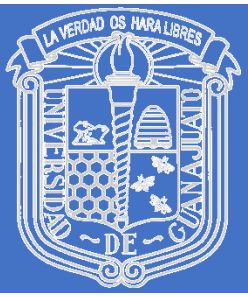

UNIVERSIDAD DE GUANAJUATO

GUÍA PRÁCTICA DE USO DE ALGORITMO GENÉTICO EN MATLAB

Autores:

José Ángel Morales Hernández

María Jimena Ramírez Esteban

Omar Alejandro Sánchez Ramírez

Fecha: 25/07/2019

Contenido

Guía práctica de uso de Algoritmo Genético en Matlab

Introducción

El presente trabajo muestra casos prácticos del uso del algoritmo de búsqueda estocástica conocido como algoritmo genético. Se revisarán casos de mínimos y máximos globales para problemas sin restricciones. En los casos de problemas con restricciones, se discutirá sobre los mínimos y máximos de frontera, así como los tipos de restricción, lineal y no lineal. Todos los casos se programaron en el software Matlab R2017a. Se supone que el lector tiene conocimientos básicos de programación y está familiarizado con la interfaz de Matlab.

Antes de comenzar: ¿Cómo funciona el algoritmo genético?

El algoritmo genético hace referencia a su análogo biológico. Al igual que las células, sufre procesos de replicación y mutación. El funcionamiento es el siguiente (figura 1):

1. **Arranque**: El algoritmo crea un grupo de posibles soluciones para el problema dado llamado *población*; usualmente, entre 50 y 100 candidatos. No requiere un valor de partida, como la mayoría de los algoritmos determinísticos.
2. **Evaluación**: Utiliza una función basada en el problema a resolver para evaluar la calidad de las soluciones propuestas, llamada *fitness*.
3. **Selección**: En base a esto, se distinguen tres grupos de soluciones: de *élite*, de *cruza* y de *mutación*. El grupo de *élite* constituye a las mejores soluciones, por lo que permanece inalterado.
4. **Cruza:** El grupo de *cruza*, como lo dice su nombre, se combina para formar un nuevo conjunto de soluciones mejores a sus predecesoras. Por ello, se considera al actual grupo *padres* y al nuevo grupo *hijos*. Este se considera el grupo más grande de los tres.
5. **Mutación:** El último grupo, de *mutación*, sufre alteraciones aleatorias en su valor.
6. **Reemplazo:** Los *hijos,* las *mutaciones* y la *élite* se reagrupan. Esta será la nueva *generación*.
7. **Terminación:** La nueva *generación* pasa por los pasos 2 al 6 hasta que no haya cambios apreciables entre sus miembros. Se dice que el programa ha convergido a una solución.

Cabe mencionar que el algoritmo genético, al estar basado en una búsqueda estocástica, tratará de hallar el óptimo global y no el punto crítico más cercano, como la mayoría de los algoritmos determinísticos.

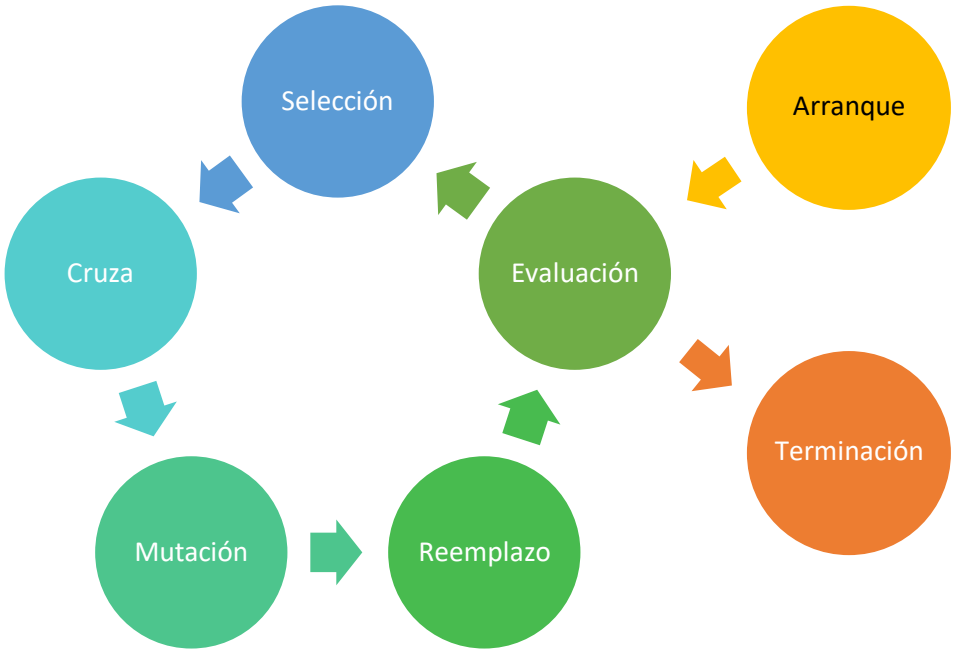

Figura 1. Esquema de funcionamiento de un algoritmo genético.

Plantear un problema de optimización en Matlab

En ocasiones, el usuario conoce la forma matemática de su problema, pero desconoce la manera de *comunicarlo* a Matlab. Aquí se muestra la forma de codificarlo a través de ejemplos prácticos.

Nota:

- Los algoritmos de optimización buscan un mínimo. En caso de optimizar un máximo, éste deberá plantearse como sigue:

$$\max f(\boldsymbol{x}) = \min -f(\boldsymbol{x})$$

  De esta manera, se consigue invertir la forma de la función cambiando la posición del punto crítico de máximo a mínimo. A esta forma del problema se le llama *problema dual.* Por ello, los resultados hallados serán válidos.

La forma de llamar al algoritmo genético en Matlab es la siguiente:

```
x = ga(fun,nvars,A,b,Aeq,beq,lb,ub,nonlcon)
```

A lo largo de esta guía se explican los distintos argumentos que componen esta función.

Función sin restricciones

A continuación, se revisa un ejemplo básico de cálculo multivariable.

Ejemplo 1: Hallar el máximo de la siguiente función.

$$z = f(x, y) = 25 - x^2 - y^2$$

Solución: Se procede a tomar el problema dual, lo cual deja a la función de esta manera.

$$-f(x, y) = x^2 + y^2 - 25$$

Los argumentos de la función objetivo deben tener la forma de un vector, y no ser escalares, como lo proporciona el ejemplo. Por ello, se cambian las variables x y y por elementos del vector $\boldsymbol{x}$, x_1 y x_2, respectivamente.

$$-f(\boldsymbol{x}) = x_1^2 + x_2^2 - 25$$

Ahora, se escribe el siguiente código en Matlab para declarar la función objetivo.

```
function z = f(x)
    z = x(1)^2 + x(2)^2 - 25;
end
```

Los primeros dos argumentos por revisar son `fun`, que corresponde a la función objetivo; y `nvars`, que se refiere al número de variables que conforman la función objetivo. En este caso, el código para resolver este problema debería verse así:

```
clc; clear; close all;
fun = @f;
nvars = 2;
x = ga(fun,nvars);
```

Otra forma de escribir el código para ahorrar líneas:

```
clc; clear; close all;
x = ga(@f,2);
```

El resultado para ambos casos es el mismo.

```
Command Window
Optimization terminated: average change in the fitness value less than options.Funct
>> x

x =

    0.0278   -0.0316
```

Note que el resultado no es idéntico al analítico $(0,0)$, pero esta aproximación es bastante buena.

Función con restricciones

Los problemas prácticos de ingeniería conllevan al desarrollo de modelos matemáticos con una serie de restricciones, las cuales pueden ser lineales o no lineales, dependiendo del caso. Debajo se explicará una serie de ejemplos para seguir desentrañando el resto de los argumentos del algoritmo genético.

Restricciones lineales

Ejemplo 2: Hallar el mínimo de la siguiente función objetivo sujeta a las condiciones dadas.

$$\min f(\boldsymbol{x}) = 2x_1^2 - 2x_1x_2 + 2x_2^2 - 6x_1 + 6$$

$$\text{s.t.} \quad g_1(\boldsymbol{x}) = x_1 + x_2 \leq 2$$

Solución: El ejercicio pide minimizar, así que no será necesario escribir el problema dual. Ahora bien, la forma de las restricciones de desigualdad que solicita Matlab es $A\boldsymbol{x} \leq b$, donde A es la matriz de

coeficientes del vector $\boldsymbol{x}$ y b es el vector de constantes. Entonces, se asignan los coeficientes y las constantes como sigue:

$$A = [1\ 1], \qquad b = 2$$

Al igual que el ejemplo anterior, se procede en primer lugar a escribir el código para la función objetivo y luego para el algoritmo genético.

```
function y = myfun(x)
    y = 2*x(1)^2 - 2*x(1)*x(2) + 2*x(2)^2 - 6*x(1) + 6;
end
```

```
clc; clear; close all;
%% Restricciones de desigualdad
A = [1 1]; % Matriz de coeficientes
b = 2; % Constantes
x = ga(@myfun,2,A,b);
```

El resultado se muestra a continuación junto a una gráfica de contorno del problema.

```
Command Window
Optimization terminated: average change in the fitness value less than options.Funct
>> x

x =

    1.4962    0.5048
```

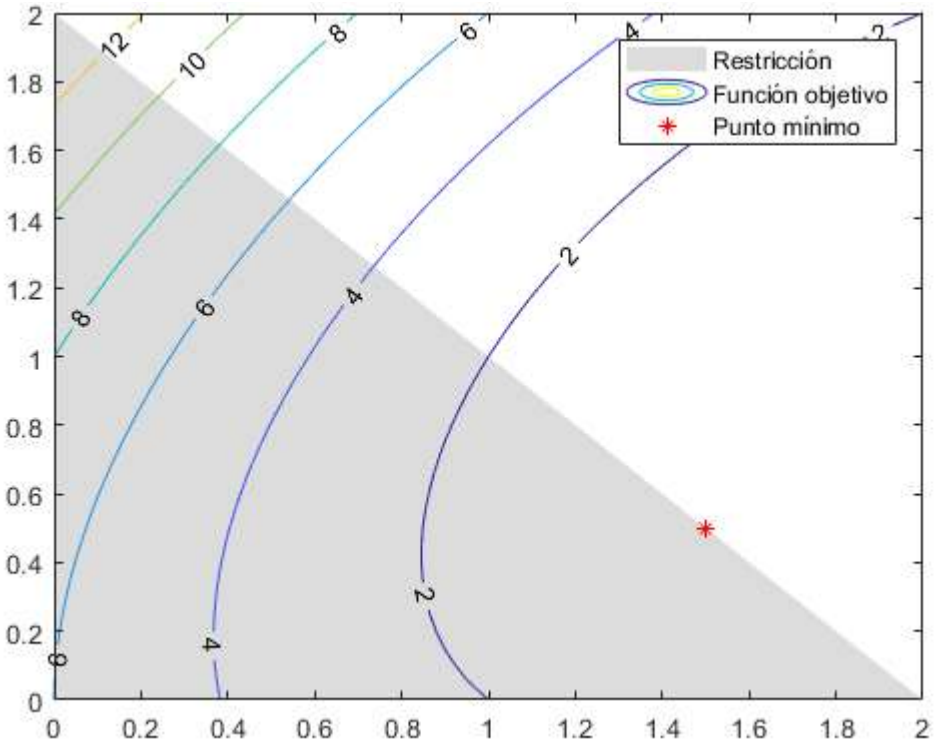

Figura 2. Gráfica de contorno del ejemplo 2.

Ejemplo 3: Resolver el siguiente problema de optimización.

$$\min f(\boldsymbol{x}) = x_1^2 + x_2^2$$

$$\text{s.t.} \quad h_1(\boldsymbol{x}) = 2x_1 + x_2 - 2 = 0$$

Solución: Para empezar, el problema solicita minimizar la función objetivo, por lo que no es necesario escribir el problema dual. Sin embargo, Matlab requiere que las restricciones de igualdad tengan la forma $A_{eq}\boldsymbol{x} = b_{eq}$, donde A_{eq} es la matriz de coeficientes del vector $\boldsymbol{x}$ y b_{eq} es el vector de constantes. Así que lo primero a realizar es cambiar la forma de la restricción a como sigue:

$$2x_1 + x_2 = 2$$

Entonces, los coeficientes y la constante se asignan del modo siguiente:

$$A_{eq} = [2\ 1], \qquad b_{eq} = 2$$

Por su parte, los códigos de Matlab se muestran debajo; en primer lugar, el código de la función objetivo sucedido por el código del algoritmo genético.

```
1   function z = parabola(x)
2 -     z = x(1)^2 + x(2)^2;
3 - end
```

```
1 - clc; clear; close all;
2   %% Restricciones de igualdad
3 - Aeq = [2 1];
4 - beq = 2;
5 - x = ga(@parabola,2,[],[],Aeq,beq);
```

Los corchetes puestos en lo que debería ser `A` y `b` denotan que no hay restricciones de desigualdad. Cuando se tope con un problema en el que no haya alguna de estas restricciones, pero tenga que seguir añadiendo argumentos recuerde agregar estos corchetes para cada argumento vacío.

Los resultados son los siguientes. Al igual que el ejemplo anterior, se hizo una gráfica de contorno para mostrar el punto mínimo.

```
Command Window
Optimization terminated: average change in the fitness value less than options.Funct
>> x

x =

    0.8003    0.3984
```

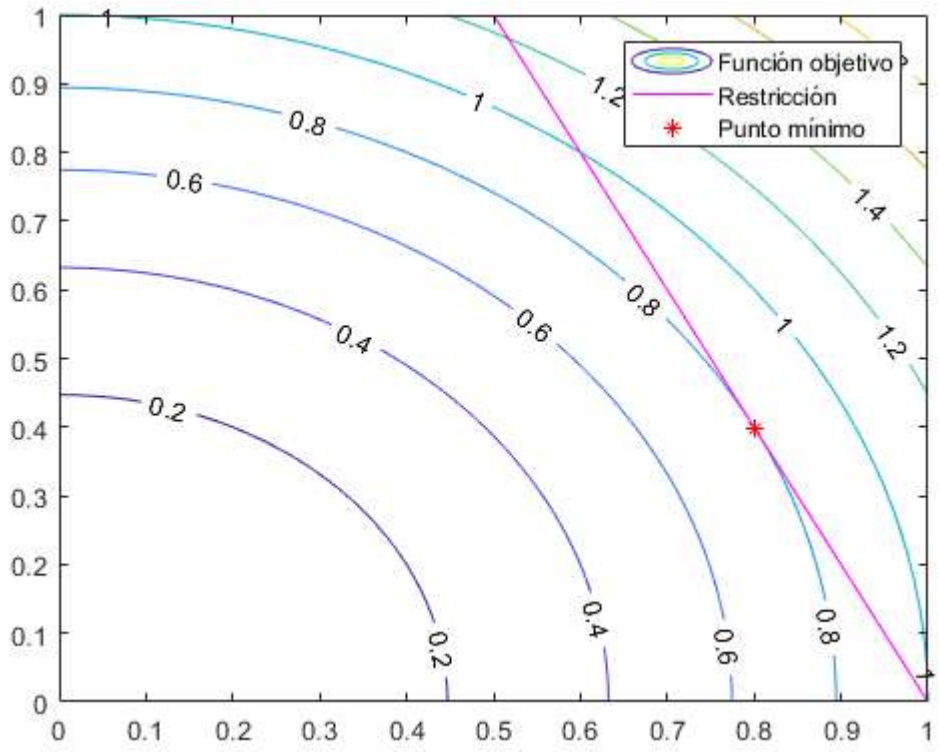

Figura 3. Gráfica de contorno del ejemplo 3.

Restricciones no lineales

Ejemplo 4: Se desea diseñar una lata de refresco, tal que contenga al menos 400 mL al menor costo por material. Por experiencia se sabe que el diámetro debe oscilar entre 3.5 y 8 cm; y la altura entre 8 y 18 cm. Encuentre las dimensiones que minimicen el costo por material.

Solución: Se trata de un problema geométrico. La lata puede aproximarse a un cilindro, donde la superficie de éste se describe con la función

$$S = 2A_b + A_c = 2 \cdot \frac{\pi}{4} D^2 + \pi D h = \frac{\pi}{2} D^2 + \pi D h$$

Por tanto, S es la función objetivo. La única restricción es que el volumen del cilindro sea mayor o igual a 400 mL, matemáticamente:

$$V = A_b h = \frac{\pi}{4} D^2 h \geq 400$$

Además, los límites de las variables son $3.5 \leq D \leq 8$ y $8 \leq h \leq 18$.

Entonces, la estructura matemática de este problema de optimización es como sigue:

$$\min f(D, h) = \frac{\pi}{2} D^2 + \pi D h$$

$$\text{s.t.} \quad \frac{\pi}{4} D^2 h \geq 400$$

$$5 \leq D \leq 8, \qquad 8 \leq h \leq 18$$

Se sugiere que el programa contenga la asignación de variables $D = x_1$ y $h = x_2$ para facilitar la escritura del código y cumplir con la vectorización de la función objetivo.

El argumento que corresponde a la función de restricciones no lineales es `nonlcon`, el cual constituye una función; y está compuesto por dos partes, las restricciones de desigualdad (`c`) y las restricciones de igualdad (`ceq`). La forma que Matlab espera para las restricciones de desigualdad es $c = g(\boldsymbol{x}) \leq 0$; y en el caso de las restricciones de igualdad, la forma es $c_{eq} = h(\boldsymbol{x}) = 0$.

Se requiere escribir un código para la función objetivo y otro más para la función de las restricciones no lineales, pues, a diferencia de las restricciones lineales, éstas primeras son tratadas de forma diferente por el algoritmo genético. Puesto que la única restricción que tiene el problema es una desigualdad, la forma de que debe presentar es la siguiente:

$$c = 400 - V \leq 0, \qquad V = \frac{\pi}{4} D^2 h$$

El código para la función objetivo es el mostrado a continuación.

```
function C = costo(x)
    D = x(1); % Asignación de variables
    h = x(2);
    C = pi*D*h + pi*D^2/2; % Función objetivo
end
```

Por su parte, el código para las restricciones es el que aparece abajo. El primer resultado que arroja la función de restricciones corresponde a las restricciones de desigualdad y el segundo corresponde a las restricciones de igualdad. En particular, este problema no tiene restricciones de igualdad, por eso se han fijado con un par de corchetes.

```
function [c,c_eq] = volumen(x)
    D = x(1); % Asignación de variables
    h = x(2);
    V = pi*D^2*h/4; % Volumen del cilindro
    % Funciones de restricción no lineal
    c = 400 - V; % Desigualdad
    c_eq = []; % Igualdad (no hay)
end
```

El problema cuenta con un par de límites, los cuales se fijan en el código de algoritmo genético con los argumentos `LB` y `UB`, donde el primero se refiere al límite inferior de cada variable y el segundo al límite superior, respectivamente.

$$LB = [3.5\ 8], \qquad UB = [8\ 18]$$

Así, para el problema en cuestión, éste es el código.

```
clc; clear; close all;
%% Restricciones no lineales
LB = [3.5 8]; % Límites inferiores
UB = [8 18]; % Límites superiores
% Los límites van antes que las restricciones no lineales
x = ga(@costo,2,[],[],[],[],LB,UB,@volumen);
```

Se escribieron unas líneas para reportar los resultados de la optimización en pantalla.

```
%% Solución
D = x(1); % Asignación de variables
h = x(2);
C_min = costo(x); % Costo material mínimo
V_lata = pi*D^2*h/4; % Volumen correspondiente
% Reporte de resultados en pantalla
fprintf('Se fabricó una lata con las siguientes especificaciones\n');
fprintf('Diámetro: %.1f cm\n',D);
fprintf('Altura: %.1f cm\n',h);
fprintf('Volumen: %0.f mL\n',V_lata);
fprintf('Costo por material: %.2f cm^2 por unidad',C_min);
```

La pantalla de resultados luce así.

```
Command Window
Optimization terminated: average change in the fitness value less than options.Funct
 and constraint violation is less than options.ConstraintTolerance.
Se fabricó una lata con las siguientes especificaciones
Diámetro: 7.5 cm
Altura: 9.1 cm
Volumen: 400 mL
Costo por material: 301.73 cm^2 por unidad>>
```

Debajo se encuentran superpuestas las gráficas de contorno, tanto para la función objetivo como para la restricción no lineal; donde se indica la posición del punto mínimo.

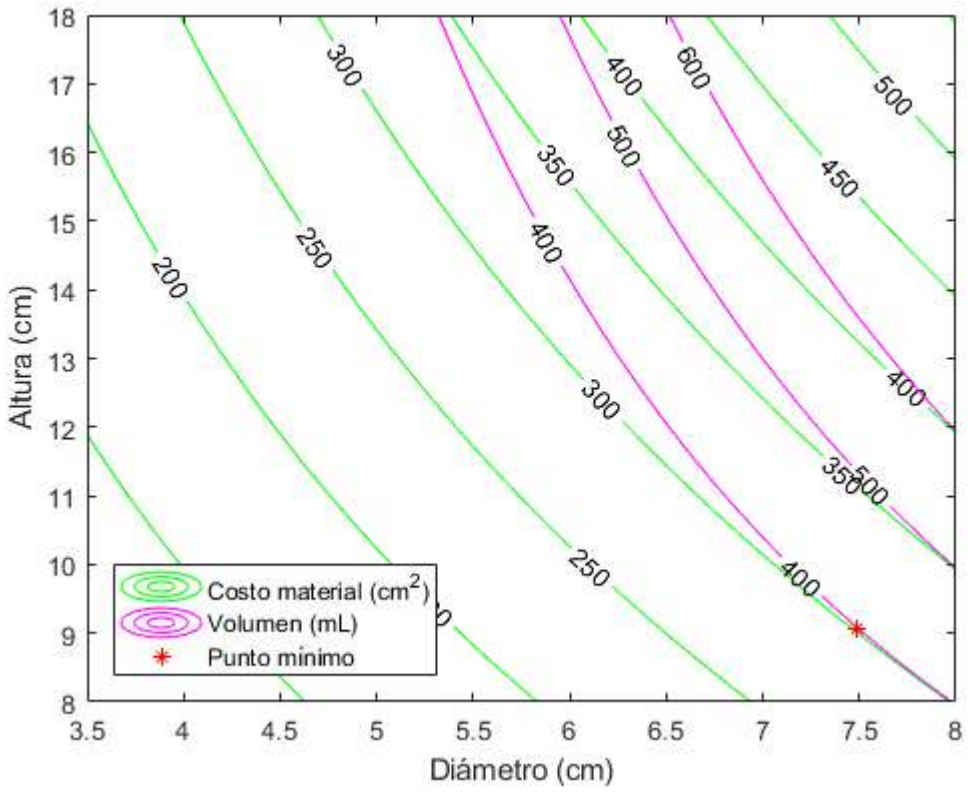

Figura 4. Gráfica de contorno para el ejemplo 4.

Conclusiones

El algoritmo genético es una herramienta de búsqueda muy poderosa, pues es capaz de resolver un problema de optimización con suma facilidad debido a su naturaleza estocástica, puede resolver incluso aquellos problemas de programación no lineal (NLP) que suponen un desafío como para resolverse analíticamente, dado el caso que pudieran plantearse.

Los argumentos de la función `ga` (algoritmo genético) en Matlab se encuentran ordenados de forma bastante intuitiva, aunque, por la falta de familiaridad con la función pudiera resultar un tanto confuso utilizarla al principio. He aquí el propósito de esta guía, donde se ha explicado paso a paso la declaración de cada uno de sus argumentos, facilitando al usuario la correcta aplicación de esta función. Debajo se integran en la Tabla 1 los argumentos que requiere la función `ga`.

Tabla 1. Argumentos de la función `ga` *en Matlab.*

fun	Aquí va la función objetivo.
nvars	El número de variables que tiene la función objetivo.
A	Matriz de coeficientes de las restricciones de desigualdad.
b	Vector de constantes a la derecha de las restricciones de desigualdad.
Aeq	Matriz de coeficientes de las restricciones de igualdad.
beq	Vector de constantes a la derecha de las restricciones de igualdad.
LB	Vector de límite inferior de las variables.
UB	Vector de límite superior de las variables.
nonlcon	Aquí va la función de restricciones no lineales, de igualdad y desigualdad.