MEMORIA INTELIGENCIA ARTIFICIAL:

PRÁCTICA 3

NIP: 871627
Nombre: Irene Pascual Albericio

Irene Pascual Albericio
NIP: 871627

Índice:

Algoritmos de búsqueda:

A continuación vamos a explicar detalladamente los resultados de cada algoritmo empleado en el problema de las 8 reinas.

1) Hill-Climbing Search:

Resumen de Resultados

- **Fallos:** 85.47%
- **Coste medio en fallos:** 3.06 pasos
- **Éxitos:** 14.53%
- **Coste medio en éxitos:** 4.03 pasos

Los resultados de este algoritmo muestran las limitaciones que tiene ya que se suele estancar frecuentemente en máximos locales (lo cual podemos comprobar con el alto porcentaje de fallos). Este estancamiento lo suele tener porque el Hill-Climbing considera los vecinos inmediatos, quedando atrapado en configuraciones menos óptimas y no pudiendo retroceder para explorar otras partes del espacio.

No obstante, un pequeño porcentaje de los casos, se logra alcanzar una solución completa, aunque al ser este éxito esporádico, provoca que este algoritmo no sea confiable para usar si el problema puede tener muchos máximos locales.

Respecto al bajo número de pasos hasta el fallo, sugiere que el algoritmo detecta rápidamente si un estado inicial no tiene una solución factible cercana, y es por eso que termina el intento en unos pocos movimientos.

En cambio, cuando hay solución, el coste medio en éxitos es algo mayor que el de fallos, ya que puede llegar rápidamente a la solución sin desviarse por caminos no óptimos.

2) Simulated Annealing Search:

Resumen de Resultados

- **Fallos:** 44.20%
- **Coste medio en fallos:** 58.73 pasos
- **Éxitos:** 55.80%
- **Coste medio en éxitos:** 45.15 pasos

Aunque la tasa de fallos sigue siendo bastante alta (ya que es casi la mitad), se nota que es menor en comparación al Hill-Climbing, ya que encuentra muchas más veces la solución gracias a su capacidad de escapar de óptimos locales gracias al control del parámetro de enfriamiento. El enfriamiento simulado permite explorar configuraciones de menor calidad en las etapas iniciales, permitiendo así escapar de los óptimos locales y tener más probabilidades de éxito.

En cuanto a los éxitos, se nota que es más robusto, porque la reducción de la probabilidad de aceptar movimientos de peor calidad, permite al algoritmo progresar hacia configuraciones más óptimas en muchos casos.

Con lo que respecta al coste medio en fallos, este es bastante elevado porque su búsqueda es más exhaustiva (el algoritmo tarda en “enfriarse” para identificar que una solución no es viable).

El coste medio en éxitos también es alto, y esto se debe a que el algoritmo permite explorar varias configuraciones antes de decidirse por una solución.

3) Genetic Algorithm:

Resumen de Resultados

- **Población Inicial:** 50
- **Probabilidad de Mutación:** 0.15
- **Fitness de la Solución:** 28.0
- **Iteraciones:** 1626
- **Tiempo de Ejecución:** 3099 ms

El Algoritmo Genético es el más complejo de todos, ya que se propone encontrar las soluciones óptimas a través de la evolución poblacional.

Este algoritmo usa una población de 50 individuos y una probabilidad de mutación del 15%, con lo que se logra una solución completa. Esta configuración permite tener un buen equilibrio entre explorar nuevos individuos gracias a la mutación (que provoca que se hagan cambios aleatorios en algunos individuos), y explotar los mejores individuos encontrados dentro de la población.

El fitness es una medida de la calidad de una solución. Cuanto más alto sea el fitness, mejor es la configuración del tablero. Y esto muestra que con los parámetros seleccionados, el algoritmo es capaz de evolucionar hasta lograr una solución que cumpla todas las condiciones.

Sin embargo, sí que podemos comprobar que emplea muchas más iteraciones y tiempo que el resto de los algoritmos, pero esto es razonable dado que se necesitan muchas iteraciones para cruzar y mutar generaciones, y consecuentemente más tiempo.

4) Conclusión:

El algoritmo Hill-Climbing es más rápido pero más ineficaz al estancarse en máximos locales.

El algoritmo Simulated Annealing permite mayor exploración y fiabilidad, aunque es más lento.

El algoritmo Genético tiene una alta eficiencia y un mejor balance entre exploración y explotación.

Irene Pascual Albericio
NIP: 871627

Juegos:

Los algoritmos MINIMAX y PODA ALPHA-BETA, son fundamentales en juegos de toma de decisiones donde se enfrentan dos jugadores (adversario), con objetivos opuestos, como es el caso de los árboles de decisión.
Ambos algoritmos están implementados utilizando la clase ArboldeJuego, que representa un árbol de juego en el que cada nodo tiene un valor y puede ser de tipo max o min, indicando el objetivo del jugador en ese nivel.
Estos algoritmos se han aplicado sobre el árbol de decisión construido en EjemploArbolJuego.

1) Algoritmo MINIMAX:

Este algoritmo explora completamente el árbol de juego para determinar el valor óptimo en la raíz, evaluando los nodos terminales y propagando los valores hacia arriba. Los nodos de tipo max buscan maximizar el valor, y los de tipo min, buscan minimizarlo.

Resumen de Resultados

Valor con MINIMAX: 3.0
3.0-MAX
3.0-MIN
[3.0]
[12.0]
[8.0]
2.0-MIN
[2.0]
[4.0]
[6.0]
2.0-MIN
[14.0]
[5.0]
[2.0]

Nodos visitados: 12

El valor final calculado por el algoritmo es la raíz de 3.0, lo cual indica que la mejor estrategia para el jugador max puede hacer en el juego, asumiendo que el jugador min también toma las mejores soluciones. Así se asegura que el jugador max obtendrá como mínimo un valor 3.0 siempre.

Como MINIMAX explora el árbol completo, se visitan todos los nodos del árbol, asegurando así la obtención de la mejor decisión (aunque su coste sea mayor).

En conclusión, el algoritmo MINIMAX es efectivo para encontrar la mejor estrategia en un juego, aunque se tendrá por otro lado un número elevado de nodos visitados. Por tanto, esto es adecuado en árboles pequeños o en caso donde se necesitan evaluar todas las posibilidades.

2) Algoritmo de PODA ALPHA-BETA:

La poda alpha-beta optimiza el algoritmo MINIMAX, ya que se evita la evaluación de ramas que no influirán en la decisión final. Esto se consigue estableciendo límites, alpha y beta, que determinan cuándo es innecesario explorar ciertos nodos, reduciendo así el número de evaluaciones.

Resumen de Resultados

```
Valor con poda Alfa Beta: 3.0
3.0-MAX
   3.0-MIN
        [3.0]
       [12.0]
        [8.0]
   2.0-MIN
        [2.0]
         [X]
         [X]
   2.0-MIN
       [14.0]
        [5.0]
        [2.0]

Nodos visitados: 10
```

Al igual que MINIMAX, la poda alpha-beta produce también un valor de 3.0, lo cual demuestra que alpha-beta mantiene la calidad de la solución al mismo tiempo que reduce la cantidad de nodos evaluados.

En este algoritmo, se visitan 10 nodos en vez de 12 gracias a evitar visitar caminos innecesarios que no influyen. Esto sobre todo será más útil cuando se realice sobre árboles más grandes.

3) Conclusión:

El algoritmo MINIMAX es más exhaustivo debido a que tiene que evaluar todas las ramas del árbol.

El algoritmo PODA ALPHA-BETA está más optimizado con respecto al anterior algoritmo, ya que elimina algunos caminos que sean inútiles.