Ingeniería y Ciencia
ISSN:1794-9165 | ISSN-e: 2256-4314
ing. cienc., vol. 15, no. 30, pp. 117–140, julio-diciembre. 2019.
http://www.eafit.edu.co/ingciencia

Algoritmo de recocido simulado generalizado para Matlab

Jorge Homero Wilches Visbal[1]y Alessandro Martins Da Costa[2]

Recepción: 03-05-2019 | Aceptación: 20-08-2019 | En línea: 29-11-2019

PACS:02.60.-x, 02.70.-c,

doi:10.17230/ingciencia.15.30.6

Resumen

Muchos problemas en física, matemáticas e ingeniería, demandan la determinación del óptimo global de funciones multidimensionales. El recocido simulado es un método metaheurístico que tiene por objeto dar solución a problemas de optimización global. Existen tres tipos de recocido simulado: i) recocido simulado clásico; ii) recocido simulado rápido y iii) recocido simulado generalizado. De entre estos, el recocido simulado generalizado es demostradamente el más eficiente. Matlab, uno de los softwares más ampliamente usados en simulación numérica y programación científica, dispone de una caja de herramientas con funciones basadas tanto en métodos determinísticos como estocásticos capaces de resolver una gran cantidad de problemas de optimización. En este artículo se describió el método de recocido simulado generalizado, se elaboró la función GSA que alberga este método y se aplicó en algunos problemas matemáticos que permitieron evaluar la eficiencia de GSA respecto de algunas funciones de optimización de Matlab. Como resultado, se obtuvo que la función GSA no solo consigue ser efectiva en su convergencia al óptimo global sino que, además, lo hace con rapidez. Así mismo se observó que, en lineas generales, GSA fue más

[1] Universidad del Magdalena, jhwilchev@gmail.com, ORCID:0000-0003-3649-5079, Santa Marta, Colombia.
[2] Universidad de São Paulo, amcosta@usp.br, São Paulo, Brasil.

eficiente que las funciones con las que fue comparada. Por tanto, puede concluirse que la función GSA es en una alternativa novedosa y efectiva para el abordaje de problemas de optimización utilizando Matlab.

Palabras clave: Recocido simulado; optimización; eficiencia; GSA; Matlab.

Generalized Simulated Annealing Algorithm for Matlab

Abstract

Many problems in biology, physics, mathematics, and engineering, demand the determination of the global optimum of multidimensional functions. Simulated annealing is a meta-heuristic method that solves global optimization problems. There are three types of simulated annealing: i) classical simulated annealing; ii) fast simulated annealing and iii) generalized simulated annealing. Among them, generalized simulated annealing is the most efficient. Matlab is one of the most widely software used in numeric simulation and scientific computation. Matlab optimization toolbox provides a variety of functions able to solve many complex problems. In this article, the generalized simulated annealing method was described, the GSA function that contains this method was applied to some mathematical problems were solved in order to evaluate the efficiency of GSA with respect to some of Matlab optimization functions. As a result, it was found that the GSA function not only manages to be effective in its convergence to the global optimum but also it does so quickly. Likewise, it was observed that, in general terms, GSA was more efficient than the functions with which it was compared. Therefore, it can be concluded that the GSA function is a novel and effective alternative for addressing optimization problems using Matlab.

Keywords: Simulated annealing; efficiency; optimization; GSA; Matlab.

1 Introducción

Frecuentemente, una gran cantidad de problemas relevantes en matemática, estadística, física e ingeniería tienen por objeto la determinación del óptimo global de una función multidimensional [1],[2],[3]. Un método de optimización es aquel que involucra el hallazgo del óptimo (máximo o mínimo) global de una función objetivo que representa a un determinado problema [4].

Métodos de optimización son rutinariamente clasificados como estocásticos o determinísticos. Un método de optimización determinístico es aquel que bajo las mismas entradas producirá los mismos resultados. Uno de carácter estocástico será aquel que para iguales entradas puede arrojar resultados diferentes [5],[6]. Cuando un problema de optimización global envuelve una función objetivo convexa (un óptimo global), cualquier método determinístico tradicional, tal como el simplex, el de gradiente y el de cuasi-Newton resolverá fácilmente el problema. No obstante, si el problema implica una función objetivo no convexa (varios óptimos locales distanciados entre sí) es muy probable que métodos determinísticos, aunque rápidos, puedan quedar atrapados en algunos de los óptimos locales sin posibilidad de convergencia al óptimo global [7],[8],[9]. Contrariamente, aunque los métodos estocásticos suelan ser más lentos, estos evitan más fácilmente quedar atrapados en mínimos locales, siendo generalmente más exitosos en alcanzar el óptimo global [8],[9].

Varios métodos estocásticos han sido propuestos para tratar problemas de optimización global [10]. Algunos de los más conocidos son: mínimos cuadrados [11]; algoritmo genético [12]; recocido simulado [13] y búsqueda tabú [14]. Entre estos, el recocido simulado es considerado como uno de los más populares debido a su alta eficiencia [7],[15].

El recocido simulado es un método metaheurístico que permite solucionar problemas de optimización global. Debido a su naturaleza estocástica, este método garantiza que, en sentido probabilístico, el óptimo global será alcanzado. Su principal desventaja es que el tiempo de búsqueda puede tornarse infinito [16]. El recocido simulado es especialmente efectivo para resolver problemas de larga escala (multidimensionales) en los que el óptimo global se encuentra escondido detrás de muchos óptimos locales [2],[17].

La idea central del recocido simulado es imitar al proceso de recocido físico. En este último, un metal es calentado hasta una temperatura mayor a la de su punto de fusión (muchos estados físicos permitidos) para inmediatamente después ser gradualmente enfriado hasta llegar a una temperatura muy baja (único estado físico permitido). Si el proceso de enfriamiento es lo suficientemente lento, los átomos del metal formarán una estructura libre de defectos o estructura estable. En contrapartida, si el proceso de enfriamiento se realiza con rapidez, los átomos del metal formarán una estructura defectuosa, llena de irregularidades e imperfecciones, o estructura

metaestable [2],[18],[19].

En consecuencia, es posible establecer un paralelo entre recocido simulado y recocido físico tal que, en cada ciclo del proceso, la estructura física del metal corresponda a la solución del problema de optimización; el valor de la energía en dicha estructura equivalga al valor de la función objetivo en la solución y la temperatura del metal funcione como un parámetro de control [20]. De este modo, un estado físico con irregularidades sería un óptimo local. Entretanto, un estado libre de imperfecciones correspondería a un óptimo global [2].

En el recocido simulado, tres factores son cruciales para la convergencia al óptimo global: i) mecanismo de caminada o distribución de visitación; ii) mecanismo de decisión o probabilidad de aceptación; iii) mecanismo de direccionamiento o esquema de enfriamiento [16].

El mecanismo de caminada se refiere a la búsqueda de una nueva solución a partir de la solución actual, por medio de un muestreo inteligente del espacio solución. Lo que caracteriza a un buen mecanismo de caminada es que cuando la temperatura de recocido es alta, el algoritmo es capaz de dar saltos largos desde la solución actual, mientras que cuando la temperatura es baja, el algoritmo apenas consigue saltar a puntos vecinos, indicando la presencia del algoritmo en la cuenca de atracción al óptimo global [16].

El mecanismo de decisión o probabilidad de aceptación indica la regla de cumplimiento mediante la cual una nueva solución puede ser aceptada. Una nueva solución será aceptada siempre que disminuya el valor de la función objetivo. Sin embargo, aquellas soluciones indeseadas que aumentan el valor de la función objetivo, pueden ser eventualmente aceptadas. Esto último permite al algoritmo evitar quedar atrapado en óptimos locales [16].

El mecanismo de direccionamiento o esquema de enfriamiento se refiere a la manera en que la temperatura dirige el proceso de enfriamiento. Cuanto más alta es la temperatura, más fácil es dar saltos largos. En cambio, cuanto menor es, los saltos en torno a la vecindad de la solución actual son recurrentes. Un buen mecanismo de direccionamiento evitará movimientos alrededor de un óptimo local a fin de acortar el tiempo de convergencia al óptimo global. En otras palabras, la temperatura actúa como una fuente de estocasticidad extremadamente conveniente para rehuir de óptimos locales, tal que, cerca del final del proceso de optimización, el sistema esté inmerso

en la cuenca de atracción hacia el óptimo global [7],[16].

De acuerdo con el mecanismo de caminada empleado, el recocido simulado puede ser clasificado en tres categorías [7],[16]: i) Recocido Simulado Clásico (Classical Simulated Annealing (CSA) en inglés) o Máquina de Boltzmann [13],[17]; ii) Recocido Simulado Rápido (Fast Simulated Annealing (FSA) en inglés) o Máquina de Cauchy [17],[21] y iii) Recocido Simulado Generalizado (Generalized Simulated Annealing (GSA) en inglés) o Máquina de Tsallis [7],[17]. Cada tipo de recocido simulado surgió del intento de agilizar el mecanismo de direccionamiento mientras se garantizaba que la probabilidad de quedar atrapado en un óptimo local era mínima [7].

En este artículo, se propone la función GSA como una herramienta rápida y eficaz para la solución de problemas de optimización global. Para evaluar su eficiencia, en términos de rapidez y eficacia, se realiza un análisis comparativo con respecto a algunas funciones del paquete de optimización de Matlab. Por último, se concluye que GSA no solo es capaz de obtener el óptimo global de problemas de optimización complejos, con mayor eficacia que sus pares del paquete de optimización de Matlab, sino que lo hace más rápidamente. Así, la función GSA se constituye en una alternativa novedosa y confiable para la resolución de problemas de optimización usando Matlab.

2 Materiales y Métodos

2.1 Recocido Simulado Clásico

El método de recocido simulado clásico fue propuesto por Kirkpatrick *et al* [13], a inicios de la década de los 80, como solución al problema del vendedor ambulante [18]. Este método extiende el reconocido procedimiento de Metrópolis para la estadística de Boltzmann-Gibbs [18]. En CSA, el mecanismo de direccionamiento se fundamenta en la introducción de una temperatura artificial, $T(t)$, a fin de evitar óptimos locales. La referencia [22] muestra que, si tal temperatura disminuye con el inverso del logaritmo del tiempo, como en la Ec. (1), el sistema alcanzaría el óptimo global.

$$T(t) = \frac{T_1}{\log(1+t)}, \tag{1}$$

en el que T_1 es el valor inicial de la temperatura de visitación, T y t es el número de ciclos temporales (iteraciones) [10].

El mecanismo de caminada, descrito a través de su distribución de visitación o función densidad de probabilidad, tiene que ver con el salto de exploración en el espacio solución, Δr_t, entre la solución actual r_t y la nueva solución $r_{(t+1)}$.

La distribución de visitación para CSA es la distribución gaussiana de probabilidad,

$$g(\Delta\vec{r}_t) = \frac{1}{\pi^{D/2}} \frac{e^{-\Delta\vec{r}_t^2/k_B T}}{T(t)^{D/2}}, \tag{2}$$

donde D es la dimensión del espacio solución y k_B es la constante de Boltzmann. Dado que la distribución de visitación de CSA es de búsqueda local, esta apenas permite dar pequeños saltos desde la solución actual [23].

El mecanismo de decisión o probabilidad de aceptar una nueva solución viene dado por la distribución de Boltzmann-Gibbs [18],

$$P_a(r_t \to r_{t+1}) = e^{-\Delta f_t/k_B T}, \tag{3}$$

en el que Δf_t es el cambio de valor de la función objetivo entre la solución actual, f_t, y la solución siguiente, f_{t+1}, o sea, $\Delta f_t = f_{t+1} - f_t$. Además, siempre que $f_{t+1} < f_t$, CSA admitirá la nueva solución y la actualizará en el siguiente ciclo de tiempo. Caso contrario, CSA admitirá la nueva solución solo de forma aleatoria.

2.2 Recocido simulado rápido

El método de recocido simulado rápido fue desarrollado por [21], en 1987, como una mejora al método clásico por medio de una modificación en la distribución de visitación de este. El objetivo era acelerar la convergencia al óptimo global [18], de ahí el adjetivo de "rápido".

La distribución de visitación, en lugar de ser una distribución gaussiana como en CSA, es una distribución de Cauchy-Lorentz [7],[10], o sea,

$$g(\Delta\vec{r}_t) = \frac{1}{\pi^{(D+1)/2}} \frac{\Gamma(\frac{D+1}{2})T(t)}{(\Delta\vec{r}_t^2 + T(t)^2)^{(D+1)/2}}, \tag{4}$$

donde Γ corresponde a la función gamma. La distribución de Cauchy, al ser de búsqueda semilocal, puede dar saltos lejanos desde la solución actual. No obstante, saltos en las cercanías del punto actual continuan siendo más frecuentes [18],[24].

El mecanismo de direccionamiento del FSA se fundamenta en que la temperatura de visitación disminuye inversamente con el tiempo de simulación [7],[10], es decir,

$$T(t) = \frac{T_1}{(1+t)}, \tag{5}$$

provocando que el enfriamiento sea mucho más rápido que en CSA [24].

El mecanismo de decisión del FSA permanece inalterado respecto al CSA. En otras palabras, la probabilidad de aceptación de un nuevo punto de solución viene dada por la Ec. (3) [24].

El hecho de que el FSA posea una distribución de visitación de búsqueda semilocal, no solo lo torna más rápido que el CSA, sino que también le representa mayor éxito en escapar de óptimos locales y así encontrar más fácilmente el óptimo global. No obstante, en problemas realistas de reconocida complejidad, ni el FSA, ni el CSA, son lo suficientemente eficientes como para hallar el óptimo global [24].

2.3 Recocido simulado generalizado

El método de recocido simulado generalizado, ideado de manera conjunta por Tsallis y Stariolo [7] y basado en el formalismo de Tsallis [25], generaliza la distribución de visitación, la probabilidad de aceptación y la temperatura de visitación, bajo la premisa de mejorar la eficiencia del proceso de optimización [10]. De esta manera, el CSA y el FSA vienen a ser situaciones particulares del GSA [10].

El GSA se desarrolla en tres etapas [15]: I) mecanismo generalizado de caminada, $g_v(\Delta\vec{r}_t)$; II) mecanismo generalizado de visitación, $P_{q_a}(r_t \rightarrow r_{t+1})$ y III) mecanismo generalizado de direccionamiento, $T_{q_v}(t)$.

La distribución de visitación del GSA es una distribución de Cauchy-Lorentz levemente distorsionada, también conocida como distribución gaussiana generalizada [8],

$$g_{q_v}(\Delta r_t) = \left(\frac{q_v - 1}{\pi}\right)^{D/2} \frac{\Gamma\left(\frac{1}{q_v - 1} + \frac{D-1}{2}\right)}{\Gamma\left(\frac{1}{q_v - 1} - \frac{1}{2}\right)} \times \frac{T_{q_v}^{-D/(3-q_v)}}{\left(1 + (q_v - 1)\frac{(\Delta \vec{r})^2}{T_{q_v}^{2/(3-q_v)}}\right)^{1/(q_v-1)+(D-1)/2}}, \tag{6}$$

cuya forma y apertura es controlada por el parámetro de visitación q_v. El parámetro D denota el número de dimensiones de r_t.

En la Figura 1 se muestra como g_{q_v} cambia su forma y apertura conforme cambia el valor de q_v.

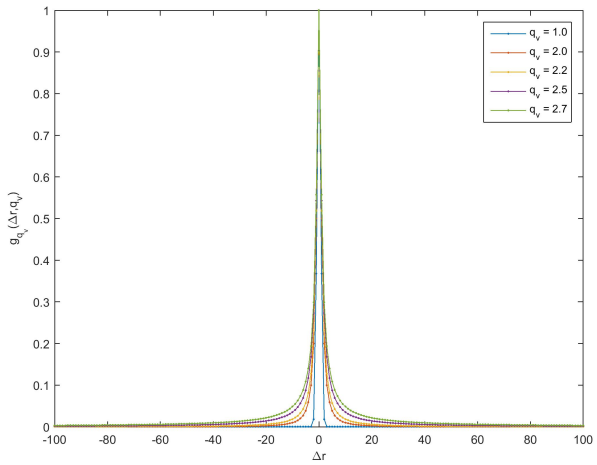

Figura 1: Distribución generalizada de visita g_{q_v} en función del salto en el espacio solución, para diferentes valores de q_v y $D = 1$. Se nota que a medida que aumenta el parámetro de visita, más apertura (espacio muestral) posee la función. Las curvas fueron normalizadas por su máximo valor. Fuente: elaboración propia.

La probabilidad de visitación del GSA es una generalización del algoritmo de Metrópolis [8],[24], de manera que,

$$P_{q_a}(x_t \to x_{t+1}) = \begin{cases} 1, & \Delta f_t < 0 \\ \left(1-(1-q_a)\frac{\Delta f_t}{k_B T_{q_a}}\right)^{1/(q_a-1)}, & \Delta f_t \geq 0 \\ 0, & \left(1-(1-q_a)\frac{\Delta f_t}{k_B T_{q_a}}\right) < 0 \quad \text{y} \quad q_a < 1, \quad \Delta f_t \geq 0. \end{cases} \tag{7}$$

donde q_a es el denominado parámetro de aceptación.

Finalmente, el esquema de enfriamiento viene dado por,

$$T_{q_v}(t) = T_{q_v}(1)\frac{2^{q_v-1}-1}{(1+t)^{q_v-1}-1}, \tag{8}$$

donde $T_{q_v}(1)$ es la temperatura de visitación inicial.

En la Figura 2 se visualiza cómo la temperatura de visitación disminuye en función del valor de q_v.

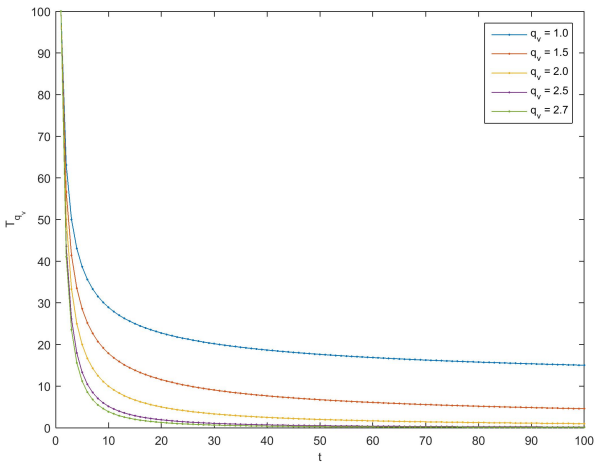

Figura 2: Esquema generalizado de enfriamiento expresado en función de los ciclos de tiempo. Se observa que a medida que q_v es incrementado, el enfriamiento se acelera. El valor inicial de la temperatura fue de 100. Fuente: elaboración propia.

Allí se puede observar que cuanto más altos son los valores de q_v, más rápidamente cae la temperatura de visitación. Esto representa un enfriamiento más rápido, es decir, menor tiempo de cálculo.

El proceso de recuperación del CSA y el FSA a partir del GSA depende del valor asignado a q_v [7]. Si $q_v = 1$, GSA recupera a CSA. Entretanto, cuando $q_v = 2$, GSA retorna a FSA [8],[15],[24]. Adicionalmente, cuando $q_v > 2$ el esquema de enfriamiento es acelerado, por lo que GSA será más eficiente, en la convergencia al global, que el CSA y el FSA [24]. Esto se atribuye al proceso de búsqueda no local y homogeneo implícito en la distribución de visitación del GSA.

Una búsqueda no local implica dar saltos fuera de la vecindad inmediata de la solución actual [24],[26]. El proceso de búsqueda no local se refleja en la mayor apertura de la distribución generalizada de visitación y en la mayor caída en la temperatura de visitación, sobre todo para $q_v > 2$ (modo GSA), como visto en la Figura 1 y la Figura 2, respectivamente. Por otra parte, una búsqueda homogenea es aquella que permite realizar transiciones lejanas a partir de la solución actual con bastante frecuencia, incluso, cuando $T_{q_v}(t) \to 0$ [24].

Para GSA, el parámetro de visitación está sujeto a $2 < q_v < 3$, mientras que el parámetro de aceptación, q_a, se recomienda sea tan negativo cuanto fuese posible, a fin de garantizar que una menor cantidad de soluciones puedan ser aceptadas [8].

En ese sentido, en [7] se sugiere que la combinación $q_v = 2.7$ y $q_a = -5.0$ es altamente eficiente para encontrar el óptimo global. Por otro lado, en [8] se llega a la conclusión de que la combinación $q_v = 2.62$ y $q_a = -5.0$ funciona bien para muchas aplicaciones prácticas. Finalmente, en [8],[23] y [27] se subraya que la eficiencia de GSA podría verse mejorada a través de una sinergia con algún método de optimización de búsqueda local.

2.4 Función GSA

La metodología empleada para el desarrollo de la función GSA consta de dos partes: la primera, relacionada con un generador de números aleatorios de Tsallis; la segunda, referente a la estructura del algoritmo de la función. El seccionamiento del desarrollo de la función obedece a que la eficiencia de

GSA depende críticamente de que tan apropiada es la técnica de generación de números aleatorios [16]. Se advierte que, si bien el propósito del presente artículo es adecuar el algoritmo de la función al lenguaje propio de Matlab, este puede, en principio, ser extendido a otros lenguajes de programación.

2.4.1 Generador de números aleatorios de Tsallis

En general, la parte más difícil de GSA es la generación de números aleatorios de Tsallis que obedezcan a la distribución generalizada de visitación expresada por la Ec. (6) [15]. El generador es requerido para simular la distribución generalizada de visitación de forma que el GSA pueda trabajar adecuadamente [16]. Una vez que las distribuciones de visitación no son matemáticamente invertibles, se hacen necesarios procedimientos numéricos para generar tales números [16]. La función generadora de números aleatorios de Tsallis viene estructurada de la siguiente manera [15]:

Función Z = `Tsallis_rnd` (T_{q_v}, q_v, D)

- **Definición de variables**
  - Variable de salida
    (i) Z: números aleatorios de Tsallis
  - Variable de entrada
    (i) T_{q_v}: temperatura de visitación
    (ii) q_v: coeficiente de visitación
    (iii) D: dimensión del vector solución

- **Desarrollo del algoritmo**
  - Inicializar las variables, fijando T_{q_v}, q_v, D.
  - Inicializar el bucle.
    Para t = 1:$t_{\text{máx}}$
  - Sea p un parámetro que depende de q_v tal que,

$$p = (3 - q_v)/(2(q_v - 1)). \tag{9}$$

– Sea s un parámetro que depende de T_{q_v} y q_v de forma que,

$$s = \sqrt{2(q_v - 1)}/T_{q_v}(t)^{\frac{1}{3-q_v}}. \tag{10}$$

– Sea x una variable que genera números aleatorios independientes e idénticos de una distribución normal [28]

$$x = \mu + \sigma \text{randn}(D), \tag{11}$$

donde μ es el valor medio de la distribución, σ es su desviación estándar y randn es la función generadora de números aleatorios normalmente distribuidos de Matlab [29]. La desviación estándar es calculada como:

$$\sigma = \sqrt{\frac{1}{2p(q_v - 1)/T_{q_v}(t)^{\frac{2}{3-q_v}}}}, \tag{12}$$

mientras el valor medio se establece en cero (μ=0).

– El parámetro y está relacionado con un generador de números aleatorios gamma γ(p,1), de modo que,

$$y = s\sqrt{u}, \tag{13}$$

en que u = gamrnd(p,1) es la función generadora de números gamma de Matlab [30].

– Finalmente, el generador de números aleatorios de Tsallis de dimensión D es,

$$\text{Z} = \frac{x}{y} \tag{14}$$

Final del bucle.

Fin de la función.

2.4.2 Algoritmo de la función GSA

A continuación, se presenta el desarrollo secuencial de la función GSA, teniendo en cuenta el resultado que proporciona la función auxiliar `Tsallis_rnd`.

Función [r, fval] = `GSA`(f, q_v, q_a, r_0, r_l, r_u, $t_{\text{máx}}$)

- **Definición de variables**
  - Variables de salida
    (i) r: solución óptima
    (ii) fval: valor de la función objetivo en la solución
  - Variables de entrada
    (i) f: valor de la función objetivo por ciclo de tiempo t
    (ii) q_a: coeficiente de aceptación
    (iii) q_v: coeficiente de visitación
    (iv) r_0: tentativa inicial de solución
    (v) r_l: restricciones inferiores en la solución
    (vi) r_l: restricciones superiores en la vector solución
    (vii) t_{mx}: número máximo de ciclos de ejecución del algoritmo

- **Desarrollo del algoritmo**
  - Inicializar la configuración del sistema, fijando f, q_v, q_a, r_0, r_l, r_u, $t_{\text{máx}}$.
  - Inicializar bucle externo.
    Para t = 1:$t_{\text{máx}}$
  - Definir $T_{q_v}(1)$ tan alto como sea posible. Una opción sería $T_{q_v}(1)$= $t_{\text{máx}}$.
  - (iv) Iniciar el esquema de enfriamiento dado por la Ec. (8).
  - (v) Iniciar el esquema de enfriamiento para la temperatura de aceptación, como [8], [26].

$$T_{q_a}(t) = \frac{T_{q_v}(t)}{t}, \tag{15}$$

  - Inicializar bucle interno.
    Para j = 1:$t_{\text{máx}}/5$
  - Generar números aleatorios de Tsallis por medio de la Ec. (14). Para problemas de optimización con restricciones, se sugiere usar,

$$\Delta r_t = \mathrm{Z}(r_u - r_l) \tag{16}$$

pues garantiza mejores resultados en términos de convergencia y rapidez. Aunque se puede optar por tomar $t_{\text{máx}}$, en lugar de $t_{\text{máx}}/5$, como número máximo de ciclos del bucle interno, se recomienda la elección de este último ya que consigue convergir en menor tiempo de cálculo sin afectar apreciablemente la eficacia.

– Generar un nuevo vector solución a partir de,

$$r_{t+1} = r_t + \Delta r_t \tag{17}$$

aplicando restricciones en las componentes como en la referencia [4].

– Calcular $f(r_{t+1})$ y comparar con $f(r_t)$. Aplicar la Ec. (7) de manera que siempre que $\Delta f(r_t) < 0$, entonces la solución r_t es aceptada. En caso de que $f(r_t*) \geq 0$, un número aleatorio es sorteado, usando la función rand, y comparado con el valor de P_{q_a}. Si $P_{q_a} >$ rand, entonces la solución r_t* es aceptada.

– Tornar a la solución aceptada en la solución actual del siguiente ciclo temporal, según sea el caso, esto es:

$$\begin{aligned} r_t &\rightarrow r_{t+1} \\ r_t* &\rightarrow r_{t+1}* \end{aligned} \tag{18}$$

Final del bucle interno.

– Repita todo el procedimiento hasta que $t = t_{\text{máx}}$ o introduzca una condición de parada distinta.
Final del bucle externo.

Fin de la función.

Con el propósito de evaluar su eficiencia, la función GSA es empleada para determinar el óptimo global de un conjunto de problemas de optimización. Asimismo, se realiza un análisis comparativo con las funciones `fmincon` [31], `lsqnonlin` [32] y `simannelbnd` [33], bajo esos mismos problemas. Cada una de estas funciones es usada en su configuración estándar, exceptuando el número máximo de ciclos temporales (iteraciones).

En general, todas estas funciones demandan los siguientes pasos básicos para su adecuado funcionamiento: i) escribir la función objetivo como una función anónima [34] asegurándose de que al ser evaluada resulte en un escalar; ii) definir e ingresar la tentativa inicial de solución (información a priori sobre la solución) y las restricciones en sus componentes (si las hubiese); iii) ingresar los valores de los parámetros iniciales (en el caso de GSA, estos son q_v, q_v, $t_{\text{máx}}$) y iv) ejecutar la función.

3 Resultados y discusión

La eficiencia de la función GSA, con respecto a las demás funciones, será evaluada por medio de la resolución de cinco problemas de optimización de variada complejidad. Para una comprensión idónea de los resultados, eficiencia significa la conjunción de rapidez y eficacia.

Rapidez será el tiempo de cálculo medio de las cinco ejecuciones. Eficacia se entiende como la tasa de éxito del algoritmo, esto es, el número de veces que cada función encuentra el óptimo dividido el número de veces que el algoritmo es ejecutado (tasa de convergencia al óptimo). Idealmente, una función es 100 % eficiente si posee 100 % de eficacia con tiempo de calculo igual a 0 s.

En todos los ejemplos, se asume $q_v = 2.7$, $q_a = -5$ mientras que $t_{\text{máx}}$, número máximo de iteraciones, será igual para todas las funciones de optimización. Las funciones de optimización se ejecutan veinte (20) veces en cada problema propuesto.

Ejemplo 1.

Considere la función $f(x)$ que describe un potencial de pozo doble [21], de manera que,

$$f(x) = x^4 - 16x^2 + 5x, \tag{19}$$

con tentativa inicial de solución $x_0 = 0$ y cuya solución debe estar restringida a $-100 < x < 100$. Se pide encontrar el mínimo global.

Solución.

Los resultados del proceso de minimización de la Ec. (19) arrojados por cada función de optimización, son mostrados en la Tabla 1 El número máximo de iteraciones fue establecido en $t_{\text{máx}} = 400$.

Tabla 1: Evaluación de la eficiencia de las funciones de optimización consideradas en el Ejemplo 1.

Función de Optimización	Rapidez (s)	Eficacia (%)
GSA	0.35	100
lsqnonlin	0.030	0
simannelbnd	0.13	100
fmincon	0.19	100

De la Tabla 1 se extrae que todas las funciones de optimización alcanzaron el mínimo global en cada una de las cinco ejecuciones, menos `lsqnonlin`. En otras palabras, todas las funciones fueron 100 % eficaces, excepto `lsqnonlin`. El mínimo global hallado se ubicó en $x = -2.9$ con $f(x) = -78.3$.

Los resultados de la Tabla 1 indican que la función `fmincon` fue igual de eficaz que `simannelbnd` y `GSA`, y evidentemente más rápida. Entretanto, la `GSA` consiguió el mismo nivel de eficacia de `fmincon` y `simannelbnd`, pero con menor rapidez que estas.

Con esto, se concluye que `fmincon` fue la función más eficiente en encontrar el mínimo global del Ejemplo 1. Aunque la función `GSA` no logró la ser más rápida, demostró alcanzar el mínimo con una rapidez razonablemente corta.

Ejemplo 2.

La función objetivo $f(r)$, es una versión bidimensional de la función Sech [35]:

$$f(r) = -10\text{sech}(|r - r_1|) - 20\text{sech}(0.0003(-|r - r_2|)) - 1, \qquad (20)$$

donde $r_1 = (1, 1)$, $r_2 = (10^5, -10^5)$. Se pide hallar el mínimo global de esta función.

Solución.

Para este ejemplo, se optó por restringir la solución r a $-10^6 < r_x < 10^6$ y $-10^6 < r_y < 10^6$, donde $r = (r_x, r_y)$. Hecho esto, se propuso como tentativa inicial de solución, $r = (0, 0)$. El número máximo de iteraciones fue definido en $t_{\text{máx}} = 800$, puesto que fue el valor a partir del cual, al menos una de las funciones de optimización, alcanzó el mínimo global.

El resultado de la minimización de la Ec. (20) se visualiza en la Tabla 2.

Tabla 2: Evaluación de la eficiencia de las funciones de optimización consideradas en el Ejemplo 2.

Función de Optimización	Rapidez (s)	Eficacia (%)
GSA	1.8	100
lsqnonlin	0.032	0
simannelbnd	0.14	0
fmincon	0.034	0

En la Tabla 2, se observa que solo `GSA` dió con el mínimo global en todas las ejecuciones. El mínimo global fue encontrado en $r = (10^5, -10^5)$ con $f(r) = -21$. Por otro lado, `GSA` alcanzó el mínimo global con una rapidez aceptable. Se concluye que `GSA` fue la única función eficiente del Ejemplo 2.

A pesar de que las funciones `simannelbnd` y `fmincon` fueron absolutamente ineficientes para hallar el mínimo global, estas pudieron encontrar el mínimo local, ubicado en $r = (1, -1)$ con $f(r) = -11$, en el 15 % y 100 % de las ejecuciones, respectivamente.

Ejemplo 3.

La minimización de la función de Rosenbrock es reconocido como un problema de optimización notoriamente difícil de resolver para muchos algoritmos de optimización. Esta función viene dada por [36],

$$f(x) = 100(x_2 - x_1^2) + (1 - x_1)^2, \tag{21}$$

donde se sugiere como tentativa inicial de solución, $x_1 = -1.2$ y $x_2 = 1$.

Solución.

Para dar solución a la Ec. (21) se procede como en los anteriores problemas. Las restricciones impuestas al vector solución $x = (x_1, x_2)$ fueron las mismas que en el ejemplo 2. El número máximo de iteraciones fue establecido en $t_{\text{máx}} = 200$, pues a partir de este valor, al menos una de las funciones de optimización, consiguió el mejor resultado.

En la Tabla 3 se presentan los resultados del proceso de minimización, para cada una de las funciones de optimización.

Tabla 3: Evaluación de la eficiencia de las funciones de optimización consideradas en el Ejemplo 3.

Función de Optimización	Rapidez (s)	Eficacia (%)
GSA	0.16	100
lsqnonlin	0.048	0
simannelbnd	0.15	20
fmincon	0.080	100

A partir de la Tabla 3, se colige que las funciones `GSA` y `fmincon` alcanzaron el mínimo global en todas las ejecuciones. Entretanto, `simannelbnd` solo lo hizo en cuatro de las ejecuciones. Además, se observó que a la función `fmincon` le tomó menor tiempo de cálculo encontrar el mínimo que a `GSA`.

Vale destacar que al aumentar el número máximo de iteraciones a 800, la eficiencia de `simannelbnd` pasó a ser 40 %, sugiriendo que, en algunos casos, el incremento de $t_{\text{máx}}$ puede repercutir positivamente en la eficiencia de la función.

Se concluye que la función `fmincon` fue la más eficiente para dar con el mínimo global, encontrado en $x = [1, 1]$ con $f(x) = 10^{-7}$, seguida por `GSA` al ser menos rápida.

Ejemplo 4.

La función objetivo $f(x)$, en la Ec. (22), representa un problema de minimización global difícil de resolver por cuanto posee un mínimo global escondido detrás de quince mínimos locales [7].

$$f(x) = \sum_{i=1}^{4} (x_i^2 - 8)^2 + \sum_{i=1}^{4} x_i + E_0, \tag{22}$$

en que $E_0 = 57,33$ con $f(x) \geq 0$. Se sugiere como tentativa inicial de solución, $x_0 = [1, 1, 1, 1]$.

Solución.

Los resultados del proceso de minimización de la Ec. (22) pueden verse en la Tabla 4. Para este ejemplo, las restricciones del vector solución x se definieron como siendo $(-10^6, -10^6, -10^6, -10^6) < x < (10^6, 10^6, 10^6, 10^6)$. El número máximo de iteraciones se fijó en $t_{\text{máx}} = 1400$ pues fue el valor a partir del cual una de las funciones de optimización alcanzó, al menos una vez, el mínimo global.

Tabla 4: Evaluación de la eficiencia de las funciones de optimización consideradas en el Ejemplo .

Función de Optimización	Rapidez (s)	Eficacia (%)
GSA	5.94	100
lsqnonlin	0.021	0
simannelbnd	0.31	15
fmincon	0.065	0

De la Tabla 4 se evidencia que la única función de optimización que obtuvo el mínimo global con 100 % de eficacia fue `GSA`. Por su parte, la función `simannelbnd`, apenas pudo alcanzar el mínimo global en tres de las veinte ejecuciones. El mínimo global fue hallado $x = [2, 9, 2, 9, 2, 9, 2, 9]$ con $f(x) = -0.0017$. Cabe mencionar que `simannelbnd` dió con al menos uno de los quince mínimos locales en el 55 % de las ejecuciones.

Se destaca que, a diferencia de algunos ejemplos anteriores, la función `fmincon` fue absolutamente ineficaz para dar con el mínimo global de esta función objetivo. Tampoco alcanzó ningún mínimo local. La función `lsqnonlin` mantuvo la ineficiencia mostrada en ejemplos anteriores.

De este ejemplo se concluye que la función de optimización más eficiente fue `GSA` seguida por `simannelbnd`.

Ejemplo 5.

Considere la siguiente función objetivo,

$$f(x) = \tan(\cos(x)), \tag{23}$$

donde se pide hallar el máximo global alrededor de $x_0 = 5$ [37].

Solución.

Antes de empezar a resolver el problema, cabe señalar que, como la Ec. (23) expresa una función periódica, se ha pedido encontrar el máximo cerca de x_0, aunque en verdad existan otros máximos con igual valor de la función objetivo.

Para problemas de maximización, lo primero a ser realizado es la modificación del signo de la función objetivo, esto es, $f(x) \rightarrow -f(x)$. Además, como la búsqueda del óptimo debe ser en las cercanías de 5, se adoptó el intervalo $3 < x < 7$ como restricción. El número máximo de ciclos temporales usados para `GSA` se fijó en $t_{\text{máx}} = 100$.

Los resultados del proceso de minimización de la Ec. (23) se encuentran consignados en la Tabla 5.

Tabla 5: Evaluación de la eficiencia de las funciones de optimización consideradas en el Ejemplo 5.

Función de Optimización	Rapidez (s)	Eficacia (%)
GSA	0.032	100
lsqnonlin	0.080	0
simannelbnd	0.051	100
fmincon	0.018	100

De la Tabla 5, se nota que las funciones `fmincon`, `GSA` y `simannelbnd` fueron absolutamente eficacez para hallar el máximo local ubicado en $x = 6.28$ con $f(x) = 1.56$. En términos de rapidez, las funciones `fmincon` y `GSA` arrojaron los mejores resultados.

Se concluye que la función `fmincon` fue la más eficiente de todas, seguida por `GSA` y `simannelbnd`, respectivamente.

A partir del análisis de los cinco anteriores ejemplos, algunas apreciaciones generales acerca del rendimiento de la función `GSA` son: i) `GSA` fue la única función en encontrar el óptimo global, al menos una vez, en todos los ejemplos; ii) `GSA` alcanzó el óptimo global en tiempos razonablemente cortos a pesar de ser una función estocástica (inherentemente más lentas que las deterministas); iii) `GSA` fue capaz de resolver diversos problemas de optimización independiente de su dificultad y del tipo de problema (minimización o maximización); iv) Para algunos problemas, pocas iteraciones

fueron necesarias para que GSA logre su acometido, en otros, se precisó de una mayor cantidad (en aquellos con problemas de mayor escala: Ejemplos 3 y 4); v) A partir de los cinco ejemplos propuestos, se colige que GSA resultó ser la función de optimización más eficiente.

4 Conclusiones

La función GSA basada en el método de recocido simulado generalizado mostró ser altamente eficiente, en términos de rapidez y eficacia, para la obtención del óptimo global de funciones multidimensionales de difícil solución. Inclusive, al compararse con algunas de las funciones del paquete de optimización de Matlab, llegó a ser, en líneas generales, más eficiente que todas estas.

La elección de los parámetros de visitación, aceptación y número máximo de iteraciones, es crucial en el propósito de alcanzar el óptimo. Sin embargo, esto también dependerá de las características intrínsecas del problema abordado, como son, el espacio solución, la información a priori y el número de dimensiones de la solución a ser alcanzada.

Se constató que cuanto mayor eran las dimensiones del problema, mayor debía ser el número de iteraciones máximas de la función GSA a fin de alcanzar el óptimo global en muchas o todas las ejecuciones de la función.

La función GSA con parámetros de visitación y aceptación de 2.7 y -5 respectivamente y número de iteraciones máximas entre 100 y 1500, podría ser una configuración de optimización adecuada para resolver una gran variedad de problemas.

Agradecimientos

Los autores agradecen a la Facultad de Ciencias de la Salud de la Universidad del Magdalena por el tiempo prestado para redactar este trabajo, así como a la Fundação Amparo à Pesquisa do Estado de São Paulo (FAPESP) y a la Coordenação de Aperfeiçoamento de Pessoas do Nível Superior (CAPES) por el apoyo económico en el desarrollo de la tesis doctoral de la cual deriva este trabajo.

Referencias

[1] L. Ingber, "Simulated annealing: Practice versus theory," *Mathematical and Computer Modelling*, vol. 18, no. 11, pp. 29 – 57, 1993. [Online]. Available: https://doi.org/10.1016/0895-7177(93)90204-C 118

[2] C. Carletti, P. Meoli, and W. R. Cravero, "A modified simulated annealing algorithm for parameter determination for a hybrid virtual model," *Physics in Medicine and Biology*, vol. 51, no. 16, pp. 3941–3952, jul 2006. 118, 119, 120

[3] R. V. Vidal, *Applied simulated annealing.* Springer, 1993, vol. 396. 118

[4] W. Y. Yang, W. Cao, T.-S. Chung, J. Morris *et al.*, *Applied numerical methods using MATLAB.* Wiley Online Library, 2005. 118, 130

[5] A. Barbato and A. Capone, "Optimization models and methods for demand-side management of residential users: A survey," *Energies*, vol. 7, no. 9, pp. 5787–5824, sep 2014. [Online]. Available: https://doi.org/10.3390/en7095787 119

[6] O. Erdinc, *Optimization in renewable energy systems: recent perspectives.* Butterworth-Heinemann, 2017. 119

[7] C. Tsallis and D. A. Stariolo, "Generalized simulated annealing," *Physica A: Statistical Mechanics and its Applications*, vol. 233, no. 1-2, pp. 395–406, nov 1996. [Online]. Available: https://doi.org/10.1016/s0378-4371(96)00271-3 119, 121, 122, 123, 126, 134

[8] Y. Xiang, S. Gubian, B. Suomela, and J. Hoeng, "Generalized simulated annealing for global optimization: The GenSA package," *The R Journal*, vol. 5, no. 1, p. 13, 2013. [Online]. Available: https://doi.org/10.32614/rj-2013-002 119, 123, 124, 126, 129

[9] G. Giorgi and K. Yamashita, *Theoretical Modeling of Organohalide Perovskites for Photovoltaic Applications.* CRC Press, 2017. 119

[10] M. D. de Andrade, K. C. Mundim, and L. A. C. Malbouisson, "Convergence of the generalized simulated annealing method with independent parameters for the acceptance probability, visitation distribution, and temperature functions," *International Journal of Quantum Chemistry*, vol. 108, no. 13, pp. 2392–2397, 2008. [Online]. Available: https://doi.org/10.1002/qua.21736 119, 122, 123

[11] J. E. Dennis, D. M. Gay, and R. E. Welsch, "An adaptive nonlinear least square algorithm," 1977. 119

[12] D. E. Golberg, "Genetic algorithms in search optimization & machine learning. 1953." 119

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," in *World Scientific Lecture Notes in Physics*. WORLD SCIENTIFIC, nov 1986, pp. 339–348. [Online]. Available: https://doi.org/10.1142/9789812799371_0035 119, 121

[14] F. Glover, M. Laguna, E. Taillard, and D. de Werra, *Tabu search*. Springer, 1993. 119

[15] T. Schanze, "An exact d-dimensional tsallis random number generator for generalized simulated annealing," *Computer Physics Communications*, vol. 175, no. 11-12, pp. 708–712, dec 2006. [Online]. Available: https://doi.org/10.1016/j.cpc.2006.07.012 119, 123, 126, 127

[16] J. Deng, C. Chang, and Z. Yang, "An exact random number generator for visiting distribution in gsa," *energy*, vol. 2, no. 2, 1987. 119, 120, 121, 127

[17] M. A. Moret, P. G. Pascutti, P. M. Bisch, and K. C. Mundim, "Stochastic molecular optimization using generalized simulated annealing," *Journal of Computational Chemistry*, vol. 19, no. 6, pp. 647–657, apr 1998. [Online]. Available: https://doi.org/10.1002/(sici)1096-987x(19980430)19:6<647::aid-jcc6>3.0.co;2-r 119, 121

[18] T. J. P. Penna, "Traveling salesman problem and tsallis statistics," *Physical Review E*, vol. 51, no. 1, pp. R1–R3, jan 1995. [Online]. Available: https://doi.org/10.1103/physreve.51.r1 120, 121, 122, 123

[19] G. Haeser and M. G. Ruggiero, "Aspectos teóricos de simulated annealing e um algoritmo duas fases em otimização global," *Trends in Applied and Computational Mathematics*, vol. 9, no. 3, pp. 395–404, 2008. 120

[20] K.-L. Du and M. Swamy, "Search and optimization by metaheuristics," *Techniques and Algorithms Inspired by Nature; Birkhauser: Basel, Switzerland*, 2016. 120

[21] H. Szu and R. Hartley, "Fast simulated annealing," *Physics Letters A*, vol. 122, no. 3-4, pp. 157–162, jun 1987. [Online]. Available: https://doi.org/10.1016/0375-9601(87)90796-1 121, 122, 131

[22] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721–741, nov 1984. [Online]. Available: https://doi.org/10.1109/tpami.1984.4767596 121

[23] H. Peyvandi, *Computational Optimization in Engineering: Paradigms and Applications*. BoD–Books on Demand, 2017. 122, 126

[24] Y. Xiang and X. G. Gong, "Efficiency of generalized simulated annealing," *Physical Review E*, vol. 62, no. 3, pp. 4473–4476, sep 2000. [Online]. Available: https://doi.org/10.1103/physreve.62.4473 123, 124, 126

[25] C. Tsallis, "Possible generalization of boltzmann-gibbs statistics," *Journal of Statistical Physics*, vol. 52, no. 1-2, pp. 479–487, jul 1988. [Online]. Available: https://doi.org/10.1007/bf01016429 123

[26] Y. Xiang, D. Sun, W. Fan, and X. Gong, "Generalized simulated annealing algorithm and its application to the thomson model," *Physics Letters A*, vol. 233, no. 3, pp. 216–220, aug 1997. [Online]. Available: https://doi.org/10.1016/s0375-9601(97)00474-x 126, 129

[27] Y. Xiang, S. Gubian, and F. Martin, "Generalized simulated annealing," in *Computational Optimization in Engineering-Paradigms and Applications*. InTech, 2017. 126

[28] M. Viswanathan, "Simulation and analysis of white noise in matlab," Nov. 2019. [Online]. Available: https://www.gaussianwaves.com/2013/11/simulation-and-analysis-of-white-noise-in-matlab/ 128

[29] "Normally distributed random numbers," 2006. [Online]. Available: https://www.mathworks.com/help/matlab/ref/randn.html 128

[30] "Gamma random numbers," 2006. [Online]. Available: https://www.mathworks.com/help/stats/gamrnd.html 128

[31] "Fmincon function," 2006. [Online]. Available: https://la.mathworks.com/help/optim/ug/fmincon.html?lang=en 130

[32] "Lsqnonlin function," 2006. [Online]. Available: https://la.mathworks.com/help/optim/ug/lsqnonlin.html?lang=en 130

[33] "Simannelbnd function," 2007. [Online]. Available: https://la.mathworks.com/help/gads/simulannealbnd.html 130

[34] "Anonymous functions," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html 131

[35] "Isolated global minimum," 2006. [Online]. Available: https://www.mathworks.com/help/gads/isolated-global-minimum.html 132

[36] "Rosenbrock function," 2006. [Online]. Available: https://la.mathworks.com/help/matlab/ref/fminsearch.html#bvadxhn-6 133

[37] "Maximing an objective function," 2006. [Online]. Available: https://www.mathworks.com/help/optim/ug/maximizing-an-objective.html 135