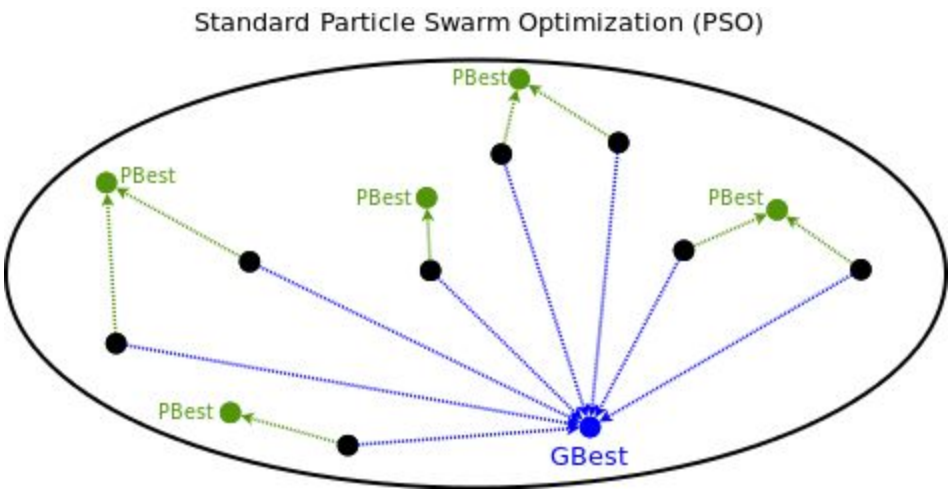
Diego Antonio Vallejo Jimenez

211573851

Algoritmo enjambre de Partículas (PSO)

SSP. Inteligencia Artificial 1

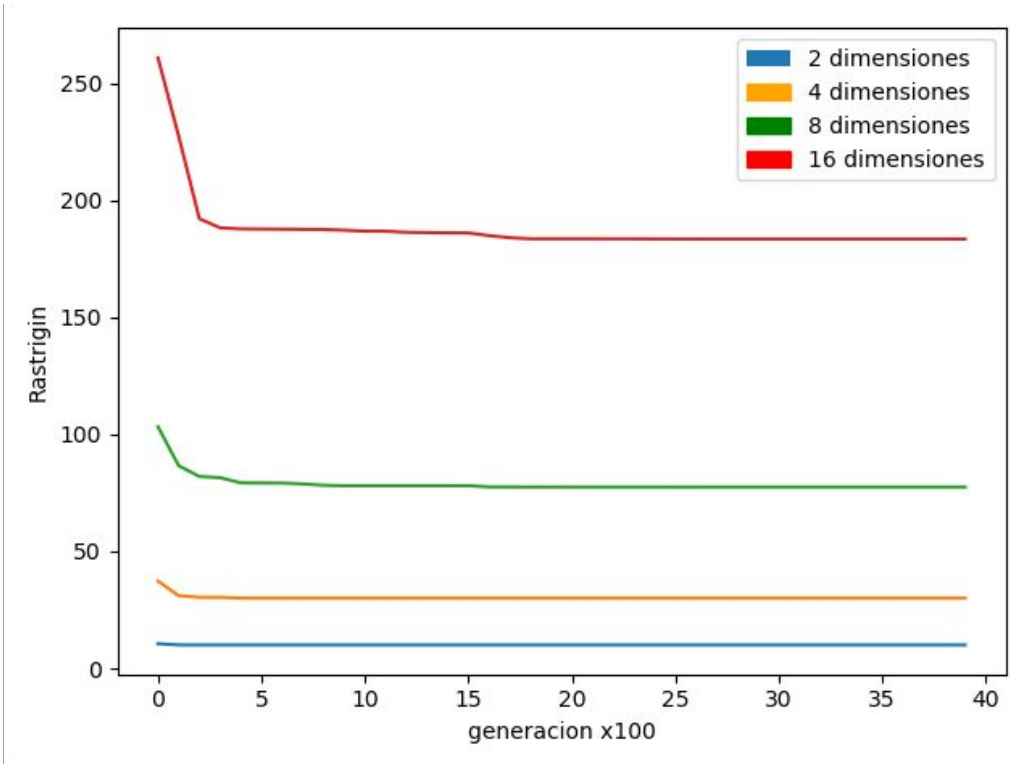


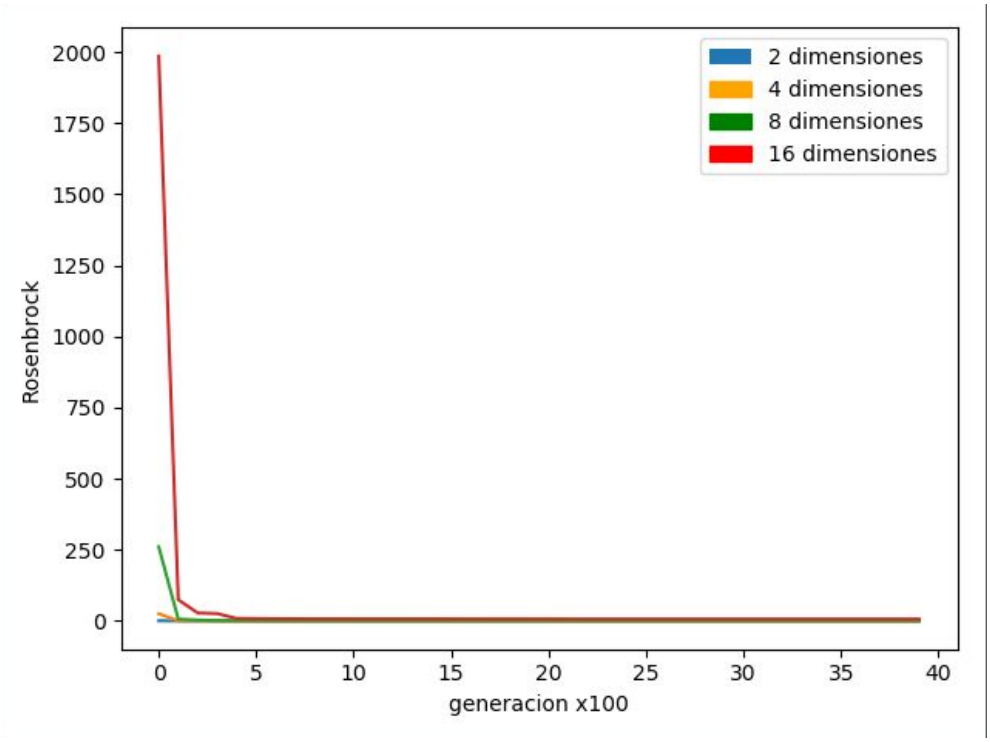
La **Optimización por Enjambres de Partículas** (conocida como **PSO**, por sus siglas en inglés, **Particle Swarm Optimization**) es una técnica de optimización/búsqueda. Aunque normalmente el PSO se usa en espacios de búsqueda con muchas dimensiones, los ejemplos que se mostrarán aquí harán uso de un espacio bidimensional, con el fin de facilitar la visualización, y porque nuestro objetivo es puramente didáctico, esperando que el interesado no encuentre dificultades en extenderlo a otros casos.

Este método fue descrito alrededor de 1995 por James Kennedy y Russell C. Eberhart (Kennedy, J. & Eberhart, R. (1995), 'Particle swarm optimization', Neural Networks, 1995. Proceedings., IEEE International Conference), y se inspira en el comportamiento de los enjambres de insectos en la naturaleza. En concreto, podemos pensar en un enjambre de abejas, ya que éstas a la hora de buscar polen buscan la región del espacio en la que existe más densidad de flores, porque la probabilidad de que haya polen es mayor. La misma idea fue trasladada al campo de la computación en forma de algoritmo y se emplea en la actualidad en la optimización de distintos tipos de sistemas.

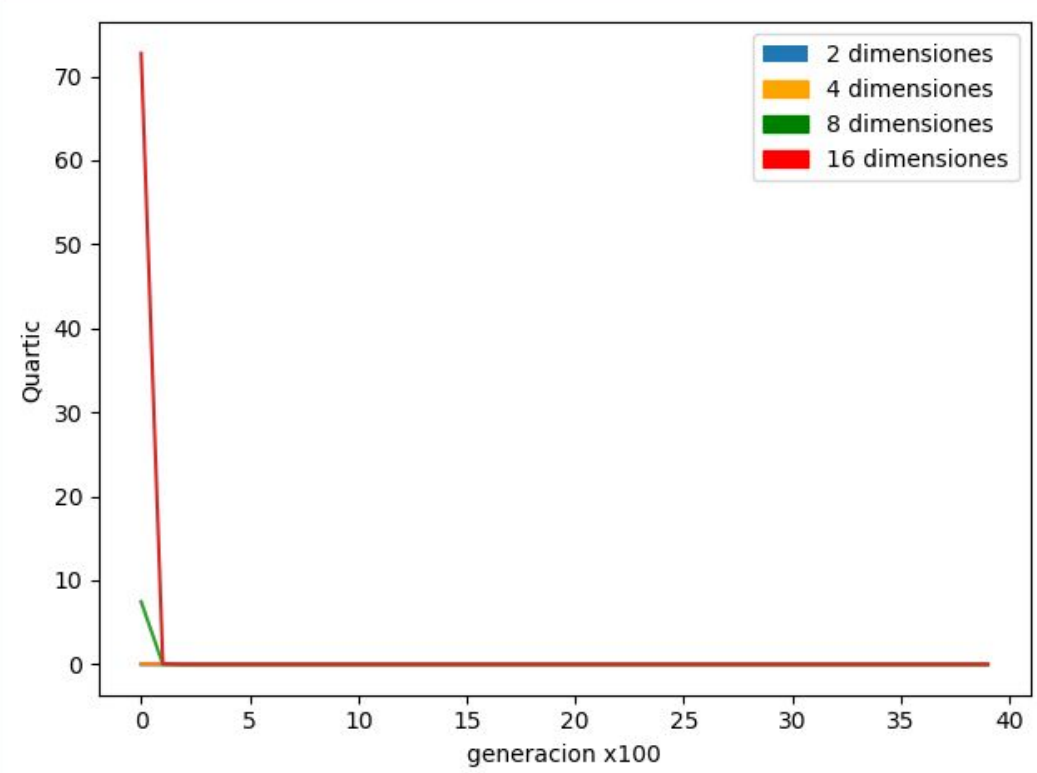
Formalmente hablando, se supone que tenemos una función desconocida, f(x,y)f(x,y), que podemos evaluar en los puntos que queramos pero a modo de caja negra, por lo que no podemos conocer su expresión. El objetivo es el habitual en optimización, encontrar valores de xx e yy para los que la función f(x,y)f(x,y) sea máxima (o mínima, o bien verifica alguna relación extremal respecto a alguna otra función). Como ya hemos visto en otras entradas similares, a f(x,y)f(x,y) se le suele llamar **función de fitness**, ya que va a determinar cómo de buena es la posición actual para cada partícula (a la función de fitness a veces también se le llama "**paisaje de fitness**", ya que puede verse como un paisaje con valles y colinas formados por los valores que toma la función).

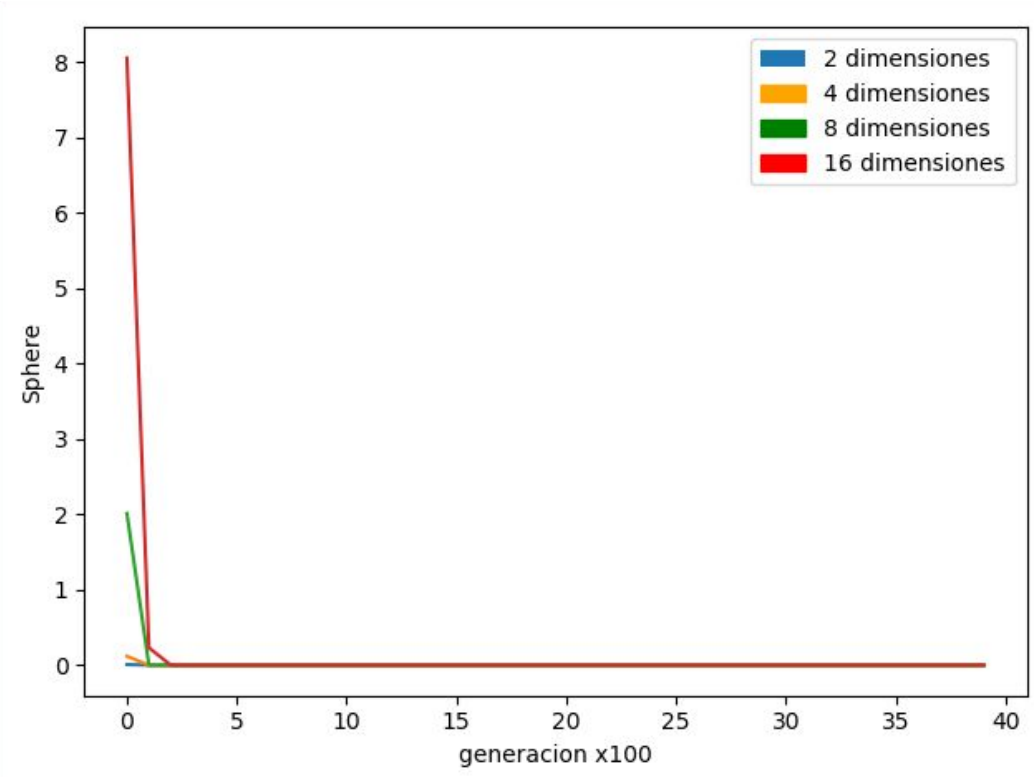
Desarrollo de la actividad.





4


2 dimensiones
4 dimensiones
8 dimensiones
16 dimensiones
Quartic
generacion x100


2 dimensiones
4 dimensiones
8 dimensiones
16 dimensiones
Sphere
generacion x100

Conclusión:

PSO es muy eficiente en tiempo además y además es el más adecuado para trabajar con datos dinámicos y otra que permita hacer predicciones. Es decir, debido a la forma en que funciona el algoritmo, si la función que se desea optimizar es dinámica (cambia en el tiempo), y esta dinámica verifica algunas propiedades de continuidad (el cambio no es muy brusco), entonces el mismo algoritmos hace que en cada iteración las partículas vayan evaluando el valor presente de la función a optimizar, por lo que pueden ir adaptando sus trayectorias a la situación real de la función.

Los algoritmos más habituales de optimización consideran la función en un instante determinado, intentarían optimizarla para ese instante, y tras ejecutar el algoritmo la función ya habría cambiado... dependiendo del coste en tiempo del algoritmo de optimización usado podemos encontrarnos que el valor devuelto como óptimo poco tiene que ver con el valor actual de la función, ya que en el transcurso de su ejecución ha ido cambiando. En este sentido, PSO funciona bien desde el punto de vista de optimización de funciones dinámicas.

Referencias:

http://www.cs.us.es/~fsancho/?e=70