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)

PSO 3D

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

```
@(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)
```

Utilizando la plataforma de MATLAB, y además que grafique la solución en 3D.

Introducción
El algoritmo PSO (Particle Swarm Optimization, por sus siglas en inglés) es un algoritmo de optimización heurística que se utiliza para resolver problemas de optimización global. Fue propuesto por Kennedy y Eberhart en 1995, y está inspirado en el comportamiento de los enjambres de aves y peces.

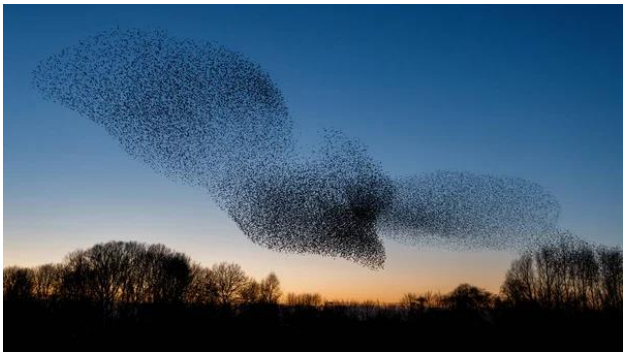

Ilustración 1 Representación gráfica de un enjambre

En PSO, un grupo de partículas (soluciones candidatas) se mueven en un espacio de búsqueda multidimensional, buscando la solución óptima al problema. Cada partícula ajusta su posición y velocidad en función de su experiencia personal y de la experiencia del grupo. De esta manera, las partículas exploran el espacio de búsqueda de manera cooperativa y convergen hacia la solución óptima.

Aplicaciones
El algoritmo PSO se ha utilizado en una variedad de aplicaciones, como en la optimización de redes neuronales artificiales, la planificación de trayectorias en robótica, la optimización de sistemas de energía renovable, la optimización de estructuras y muchos otros campos.

Ejemplos

- El algoritmo PSO se ha utilizado en una amplia variedad de aplicaciones en diferentes campos. Algunos ejemplos de aplicaciones donde se ha utilizado el algoritmo PSO son:

- Optimización de redes neuronales: El algoritmo PSO se ha utilizado para optimizar los pesos y umbrales de una red neuronal, mejorando así su capacidad de aprendizaje y precisión.

- Optimización de sistemas de energía renovable: El algoritmo PSO se ha utilizado para optimizar la operación de sistemas de energía renovable, como la energía eólica y solar, maximizando la producción de energía y minimizando los costos.

- Optimización de estructuras: El algoritmo PSO se ha utilizado para optimizar la geometría y la distribución de materiales en estructuras,

minimizando su peso y maximizando su capacidad de carga.

- Planificación de trayectorias en robótica: El algoritmo PSO se ha utilizado para optimizar la trayectoria de los robots, minimizando el tiempo y la energía necesaria para realizar una tarea determinada.

- Optimización de sistemas de transporte: El algoritmo PSO se ha utilizado para optimizar la adquisición de vehículos en sistemas de transporte, minimizando los costos y los tiempos de espera de los pasajeros.

- Diseño de antenas: El algoritmo PSO se ha utilizado para optimizar el diseño de antenas, maximizando su ancho de banda y su ganancia.

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

El algoritmo PSO tiene varias ventajas y desventajas, que se describe a continuación:

Ventajas:

- Es un algoritmo de optimización global, lo que significa que puede encontrar la solución óptima en un espacio de búsqueda complejo y de alta dimensionalidad.
- Es fácil de implementar y puede ser utilizado para resolver una amplia variedad de problemas de optimización en diferentes campos.
- Es un algoritmo metaheurístico que no requiere información previa sobre el problema, lo que lo hace útil para problemas donde no se dispone de un modelo

A continuación, se describen algunas desventajas del algoritmo PSO:

- Convergencia a óptimos locales: El algoritmo PSO puede converger a óptimos locales en lugar de al óptimo global, especialmente en problemas con múltiples óptimos locales.
- Sensible a la selección de parámetros: El rendimiento del algoritmo PSO puede ser muy sensible a la selección de los parámetros del algoritmo, como la inercia, los pesos cognitivos y sociales, y la velocidad de convergencia.
- Requiere múltiples ejecuciones: Para aumentar la probabilidad de encontrar la solución global óptima, a menudo se requieren múltiples ejecuciones del algoritmo PSO con diferentes parámetros y condiciones iniciales, lo que aumenta el costo computacional y la complejidad del proceso de optimización.
- No garantiza la convergencia: Aunque el algoritmo PSO es capaz de encontrar soluciones óptimas en muchos problemas, no se garantiza la convergencia a una solución óptima en todos los casos.

Desarrollo
Para realizar el código se deben seguir los siguientes pasos.

1. Inicializar la población de partículas aleatoriamente con una posición y velocidad iniciales.
2. Evaluar el valor de aptitud de cada partícula en la población.
3. Establecer las mejores posiciones

individuales y globales iniciales.
4. Mientras no se alcance el criterio de parada (por ejemplo, número máximo de iteraciones, valor de aptitud objetivo alcanzado, etc.) hacer lo siguiente:
4.1 Actualizar la velocidad y posición de cada partícula utilizando las siguientes fórmulas:
Para cada partícula i hacer lo siguiente:
4.1.1 Calcular un número aleatorio r1 y r2 en el rango (0, 1).
4.1.2 Para cada dimensión d de la partícula i, calcular la nueva velocidad v_i,d y la nueva posición x_i,d utilizando las siguientes fórmulas:
4.1.3 Evaluar el valor de aptitud de la nueva posición x_i.
4.1.4 Si la nueva posición x_i es mejor que la mejor posición individual p_i de la partícula i, actualizar p_i = x_i.
4.1.5 Si la nueva posición x_i es mejor que la mejor posición global g, actualizar g = x_i.
5. Devolver la mejor posición global encontrada.

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

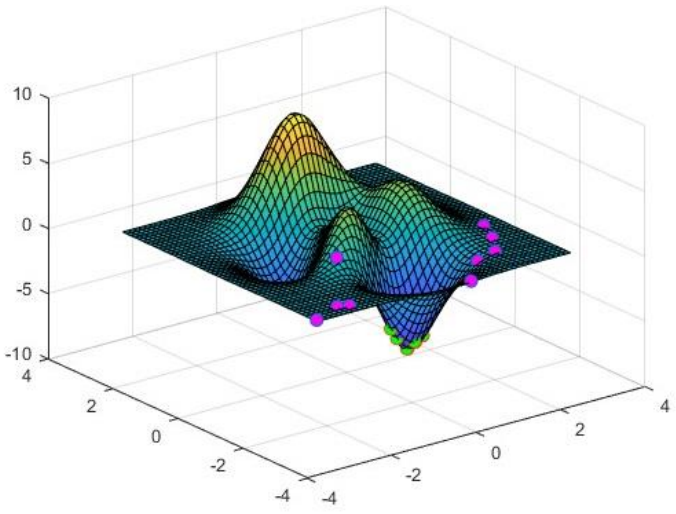

Ilustración 4 Figure 1 en el código

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.

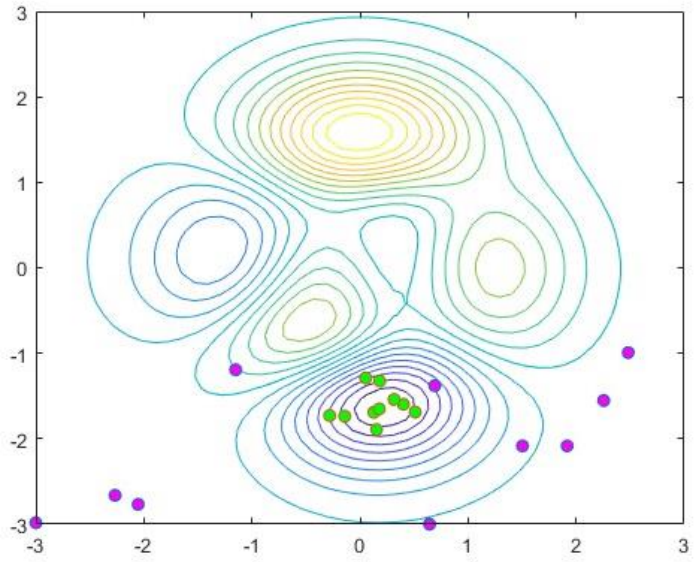

Ilustración 5 Figure 2 en el código

Resultado de la ejecución;

```
The best Solution :0.17846      -1.6506
The best fitnees : -6.5231
```

Ilustración 6 Impresión de pantalla del resultad

Evolución del proceso.

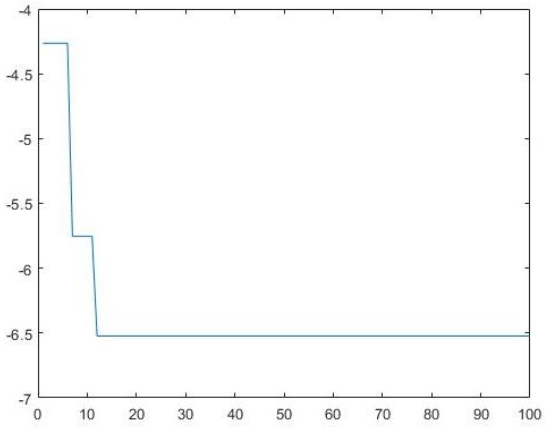

Ilustración 7 Figure 3 del código

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

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