PARTICLE SWARM OPTIMIZATION

Dr. Andrés Espinal Jiménez

Universidad de Guanajuato

Introducción

- **Concebido en 1995 por el doctor en psicología James Kennedy y el doctor en ingeniería Russell C. Eberhart.**
- **Es un algoritmo de inteligencia colectiva que tiene su inspiración en sistemas donde la inteligencia no reside en individuos sino que está distribuida entre un grupo de muchos individuos; como los cardúmenes de peces o las parvadas de pájaros.**

Fundamentos

- **Está basado en la observación que grupos de individuos trabajan juntos para mejorar no solo su desempeño colectivo en alguna tarea, sino también en el desempeño individual.**
- **Algunas ideas en las que se soporta son:**
- Inercia
- Influencia de la sociedad y vecinos

Algoritmo

[Preámbulo]

- **Problema de optimización sobre un dominio continuo de d dimensiones**
- **N soluciones candidatas $x_i, i \in [1, N]$**
- **Cada individuo x_i se mueve con una velocidad v_i a través del espacio de búsqueda**
- **En PSO, un individuo se mueve a través del espacio de búsqueda considerando su inercia y manteniendo cierta velocidad, que puede ser modificada por:**
- El recuerdo de su mejor posición histórica (p_i almacena la posición y $pbest_i$ la aptitud de la i-esíma partícula)
- El conocimiento de la mejor posición de sus vecinos (p_g)

Algoritmo

[Pseudocódigo]

1. **Inicializar una población de N partículas con posiciones y velocidades aleatorias D dimensionales en el espacio de búsqueda**
2. **Ciclo**

- Para todas las partículas evaluar su aptitud
- Para cada partícula
- *Comparar la aptitud actual contra su $pbest_i$, si la aptitud actual es mejor que $pbest_i$, entonces hacer $pbest_i$ igual a la aptitud actual y p_i igual a la posición actual x_i*
- Identificar la partícula en el vecindario con la mejor aptitud y asignar su índice a g
- Cambiar la velocidad y posición de las partículas de acuerdo a:
- $$\begin{cases} \overrightarrow{v_i} \leftarrow \chi\left(\overrightarrow{v_i} + \vec{U}(0,\phi_1) \otimes (\overrightarrow{p_i} - \overrightarrow{x_i}) + \vec{U}(0,\phi_2) \otimes \left(\overrightarrow{p_g} - \overrightarrow{x_i}\right)\right), \\ \overrightarrow{x_i} \leftarrow \overrightarrow{x_i} + \overrightarrow{v_i} \end{cases}$$
- Si el criterio de término es alcanzado salir del ciclo

3. **Fin Ciclo**

Algoritmo

[Consideraciones]

- **La ecuación para actualizar la velocidad se conoce como la *constricción* de Clerc, donde:**
- $\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}$
- $\phi = \phi_1 + \phi_2$, se suele establecer $\phi = 4.1, \phi_1 = \phi_2$, dejando a $\chi = 0.7298$
- **Al inicializar las partículas, se recomienda asignar los valores p_i y $pbest_i$ iguales a x_i y $f(x_i)$; respectivamente.**

Ejemplo

[Continuo]

- **Dada la función esfera bidimensional ($D = 2$):**

$$f(x) = \sum_{d=1}^{2} (x_d)^2 \; ; x \in \mathbb{R}^D$$

- **La evaluación y actualización de la i-ésima partícula en la k-ésima iteración es:**

Componente	x_1	x_2	v_1	v_2	Evaluación
x_i	1.34	0.57	0.10	0.79	$f(x_i) = 2.13$
p_i	0.73	-1.01	-	-	$pbest_i = f(p_i) 1.553$

- **Dado que la posición actual x_i no es mejor que la mejor posición histórica p_i no hay actualización de p_i ni de $pbest_i$**

Ejemplo

[Continuo]

- Suponga que la mejor solución en la población es:

Componente	x_1	x_2	v_1	v_2	Evaluación
p_g	0.57	-0.98	-	-	$f(p_g) = 0.9604$

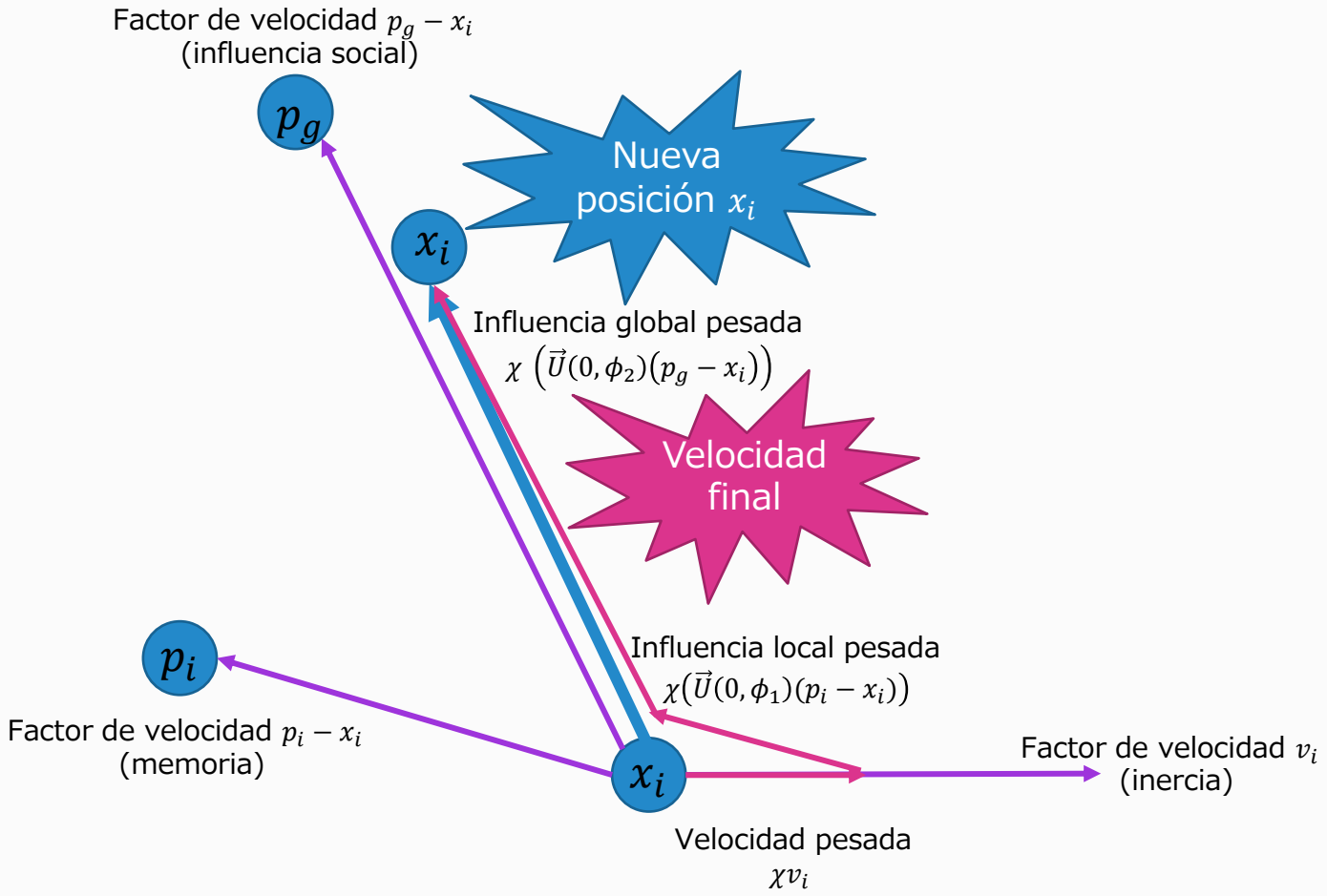
- La actualización de la i-ésima partícula se efectúa de la siguiente manera:

$$\begin{cases} \overrightarrow{v_i} \leftarrow \chi\left(\overrightarrow{v_i} + \vec{U}(0,\phi_1) \otimes (\overrightarrow{p_i} - \overrightarrow{x_i}) + \vec{U}(0,\phi_2) \otimes (\overrightarrow{p_g} - \overrightarrow{x_i})\right), \\ \overrightarrow{x_i} \leftarrow \overrightarrow{x_i} + \overrightarrow{v_i} \end{cases}$$

- $v_i \leftarrow 0.7298([0.10, 0.79] + [1.3, 0.4] \otimes [0.73 - 1.34, -1.01 - 0.57] + [1.9, 1.5] \otimes [0.57 - 1.34, -0.98 - 0.57])$
- $v_i \leftarrow 0.7298([0.10, 0.79] + [1.3, 0.4] \otimes [0.61, -1.58] + [1.9, 1.5] \otimes [-0.77, -1.55])$
- $v_i \leftarrow 0.7298([0.10, 0.79] + [0.79, -0.63] + [-1.46, -2.33])$
- $v_i \leftarrow 0.7298([-0.87, -2.17])$
- $v_i \leftarrow [-0.63, -1.58]$
- $x_i \leftarrow [1.34, 0.57] + [-0.63, -1.58]$
- $x_i \leftarrow [0.71, 1.01]$

Ejemplo

[Continuo]



Partículas binarias

- El algoritmo canónico de PSO puede operar cadenas de bits en lugar de números reales mediante una adaptación sencilla.
- La velocidad es usada como un umbral de probabilidad para determinar si la d-ésima componente de la i-ésima partícula (x_{id}) debe ser evaluada como 0 o 1:

$$s(v_{id}) = \frac{1}{1 + e^{-v_{id}}}$$

- Entonces se genera un número aleatorio r uniformemente distribuido entre 0 y 1 por cada bit en la cadena y se compara contra $s(v_{id})$. Si r es menor que el umbral, entonces x_{id} es interpretado como 1, de lo contrario como 0.

Ejemplo

[Binario]

- **Dada la función One-Max bidimensional ($D = 2$):**

$$f(x) = \sum_{d=1}^{2} x_d \; ; x \in \mathfrak{B}^D$$

- **La evaluación de la i-ésima partícula en la k-ésima iteración es:**

Componente	x_1	x_2	v_1	v_2	$f(x)$
x_i	1	0	5.7	-2.4	2
$s(v_{id})$	-	-	0.99	0.08	-
r	¿ $0.97 < 0.99$?	¿ $0.21 < 0.08$?	0.97	0.21	-

- **Los valores de x no son generados aleatoriamente, son definidos a partir de v y la función sigmoide; incluso al inicializar las partículas.**