Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Proyecto Final Busqueda con Información

Equipo:
Cornejo Chavez Edwin Joel
Veloz Alcaraz Axel Abraham
Espinoza Sucilla Samuel

Carrera: Ingeniería en computación

Materia: Seminario Inteligencia Artificial I

Maestro: Beltrán Carrillo Luis Ángel

Sección: D07 Viernes 9:00 AM a 1:00 PM.

Fecha: 02/12/2023

Introducción

Objetivo:
Comprender el funcionamiento de las técnicas de optimización metaheurísticas, mediante su implementación en problemas matemáticos y problemas reales.

Desarrollo e Implementación:
Esta práctica se divide en varias etapas, en las que el alumno será capaz de recopilar información e implementar soluciones a problemas matemáticos y problemas reales. El alumno trabajara en equipos de máximo tres integrantes. Los pasos a seguir para llevar a cabo esta práctica se describen a continuación:

Paso 1:

Realizar una investigación sobre los siguientes temas:
- Algoritmos Metaheurísticos
- Particle Swarm Optimization (PSO) (Optimización por enjambre de partículas)
- Genetic Algorithm (GA) (Algoritmo Genético). Usar la versión para números reales.
- Differential Evolution (DE) (Evolución Diferencial)

Algoritmos Metaheurísticos

Los algoritmos metaheurísticos son técnicas de optimización utilizadas para encontrar soluciones aproximadas a problemas de optimización combinatoria, continua o discreta. Estos algoritmos son útiles cuando se enfrentan a problemas difíciles o complejos para los cuales los métodos de optimización tradicionales pueden resultar ineficientes o impracticables.

Aquí hay algunas características clave y propósitos de los algoritmos metaheurísticos:

Exploración de Espacio de Búsqueda: Los problemas de optimización a menudo involucran la exploración de un gran espacio de soluciones posibles. Los algoritmos metaheurísticos son diseñados para explorar este espacio de manera eficiente, buscando soluciones que satisfagan ciertos criterios de optimización.

No Garantía de Optimalidad: A diferencia de algunos algoritmos de optimización exacta que garantizan encontrar la solución óptima, los algoritmos metaheurísticos no ofrecen tal garantía. En cambio, buscan soluciones que sean lo suficientemente buenas en un tiempo razonable.

Flexibilidad: Los algoritmos metaheurísticos son flexibles y pueden adaptarse a una variedad de problemas sin necesidad de modificar sustancialmente su estructura básica. Esto los hace aplicables a una amplia gama de dominios.

Aplicaciones Diversas: Se utilizan en una variedad de campos, como la ingeniería, la logística, la planificación, la economía, la inteligencia artificial, entre otros. Pueden abordar problemas como la programación de horarios, el diseño de redes, la asignación de recursos y la optimización de procesos.

Iterativos y Heurísticos: Estos algoritmos trabajan de manera iterativa, generando y evaluando soluciones repetidamente hasta alcanzar una solución aceptable. Utilizan heurísticas para guiar la búsqueda hacia regiones prometedoras del espacio de soluciones.

Adaptativos: Algunos algoritmos metaheurísticos son adaptativos y pueden ajustar sus parámetros durante la ejecución en función del progreso de la búsqueda o de la dinámica del problema.

Algunos ejemplos de algoritmos metaheurísticos incluyen algoritmos genéticos, búsqueda tabú, optimización por enjambre de partículas (PSO), recocido simulado, algoritmos de colonia de hormigas (ACO) y muchos más.

Particle Swarm Optimization (PSO) (Optimización por enjambre de partículas)

La Optimización por Enjambre de Partículas (PSO) es un algoritmo de optimización metaheurística inspirado en el comportamiento social de las aves y los peces. Fue propuesto por Eberhart y Kennedy en 1995 y desde entonces ha sido ampliamente utilizado en diversos campos debido a su simplicidad y eficacia.

Principios Básicos:

En PSO, una población de partículas se mueve a través de un espacio de búsqueda multidimensional, donde cada partícula representa una solución potencial al problema de optimización. Cada partícula ajusta su posición y velocidad de acuerdo con su experiencia local y la información compartida con otras partículas en el enjambre.

Estructura del Algoritmo:
El algoritmo PSO se puede describir en términos de los siguientes elementos clave:

Partículas: Cada partícula tiene una posición y una velocidad en el espacio de búsqueda. Estos atributos se actualizan en cada iteración del algoritmo.

Función Objetivo: El problema de optimización se modela mediante una función objetivo que evalúa la calidad de una solución en función de los parámetros del problema.

Mejor Posición Personal (pbest): Cada partícula mantiene un registro de la mejor solución que ha encontrado hasta el momento.

Mejor Posición Global (gbest): El enjambre también mantiene un registro de la mejor solución encontrada por cualquier partícula en el conjunto.

Actualización de Posición y Velocidad: En cada iteración, las partículas ajustan su velocidad y posición utilizando información local y global para buscar soluciones más óptimas.

Flujo de Trabajo:

Inicialización: Se generan aleatoriamente las posiciones y velocidades iniciales de las partículas.
Evaluación: Se evalúa la calidad de cada solución utilizando la función objetivo.
Actualización pbest y gbest: Se actualizan las mejores posiciones personales y globales.
Actualización de Posición y Velocidad: Se ajustan las posiciones y velocidades de las partículas.
Convergencia: El algoritmo continúa iterando hasta que se alcanza un criterio de convergencia o se agota el número de iteraciones predeterminado.
Aplicaciones y Ventajas:

Problemas de Optimización Continua y Discreta: PSO se ha aplicado con éxito a problemas de optimización en una variedad de dominios, incluyendo ingeniería, finanzas, logística y aprendizaje automático.

Eficiencia y Simplicidad: La simplicidad del algoritmo y su capacidad para evitar el estancamiento en óptimos locales son factores clave en su popularidad.

Adaptabilidad: PSO es altamente adaptable y puede ser modificado para abordar problemas específicos mediante la incorporación de estrategias de búsqueda local, restricciones y variación en la topología del enjambre.

En conclusión, la Optimización por Enjambre de Partículas es una poderosa técnica de optimización que ha demostrado ser eficaz en una variedad de aplicaciones. Su inspiración en la naturaleza social de las especies animales lo hace único y atractivo para abordar problemas complejos en la optimización.

Genetic Algorithm (GA) (Algoritmo Genético). Usar la versión para números reales.

Los Algoritmos Genéticos (GA) son técnicas de optimización basadas en la evolución biológica que buscan soluciones aproximadas a problemas de optimización. En el contexto de números reales, el GA se adapta para trabajar con cromosomas que representan soluciones codificadas en forma de vectores de números reales. Este enfoque es ampliamente utilizado en la optimización de funciones continuas y en problemas de diseño en los cuales las soluciones son mejor representadas por valores continuos.

Principios Básicos:
El GA para números reales sigue los principios generales de los algoritmos genéticos, pero con algunas modificaciones específicas para manejar números reales. Los elementos clave incluyen:

Codificación de Cromosomas: Cada individuo en la población es representado por un cromosoma que consiste en un vector de números reales. La longitud del vector corresponde al número de variables en el problema de optimización.

Inicialización: Se genera una población inicial de soluciones aleatorias, donde cada cromosoma representa una posible solución al problema.

Función Objetivo: Se define una función objetivo que evalúa la calidad de cada solución en función de los valores reales en el cromosoma.

Selección: Los individuos se seleccionan para reproducción proporcionalmente a su aptitud, determinada por la evaluación de la función objetivo.

Cruzamiento (Crossover): Pares de padres seleccionados se cruzan para producir descendencia, combinando sus valores reales en los cromosomas.

Mutación: Se introduce una pequeña probabilidad de mutación para alterar aleatoriamente algunos valores en los cromosomas de la descendencia.

Reemplazo: La nueva población se forma combinando los padres y la descendencia, y luego se seleccionan los individuos para la siguiente generación.

Criterio de Parada: El algoritmo continúa iterando hasta que se alcanza un criterio de parada, como un número máximo de generaciones o una convergencia aceptable.

Ventajas y Aplicaciones:

Optimización de Funciones Continuas: Los GA para números reales son especialmente efectivos en la optimización de funciones continuas, donde las soluciones son mejor representadas por valores numéricos.

Robustez: Pueden manejar problemas con múltiples óptimos locales y converger hacia soluciones de alta calidad.

Adaptabilidad: Los GA son altamente adaptables y pueden ser ajustados para abordar problemas específicos mediante la modificación de operadores genéticos y estrategias de selección.

Aplicaciones Diversas: Se aplican en una amplia gama de campos, incluyendo ingeniería, finanzas, diseño de sistemas, aprendizaje automático y más.

Desafíos y Consideraciones:

Tamaño de Población: La elección del tamaño de la población puede influir en la exploración y explotación del espacio de búsqueda.

Operadores Genéticos: La selección y ajuste de operadores genéticos, como el crossover y la mutación, puede requerir experimentación para optimizar el rendimiento del algoritmo.

Differential Evolution (DE) (Evolución Diferencial)

La Evolución Diferencial (DE) es un algoritmo de optimización que pertenece a la familia de algoritmos evolutivos. Fue propuesto por primera vez por Storn y Price en 1997 y ha demostrado ser eficaz en la optimización de funciones no lineales, especialmente en problemas continuos y con espacio de búsqueda multidimensional.

Principios Básicos:
DE se basa en la evolución de un conjunto de vectores, cada uno representando una solución candidata al problema de optimización. La característica distintiva de DE es la operación de diferencia entre soluciones para generar nuevas soluciones candidatas. El proceso evolutivo sigue estos pasos básicos:

Inicialización: Se genera una población inicial de soluciones candidatas, cada una representada por un vector en el espacio de búsqueda.

Diferenciación: Se seleccionan tres vectores aleatorios de la población, y se calcula la diferencia ponderada entre dos de ellos. Esta diferencia se suma al tercer vector para obtener un nuevo vector de prueba.

Evaluación: La calidad de la solución de prueba se evalúa utilizando la función objetivo del problema.

Selección: Se compara la calidad de la solución de prueba con la de un vector de la población original. Si es mejor, se reemplaza el vector antiguo por la nueva solución.

Iteración: El proceso se repite para generar nuevas soluciones en cada iteración hasta que se alcanza un criterio de convergencia.

Características Clave:

Operadores Diferenciales: La operación de diferencia es clave en DE y permite una exploración eficiente del espacio de búsqueda.

Adaptabilidad: DE es robusto y adaptable a diferentes tipos de funciones objetivo, especialmente en problemas de optimización global.

Eficacia en Espacios de Alta Dimensión: DE ha demostrado ser eficaz en problemas con un alto número de variables, donde otros algoritmos pueden encontrar dificultades.

Ventajas y Aplicaciones:

Eficacia en Problemas Multimodales: DE es especialmente eficaz en problemas con múltiples óptimos locales y globales, ya que puede explorar diferentes regiones del espacio de búsqueda.

Robustez: DE es menos sensible a la elección de parámetros y a menudo requiere menos ajuste en comparación con algunos otros algoritmos de optimización.

Aplicaciones en Ingeniería y Ciencia: Se ha aplicado con éxito en una variedad de campos, como diseño de ingeniería, ajuste de modelos, optimización de funciones no lineales y más.

Desafíos y Consideraciones:

Ajuste de Parámetros: Aunque es menos sensible a los parámetros que algunos algoritmos, DE aún requiere una elección adecuada de parámetros para un rendimiento óptimo.

Convergencia: La convergencia puede ser más lenta en comparación con otros algoritmos, dependiendo del problema y los parámetros seleccionados.

Paso 2:

Responde las siguientes preguntas:

1.- ¿Qué son y para qué se usan los algoritmos Metaheurísticos?

Los algoritmos metaheurísticos son técnicas de optimización que se utilizan para encontrar soluciones aproximadas a problemas complejos de optimización, donde las soluciones exactas son difíciles de obtener en un tiempo razonable. Estos algoritmos son adecuados para una amplia gama de problemas en diversas disciplinas, como ingeniería, logística, finanzas, inteligencia artificial, entre otros.

2.- De forma genérica, ¿Cuáles son las etapas básicas de un algoritmo Metaheurístico?

Las etapas básicas de un algoritmo metaheurístico son:
- Inicialización: Se genera una población inicial de soluciones.
- Evaluación: Se evalúa la calidad de cada solución utilizando una función objetivo.
- Selección: Se eligen soluciones para reproducirse o avanzar a la siguiente iteración basándose en su aptitud.
- Reproducción: Se generan nuevas soluciones mediante operadores como cruzamiento y mutación.
- Reemplazo: Se actualiza la población con las nuevas soluciones.
- Convergencia: Se repiten los pasos anteriores hasta que se cumple un criterio de convergencia.

3.- Explica las diferencias entre un algoritmo basado en operadores evolutivos y uno basado en enjambre.

En un algoritmo basado en operadores evolutivos, como los algoritmos genéticos, se utilizan conceptos inspirados en la evolución biológica, como la selección natural, cruzamiento y mutación. En cambio, un algoritmo basado en enjambre, como el PSO, se inspira en el comportamiento colectivo de organismos sociales, donde las soluciones se mueven en el espacio de búsqueda siguiendo reglas de interacción basadas en información local y global.

4.- Explica el funcionamiento de la ecuación que permite calcular la velocidad de una partícula en el algoritmo PSO.

La ecuación que calcula la velocidad (vi) de una partícula en el algoritmo PSO es:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pbest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gbest - x_i(t))$$

donde:

- w es el peso inercial que regula la influencia de la velocidad anterior.
- c1 y c2 son coeficientes de aceleración para la influencia de la mejor posición personal (pbest) y la mejor posición global (gbest) respectivamente.
- r1 y r2 son números aleatorios en el intervalo [0, 1].
- xi (t) es la posición de la partícula i en la iteración t.

5.- Describe el funcionamiento de los operadores de mutación y cruzamiento en los algoritmos DE y GA.

Differential Evolution (DE):
- Mutación: Se seleccionan tres soluciones aleatorias (a, b, c) de la población. La nueva solución de prueba se calcula como a + F ° (b - c), donde F es un factor escala.
- Cruzamiento (Crossover): Se compara cada componente de la solución de prueba con la solución original. Si un valor aleatorio es menor que una probabilidad de cruzamiento (CR) se conserva el valor de la solución de prueba; de lo contrario, se conserva el valor original.

Genetic Algorithm (GA) para Números Reales:
- Mutación: Cada componente del cromosoma (solución) puede ser alterado con cierta probabilidad. Un nuevo valor se genera aleatoriamente en un rango alrededor del valor original.
- Cruzamiento (Crossover): Se elige un punto de corte aleatorio en los cromosomas padres y se intercambian las secciones adyacentes para producir descendencia.

Paso 3:

Implementar los algoritmos PSO, GA y DE en cualquier lenguaje de programación. En este paso se debe experimentar con (al menos dos) funciones matemáticas con la finalidad de visualizar y entender mejor el funcionamiento de los métodos usados.

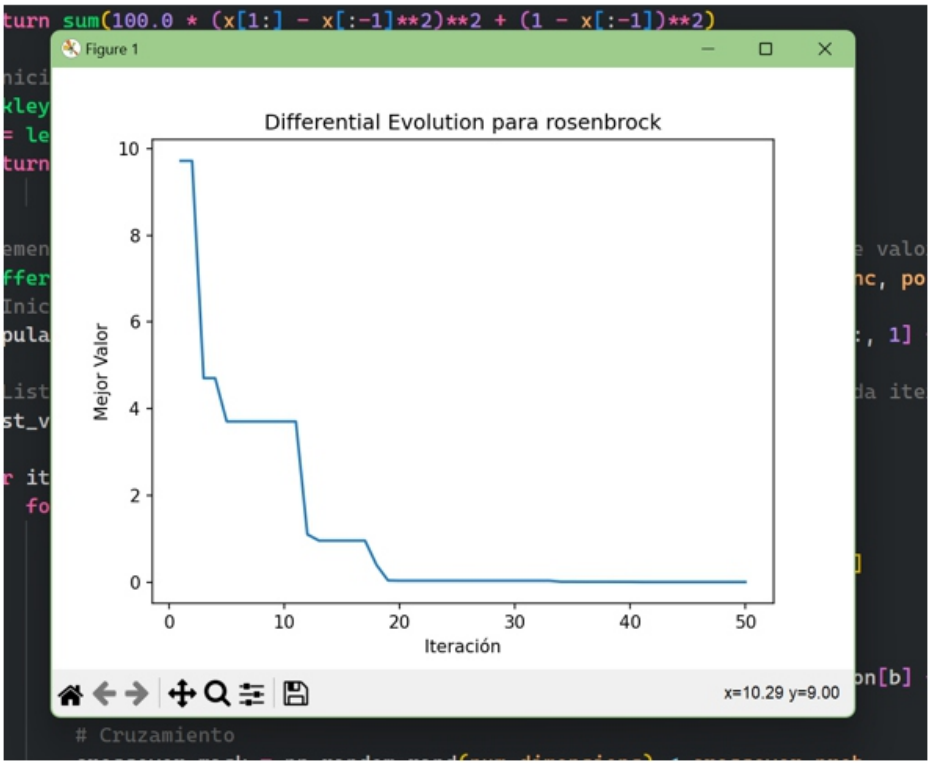
Las funciones matemáticas las puedes obtener de la siguiente página:
https://www.sfu.ca/~ssurjano/optimization.html

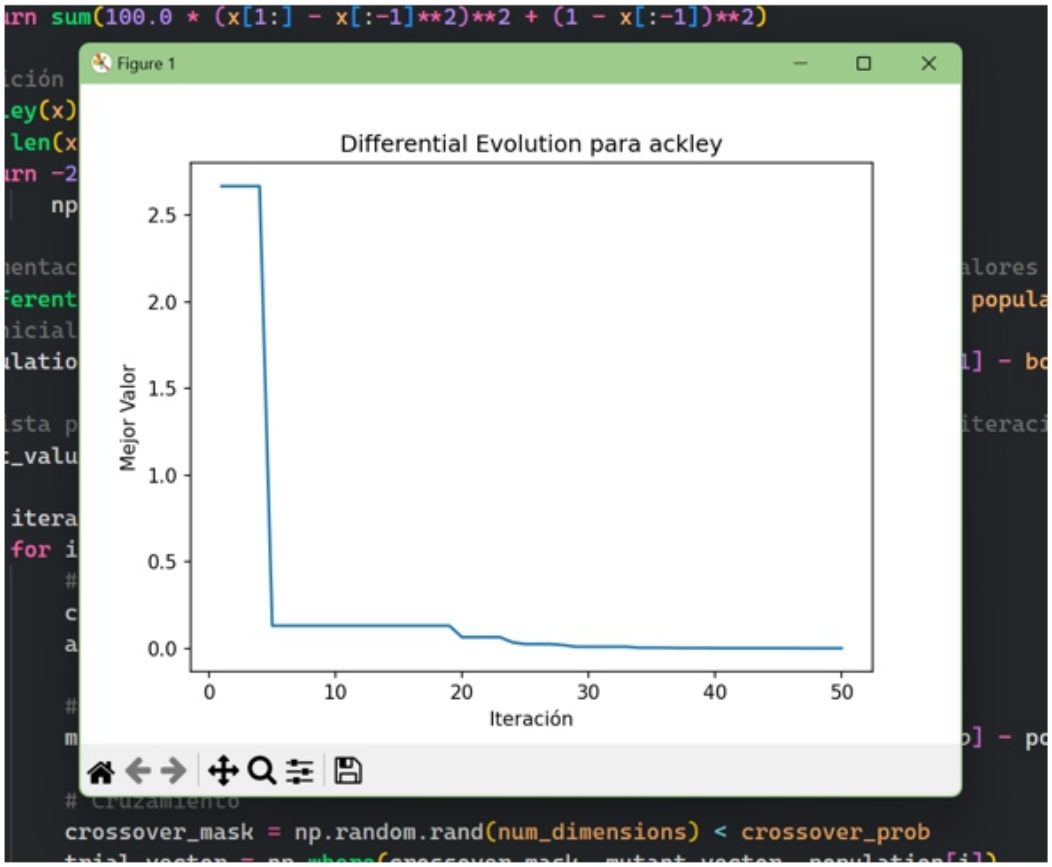
Comparativa de resultados:

• Grafica la evolución del valor de la función objetivo a lo largo de las iteraciones para una sola corrida (ejecución) de los tres algoritmos. Explica cuál es el mejor y por qué.
• Ejecuta cada algoritmo 20 veces (para cada problema) y almacena el mejor valor de la función objetivo que se obtiene al final de cada ejecución. Obtén el promedio y la desviación estándar para cada algoritmo en cada problema considerando los mejores valores obtenidos. Elabora un ranking de los 3 algoritmos usando el promedio.

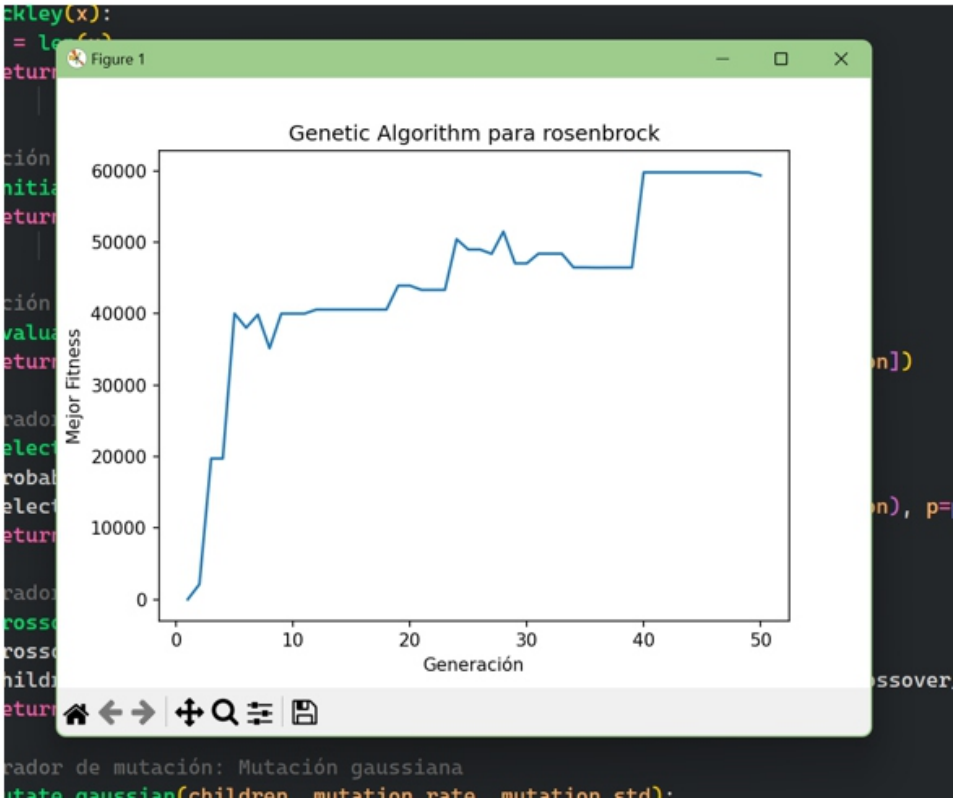
Consideraciones, los 3 algoritmos deben tener el mismo tamaño de población y deben correr el mismo número de iteraciones.

Algoritmo DE

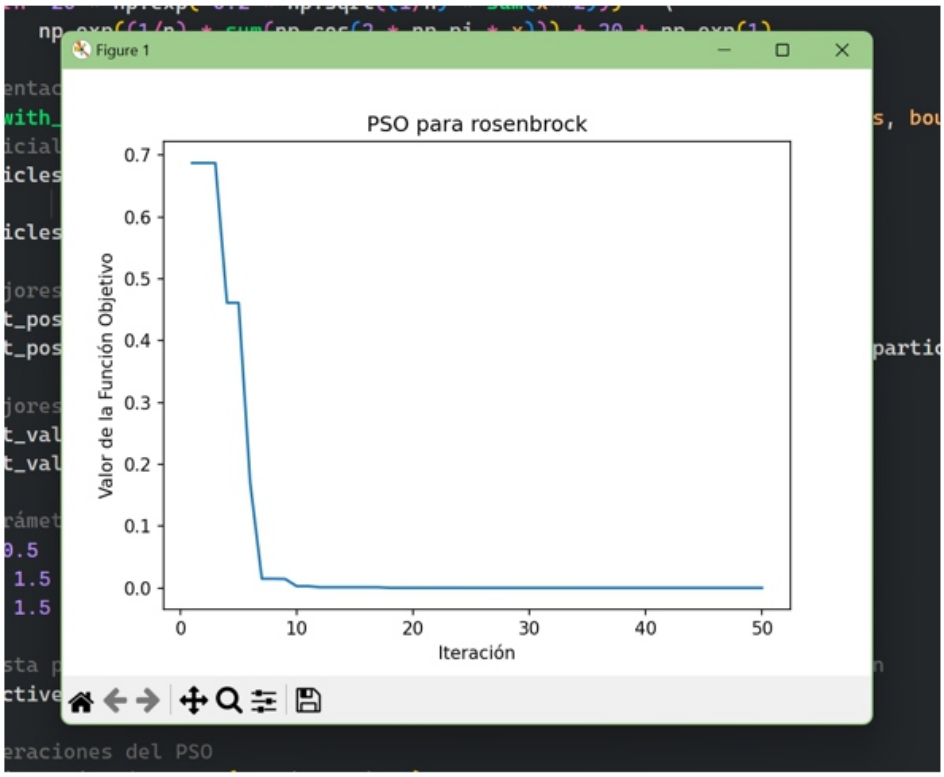




Algoritmo GA



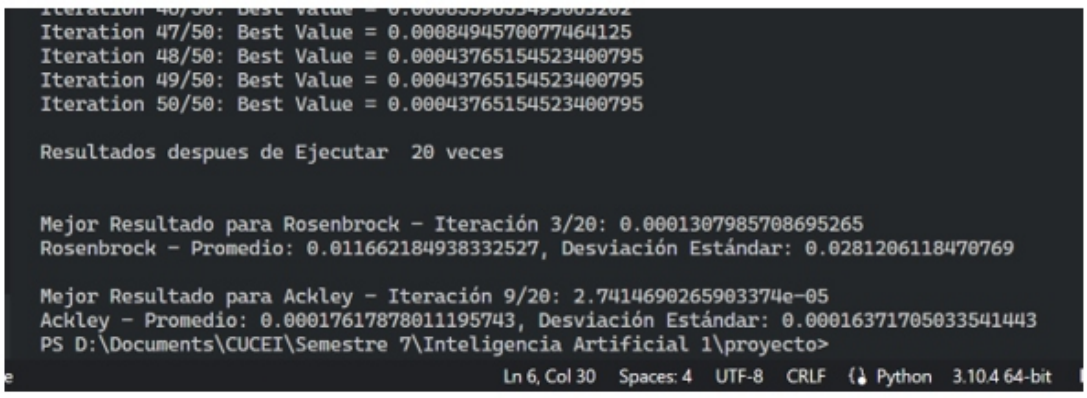
Algoritmo PSO



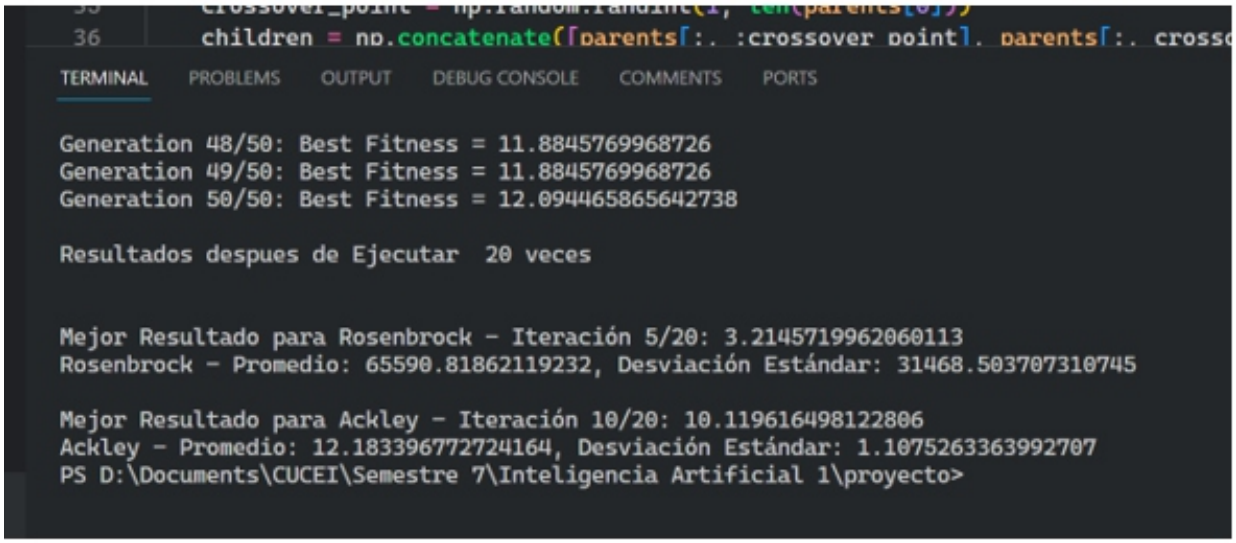
Después de correr los algoritmos una vez para ver los resultados y saber cual es mejor la respuesta como tal no hay definida, cada algoritmo tiene un uso y ese uso depende de para que problema o función lo necesitemos, en este caso vemos que el algoritmo GA se comporta de una manera muy diferente a los otros dos pero puede ser por el tipo de funciones que estamos utilizando para este tipo de algoritmo en cambio PSO y DE se comportan de una manera mas logica al momento de encontrar el mejor resultado a las dos funciones de prueba que pusimos para estos 3 algoritmos.

Ahora para el segundo punto se enseñaran las siguientes imágenes y después daremos una breve explicación del comportamiento de los algoritmos al correrlos 20 veces con su respectivo avg.

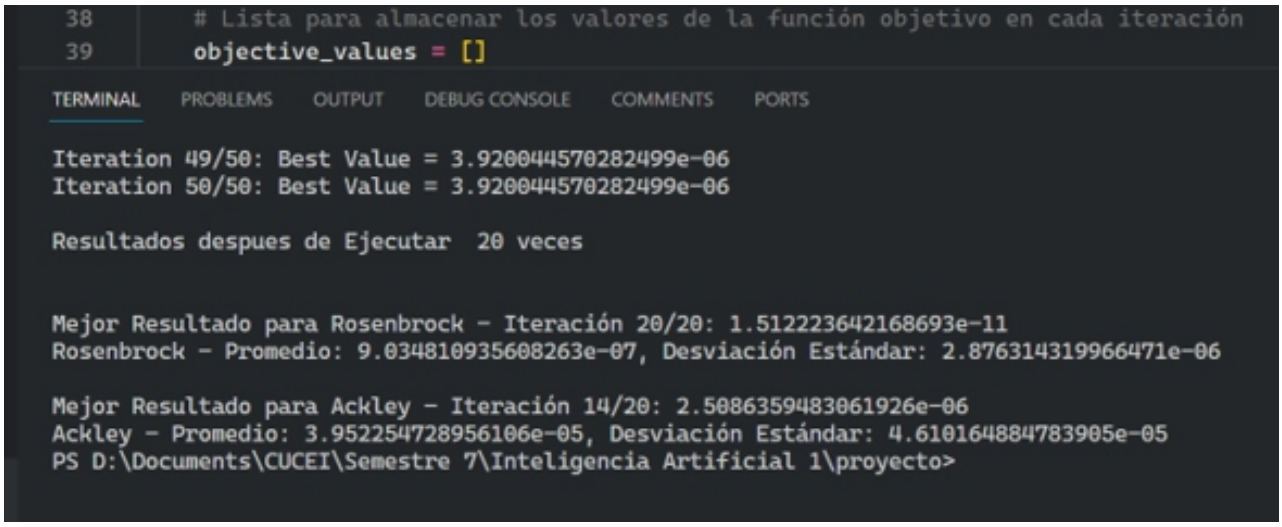
Algoritmo DE



Algoritmo GA



Algoritmo PSO



Después de ejecutarlo 20 veces podemos observar que de los 3 el mejor resultado para la función rosenbrock es el del algoritmo DE pero para la función Ackley es el PSO, siendo el algoritmo GA estar muy lejos de los resultados de los otros 2 algoritmos en este caso por el tipo de función que estamos utilizando, cada uno tiene su promedio y desviación de las repeticiones que se les dio en el código para comparar los resultados y en todos tienen los mismos valores iniciales.

Paso 5:

Redactar un reporte en el cual se describa todo el proceso del desarrollo de esta práctica, así como los resultados y las experiencias que se obtuvieron al tratar este tipo de técnicas.

Libros de consulta:

- Engineering Optimization: An Introduction with Metaheuristic Applications Xin-She Yang ISBN: 978-0-470-58246-6, 347 pages, Wiley 2010

- Evolutionary Optimization Algorithms: Biologically-Inspired and Population-Based Approaches to Computer Intelligence, Dan Simon, John Wiley & Sons, 2013.

- Advances of Evolutionary Computation: Methods and Operators, Studies in Computational Intelligence, Erik Valdemar Cuevas Jimenez, Margarita Díaz Cortes, Diego Oliva Navarro, Vol. 629, Springer International Publishing. DOI10.1007/978-3-319-28503-0.

- Erik V. Cuevas Jiménez, Diego A. Oliva Navarro, José V. Osuna Enciso, Margarita A. Diaz Cortes. “Optimización: Algoritmos programados con Matlab”. Primera Edición, Alfaomega Grupo Editor, México, June 2016.

Funciones matemáticas de prueba:

- http://www.sfu.ca/~ssurjano/optimization.html

- http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm