U N I V E R S I D A D D E G U A D A L A J A R A

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍA
DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

Seminario de Solución de Problemas de Sistemas Basados en Conocimiento

Práctica No. 1

Nombre: Hurtado González Edgar Arturo
Código: 212597894

Introducción

Los algoritmos evolutivos y de optimización son métodos computacionales inspirados en procesos naturales que buscan encontrar soluciones óptimas para problemas complejos. Estos algoritmos se basan en principios de evolución y selección natural para generar soluciones progresivamente mejores a lo largo del tiempo.

Los algoritmos evolutivos incluyen diferentes enfoques, como:

1. **Algoritmos Genéticos (AG):** Están inspirados en la teoría de la evolución de Darwin y la selección natural. Usan operadores genéticos como mutación, recombinación (crossover) y selección para generar una población de soluciones candidatas que se van mejorando en sucesivas generaciones.
2. **Programación Evolutiva (PE):** Similar a los algoritmos genéticos, pero se enfoca más en la evolución de programas informáticos, representando las soluciones como estructuras de programas o reglas.
3. **Estrategias de Evolución (EE):** Se centran en la evolución de vectores de números reales. Emplean operadores específicos, como mutación y recombinación, en estos vectores para optimizar funciones de aptitud.
4. **Programación Genética (PG):** Evoluciona estructuras de programas informáticos utilizando estructuras de árboles y operadores genéticos.

Por otro lado, los algoritmos de optimización son técnicas que buscan encontrar la mejor solución entre un conjunto de posibles soluciones. Estos algoritmos pueden ser utilizados en una amplia gama de problemas, como la optimización de funciones matemáticas, el diseño de sistemas, la planificación, la logística, entre otros.

Algunos ejemplos de algoritmos de optimización incluyen:

1. **Algoritmo de búsqueda aleatoria:** Explora soluciones de manera aleatoria hasta encontrar una solución satisfactoria o converger hacia una óptima.
2. **Búsqueda local:** Comienza desde una solución inicial y realiza movimientos iterativos hacia soluciones vecinas, buscando mejorar continuamente hasta alcanzar un óptimo local.
3. **Algoritmos de optimización por enjambre de partículas (PSO):** Modela el comportamiento de un enjambre de partículas que se mueven en el espacio de búsqueda para encontrar la mejor solución posible.
4. **Algoritmos de enjambre de partículas (ACO):** Inspirados en el comportamiento de colonias de hormigas, utilizan feromonas para construir soluciones óptimas en problemas combinatorios.

Estos algoritmos se aplican en una amplia gama de campos, como la ingeniería, la inteligencia artificial, la ciencia de datos, la economía, la logística y más, para resolver problemas de optimización en diversas áreas. Su eficacia y aplicabilidad dependen del tipo de problema y de cómo se adapten y configuren los parámetros para cada situación específica.

En el algoritmo que se implementó, se utilizaron varios operadores típicos de un algoritmo genético. Aquí hay una descripción de estos operadores:

1. **Inicialización:** Se generó una población inicial de soluciones aleatorias dentro de un rango específico definido por limite_inferior y limite_superior.
2. **Mutación:** En el proceso de mutación, se seleccionan tres soluciones aleatorias de la población y se combinan para generar una nueva solución (v) mediante la fórmula de mutación que involucra el factor de mutación F. Este operador busca explorar el espacio de búsqueda generando nuevas soluciones.
3. **Cruzamiento (Crossover):** El operador de cruzamiento (a veces llamado recombination) se implementó de manera bastante peculiar. Se utilizó un enfoque basado en una probabilidad de cruzamiento (cr) para determinar si cada dimensión de la nueva solución (u) generada mediante mutación sería heredada de la solución mutada (v) o de la solución original. Esto está relacionado con la creación de la nueva solución y su exploración del espacio de búsqueda.
4. **Selección:** En esta implementación, se utilizó una estrategia de selección basada en el valor de la función de Levy. Si la nueva solución generada (u) tiene un fitness (calculado con la función de Levy) mejor que el fitness de la solución original, se reemplaza la solución original por la nueva en la población. Este operador busca favorecer soluciones con mejor fitness, buscando mejorar la población en cada iteración.
5. **Elitismo:** En este caso específico, no se implementó explícitamente un operador de elitismo, que implica mantener las mejores soluciones sin cambios a lo largo de las generaciones. Sin embargo, se almacena la mejor solución encontrada (best_solution) a lo largo de las iteraciones para su posterior análisis.

Estos operadores son fundamentales en un algoritmo genético y juegan roles específicos para explorar, explotar y mejorar las soluciones en el espacio de búsqueda. Cada uno contribuye a la diversidad, la exploración y la explotación del espacio de búsqueda en la optimización del problema dado.

Desarrollo

Cómo abordé los puntos de la tarea con respecto al algoritmo genético:

1. **Inicialización de la población:** Establecí los límites y dimensiones del espacio de búsqueda para generar una población inicial de soluciones aleatorias. Se utilizo la función **np.random.rand()** para generar valores dentro del rango definido por los límites superior e inferior.

2. **Función de Evaluación (Función Levy N. 13):** Implementé la función de Levy N. 13 (cambiando la ya implementada Ackley) utilizando las expresiones matemáticas definidas para esta función. Esta función calcula el valor de fitness de cada solución en el espacio de búsqueda.

3. **Convergencia y Mejor Solución Encontrada:** Almacené el mejor fitness encontrado en cada iteración en una lista best_fitness_per_iteration para graficar la convergencia del algoritmo. Además, implementé un mecanismo para almacenar la mejor solución encontrada (best_solution) junto con su fitness correspondiente durante el proceso.

4. **Graficación de la Convergencia:** Utilicé Matplotlib para generar una gráfica que muestra la convergencia del algoritmo genético a través de las iteraciones, utilizando el mejor fitness encontrado en cada iteración.

Conclusión/Resultados

```
Mejor solución encontrada: [ 3.22675682 -1.64970797]
Fitness de la mejor solución: 0.025888203259727675
```