Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Inteligencia Artificial I

Prof. Blai Bonet

Proyecto 2: Árboles de Juego

**Othello**

Salcedo Andrea 10-10666

Verdugo Reinaldo 10-10757

Para este proyecto se hizo uso de los algoritmos Negamax, Negamax con podado alpha beta, Scout y Negascout para cada nodo de la variación principal del juego Othello. Usando un iterador en reversa desde el último nodo, (aquél donde ya no existe jugada posible) se contó, para cada profundidad, el tiempo en que se tomó llegar desde dicha profundidad hasta un nodo terminal, así como la cantidad de nodos generados (contados a la hora de buscar a los estados adyacentes) y expandidos (terminales) en el proceso.

Los resultados serán mostrados en cuatro tablas (una tabla por algoritmo). Cada algoritmo fue limitado a una duración de 10 minutos, por lo que el nivel de la profundidad varía para cada uno.

Las tablas de resultados pueden consultarse en el output de la corrida de cualquiera de los algoritmos en el directorio results/

**Algoritmo Negamax**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Profundidad | Valor | Tiempo | Generados | Expandidos  (hojas) |
| 0 | -4 | 2e-06 | 0 | 1 |
| 1 | 4 | 4.000000e-06 | 1 | 1 |
| 2 | -4 | 1.000000e-05 | 4 | 2 |
| 3 | 4 | 7.000000e-06 | 5 | 2 |
| 4 | -4 | 1.900000e-05 | 12 | 4 |
| 5 | 4 | 2.000000e-05 | 13 | 4 |
| 6 | -4 | 1.270000e-04 | 90 | 27 |
| 7 | 4 | 2.390000e-04 | 176 | 52 |
| 8 | -4 | 5.920000e-03 | 1048 | 305 |
| 9 | 4 | 8.426000e-03 | 4497 | 1330 |
| 10 | -4 | 1.759100e-02 | 11977 | 3381 |
| 11 | 4 | 1.131200e-01 | 76825 | 21699 |
| 12 | -4 | 6.179830e-01 | 428401 | 119923 |
| 13 | 4 | 5.027476e+00 | 3478734 | 953486 |
| 14 | -4 | 1.877060e+01 | 13078932 | 3619363 |
| 15 | 4 | 1.289262e+02 | 90647894 | 25526376 |

**Algoritmo Negamax-αβ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Profundidad | Valor | Tiempo | Generados | Expandidos  (hojas) |
| 0 | -4 | 1e-05 | 0 | 1 |
| 1 | 4 | 8.000000e-06 | 1 | 1 |
| 2 | -4 | 1.500000e-05 | 4 | 2 |
| 3 | 4 | 1.500000e-05 | 5 | 2 |
| 4 | -4 | 3.800000e-05 | 12 | 4 |
| 5 | 4 | 3.900000e-05 | 13 | 4 |
| 6 | -4 | 8.200000e-05 | 26 | 6 |
| 7 | 4 | 2.380000e-04 | 81 | 20 |
| 8 | -4 | 7.280000e-04 | 237 | 52 |
| 9 | 4 | 4.017000e-03 | 1002 | 234 |
| 10 | -4 | 4.017000e-03 | 1501 | 350 |
| 11 | 4 | 1.047900e-02 | 4067 | 900 |
| 12 | -4 | 2.349500e-02 | 9129 | 2099 |
| 13 | 4 | 2.319760e-01 | 98754 | 22734 |
| 14 | -4 | 2.939610e-01 | 127643 | 29515 |
| 15 | 4 | 5.936100e-01 | 267603 | 62587 |
| 16 | -4 | 2.850111e+00 | 1259429 | 299087 |
| 17 | 4 | 4.777300e+00 | 2031923 | 482139 |
| 18 | -4 | 6.449998e+01 | 29501797 | 7176690 |
| 19 | 4 | 9.597829e+01 | 43574642 | 10625624 |
| 20 | -4 | 2.456443e+02 | 107642870 | 25626713 |

Para el algoritmo con podado alpha-beta, se alcanza una mayor profundidad puesto que el árbol de juego a revisar es mucho más limitado. El número de nodos generados (y expandidos) por nivel es menor.

**Algoritmo Scout**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Profundidad | Valor | Tiempo | Generados | Expandidos  (hojas) |
| 0 | -4 | 1e-05 | 0 | 1 |
| 1 | -4 | 8.000000e-06 | 1 | 1 |
| 2 | -4 | 1.500000e-05 | 3 | 1 |
| 3 | -4 | 1.500000e-05 | 4 | 1 |
| 4 | -4 | 3.800000e-05 | 11 | 3 |
| 5 | -4 | 3.900000e-05 | 12 | 3 |
| 6 | -4 | 8.200000e-05 | 15 | 3 |
| 7 | -4 | 2.380000e-04 | 17 | 3 |
| 8 | -4 | 7.280000e-04 | 72 | 10 |
| 9 | -4 | 4.017000e-03 | 278 | 40 |
| 10 | -4 | 4.017000e-03 | 363 | 50 |
| 11 | -4 | 1.047900e-02 | 447 | 61 |
| 12 | -4 | 2.349500e-02 | 1038 | 147 |
| 13 | -4 | 2.319760e-01 | 2623 | 389 |
| 14 | -4 | 2.939610e-01 | 3234 | 473 |
| 15 | -4 | 5.936100e-01 | 3986 | 548 |
| 16 | -4 | 2.850111e+00 | 8766 | 1350 |
| 17 | -4 | 4.777300e+00 | 10155 | 1518 |
| 18 | -4 | 6.449998e+01 | 60899 | 9556 |
| 19 | -4 | 9.597829e+01 | 130407 | 20118 |
| 20 | -4 | 2.456443e+02 | 193404 | 29728 |

Para el algoritmo Scout, el número de nodos generados y expandidos disminuye notoriamente, debido al chequeo previo que se realiza para conocer si amerita buscar por una rama, o si es mejor simplemente podarla. Los valores de todos los niveles siempre es el mismo (-4) puesto que este algoritmo no cambia el signo dependiendo del jugador, simplemente devuelve el valor máximo o mínimo.

**Algoritmo** **Negascout**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Profundidad | Valor | Tiempo | Generados | Expandidos  (hojas) |
| 0 | -4 | 9e-06 | 0 | 1 |
| 1 | 4 | 1.400000e-05 | 1 | 1 |
| 2 | -4 | 4.000000e-05 | 4 | 2 |
| 3 | 4 | 3.000000e-05 | 5 | 2 |
| 4 | -4 | 1.080000e-04 | 19 | 6 |
| 5 | 4 | 1.080000e-04 | 20 | 6 |
| 6 | -4 | 2.000000e-04 | 33 | 8 |
| 7 | 4 | 4.960000e-04 | 83 | 20 |
| 8 | -4 | 2.626000e-03 | 397 | 86 |
| 9 | 4 | 4.726000e-03 | 1667 | 393 |
| 10 | -4 | 4.177000e-03 | 2464 | 571 |
| 11 | 4 | 6.847000e-03 | 3897 | 847 |
| 12 | -4 | 2.098700e-02 | 12066 | 2751 |
| 13 | 4 | 8.498800e-02 | 48673 | 10715 |
| 14 | -4 | 1.429530e-01 | 81825 | 18304 |
| 15 | 4 | 3.195520e-01 | 184360 | 41816 |
| 16 | -4 | 1.058994e+00 | 606377 | 140325 |
| 17 | 4 | 1.964261e+00 | 1134796 | 264869 |
| 18 | -4 | 1.248701e+01 | 7223327 | 1705777 |
| 19 | 4 | 4.445827e+01 | 25833439 | 6128742 |
| 20 | -4 | 1.067079e+02 | 62053924 | 14453750 |
| 21 | 4 | 4.084185e+02 | 242589300 | 57293208 |

El algoritmo Negascout sigue generando y expandiendo menos nodos que los Negamax, y de una manera más veloz que el algoritmo Scout. Logra ser, para el tiempo límite de 10 minutos, el algoritmo que llega al nivel más profundo.

**CONCLUSIÓN**

Usando como ejemplo la variación principal del juego Othello (aquella donde ambos jugadores son perfectos, el primero en jugar corresponde a las fichas negras, y el ganador las fichas blancas) observamos que a la hora de construir y explorar árboles de juego, Negamax queda corto como el más ineficiente en términos de espacio.

Construyendo el árbol parcial de juego para Othello con un tiempo máximo de 10 minutos, Negascout logró ser aquél en llegar a la mayor profundidad del mismo. De esto repetirse para otros juegos, Negascout es el algoritmo más recomendado a la hora de construir árboles parciales lo más completos posible.

En términos de espacio, Scout llegó a generar menos nodos, resultando ser una buena opción en casos en que se desee ahorrar memoria del sistema.En ara cada uno.juegos, Negascout es el algoritmo m