Monte Carlo Tree Search (MCTS) es un algoritmo de búsqueda utilizado principalmente en inteligencia artificial (IA) para la toma de decisiones en juegos y problemas de planificación. Es una técnica basada en simulaciones aleatorias que equilibra la exploración y la explotación para encontrar las mejores acciones en un espacio de decisiones complejo.

Funcionamiento de MCTS

MCTS se basa en **cuatro fases** que se repiten en múltiples iteraciones para mejorar la calidad de la decisión:

1️⃣ Selección

Se parte desde el nodo raíz y se elige iterativamente el mejor nodo hijo hasta llegar a un nodo que aún no ha sido completamente explorado.

- Para seleccionar los nodos, se utiliza una estrategia de equilibrio entre **exploración** (probar nuevas opciones) y **explotación** (elegir la mejor opción conocida).
- Una técnica común es **Upper Confidence Bound for Trees (UCT)**, que selecciona el nodo con la mejor relación entre su valor medio y el número de veces que ha sido explorado.

Fórmula de UCT:

UCT=WiNi+ClnNNiUCT = \frac{W_i}{N_i} + C \sqrt{\frac{\ln N}{N_i}}UCT=NiWi+CNilnN

Donde:

- WiW_iWi = Número de victorias en ese nodo.
- NiN_iNi = Número de veces que se ha visitado el nodo.
- NNN = Número total de visitas al nodo padre.
- CCC = Constante de exploración (ajusta el equilibrio entre exploración y explotación).

2️⃣ Expansión

Cuando se llega a un nodo no completamente explorado, se añaden nuevos nodos hijos al árbol, representando posibles movimientos futuros.

Si un nodo ya ha sido visitado antes, se continúa con la fase de selección hasta encontrar un nodo con hijos no explorados.

3️⃣ Simulación (Rollout)

Desde el nodo recién agregado, se realizan simulaciones aleatorias hasta alcanzar un estado terminal (por ejemplo, el final del juego en un entorno de juegos).

- En esta fase no se usa el árbol de búsqueda, sino que se toman decisiones aleatorias o siguiendo una estrategia simple.
- Se obtiene un resultado que puede ser una victoria, derrota o empate.

4️⃣ Retropropagación (Backpropagation)

El resultado de la simulación se propaga hacia arriba en el árbol, actualizando los valores de los nodos visitados.

- Se actualiza el número de visitas y la tasa de éxito de cada nodo en el camino de regreso hasta la raíz.
- Esto permite que futuras selecciones favorezcan movimientos que han llevado a buenos resultados.

Ejemplo en un juego

Supongamos que MCTS se usa en un juego de ajedrez:

1. **Selección**: Se elige una jugada entre las opciones actuales basándose en el balance entre exploración y explotación.
2. **Expansión**: Si la jugada no ha sido explorada antes, se agrega un nuevo nodo al árbol.
3. **Simulación**: Se juega una partida rápida de manera aleatoria desde ese estado hasta que termine.
4. **Retropropagación**: Si la simulación resultó en una victoria, se refuerza el valor de esa jugada en el árbol.

Tras muchas iteraciones, el algoritmo tendrá una mejor estimación de la mejor jugada disponible.

Ventajas y desventajas de MCTS

✅ **Ventajas:**
✔️ No necesita evaluar todos los movimientos posibles, lo que lo hace más eficiente que otros algoritmos como Minimax.
✔️ Funciona bien en espacios de búsqueda muy grandes.
✔️ Se puede usar con información parcial, adaptándose a entornos inciertos.

❌ **Desventajas:**
❌ Puede ser lento si el número de simulaciones es bajo.
❌ Depende de la calidad de las simulaciones para obtener buenos resultados.
❌ No siempre encuentra la mejor solución en problemas con reglas muy complejas.

MCTS ha sido clave en el éxito de inteligencias artificiales avanzadas, como **AlphaGo**, que venció a los mejores jugadores humanos de Go.

Conclusión

Monte Carlo Tree Search (MCTS) es un algoritmo poderoso y eficiente para la toma de decisiones en entornos de gran complejidad, como juegos y problemas de planificación. Su enfoque basado en simulaciones le permite encontrar soluciones sin necesidad de evaluar todas las posibilidades, equilibrando exploración y explotación de manera inteligente.

MCTS es ampliamente utilizado en inteligencia artificial para juegos como ajedrez, Go y videojuegos, así como en áreas como robótica y optimización. Su capacidad de adaptarse a espacios de búsqueda grandes y su flexibilidad lo convierten en una herramienta clave en la IA moderna.

Sin embargo, su rendimiento depende del número de simulaciones realizadas y de la calidad de la estrategia de selección de movimientos. Aunque no garantiza la solución óptima en todos los casos, su capacidad para aproximarse a decisiones sólidas lo hace una alternativa efectiva a métodos tradicionales como Minimax.