Modelos Bioinspirados y Heurísticas de Búsqueda

Actividad Académica Dirigida

Daniel Pérez Rodríguez

Índice

Introducción....3
Funciones a estudiar....3
Función Rosenbrok....3
Función Rastrigin....4
Algoritmos....4
Algoritmo de Optimización Basados en nube de particulas local (Local PSO)....4
Partículas....4
Población inicial....5
Comunicación social....6
Límite del espacio de búsqueda....6
Condición de parada....6
Resultados....7
Algoritmo de Optimización Basados en nube de particulas global (Global PSO)....7
Comunicación social....7
Resultados....7
Búsqueda local....8
Solución inicial....8
Operador de movimiento....8
Condición de parada....9
Resultados....9
Comparación de resultados....9
Como ejecutar la práctica....11

Introducción

El objetivo de la práctica es estudiar y desarrollar los siguientes algoritmos:

- Algoritmos de Optimización Basados en nube de partículas local (Local PSO)
- Algoritmos de Optimización Basados en nube de partículas global (Global PSO)

El código desarrollado deberá encontrar el mínimo valor posible en dos funciones diferentes. La función Rosenbrok y la función Rastrigin. Además, ambos algoritmos deberán ser comparados con un algoritmo de búsqueda local.

Funciones a estudiar

El espacio de búsqueda en ambas funciones esta comprendido entre los siguientes valores de *x* e *y*:

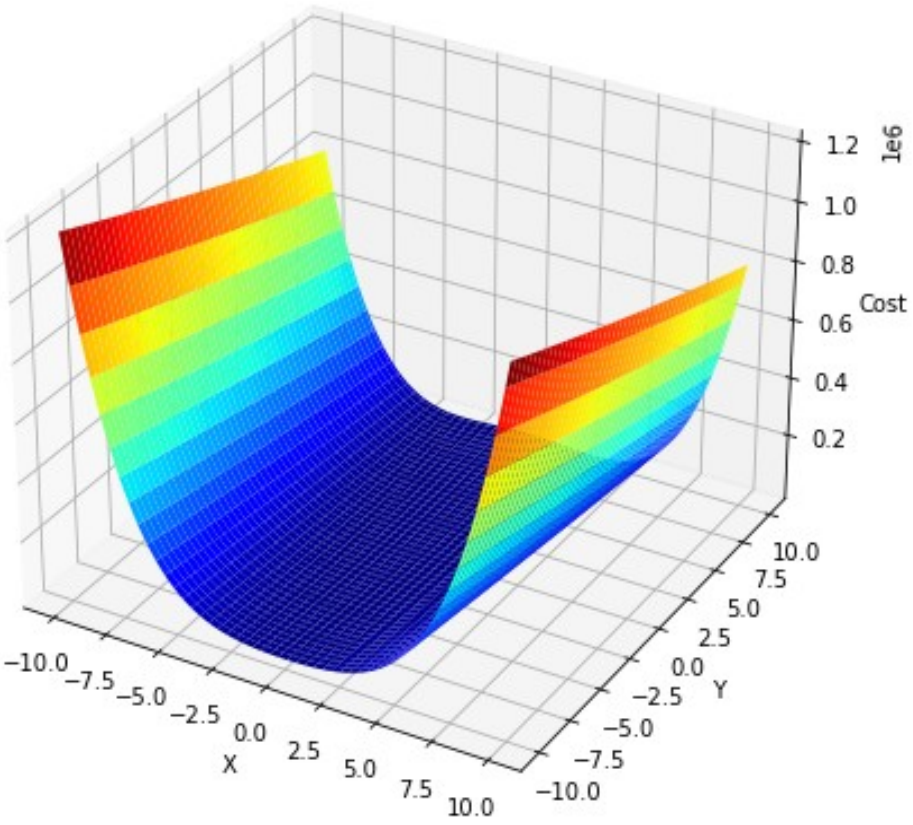
$$x = \pm 10 \,|\, y = \pm 10$$

Función Rosenbrok

La función Rosenbrok está caracterizada por la siguiente fórmula:

$$f(x, y) = (1 - x)^2 + 100 * (y - x^2)^2$$

Representación función Rosenbrok

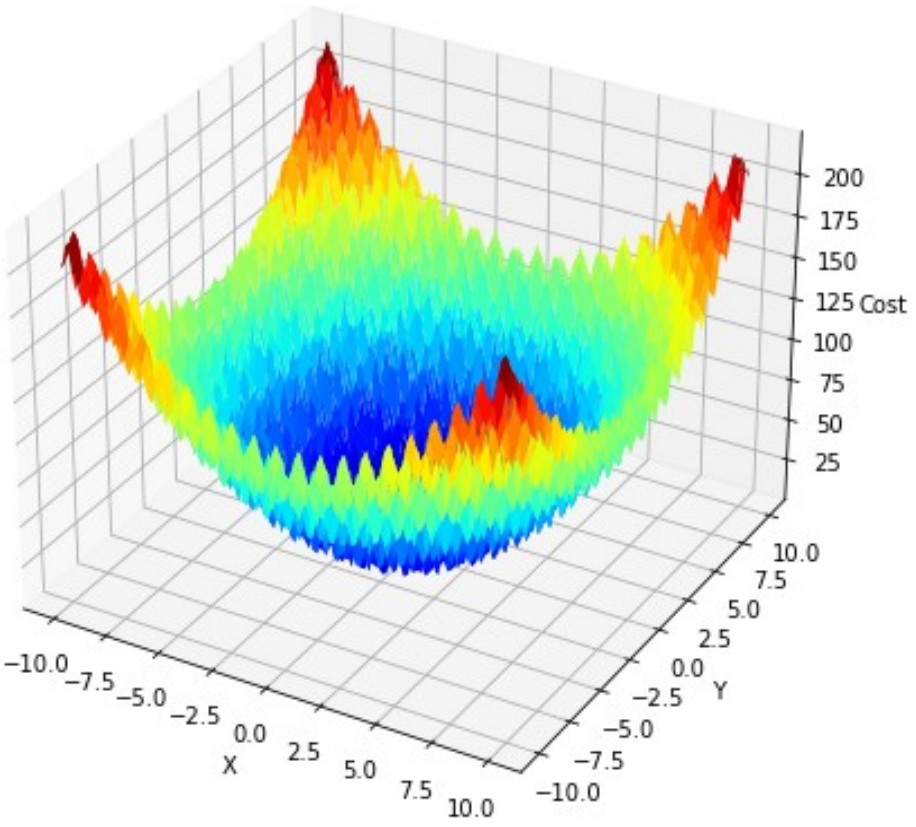


Función Rastrigin

La función Rastrigin está caracterizada por la siguiente fórmula:

$$\text{Ras}(x, y) = 20 + x^2 + y^2 - 10 * (\cos(2 * \pi * x) + \cos(2 * \pi * y))$$

Representación función Rastrigin



Algoritmos

Algoritmo de Optimización Basados en nube de particulas local (Local PSO)

El algoritmo consiste en utilizar un conjunto de partículas(agentes), que se moverán por el espacio de búsqueda y que guardarán la mejor solución que han encontrado mientras se comunican entre ellas(sistema multiagente). El algoritmo PSO presenta las siguientes características: **partículas, población inicial, comunicación social, límite del espacio de búsqueda y condición de parada.**

Partículas

Una partícula es una agente que se desplaza por la función búscando una solución. Para ello se ha creado la clase partícula, que **guarda:**

- **Posición:** Array de dos posiciones que guarda la posición *x* e *y* de la partícula.

- **Velocidad:** Array de dos posiciones que guarda la velocidad en cada eje, *x* e *y*.
- **Fitness:** Variable que almacena el último fitness calculado.
- **Mejor posición:** Array de dos posiciones que guarda la mejor posición encontrada hasta el momento.
- **Mejor fitness:** Variable que almacena el fitness asociado a la mejor posición encontrada.

Para mover una partícula de una posición a otra del espacio, se sumarán el vector posición y el vector velocidad para obtener una nueva posición:

$$P_i = P_i + V_i$$

Una vez calculada la nueva posición, se evalúa ésta. Si el nuevo fitness es mejor que el mejor encontrado previamente, se actualizarán las variables mejor posición y mejor fitness.

Para calcular el valor del vector velocidad se empleará la siguiente función:

$$V_i = w * V_i + \varphi_1 * rnd() * (pBest_i - P_i) + \varphi_2 * rnd() * (gBest_i - P_i)$$

- $\mathbf{P_i}$: Vector posición
- $\mathbf{V_i}$: Vector velocidad
- **w**: Inercia de la partícula. Tendrá valor igual a 0,729
- $\boldsymbol{\varphi_1}$: Factor de aprendizaje cognitivo de la particula, cuanto más grande sea este valor mayor será la influencia de la mejor solución encontrada por la partícula en el movimiento de la misma. Su valor será igual a 1,49445
- $\boldsymbol{\varphi_2}$: Factor de aprendizaje social de la particula, cuanto más grande sea este valor mayor será la influencia que ejercerá el resto de particulas de la población sobre el movimiento de la misma. Su valor será igual a 1,49445
- $\mathbf{pBest_i}$: Mejor vector posición encontrada por la partícula
- $\mathbf{gBest_i}$: Mejor vector posición encontrada por la población. En un PSO Local, este valor será igual a la mejor posición encontrada por los vecinos de la partícula.
- **rnd()**: Valor aleatorio entre 0 y 1.

Población inicial

Se creará una población de 10 partículas. La posición inicial de las mismas será un valor aleatorio comprendido entre `-10` y `+10` para cada eje. De la misma forma, el vector velocidad será un valor aleatorio comprendido entre *-1.0* y *+1.0* para cada eje.

Comunicación social

Como se ha comentado brevemente en el apartado de la partícula, el movimiento de cada una de las mismas estará influenciado por sus partículas vecinas. Para seleccionar el vecindario de cada una, seleccionaremos las partítculas colindantes a izquierda y derecha de la misma en la lista. Utilizaremos una vecindad 2 (cada particula tiene 4 vecinos). Ejemplo.

Población: [0,1,2,3,4,5,6,7,8,9]

Vecindario de la particula 4: [2,3,5,6]

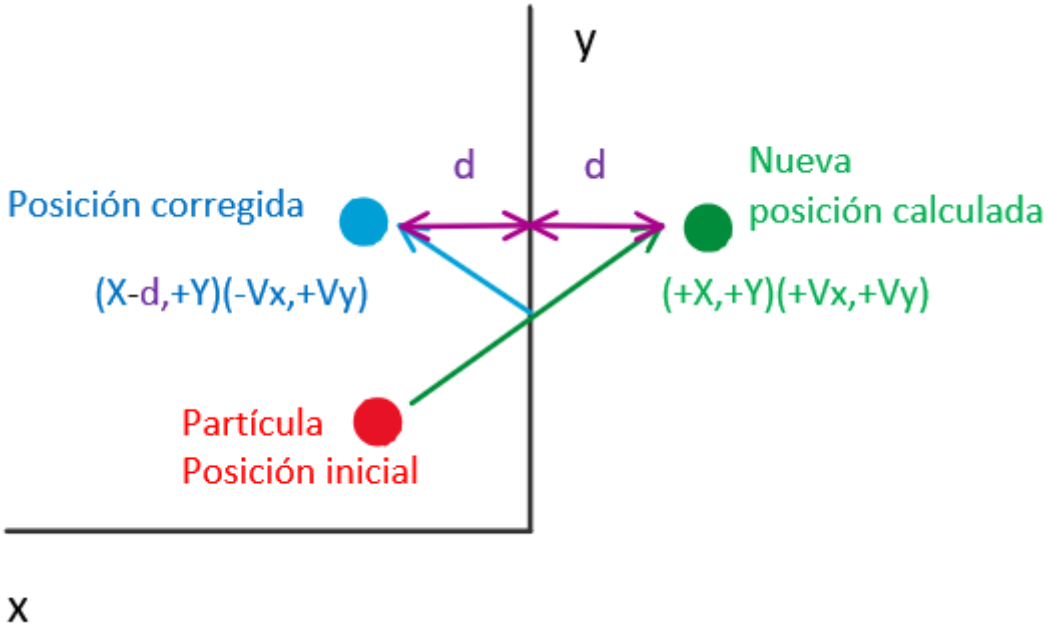
Vecindario de la particula 0: [8,9,1,2]

En el algoritmo, una vez calculado la posición de cada partícula. Escogeremos cual es la mejor posición encontrada por el vecindario de dicho vecindario(gBest). Después evaluaremos su nuevo vector velocidad con el vector posición obtenido.

Límite del espacio de búsqueda

Al recalcular los valores posición de cada partícula, es posible que se salgan del espacio de búsqueda. Por lo tanto, es necesario tratar su comportamiento cuando lleguen al límite.

Sea optado por hacer que las partículas reboten diametralmente contra el límite, actualizando su vector velocidad en el sentido contrario.



Corregiremos su posición restando la distancia respecto al límite e invertiremos el valor de velocidad en dicho eje. Esta comprobación se realizará para el mínimo y máximo de *x* e *y*.

Condición de parada

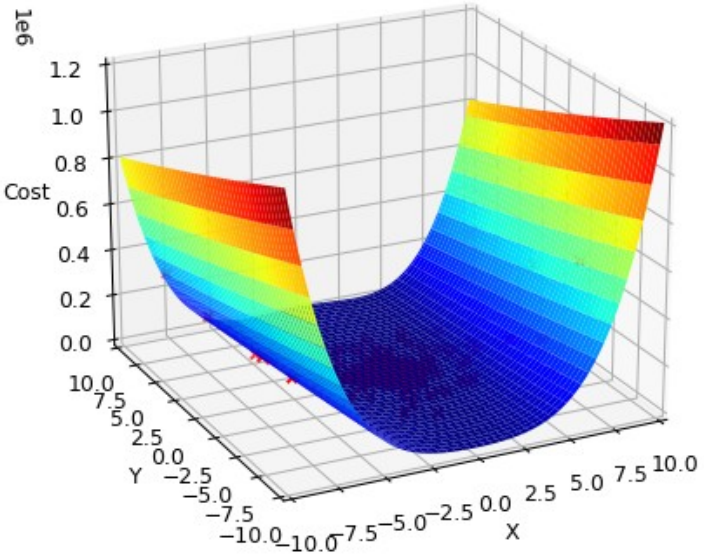
Se ha fijado dos condiciones de parada. La primera consiste en fijar un tiempo máximo de búsqueda (5 minutos). Y la segunda, termina la ejecución cuando encuentra un valor menor o igual a 0.001.

Resultados

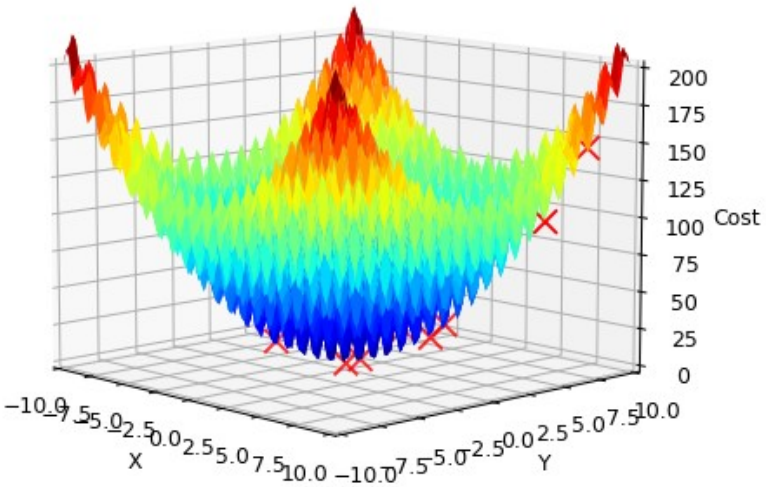
	Rosenbrok Cost	Rosenbrok Ev	Rastrigin Cost	Rastrigin Ev
Local PSO	**0.001**	**2430**	**0.0**	**10**

Soluciones encontradas hasta alcanzar el mínimo

Función Rosenbrok



Función Rastrigin



Podemos observar que el número de evaluaciones realizadas difiere bastante entre una función y otra. Calculando el mínimo de forma mucho más rápida en la función Rastrigin.

Algoritmo de Optimización Basados en nube de particulas global (Global PSO)

El algoritmo Global PSO se basa en todos los elementos del Local PSO modificando uno de sus parámetros. La comunicación social.

Comunicación social

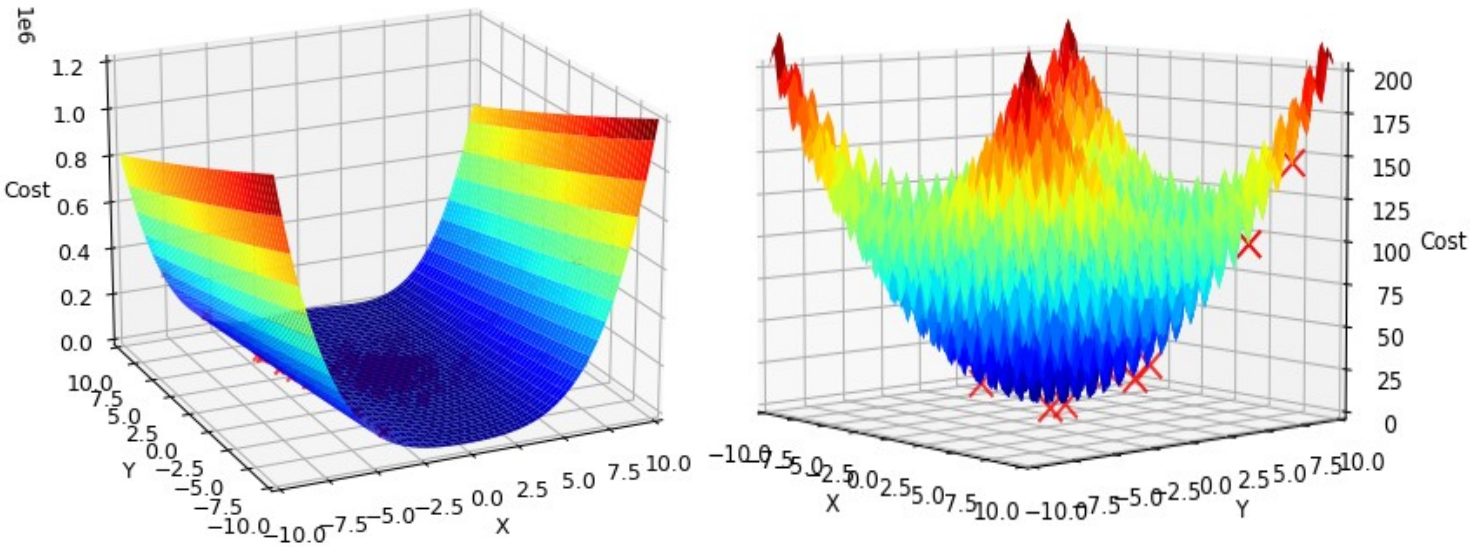
Mientras que en Local PSO, cada partícula tiene un vecindario del cual extraer la posición de mayor influencia. En el Global PSO, la mejor posición encontrada hasta el momento por toda la población será la posición de influencia para cada una de las partículas. Esta posición tomará el valor de la variable gBest en la función para calcular la velocidad.

Resultados

	Rosenbrok Cost	Rosenbrok Ev	Rastrigin Cost	Rastrigin Ev
Global PSO	**0.0007**	**710**	**0.0**	**10**

Soluciones encontradas hasta alcanzar el mínimo

Función Rosenbrok **Función Rastringin**



El comportamiento del algoritmo es similar al Local PSO, pero podemos ver que encuentra la solución de forma más rápida en la función Rosenbrok. Este es debido a que toda la nube de particulas comparte la misma mejor posición encontrada hasta el momento como influencia.

Búsqueda local

La búsqueda local consiste en mejorar la solución inicial buscando soluciones vecinas a la actual. Para ello debemos comentar los siguientes **aspectos del algoritmo: solución inicial, operador de movimiento y condición de parada.**

Solución inicial

La solución inicial será un punto concreto para cada tipo de función a evaluar. Para la función Rosenbrok, el punto inicial es (10.0, 0.0). Para la función Rastrigin, el punto inicial es (5.0, 5.0).

Operador de movimiento

Cada vez que debamos encontrar soluciones vecinas a la solución actual. Crearemos 10 nuevas soluciones a partir de la misma. Para ello, generaremos un nuevo valor de *x* e *y* aumentando o reduciendo de forma aleatoria ese valor en 0.1.

$$x_{new} = rnd\left(x_{actual} - 0{,}1, x_{actual} + 0{,}1\right)$$

$$y_{new} = rnd\left(y_{actual} - 0{,}1, y_{actual} + 0{,}1\right)$$

Condición de parada

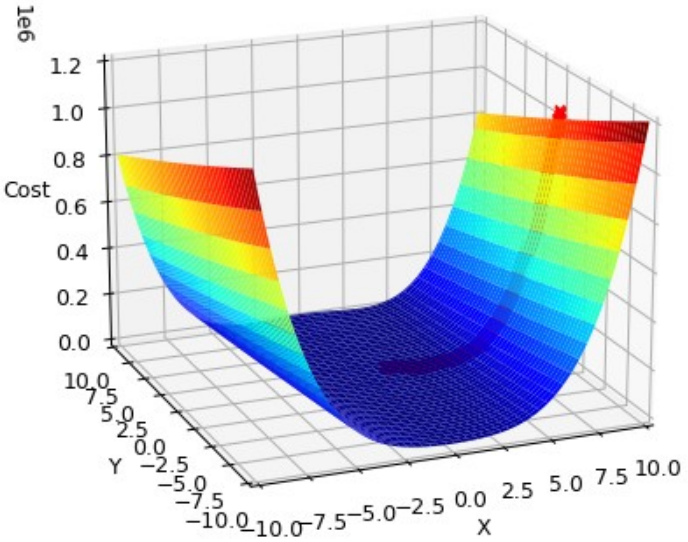
Una vez generado el vecindario, escogeremos la mejor solución encontrada en el mismo. Si dicha solución es mejor que la actual encontrada, seguiremos explorando. El algoritmo finaliza cuando no se ha conseguido mejorar la solución actual.

Resultados

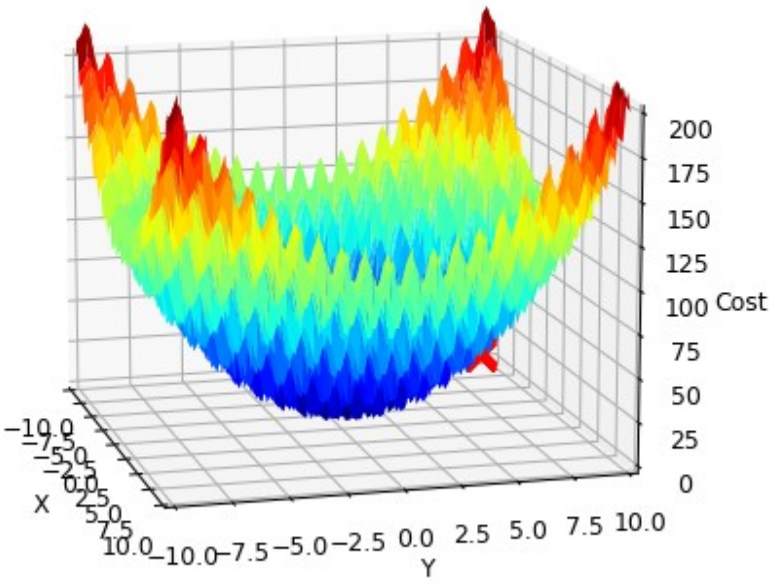
	Rosenbrok Cost	Rosenbrok Ev	Rastrigin Cost	Rastrigin Ev
Local Search	**0.0168**	**1110**	**49.7603**	**20**

Soluciones encontradas hasta alcanzar el mínimo

Función Rosenbrok



Función Rastringin



En la función Rosenbrok se puede observar claramente como el algoritmo ha ido explorando desde la solución inicial hasta alcanzar el mínimo. Por otro lado, en la función Rastrigin el número de evaluaciones es muy pequeño al encontrar un mínimo local de forma prematura.

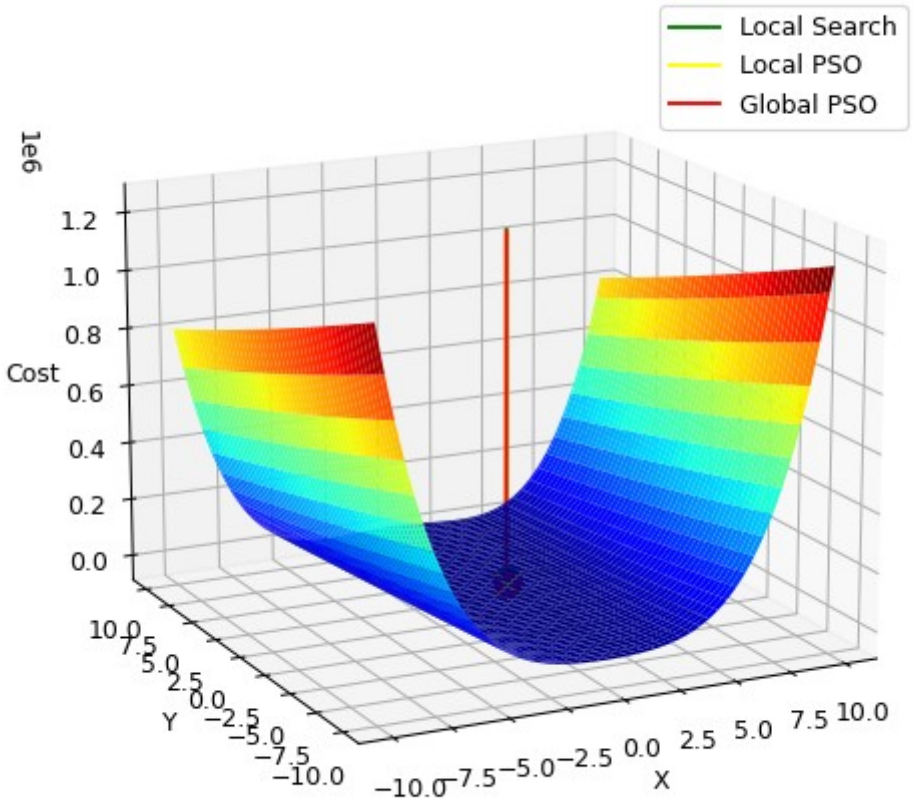
Comparación de resultados

	Rosenbrok Cost	Rosenbrok Ev	Rastrigin Cost	Rastrigin Ev
Local Searh	0.0168	1110	49.7603	20
Local PSO	0.0009	2430	0	10
Global PSO	0.0007	710	0	10
Mean	**0.0062**	**1416.6667**	**16.5867**	**13.3333**
Std	**0.0092**	**900.0740**	**28.7291**	**5.7735**

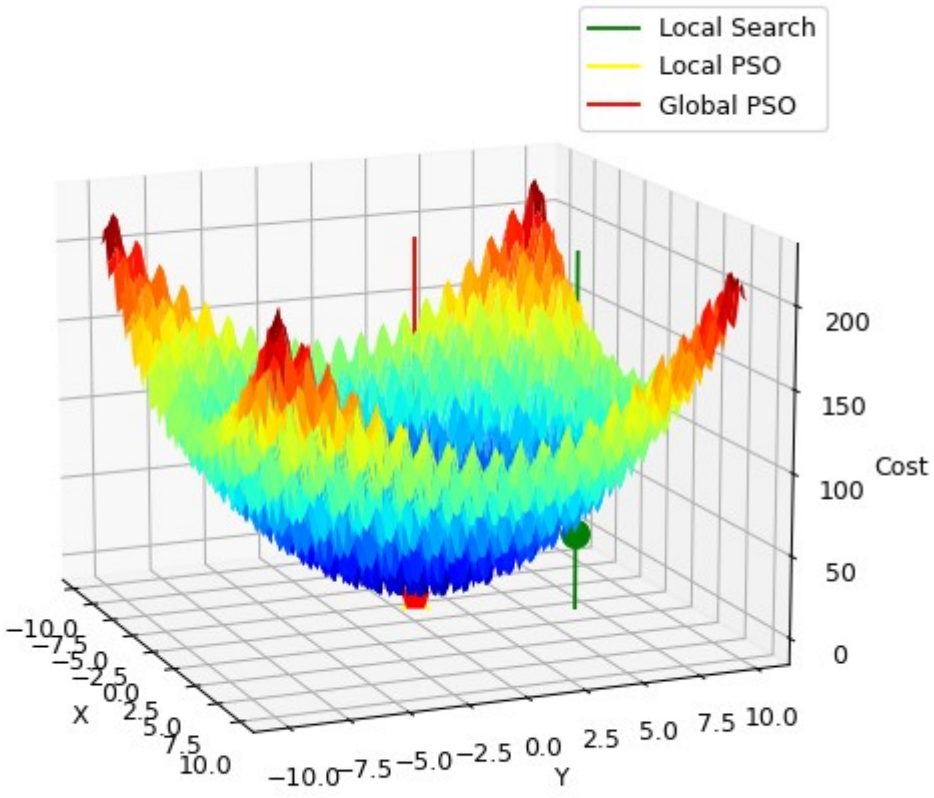
Si nos centramos primero en la función Rosenbrok, observamos que todos los algoritmos son capaces de encontrar el mínimo. Pero con diferencias en el número de evaluaciones. El algoritmo Global PSO resulta ser la mejor opción, gracias a que la población comparte la misma información respecto a la mejor posición encontrada. Por el contario, el Local

PSO tiene problemas para encontrar el mínimo por su método de compartir información entre partículas. Con respecto a la función Rastrigin, podemos observar que la búsqueda local presenta problemas a la hora de encontrar el mínimo. Ya que una vez que encuentra un mínimo local deja de buscar. Los otros dos algoritmos por el contrario, tienen un rendimiento idéntico.

Función Rosenbrok

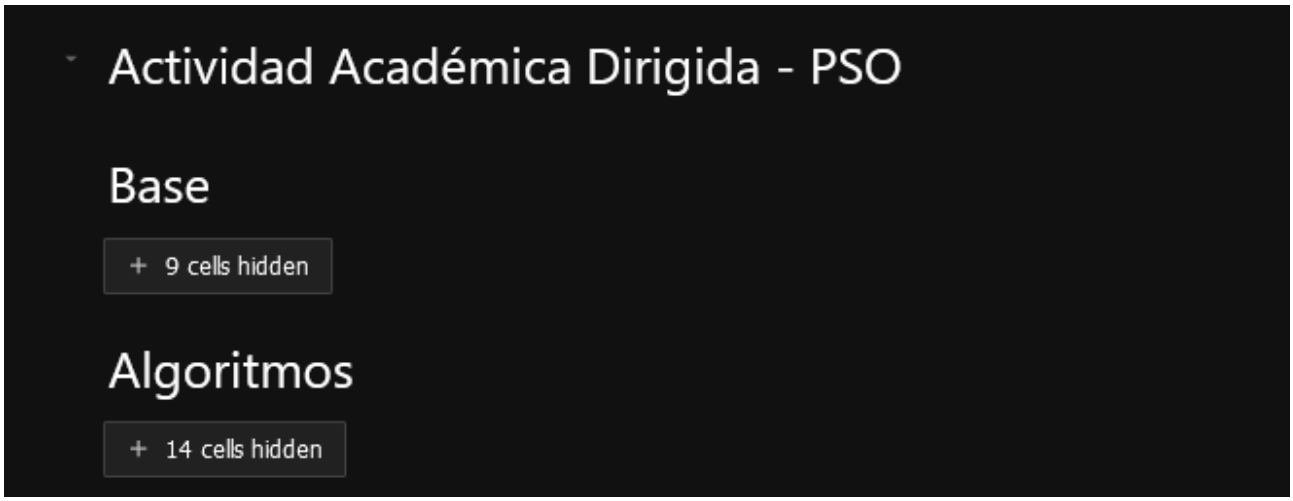


Función Rastrigin



Como ejecutar la práctica

Primero se ejecutarán las secciones *Base* y *Algoritmo.*



Después, en la sección de *Test Base* se especificará la semilla a utilizar. Así como los límites de la función. Una vez definido, se ejecutarán los algoritmos.

Finalmente, se ejecutará la sección *Resultados* para ver los resultados obtenidos