

***Tarea 4 Aplicación PSO***

# 

**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.

# Esto es lo que debes hacer si te ataca un enjambre de abejas - Información

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.

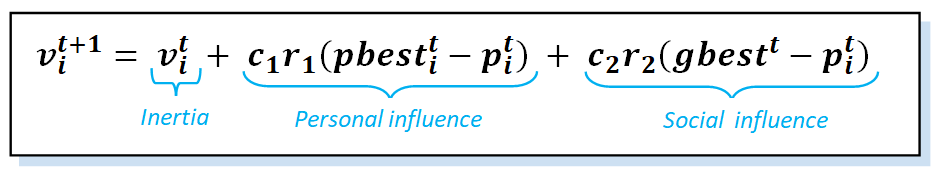


Ilustración 2 Ecuación velocidad

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

# 

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:

# Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto Descripción generada automáticamente

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:

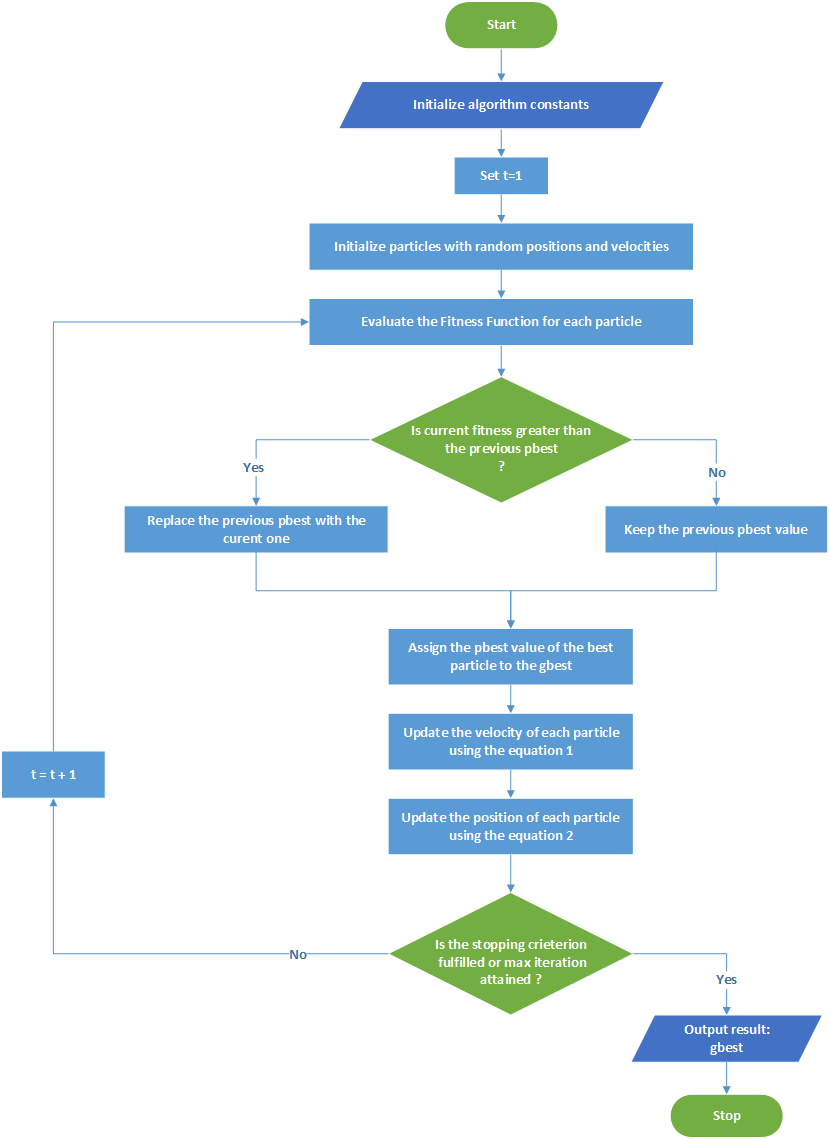


Ilustración 6 Diagrama de flujo algoritmo PSO

**Código**

clear all; close all;

funObj = @(xi) 3\*(1-xi(1))^2\*exp(-(xi(1)^2)-(xi(2)+1)^2)-10\*(xi(1)/5-xi(1)^3 - xi(2)^5) ...

\*exp(-xi(1)^2-xi(2)^2)-1/3\*exp(-(xi(1)+1)^2 -xi(2)^2);

%Configuración de parametros

N=10; %Paricle number

d=2; %Dimensions

lb = [-3 -3]; %Lower limit of search

ub = [3 3];

k = 0;

kmax = 100;

c1 = 2;

c2 = 2;

for i = 1:N

x(i,:) = rand(1,d).\*(ub-lb)+lb;

v(i,:) = zeros(1,d);

end

for i = 1:N

xi=x(i,:);

fx(i,:) = funObj(xi);

end

[gfit, ind] = min(fx);

g = x(ind,:);

fp= fx;

p = x;

axisx=linspace(min(lb), max(ub), 50);

axisy=axisx;

axisz=[];

for i = 1:length(axisx)

for j = 1:length(axisy)

axisz(i,j) = funObj([axisx(i) axisy(j)]);

end

end

[axisy axisx] = meshgrid(axisx,axisy);

while k < kmax

k = k + 1;

figure(1);

surf(axisx,axisy,axisz);

hold on;

plot3(x(:,1),x(:,2),fx+0.1, 'o', 'MarkerFaceColor', 'm', 'MarkerSize', 7);

plot3(p(:,1),p(:,2),fp+0.1, 'o', 'MarkerFaceColor', 'g', 'MarkerSize', 7);

pause (0.1)

hold off

figure(2)

contour(axisx,axisy,axisz,20)

hold on

plot(x(:,1),x(:,2), 'o', 'MarkerFaceColor', 'm');

plot(p(:,1),p(:,2), 'o', 'MarkerFaceColor', 'g');

pause(0.3)

hold off

for i=1:N

xi=x(i,:);

pi=p(i,:);

v(i,:) = v(i,:)+c1\*rand(1,d).\*(pi-xi)+c2\*rand(1,d).\*(g-xi);

end

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

for i=1:N

xi=x(i,:);

fx(i,:)=funObj(xi);

end

[gfitkplus1, ind] = min(fx);

if gfitkplus1<gfit

gfit=gfitkplus1;

g=x(ind,:);

end

for i=1:N

if fx(i,:)<fp(i,:)

fp(i,:)=fx(i,:);

p(i,:)=x(i,:);

end

end

Evolution(k)=gfit

end

figure

plot(Evolution)

disp(['The best Solution :', num2str(g)]);

disp(['The best fitnees : ', num2str(gfit)]);

# Resultados

# Representación gráfica en 3D, donde las partículas son los puntos morados, y los puntos verdes son los mejores mínimos locales

Gráfico, Gráfico de superficie

Descripción generada automáticamente

Ilustración 7 Figure 1 en el código

Diagrama de contorno en 2D, donde se puede apreciar desde una vista desde la parte de arriba donde se encuentran las partículas.

Donde como en la imagen anterior, los puntos morados son partículas y los verdes son los mejores mínimos locales.

Diagrama

Descripción generada automáticamente

Ilustración 8 Figure 2 en el código

Resultado de la ejecución;

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 9 Impresión de pantalla del resultado

Evolución del proceso.

# Gráfico Descripción generada automáticamente

Ilustración 10 Figure 3 del 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. Sin embargo, al compilar el programa varias veces pude encontrar que tiene también un detalle, este se puede quedar estancando en óptimos locales, si las partículas quedan atrapadas en una región subóptima.

# Referencias

# Baeldung, & Baeldung. (2022). How Does Particle Swarm Optimization Work? | Baeldung on Computer Science. Baeldung on Computer Science. <https://www.baeldung.com/cs/pso>

# Alejandro, G. M. R. (2022). Aplicaciones del algoritmo PSO en problemas de identificación paramétrica y sintonización. <https://repositorio.cinvestav.mx/handle/cinvestav/3834>

# 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>