Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-5437 - Inteligencia Artificial I
Abril - Julio 2021

Proyecto 1: Búsqueda heurística

Jesús De Aguiar 15-10360
Neil Villamizar 15-11523
Jesús Wahrman 15-11540

Problema

El objetivo del proyecto es aprender sobre el modelo de espacio de estados y sobre los diferentes algoritmos de búsqueda heurística. Para esto, se realizó un análisis y comparación de rendimiento de distintos algoritmos informados aplicados a la resolución de distintos puzzlez de manera eficiente, realizando una búsqueda sobre el grafo generado al conectar los distintos estados del problema utilizando las posibles transiciones entre ellos.

Los distintos experimentos realizados son:

- Conteo de nodos en el árbol de búsqueda comparando BFS sin eliminación de replicados y con eliminación parcial de replicados para cada uno de los problemas.
- Comparación de rendimiento entre A^* con eliminación retardada de duplicados e IDA^* con eliminación parcial de duplicado, utilizando distintas heurísticas PDB en cada uno de los problemas.

En este informe se presenta los resultados obtenidos al aplicar las distintas búsquedas informadas sobre los siguientes problemas:

- N-Puzzle: 15 Puzzle y 24 Puzzle
- Rubik's Cube: 3x3x3 cube
- Top Spin: 12-4, 14-4, 17-4
- Torres de Hanoi: 4 astas y 12, 14 y 18 discos.

Para este proyecto se hará uso del software PSVN, el cuál nos permitirá modelar los problemas, sus estados y transiciones de una manera sencilla de manera que sólo debemos ocuparnos de la implementación de los algoritmos y las heurísticas.

Este informe incluye una documentación de la impplementación de los algoritmos y de los resultados obtenidos en los experimentos.

Descripción de la Implementación

La implementación del proyecto se puede encontrar en el repositorio de GitHub neilvillamizar/proyecto-1-ci5437.

Modelo de espacio de estados

El código que contiene la implementacion del modelo de estado se puede encontrar en el directorio `src` del repositorio del proyecto. Como se explicó anteriormente, la implementación del modelo de estados se realizó con la ayuda del software PSVN con la intención de facilitar el modelo de cada uno de los problemas atacados.

PSVN utilizados

Los problemas fueron modelados utilizando los archivos .psvn que se encuentran en el directorio `src/problems/source`.

15 puzzle y 24 puzzle: Para este problema en particular se creó un programa en c++ para general el modelo psvn del n-puzzle, creando un dominio con n^2 piezas, una b y los numeros [1, n^2). Los movimientos implementados son aquellos donde el blank cambia de posición a cualquier vecino válido. Por último, el goal es b 1 2 3 ... n^{2-1}.

Top Spin: Para este caso, se utilizó el psvn que fue entregado en el enunciado del proyecto, con una pequeña modificación donde se le eliminó el ultimo elmento del vector ya que no era relevante.

Para los demás problemas se utilizaron los psvn entregados sin modificarlos.

Algoritmos

La búsqueda sobre el espacio de estados se realizó con los siguientes:

BFS
Para analizar los árboles de búsqueda, se implementó el algoritmo de BFS con y sin eliminación de duplicados. La decición de escoger este algorimo sobre otros como UCS se toma debido a que, por la naturaleza de los problemas cuyos movimientos tienen costo 1, UCS terminaría haciendo un recorrido al estilo de BFS.

Para la implementación de la eliminación de duplicados en el algoritmo, se utilizó un set de estados en el que puedes insertar, consultar y eliminar los estados visitados, utliziando la librería estandar de c++. La implementación se encuentra en el archivo `src/search_algorithms.cpp`.

A*
Se llevó a cabo una implementación directa de los algoritmos aprendidos en clase. Para lograrlo se utilizó la *priority queue* de la librería esstandar de c++. También se implementó un mapa de estados para guardar el costo de cada uno de ellos. La implementación se encuentra en el archivo `src/search_algorithms.cpp`

IDA*
Para este, se utilizó la implementación del algoritmo visto en clase que sólo hace uso de un nodo para mantener el estado actual en el recorrido, con una pequeña modificación donde este nodo se pasa como parámetro del paso recursivo del algoritmo. Para la eliminación de duplicados se uso el set de estados, de forma de que no se repita un nodo en el camino actual. La implementación se encuentra en el archivo `src/search_algorithms.cpp`

Árboles de Búsqueda

Para estudiar los árboles de búsqueda generados al recorrer el modelo de estados de cada uno de los problemas, se ejecutó el algoritmo de BFS en dos versiones, sin eliminación de estados duplicados y con eliminación parcial de duplicados, con un ejemplo en particular para cada problema, y se comparó el número de estados generados en un tiempo determinado.

En un principio, se esperaba ejecutar el algoritmo por un tiempo alrededor de los 15 minutos y mantener el conteo de los nodos generados en ese tiempo. Debido a la cantidad limitada de memoria que poseen los equipos donde los algoritmos fueron ejecutados, para alguno de estos problemas este tiempo se vio reducido drásticamente, ya que en cada uno de los ejemplos la memoria disponible era agotada pronto en la ejecucicón del recorrido.

Para mantener la integridad de la comparación, ambas versiones del algoritmo se ejecutaron por la misma cantidad de tiempo.

Resultados

Los resultados obtenidos en las distintas ejecuciones se encuentran en las siguientes tablas y gráficas

En primer lugar, se presentan los recursos utilizados en la ejecución de los algoritmos.

Problema	Búsqueda	Tiempo de Ejecucion	Memoria Utilizada	Memoria Agotada
15Puzzle	BFS	42 seg	10.7 GB	si
	BFS with Duplicate Prunning	42 seg	1.5 GB	no
24Puzzle	BFS	36 seg	11 GB	si
	BFS with Duplicate Prunning	36 seg	1.59 GB	no
Rubik 3x3x3	BFS	15 seg	11.7 GB	si
	BFS with Duplicate Prunning	15 seg	1 GB	no
Top spin 12-4	BFS	22 seg	9.5 GB	si
	BFS with Duplicate Prunning	22 seg	700 MB	no
Top spin 14-4	BFS	22 seg	9.8 GB	si
	BFS with Duplicate Prunning	22 seg	805 MB	no
Top spin 18-4	BFS	21 seg	11.5 GB	si
	BFS with Duplicate Prunning	21 seg	760 MB	no
Hanoi 4-12	BFS	22 seg	10.2 GB	si
	BFS with Duplicate Prunning	22 seg	505 MB	no
Hanoi 4-14	BFS	22 seg	10.3 GB	si
	BFS with Duplicate Prunning	22 seg	500 MB	no
Hanoi 4-18	BFS	21 seg	12 GB	si
	BFS with Duplicate Prunning	21 seg	550 MB	no

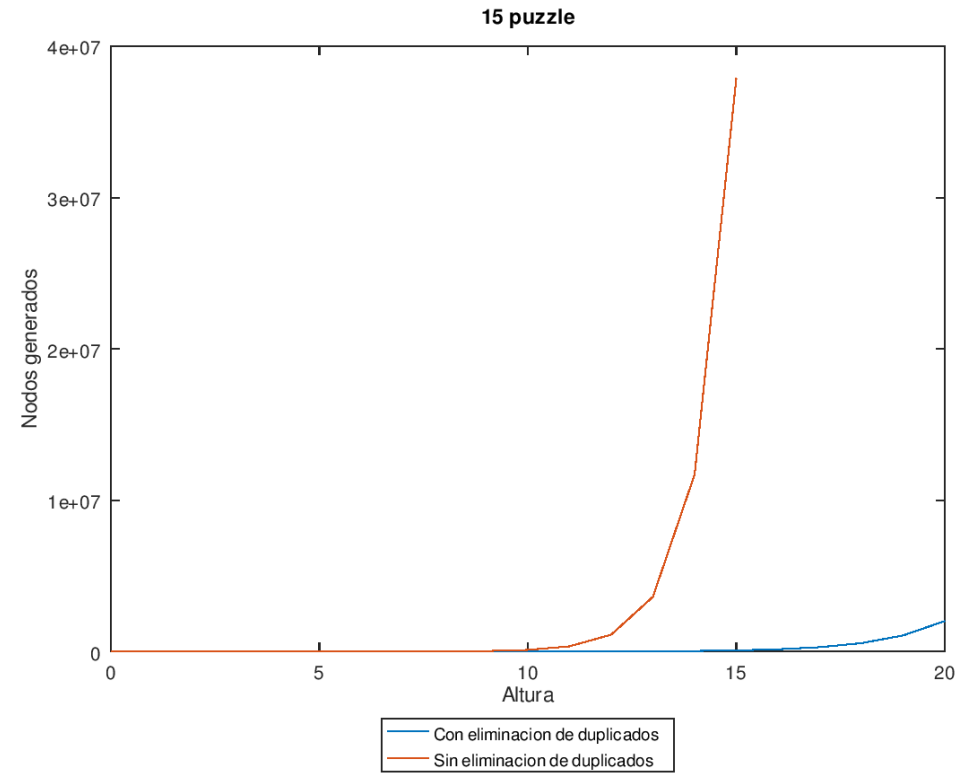
Table 1: Recursos utilizados

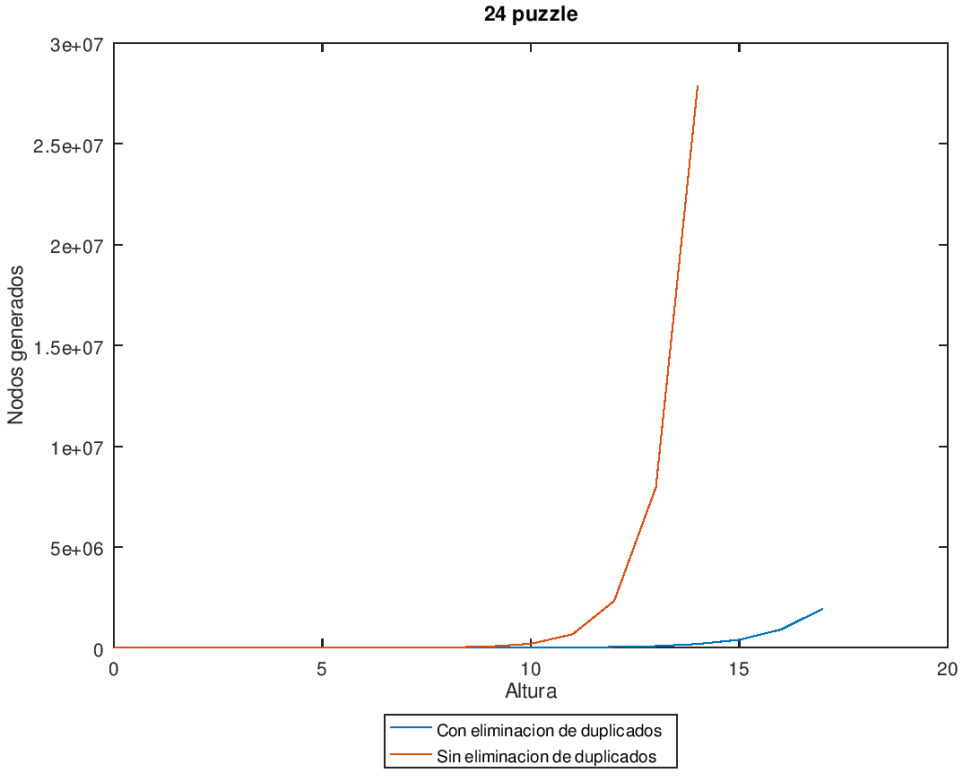
Seguidamente, se presentan los resultados para cada uno de los problemas:

n-puzzle

Capa	Número de Nodos BFS	Número de Nodos BFS + Prun	Capa	Número de Nodos BFS	Número de Nodos BFS + Prun
0	1	1	0	1	1
1	3	3	1	3	3
2	9	6	2	10	7
3	29	14	3	33	17
4	93	32	4	114	43
5	301	66	5	387	100
6	973	134	6	1350	240
7	3149	280	7	4617	544
8	10189	585	8	16146	1296
9	32973	1214	9	55323	2906
10	106701	2462	10	193590	6888
11	345293	4946	11	663633	15368
12	1117389	9861	12	2322594	35910
13	3615949	19600	13	7962867	78596
14	11701453	38688	14	27869670	181660
15	37866701	76086	15	8423149	393582
16	8865932	148435	16	-	900994
17	-	288098	17	-	1930545
18	-	554970	18	-	995997
19	-	1062628	19	-	-
20	-	2016814	20	-	-
21	-	1433710	21	-	-

Table 2: 15Puzzle y 24Puzzle

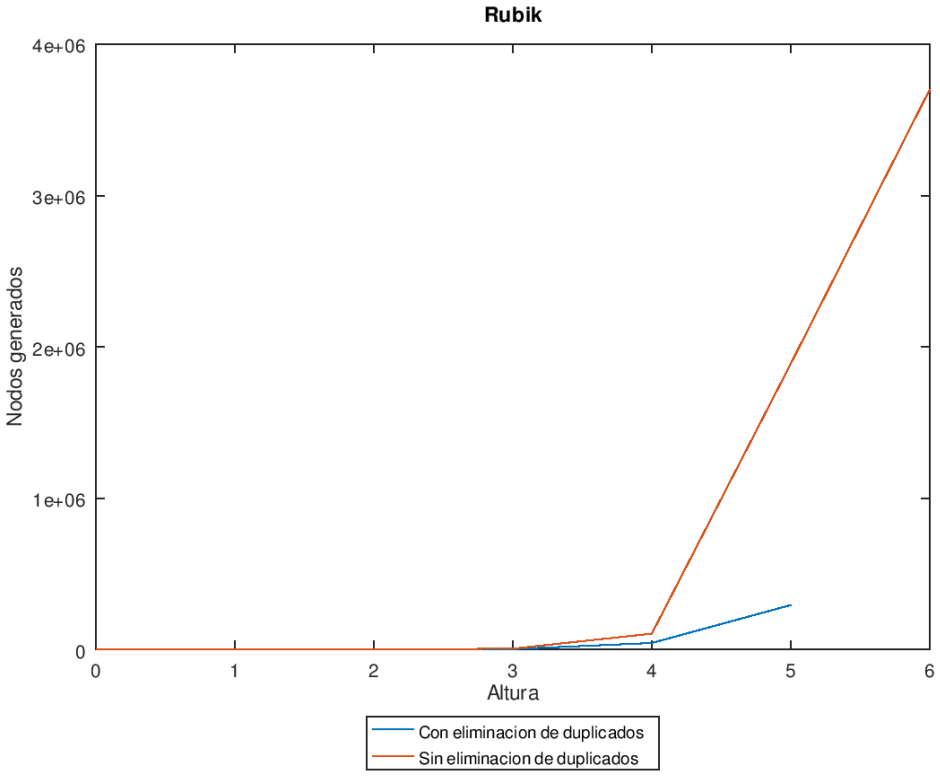

15 puzzle
4e+07
3e+07
2e+07
1e+07
0
Nodos generados
0
5
10
15
20
Altura
Con eliminacion de duplicados
Sin eliminacion de duplicados


24 puzzle
3e+07
2.5e+07
2e+07
1.5e+07
1e+07
5e+06
0
Nodos generados
0
5
10
15
20
Altura
Con eliminacion de duplicados
Sin eliminacion de duplicados

Cubo de Rubik

Capa	Número de Nodos BFS	Número de Nodos BFS + Prun
0	1	1
1	18	18
2	324	243
3	5832	3240
4	104976	43239
5	1889568	294074
6	3702014	-

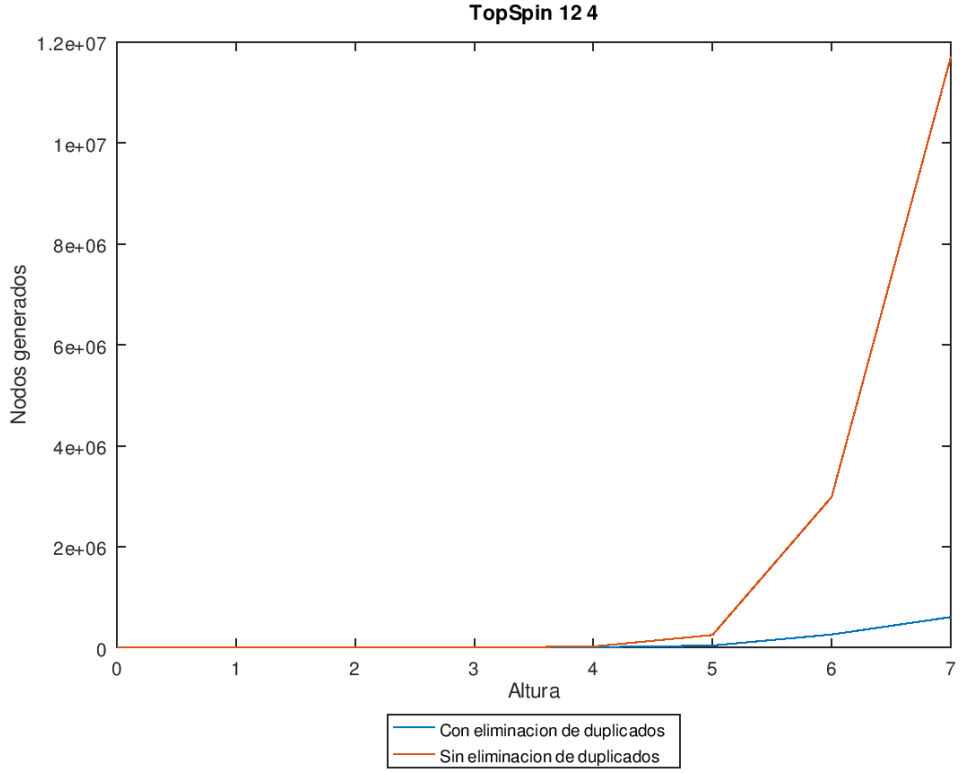
Table 3: Rubik 3x3x3

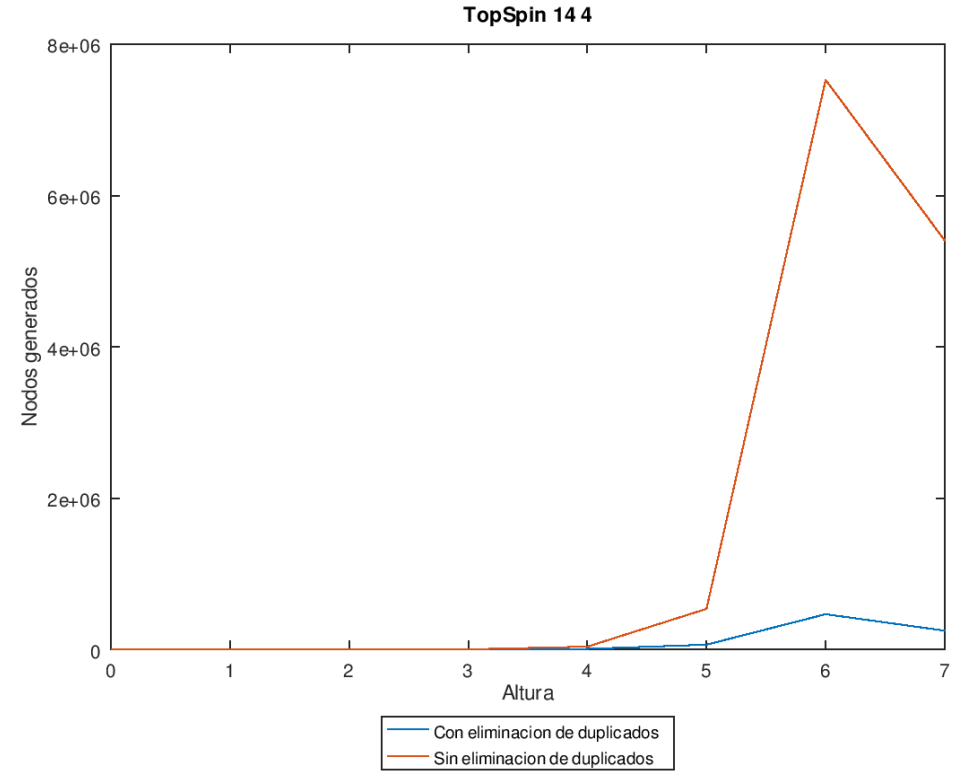


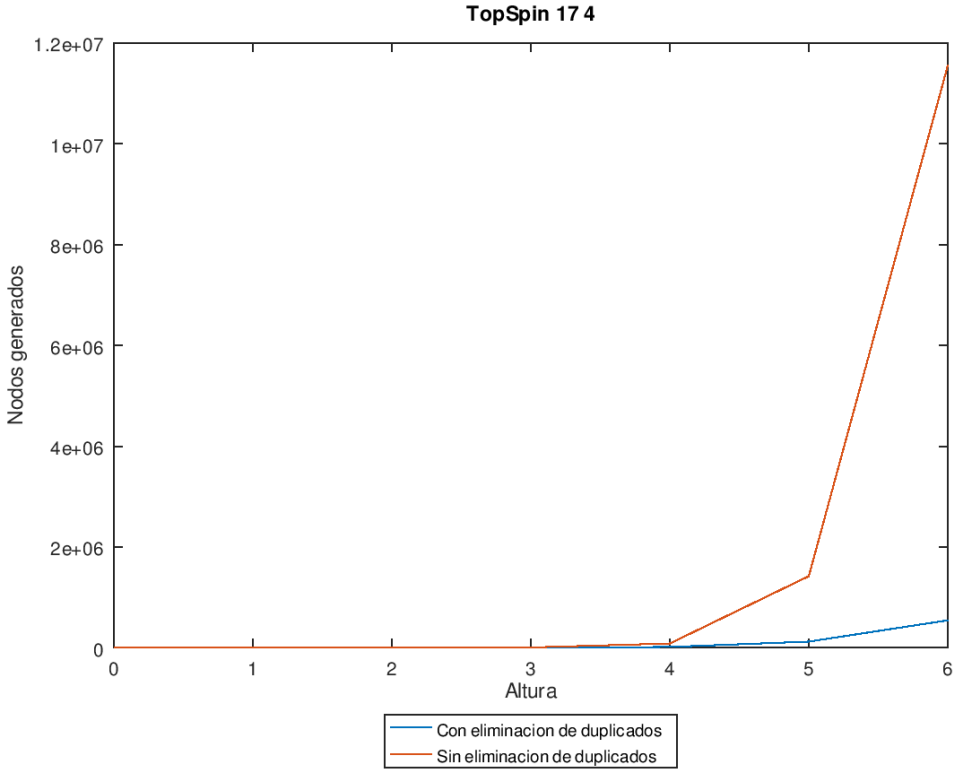
Top Spin

Capa	Número de Nodos BFS	Número de Nodos BFS + Prun	Capa	Número de Nodos BFS	Número de Nodos BFS + Prun	Capa	Número de Nodos BFS	Número de Nodos BFS + Prun
0	1	1	0	1	1	0	1	1
1	12	12	1	14	14	1	17	17
2	144	102	2	196	133	2	289	187
3	1728	784	3	2744	1106	3	4913	1734
4	20736	5725	4	38416	8631	4	83521	14841
5	248832	39990	5	537824	64722	5	1419857	121261
6	2985984	258893	6	7529536	467257	6	11556724	543539
7	11702268	604117	7	5405583	249329	7	-	-

Table 4: Top Spin: 12-4, 14-4, 17-4



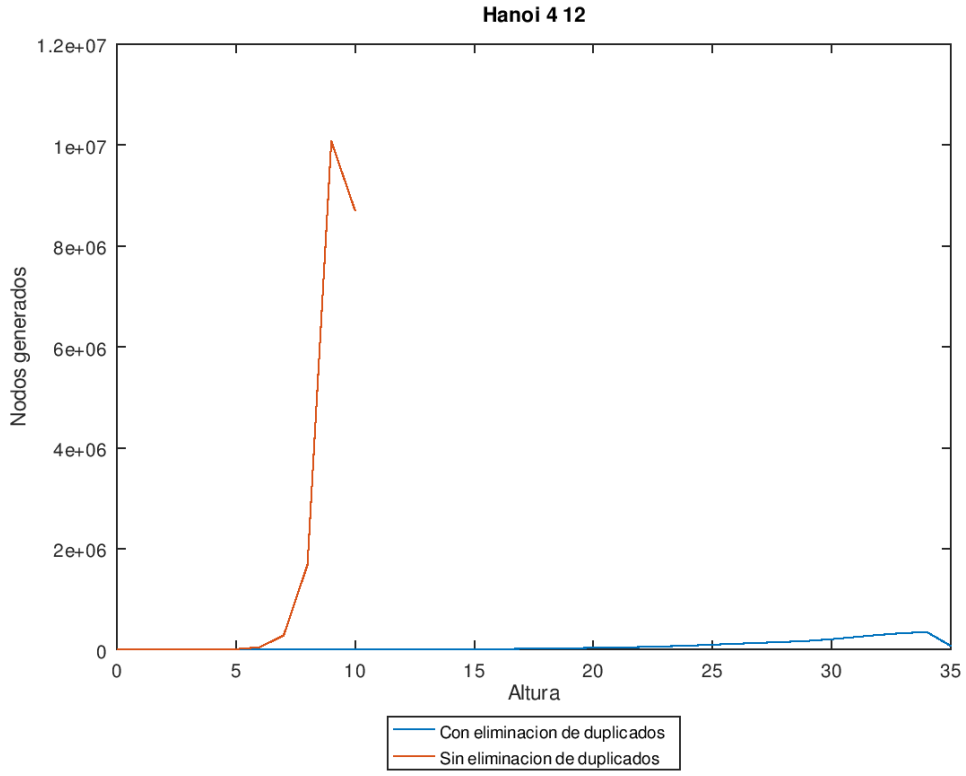

TopSpin 14 4
8e+06
6e+06
4e+06
2e+06
0
Nodos generados
0
1
2
3
4
5
6
7
Altura
Con eliminacion de duplicados
Sin eliminacion de duplicados

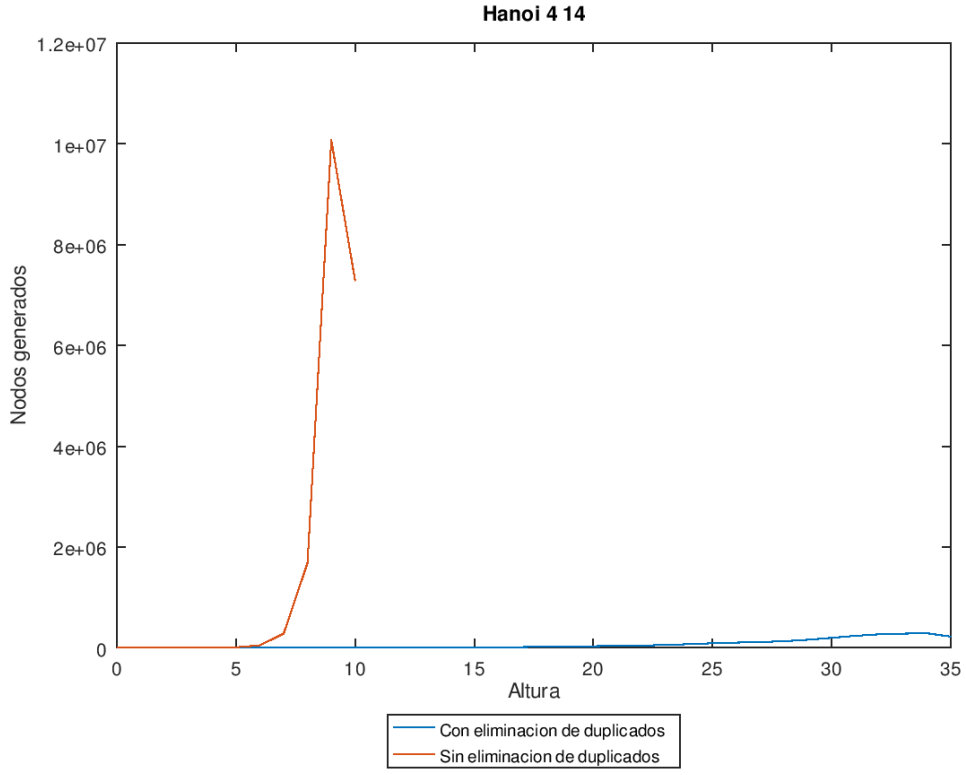

TopSpin 17 4
1.2e+07
1e+07
8e+06
6e+06
4e+06
2e+06
0
Nodos generados
0
1
2
3
4
5
6
Altura
Con eliminacion de duplicados
Sin eliminacion de duplicados

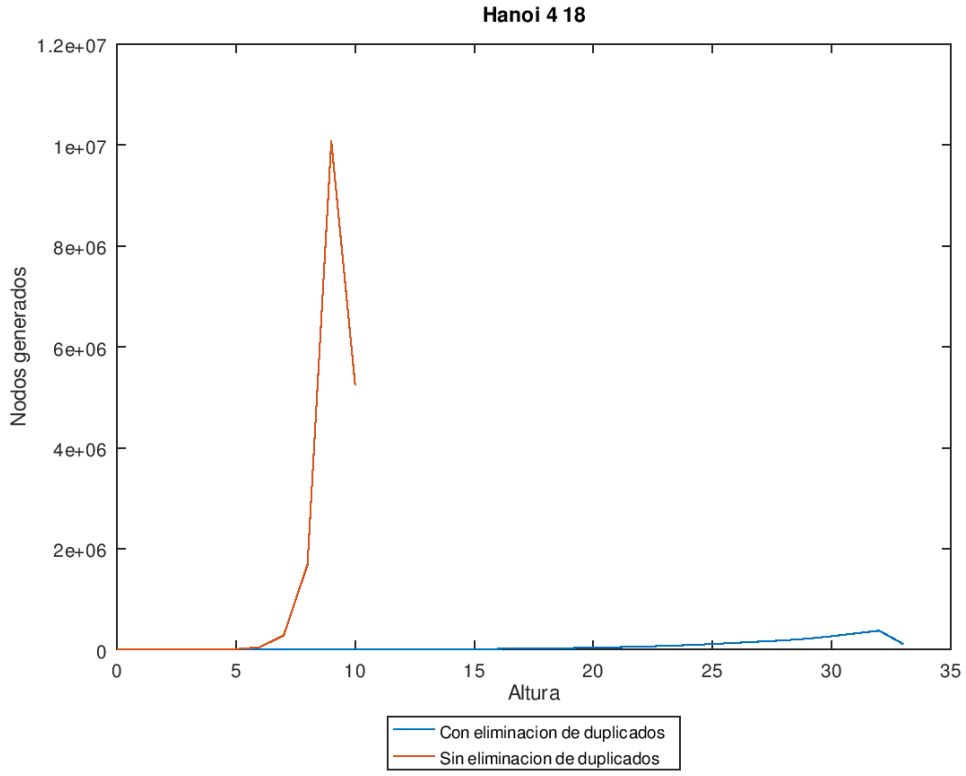
Torres de Hanoi

Capa	Número de Nodos BFS	Número de Nodos BFS + Prun	Capa	Número de Nodos BFS	Número de Nodos BFS + Prun	Capa	Número de Nodos BFS	Número de Nodos BFS + Prun
0	1	1	0	1	1	0	1	1
1	6	6	1	6	6	1	6	6
2	36	19	2	36	19	2	36	19
3	216	46	3	216	47	3	216	45
4	1296	99	4	1296	110	4	1296	94
5	7776	186	5	7776	221	5	7776	181
6	46656	324	6	46656	389	6	46656	319
7	279936	514	7	279936	640	7	279936	511
8	1679616	858	8	1679616	987	8	1679616	843
9	10077696	1148	9	10077679	1447	9	10077696	1128
10	8700115	1721	10	7284880	2051	10	5246576	1828
11	-	2712	11	-	2746	11	-	2768
12	-	3317	12	-	3511	12	-	3557
13	-	4350	13	-	4819	13	-	5339
14	-	6762	14	-	6947	14	-	7263
15	-	9587	15	-	8681	15	-	10099
16	-	10944	16	-	9897	16	-	13337
17	-	14211	17	-	12796	17	-	17567
18	-	19637	18	-	18034	18	-	22805
19	-	26867	19	-	24351	19	-	29665
20	-	33734	20	-	30380	20	-	38136
21	-	40819	21	-	34880	21	-	47305
22	-	50203	22	-	42560	22	-	58155
23	-	62828	23	-	54545	23	-	72128
24	-	77680	24	-	70345	24	-	88832
25	-	96266	25	-	88261	25	-	109414
26	-	114622	26	-	101504	26	-	133390
27	-	133498	27	-	111718	27	-	158062
28	-	150095	28	-	128607	28	-	181838
29	-	171540	29	-	157261	29	-	216213
30	-	205841	30	-	195918	30	-	263556
31	-	248953	31	-	236209	31	-	320033
32	-	290878	32	-	266180	32	-	374593
33	-	326419	33	-	279322	33	-	109751
34	-	345682	34	-	287036	34	-	-
35	-	72034	35	-	220095	35	-	-

Table 5: Torre de Hanoi: 12-4, 14-4, 18-4


Hanoi 4 12
1.2e+07
1e+07
8e+06
6e+06
4e+06
2e+06
0
Nodos generados
0
5
10
15
20
25
30
35
Altura
Con eliminacion de duplicados
Sin eliminacion de duplicados


Hanoi 4 14
1.2e+07
1e+07
8e+06
6e+06
4e+06
2e+06
0
Nodos generados
0
5
10
15
20
25
30
35
Altura
Con eliminacion de duplicados
Sin eliminacion de duplicados



El experimento arrojó un claro resultado donde se muestra la importante diferencia entre el número de estados generados cuando se permite repetir nodos en contra de cuando se prohibe. Los resultados evidencian que evitar la repetición de estados es fundamental para reducir el uso de memoria al ejecutar los algoritmos deseados.

Heurísticas

Seguidamente se procedió a implementar los algoritmos informados A* con eliminación retardada de duplicados e IDA* con eliminación parcial de duplicados, con la intención de comparar el rendimiento de estos algoritmos en la ejecución de distintos casos de prueba en nuestros problemas. Para aportarle información a estas búsquedas, se implementaron distintas heurísticas PDB específicas para cada problema. En partícular, para los N-Puzzles también se implementó la heurística de la distancia Manhattan. Los experimentos realizados se analisarán a continuación:

N-Puzzles

Distancia Manhattan. Se implementó la heurística de la distancia Manhattan, que toma tiempo lineal en el tamaño de la matriz y espacio O(1) en ser calculado. Se toman los valores de la matriz en orden y, sabiendo la posición en la que están y la posición en la que deberían estar, se logra calcular la distancia Manhattan de forma eficiente.

PDB aditivas: Para el caso de los N-Puzzles, las abstracciones utilizadas para generar los PDB consisten en una subdivisión del problema, donde se eliminan algunos de los estados del problema, se resuelven por separado ignorando aquellos estados eliminados, y luego se calcula la heurística a partir de la adición de las distintas heurísticas generadas.

Estas heurísticas fueron probadas con distintos casos de prueba. En el caso del 24-Puzzle, se generaron nuevos casos ya que los que se presentaban en el repositorio eran de gran complejidad y no podían ser resueltos por nuestros algoritmos con los recursos limitados que poseíamos.

Los resultados después de ejecutar los algoritmos A* e IDA* se encuentran en las siguientes tabas:

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con pdb's aditivas	Fácil	0.1106 seg	198516 kb	si	26773
	Media	23.8061 seg	1059232 kb	si	3375301
	Difícil	126.8676 seg	5396196 kb	si	20386946
A* con Manhattan	Fácil	1.9941 seg	39088 kb	si	127297
	Media	366.2383 seg	4685296 kb	si	18102411
	Difícil	720.000 seg	9710696 kb	no	38369627
IDA* con pdb's aditivas	Fácil	0.1954 seg	193592 kb	si	65998
	Media	103.1965 seg	505504 kb	si	22708381
	Difícil	1368.0539 seg	3841704 kb	si	244579509
IDA* con Manhattan	Fácil	4.6410 seg	15172 kb	si	310958
	Media	3600.0003 seg	1645884 kb	no	226895393
	Difícil	3600.0004 seg	3159636 kb	no	215020637

Table 6: 15 Puzzle

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con pdb's aditivas	facil	4.0325 seg	329860 kb	si	683153
	media	64.1361 seg	2905656 kb	si	8528812
	dificil	300.0000 seg	10842984 kb	no	33050877
IDA* con pdb's aditivas	facil	6.8692 seg	143956 kb	si	1466120
	media	281.8952 seg	836476 kb	si	46801616
	dificil	3600.000 seg	6438336 kb	no	485166599

Table 7: 24 Puzzle

El primer resultado que se puede extraer de las tablas es la evidencia de que la información otorgada por las heuristicas a ambos algoritmos permite una resolución del problema en un tiempo considerablemente menor, dado a que permite generar una menor cantidad de estados y explorar los camnos que se acercan de mejor manera a la meta.

Por otro lado, los resultados muestran que IDA* puede tomar un tiempo considerablemente mayor a A* en resolver el problema, pero tiene un uso mucho más eficiente en la memoria. Aún así, los ejemplos más difíciles ejecutados no pudieron ser resueltos debido al agotamiento de esta.

Rubik

Maximización de PDBs: Para el caso del cubo de Rubik, la heurística a utilizar consiste en la maximización de distintas heurísticas PDB precalculadas anteriormente.

Las abstracciones implementadas para generar estos PDBs consisten en la división de los estados del cubo de Rubik en dos grupos: Los corners y edges. Por otro lado, debido al tamaño final de las PDB con dichas abstracciones, se tomaron casos más pequeños donde se eliminaban al menos 3 colores del cubo.

En resumidas cuentas, se generaron ocho PDBs, utilizando cuatro abstracciones donde se consideraban solo las esquinas y tres colores en el problema, mapeandolos a uno de los colores preexistentes; y cuatro abstracciones donde solo se consideran los edges y tres colores.

Uno de los problemas encontrados generando las PDB fue la imposibilidad de generar una PDB utilizando 6 edges independientes. El conversor de PSVN no era capaz de generar dicha PDB.

Los resultados se muestran a continuación:

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con maximo de pdb's	facil	0.0001 seg	8523560 kb	si	4
	media	0.0026 seg	8523690 kb	si	77.5
	dificil	16.4536 seg	10577806.6 kb	si	442976
	dificil (extra)	139.43 seg	Agotada	no	NA
IDA* con maximo de pdb's	facil	0.0001 seg	8523444 kb	si	4
	media	0.005 seg	8523854 kb	si	165.5
	dificil	19.8377 seg	10061273 kb	si	675017.33
	dificil (extra)	76.7465 seg	14680924 kb	si	2711447

Table 8: Rubik

Las observaciones realizadas en el problema anterior respecto a las comparaciones entre A* y IDA* se mantienen en este problema, tardando IDA* una mayor cantidad de tiempo en lograr el objetivo. Para este problema tenemos un caso interesante donde un ejemplo no pudo ser ejecutado en A* debido al límite de memoria disponible, pero si pudo ser recorrido con IDA* debido a su manejo eficiente de memoria.

Top Spin

Maximización de PDBs: En el caso de Top Spin, las abstracciónes utilizadas para generar las PDB consistieron en dividir el problema en pequeños subproblemas y resolverlos independientemente. Para esto, se intentó realizar distintos mapeos de valores en el modelado del problema. Entre las abstracciones implementadas se encuentra mapear cada una de las posicicones en el problema a la posición correspondiente en el subconjunto de los enteros módulo n, con n 2 y 4. Luego, también se intentó descartar la mitad de las posicicones al mapear todos sus valores a un número determinado.

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con maximo de pdb's	Fácil	0.0001 seg	13664 kb	si	7
	Media	0.0076 seg	14344 kb	si	703
	Difícil	0.9197 seg	96596 kb	si	68446
IDA* con maximo de pdb's	Fácil	0.0001 seg	13652 kb	si	5
	Media	0.0119 seg	13748 kb	si	1272
	Difícil	6.0385 seg	18744 kb	si	640312

Table 9: Spin 12-4

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con maximo de pdb's	Fácil	0.0030 seg	377180 kb	si	198
	Media	0.0206 seg	378704 kb	si	838
	Difícil	1.6236 seg	516176 kb	si	94858
IDA* con maximo de pdb's	Fácil	0.0067 seg	376900 kb	si	542
	Media	0.1194 seg	377112 kb	si	838
	Difícil	10.2026 seg	380492 kb	si	903592

Table 10: Spin 14-4

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con maximo de pdb's	Fácil	0.1214 seg	281208 kb	si	6813
	Media	126.7007 seg	7400988 kb	si	6284987
	Difícil	211.25 seg	Agotado	no	NA
IDA* con maximo de pdb's	Fácil	0.3739 seg	270468 kb	si	27991
	Media	10800.0003 seg	1138012 kb	si	712898107
	Difícil	0.0620 seg	272868 kb	si	5841

Table 11: Spin 17-4

Nuevamente, se presentan resultados similares a los obtenidos en los casosanteriores. En este caso también logramos conseguir un ejemplo que no pudiese ejecutar el A* debido a las restricciones de memoria pero si pudiese desarrollar IDA*.

Torres de Hanoi

Maximización de PDBs: Por último, para las PDB en el problema de las torres de Hanoi se generó una abstracción se utilizaron proyecciónes con la intención de eliminar distintas partes del estado en distintas PDBs.

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con maximo de pdb's	Fácil	0.000066 seg	5846.66 kb	si	4.66
	Media	0.00043 seg	5824.0 kb	si	17.66
	Difícil	516.4651 seg	9610156 kb	si	16733974
IDA* con maximo de pdb's	Fácil	0.00033 seg	5804.0 kb	si	9.33
	Media	0.00083 seg	5788 kb	si	70.66
	Difícil	Limit	8860 kb	no	416739742

Table 12: Torre de Hanoi 4-12

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con maximo de pdb's	Fácil	0.000023 seg	14256.0 kb	si	25.66
	Media	0.0007 seg	14276 kb	si	74.66
	Difícil	476.09	Agotado	no	103569241
IDA* con maximo de pdb's	Fácil	0.00066 seg	14256.33 kb	si	101
	Media	0.0085 seg	14252 kb	si	1090.33
	Difícil	Limit	17260 kb	no	126579132

Table 13: Torre de Hanoi 4-14

Resultado	Dificultad	Tiempo	Memoria	Goal	Nodos Generados
A* con maximo de pdb's	Fácil	0.000013 seg	57502.66 kb	si	13
	Media	0.0019 seg	57536 kb	si	190
	Difícil	395 seg	Agotado	no	105548342
IDA* con maximo de pdb's	Fácil	0.00023 seg	57372 kb	si	22.66
	Media	0.023 seg	57416 kb	si	2703
	Difícil	Limit	60480 kb	no	309419277

Table 14: Torre de Hanoi 4-18

En principio los resultados siguen los mismos patrones que en los ejemplos anteriores, pero en este caso se presenta un ejemplo interesante, donde una de las configuraciones del problema no puede ser resuelta por el algoritmo IDA* debido a que rompe el límite de tiempo impuesto por el equipo, pero A* es capaz de resolverlo en un menor tiempo aunque utilice una mayor cantidad de