Trabajo Práctico N°1

Métodos de Búsqueda No Informados e Informados

Materia: Sistemas de Inteligencia Artificial

Grupo: 3

Integrantes: Biagini, Martín

Clozza, Nicolás

Filipic, Joaquín

Mamone, Federico

Fecha: 03/04/2019

Índice

Introducción

El presente trabajo práctico consiste en emplear un motor de inferencia y resolver un problema en particular (resolver el juego 0hh1, en este caso). Para ello se implementaron algoritmos de búsqueda desinformada y de búsqueda informada (desarrollando 2 heurísticas y una función de costo asociado), para poder compararlos mediante diferentes métricas y poder obtener conclusiones.

Todo el proyecto fue desarrollado en Java y utilizando el framework Spring.

Descripción del Trabajo

La idea en general es que el sistema sea compatible con cualquier base de conocimientos (por lo que se respetaron las interfaces provistas y se modularizó el proyecto para cumplir con este requisito), y que se pueda correr con distintos algoritmos de búsqueda dependiendo de los parámetros de entrada.

Parámetros de Entrada

Se decidió resolver esta cuestión exponiendo controladores REST, de modo que se debe levantar la aplicación ejecutando el método *main* del proyecto y realizar una invocación HTTP-POST indicando el tablero que se quiere resolver y bajo qué parámetros (todo esto se especifica en el README asociado). Los datos que deben enviarse en la invocación son los siguientes:

- Tablero del juego, que indique su dimensión y la distribución inicial de las fichas.
- El algoritmo de búsqueda mediante el cual se desea resolver el tablero, dentro de los cuales se implementaron 3 de búsqueda desinformada (DFS, BFS e IDDFS) y 2 de búsqueda informada (GREEDY y A*)
- La heurística a utilizar en caso de que el algoritmo sea GREEDY o A* (descriptas más adelante).

Como resultado de la intención de agilizar el uso del sistema, se determinó a su vez que el parámetro de la heurística no sea obligatorio, configurándose con un valor de 1 (primera heurística). Por el contrario, tanto el tablero de juego como el algoritmo a utilizar sí son obligatorios en la invocación.

Reglas y Resolución

El estado inicial (o nodo raíz, en nuestra estructura de árbol), es definido partiendo de la base del tablero pasado como parámetro (que tiene las fichas fijas, de ambos colores) y completándose cada fila de la siguiente manera (esto es, aplicar una regla): se recorre cada fila y se ubican las fichas restantes de modo que se complete la misma con igual cantidad de fichas de un color que de otro. Adicionalmente, se debe tener en cuenta que no puede haber 3 fichas consecutivas de un mismo color (restricciones del juego). Para esto, se comienza a computar las posibilidades de hacerlo y se elige la primera que cumpla las restricciones. De esta forma se obtiene el nodo inicial a partir del cual se busque la solución.

A partir de este punto y en adelante, la expansión de cada nodo y la búsqueda a través del árbol se realiza de acuerdo al algoritmo utilizado. Lo que hay que comentar, son las reglas o acciones posibles implementadas para pasar de un nodo o estado a otro: a modo de no ser reiterativo, se realiza de manera análoga a lo descrito anteriormente, pero esta vez computándose todas las posibilidades de reglas a aplicar (con las restricciones de colores mencionadas) y para todas las filas restantes. En otras palabras, cada nodo padre tendrá una cantidad de hijos definida como la suma de las reglas válidas para cada fila.

Mediante esta técnica y a través de cada algoritmo se va expandiendo el árbol. Como comprobación de que un estado sea solución o no, se verifica que no se rompa ninguna otra restricción propia del juego (que no haya filas ni columnas con una distribución de fichas idéntica).

Se le brinda una mención al algoritmo IDDFS, el que cuenta con el parámetro de profundidad. Si bien la forma óptima de implementarlo contempla que este valor sea la mitad de la dimensión del tablero inicialmente, y luego se lo vaya reduciendo a la mitad para ver si se encuentra una mejor solución, por cuestiones de practicidad, en este trabajo de va incrementando a medida que lo requiera sumando su valor actual.

Otra cuestión a destacar es que no se tienen en cuenta límites para los problemas y parámetros de entrada, aunque es obvio que dependiendo del hardware en donde se ejecute la aplicación van a aparecer errores de falta de memoria que impidan su resolución.

Función de Costo y Heurísticas

Para los algoritmos de búsqueda informada, la función de costo de ruta de un estado a otro utilizada es simplemente la adición de 1, debido a que simplemente la acción a aplicar es siempre la de modificar la distribución de fichas de una fila (en el caso de las búsquedas no informadas, el costo total será igual a la profundidad del árbol).

En cuanto a las heurísticas implementadas, se encuentra el siguiente par:

- Heurística 1: La primera heurística que implementamos, se va a fijar en 4 casos de restricciones propias del juego que no se cumplan:
    - Conjuntos de 3 o más fichas de un mismo color que estén consecutivas.
    - No igualdad de cantidad de fichas por color en una misma columna (en una fila sí se cumplirá, debido a la forma en que se va llenando el tablero y realizándose las permutaciones).
    - Existencia de columnas iguales.
    - Existencia de filas iguales.

  Tomando estos casos, para cada uno se sumará la cantidad de veces que se rompe cada restricción y se elegirá el máximo de estos valores. Teniendo en cuenta que para cada caso se necesitan realizar tantas o más permutaciones para corregir el total de veces que se rompe la restricción, se asegura que la heurística es admisible.

- Heurística 2: La segunda heurística que se implementó se fija de manera similar a la primera las restricciones del juego no cumplidas (con la excepción de que acumula las filas/columnas que se encuentran repetidas), pero esta vez hace una sumatoria de ellas y devuelve el total de ellas. Nótese que existen casos en donde una permutación de fila, es decir, la aplicación de una regla para cambiar de estado, puede llegar a resolver más de un conflicto, por lo que el valor de costo de esta heurística puede llegar a ser mayor al menor costo posible de alcanzar el objetivo. Esto hace que esta heurística no sea admisible. Sin embargo, se decidió optar por su implementación y uso debido a que empíricamente observamos que no es erróneo pensar en que un estado con mayor cantidad de restricciones incumplidas esté más lejos o tenga un mayor costo de llegar al objetivo.

Análisis de Resultados

Para obtener los resultados, anotando las métricas exigidas en el trabajo, se ejecutó la aplicación bajo iguales condiciones de hardware para cada algoritmo. Cabe destacar que el tiempo de ejecución se considera desde que comienza a correr el algoritmo hasta llegar a una solución o no, en caso de que no sea posible (no se contempla el tiempo de respuesta de invocación, el cálculo del estado inicial, etc.), y fue tomado como un promedio de 5 ejecuciones.

Los resultados arrojados por el sistema se contemplan en la *tabla1*, *tabla2*, *tabla3*, y *tabla4* (dispuestas en el anexo) todos los algoritmos. Son resultados pasando distintos tableros iniciales como entrada. En donde no se completaron datos, el sistema lanzó una excepción por falta de memoria, por lo que no se terminó de ejecutar el algoritmo.

A modo de analizar los resultados visualmente, se dispusieron los siguientes gráficos con las métricas más relevantes para comparar los algoritmos:
El *gráfico1*, *gráfico3*, *gráfico5* y *gráfico7* muestran, para los distintos tableros, la cantidad de nodos expandidos y nodos explorados de cada algoritmo. Esto es un parámetro muy importante ya que aquel que tenga menor cantidad, habrá llegado a la solución formando un árbol de búsqueda más chico.

El *gráfico2*, *gráfico4*, *gráfico6* y *gráfico8* muestran, en cambio, el tiempo que tardaron los algoritmos en llegar a la solución. Esto en algunos casos se correlaciona con la métrica mencionada anteriormente, aunque no siempre, y permite obtener una segunda variable a tener en cuenta a la hora de elegir un algoritmo, debido a que el tiempo de resolución es un dato nada menor.

Conclusiones

Se pueden obtener varias conclusiones a partir de los datos obtenidos.

En primer lugar, analizando los algoritmos de búsqueda no informados, en 3 de los 4 (no con el Tablero 3) casos el IDDFS presenta una menor cantidad de nodos expandidos que BFS y DFS, si bien esta ventaja no se ve necesariamente reflejada en el tiempo de ejecución.

Observando los algoritmos de búsqueda informada, indudablemente A* es sustancialmente más eficiente que Greedy, en cuanto a todas las métricas, e independientemente de la heurística utilizada. Con respecto a esto último, la heurística 1 que se utilizó, que fue la admisible, cuenta con una visible ventaja frente a la heurística 2, tanto en nodos expandidos como en tiempo.

Finalmente y a modo global, hay que mencionar que el algoritmo Greedy para nuestro juego resultó igual, o en algunos casos peor, de ineficiente que BFS y DFS. Sin embargo, y si bien IDDFS tuvo una buena performance (compartida con DFS en el caso del Tablero 3), es evidente que A* es el mejor de los algoritmos implementados, y de aquí se ve la importancia del uso de las heurísticas (y su correcta adecuación al problema en cuestión) y las funciones de costo para poder llegar a la solución de manera óptima, con menos esfuerzo y en menor tiempo.

Anexo

TABLERO 1 (4x4)							
Algoritmos de Búsqueda	**BFS**	**DFS**	**IDDFS**	**Greedy (H1)**	**Greedy (H2)**	**A* (H1)**	**A* (H2)**
Costo Nodo Solución	2	17	2	15	15	2	2
Nodos Expandidos	9	17	2	15	15	2	7
Profundidad Nodo Solución	2	17	2	15	15	2	2
Nodos Explorados	15	41	9	42	39	3	8
Nodos Frontera	49	79	6	64	67	12	42
Solución Encontrada	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Tiempo (us)	1.837	195	149.766	391,9	566,3	161,1	2.086

Tabla1

TABLERO 2 (6x6)							
Algoritmos de Búsqueda	**BFS**	**DFS**	**IDDFS**	**Greedy (H1)**	**Greedy (H2)**	**A* (H1)**	**A* (H2)**
Costo Nodo Solución	3	292	4	731	735	4	3
Nodos Expandidos	334	292	151	732	739	5	104
Profundidad Nodo Solución	3	292	4	731	735	4	3
Nodos Explorados	1161	885	3279	3867	2990	6	334
Nodos Frontera	6188	5540	45	12238	13269	105	1955
Solución Encontrada	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Tiempo (us)	16.290	22.936	10.192	96245,2	121555	411,9	30733,1

Tabla2

TABLERO 3 (4x4)							
Algoritmos de Búsqueda	**BFS**	**DFS**	**IDDFS**	**Greedy (H1)**	**Greedy (H2)**	**A* (H1)**	**A* (H2)**
Costo Nodo Solución	3	9	4	48	48	3	3
Nodos Expandidos	42	9	13	48	48	4	23
Profundidad Nodo Solución	3	9	4	48	48	3	3
Nodos Explorados	133	21	77	118	117	5	39
Nodos Frontera	162	43	16	219	220	24	123
Solución Encontrada	Sí	Sí	Sí	Sí	Sí	Sí	Sí
Tiempo (us)	2.303	155	175	840	1643	465	2317

Tabla3

TABLERO 4 (6x6)							
Algoritmos de Búsqueda	**BFS**	**DFS**	**IDDFS**	**Greedy (H1)**	**Greedy (H2)**	**A* (H1)**	**A* (H2)**
Costo Nodo Solución	5	3129	8	-	7619	6	5
Nodos Expandidos	6377	3129	2159	-	8214	14	3287
Profundidad Nodo Solución	5	3129	8	-	7619	6	5
Nodos Explorados	101152	7619	62457	-	42127	16	25921
Nodos Frontera	83782	83123	157	-	196080	391	69403
Solución Encontrada	Sí	Sí	Sí	No	Sí	Sí	Sí
Tiempo (us)	359.524	115.333	198.008	-	1396376	2450	760830

Tabla4

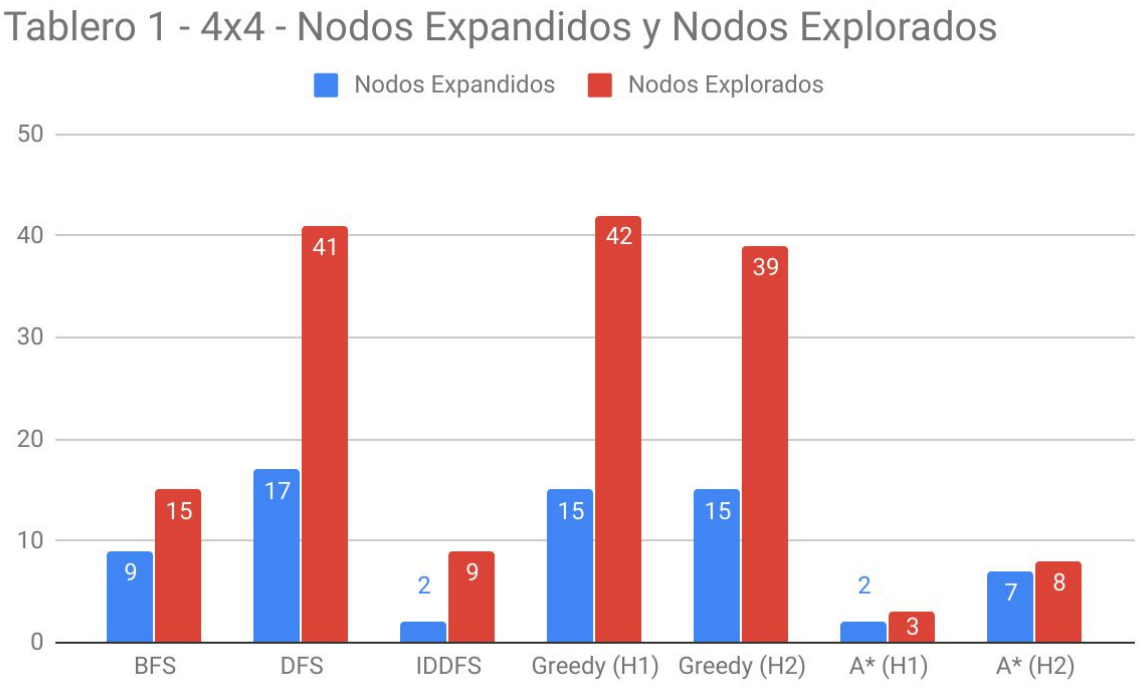

Gráfico1

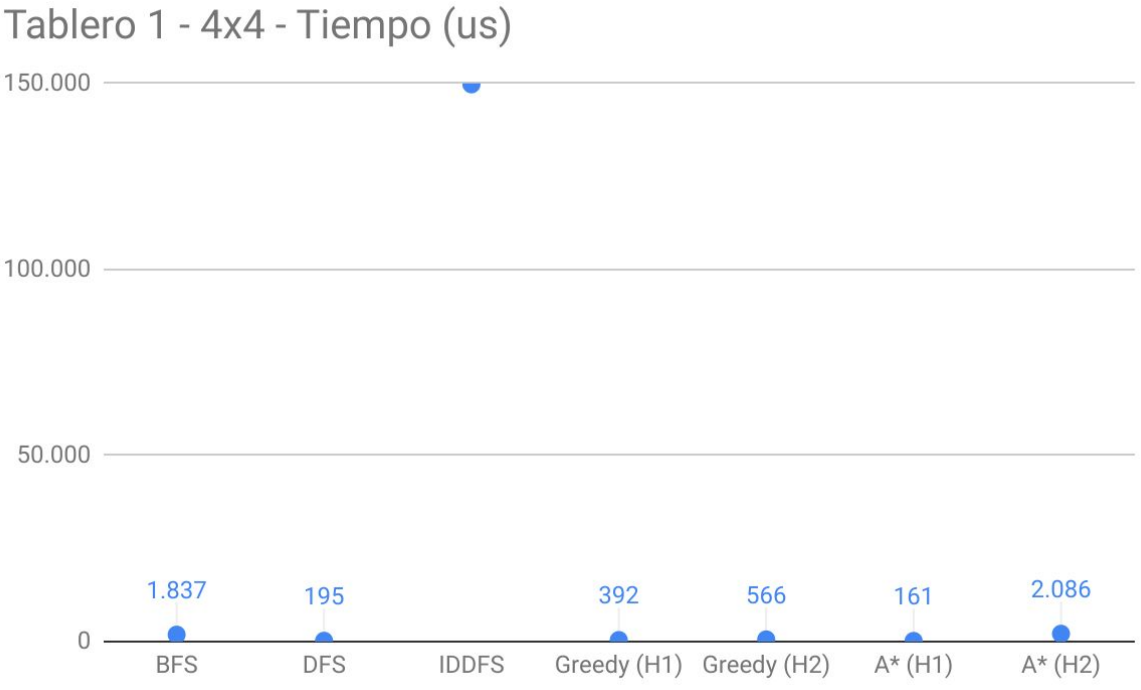

Gráfico2

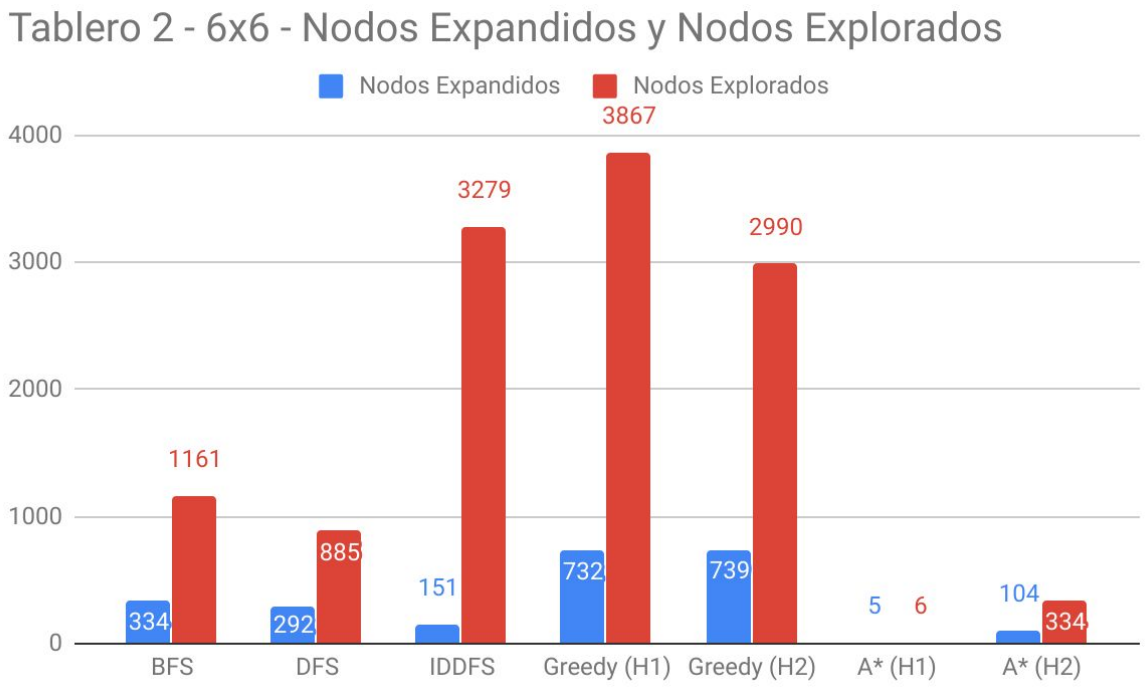

Gráfico3

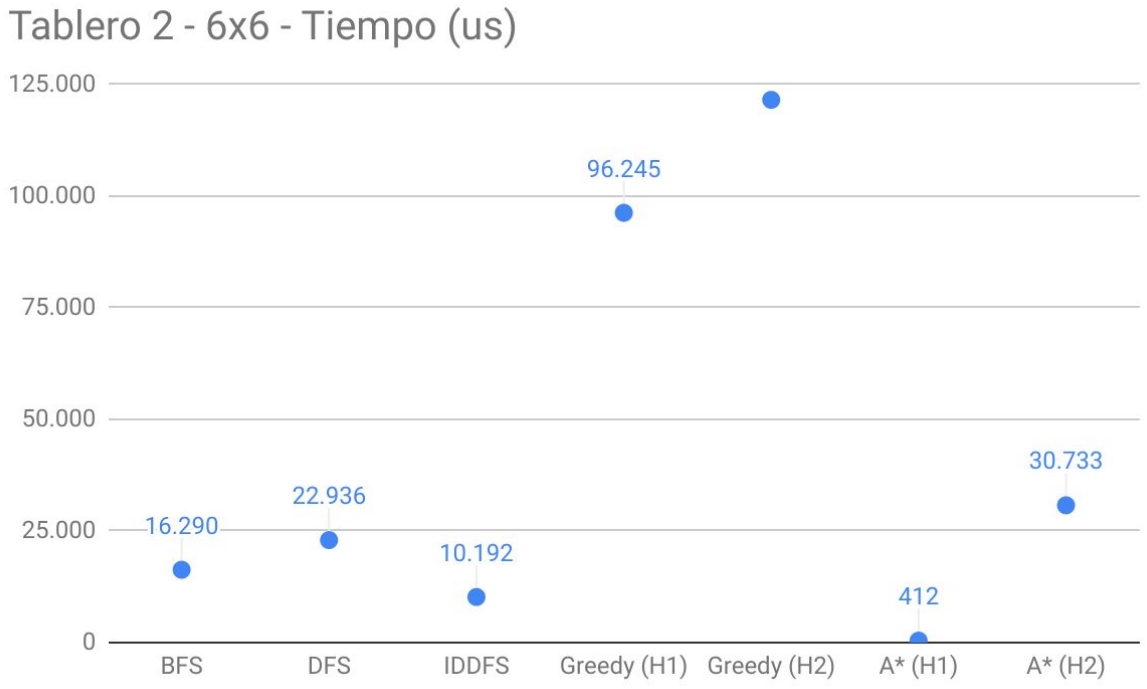

Gráfico4

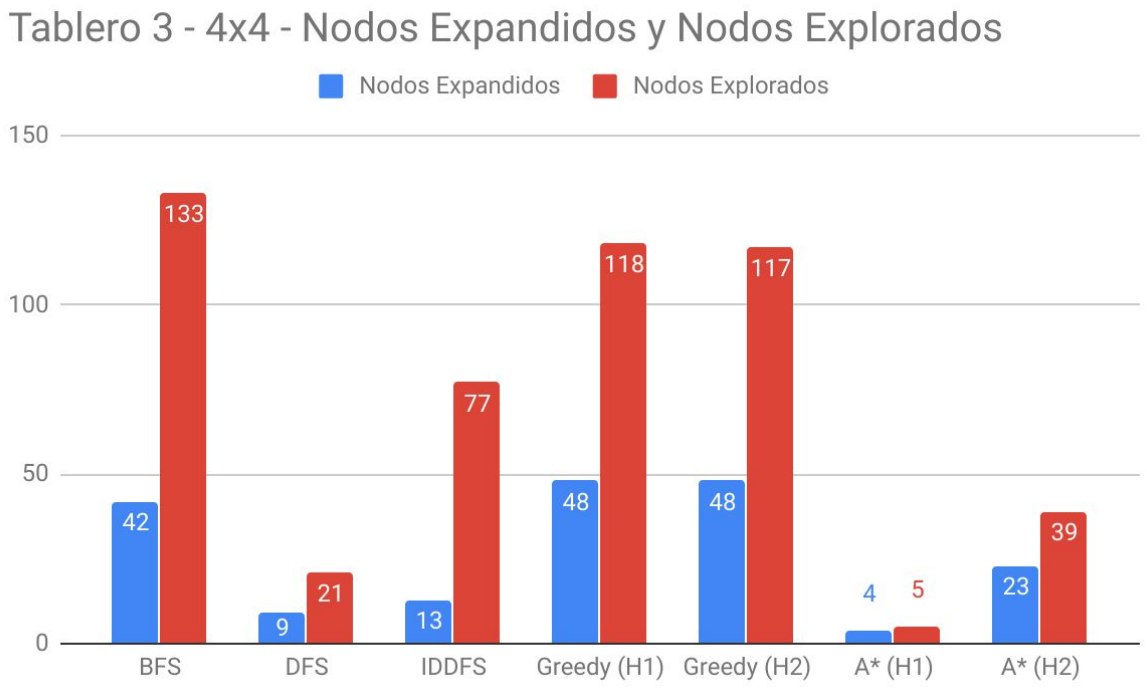

Gráfico5

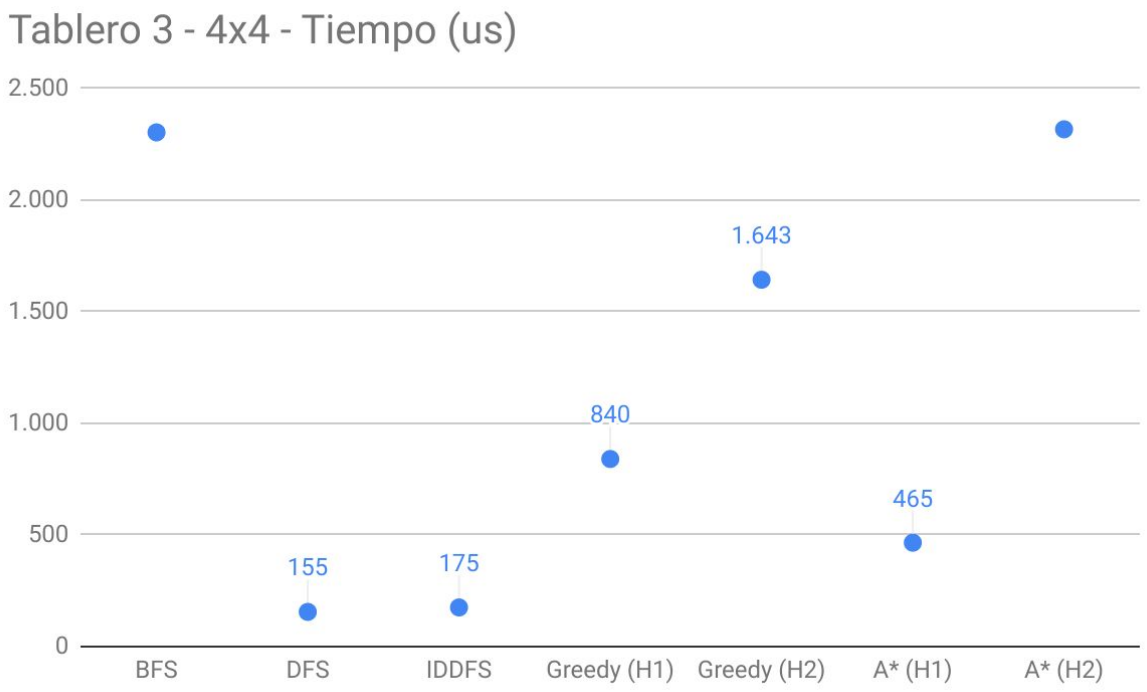

Gráfico6

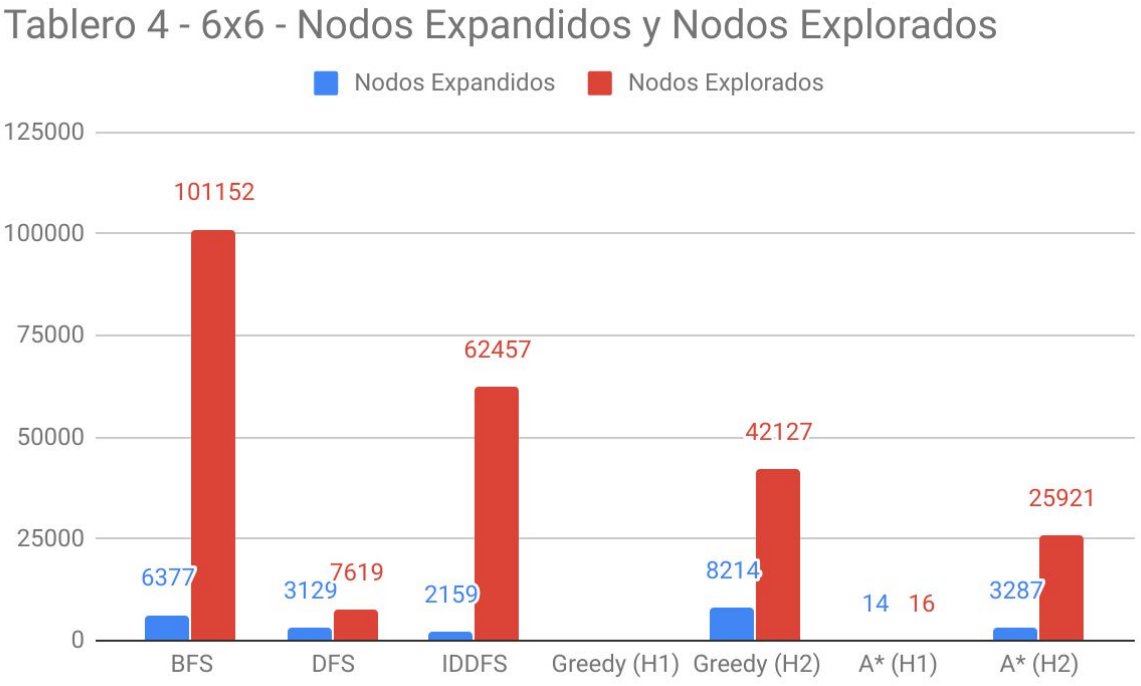

Gráfico7

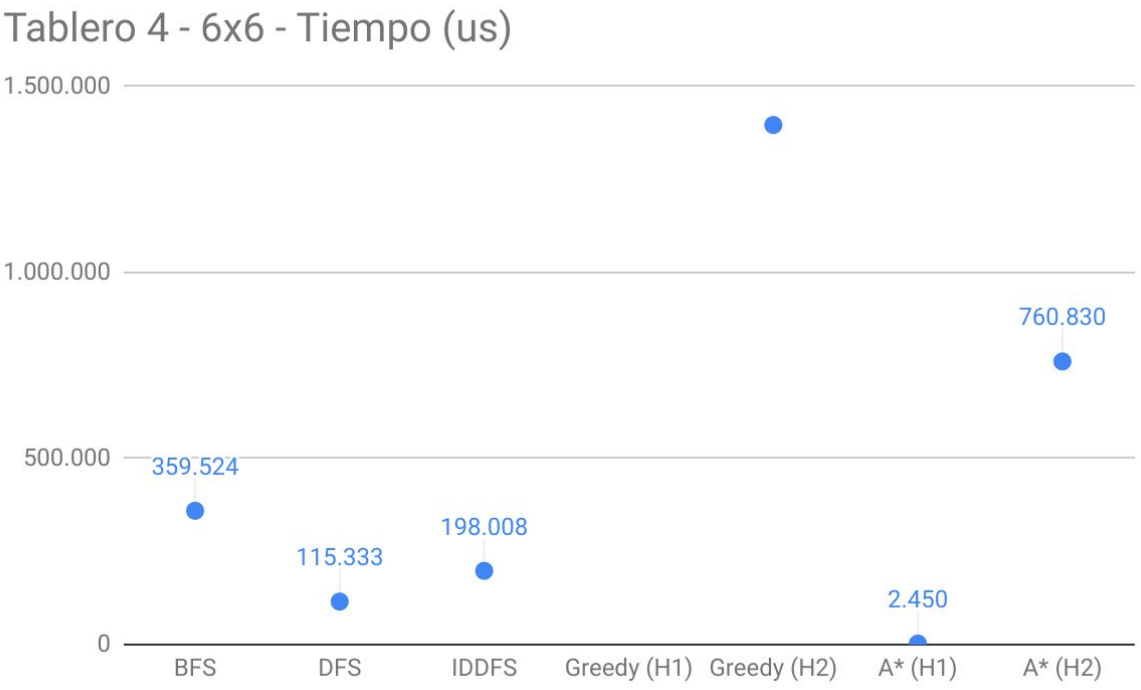

Gráfico8