

Métodos de Inteligencia Artificial

Reporte 6

IF698972

Josefina Esmeralda Arriaga Hernández

16 de febrero del 2017 Guadalajara, Jalisco

**Objetivo**

Comprender el funcionamiento de la optimización por enjambre de partículas y sus diferentes variaciones ya sea con dos o tres variables.

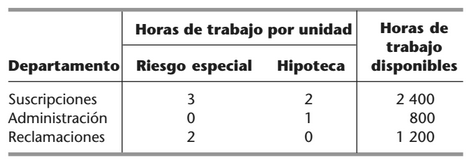
**Problema a resolver**

Se tiene tres problemas el primer tiene una función Z = 2x1 + x2 que se va a maximizar teniendo como restricciones x2 ≤10, 3x1 + x2 ≤ 44, x1 + x2 ≤ 18, 2x1 + 5x2 ≤ 60, x1 ≥ 0 y x2 ≥ 0.

El segundo problema es:

1. La compañía de seguros Primo está en proceso de introducir dos nuevas líneas de productos: seguro de riesgo especial e hipotecas. La ganancia esperada es de $5 por el seguro de riesgo especial y de $2 por unidad de hipoteca.

La administración desea establecer las cuotas de venta de las nuevas líneas para maximizar la ganancia total esperada. Los requerimientos de trabajo son los siguientes:



Este ejercicio se resolverá con algoritmo genético; teniendo como modelo de programación lineal: Z = 5x1 + 2x2 con las restricciones 3x1 +2 x2 ≤ 2400, x2 ≤ 800 y 2x1 ≤ 1200, x1 ≥ 0 y x2 ≥ 0.

Y por último un problema de programación no lineal teniendo como función Z = x14+2x12+2x1x2+4x22 con las restricciones 2x1 +1 x2 ≥ 10, x1 + 2x2 ≥ 10 y x1 ≥ 0, x2 ≥ 0.

Los pasos para crear el algoritmo de enjambre de particulas son:

1. Inicializar aleatoriamente la población de individuos (*swarm*) con velocidad y posición iniciales.
2. Repetir el siguiente proceso iterativo hasta que un criterio de parada se cumpla:
   1. Evaluar la aptitud de cada partícula
   2. Determinar las soluciones mejor local, para cada partícula, y mejor global
   3. Actualizar la posición y la velocidad de las partículas dadas las soluciones en b)
   4. Regresa la partícula mejor evaluada

Los pasos para crear el algoritmo genético son:

1. Generar la población.
2. Se evalúa los pobladores.
3. Se selecciona los progenitores.
4. Se realiza el cruzamiento para obtener hijos.
5. Se muta a hijos aleatorios.
6. Se crea una iteración para llegar al resultado deseado.
7. Por último, se promedia el eje y y se busca el mínimo.

**Código desarrollado**

**PSO**

clear all;

close all;

clc;

np= 200; % N?mero de part?culas

x1p=rand(np,1); % Define la posici?n inicial

x1pg=0; % Valor inicial del mejor global

fxpg=10000000; % Comparas el inicial global con este

x1pl=x1p; % Valores iniciales de los mejores locales

fxpl=ones(np,1)\*fxpg;

vx1=zeros(np,1); % Velocidad inicial del enjambre

x2p=rand(np,1);

x2pg=0;

x2pl=x2p;

vx2=zeros(np,1);

% x3p=rand(np,1)+8

% x3pg=0;

% x3pl=x3p;

% vx3=zeros(np,1);

c1=0.1; % Velocidad de convergencia al mejor global

c2=0.1; % Velocidad de convergencia al mejor local

% %visualizar la funcion esto solo fue por 2D

% x=0:0.01:2;%vector de 0.1 a 10 con tama?o de paso de 0.1

% %y=x.^2;

% %y = 10+x.^2-10\*cos(2\*pi\*x);

% y = -( x.^2.\*(sin(10\*x\*pi)).^2);

%PSO

a=-10000;% Max es negativo min es positivo

for k=1:10000

    %Evaluar el desempe?o

    %fx = x1p.^2;

    %fx = 10+x1p.^2-10\*cos(2\*pi\*x1p);

    %fx = -( x1p.^2.\*(sin(10\*x1p\*pi)).^2);

    %fx = 20+x1p.^2+x2p.^2-10\*cos(2\*x1p\*pi)-10\*cos(2\*x2p\*pi);

    %fx=30+x1p.^2+x2p.^2+x3p.^2-10\*cos(2\*x1p\*pi)-10\*cos(2\*x2p\*pi)-10\*cos(2\*x3p\*pi);

    %fx= -20\*exp(-.2\*sqrt(.5\*(x1p.^2+x2p.^2+x3p.^2)))-exp(.5\*(cos(2\*x1p\*pi)+cos(2\*x2p\*pi)+cos(2\*x3p\*pi)))+exp(1)+20

    %fx = -(3\*x1p+2\*x2p+a\*max(x1p-4,0)+a\*max(x2p-6,0)+a\*max(3\*x1p+2\*x2p-18,0)+a\*max(-x1p,0)+a\*max(-x2p,0));

    %fx = .4\*x1p+.5\*x2p+a\*max(.3\*x1p+.1\*x2p-2.7,0)+a\*max(.5\*x1p+.5\*x2p-6,0)+a\*max(6-.5\*x1p-.5\*x2p,0)+a\*max(-.6\*x1p-.4\*x2p+6,0)+a\*max(-x1p,0)+a\*max(-x2p,0);

    %fx = -(3\*x1p+5\*x2p+a\*max(x1p-4,0)+a\*max(9\*x1p.^2+5\*x2p.^2-216,0)+a\*max(-x1p,0)+a\*max(-x2p,0));

  %fx = -(2\*x1p+x2p+a\*max(x2p-10,0)+a\*max(3\*x1p+x2p-44,0)+a\*max(x1p+x2p-18,0)+a\*max(2\*x1p+5\*x2p-60,0)+a\*max(-x1p,0)+a\*max(-x2p,0));

    fx = x1p.^4+2\*x1p.^2+2\*x1p.\*x2p+4\*x2p.^2+a\*max(-2\*x1p-x2p+10,0)+a\*max(-x1p-2\*x2p+10,0)+a\*max(-x1p,0)+a\*max(-x2p,0);

    % Mejor global

    [val,ind]=min(fx);

    if val<fxpg  % Comparaci?n

       x1pg=x1p(ind,1); % Actualiza posici?n del mejor global

       x2pg=x2p(ind,1);

       %x3pg=x3p(ind,1);

       fxpg=val; %desempe?o valor comparado

    end

    % Mejores locales

    for p=1:np

        if fx(p,1)<fxpl(p,1) % Funci?n que acabo de evaluar contra mejor local del inicio con signo negativo

            fxpl(p,1)=fx(p,1);

            x1pl(p,1)=x1p(p,1);

            x2pl(p,1)=x2p(p,1);

            %x3pl(p,1)=x3p(p,1);

        end

    end

   %GRAFICADO opcional  para una o dos variables

    plot(x1p,x2p, 'b.', x1pg, x2pg, 'go',0,0,'rx');

    axis([300 600 300 600])

    title(['Mejor global (x1pg)=' num2str(x1pg) 'Mejor global (x1pg)=' num2str(x2pg) ]);

    pause(.1)%velocidad de grafica

    % Ecuaciones de movimiento

    vx1=vx1+c1\*rand()\*(x1pg-x1p)+c2\*rand()\*(x1pl-x1p);

    x1p=x1p+vx1; % Nueva posici?n

    vx2=vx2+c1\*rand()\*(x2pg-x2p)+c2\*rand()\*(x2pl-x2p);

    x2p=x2p+vx2;

%     vx3=vx3+c1\*rand()\*(x3pg-x3p)+c2\*rand()\*(x3pl-x3p);

%     x3p=x3p+vx3;

end

%fx = 3\*x1pg+2\*x2pg

%fx=.4\*x1pg+.5\*x2pg

%fx=2\*x1pg+x2pg

fx=x1pg.^4+2\*x1pg.^2+2\*x1pg.\*x2pg+4\*x2pg.^2

**Algoritmo genético**

clear all;

close all;

clc

np = 32; %N?mero de pobladores

tpaso = 1; %Tama?o de paso

xmin = 0;

xmax = 3;

elementos = (xmax - xmin)/ tpaso; %k

nbits = ceil(log2(elementos)); %ceil redondea hacia arriba los decimales

%Para segunda x

tpaso2 = 1; %Tama?o de paso

xmin2 = 0;

xmax2 = 2;

elementos2 = (xmax2 - xmin2)/ tpaso2; %k

nbits2 = ceil(log2(elementos2)); %ceil redondea hacia arriba los decimales

%Generamos la poblaci?n inicial

x1 = randi([0,2^nbits-1],np,1); %Genera np n?meros aleatorios enteros en una columna

x1real = ((xmax - xmin)/(2^nbits-1))\*x1 + xmin; %Convierte enteros a decimales

%Generamos la poblaci?n inicial

x2 = randi([0,2^nbits2-1],np,1); %Genera np n?meros aleatorios enteros en una columna

x2real = ((xmax2 - xmin2)/(2^nbits2-1))\*x2 + xmin2; %Convierte enteros a decimales

%(funci?n objetivo)

a=10000

for i = 1:100 %N?mero de generaciones

    %y = -((x1real+2\*x2real-7).^2+(2\*x1real+x2real-5).^2);

    %y = -((1.5-x1real+x1real.\*x2real).^2+(2.25-x1real-x1real.\*(x2real).^2).^2+(2.625-x1real-x1real.\*(x2real).^3).^2);

    %y=-3\*x1real+5\*x2real+a\*max(-5\*x1real+7\*x2real+3,0)

    y = 5\*x1real+2\*x2real+a\*max(3\*x1real+2\*x2real-2400,0)+a\*max(x2real-800,0)+a\*max(2\*x1real-1200,0)+a\*max(-x1real,0)+a\*max(-x2real,0);

    yprom(i) = mean(y); %Media del vector y

    cromosoma = [y x1 x1real x2 x2real];

    cromosomaOrd = sortrows(cromosoma,1); %Ordena en funci?n de la columna 1

    %Selecci?n

    padres = cromosomaOrd(np/2+1:np,2);

    padresbin = dec2bin(padres,nbits);

    padres2 = cromosomaOrd(np/2+1:np,4);

    padresbin2 = dec2bin(padres2,nbits2);

    %Cruzamiento

    %Un punto de cruce p

    for k = 1:(np/4)

        %Hijo1

        %p = randi([2 nbits-1]); %Aleatorios enteros del 2 a nbits-1

        p=1;

        hijobin(2\*k-1,:) = [padresbin(2\*k-1,1:p) padresbin(2\*k,p+1:nbits)];

        %Hijo2

        %p = randi([2 nbits-1]);

        p=1;

        hijobin(2\*k,:) = [padresbin(2\*k,1:p) padresbin(2\*k-1,p+1:nbits)];

    end

     for k = 1:(np/4)

        %Hijo1

        %p = randi([2 nbits2-1]); %Aleatorios enteros del 2 a nbits-1

        p=1;

        hijobin2(2\*k-1,:) = [padresbin2(2\*k-1,1:p) padresbin2(2\*k,p+1:nbits2)];

        %Hijo2

        p=1;

        %p = randi([2 nbits-1]);

        hijobin2(2\*k,:) = [padresbin2(2\*k,1:p) padresbin2(2\*k-1,p+1:nbits2)];

    end

    %Mutaci?n

    m = rand();

    if m >= 0.8

        nhijo = randi(np/4);

        bit = randi([1,nbits]);

        if hijobin(nhijo,bit) == '1'

            hijobin(nhijo,bit) = '0';

        else

            hijobin(nhijo,bit) = '1';

        end

    end

      m = rand();

    if m >= 0.8

        nhijo = randi(np/4);

        bit = randi([1,nbits2]);

        if hijobin2(nhijo,bit) == '1'

            hijobin2(nhijo,bit) = '0';

        else

            hijobin2(nhijo,bit) = '1';

        end

    end

    %Convertir a n?meros reales en base a la funci?n objetivo

    hijodec = bin2dec(hijobin);

    hijoreal = ((xmax - xmin)/(2^nbits-1))\*hijodec + xmin;

    %Convertir a n?meros reales en base a la funci?n objetivo

    hijodec2 = bin2dec(hijobin2);

    hijoreal2 = ((xmax2 - xmin2)/(2^nbits2-1))\*hijodec2 + xmin2;

    %Juntamos padres e hijos en el vector de x1

    %Aqu? se eliminan los m?s d?biles

    x1real = [cromosomaOrd(np/2+1:np,3);hijoreal];

    x1 = [cromosomaOrd(np/2+1:np,2);hijodec];

    x2real = [cromosomaOrd(np/2+1:np,5);hijoreal2];

    x2 = [cromosomaOrd(np/2+1:np,4);hijodec2];

end

plot(yprom)

[val,ind] = max(y)

disp(['Resultado: x1= ' num2str(cromosoma(ind,3))...

    ' x2=' num2str(cromosoma(ind,5)) ', Desempe?o = ' num2str(val)])

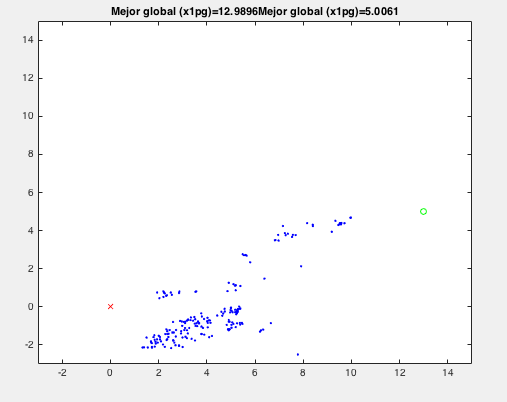
[val,ind]=max(y)

x1=cromosoma(ind,3)

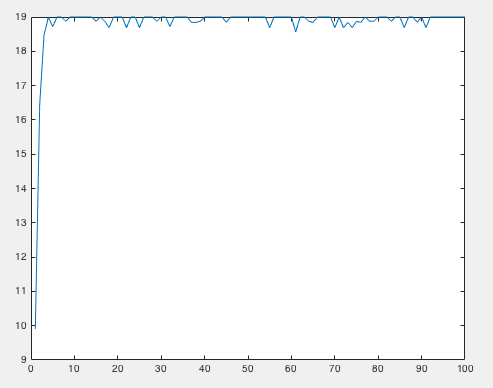
x2=cromosoma(ind,5)

**Gráficos**

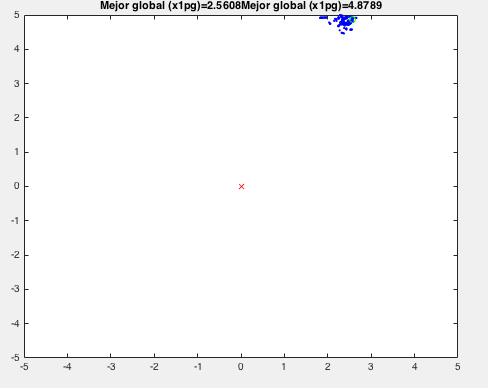
**Primera función**

****

**Segunda función**

****

**Tercera función**

****

**Interpretación de gráficos**

La primera función se graficó entre -3 y 15 para poder observar el acercamiento, en donde se observa que el resultado (punto verde) del gráfico es un acercamiento de x1 tiende a 12 y de x2 a 5 gracias al enjambre con 200 partículas.

La segunda función se observa que se llega el valor esperado de x1 y x2, entonces el promedio del eje x1 y x2 da como resultado el máximo de la función. Si no se tuviera la mutación podría salir súper individuos y los hijos saldrían igual haciendo que se estanque el promedio.

La tercera función se obtuvo a base de la información dada, con las restricciones pertinentes se obtuvo que x1 tiende a 524 y x2 tiende a 413, se graficó entre 300 y 600 para observar el comportamiento del enjambre de partículas.

**Resultados**

El resultado de la primera función es x1=12.8 y x2=5.1 aproximadamente, dado que el acercamiento no se detiene por si solo, se debe forzar a que se termine haciendo que las respuestas no sean del todo exactas.

En la segunda función se utilizó algoritmo genético, teniendo como resultado x1=3 y x2=2, la razón por la que no se usó el enjambre de partículas es porque se deseaba resultados con números enteros, el ejemplo es similar a lo que habíamos realizado en reportes anteriores donde son 32 pobladores que van generando 1000 generaciones, la diferencia es que se agregó restricciones.

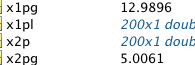
Por último, lo mismo sucedió con la tercera función como con la primera, x1= 484 y x2=473 aproximadamente, se obtiene este resultado con los mejores locales y globales, cuando las partículas se acercan van cambiando la trayectoria hacía el mejor resultado.

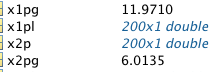
**Conclusiones**

En conclusión, considero que la optimización por enjambre de partículas puede ser exacto con la mayoría de las funciones, pero con ciertas funcionas hace que se cicle Matlab entonces quedan dos opciones o se cancela y se usa ese resultado o esperar a que no se trabe por la indisponibilidad de la computadora. Se aprendió a programar funciones con restricciones en PSO y con el algoritmo genético en el que habíamos trabajado anteriormente. El algoritmo genético es heurístico siendo esto que se obtuvo una buena solución.

**Funcionamiento**

**Primera función**



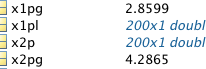


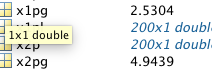
**Segunda función**

****

****

**Tercera función**

****

****