Asignatura: Optimización
Profesor: D. Sc. Gerardo García Gil
Alumno: 2023-A Miguel Angel Luis Espinoza 20110393
Ingeniería en Desarrollo de Software
Centro de Enseñanza Técnica Industrial (CETI)

Aplicación PSO

Presentación

Realizar un programa con el algoritmo de PSO para resolver la siguiente función objetivo:

- funObj = @(xi) D*C + D/xi*S + xi/2*M;

Donde:

D = 5000; C = 5; S = 49; P = 0.2; M = P*C

Utilizando la plataforma de MATLAB.

Introducción

El algoritmo PSO (Particle Swarm Optimization por sus siglas en inglés) es una metaheurística utilizada para resolver problemas de optimización numéricos mediante un proceso de búsqueda en el espacio de soluciones factibles Ω.

La idea principal en la que se basa este algoritmo es en la imitación del comportamiento social de un grupo de ρ individuos denominado enjambre. A los individuos se les denomina partículas y la posición de cada una de ellas corresponde a una solución potencial de un problema de optimización.

Ilustración 1 Analogía del enjambre

PSO es una técnica basada en la población. Utiliza múltiples partículas que forman el enjambre. Cada partícula se refiere a una solución candidata. El conjunto de soluciones candidatas coexiste y coopera simultáneamente. Cada partícula del enjambre vuela en el área de búsqueda, buscando la mejor solución para aterrizar. Entonces, el área de búsqueda es el conjunto de posibles soluciones, y el grupo (enjambre) de partículas voladoras representa las soluciones cambiantes.
A lo largo de las generaciones (iteraciones), cada partícula realiza un seguimiento de su mejor solución personal (óptima), así como de la mejor solución (óptima) del enjambre. Luego, modifica dos parámetros, la velocidad de vuelo (velocidad) y la posición. Específicamente, cada partícula ajusta dinámicamente su velocidad de vuelo en respuesta a su propia experiencia de vuelo y la de sus vecinos. Del mismo modo, intenta cambiar su posición utilizando la información de su posición actual, la velocidad, la distancia entre la posición actual y el óptimo personal, y la posición actual y el óptimo del enjambre.

Modelos matemáticos.
Dos ecuaciones principales están involucradas en el algoritmo PSO. La primera es la ecuación de velocidad, donde cada partícula en el enjambre actualiza su velocidad utilizando los valores calculados de las mejores soluciones individuales y globales y su posición actual.

$$v_i^{t+1} = \underbrace{v_i^t}_{\text{Inertia}} + \underbrace{c_1 r_1 (pbest_i^t - p_i^t)}_{\text{Personal influence}} + \underbrace{c_2 r_2 (gbest^t - p_i^t)}_{\text{Social influence}}$$

Ilustración 2 Ecuación velocidad

La segunda es la ecuación de posición, donde cada partícula actualiza su posición usando la velocidad recién calculada:

$$p_i^{t+1} = p_i^t + v_i^{t+1}$$

Ilustración 3 Ecuación de posición

Los parámetros de posición y velocidad son codependientes, es decir, la velocidad depende de la posición y viceversa. Podemos ilustrar la partícula en movimiento en la siguiente figura:

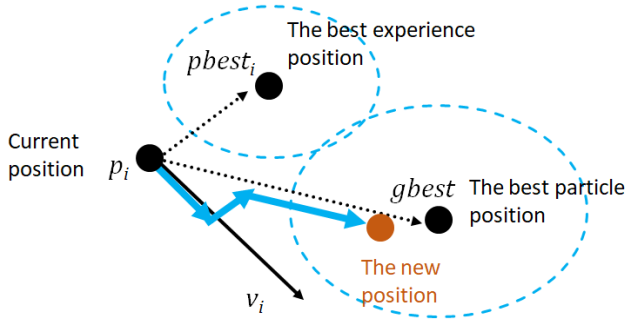

Ilustración 4 Representación de PSO

Las principales variables son:

$S(n) = \{s_1, s_2, \ldots, s_n\}$: un enjambre de $n$partículas
s_i: un individuo en el enjambre con una posición p_iy velocidad $v_i, i \in [|1, n|]$
p_i: la posición de una partículas$_i$
v_i: la velocidad de una partículap$_i$
$pbest_i$: la mejor solución de una partícula
$gbest$: la mejor solución del enjambre (Global)
f: función de fitness
c_1, c_2: constantes de aceleración (parámetros cognitivos y sociales)
r_1, r_2: números aleatorios entre o y 1
t: el número de iteración

Ilustración 5 Variables en fórmulas

Aplicaciones.
Se sabe que PSO es ventajoso en muchos aspectos. En primer lugar, es fácil de implementar. En segundo lugar, no tiene derivados y utiliza muy pocos parámetros. En tercer lugar, tiene un proceso de búsqueda global eficiente. Por eso podemos decir que ha sido una técnica popular explotada para resolver varios problemas de optimización. Profundicemos en algunos ejemplos:

- El entrenamiento de redes neuronales que se utiliza para identificar la enfermedad de Parkinson, extraer reglas de redes difusas o reconocer imágenes.
- La optimización de las redes de distribución de energía eléctrica
- Optimización estructural, donde la industria de la construcción busca la forma, el tamaño y la topología óptimos durante el proceso de diseño
- Identificación de sistemas en biomecánica

Desarrollo
Con este algoritmo se desarrollará un programa con el IDE de MATLAB, donde la función será: @(xi) D*C + D/xi*S + xi/2*M. Se grafica en 3D.

Después de explicar el principio de PSO y su modelo matemático, examinemos los pasos de ejecución de PSO:

1) Inicializa las constantes del algoritmo.
2) Inicialice la solución desde el espacio de solución (valores iniciales para posición y velocidad).
3) Evaluar la aptitud de cada partícula.
4) Actualiza los mejores récords individuales y globales (pbest y gbest).
5) Actualice la velocidad y la posición de cada partícula.
6) Vaya al paso 3 y repita hasta la condición de terminación.

Un diagrama de flujo que detalle y organice los pasos de ejecución puede ayudarnos a comprender el método PSO:

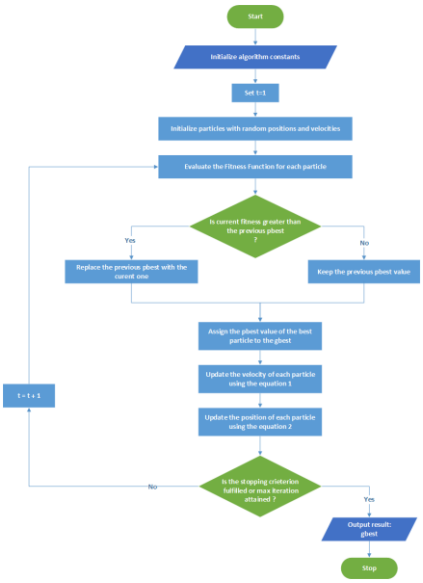

Ilustración 6 Diagrama de flujo algoritmo PSO

Código

```
clear all %clear memory
close all %close matlab windows

D = 5000; %Annual demand
C = 5;    %Cost per unit
S = 49;   %Cost for order
P = 0.2;  %Percent storage cost
M = P*C   %Storage cost

funObj = @(xi) D*C + D/xi*S + xi/2*M;
%Objective function

N = 5; %Particle number
d = 1; % Dimensions
lb = [400]; %Lower limit of search space
ub = [1200]; %Upper limit of search space

k = 0; %iteration
kmax = 150; %Maximun number of iterations
c1 = 2; %Cognitive constant
c2 = 2; %Social constant

%Initialization of particles and velocity
for i=1:N
    x(i,:) = rand(1,d).*(ub-lb)+lb;
%initialization of particles
    v(i,:) = zeros(1,d);            %
Velocity initialization
end

%%Evaluation of the initial particles with
the objective function
for i=1:N
    xi=x(i,:)   %Extraction of the particle
xi
    fx(i,:) = funObj(xi);  %Evaluation of the
particle xi
end

%%Record of the best global particle and the
best local particles
[gfit, ind] = min(fx); %Fitness of the best
global particle
g = x(ind,:); %Location of the best global
particle
fp = fx;      %Fitness of the best local
particle
p = x;        %Position of the best local
particle
axisx = lb:ub; %solution vector
axisy = [];
for i = 1:length(axisx)
    axisy(i) = funObj(axisx(i));
end
%Iterative process
while k < kmax %Stop criterion
k = k+1; %New generation
%%The optimization surface is drawn
figure(1);
%Particles are drawn in red color
plot(x,fx,'o', 'MarkerFaceColor', 'm',
'MarkerSize', 10)
pause(0.3)
%Draw the best local particles in green
plot(p,fp,'o', 'MarkerFaceColor', 'g',
'MarkerSize', 10)
%Pause to allow visualization
pause(0.3)
hold off
%Compute the new velocity for each particle
for i=1:N
    xi = x(i,:); %Extraction of particle xi
    pi = p(i,:); %Extraction of local
particle pi
    v(i,:) = v(i,:)+c1*rand(1,d).*(pi-
xi)+c2*rand(1,d).*(g-xi); %Determination of
the new velocitity for each particle vi
end

%%Determination of the new position of each
particle
x=x+v
for i=1:N
    for j=1:d
        if x(i,j) < lb(j)
            x(i,j) = lb(j);
        elseif x(i,j) > ub(j)
            x(i,j) = ub(j);
        end
    end
end

%%Evaluation of the new particle withc the
objective function
for i=1:N
    xi = x(i,:);
    fx(i,:) = funObj(xi);
end

%Record of the best global particle and the
best local particles
[gfitkplus1,ind] = min(fx);

%if a better solution is found, update the
global particle
%fitness = desempeño
if gfitkplus1 < gfit
    %Update the fitness of the best global
particle
    gfit = gfitkplus1;
    %Update the position of the best global
particle
    g = x(ind,:);
end
for i=1:N
    %update your best local particle
    if fx(i,:)<fp(i,:)
        %Update the fitness of the best local
particles
        fp(i,:) = fx(i,:);
```

```
        %Update the position on the best
local particles
        p(i,:) = x(i,:);
    end
end

%Register the best solutions found in each
generation
Evolution(k) = gfit;
end

figure
plot(Evolution)
disp(['Best Result:',num2str(g)])
disp(['Best Fitness:',num2str(g)])
disp(['Best:',num2str(gfit)]);
```

Resultados

Representación gráfica en 2D, donde las partículas son los puntos morados, y los puntos verdes son los mejores mínimos locales

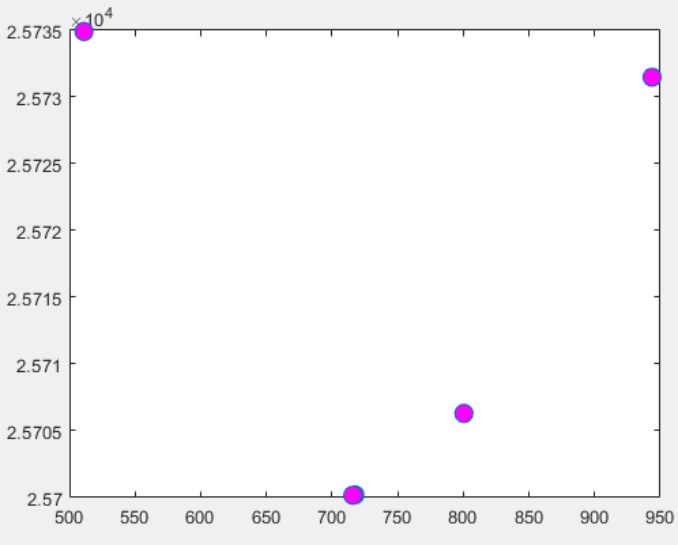

Ilustración 7 Figure 1 partículas

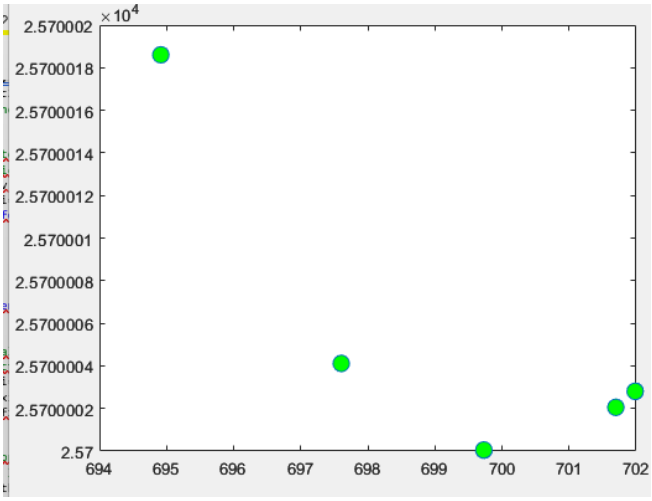

Ilustración 8 Figure 1 mínimos locales

Resultado de la ejecución
Best Result:699.7306
Best Fitness:699.7306
Best:25700.0001
Dándonos en x = 699.7306 y y = 25700.0001

Evolución del proceso.

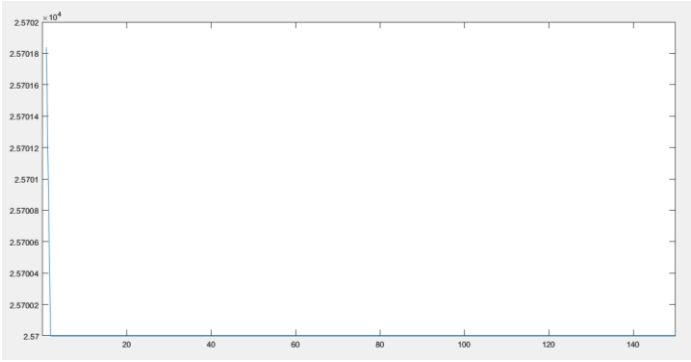

Ilustración 9 Figure 2 en código

Conclusión

El algoritmo PSO es una técnica para optimizar que tiene el principio del comportamiento colectivo de una población de partículas. Lo que lo hace un gran algoritmo es su fácil implementación y que no requiere de matemáticas avanzadas, además que encuentra soluciones óptimas en un tiempo razonable. Además, en este caso particular, el resultado cada vez que ejecute el programa era siempre muy parecido, andaba en los 699 a 701 en X y en Y 25700.

Referencias

1. Baeldung, & Baeldung. (2022). How Does Particle Swarm Optimization Work? | Baeldung on Computer Science. Baeldung on Computer Science. https://www.baeldung.com/cs/pso
2. Alejandro, G. M. R. (2022). Aplicaciones del algoritmo PSO en problemas de identificación paramétrica y sintonización. https://repositorio.cinvestav.mx/handle/cinvestav/3834
3. Nabé, M. (2018, November 6). A tutorial on Optimization Algorithms, the example of Particle Swarm Optimization. Medium. https://medium.com/@mamady94/a-tutorial-on-optimization-algorithms-the-example-of-particle-swarm-optimization-981d883be9d5