Universidad Autónoma de Nuevo León

Facultad de Ciencias Físico Matemáticas

Producto Integrador de Aprendizaje:
Mundo de los Bloques

Inteligencia Artificial

Docente: Juan Pablo Rosas Baldazo

Grupo 031

Integrantes:

- José Raúl Evangelista Mendoza 1810806
- José Santos Flores Silva 1851125
- Luis Ángel Martínez Trejo 1941437
- Andrés Isaac Montes Bartolo 1854017
- Ángel Tejeda Tiscareño 1851833

A 29 de mayo del 2021
San Nicolás de los Garza, Nuevo León.

Planteamiento del Problema

El mundo de los bloques consiste en lo siguiente:

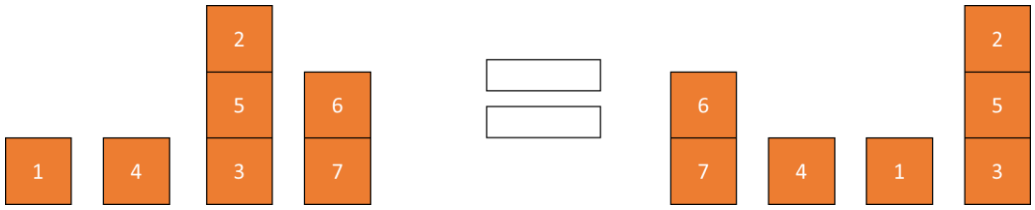
Se tienen n bloques diferentes, todos distinguibles entre sí, los cuales se pueden apilar sobre una superficie. Para esta variante se considera que hay espacio infinito en la superficie, por lo que los bloques se pueden acomodar en un número infinito de pilas, aunque es posible restringir este valor. Los bloques se colocan uno directamente encima del otro, esto quiere decir que un bloque no puede estar apoyado en dos pilas.

Una garra mecánica puede tomar el bloque en el tope de cualquiera de las pilas y colocarlo en otra pila o sobre la mesa, creando una nueva pila. Sólo un bloque se puede mover a la vez.

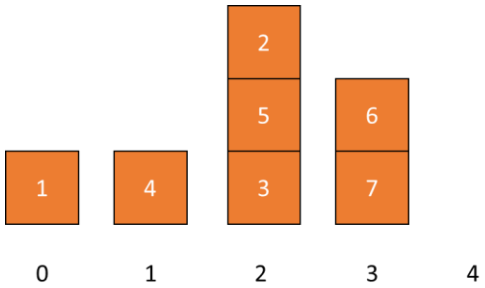
Estados

Un estado está definido por el número de pilas, los bloques que contiene cada una de esas pilas y en qué orden se encuentran apilados los bloques.

Para la resolución de este problema no importa el orden en que se definan las pilas. Por ejemplo, los estados que se muestran a continuación se consideran iguales debido a que las pilas en ambos estados tienen la misma configuración.



Las pilas se numeran de izquierda a derecha empezando por el 0, y siempre se considera una pila vacía al final que no contribuye al número de pilas del estado. Por ejemplo, se considera que el siguiente estado tiene 4 pilas a pesar de que se muestre numerada una quinta pila vacía.



Estado Inicial

Cualquier configuración que indique cómo se encuentran dispuestos los bloques al inicio del problema.

Estado Final

Cualquier configuración que indique cómo deberán encontrarse dispuestos los bloques al finalizar el problema.

Tamaño de Estados

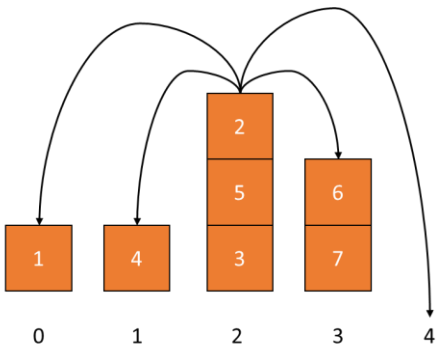
Considerando que hay n bloques sobre la mesa, estos se podrían colocar en una pila de n bloques o en n pilas de dos bloques. De esta forma podemos imaginar una matriz de $n \times n$ en la que podemos colocar los bloques en cualquier orden. El espacio de estados será menor a maneras de colocar los n bloques formas de colocar los bloques en una matriz de $n \times n$ porque no consideramos los casos inválidos en que los bloques quedan flotando, ni que el orden de las columnas no importa.

Si $N(n)$ es el tamaño del espacio de estados para n bloques, entonces:

$$N(n) \leq \frac{n^2!}{(n^2 - n)!}$$

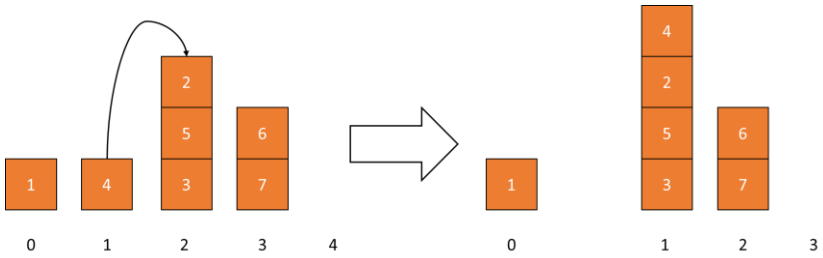
Función de Transformación

Dado un estado válido, se le puede aplicar la función de transformación $m(p_1, p_2)$ que significa tomar el bloque en el tope de la pila p_1 y colocarlo sobre la pila p_2.



En este ejemplo en el que $p_1 = 2$ se muestran los posibles movimientos que se pueden realizar.

Si una pila tiene únicamente un bloque, y este se mueve a otra pila, la numeración se recorre, por ejemplo:



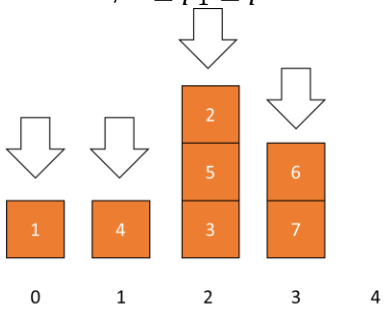
Función de Costo

Cada movimiento (es decir, mover un bloque de una pila a otra) tiene un costo de 1.

Condiciones de Aplicabilidad

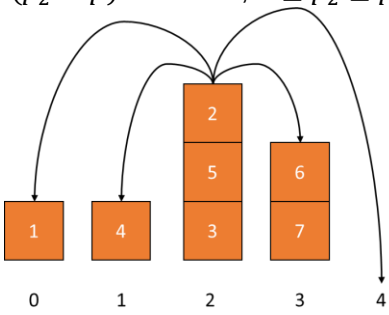
Para que se pueda aplicar la función $m(p_1, p_2)$ a un estado que cuenta con p pilas se deben de cumplir las siguientes condiciones:

- Para p_1 se debe seleccionar una pila que contenga bloques. La pila vacía p no puede seleccionarse. Dicho de otra forma, $0 \leq p_1 \leq p - 1$.



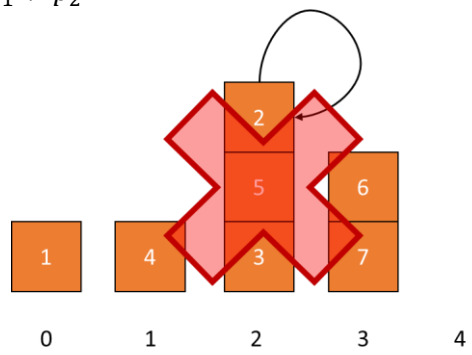
Para este ejemplo, $0 \leq p_1 \leq 3$.

- El bloque de la pila p_1 seleccionada debe moverse a una de las pilas existentes $(0 \leq p_2 \leq p - 1)$ o a una nueva $(p_2 = p)$. Es decir, $0 \leq p_2 \leq p$.

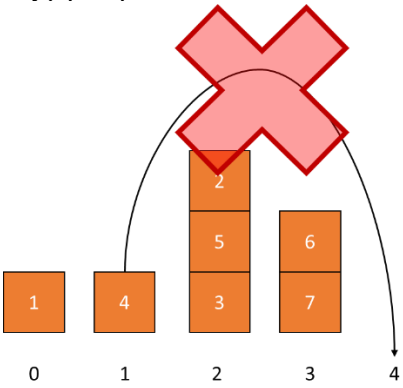


En este ejemplo $0 \leq p_2 \leq 4$.

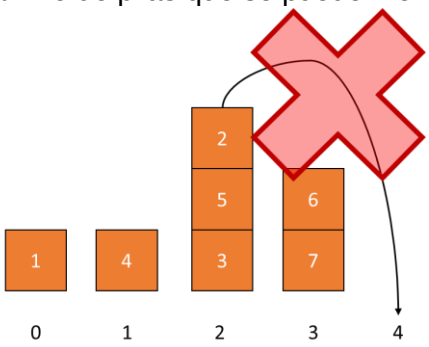
- No se puede tomar un bloque y colocarlo en la misma pila donde se encontraba. Dicho de otra forma, $p_1 \neq p_2$.



- Si la pila p_1 tiene solamente un bloque, éste no se puede colocar nuevamente sobre la mesa (es decir, en la pila p) porque nos llevaría al mismo estado.



- Si se establece que como mucho puede haber P pilas, para poder mover un bloque a la pila p se debe cumplir que $p < P$, es decir, el número de pilas del estado es menor que el número máximo de pilas que se pueden formar.



En este estado hay 4 pilas. Si en la definición del problema se establece un límite de 4 pilas, entonces el bloque no podrá ser movido a la pila 4 porque el estado resultante tendría 5 pilas.

Factor de Ramificación

- Para un estado con p pilas hay p bloques que se pueden seleccionar para moverse.
- Para cada uno de esos bloques hay p posibles lugares donde se puede colocar (las $p-1$ pilas existentes donde $p_1 \neq p_2$ y una nueva pila sobre la mesa si $p_2 = p$).

Esto significa que, en general, un estado tiene un factor de ramificación de p^2, aunque hay estados en los que este número es menor debido a que algunas pilas constan de un único bloque, por lo que la opción de colocarlo sobre la mesa no aplica ya que nos llevaría al mismo estado. Este número también podría ser menor si se aplica una restricción en cuanto al número de pilas que puede haber.

Método de Solución

En estos problemas se pretende encontrar una secuencia de movimientos que nos permitan llegar del estado inicial al estado final en el menor número de pasos posible. Para hacer esto utilizaremos dos algoritmos de búsqueda: uno informado y uno desinformado.

Algoritmo Informado: A*

En estos problemas se pretende encontrar una secuencia de movimientos que nos permitan llegar del estado inicial al estado final en el menor número de pasos posible. Para hacer esto utilizaremos dos algoritmos de búsqueda: uno informado y uno desinformado.

Se tiene que la función para el algoritmo de búsqueda A* es:

$$f(n) = g(n) + h(n)$$

Donde:

$f(n)$: es el costo estimado de la mejor solución para cierto nodo n.

$g(n)$: es el costo acumulado desde el nodo inicial hasta el nodo n.

$h(n)$: es el costo estimado desde el nodo n hasta el objetivo (la heurística)

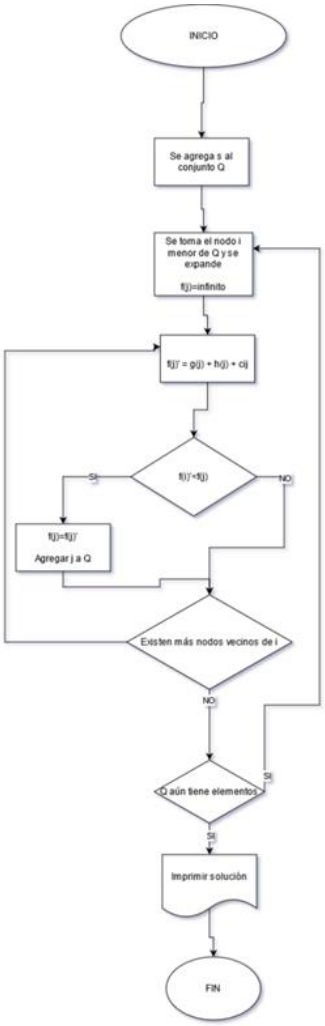
El problema tiene solución óptima siempre y cuando la heurística sea admisible, esto es, que la heurística no sobreestime al costo real para todo nodo.

El método A* es un algoritmo informado, ya que cuenta con la función heurística para determinar la mejor solución.

Algoritmo:

1. Se establece un nodo como objetivo y otro como el inicial, se calcula $f(s)$ y se agrega a un conjunto vacío
2. Se calcula para cada nodo vecino del nodo del conjunto que tenga menor costo, esto con la función
$$f(j)' = g(j) + h(j) + costo_{i,j}$$
Se inicia con $f(j) = \infty$
3. Si $f(j)' < f(j)$ se hace $f(j) = f(j)'$ y se agrega j al conjunto.
4. Si existen más nodos vecinos de i, volver al paso 3
5. Si el conjunto aún tiene nodos por analizar, volver al paso 2, de lo contrario terminar el algoritmo.

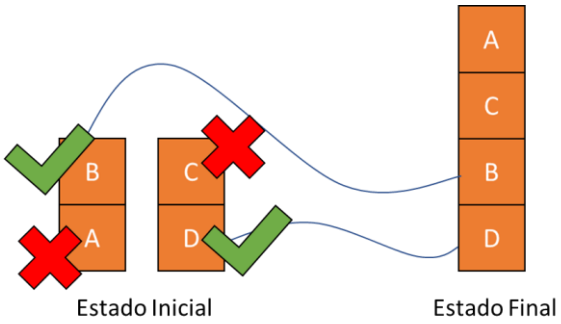
Diagrama De Flujo Del A*



Heurística

$h(n)$: Número de bloques colocados incorrectamente respecto al estado meta.

Se considera que un bloque no está bien colocado si no existe una pila en el estado meta que tenga a dicho bloque en el mismo nivel.



Algoritmo Desinformado: DFS

Depth First Search es un algoritmo gráfico transversal muy popular. El primer paso del Depth First Search es observar el nodo raíz de un grafo. Existen ciertas ocasiones en las que el nodo raíz no llega a los demás nodos, por lo que es necesario utilizar otro nodo. Si realizamos un recorrido por todo el gráfico, éste visita el primer hijo de un nodo de la raíz, luego, a su vez, mira el primer hijo de este nodo y continúa a lo largo de esta rama hasta llegar a un nodo de la hoja.

Algoritmo:

Se plantea el algoritmo siguiendo un esquema recursivo:

1. Marcar todos los nodos como no visitados.

2. Elegir el nodo raíz (u) como nodo de partida.

3. Marcar u como visitado.

4. Mientras haya nodos:

4.1 Sacar nodo cabeza.

4.2 Para todo nodo adyacente (v) a u

4.3 Si v no ha sido visitado:

4.3.1 Si es nodo meta, entonces:

4.3.1.1 Regresar.

Sino:

4.3.1.2 Poner el nodo hijo en la cola.

4.4 Terminar.

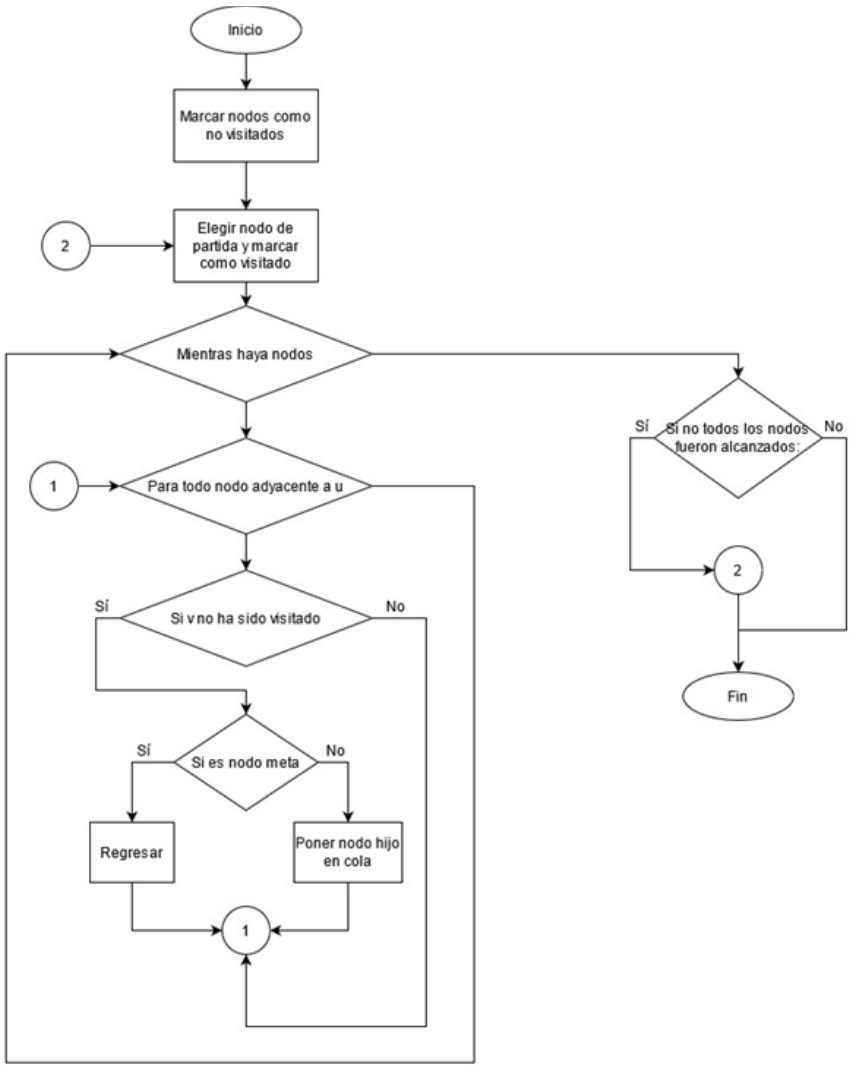
5. Si no fueron alcanzados todos los nodos, entonces:

5.1 Volver a 2 y elegir un nuevo nodo de partida.

6. Finalizar.

El algoritmo DFS posee varias aplicaciones la más importante es para problemas de conectividad, si un grafo es conexo, detectar ciclos en un grafo, numero de componentes conexas, etc. Es bastante útil en otros algoritmos como para hallar las componentes fuertemente conexas en un grafo

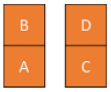
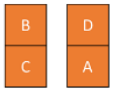
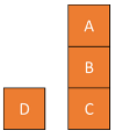
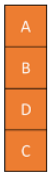
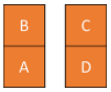
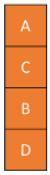
Diagrama de Flujo

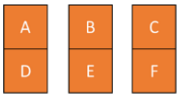
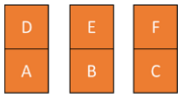
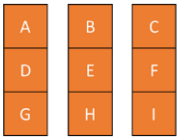
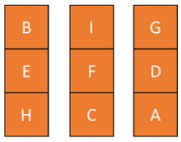


Instancias Empleadas

Para las instancias realizaremos algunos experimentos variando los siguientes parámetros:

- Número de bloques.
- Configuración Inicial y Final.
- Cantidad máxima de pilas.

Instancia	Estado Inicial	Estado Final	Cantidad de Bloques	Límite de Pilas
1			4	∞
2				2
3			4	∞
4				3
5			4	∞
6				3

Instancia	Estado Inicial	Estado Final	Cantidad de Bloques	Límite de Pilas
7			5	∞
8				3
9			6	∞
10				5
11			9	∞
12				5

Experimentación

En esta experimentación nos gustaría principalmente observar qué tanta diferencia existe entre un algoritmo informado y uno desinformado, principalmente en cuanto al costo de las soluciones encontradas y al tiempo que los algoritmos tardan en encontrar dichas soluciones, pero también nos interesan algunos aspectos como el número de nodos que genera cada algoritmo y cómo varían las soluciones cuando limitamos el número de pilas que se pueden formar.

Para llevar a cabo este experimento se ejecutó el algoritmo A* 100 veces en cada instancia y el algoritmo DFS 100 veces en cada instancia (exceptuando las últimas dos, en las que se consideraron 35 iteraciones para cada instancia). En cada repetición se tomó nota de los resultado obtenidos para poder analizarlos.

Resultados

Resumen de Resultados

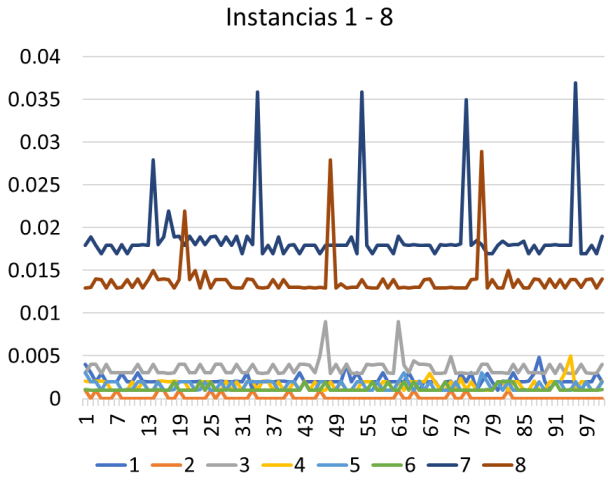
Algoritmo A*						
Instance	Max Open List Size	Open List Operations	Total Nodes	Cost	Repeticiones	Average Time
1	24	43	66	3	100	0.0120
2	2	10	8	0	100	0.0200
3	30	71	126	5	100	0.0332
4	19	50	64	5	100	0.0412
5	17	37	54	4	100	0.0510
6	15	35	42	4	100	0.0605
7	121	223	409	8	100	0.0879
8	96	227	307	9	100	0.0929
9	4842	6645	15540	6	100	7.0195
10	1890	2637	5922	6	100	1.5837
11	3665	4538	7841	6	100	4.4664
12	3301	4146	6841	6	100	3.6295

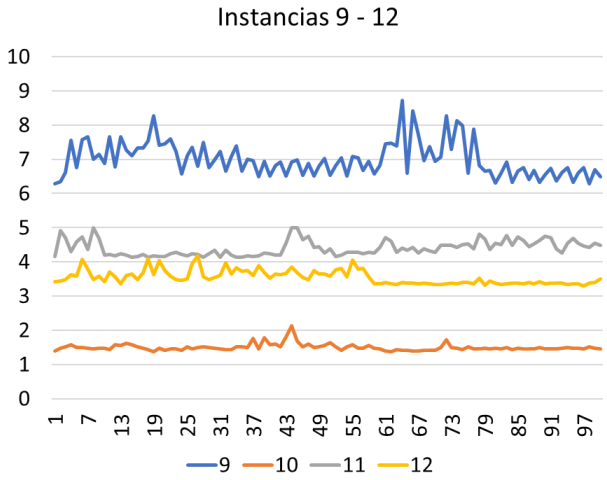
Algoritmo DFS						
Instance	Max Open List Size	Open List Operations	Total Nodes	Cost	Repeticiones	Average Time
1	36	79	111	11	100	0.0130
2	2	10	8	0	100	0.0199
3	48	137	208	20	100	0.0362
4	37	124	162	19	100	0.0447
5	55	208	357	11	100	0.0635
6	44	195	299	10	100	0.0707
7	375	732	1074	112	100	0.1417
8	214	543	676	98	100	0.1209
9	5792	11260	17051	1807	100	17.7923
10	5783	11251	17006	1807	100	18.2509
11	13150	21728	28193	0	35	58.6534
12	12864	21264	27596	0	35	58.7264

Comparativas de los Algoritmos

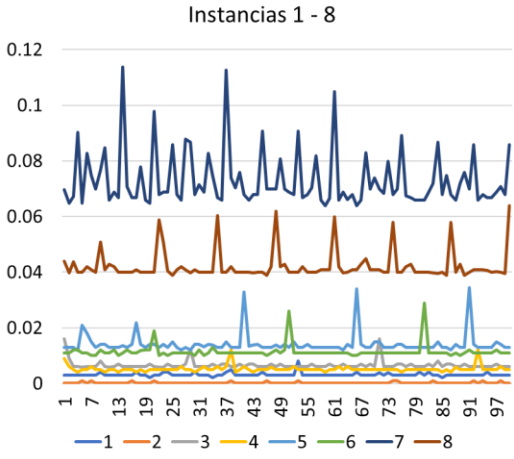
Para facilitar la visualización de los datos, se separaron en dos gráficas las instancias cuando la diferencia de valores entre instancias no permite apreciar bien los datos obtenidos.

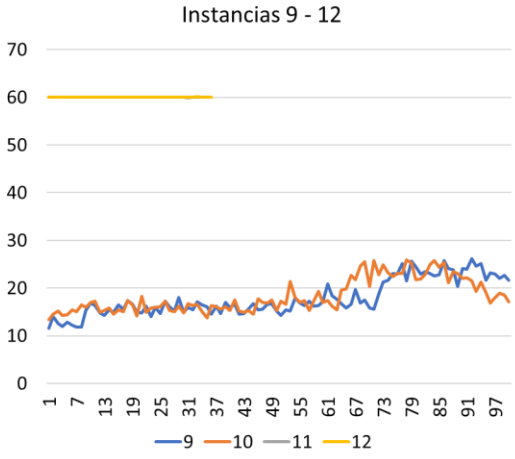
Tiempos medidos con el Algoritmo A*



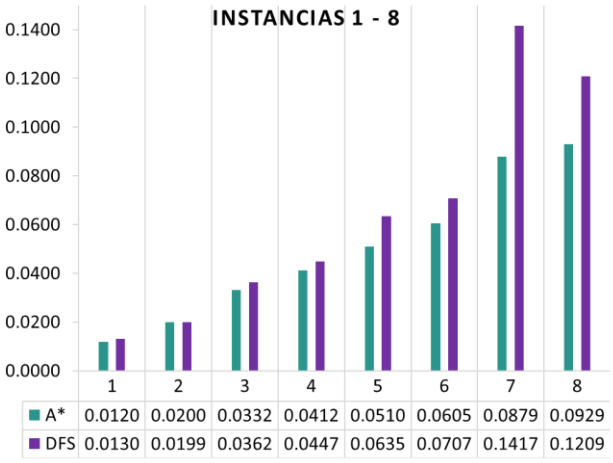


Tiempos medidos con el Algoritmo DFS

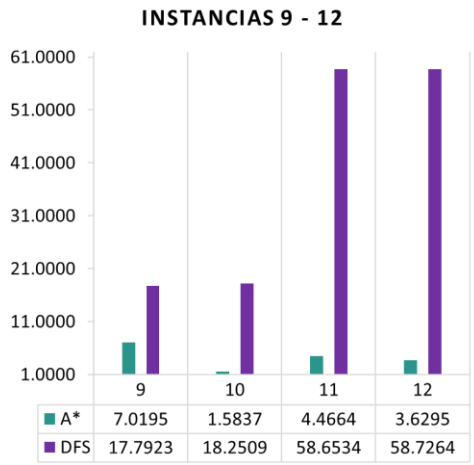




Tiempos Promedio

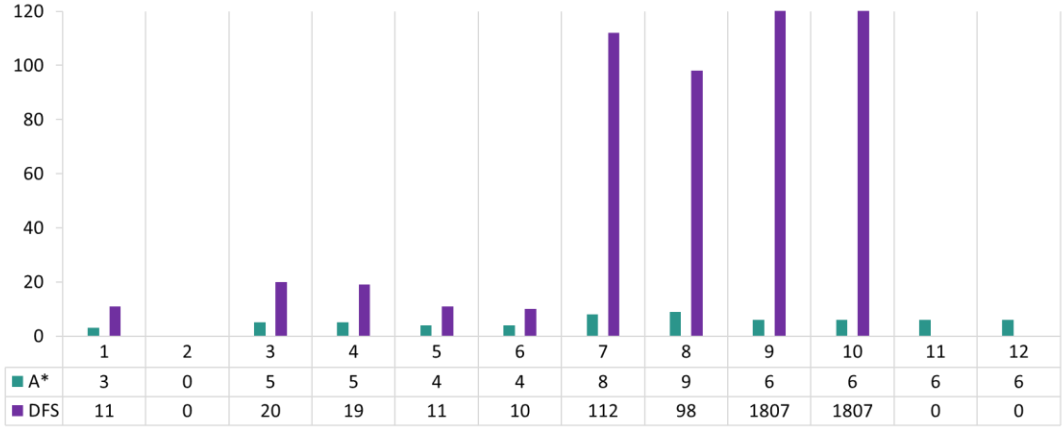


	1	2	3	4	5	6	7	8
A*	0.0120	0.0200	0.0332	0.0412	0.0510	0.0605	0.0879	0.0929
DFS	0.0130	0.0199	0.0362	0.0447	0.0635	0.0707	0.1417	0.1209



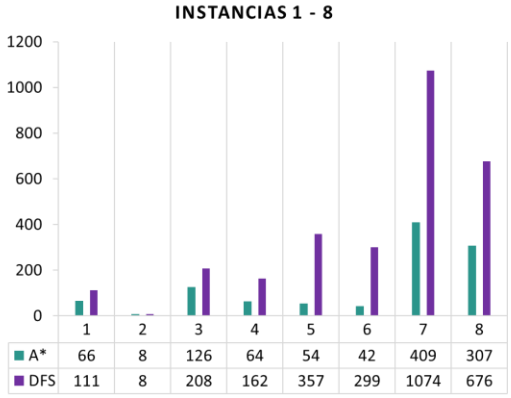
	9	10	11	12
A*	7.0195	1.5837	4.4664	3.6295
DFS	17.7923	18.2509	58.6534	58.7264

Costos

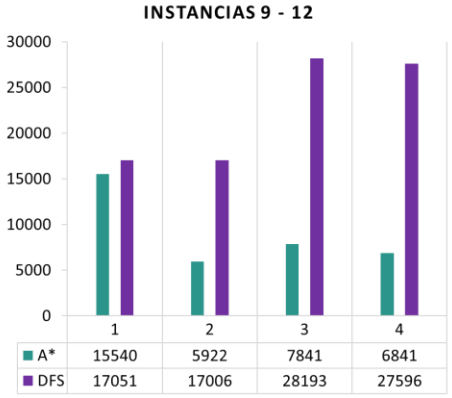


	1	2	3	4	5	6	7	8	9	10	11	12
A*	3	0	5	5	4	4	8	9	6	6	6	6
DFS	11	0	20	19	11	10	112	98	1807	1807	0	0

Nodos Generados en Total

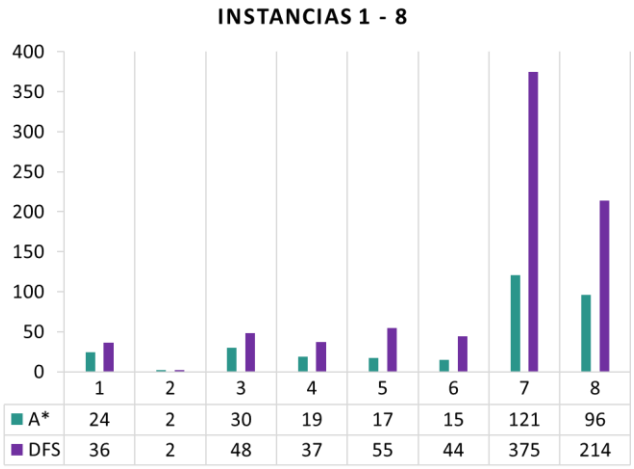


	1	2	3	4	5	6	7	8
A*	66	8	126	64	54	42	409	307
DFS	111	8	208	162	357	299	1074	676

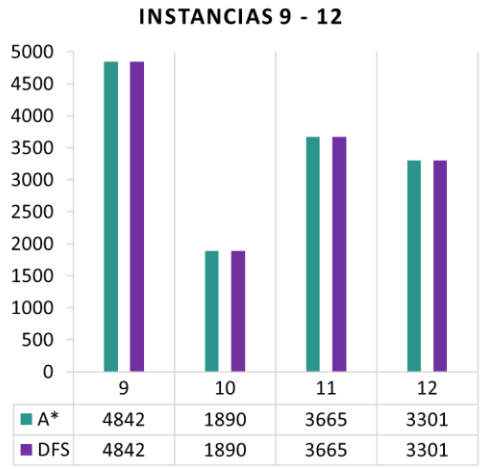


	1	2	3	4
A*	15540	5922	7841	6841
DFS	17051	17006	28193	27596

Tamaño Máximo de Lista Abierta

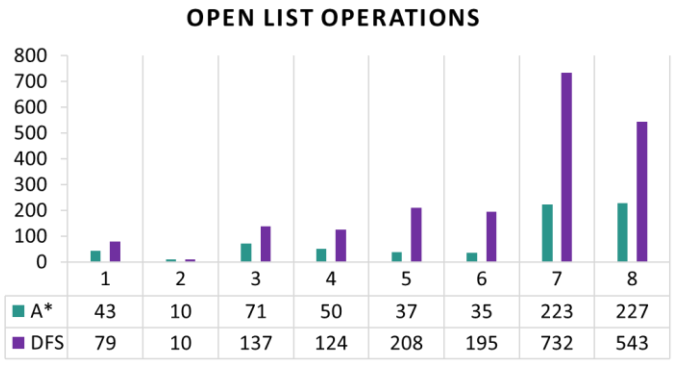


	1	2	3	4	5	6	7	8
A*	24	2	30	19	17	15	121	96
DFS	36	2	48	37	55	44	375	214

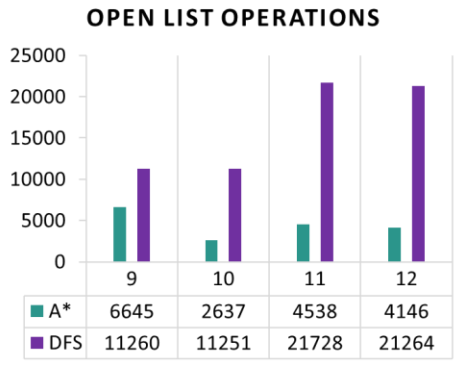


	9	10	11	12
A*	4842	1890	3665	3301
DFS	4842	1890	3665	3301

Operaciones en la Lista Abierta



	1	2	3	4	5	6	7	8
A*	43	10	71	50	37	35	223	227
DFS	79	10	137	124	208	195	732	543



	9	10	11	12
A*	6645	2637	4538	4146
DFS	11260	11251	21728	21264