Introducción a los Algoritmos Evolutivos

SEMINARIO DE SOLUCION DE PROBLEMAS DE INTELIGENCIA ARTIFICIAL I

Vázquez Pérez Ignacio David

218292866

Ingeniería en computación

Definición de algoritmo evolutivo

El algoritmo evolutivo es una técnica de búsqueda que imita el proceso natural que ocurre en el ambiente, tomando en consideración la teoría darwiniana de la evolución de las especies. En los algoritmos genéticos (AG), tenemos una población de individuos, cada uno representado por un cromosoma, que a su vez representa una solución potencial al problema que se está tratando de resolver.

El problema en cuestión está definido por una función objetivo. Dependiendo de qué tan "bueno" sea el individuo en relación con la función objetivo, se le atribuye un valor que representa su calidad. Este valor se conoce como la aptitud del individuo y es un factor de evaluación importante. Los individuos con una aptitud alta tienen una mejor probabilidad de ser seleccionados para la nueva generación de la población.

En los AG, se utilizan tres operadores principales: selección (se crea una nueva población de individuos basada en la aptitud de los individuos de la generación anterior), cruce (típicamente se intercambian partes de los individuos entre dos individuos seleccionados para el cruce) y mutación (los valores de genes particulares se cambian aleatoriamente).

Algorithm 1 Genetic Algorithm

```
1: determine objective function (OF)
2: assign number of generation to 0 (t=0)
3: randomly create individuals in initial population P(t)
4: evaluate individuals in population P(t) using OF
5: while termination criterion is not satisfied do
6:     t=t+1
7:     select the individuals to population P(t) from P(t-1)
8:     change individuals of P(t) using crossover and mutation
9:     evaluate individuals in population P(t) using OF
10: end while
11: return the best individual found during the evolution
```

Problema:

Imaginemos una empresa de distribución de alimentos que debe entregar productos frescos a una red de clientes en una ciudad. El desafío es encontrar las rutas más eficientes para minimizar los costos de transporte y maximizar la puntualidad de las entregas, considerando restricciones como horarios de entrega, capacidad de carga de los vehículos y la ubicación de los clientes.

Aplicación del algoritmo:

La Programación Genética (GP) puede ser utilizada para diseñar algoritmos que generen soluciones óptimas para este problema. Se puede representar cada ruta como un cromosoma, donde cada gen representa una parada en la ruta. El algoritmo evolutivo de GP crea una población inicial de posibles rutas aleatorias y las evalúa en función de criterios como la distancia recorrida, el tiempo de entrega y la capacidad de carga de los vehículos. Luego, mediante la aplicación de operadores genéticos como la mutación y la recombinación, se generan nuevas generaciones de rutas que se adaptan y mejoran gradualmente a través del tiempo.

Resultados obtenidos:

Después de varias generaciones de evolución, el algoritmo de Programación Genética converge hacia soluciones óptimas o cercanas a óptimas para las rutas de reparto de alimentos. Estas soluciones pueden reducir significativamente los costos operativos de la empresa al minimizar la distancia recorrida y optimizar la utilización de los vehículos de entrega. Además, al garantizar la puntualidad de las entregas, se mejora la satisfacción del cliente y se fortalece la reputación de la empresa en el mercado.

La Programación Genética (GP)

Es una forma especializada de los Algoritmos Genéticos (AG) que opera en tipos muy específicos de soluciones, utilizando operadores genéticos modificados. Fue desarrollada por Koza como un intento de encontrar la manera de generar automáticamente códigos de programas cuando se conoce el criterio de evaluación para su correcto funcionamiento.

Debido a que la solución buscada es un programa, las soluciones potenciales evolucionadas se codifican en forma de árboles en lugar de cromosomas lineales (de bits o números) comúnmente utilizados en los AG. A diferencia de los AG, GP utiliza un esquema de codificación diferente, donde las soluciones se representan como árboles.

Sin embargo, el bucle principal de GP es similar al de los AG. Por supuesto, los operadores genéticos están especializados para trabajar en árboles, por ejemplo, el cruce consiste en intercambiar subárboles, y la mutación implica cambiar un nodo o una hoja.

El algoritmo de evolución diferencial (DE)

Es un tipo de algoritmo evolutivo útil principalmente para la optimización de funciones en un espacio de búsqueda continuo. Aunque también se ha discutido una versión del algoritmo DE para problemas

combinatorios, la versión principal fue discutida por Storn y Price. Las principales ventajas del DE sobre un GA tradicional son su facilidad de uso, una utilización eficiente de la memoria, una menor complejidad computacional (escala mejor al manejar problemas grandes) y un menor esfuerzo computacional (convergencia más rápida). El procedimiento estándar del DE, se optimiza el problema con n variables de decisión. El parámetro F escala los valores añadidos a las variables de decisión particulares (mutación), y el parámetro CR representa la tasa de cruce.

La idea principal del algoritmo DE está conectada con el cálculo de la diferencia entre dos individuos elegidos al azar de la población. Por lo tanto, el algoritmo DE evita que la solución se quede pegada en un extremo local de la función optimizada.

Algorithm 2 Differential Evolution

```
determine objective function (OF)
assign number of generation to 0 (t=0)
randomly create individuals in initial population P(t)
while termination criterion is not satisfied do
    t=t+1
    for each i-th individual in the population P(t) do
        randomly generate three integer numbers:
        r1, r2, r3∈[1; population size], where r1≠r2≠r3≠i
        for each j-th gene in i-th individual (j∈[1; n]) do
            v_{i,j} = x_{r1,j} + F · (x_{r2,j} − x_{r3,j})
            randomly generate one real number rand_j ∈ [0; 1)
            if rand_j < CR then u_{i,j} := v_{i,j}
            else
                u_{i,j} := x_{i,j}
            end if
        end for
        if individual u_i is better than individual x_i then
            replace individual x_i by child u_i individual
        end if
    end for
end while
return the best individual in population P(t)
```

Problema:

Consideremos una empresa de energía renovable que opera una granja eólica. El objetivo es maximizar la eficiencia y la producción de energía de la granja ajustando los parámetros de operación, como la velocidad de rotación de las turbinas eólicas, la orientación de las palas y la distribución de carga en la red eléctrica, mientras se minimizan los costos operativos y se cumplen las restricciones ambientales y de seguridad.

Aplicación del algoritmo:

El algoritmo de Evolución Diferencial (DE) se utiliza para encontrar la configuración óptima de los parámetros del sistema de energía renovable. Se define una función objetivo que representa la eficiencia y la producción de energía de la granja eólica en función de los parámetros ajustables. Luego, se inicializa una población de soluciones candidatas, cada una representando una

configuración diferente de parámetros. El algoritmo DE opera mediante la aplicación de operadores de mutación, recombinación y selección para generar nuevas soluciones y mejorar gradualmente la calidad de la población en cada generación.

Resultados obtenidos:

Después de varias iteraciones del algoritmo DE, se obtiene una configuración óptima de parámetros que maximiza la eficiencia y la producción de energía de la granja eólica. Esta configuración permite a la empresa de energía renovable aumentar la cantidad de energía generada, reducir los costos operativos y cumplir con los requisitos ambientales y de seguridad.

Estrategias de evolución

Las estrategias de evolución (ES) son diferentes en comparación con los algoritmos genéticos (GAs), principalmente en el procedimiento de selección. En el GA, la siguiente generación se crea a partir de la población parental eligiendo individuos según su valor de adaptación, manteniendo un tamaño constante de la población. En la ES, se crea una población temporal que tiene un tamaño diferente al de la población parental (dependiendo de los parámetros asumidos k y l). En este paso, los valores de adaptación no son importantes. Los individuos en la población temporal sufren cruces y mutaciones. A partir de estas poblaciones, se selecciona un número asumido de los mejores individuos para la siguiente generación de la población (de manera determinista). Las ES operan en vectores de números de punto flotante, mientras que el GA clásico opera en vectores binarios.

Aspectos técnicos del algoritmo:

- En las estrategias de evolución, se crea una población temporal con un tamaño diferente al de la población parental, según los parámetros k y l.
- Los valores de adaptación no se tienen en cuenta al crear la población temporal en las ES.
- Los individuos en la población temporal se someten a cruces y mutaciones.
- Se selecciona un número de los mejores individuos de la población temporal para la siguiente generación de manera determinista.
- Las ES operan en vectores de números de punto flotante, en contraste con los vectores binarios utilizados en los GAs clásicos.

La programación evolutiva

La programación evolutiva (EP) se desarrolló como una herramienta para descubrir la gramática de un idioma desconocido. Sin embargo, la EP se volvió más popular cuando se propuso como técnica de optimización numérica. La EP es similar a las estrategias de evolución (ES) (l + k), pero con una diferencia esencial. En la EP, la nueva población de individuos se crea mutando cada individuo de la población parental, mientras que en las ES (l + k), cada individuo tiene la misma probabilidad de ser seleccionado para la población temporal en la que se realizan las operaciones genéticas. En la EP, la mutación se basa en la perturbación aleatoria de los valores de los genes particulares del individuo mutado. Las poblaciones recién creadas y las parentales tienen el mismo tamaño (l = k). Finalmente,

se crea la nueva generación de la población utilizando la selección por rango de los individuos tanto de las poblaciones parentales como de las mutadas.

Aspectos técnicos del algoritmo:

- En la programación evolutiva, la nueva población se crea mutando cada individuo de la población parental.
- La mutación en la EP se basa en la perturbación aleatoria de los valores de los genes particulares del individuo mutado.
- Las poblaciones recién creadas y las parentales tienen el mismo tamaño.
- Se utiliza la selección por rango de los individuos tanto de las poblaciones parentales como de las mutadas para crear la nueva generación de la población.

Algorithm 4 Evolutionary Programming

```
1: determine objective function (OF)
2: assign number of generation to 0 (t=0)
3: randomly create individuals in initial population P(t)
4: evaluate individuals in population P(t) using OF
5: while termination criterion is not satisfied do
6:     t=t+1
7:     create population M(t) by the mutation of every individual from the population P(t-1)
8:     evaluate individuals in population M(t) using OF
9:     select the individuals to population P(t) from the sum of individuals in P(t-1) and M(t) using ranking selection
10: end while
11: return the best individual in population P(t)
```

Problema:

En una universidad, la asignación de horarios para clases, exámenes y actividades extracurriculares es un desafío complejo. Se deben tener en cuenta diversas restricciones, como la disponibilidad de aulas, los horarios de disponibilidad de profesores y estudiantes, la duración de las clases y los descansos entre ellas, así como las preferencias individuales de los estudiantes y la necesidad de equilibrar la carga académica de los mismos.

Aplicación del algoritmo:

La Programación Evolutiva (PE) se utiliza para diseñar un sistema de planificación de horarios que optimice la distribución de clases y actividades en la universidad. Se define una representación adecuada para los horarios, donde cada individuo de la población representa una posible programación de clases y actividades. La función de aptitud evalúa cada programación en función de criterios como la cantidad de conflictos de horario, la equidad en la distribución de clases entre los estudiantes y la utilización eficiente de los recursos disponibles. Mediante la aplicación de operadores genéticos como la selección, la recombinación y la mutación, se generan nuevas programaciones que se adaptan y mejoran en cada generación.

Resultados obtenidos:

Después de varias generaciones de evolución, el algoritmo de Programación Evolutiva converge hacia una programación óptima o cercana a la óptima para los horarios universitarios. Esta programación permite maximizar la utilización de recursos, minimizar los conflictos de horario y satisfacer las necesidades individuales de los estudiantes y profesores.

Problemas y retos

Los algoritmos evolutivos (EAs) representan un área de investigación muy interesante con numerosos problemas abiertos y desafíos. Entre estos se incluye el control del equilibrio entre las propiedades de exploración y explotación, la adaptación automática de los parámetros de control, la reducción de los parámetros del algoritmo CMA-ES, la introducción de nuevos esquemas de selección y el aumento de su efectividad, especialmente en diseño evolutivo y hardware evolutivo.

En el ámbito científico, se necesitan técnicas más eficientes para el manejo de restricciones, así como una mayor investigación sobre la aplicación de EAs en problemas de optimización dinámica, en entornos ruidosos y no estacionarios, y en problemas de optimización multiobjetivo, especialmente con un gran número de variables de decisión. También se requiere más investigación en la adaptación del tamaño de la población en diferentes escenarios de optimización y el desarrollo de estrategias novedosas para abordar problemas costosos de manera más competitiva.

En la vida real, la aplicación de EAs con técnicas de manejo de restricciones se ve influenciada por la complejidad y dificultad de implementación, además de la efectividad. Métodos como penalización, reglas de viabilidad y métodos de clasificación estocástica se utilizan con frecuencia debido a su simplicidad. Sin embargo, no existe un enfoque general para manejar las restricciones con EAs que pueda abordar cualquier problema del mundo real, lo que hace que la investigación sobre técnicas de manejo de restricciones en EAs para aplicaciones del mundo real siga siendo un tema candente en la actualidad.

Conclusiones

Me llamó la atención la diversidad y complejidad de los problemas abordados en el texto, así como la necesidad de desarrollar técnicas más efectivas para abordarlos. La adaptabilidad de los algoritmos evolutivos para enfrentar desafíos como el equilibrio entre exploración y explotación, la optimización en entornos dinámicos y ruidosos, y la gestión de restricciones en problemas de ingeniería es impresionante.

Imagino que los algoritmos evolutivos podrían ser utilizados en una variedad de problemas cotidianos, como la optimización de rutas de transporte para minimizar el tiempo de viaje, la planificación de horarios escolares para maximizar la eficiencia y la satisfacción de los estudiantes, o incluso en el diseño de productos para optimizar características como el costo y la durabilidad. La capacidad de los algoritmos evolutivos para adaptarse y encontrar soluciones óptimas en entornos complejos los hace herramientas potencialmente valiosas en una amplia gama de aplicaciones prácticas.

Bibliografía

Slowik, Adam & Kwasnicka, Halina. (2020). Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications. 32. 12363-12379. 10.1007/s00521-020-04832-8.