UNIVERSIDAD SIMÓN BOLÍVAR
DPTO. COMPUTACIÓN Y TEC. DE INF.
CI-5437 INTELIGENCIA ARTIFICIAL I

Informe Proyecto 1

Estudiantes
LEONEL GUERRERO
EROS CEDEÑO

Profesor
CARLOS INFANTE

13 de junio de 2023

Contenido General

1. Introducción

La inteligencia artificial esta ganando renombre cada vez mas. Comenzó en 1943 con la publicación del artículo «A Logical Calculus of Ideas Immanent in Nervous Activity» de Warren McCullough y Walter Pitts. En ese trabajo, los científicos presentaron el primer modelo matemático para la creación de una red neuronal. Y desde entonces ha cruzado caminos inhóspitos con avances importantes y épocas de estancamiento que sugerían el final de esta área. Sin embargo cada vez la inteligencia artificial a salido adelante y en nuestro días promete ser una realidad y tan común como lo es hoy la luz eléctrica. Y por este y otros motivos, esta asignatura cobra mayor relevancia

Para este primer proyecto de la asignatura se solicita elaborar un conjunto de algoritmos informados de búsqueda y recorrido en grafos para resolver un conjunto de juegos populares. Los juegos estudiados son: la torre de Hanoi, los sliding puzzles, el cubo de Rubik y el top spin. Cada uno de estos problemas con variaciones. Los Algoritmos implementados son A*, BFS, DFS, IDA* y IDDFS. Se solicita utilizar diversas heurísticas de búsquedas para estimar adecuadamente los costos de los caminos.

En este informe se presentan las consideraciones, resultados y observaciones obtenidas durante la elaboración de este primer proyecto de la asignatura.

2. Sliding Puzzle

2.1. ¿En que consiste Sliding Puzzle?

Un rompecabezas deslizante o rompecabezas de piezas deslizantes es un rompecabezas que reta al usuario a deslizar piezas a lo largo de rutas para llegar a una configuración determinada final. En nuestro caso la configuración objetivo es un ordenamiento correcto de los números escritos en las fichas del tablero.

2.2. Implementacion

La libreria PSVN originalmente creada en la Universidad de Alberta, proporcionada una API en C++ para modelar y consultar adecuadamente Espacios de Estados. Esta fue sumamente util para abstraer los problemas 15-puzzle y 24-puzzle.

Para obtener heuristicas adecuadas se crearon multiples PDBs aditivos y se suman para obtener un estimado del costo de los caminos. Para el 15-puzzle se agruparon las fichas del tablero en dos grupos: uno correspondiente a la mitad superior del tablero y el segundo a mitad inferior. Para el caso del 25-puzzle se dividio el tablero en 6 conjuntos disjuntos y adyacentes y se genero para cada uno de estos su correspondiente PDB.

2.3. Resultados

pruning	15-puzzleh2	15-puzzleh2	15-puzzleh0	15-puzzleh0	15-puzzleh1	15-puzzleh1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	100.0	100.0	100.0	100.0	100.0	100.0
mean	4620991.48	4652295.8	4470656.97	4561121.1	4443126.96	4565334.39
std	680884.26	681823.74	616500.82	647971.09	624659.54	671967.91
min	3916481.0	4007145.0	3759262.0	3721949.0	3281851.0	3532204.0
25	4073432.5	4155024.75	4023380.75	4113462.25	3984223.75	4061325.0
50	4311265.5	4324129.5	4200485.5	4282238.5	4162065.0	4270292.0
75	5343368.25	5424798.25	5181381.0	4620512.0	5102949.25	5287804.5
max	5829077.0	5955792.0	5797844.0	5927764.0	5749540.0	5922601.0

pruning	15-puzzleh2	15-puzzleh2	15-puzzleh0	15-puzzleh0	15-puzzleh1	15-puzzleh1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	100.0	100.0	100.0	100.0	100.0	100.0
mean	14953811.59	15055077.12	14467318.74	14760029.33	14378224.5	14773655.08
std	2203378.93	2206418.38	1995031.16	2096870.51	2021428.62	2174525.32
min	12673995.0	12967431.0	12165214.0	12044372.0	10620269.0	11430325.0
25	13181895.0	13445949.25	13019899.5	13311440.5	12893186.75	13142755.75
50	13951517.5	13993145.0	13593027.5	13857533.0	13468695.5	13818935.0
75	17291453.25	17554990.25	16767263.5	14952203.0	16513372.5	17111716.5
max	18863244.0	19273224.0	18762174.0	19182466.0	18605864.0	19165972.0

pruning	24-puzzleh0	24-puzzleh0	24-puzzleh2	24-puzzleh2	24-puzzleh1	24-puzzleh1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	4177880.5	4040630.56	4106996.08	4219368.48	4146803.82	4063523.54
std	619180.37	608345.49	626098.34	631713.6	612025.33	583018.89
min	3417672.0	3355108.0	3372255.0	3423747.0	3402546.0	3402184.0
25	3615348.75	3513449.5	3601894.75	3699288.75	3656703.0	3635880.5
50	3994659.0	3733072.5	3753228.5	3838992.5	3766000.5	3802032.5
75	4829350.5	4672384.5	4801471.5	4889829.5	4876686.25	4708202.5
max	5053954.0	5125313.0	5032459.0	5126618.0	5065743.0	5217035.0

pruning	24-puzzleh0	24-puzzleh0	24-puzzleh2	24-puzzleh2	24-puzzleh1	24-puzzleh1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	14463672.16	14000010.18	14218553.58	14616728.1	14359028.04	14079075.28
std	2136398.7	2106377.01	2159790.27	2183222.93	2116408.03	2011532.44
min	11871475.0	11615009.0	11710809.0	11877174.0	11819836.0	11778172.0
25	12498845.5	12172297.25	12495127.75	12822943.25	12666834.0	12601876.75
50	13794699.0	12949732.5	12948351.5	13291694.0	13021590.5	13183039.5
75	16744326.25	16195741.75	16602707.75	16960282.5	16896098.5	16291113.75
max	17417221.0	17714465.0	17431676.0	17727084.0	17521165.0	18034051.0

3. Torres de Hanoi

3.1. ¿En que consiste las Torres de Hanoi?

El juego, en su forma más tradicional, consiste en tres varillas verticales. En una de las varillas se apila un número n-esimo de discos que determinará la complejidad de la solución. Los discos se apilan sobre una varilla en tamaño decreciente. No hay dos discos iguales, y todos ellos están apilados de mayor a menor radio en una de las varillas, quedando las otras dos varillas vacantes. El juego consiste en pasar todos los discos de la varilla ocupada (es decir la que posee la torre) a una de las otras varillas vacantes. Para realizar este objetivo, es necesario seguir tres simples reglas:

1. Sólo se puede mover un disco cada vez.
2. Un disco de mayor tamaño no puede descansar sobre uno más pequeño que él mismo.
3. Sólo puedes desplazar el disco que se encuentre arriba en cada varilla.

3.2. Implementacion

Para las heuristicas de busquedas se siguio el consejo dado por los autores Ariel Felner, Richard E Korf y Sarit Hanan en su articulo Additive Pattern Database Heuristic y se abstrajo el problema mediante el agrupamiento de los discos en grupos disjuntos y adyacentes.

Para la version del juego con 4 pilares y 12 discos se crearon dos grupos de discos, uno

de 8 y otro de 4. Para el de 4 pilares con 14 los grupos fueron de 8 y 6. Y finalmente para la version de 4 pilares con 18 discos se dividio en 3 PDBs de 8-6-4.

3.3. Resultados

pruning	hanoi4-12h1	hanoi4-12h1	hanoi4-12h0	hanoi4-12h0	hanoi4-12h2	hanoi4-12h2
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	49.0	49.0	49.0	49.0	49.0	49.0
mean	974915.27	984759.14	1045987.53	1005340.1	956070.1	939772.51
std	1202224.63	1212488.43	1293757.38	1241400.86	1179593.48	1152766.45
min	1.0	2.0	1.0	2.0	1.0	2.0
25	49.0	62.0	49.0	62.0	49.0	62.0
50	934.0	1134.0	934.0	1134.0	934.0	1134.0
75	2243372.0	2269812.0	2254450.0	2305791.0	2205654.0	2207234.0
max	3050078.0	3001850.0	3010464.0	3110162.0	3025188.0	3067123.0

pruning	hanoi4-12h1	hanoi4-12h1	hanoi4-12h0	hanoi4-12h0	hanoi4-12h2	hanoi4-12h2
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	49.0	49.0	49.0	49.0	49.0	49.0
mean	5849327.65	5908367.29	6275761.14	6031834.12	5736256.86	5638449.04
std	7213305.66	7274898.8	7762519.49	7448328.24	7077497.89	6916560.93
min	6.0	7.0	6.0	7.0	6.0	7.0
25	271.0	330.0	271.0	330.0	271.0	330.0
50	5355.0	6485.0	5355.0	6485.0	5355.0	6485.0
75	13458131.0	13618838.0	13526671.0	13834714.0	13233895.0	13243360.0
max	18300433.0	18011052.0	18062747.0	18660906.0	18151103.0	18402662.0

pruning	hanoi4-14h2	hanoi4-14h2	hanoi4-14h1	hanoi4-14h1	hanoi4-14h0	hanoi4-14h0
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	47.0	47.0	47.0	47.0	47.0	47.0
mean	969364.55	1050374.96	997317.4	1066300.62	974170.7	948092.94
std	1135143.71	1238566.2	1174313.97	1251544.1	1146122.89	1095161.78
min	1.0	2.0	1.0	2.0	1.0	2.0
25	78.0	97.0	78.0	97.0	78.0	97.0
50	8311.0	9993.0	8311.0	9993.0	8311.0	9993.0
75	2116306.0	2170410.5	2136374.5	2220990.0	2080020.5	2191644.5
max	2813423.0	3024129.0	2889876.0	3013315.0	2923138.0	2730315.0

pruning	hanoi4-14h2	hanoi4-14h2	hanoi4-14h1	hanoi4-14h1	hanoi4-14h0	hanoi4-14h0
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	47.0	47.0	47.0	47.0	47.0	47.0
mean	5813755.94	6298847.94	5981333.04	6394653.19	5842683.94	5685871.23
std	6809310.0	7427305.87	7044089.56	7506268.92	6875398.59	6569140.38
min	6.0	7.0	6.0	7.0	6.0	7.0
25	436.0	530.0	436.0	530.0	436.0	530.0
50	49246.0	59234.0	49246.0	59234.0	49246.0	59234.0
75	12697792.0	13022423.5	12815217.5	13325891.5	12480102.0	13149770.5
max	16880509.0	18021885.0	17339221.0	18079829.0	17538167.0	16374585.0

pruning	hanoi4-18h2	hanoi4-18h2	hanoi4-18h0	hanoi4-18h0	hanoi4-18h1	hanoi4-18h1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	46.0	46.0	46.0	46.0	46.0	46.0
mean	958571.54	986835.63	902585.07	922304.22	912258.07	943574.2
std	1087144.55	1109587.7	1018233.72	1034018.41	1029928.98	1056709.84
min	0.0	0.0	0.0	0.0	0.0	0.0
25	56.25	70.75	56.25	70.75	56.25	70.75
50	13235.5	15926.5	13235.5	15926.5	13235.5	15926.5
75	1969254.5	2026692.75	1917478.25	1871888.25	1842175.25	2005592.75
max	2645792.0	2723541.0	2643894.0	2564696.0	2609363.0	2688408.0

pruning	hanoi4-18h2	hanoi4-18h2	hanoi4-18h0	hanoi4-18h0	hanoi4-18h1	hanoi4-18h1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	46.0	46.0	46.0	46.0	46.0	46.0
mean	5743303.91	5913074.17	5407564.2	5526339.07	5464957.11	5653625.87
std	6515773.68	6651555.5	6102132.15	6198690.39	6170375.0	6333891.31
min	1.0	1.0	1.0	1.0	1.0	1.0
25	311.75	379.5	311.75	379.5	311.75	379.5
50	78267.0	94188.0	78267.0	94188.0	78267.0	94188.0
75	11815507.0	12128852.75	11499691.25	11220582.25	11048900.5	12033507.0
max	15874723.0	16341188.0	15863341.0	15388116.0	15634189.0	16102924.0

4. Cubo de Rubik

4.1. ¿En que consiste el cubo de Rubik?

El cubo de rubik es un mecanismo de ejes que permite que cada una de sus caras gire independientemente, mezclando los colores, para solucionar el rompecabezas, cada una de sus caras debe estar compuesto en un solo color.

4.2. Implementación

La libreria PSVN proporciona la definicion del espacio de estados para el cubo rubik, y esta fue utilizada como estructura para recorrer el grafo permutaciones validas del cubo de rubik.

En concordancia con el articulo publicado por Richard E. Korf "Finding Optimal Solutions to Rubik's Cube Using Pattern Databases"la funcion de costo heurística consistió en crear PDBs no aditivas y calcular el valor máximo entre diversas PDBs. Inicialmente como abstracción se considero las 8 esquinas del cubo y adicionalmente 2 PDBs considerando 6 esquinas cada una. Sin embargo al ejecutar el analisis y el mapeo de los PDBs se encontro una restriccion de memoria que dificulto el calculo de las PDBs con estas caracteristicas por lo que se opto por generar una mayor cantidad de PDBs pero con menor cantidad de variables libres de tal manera que de las PDBs se simplifique aun mas. Se crearon 4 PDBs representando las esquinas del cubo y 4 PDBs adicionales que almacenan patrones relacionados a las esquinas.

4.3. Resultados

4.3.1. Cubo de Rubik 3x3

Numero de estados expandidos y generados para el cubo de rubik 3x3

pruning	rubik3h0	rubik3h0	rubik3h2	rubik3h2	rubik3h1	rubik3h1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	70.0	70.0	70.0	70.0	70.0	70.0
mean	820254.73	826808.14	823240.09	847666.94	850404.24	833730.01
std	321558.26	318588.35	326587.58	336560.77	337390.84	322163.74
min	3156.0	3343.0	3156.0	3343.0	3156.0	3343.0
25	819987.75	849430.75	815922.5	838214.75	821318.0	847298.25
50	866011.0	889922.5	879779.0	911474.5	887567.5	906138.0
75	1062164.0	918560.75	903185.0	1012592.75	1095121.25	920150.75
max	1181791.0	1223893.0	1203528.0	1235718.0	1200983.0	1228903.0

pruning	rubik3h0	rubik3h0	rubik3h2	rubik3h2	rubik3h1	rubik3h1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	70.0	70.0	70.0	70.0	70.0	70.0
mean	14764553.26	14882477.16	14818288.16	15257927.57	15307246.01	15007067.53
std	5788033.54	5734573.2	5878563.53	6058077.33	6073021.05	5798931.07
min	56809.0	60149.0	56809.0	60149.0	56809.0	60149.0
25	14759762.5	15289696.0	14686588.0	15087487.5	14783621.5	15251289.25
50	15588163.0	16018539.0	15835987.0	16406463.0	15976190.5	16310406.5
75	19118917.0	16534009.5	16257295.0	18226573.25	19712134.0	16562645.75
max	21272185.0	22029978.0	21663469.0	22242845.0	21617641.0	22120164.0

5. Top Spin

5.1. ¿En que consiste Top Spin?

El rompecabezas TopSpin fue diseñado originalmente por Ferdinand Lammertink y patentado en 1988. El rompecabezas en sí consiste en una pista ovalada que contiene N elementos. Los elementos se pueden mover en la pista cambiando todos a la vez en cualquier dirección. La única forma de cambiar el orden de los elementos es girar la parte superior del rompecabezas que resulta en invertir el orden de los 4 elementos en la rueda. El rompecabezas TopSpin es uno de los más intrigantes que se pueden colocar en el mismo estante como el cubo de Rubik al comparar posibles permutaciones. Con su estándar 20! distintas permutaciones casi supera al cubo de Rubik en complejidad pero obviamente se ve menos impresionante y no es una hazaña de ingeniería; nunca alcanzó el mismo nivel de fama. Pero resolverlo es todavía bastante complicado incluso para una computadora. El el espacio de búsqueda para este problema es demasiado grande que la memoria necesaria para encontrar una solución puede superar a la mayoría de las PC estándar. Por otro lado, encontrar una estrategia de resolución eficiente es igualmente desafiante

5.2. Implementacion

Al igual que en la mayoría de los problemas anteriores se recurrió a un PDBs no aditivas y se retorna el valor de la heurística como el máximo entre varios PDBs, pero a diferencia de las heurísticas empleadas en los problemas anteriores, para este no se realizaron proyecciones sobre las variables del dominio, sino que se opto por mapear valores de fichas a un valor constante. Se considero TopSpin 12-4, 14-4 y 17-4. Para los diferentes tamaños y tipos de tableros considerados se mapearon rangos finitos de fichas hacia la ficha cero tal que sean indistinguibles entre si y obtener una abstracción del problema original y con esta obtener estimaciones adecuadas.

5.3. Resultados

pruning	topspin124h2	topspin124h2	topspin124h1	topspin124h1	topspin124h0	topspin1
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	1355136.98	1347583.74	1313095.58	1313258.32	1327399.84	133016
std	753270.65	738837.28	722499.97	717020.0	734938.94	727924.9
min	12997.0	14181.0	12997.0	14181.0	12997.0	14181.0
25	1388413.0	1475218.5	1423919.5	1489036.0	1452815.25	146832
50	1564077.5	1592247.5	1552726.0	1566186.5	1524333.5	155861
75	1966236.75	1671739.0	1589842.25	1619921.25	1870779.5	167403
max	2177074.0	2200848.0	2106867.0	2186068.0	2209494.0	224598

pruning	topspin124h2	topspin124h2	topspin124h1	topspin124h1	topspin124h0	topspin124
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	16261608.36	16170942.86	15757105.72	15759035.6	15928763.46	15961909.9
std	9039225.61	8866024.2	8669975.95	8604214.36	8819245.03	8735073.84
min	155965.0	170142.0	155965.0	170142.0	155965.0	170142.0
25	16660912.0	17702543.25	17087005.75	17868363.5	17433742.0	17619822.0
50	18768883.0	19106903.5	18632683.0	18794177.0	18291964.5	18703254.5
75	23594794.0	20060801.5	19078081.0	19438992.25	22449310.0	20088373.5
max	26124853.0	26410056.0	25282321.0	26232751.0	26513869.0	26951732.0

pruning	topspin144h2	topspin144h2	topspin144h1	topspin144h1	topspin144h0	topspin144
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	1279156.6	1298107.16	1278722.58	1321681.7	1292655.52	1327452.32
std	567540.27	570817.64	567718.15	576670.83	570845.47	574539.85
min	69011.0	74323.0	69011.0	74323.0	69011.0	74323.0
25	1288704.25	1329132.75	1311650.5	1352346.5	1334540.75	1373036.75
50	1379715.0	1391249.0	1403879.0	1437107.5	1409326.5	1443339.5
75	1772053.0	1741319.75	1709892.25	1801079.75	1756515.25	1817228.25
max	1925126.0	1967424.0	1925236.0	1950613.0	1978183.0	1950343.0

pruning	topspin144h2	topspin144h2	topspin144h1	topspin144h1	topspin144h0	topspin144
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	17908155.84	18173417.46	17902079.08	18503463.6	18097139.9	18584250.6
std	7945544.48	7991414.91	7948032.09	8073362.16	7991813.39	8043527.23
min	966155.0	1040473.0	966155.0	1040473.0	966155.0	1040473.0
25	18041819.25	18607743.75	18363073.0	18932787.25	18683533.0	19222408.2
50	19315979.5	19477407.5	19654272.0	20119412.5	19730535.0	20206678.0
75	24808679.75	24378376.75	23938448.5	25215002.25	24591129.25	25441081.0
max	26951723.0	27543857.0	26953249.0	27308468.0	27694518.0	27304697.0

pruning	topspin174h0	topspin174h0	topspin174h1	topspin174h1	topspin174h2	topspin174
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	1067112.26	1101469.74	1004423.12	1064522.48	1026314.98	1020206.36
std	378970.66	381296.42	332155.9	362315.81	347417.85	338704.39
min	593.0	631.0	593.0	631.0	593.0	631.0
25	995575.75	1015009.75	1022711.5	1037958.75	1004056.75	1014862.0
50	1060277.0	1093595.5	1079973.5	1087268.0	1082351.0	1055692.0
75	1413655.25	1406348.0	1112375.25	1368341.5	1118007.25	1128813.75
max	1492795.0	1500253.0	1467502.0	1506725.0	1491341.0	1499909.0

pruning	topspin174h0	topspin174h0	topspin174h1	topspin174h1	topspin174h2	topspin174h
algorithm	bfs	iddfs	bfs	iddfs	bfs	iddfs
count	50.0	50.0	50.0	50.0	50.0	50.0
mean	18140874.62	18724912.34	17075161.7	18096812.26	17447313.14	17343436.1
std	6442482.62	6482019.31	5646634.11	6159351.55	5906086.75	5757957.17
min	10082.0	10710.0	10082.0	10710.0	10082.0	10710.0
25	16924780.25	17255086.0	17386058.25	17645240.75	17068934.25	17252584.0
50	18024667.5	18591045.0	18359516.5	18483480.5	18399942.5	17946702.5
75	24032085.0	23907846.0	18910359.0	23261720.75	19006090.25	19189760.5
max	25377465.0	25504229.0	24947484.0	25614258.0	25352747.0	25498377.0

6. Conclusiones

Hemos explorado el desempeño y los resultados obtenidos en durante el recorrido de espacios de estados haciendo uso de la Librería PSVN creada por la universidad de Alberta y analizamos los resultados de la aplicación de diversas heurísticas sobre rompecabezas famosos y con grandes factores de ramificación. Mediante el uso de Pattern Data Bases se pudo crear abstracciones a los distintos problemas presentados y generar funciones de estimación de costo adecuados tales que mejoran los recorridos en busca de los estados objetivos.

Junto a este informe se anexa el código fuente junto con su documentación de uso así como un conjunto de archivos csv con resultados de ejecución en las computadoras de prueba.

En definitiva, la inteligencia artificial es un recurso valioso para la resolucion de problemas. Se observo como buenas heuristicas son capaces de acercarnos mas rapidamente a nuestro objetivo y gastando menos recursos. Las posibles aplicaciones de esta tecnologia estan por doquier, desde enrutamiento de paquetes de datos en redes hasta el monitoreo y planificacion de vuelos para aerolineas. Aunque este es solo la punta del icebearg de las implicaciones de la IA ya podemos observar los beneficios que esta trae consigo.