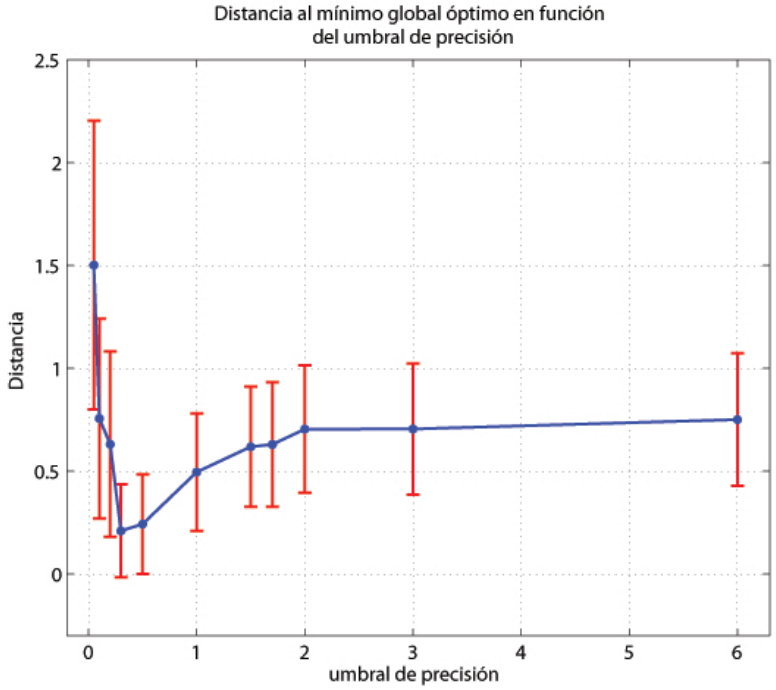

Figura 6.30. Distancia al mínimo global en función del valor del umbral de precisión ρ.

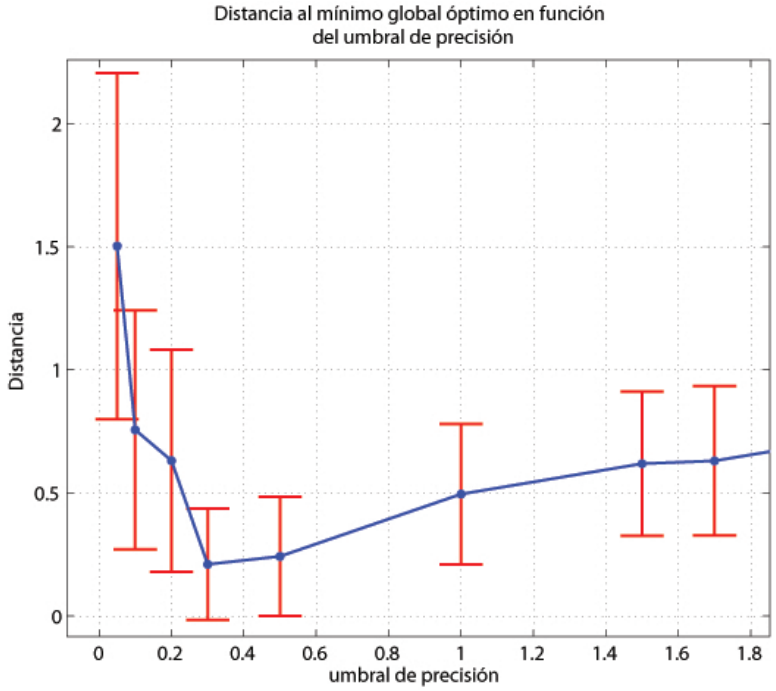

Figura 6.31. Distancia al mínimo global en función del valor de la constante de precisión ρ (ampliación del intervalo óptimo).

Se puede observar que las mejores prestaciones del método se dan para un valor del umbral de precisión $\rho \in [0.2, 0.6]$. En particular, para un valor de $\rho = 0{,}3$, se consigue una distancia media al mínimo global óptimo de 0.22.

Para poder comparar la velocidad de convergencia de Acelerated Random Search con la de Simulated Annealing se realizará una prueba más. En concreto, se pondrá como condición de parada la consecución de un mínimo cuyo valor de la función de coste sea menor o igual que $1e-6$ (tal y como ya se hizo en la Sección 6.1). Los resultados de esta prueba se muestran en la Figura 6.32.

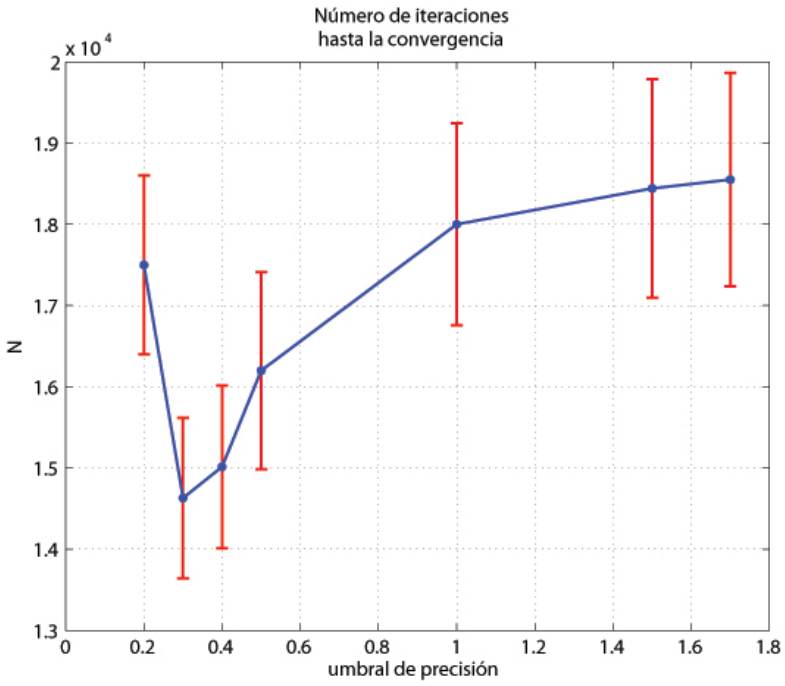

Figura 6.32. Iteraciones necesarias en función del umbral de precisión.

El algoritmo necesita de aproximadamente de media 14630 iteraciones para conseguir el objetivo fijado. Si comparamos este número de iteraciones con el de Simulated Annealing vemos que los resultados son ligeramente peores. Simulated Annealing necesitaba 13000 iteraciones para lograr un valor de la función objetivo menor que $1e-6$, lo que supone un incremento del 12 % en el número de iteraciones para Acelerated Random Search.

6.2.4. Cuarta prueba sobre Acelerated Random Search

En esta sección manteniendo constantes $c = 2.2$ y $\rho = 0.3$ para un número de iteraciones fijas $N = 30000$, cambiaremos la forma de la función tentativa $\pi(\mathbf{x}_t|\mathbf{x}_{t-1})$. En particular, utilizaremos

una normal $\mathcal{N}(\mathbf{x}_{t-1}, \sigma^2)$ centrada en $\mathbf{x}_{t-1}$ con varianza σ^2. Los resultados obtenidos se muestran en la Figura 6.33.

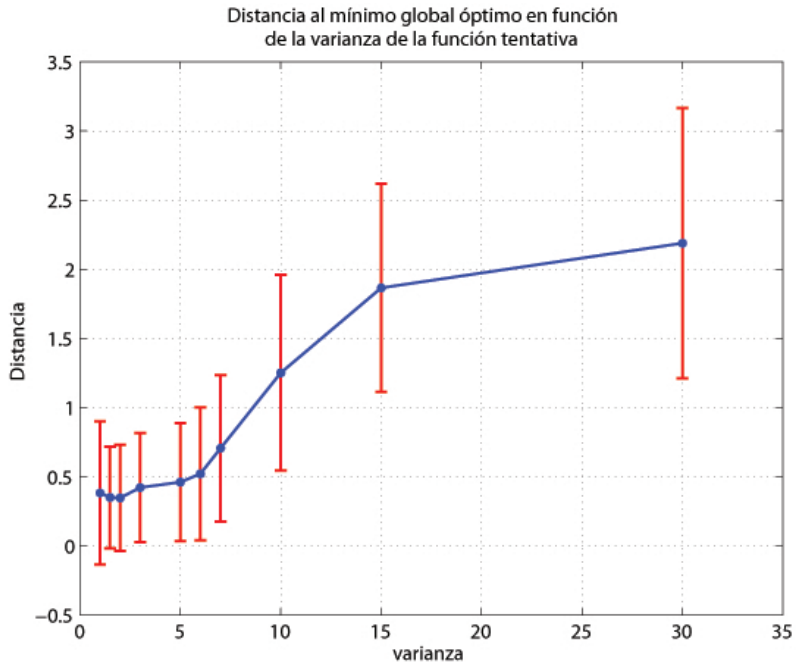

Figura 6.33. Distancia al mínimo global en función del valor de la varianza σ^2 de la función tentativa.

Se puede observar que los valores de la varianza que mejores resultados obtienen son los pertenecientes al intervalo $\sigma^2 \in [0.5, 6]$. Estos resultados empeoran ligeramente respecto a los conseguidos en la sección anterior ya que los valores de la constante c y el umbral de precisión están optimizados para una forma de la función tentativa distinta. Aun así, para un valor de de $\sigma^2 = 2$ la distancia al mínimo global óptimo sigue siendo muy pequeña. En concreto de 0.32.

6.3. Implementación de Multiple Particle Simulated Annealing (MPSA)

En esta prueba se implementará una técnica de Simulated Annealing que maneja varias partículas. En concreto, el método *Multiple Particle Simulated Annealing* visto en la Sección 5.7.1 perteneciente al Capítulo 5 de este proyecto. Se tomó como temperatura inicial $T_0 = 250$ y la ley de enfriamiento vista en la Sección 6.1.6 ($T_t = T_0 \exp(-(1-c)t)$), con un valor de la constante $c = 0.5$ y una función tentativa gaussiana de varianza 1.2. Se realizarán $N = 30000$ iteraciones para $1, 2, 3, 5, 7$ y 10 partículas. En la Figura 6.34 se muestra la distancia media al

mínimo global óptimo en función del número de partículas utilizadas.

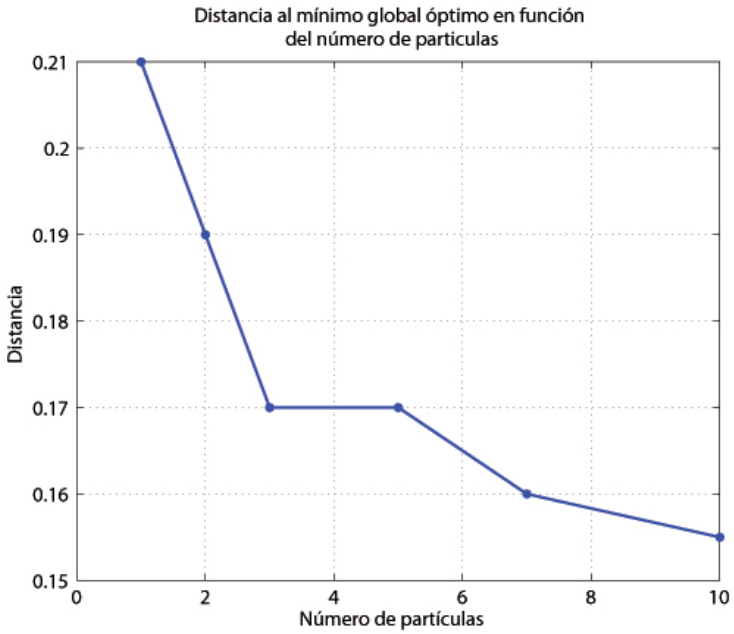

Figura 6.34. Se muestra la distancia al mínimo global en función del número de partículas utilizadas.

La distancia media al mínimo global óptimo no varía mucho respecto a la utilización de una o varías partículas. Esto es así, ya que para el número de iteraciones propuesta, utilizando tan solo una partícula el algoritmo tiene iteraciones (tiempo) para converger adecuadamente. Será más interesante analizar el número de iteraciones que necesita el algoritmo para converger en función del número de partículas. Este resultado se muestra en la Figura 6.35. Como condición de parada se impuso la consecución de un mínimo con valor de la función de coste $f(x, y) \leq 1e - 6$.

Se observa que a medida que aumenta el número de partículas utilizadas disminuyen las iteraciones necesarias para cumplir el requisito de parada. Sin embargo, resulta interesante ver que la disminución en las iteraciones no es lineal con respecto al número de partículas, sino mas bien exponencial. Es evidente que al aumentar el número de partículas en paralelo del algoritmo, aumentan el número de recursos utilizados por la técnica, y es por ello, que se debe establecer un compromiso entre número de partículas y recursos utilizados.

Para mostrar la potencia del método se muestra la evolución (sobre las curvas de nivel del logaritmo de la función de coste) del algoritmo *Multiple Particle Simulated Annealing* en la Figura 6.36 (para tres partículas). En esta ocasión se fijaron como puntos iniciales de comienzo de cada una de las partículas puntos bastante alejados al mínimo global óptimo. También se muestra la evolución para cuatro partículas en la Figura 6.37.

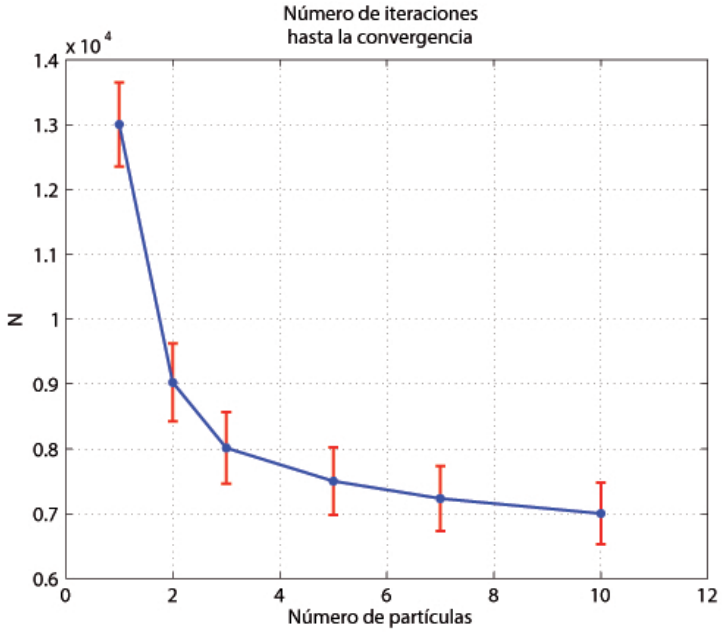

Figura 6.35. Se muestra la distancia media al mínimo global en función del número de partículas utilizadas.

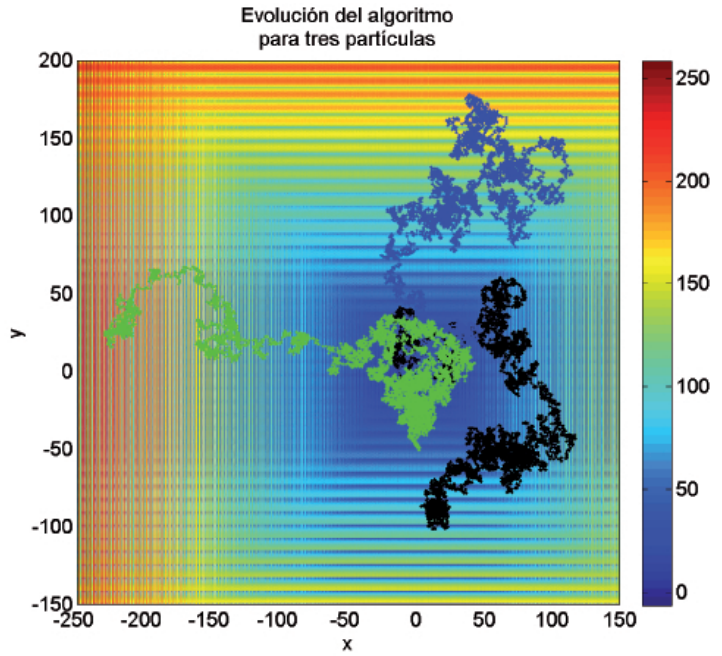

Figura 6.36. Evolución de MPSA para tres partículas.

Se aprecia claramente como el algoritmo explora todo el espacio $\mathcal{S}$, para iteración a iteración, ir encaminandose hacia el mínimo global óptimo. Una vez encontrado, concentra la búsqueda en una región del espacio más pequeña.

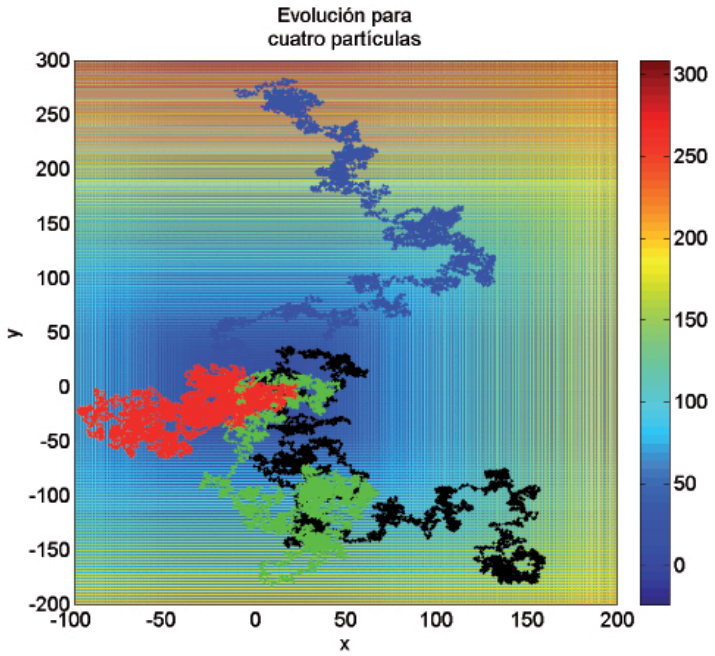

Figura 6.37. Evolución de MPSA para cuatro partículas.

6.4. Comparación de resultados

En este Capítulo se ha probado como dos filosofías de optimización distintas, como son Simulated Annealing y Acelerated Random Search, trabajan en un problema de optimización real de una función de coste compleja.

Con Simulated Annealing se ha visto la importancia de elegir los parámetros cuidadosamente. Todos los parámetros se encuentran en mayor o menor medida interrelacionados entre sí, y ligeros cambios en la función tentativa o la ley de enfriamiento pueden estropear los resultados. Y dada la naturaleza estocástica de los métodos fue de vital importancia la repetición de los experimentos un número elevado de veces, de manera que se obtuvieran resultados consistentes sobre el funcionamiento del algoritmo. Un número insuficiente de experimentos conduce a resultados *fantasma* que enmascaran el funcionamiento real de la técnica y aportan una variabilidad exagerada.

Es muy importante resaltar también la importancia de establecer unas condiciones de parada y de reinicio eficientes. En cuanto a la condición de parada tuvimos que estudiar con detenimiento cuando el algoritmo ha convergido a un resultado aceptable y debemos pararlo. No era nuestra intención ni parar el algoritmo demasiado pronto (malos resultados), ni dejarlo iterar eternamente (desperdicion de recursos computacionales). Fue también muy interesante abordar el problema de los reinicios. Los reinicios debían aplicarse para que el algoritmo no quedará anclado en mínimos

locales, y debían ejecutarse de forma inteligente de forma que no se malgastaran recursos.

El proceso de optimización seguido con Acelerated Random Search fue muy similar al realizado con Simulated Annealing. También se vio con este método la importancia de elegir bien los parámetros. Una vez optimizados sus parámetros se pudo ver que los resultados obtenidos con una técnica y otra son muy similares. Ambos métodos funcionan satisfactoriamente para optimizar funciones de coste complejas en un tiempo asumible y con un grado de precisión elevado.

Por último se dio un paso más y se implementó el algoritmo Simulated Annealing para varias partículas ó *Multiple Particle Simulated Annealing*. Este método nos permitió fusionar las técnicas de una partículas con las basadas en poblaciones. La existencia de varias partículas trabajando en paralelo nos permite explorar exhaustivamente el espacio de búsqueda, y el intercambio de información nos permite incrementar la velocidad de convergencia. Sin embargo, debemos ser conscientes de que trabajar con muchas partículas en paralelo intercambiando información consume muchos recursos y tiempo. No se puede incrementar desmesuradamente el número de partículas y debemos guardar un cierto compromiso entre recursos utilizados y velocidad de convergencia.

Capítulo 7

Conclusiones y líneas futuras de investigación

Una primera conclusión que se ha intentado mostrar en este estudio es que los conceptos de muestreo y optimización se encuentran estrechamente relacionados. De hecho, se puede decir que *el muestreo es un problema más fácil que la optimización, se puede considerar una primera fase* [Liang et al., 2010]. De forma general, un problema de optimización se puede reducir, casi por completo a un problema de generación de números aleatorios con una adecuada densidad de probabilidad (por lo menos conceptualmente).

De hecho, se puede observar que cualquier función de coste $f(\mathbf{x})$ se puede asociar a una densidad de probabilidad sin más que realizar una transformación del tipo $q_T(\mathbf{x}) = H(f(\mathbf{x})/T)$. En concreto, los valores $\hat{\mathbf{x}}$ que minimizan la función objetivo $f(\mathbf{x})$ son los mismos que maximizan la función $q_T(\mathbf{x})$, es decir, sus modas. De este modo, muestrear una densidad $q_T(\mathbf{x})$ significa generar valores aleatorios con una frecuencia de aparición proporcional al área por debajo de $q_T(\mathbf{x})$. Si somos capaces de muestrear $q_T(\mathbf{x})$, se obtendrán muestras cercanas a las modas de dicha función sobre todo cercanas al máximo global, o lo que es lo mismo, cercanas al mínimo global de la función de coste $f(\mathbf{x})$. El parámetro de escala T actúa a modo de varianza, de modo que al variarlo la densidad $q_T(\mathbf{x})$ mantiene sus modas pero varía su dispersión. Cuanto más pequeño se haga el valor de T, más estrecha se hará la función $q_T(\mathbf{x})$ en torno a su máximo global (mínimo global de la función de coste). Se concluye pues, que en el caso límite para $T = 0$ la función $q_T(\mathbf{x})$ se convierte en un tren de deltas y la convergencia del método de optimización será prácticamente instantánea. En este caso, se necesitaría muestrear sólo un numero finito de puntos para lograr el mínimo global de $f(\mathbf{x})$ (máximo global de $q_T(\mathbf{x})$). En este sentido, un problema de optimización se reduce a un problema de muestreo de una densidad.

Otra conclusión importante que se puede sacar de este proyecto es que los algoritmos metaheu-

rísticos de optimización son métodos potentes que pueden ser utilizados cuando otras técnicas no obtienen resultados satisfactorios. Su potencia reside en que no imponen prácticamente casi ninguna restricciones sobre la función de coste y no necesitan conocer sus propiedades analíticas. Además, se trata de algoritmos relativamente fáciles de programar, que una vez implementados (por ejemplo en un FPGA) pueden ser aplicados a la resolución de problemas de todo tipo. Es decir, no se trata de algoritmos específicos que trabajan bien para una clase determinada de problemas, sino que son algoritmos generales, que se pueden ser modificados para atender a problemas específicos.

Sin embargo, el gran problema de los métodos estocásticos de optimización radica en la definición de los parámetros que intervienen. Esto en muchos casos tienen que ser elegidos caso por caso, optimizados para la aplicación especifica.

Las técnicas estocásticas que nos han parecidos más interesantes, desde el punto de vista teórico y practico, son Accelerated Random Search y Simulated Annealing. El trabajo con estos dos métodos es el trabajo con dos filosofías de optimización distintas. Representan dos maneras de entender la convergencia al mínimo global óptimo. Utilizando el lenguaje del muestreo aleatorio podríamos decir la filosofía de Acelerated Random Search consiste en variar de forma inteligente la varianza de la función tentativa. A medida que nos vamos acercando al posible mínimo global optimo dicha varianza de la función tentativa se va haciendo más pequeña, posibilitando que se generen muchos puntos en una región pequeña del espacio $\mathcal{S}$ cercana al posible mínimo global. Por otro lado, el Simulated Annealing modifica una densidad objetivo $p_o(\mathbf{x})$ asociada a la función de coste $f(\mathbf{x})$, lo cual repercute en la función de coste $f(\mathbf{x})$ que queremos optimizar y nos guía hacia el mínimo global.

Una posible línea de investigación muy interesante sería tratar de combinar estas dos filosofías (del ARS y del SA), es decir, estudiar como cambiar simultáneamente (y adecuadamente) la densidad tentativa y la densidad objetivo manteniendo la convergencia al mínimo global. En este caso, añadiríamos otra componente variable al algoritmo (otro bloque): por ejemplo, como variar conjuntamente la varianza de la densidad tentativa y la densidad objetivo (el parámetro temperatura T del SA). Además, se podrían idear estrategias con múltiples "partículas" (como el caso de *Multiple Particle Simulated Annealing*) con distintas funciones tentativa y objetivo. En caso de fusionar las dos filosofías de optimización para una técnica de varias partículas, sería inevitable investigar sobre las formas de comunicación óptimas de las partículas o la forma de finalizar el método. Otra vertiente interesante del problema sería intentar establecer una serie de reglas generales que ayudaran a futuros implementadores en la definición de los parámetros

iniciales, ya que hoy en día, la técnica más utilizada para fijar estos valores es el métodos de ensayo/error.

Por último, creemos importante evidenciar que una aplicación interesante de estos algoritmos de exploración estocástica (donde se puede notar la versatilidad y la potencia de estos últimos) es su utilización en problemas de *clustering* (agrupamiento).

Capítulo 8

Planificación y Presupuesto

En este Capítulo se presentan justificados los costes globales de la realización de este Proyecto Fin de Carrera, así como la planificación en el tiempo del mismo. Las horas totales ocupadas por cada parte del proyecto se muestran en el Cuadro 8.1, mientras que su distribución en el tiempo se puede ver en el diagrama de la Figura 8.1.

Del Cuadro 8.1 se desprende que el número total de horas asignadas al proyecto ha sido de 956, de las cuales, aproximadamente el 5 % han sido compartidas con el tutor del proyecto. Debido a circunstancias variadas, no todos los días se trabajó el mismo número de horas en el proyecto, por lo que el diagrama de gantt no debe ser evaluado de forma exhaustiva.

En el Cuadro 8.2 se recoge el coste total del proyecto desglosado en gastos de material y de personal. Los materiales utilizados han sido escasos por lo que el 75.21 % del presupuesto total del proyecto es ocasionado por los gastos de personal. El sueldo utilizado para calcular los gastos de personal es un gasto aproximado, calculado a partir de las retribuciones de un becario estandard en la universidad Carlos III de Madrid que es de aproximadamente cuatro euros y medio por hora. En la Figura 8.2 se muestra la factura del proyecto fin de carrera presentada.

Fase	Descripción	Núm. de horas
1	**Documentación**	300 **horas**
2	**Desarrollo del software**	250 **horas**
2.1	Ejemplos	50 horas
2.2	Simulated Annealing	75 horas
2.3	Acelerated Random Search	60 horas
2.4	Multiple part Simulated Annealing	60 horas
3	**Redacción del proyecto**	380 **horas**
3.1	Redacción capítulo 3	70 horas
3.2	Redacción capítulo 4	100 horas
3.3	Redacción capítulo 5	90 horas
3.4	Redacción capítulo 6	75 horas
3.5	Resto de capítulos	50 horas
4	**Revisión del proyecto**	20 **horas**
5	**Redacción de la presentación**	6 **horas**
TOTAL		956 **horas**

Cuadro 8.1. Horas asignadas al proyecto. Distintas fases del proyecto desglosadas en sus respectivas subfases junto con el número de horas dedicadas a cada una.

Descripción	Coste
Gastos materiales	**1500 €**
Ordenador de gama media	1300 €
Material de oficina	200 €
Gastos de personal	**4552 €**
Salario	4302 €
Desplazamientos	250 €
Gasto Total	**6052 €**

Cuadro 8.2. Costes imputables del proyecto. Se dividen los costes del proyecto en costes de personal y costes materiales.

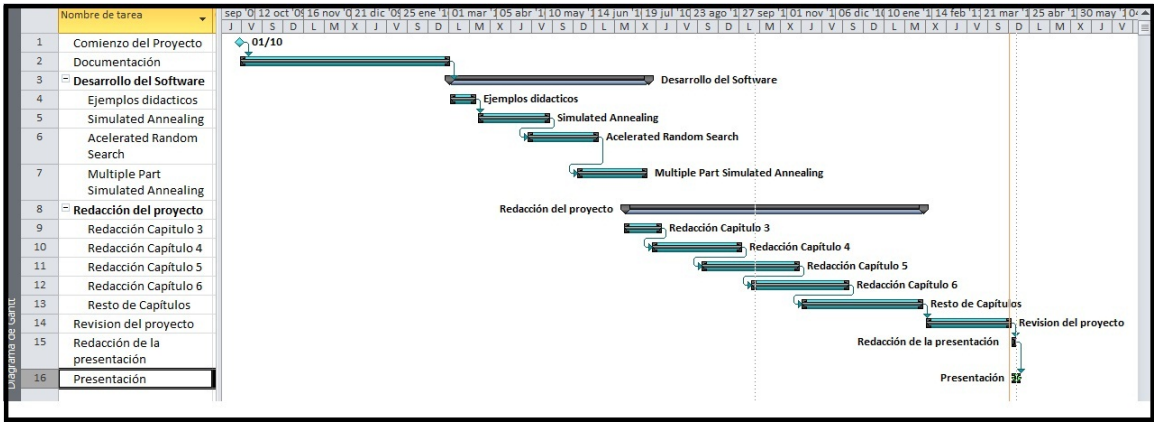

Figura 8.1. Diagrama de Gantt del PFC. Se muestra el proyecto fin de carrera dividido en tareas y la planificación en el tiempo de cada una de ellas.

PRESUPUESTO DEL PROYECTO FIN DE CARRERA

Optimización estocástica mediante métodos de Monte Carlo

PRESUPUESTO DE GASTOS 22/12/2010

Nombre de la compañía	Universidad Carlos III de Madrid
Departamento	Departamento de teoría de señal y comunicaciones
Autor	Alberto Serrano Cádiz

Gastos de Material	Coste
Ordenador de gama media	1.300 €
Material de oficina	200 €

Gastos de Personal	Coste
Salario del proyectista	4.302 €
Desplazamientos	250 €

Totales	Coste
Gastos de material	1.500 €
Gastos de personal	4.552 €
Gasto total sin I.V.A	6.052 €
I.V.A (18%)	1.089,36 €
Gasto total	**7.141,36 €**

Figura 8.2. Factura del PFC. Se muestra la factura del proyecto presentada, donde se desglosan tanto los costes de material como de personal.

Bibliografía

[Aarts y Korst, 1989] E. Aarts y J. Korst, *Simulated Annealing and Boltzmann Machines - a Stochastic Approach to Combinatorial Optimization and Neural Computers*, Wiley, 1989.

[Anderson, 1986] L. H. Anderson, "Metropolis, Monte Carlo, and the MANIAC", *Los Alamos Science*, (14):96–108, 1986.

[Andrieu et al., 2001] C. Andrieu, A. Doucet y E. Punskaya, *Sequential Monte Carlo methods in practice*, cap. Sequential Monte Carlo methods for optimal filtering, Springer-Verlag, 2001.

[Appel y Radulovic, 2000] M. J. Appel y D. Radulovic, "Acelerated Random Search", en *Proceedings of the 16th IMACS World Congress 2000 on Scientific Computing*, Laussane, Suiza, 2000.

[Asvanund y Smith, 2004] A. Asvanund y M. Smith, "Interest Based Self Organizing Peer to Peer Networks: A Club Economics Approach", *Informe Técnico*, H. John Heinz III School of Public Policy and Management (Carnegie Mellon University), September 2004.

[Baeza y Ribeiro, 1999] R. Baeza y B. Ribeiro, *Modern Information Retrieval*, Addison Wesley, 1999.

[Barker, 1965] A. A. Barker, "Monte Carlo Calculations of the radial distribution functions for a proton-electron plasma", *Journal of Physics*, (18):119–133, 1965.

[Baumert et al., 2009] S. Baumert, A. Ghate, S. Kiatsupaibul, R. L. Smith y Z. B. Zabinsky, "Discrete Hit-and-Run for Generating Multivariate Distributions over Arbitrary Finite Subsets of a Lattice", *Operations Research*, 2009.

[Baumgarten, 1999] C. Baumgarten, *Probabilistic Information Retrieval in a Distributed Heterogeneous Environment*, Tesis Doctoral, Fakultät Informatik (Technischen Universität Drenden), February 1999.

[Beichl y Sullivan, 2000] I. Beichl y F. Sullivan, "The Metropolis Algorithm", *Computing in Science and engenieering*, 2(1):65–69, 2000.

[Bélisle, 1992] C. J. P. Bélisle, "Convergence Theorems for a Class of Simulated Annealing Algorithms on $\mathbb{R}^{d}$", *J. Applied Probability*, 29(885-895), 1992.

[Bélisle et al., 1993] C. J. P. Bélisle, H. E. Romeijn y R. L. Smith, "Hit-and-Run Algorithms for Generating Multivariate Distributions", *Mathematics of Operations Research*, (18):255–266, 1993.

[Bohachevsky y Johnson, 1986] I. O. Bohachevsky y M. E. Johnson, "Generalized simulated annealing for function optimization", *Technometrics*, (28):209–217, 1986.

[Brooks y Verdini, 1988] D. G. Brooks y W. A. Verdini, "Computacional experience with generalized simulated annealing over continouos variables", *American Journal of Mathematical and management sciencies*, (8):425–449, 1988.

[Brooks, 1958] S. H. Brooks, "A discussion of random methods for seeking maxima", *Operations Research*, (6):244–251, 1958.

[Brualdi, 1991] R. A. Brualdi, *Combinatorial Matrix theory*, University of Cambridge, 1991.

[Bulger y Wood, 1998] D. W. Bulger y G. R. Wood, "Hesitant Adaptive Search for Global Optimization", *Mathematical Programming*, (81):89–102, 1998.

[Byers y Considine, 2004] J. Byers y J. Considine, "Informed Content Delivery across Adaptive Overlay Networks", *IEEE ACM Transactions on Networking*, 12(5):767–780, Octubre 2004.

[Callan, 2000] J. Callan, *Advances in Information Retrieval*, cap. Distributed Information Retrieval, págs. 127–150, Kluwer Academic Publishers, 2000.

[Chenney y Forsyth, 2000] S. Chenney y D. A. Forsyth, "Sampling plausible solutions to multi-body constraint problems", *Siggraph*, págs. 219–228, 2000.

[Cole, 2009] E. A. Cole, *Mathematical and numerical modelling of heterostructure semiconductor devices*, cap. Genetic algorithms and Simulated Annealing, págs. 339–376, Springer, 2009.

[Colorni y Manniezzo, 1992] A. Colorni y V. Manniezzo, "Distributed optimization by ant colonies", en *Proceedings of the First European Conference on Artificial Life*, págs. 134–142, Cambridge, 1992.

[Corana et al., 1987] A. Corana, M. Marchesi y S. Ridella, "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm", *ACM Transactions on Mathematical Software*, (13):262–280, 1987.

[Damien et al., 1999] P. Damien, J. Wakefield y S. Walker, "Gibbs Sampling for Bayesian non-conjugate and hierachical models by auxiliary variables", *Journal of the Royal Statistical Society*, (61-2):331–344, 1999.

[De Jong et al., 1995] K. A. De Jong, W. M. Spears y D. F. Gordon, *Foundations of Genetic Algorithms 3*, cap. Using Markov Chains to Analyze GAFOs, págs. 115–137, D. Whitley, and M. Vose, Morgan Kaufmann, 1995.

[Dekkers y Aarts, 1996] A. Dekkers y E. Aarts, "Global optimization and simulated annealing", *Mathematical Programming*, (50):367–393, 1996.

[Del Moral y Miclo, 1999] P. Del Moral y L. Miclo, "On the convergence and applications of generalized simulated annealing", *SIAM Journal of Control and Optimization*, 4(37):1222–1250, 1999.

[Dempster et al., 1977] A. P. Dempster, G. Laird y D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[Dorigo y Stutzle, 2004] M. Dorigo y T. Stutzle, "Ant colony optimization", *MIT Press*, 2004.

[Dueck y Scheuer, 1990] G. Dueck y T. Scheuer, "Threshold accepting: A general purpouse algorith appearing superior to simulated annealing", *Journal of Computational Physics*, 90:161–175, 1990.

[Dyer et al., 1991] M. Dyer, A. Frieze y R. Kannan, "A random polynomial time algorithm for aproximating the volume of convex bodies", *Journal of the ACM*, (38):1–17, 1991.

[Dyer y Frieze, 1991] M. E. Dyer y A. M. Frieze, "Computing the Volume of Convex Bodies: A Case Where Randomness Provably Helps", en *Proceedings of Symposia in Applied Mathematics*, págs. 123–169, 1991.

[Eberhart y Kennedy, 1995] R. Eberhart y J. Kennedy, "A new optimizer using particle swarm theory", en *Proceedings of the First International Symposium on Micromachine and Human Science*, págs. 39–43, Nagoya(Japan), 1995.

[Evans, 2005] M. Evans, *Probabilidad y estadística*, Reverté, 2005.

[Fan et al., 2009] J. Fan, X. Lin y J. S. Liu, *New developments in biostatistics and bioinformatics*, Higher education Press, 2009.

[Gelfand y Smith, 1990] A. E. Gelfand y A. F. M. Smith, "Sampling-based approaches to calculating marginal densities", *Journal of the American Statistical Association*, (85:410):398–409, 1990.

[Geman y Geman, 1984] S. Geman y D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images", *Journal of the American Statistical Association*, (6:6):721–741, 1984.

[Geweke, 1989] J. Geweke, "Bayesian Inference in econometrics models using Monte Carlo integration", *Econometrica*, 24:1317–1399, 1989.

[Geyer, 1996] C. J. Geyer, *Markov Chain Monte Carlo In Practice*, cap. Estimation and optimization of functions, págs. 241–258, Chapman && Hall, 1996.

[Geyer y Moller, 1993] C. J. Geyer y J. Moller, "Simulation procedures and likelihood inference for spatial point processes", *Informe Técnico*, University of Aarhus, 1993.

[Glover y Kochenberger, 2003] F. Glover y Kochenberger, *Handbook of Metaheuristics*, cap. International series in operations research an management science, p. 57, Kluwer Academic Publishers, 2003.

[Glover y Laguna, 1997] F. Glover y M. Laguna, *Tabu Search*, Kluwer Academic Publishers, 1997.

[Gordon et al., 1993] N. J. Gordon, D. J. Salmond y A. F. M. Smith, "Novel approach to nonlinear and nonGaussian Bayesian state estimation", en *IEE Proceedings*, págs. 107–113, 1993.

[Hamma et al., 1993] B. Hamma, S. Viitanen y A. Torn, "Paralellel Continuous Simulated Annealing for Global Optimization", en *NATO. Advanced Study Institute. Algorithms for Continuous Optimization: The state of Art*, 1993.

[Hansen y Ribeiro, 2001] P. Hansen y C. C. Ribeiro, *Essays and Surveys on metaheuristics*, Kluwer Academic Publishers, 2001.

[Hastings, 1970] W. K. Hastings, "Monte Carlo Sampling Methods using Markov Chains and their applications", *Biometrika*, 57:97–109, 1970.

[Hayes, 1996] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, Willey, 1996.

[Holland, 1975] J. H. Holland, "Adaptation in Natural and Artificial Systems", *University of Michigan Press*, 90, 1975.

[Horst y Pardalos, 1995] R. Horst y P. M. Pardalos, *Handbook of Global Optimization*, Kluwer Academic Publishers, 1995.

[Ingber, 1989] L. Ingber, "Very fast simulated re-annealing", *Mathematical and computer modelling*, (12):29–57, 1989.

[Ingber, 1993] L. Ingber, "Simulated annealing: Practice versus theory", *Mathematical and computer modelling*, 18(11):29–57, 1993.

[Ingber, 1996] L. Ingber, "Adaptive simulated annealing: lessons learned", *Control and Cybernetics*, 25(1):33–54, 1996.

[Isard y Blake, 1996] M. Isard y A. Blake, "Contour tracking by stochastic propagation of conditional density", en *European Conference on Computer Vision*, págs. 343–356, Cambridge, UK, 1996.

[Jones y Forbes, 1995] A. E. W. Jones y G. W. Forbes, "An adaptive simulated annealing algorithm for global optimization over continuous variables", *Journal of global optimization*, (6):1–37, 1995.

[Kanazawa et al., 1995] K. Kanazawa, K. Koller y S. Russel, "Stochastic Simulation algorithms for dynamic probabilistic networks", en *Proceedings of the eleventh Conference on Uncertainty in Artificial Intelligency*, págs. 346–351, 1995.

[Kaufmann, 1989] M. Kaufmann, *An introduction to raytracing*, Andrew S. Glassner, 1989.

[Kennedy y Gentle, 1980] W. Kennedy y J. Gentle, *Statistical Computing*, Willey, 1980.

[Kilby et al., 1997] P. Kilby, P. Prosser y P. Shaw, "Guided Local Search for the Vehicle Routing Problem", *International Conference of Metaheuristics*, 1997.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C. D. Gelatt y M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, (20):671–680, 13 Mayo 1983.

[Kleinberg y Christos, 2001] J. Kleinberg y H. Christos, "On the Value of Private Information", en *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge*, págs. 249–257, Siena (Italy), Julio de 2001.

[Klos y Kobe, 2001] J. Klos y S. Kobe, *Nonextensive Statistical mechanics and its applications*, cap. Generalized Simulated Annealing Algorithms using Tsallis statistics: Application to J spin glass model, págs. 253–258, Springer Berlin, 2001.

[Koehler, 1997] G. J. Koehler, "New Directions in Genetic Algorithm Theory", *Annals of Operations Research*, (75):49–68, 1997.

[Koehler, 1999] G. J. Koehler, "Computing Simple GA Expected Waiting Times", en *Proceedings of the Genetic and Evolutionary Computation Conference*, 1999.

[Liang et al., 2010] F. Liang, C. Liu y J. Chuanhai, *Advanced Markov chain Monte Carlo methods*, John Wiley and Sons, 2010.

[Liu, 2004] J. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer, 2004.

[Locatelli, 2000] M. Locatelli, *Journal of Optimization: Theory and applications*, cap. Simulated annealing algorithms for continuous global optimization, págs. 1–47, Springer, 2000.

[Lourenço et al., 2002] H. Lourenço, O. Martin y T. Stutzle, *Handbook of Metaheuristics*, cap. Iterated Local Search, Kluwer Academic Publishers, 2002.

[Lovasz, 1999] L. Lovasz, "Hit-and-Run Mixes Fast", *Mathematical Programming*, págs. 443–461, 1999.

[Lovasz y Vempala, 2003] L. Lovasz y S. Vempala, "Hit-and-Run is Fast and Fun", *Informe Técnico*, MSR-TR, 2003.

[Luenberger, 1984] D. Luenberger, *Linear and nonlinear Programming. Second Edition*, Addison-Wesley, 1984.

[Mackay, 2003] D. Mackay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[Marinari y Parisi, 1992] E. Marinari y G. Parisi, "Simulated Tempering: a new Monte Carlo scheme", *Europhysics letters*, 19:451, 1992.

[Maringer y Meyer, 2006] D. Maringer y M. Meyer, "Smooth transition autoregressive models- new approaches to the model selection problem", *Informe Técnico*, University of Essex, 2006.

[Metropolis et al., 1953] N. Metropolis, A. W. Rosenbluth y E. Teller, "Equations of state calculations by fast computing machines", *Chemical physics*, 21(6):1087–1092, Junio 1953.

[Metropolis y Ulam, 1949] N. Metropolis y S. Ulam, "The Monte Carlo method", *Journal of the American Statistical Association*, (44:247):335–341, 1949.

[Mladenovic y Hansen, 1997] N. Mladenovic y P. Hansen, "Variable neighbourhood search", *Computers and operators research*, 34:1097–1100, 1997.

[Mladenovic y Hansen, 2001] N. Mladenovic y P. Hansen, "Variable neighbourhood search: Principles and Applications", *European Journal of Operational Research*, 130:449–467, 2001.

[Molvalioglu et al., 2007] O. Molvalioglu, Z. Zabinsky y W. Kohn, *Models ans algorithms for global optimization*, cap. Multi-particle Simulated Annealing, págs. 215–222, Springer, 2007.

[Moscato, 1989] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms", *Informe Técnico*, Caltech, 1989.

[Mostafa et al., 2001] M. G. Mostafa, M. F. Tolba, T. Gharib y M. A. Megeed, "Medical image segmentation using a wavelet-based multiresolution EM algorithm", *IEEE international Conference on industrial electronics*, págs. 15–20, December 2001.

[Mutseniyeks y Rastrigin, 1964] V. A. Mutseniyeks y L. Rastrigin, "Extremal Control of Continuous Multiparameter Systems by the Method of Random Search", *Engineering Cybernetics*, 1(82-90), 1964.

[Neal, 1999] R. Neal, "Bayesian Learning for Neural Networks", *Springer-Verlag press*, 118, 1999.

[Neal, 1996] R. M. Neal, "Bayesian Learning for Neural Networks", *Computing in Science and engenieering*, 1996.

[Pardalos y Romeijn, 2002] P. M. Pardalos y H. E. Romeijn, *Handbook of Global Optimization*, Kluwer Academic Publishers, 2002.

[Patel et al., 1988] N. Patel, R. L. Smith y Z. B. Zabinsky, "Pure Adaptive Search in MonteCarlo Optimization", *Mathematical Programming*, (43):317–328, 1988.

[Pearl, 1984] J. Pearl, "Evidential reasoning using stochastic simulation", *Artificial Intelligence*, (32):245–257, 1984.

[Peskun, 1973] P. H. Peskun, "Optimum Monte-Carlo Sampling Using Markov Chains", *Biometrika*, 60(3):607–612, 1973.

[Pitaekskii, 2003] L. Pitaekskii, *The Bose Einstein condensation*, Oxford Sciencie publications, 2003.

[Rardin, 1998] R. L. Rardin, *Optimization in Operations Research*, Prentice Hall, 1998.

[Rastrigin, 1960] L. Rastrigin, "Extremal Control by the Method of Random Scanning", *Automation and Remote Control*, 21(891-896), 1960.

[Rastrigin, 1963] L. Rastrigin, "The Convergence of the Random Method in the Extremal Control of a Many-parameter System", *Automation and Remote Control*, 24(1337-1342), 1963.

[Reeves y Rowe, 2003] C. R. Reeves y J. E. Rowe, *Genetic Algorithms. Principles and perspectives*, Kluwer, 2003.

[Resende y Ribeiro, 1997] M. G. Resende y C. C. Ribeiro, "A GRASP for graph planarization", *Networks*, 29:173–189, 1997.

[Ribeiro y Souza, 2002] C. Ribeiro y M. C. Souza, "Variable neighbourhood search for the degree constrained minimum spanning tree problem", *Discret Applied Mathematics*, 118:43–54, 2002.

[Ripley, 1994] B. Ripley, "Neural networks and related methods for classification", *Journal of Royal Statistics Society*, (56):409–4560, 1994.

[Ripley, 1996] B. Ripley, "Pattern Recognition and Neural Networks", *Cambridge University Press*, 4, 1996.

[Robbins y Monro, 1951] H. Robbins y S. Monro, "A stochastic approximation method.", *Mathematical Statistics*, (22):400–407, 1951.

[Robert y Casella, 2008] C. Robert y G. Casella, "A History of Markov Chain Monte Carlo", *Astronomical Journals*, 2008.

[Robert y Casella, 1999] C. P. Robert y G. Casella, *Monte Carlo Statistical Methods*, Springer, 1999.

[Romeijn y Smith, 1993] H. E. Romeijn y R. L. Smith, "Simulated Annealing for Constrained Global Optimization", *Journal of Global Optimization*, 5(101-126), 1993.

[Romeijn y Smith, 1994] H. E. Romeijn y R. L. Smith, "Simulated Annealing and Adaptive Search in Global Optimization", *Probability in the Engineering and Informational Science*, 8:571–590, 1994.

[Rubinstein, 1981] R. Y. Rubinstein, *Simulation and the Monte Carlo Method*, J.Wiley, 1981.

[Sacksman, 1991] S. Sacksman, *Advances in Applied Probability*, cap. Iterated Local Search, págs. 866–893, 1991.

[Sandborn y Griffiths, 1998] A. N. Sandborn y T. L. Griffiths, "Markov chain Monte Carlo with people", *Advances in Neural Information Processing Systems*, (20), 1998.

[Schoen, 2002] F. Schoen, *Handbook of Global Optimization Volume 2*, cap. Two-phase Methods for Global Optimization, págs. 151–177, Kluwer Academic Publishers, 2002.

[Schrack y Borowski, 1972] G. Schrack y N. Borowski, *Numerical Methods For Nonlinear Optimization*, cap. An experimental comparison of three random searches, págs. 137–147, Academic Press, 1972.

[Schrack y Choit, 1976] G. Schrack y M. Choit, "Optimized Relative Step Size Random Searches", *Mathematical Programming*, 10(270-276), 1976.

[Smith y Shafi, 1997] P. Smith y M. Shafi, "Quick simulation: A review of Importance Sampling techniques in communications systems", *IEEE journal on selected areas in communications*, 15(4):597–613, 1997.

[Smith, 1984] R. L. Smith, "Eficient Monte Carlo Procedures for Generating Points Uniformly Distributed Over Bounded Region", *Operations Research*, 32:1296–1308, 1984.

[Solis y Wets, 1981] F. J. Solis y J. Wets, "Minimization by Random Search Techniques", *Mathematics of Operations Research*, 6(19-30), 1981.

[Spall, 2003] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control*, Wiley, Hoboken, 2003.

[Storn y Price, 1997] R. Storn y K. Price, "Differential evolution. A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, 11:341–359, 1997.

[Talbi, 2002] E. G. Talbi, "A taxonomy of hibrid metaheuristic", *Journal of Heuristics*, 8:541–564, 2002.

[Thisted, 1988] R. A. Thisted, *Elements of Statistical Computing: Numerical Computation*, Chapman and Hall, 1988.

[Torn y Zilinskas, 1989] A. Torn y A. Zilinskas, *Global Optimization*, Springer-Verlag, 1989.

[Ugray et al., 2007] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly y R. Marti, "Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization", *Informs Journal on Computing*, 3(19):328–340, 2007.

[Vanderbilt y Louie, 1984] D. Vanderbilt y S. Louie, "A Monte Carlo simulated annealing approach to optimization over continuous variables", *Journal of computational Physics*, (56):259–271, 1984.

[Vavasis, 1995] S. A. Vavasis, *Handbook of Global optimization*, cap. Complexity Issues in Global Optimization, págs. 27–41, Kluwer Academic Publisher, 1995.

[Von Neumann, 1951] J. Von Neumann, "Various techniques used in connection with random digits", *Applied Math Series*, 12:36–38, 1951.

[Voudouris y Tsang, 1995] C. Voudouris y E. Tsang, "Guided Local Search. Technical Report CSM-247", *Informe Técnico*, Department of Computer Science. University of Essex., 1995.

[Voudouris y Tsang, 1997] C. Voudouris y E. Tsang, "Solving the Radio Link Frequency Assignment Problem using Guided Local Search", en *NATO Symposium on Radio Length Frequency Assignment, Sharing and Conservation Systems (Aerospace)*, Aalborg, Denmark, 1997.

[Wade y Rayward-Smith, 2000] A. S. Wade y V. J. Rayward-Smith, "Effective local search for the Steiner tree problem", *Proceedings of the 44th annual Southeast regional conference*, 2000.

[Wang y Chen, 1996] P. P. Wang y D. Chen, "Continuous optimization by a variant of simulated annealing", *Computational Optimization and Applications*, (6):59–71, 1996.

[Winker, 2000] P. Winker, "Optimized multivariate lag structure selection", *Computational Economics*, 16:87–103, 2000.

[Winker, 2001] P. Winker, *Optimization Heuristics in Econometrics: Applicattions of Threshold Accepting*, Willey, 2001.

[Winker y Maringer, 2007] P. Winker y D. Maringer, *Optimisation Econometric and Financial Analysis*, cap. The threshold accepting optimisation algorithm, p. 107Ű125, Springer, 2007.

[Wood et al., 2001] G. R. Wood, Z. B. Zabinsky y B. P. Kristinsdottir, "Hesitant Adaptive Search: the Distribution of the Number of Iterations to Convergence", *Mathematical Programming*, 89(3):479–486, 2001.

[Zabinsky y Smith, 1992] Z. Zabinsky y R. Smith, "Pure Adaptive Search in Global Optimization", *Mathematical Programming*, (53):323–338, 1992.

[Zabinsky et al., 1993] Z. B. Zabinsky, R. L. Smith, J. F. McDonald y D. Kaufman, "Improving Hit-and-Run for Global Optimization", *Journal of Global Optimization*, 3:171–192, 1993.

[Zabinsky et al., 1995] Z. B. Zabinsky, G. R. Wood, M. A. Steel y W. P. Baritompa, "Pure Adaptive Search for Finite Global Optimization", *Mathematical Programming*, (69):443–448, 1995.

[Zlochin et al., 2004] M. Zlochin, M. Birattari, N. Meuleau y M. Dorigo, "Model-Based Search for Combinatorial Optimization: A Critical Survey", *Annals of Operations Research*, (131):373–395, 2004.