Inteligencia Artificial I - CI5437

Profesor: Carlos Infante

Integrantes:

Daniela Caballero 14-10140
Emmanuel Bandres 14-10071
Fernando González, 08-10464

Proyecto I: PSVN

Especificaciones del entorno de prueba:
- Memoria: 8 GB RAM
- Procesamiento: Intel Core i5-8300H CPU @ 2.30GHz

Introducción

Con el presente trabajo se desea llevar a cabo el estudio de la búsqueda de soluciones optimas por medio de algoritmos informados. Se implementaron los algoritmos Iterative Deepening Depth First Search, A* con eliminación retardada de duplicados e IDA* con eliminación parcial de duplicados, con el fin de realizar las búsquedas sobre los problemas planteados en el enunciado haciendo uso de las heurísticas Manhattan y PDBs respectivas para cada uno de los casos.

Arboles de búsqueda

Para este inciso del proyecto se implementó el algoritmo de Iterative Deepening Depth First Search a razón de las diversas ventajas que nos proporciona en uso de memoria durante la búsqueda dado que solo necesita almacenar la rama actual. El tiempo de corrida para cada problema fue de 15 minutos con o sin pruning.

Se pudo estudiar en esta fase que con el uso del pruning el factor de ramificación del algoritmo disminuye notablemente para cualquiera de los problemas con la salvedad del problema de 24 Puzzle.

En las torres de Hanoi, se mantienen cercanas las cifras por nivel de profundidad del factor de ramificación para el caso de pruning y sin pruning, pero la cantidad de nodos expandidos siempre fue menor en los casos en los que se aplicaba pruning.

15puzzle

Con pruning			Sin pruning		
Depth	Num Nodes	Branching	Depth	Num Nodes	Branching
0	1	2	0	1	2
1	2	2	1	2	3
2	4	2.5	2	6	3
3	10	2.4	3	18	3.22222
4	24	2.25	4	58	3.2069
5	54	2	5	186	3.23656
6	108	2.01852	6	602	3.23256
7	218	2.16514	7	1946	3.23638
8	472	2.1822	8	6298	3.23563
9	1030	2.13981	9	20378	3.23614
10	2204	2.11706	10	65946	3.23601
11	4666	2.11745	11	213402	3.23608
12	9880	2.13016	12	690586	3.23606
13	21046	2.13684	13	2234778	3.23607
14	44972	2.13311	14	7231898	3.23607
15	95930	2.1288	15	23402906	3.23607
16	204216	2.1286	16	75733402	3.23607
17	434694	2.13019	-	-	-
18	925980	2.13108	-	-	-
19	1973338	2.13083	-	-	-
20	4204856	2.13029	-	-	-
21	8957558	2.13015	-	-	-
22	19080940	2.13033	-	-	-
23	40648730	2.13047	-	-	-
24	86600984	2.13046	-	-	-
25	184499846	2.13039	-	-	-

24puzzle

Con pruning			Sin pruning		
Depth	Num Nodes	Branching	Depth	Num Nodes	Branching
0	1	2	0	1	2
1	2	3	1	2	2
2	6	3	2	4	2.5
3	18	3.33333	3	10	2.6
4	60	3.3	4	26	2.46154
5	198	3.45455	5	64	2.5
6	684	3.39474	6	160	2.325
7	2322	3.48837	7	372	2.3871
8	8100	3.42	8	888	2.24775
9	27702	3.49708	9	1996	2.44088
10	96876	3.42642	10	4872	2.3243
11	331938	3.49927	11	11324	2.43801
12	1161540	3.42803	12	27608	2.30093
13	3981798	3.49982	13	63524	2.4331
14	13935564	3.42844	14	154560	2.30225
15	47777202	3.49995	15	355836	2.43373
16	167218020	3.42854	16	866008	2.30223
-	-	-	17	1993748	2.43467
-	-	-	18	4854112	2.30341
-	-	-	19	11181004	2.43419
-	-	-	20	27216632	2.3025
-	-	-	21	62666404	2.43424
-	-	-	22	152545216	2.30284

Hanoi 4p12d

Con pruning			Sin pruning		
Depth	Num Nodes	Branching	Depth	Num Nodes	Branching
0	1	3	0	1	3
1	3	5	1	3	5
2	15	4.6	2	15	5
3	69	4.73913	3	75	5.24
4	327	4.87156	4	393	5.36641
5	1593	4.93597	5	2109	5.44666
6	7863	5.00229	6	11487	5.51711
7	39333	5.04897	7	63375	5.56615
8	198591	5.08843	8	352755	5.60826
9	1010517	5.12017	9	1978341	5.6417
10	5174019	5.14704	10	11161197	5.67028
11	26630877	5.16966	-	-	-

Hanoi 4p14d

Depth	Con pruning Num Nodes	Branching	Depth	Sin pruning Num Nodes	Branching
0	1	3	0	1	3
1	3	5	1	3	5
2	15	4.6	2	15	5
3	69	4.73913	3	75	5.24
4	327	4.87156	4	393	5.36641
5	1593	4.93597	5	2109	5.44666
6	7863	5.00229	6	11487	5.51711
7	39333	5.04897	7	63375	5.56615
8	198591	5.08843	8	352755	5.60826
9	1010517	5.12017	9	1978341	5.6417
10	5174019	5.14704	10	11161197	5.67028
11	26630877	5.16966	-	-	-

Hanoi 4p18d

Depth	Con pruning Num Nodes	Branching	Depth	Sin pruning Num Nodes	Branching
0	1	3	0	1	3
1	3	5	1	3	5
2	15	4.6	2	15	5
3	69	4.73913	3	75	5.24
4	327	4.87156	4	393	5.36641
5	1593	4.93597	5	2109	5.44666
6	7863	5.00229	6	11487	5.51711
7	39333	5.04897	7	63375	5.56615
8	198591	5.08843	8	352755	5.60826
9	1010517	5.12017	9	1978341	5.6417
10	5174019	5.14704	10	11161197	5.67028
11	26630877	5.16966	11	63287061	5.69441

TopSpin 12-4

Depth	Con pruning Num Nodes	Branching	Depth	Sin pruning Num Nodes	Branching
0	1	12	0	1	12
1	12	8.5	1	12	12
2	102	7.96078	2	144	12
3	812	7.88424	3	1728	12
4	6402	7.87441	4	20736	12
5	50412	7.87317	5	248832	12
6	396902	7.87301	6	2985984	12
7	3124812	7.87299	-	-	-
8	24601602	7.87298	-	-	-

TopSpin 14-4

Depth	Con pruning Num Nodes	Branching	Depth	Sin pruning Num Nodes	Branching
0	1	14	0	1	14
1	14	9.5	1	14	14
2	133	8.63158	2	196	14
3	1148	8.45906	3	2744	14
4	9711	8.43003	4	38416	14
5	81864	8.42617	5	537824	14
6	689800	8.42584	6	7529536	14
7	5812144	8.42585	-	-	-
8	48972249	8.42586	-	-	-

TopSpin 17-4

Depth	Con pruning Num Nodes	Branching	Depth	Sin pruning Num Nodes	Branching
0	1	17	0	1	17
1	17	11	1	17	17
2	187	9.60963	2	289	17
3	1797	9.24986	3	4913	17
4	16622	9.17477	4	83521	17
5	152503	9.1671	5	1419857	17
6	1398011	9.16948	-	-	-
7	12819040	9.1712	-	-	-

Heurísticas (Manhattan/PDB)

1. N-puzzles

- 15 Puzzle

Para la distancia de Manhattan se guardan las distancias de cada tile a cualquier otra posición en el tablero. Se implementó como arreglos que tenían esta información precalculada, con el fin de optimizar el tiempo.

Se generó una heurística PDB aditiva utilizando tres abstracciones en la que cada una se componen por el mapeo de todos los tiles menos de 5 seleccionados. Las abstracciones son disjuntas para lograr una heurística admisible y consistente. Se utilizaron las particiones vistas en la clase correspondiente.

La optimización de Manhattan mencionada anteriormente logró mejorar los tiempos de ejecución, pero al final no superó la eficiencia de la heurística de PDB aditiva.

- 24 Puzzle

Para este problema se intentaron generar las PDBs recomendadas en el paper de “Additive Pattern Database Heuristics” pero ningún miembro del equipo fue capaz de generarlas con los recursos disponibles. Se procedió a crear una PDB aditiva de 6 abstracciones de cuatro elementos cada una.

2. Top Spin

Para cada uno de los casos de prueba de este problema se implementó el máximo de dos PDB las cuales utilizan el mapeo de varias fichas por otra fija con la finalidad de simplificar el modelo del problema.

3. Torre de Hanoi

Para cada uno de los casos de prueba de este problema se implementó el máximo de dos PDB las cuales utilizan la proyección de varias variables de estados para simplificar el modelo del

problema. Para Hanoi con 12 discos se proyectaron 3 discos en una PDB y 9 en la otra, en Hanoi con 14 discos se hizo con 5 y 9 discos, y finalmente en Hanoi con 18 discos se proyectaron 8 y 10 discos respectivamente.

Algoritmos informados

Aquí se pone en práctica principalmente A* e IDA* para realizar las búsquedas sobre los problemas planteados en los enunciados con el uso de las heurísticas respectivas

1. **N-puzzles:**

15 Puzzle: Con la heurística PDB aditiva en el algoritmo IDA* hubo una satisfactoria impresión de resultados más eficiente que con la heurística Manhattan. Sin embargo, con el uso de A* no se pudo generar solución alguna a pesar de sobrepasar el tiempo de ejecución propuesto de 15min.

15puzzle		
IDA* Manhattan		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
29	98.22918	161134969.8
IDA* PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
29	78.03407	36563783.38

24 Puzzle: En este caso de prueba, no se logró obtener solución alguna con las diversas implementaciones a pesar de prolongar el tiempo de corrida por encima del pautado.

24puzzle		
IDA* PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
5	900	165329192.4

2. **Top Spin:**

12-4: Se lograron resolver sin problema alguno los problemas propuestos y con ambas implementaciones.

TopSpin (12-4)		
A* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.71868	149309.2
IDA* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.00678	1324.50

14-4: Para este caso de prueba de nuevo IDA* obtuvo una rapidez mayor en comparación a A*.

TopSpin (14-4)		
A* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.80447	112337.8
IDA* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.0035	456.50

17-4: En este caso de prueba el algoritmo IDA* fue notablemente más rápido que A* teniendo una gran diferencia en los tiempos de ejecución de ambas implementaciones.

TopSpin (17-4)		
A* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	1.55414	182231.0
IDA* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.00539	510.80

3. Torre de Hanoi:

En este problema se puede observar que IDA* funciono de forma más eficiente en comparación a A* para todas las corridas de los diversos casos de prueba, lo que se traduce en menor tiempo de ejecución para IDA* generando más nodos por segundo.

12d-4p: Se lograron mejores resultados de corrida con IDA* y la heurística Max PDB.

Towers of Hanoi (4 pegs, 12 disks)		
A* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	9.98E-05	5.1
IDA* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.0000149	3.1

14d-4p: Se aprecia también que la solución más eficiente es dada también por IDA* y la heurística Max PDB.

Towers of Hanoi (4 pegs, 14 disks)		
A* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.000293019	21.2
IDA* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.0000318	4.6

18d-4p: En este caso tenemos también que IDA* es más eficiente que A*.

Towers of Hanoi (4 pegs, 18 disks)		
A* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	0.000601535	39.6
IDA* max PDBs		
Nro. de casos	Tiempo promedio (s)	Nodos generados promedio
10	6.46941E-05	5.9

Conclusión

Con el presente ejercicio se pudo constatar que IDA* suele ser más eficiente para la mayoría de los casos por encima de A*. En otros casos, como en el problema de 24 Puzzle, no se logró una solución en un tiempo esperado con ninguno de los métodos implementados.

No se logró implementar las soluciones para el problema del Cubo de Rubik.