TECNOLÓGICO NACIONAL DE MÉXICO

Instituto tecnológico de Culiacán

Actividad:

Algoritmos genéticos

Alumno:

Iván Eduardo Ramírez moreno

Docente:

ZURIEL DATHAN MORA FELIX

Materia:

Tópicos de IA

Numero de control:

20170787

Semestre:

10

Algoritmos Genéticos

Introducción

Los **Algoritmos Genéticos (AG)** son una clase de algoritmos evolutivos inspirados en la **teoría de la evolución natural**. Basados en los principios de **selección, reproducción, cruce y mutación**, los AG han demostrado ser herramientas eficientes para resolver problemas de optimización, búsqueda y aprendizaje. En el ámbito de la Inteligencia Artificial, permiten explorar espacios complejos de soluciones de forma paralela, adaptativa y robusta.

1. Fundamento Biológico

Los algoritmos genéticos simulan procesos naturales descritos por **Darwin** como la "supervivencia del más apto". En este contexto:

- Cada solución posible es un **individuo** o **cromosoma**.
- Cada individuo tiene una **aptitud (fitness)** que indica qué tan buena es su solución.
- La población evoluciona mediante **selección natural**, **cruce** y **mutación**.

2. Proceso de Implementación según los 10 pasos del Profesor

A continuación se explica paso a paso el proceso típico de un algoritmo genético, en correspondencia directa con los 10 pasos que mencionaste.

Paso 1: Representación y parámetros

- **Codificación:** Generalmente binaria, aunque puede ser real o simbólica.
- **Ejemplo:** Para el problema de la mochila, un cromosoma podría ser 101100, donde cada bit representa si un objeto se incluye o no.
- **Parámetros iniciales:**
    - Tamaño de población N (por ejemplo, 50 individuos)
    - Probabilidad de cruce pc (comúnmente entre 0.6 y 0.9)
    - Probabilidad de mutación pm (pequeña, entre 0.001 y 0.1)

Paso 2: Definir la función de aptitud

- La **función de aptitud (fitness)** evalúa qué tan buena es una solución.

- Debe estar directamente relacionada con el objetivo del problema.
- Ejemplo para mochila: suma de los valores si el peso total no se excede; si se excede, fitness = 0.

Paso 3: Población inicial

- Se generan N cromosomas aleatorios, cada uno representando una solución posible al problema.

Paso 4: Evaluar aptitud

- Se calcula la función de aptitud para cada cromosoma en la población.

Paso 5: Selección

- Se escogen los cromosomas más aptos para reproducirse.
- Métodos comunes:
    - **Ruleta:** Selección proporcional a la aptitud.
    - **Torneo:** Se escoge el mejor entre un subconjunto aleatorio.
    - **Ranking:** Se asignan probabilidades según la posición.

Paso 6: Cruce y mutación

- **Cruce (crossover):** Se mezclan genes entre padres para generar hijos.
    - Ejemplo: Cruce de un punto:
      Padre 1: 101|011
      Padre 2: 010|110
      Hijo: 101110
- **Mutación:** Se alteran aleatoriamente genes individuales.
    - Ejemplo: cambiar un bit de 0 a 1.

Paso 7: Reemplazo y repetición

- La nueva generación sustituye total o parcialmente a la anterior.
- Se repite desde el paso 4 hasta que se cumpla una condición de parada:
    - Número de generaciones
    - Tiempo máximo
    - Convergencia del fitness

Paso 8: Mejor solución

- Se selecciona el mejor cromosoma como la solución óptima aproximada.

Paso 9: Análisis de resultados

- Se estudia el rendimiento del algoritmo:
    - ¿Convergió rápido?
    - ¿Se quedó atrapado en un óptimo local?
    - ¿La solución es aplicable?

Paso 10: Aplicación práctica

- El mejor cromosoma se interpreta para tomar decisiones o ejecutar acciones reales.
    - Ejemplo: asignar recursos, diseñar una estructura, controlar un robot.

3. Aplicaciones de Algoritmos Genéticos

Campo	**Aplicación**
Robótica	Optimización de trayectorias
IA en videojuegos	Evolución de comportamiento de NPCs
Finanzas	Optimización de portafolios
Medicina	Diseño de tratamientos personalizados
Ingeniería	Diseño asistido por computador
Machine Learning	Selección de atributos, tuning de modelos

4. Ventajas y Desventajas

Ventajas

- No necesitan derivadas ni información estructural.
- Capaces de escapar de óptimos locales.
- Fáciles de paralelizar.
- Flexibles para representar distintos tipos de soluciones.

Desventajas

- Sensibles al ajuste de parámetros.

- Pueden requerir mucho tiempo de cómputo.
- No garantizan encontrar la solución óptima exacta.
- Puede haber convergencia prematura sin diversidad genética.

5. Caso de Estudio: Problema de la Mochila 0/1

Objetivo:

Maximizar el valor total de los objetos sin superar un peso máximo.

Codificación:

Cada bit representa un objeto:
1 = incluido, 0 = excluido.

Ejemplo:

Objetos:
Peso = [2, 3, 4, 5],
Valor = [3, 4, 5, 8]

Capacidad de la mochila: 8

Cromosoma: 1100
Peso total = 5, Valor total = 7 → Fitness = 7

El algoritmo evoluciona poblaciones de combinaciones hasta encontrar la más rentable que no exceda la capacidad.

6. Variaciones Avanzadas

- **Programación Genética:** En lugar de cromosomas simples, se usan estructuras de árboles que representan programas.
- **AG Paralelos:** Se dividen poblaciones para evolución distribuida.
- **Algoritmos Meméticos:** Combinan AG con búsqueda local.

Conclusión

Los **algoritmos genéticos**, al seguir un proceso evolutivo en múltiples generaciones, ofrecen una solución práctica para problemas complejos donde la búsqueda exhaustiva no es viable. Al aplicar los 10 pasos sugeridos por el profesor, se puede diseñar e implementar un AG robusto que evolucione soluciones eficientes y adaptables a una gran variedad de contextos. Su uso continúa creciendo en la Inteligencia Artificial moderna debido a su versatilidad, escalabilidad y capacidad de adaptación.

Fuentes:

- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.
- Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press.
- Haupt, R. L., & Haupt, S. E. (2004). *Practical Genetic Algorithms* (2nd ed.). Wiley-Interscience.
- Eiben, A. E., & Smith, J. E. (2015). *Introduction to Evolutionary Computing* (2nd ed.). Springer. https://doi.org/10.1007/978-3-662-44874-8
- Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, 4(2), 65–85. https://doi.org/10.1007/BF00175354
- Yang, X. S. (2014). *Nature-Inspired Optimization Algorithms*. Elsevier.
- Chong, E. K. P., & Zak, S. H. (2013). *An Introduction to Optimization* (4th ed.). Wiley.
- Bäck, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. IOP Publishing and Oxford University Press.
- De Jong, K. A. (2006). *Evolutionary Computation: A Unified Approach*. MIT Press.
- Poli, R., Langdon, W. B., & McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. http://www.gp-field-guide.org.uk