
Sistemas de Inteligencia Artificial
MÉTODOS DE BÚSQUEDA
Trabajo Práctico 1
ITBA
Canva presentation

Índice

1 8 PUZZLE

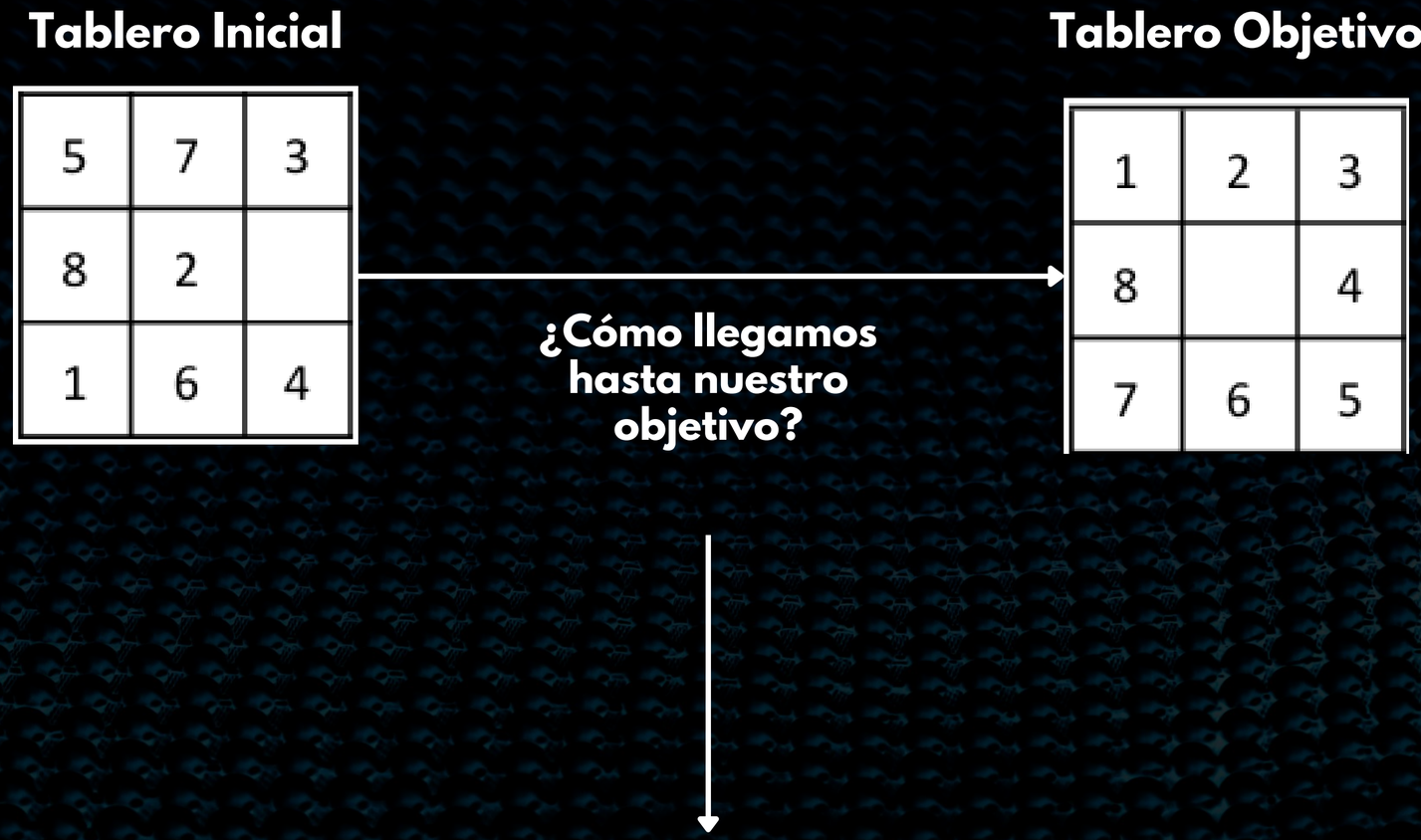
5	7	3
8	2	
1	6	4

- Estructura de Estados
- Heurísticas admisibles
- Métodos de búsqueda

2 SOKOBAN

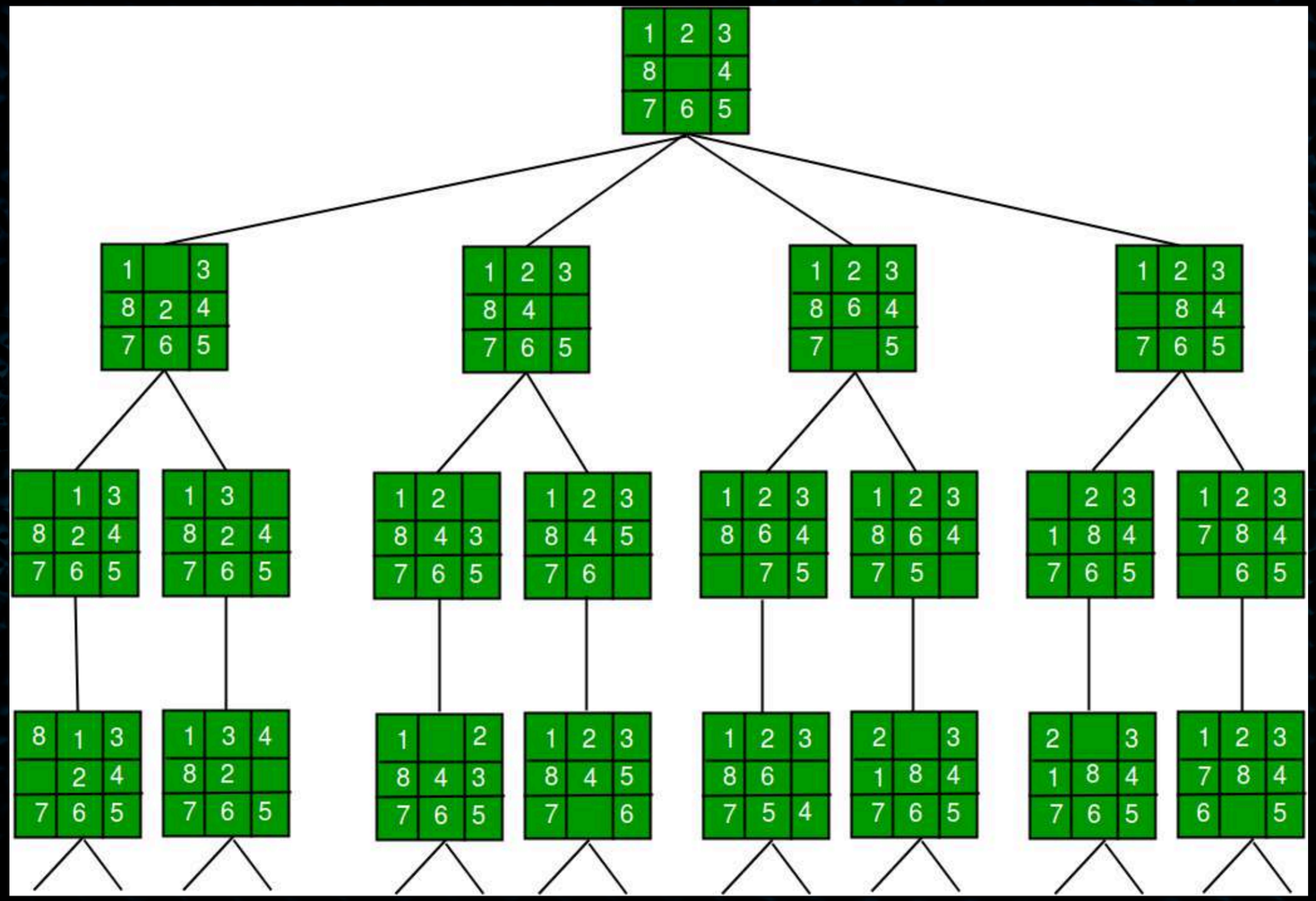
- Estructura de Estados
- Métodos de Búsqueda
- Heurísticas
- Pruebas y Resultados


8 PUZZLE

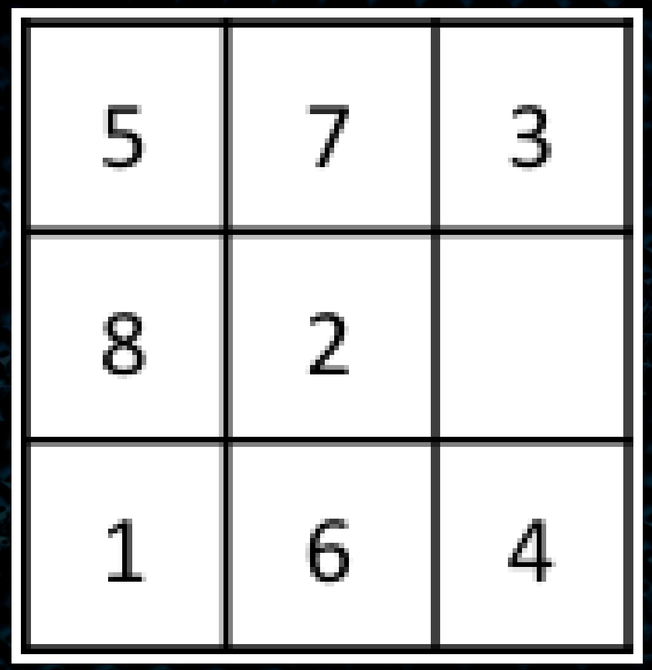


A través de las acciones

Sin embargo, existen 181440 estados posibles!!



ESTRUCTURA DE ESTADOS



5	7	3
8	2	
1	6	4

=

- Matriz de 3 filas por 3 columnas
- Cada posición puede almacenar un valor entero entre 0 y 8
- El 0 representa el espacio vacío

Ventajas

- Estructura de datos simple
- Fácil/Rápida aplicación de acciones (swap entre dos posiciones de la matriz)
- Comparación de estados sencilla

Desventajas

- Detección de estados equivalentes (tableros rotados)

HEURÍSTICAS ADMISIBLES

Distancia de Manhattan

Distancia de Euclideana

MEJORES MÉTODOS DE BÚSQUEDA

Global Greedy Search
A* Search

Si bien depende de qué tan lejos está la solución del tablero inicial, debido a la gran cantidad de estados posibles y la cantidad de ramificaciones, los algoritmos no informados podrían llegar a tardar mucho en encontrar la única solución posible. También se estima que el mejor será el Greedy si querés una solución rápida y A* si querés una solución óptima.


SOKOBAN

Estructura de Estado

Matriz con datos fijos

Una matriz donde cada celda representa paredes, espcios vacíos o casillas objetivo.

Lista de cajas

Las cajas al ser objetos móviles se consideraron en una lista por separado para facilitar la modificación, copia y comparación de los estados.

Posición jugador

La posición del jugador también se guarda individualmente por los mismos motivos.

Heurísticas

Distancia Manhatan

La suma de las distancias Manhatan del jugador a cada caja + la distancia de cada goal a su caja más cercana. Descartando además las cajas que ya están sobre un goal, para mantener la admisibilidad.

Distancia Euclideana

Análogo al anterior pero con distancias euclideanas.

Distancia Manhatan + Deadlocks

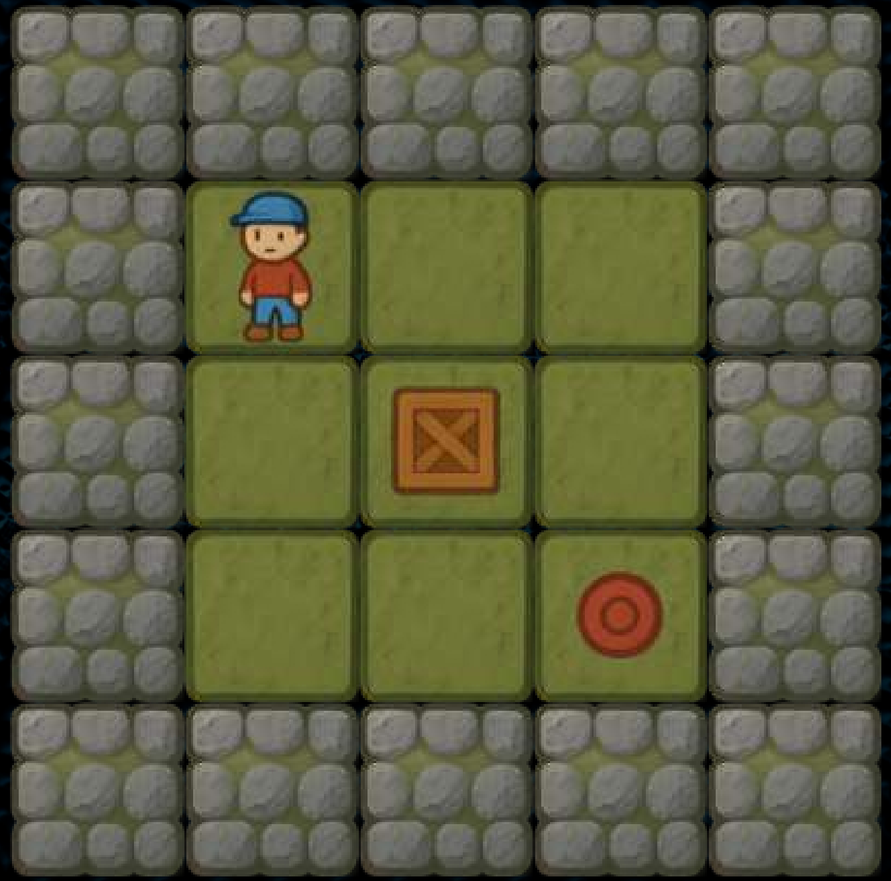
Se combina la heurística de distancia manhatan con una heurística para los estados deadlock, devolviendo un valor infinito cuando es uno de estos. Se prueba con dos casos, uno para solo deadlocks en esquinas y otro con deadlocks de esquinas + paredes

Heurísticas (cont.)

- h_{euc}
- h_{man}
- $h_{no-corners}$
- $h_{no-deadlocks}$
- $h1 = max\left(h_{man}, h_{no-corners}\right)$
- $h2 = max\left(h_{man}, h_{no-deadlocks}\right)$

En verde las heurísticas utilizables, y en rojo las auxiliares

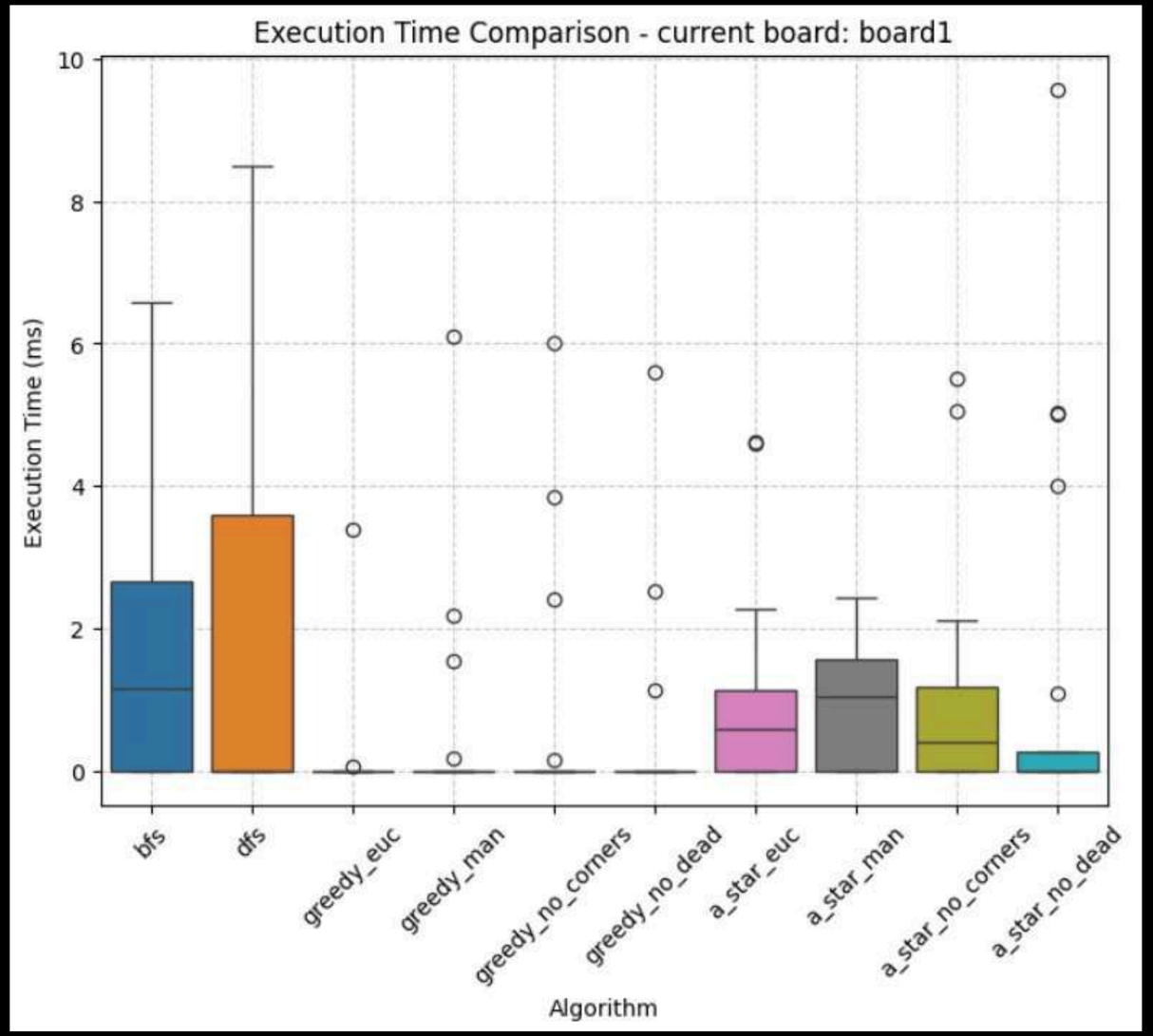
Tablero 1



Resultados:

ALGORITMO	COSTO DE SOLUCIÓN	TIEMPO DE EJECUCIÓN MEDIO
BFS	5	1.67 ms
DFS	13	1.79 ms
G-EUCLIDEAN	5	0.17 ms
G-MANHATTAN	5	0.50 ms
G-NO-CORNERS	5	0.62 ms
G-NO-DEADLOCKS	5	0.46 ms
A*-EUCLIDEAN	5	0.96 ms
A*-MANHATTAN	5	0.99 ms
A*-NO-CORNERS	5	1.01 ms
A*-NO-DEADLOCKS	5	1.23 ms

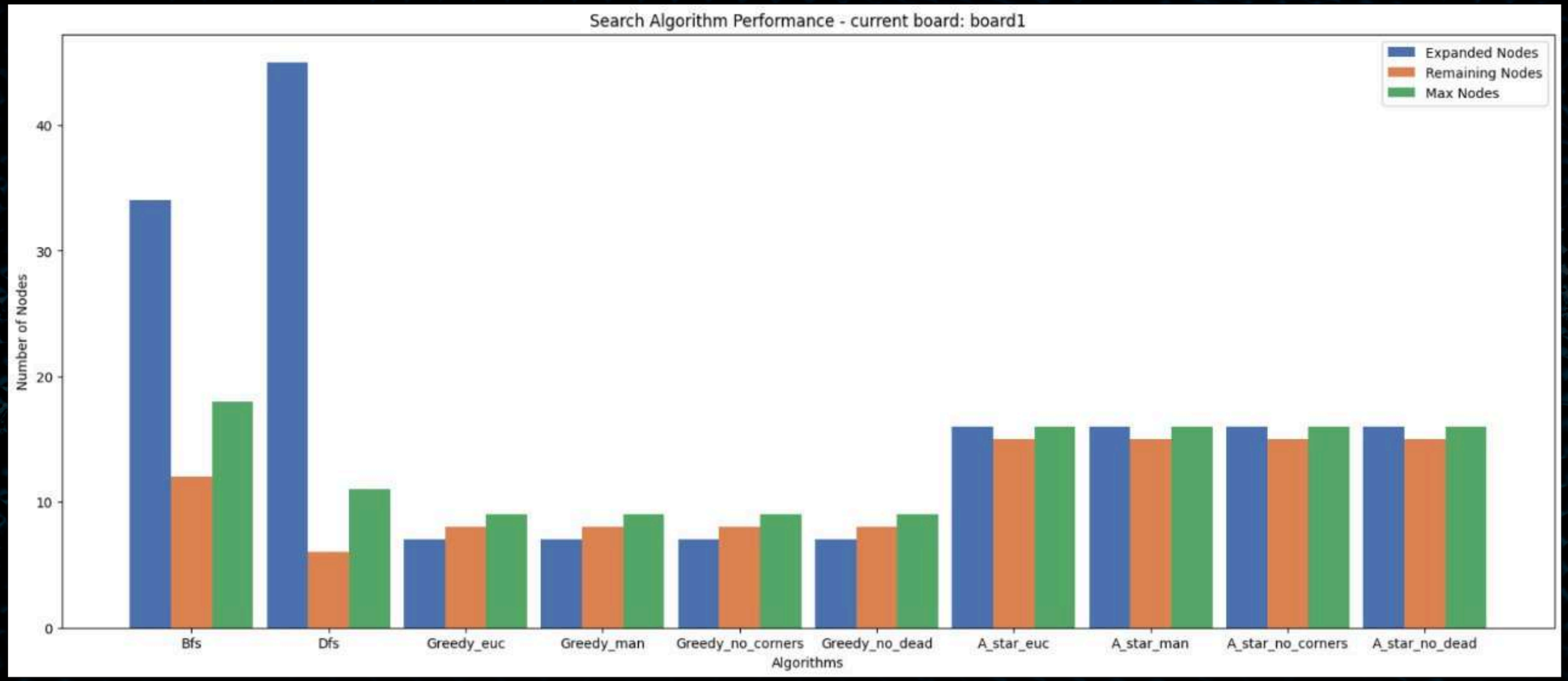
Desvíos de datos



Algorithm Performance

ALGORITMO	NODOS EXPANDIDOS	NODOS RESTANTES EN FRONT.	NODOS MÁXIMO EN FRONT.
BFS	34	12	18
DFS	45	6	11
G-EUCLIDEAN	7	8	9
G-MANHATTAN	7	8	9
G-NO-CORNERS	7	8	9
G-NO-DEADLOCKS	7	8	9
A*-EUCLIDEAN	16	15	16
A*-MANHATTAN	16	15	16
A*-NO-CORNERS	16	15	16
A*-NO-DEADLOCKS	16	15	16

Algorithm Performance

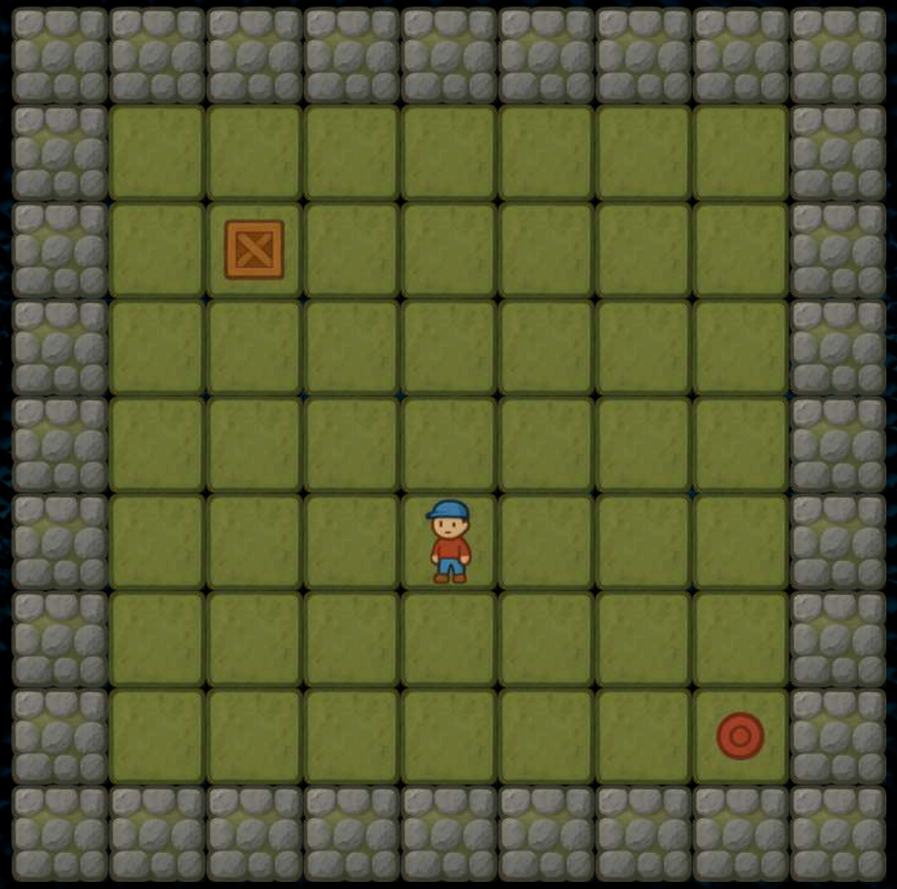


Orden de acciones

- Default: ARRIBA, ABAJO, IZQUIEDA, DERECHA
- Inverso: DERECHA, IZQUIERDA, ABAJO, ARRIBA

Los resultados estarán en el orden default, salvo que se diga lo contrario

Tablero 2



Resultados:

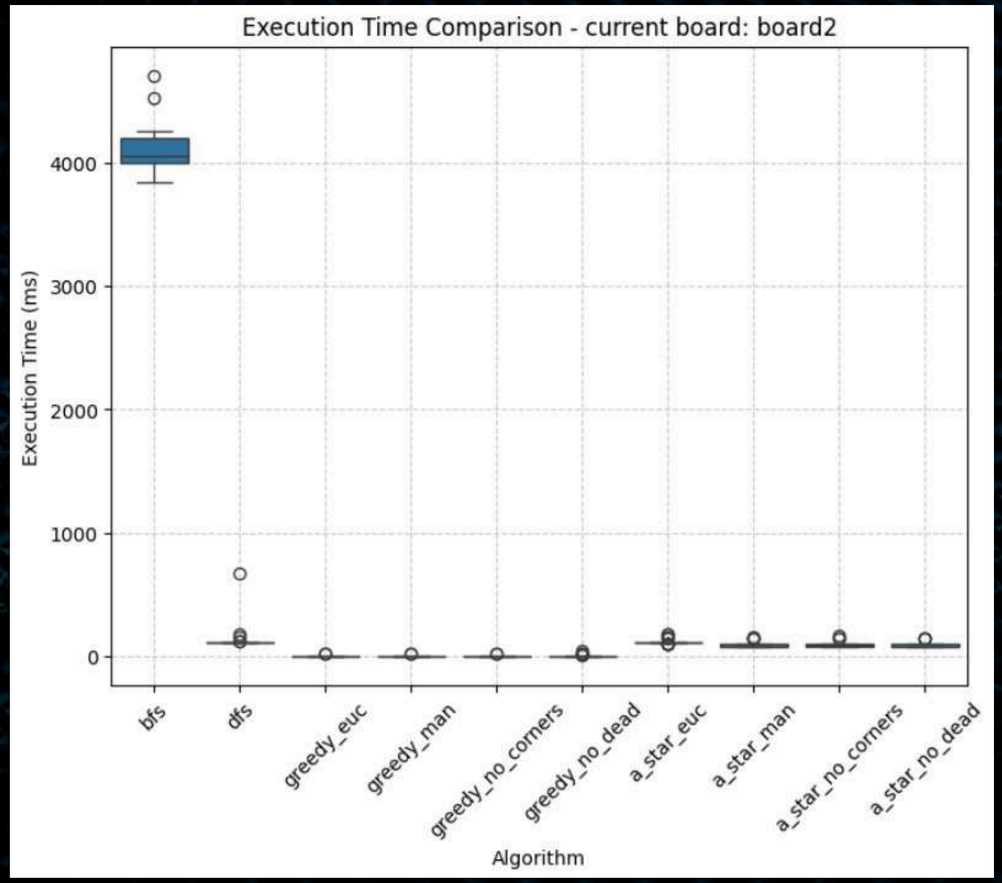
ALGORITMO	COSTO DE SOLUCIÓN	TIEMPO DE EJECUCIÓN MEDIO
BFS	18	4123.02 ms
DFS	156	145.16 ms
G-EUCLIDEAN	18	1.59 ms
G-MANHATTAN	18	1.56 ms
G-NO-CORNERS	18	1.56 ms
G-NO-DEADLOCKS	18	4.70 ms
A*-EUCLIDEAN	18	116.14 ms
A*-MANHATTAN	18	94.55 ms
A*-NO-CORNERS	18	96.58 ms
A*-NO-DEADLOCKS	18	92.73 ms

Resultados en orden de acciones inverso:

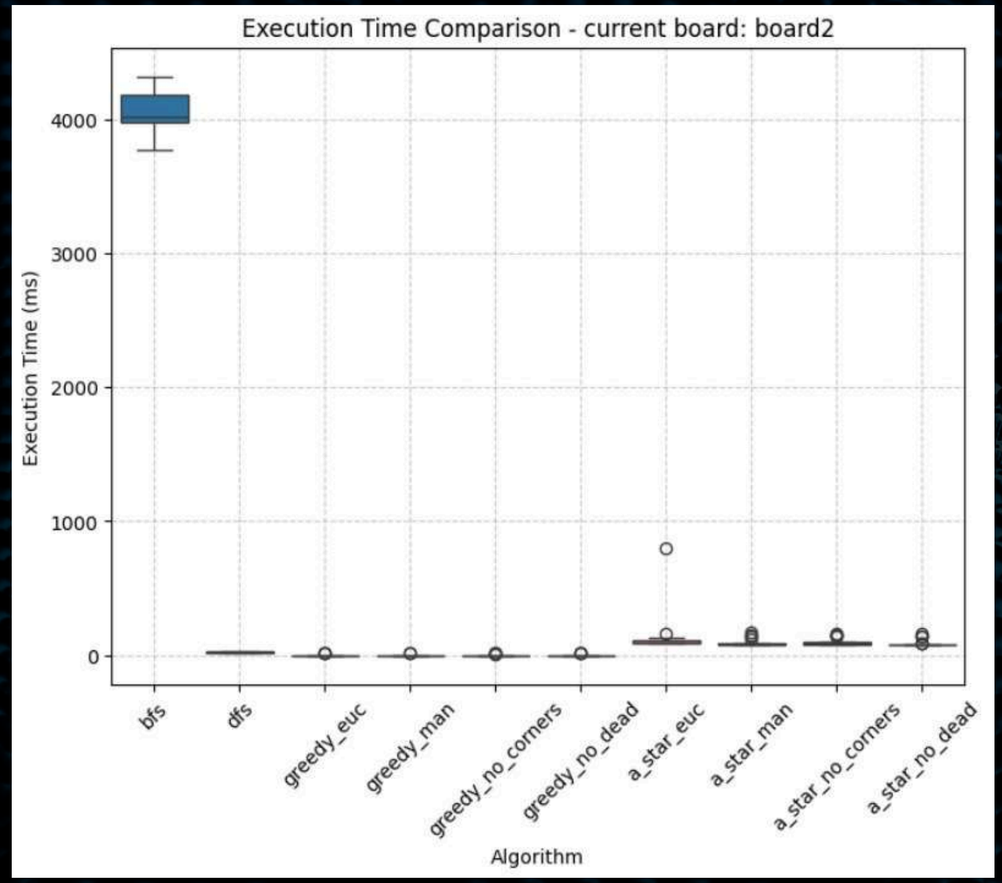
ALGORITMO	COSTO DE SOLUCIÓN	TIEMPO DE EJECUCIÓN MEDIO
BFS	18	4043.04 ms
DFS	48	26.66 ms
G-EUCLIDEAN	18	2.35 ms
G-MANHATTAN	18	1.56 ms
G-NO-CORNERS	18	2.51 ms
G-NO-DEADLOCKS	18	2.21 ms
A*-EUCLIDEAN	18	140.78 ms
A*-MANHATTAN	18	95.62 ms
A*-NO-CORNERS	18	99.60 ms
A*-NO-DEADLOCKS	18	90.35 ms

Desvíos de datos

Orden natural



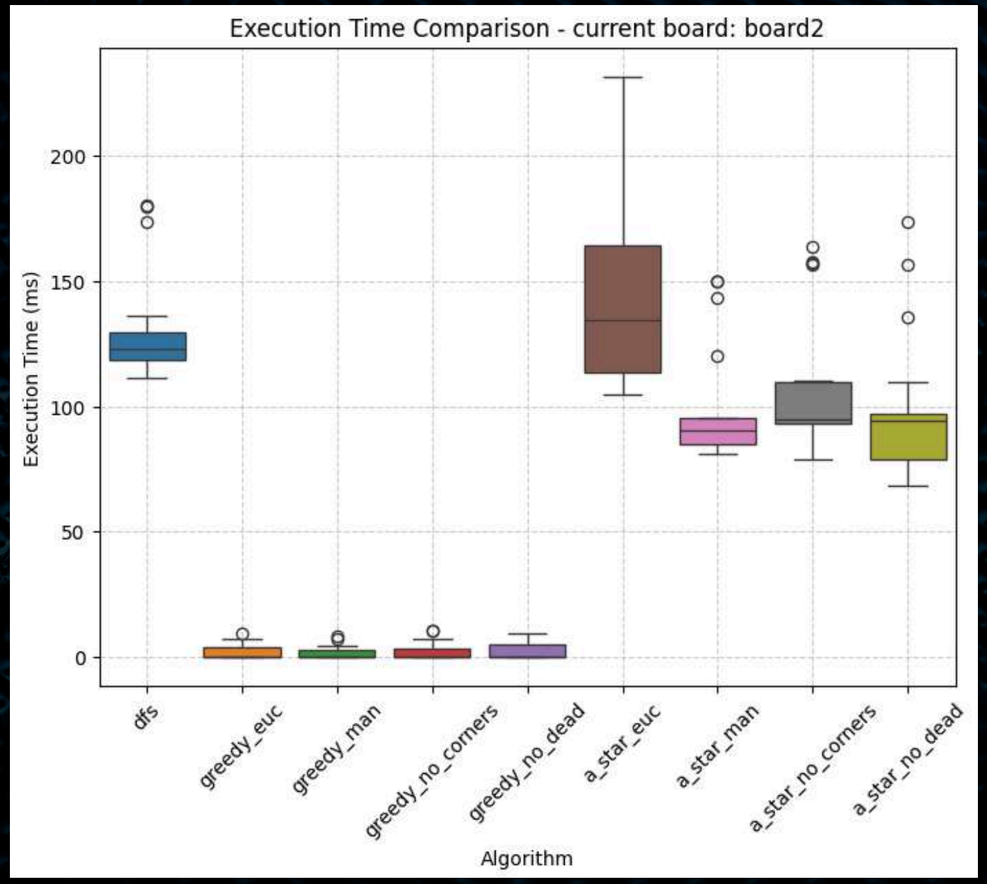
Orden inverso



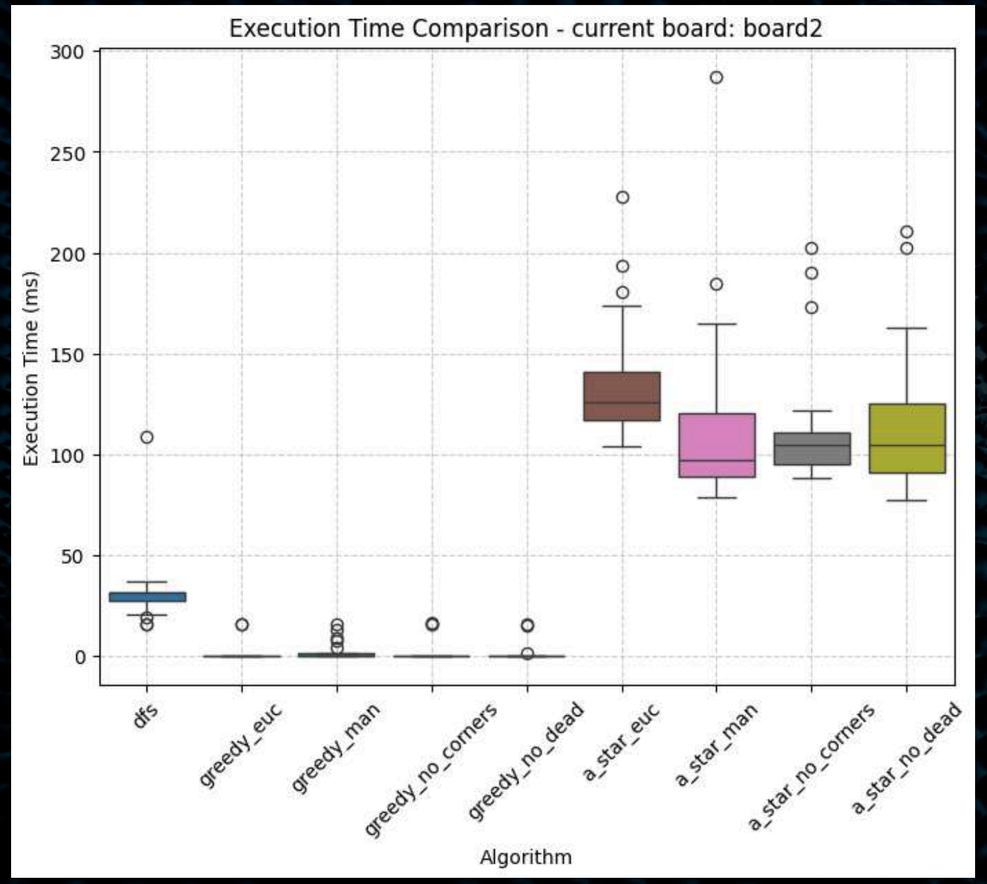
Nota: como BFS siempre generará tiempos grandes, decidimos no graficarlos para compararlos pues ya se asumirá que duran mucho

Desvíos de datos (sin BFS)

Orden natural



Orden inverso



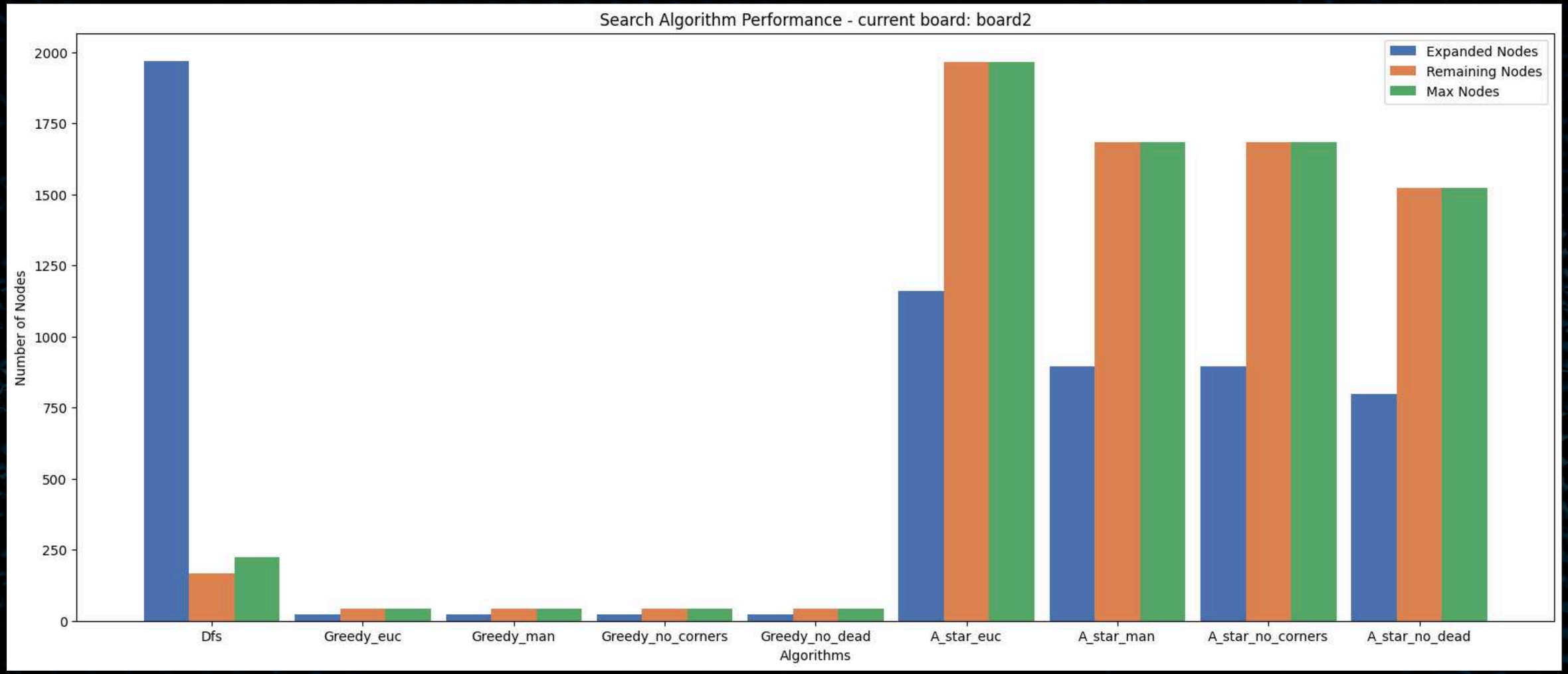
Algorithm Performance

ALGORITMO	NODOS EXPANDIDOS	NODOS RESTANTES EN FRONT.	NODOS MÁXIMO EN FRONT.
BFS	74818	26055	26206
DFS	1969	167	222
G-EUCLIDEAN	21	41	42
G-MANHATTAN	21	41	42
G-NO-CORNERS	21	41	42
G-NO-DEADLOCKS	21	41	42
A*-EUCLIDEAN	1159	1966	1967
A*-MANHATTAN	896	1683	1684
A*-NO-CORNERS	896	1683	1684
A*-NO-DEADLOCKS	796	1523	1524

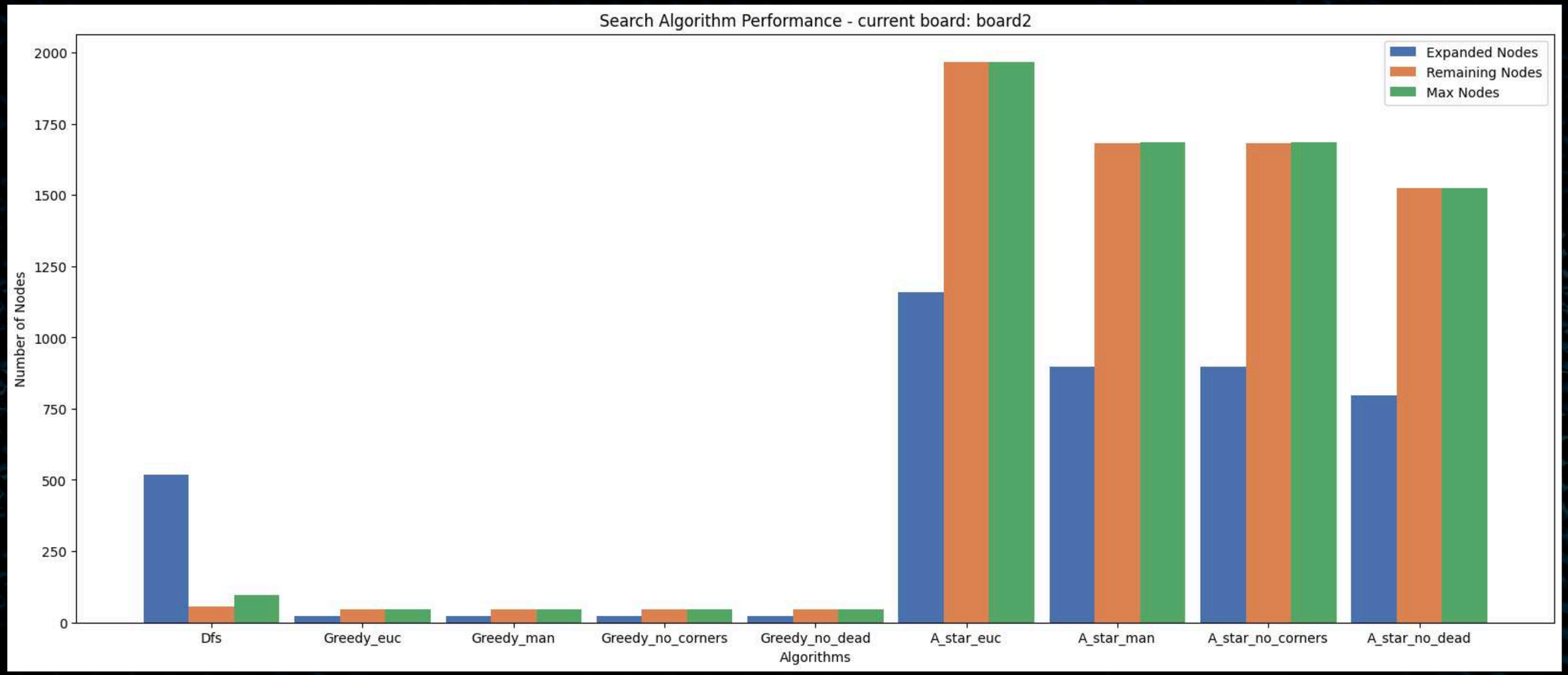
Algorithm Performance con orden de acciones inverso

ALGORITMO	NODOS EXPANDIDOS	NODOS RESTANTES EN FRONT.	NODOS MÁXIMO EN FRONT.
BFS	74812	26086	26113
DFS	519	57	97
G-EUCLIDEAN	23	45	46
G-MANHATTAN	23	45	46
G-NO-CORNERS	23	45	46
G-NO-DEADLOCKS	23	45	46
A*-EUCLIDEAN	1159	1966	1967
A*-MANHATTAN	896	1683	1684
A*-NO-CORNERS	896	1683	1684
A*-NO-DEADLOCKS	796	1523	1524

Algorithm Performance



Algorithm Performance (en orden inverso)



Comparación de orden de acciones

Se puede observar que el más afectado por el orden de acciones es el algoritmo DFS, esto tiene sentido ya que este algoritmo prioriza el orden de acciones dado para expandir. El resto de algoritmos tienen también sus fluctuaciones pero no son tan significativas.

Veamos ejemplos aún más notorios del impacto del orden de acciones sobre el algoritmo DFS...

Análisis DFS con orden en ULRD vs DRLU

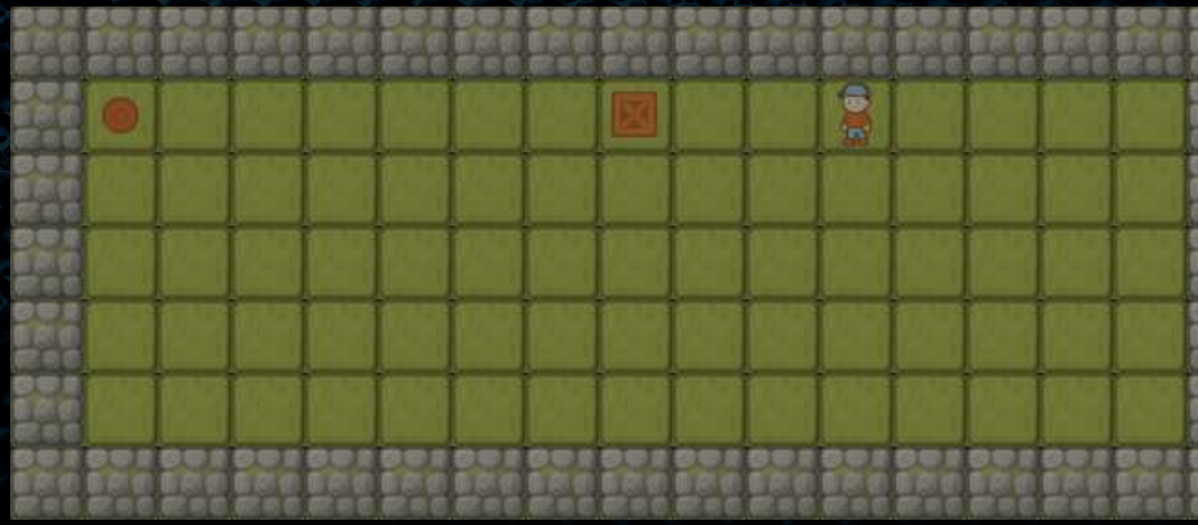
Podemos conjeturar que en este tablero va a ser más beneficioso priorizar los movimiento hacia la izquierda arriba ya que esa es la dirección en la que hay que mover las cajas. Por lo tanto deberíamos notar que el orden ULRD (Up-Left-Right-Down) es más rápido que el DRLU (Down-Right-Left-Up).

Resultados

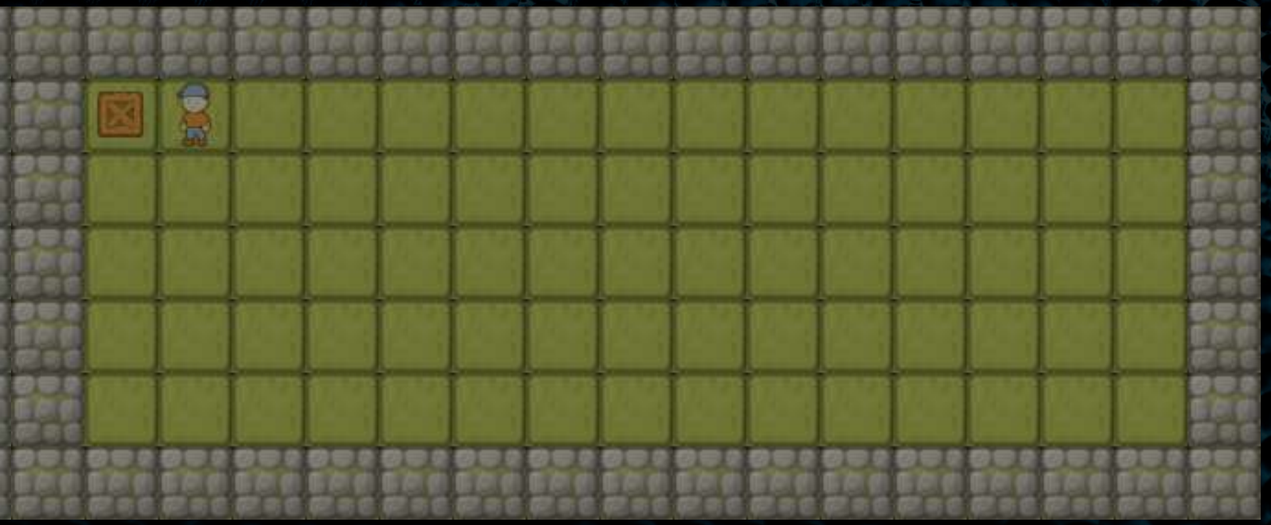
ORDEN	COSTO DE SOLUCIÓN	TIEMPO DE EJECUCIÓN	NODOS EXPANDIDOS	NODOS RESTANTES EN FRONT.	NODOS MÁXIMO EN FRONT.
ULRD	134	11.30 ms	133	141	142
DRLU	1938	7639.19 ms	32815	1914	5002
Para comparar tomemos el algoritmo A* con heurística Manhattan					
ULRD	32	405.47 ms	5894	6210	6211
DRLU	32	405.29 ms	5925	6249	6250

Claramente dependiendo el mapa, elegir un orden de acciones beneficioso tendrá un alto impacto en la rapidez y eficiencia de DFS, mientras que los otros algoritmos no se ven muy afectados.

DFS - ULRD



DFS - DRLU



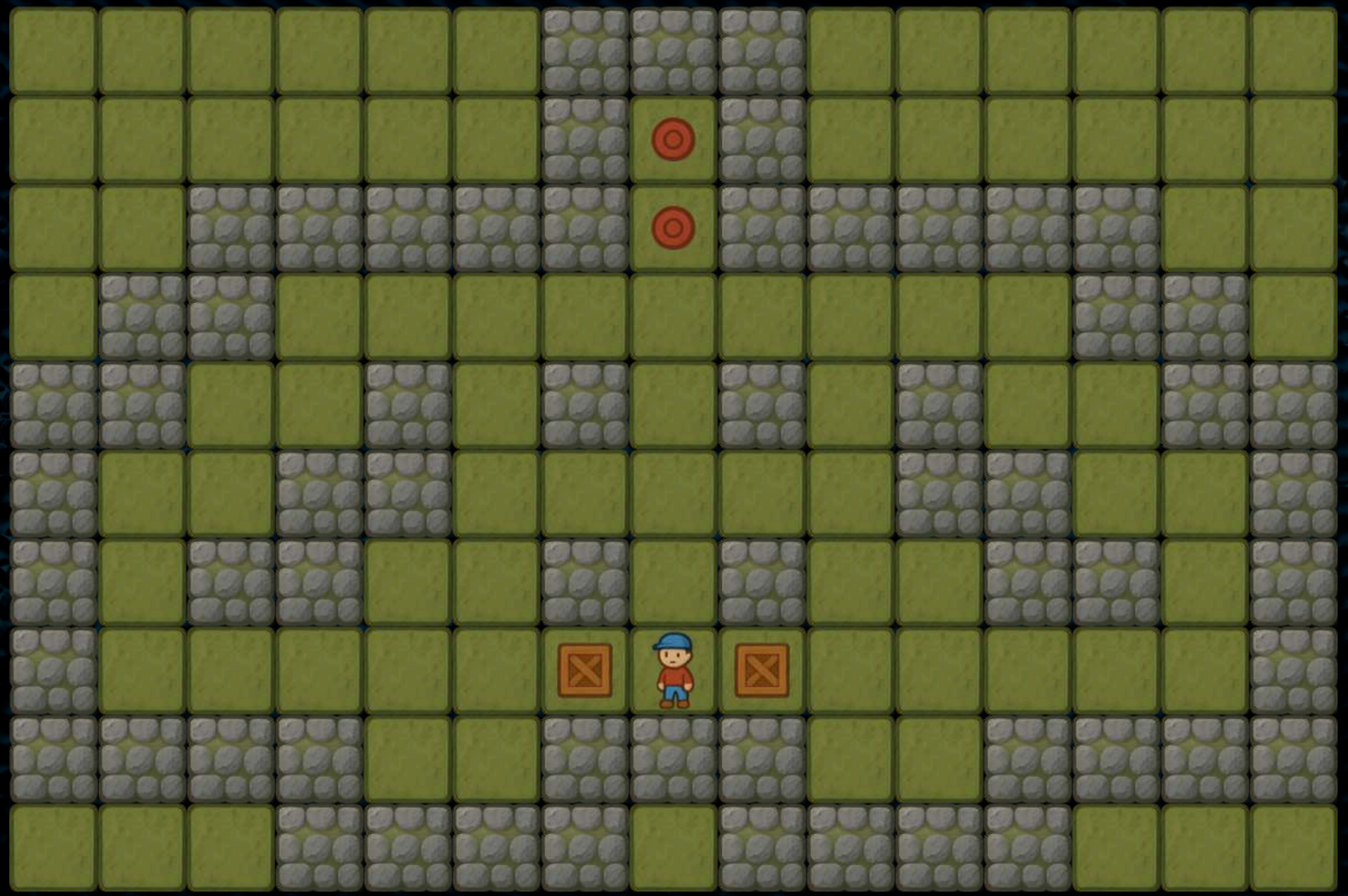
Se usa un ejemplo más simple porque el anterior de 1900 pasos sería un poco largo...

Mapas con mayor cantidad de deadlocks

En los mapas anteriores, debido a su simplicidad, no se pudo apreciar una diferencia significativa entre los algoritmos con heurísticas que detectan deadlocks vs las que no los detectan.

¿Qué pasa en tableros más complejos?

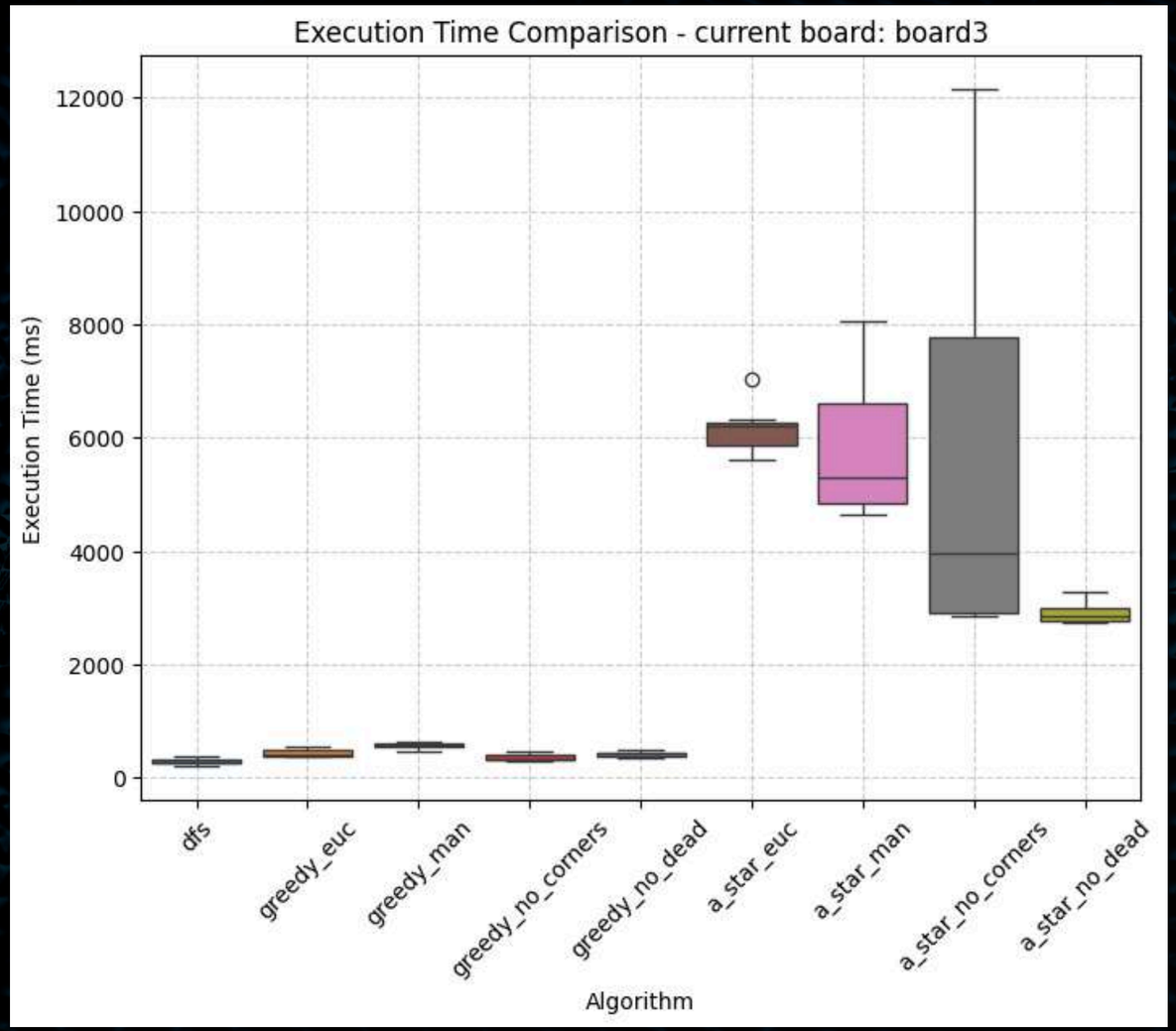
Tablero 3



Resultados:

ALGORITMO	COSTO DE SOLUCIÓN	TIEMPO DE EJECUCIÓN MEDIO
BFS	78	12784.65 ms
DFS	320	250.18 ms
G-EUCLIDEAN	138	368.95 ms
G-MANHATTAN	90	488.83 ms
G-NO-CORNERS	90	321.50 ms
G-NO-DEADLOCKS	90	378.84 ms
A*-EUCLIDEAN	78	5817.11 ms
A*-MANHATTAN	78	4607.33 ms
A*-NO-CORNERS	78	2417.81 ms
A*-NO-DEADLOCKS	78	2712.84 ms

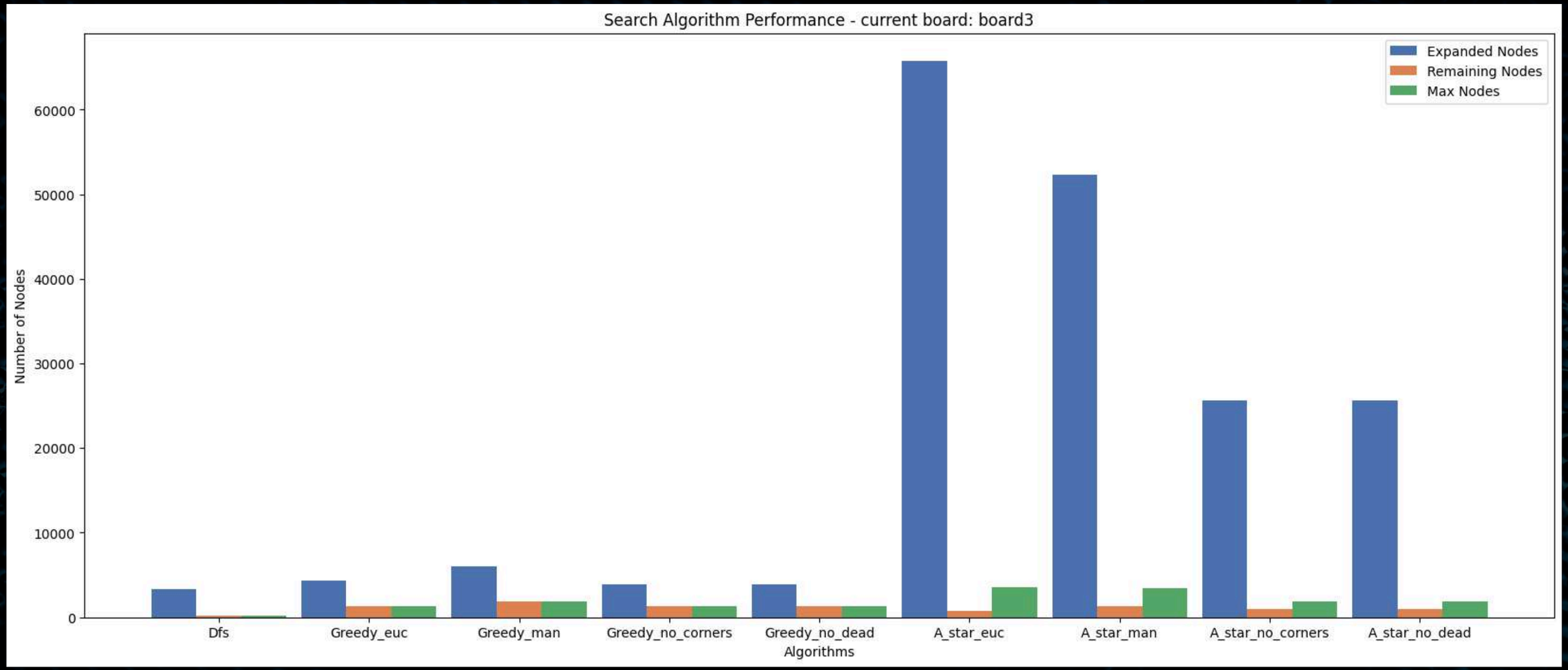
Desvíos de datos



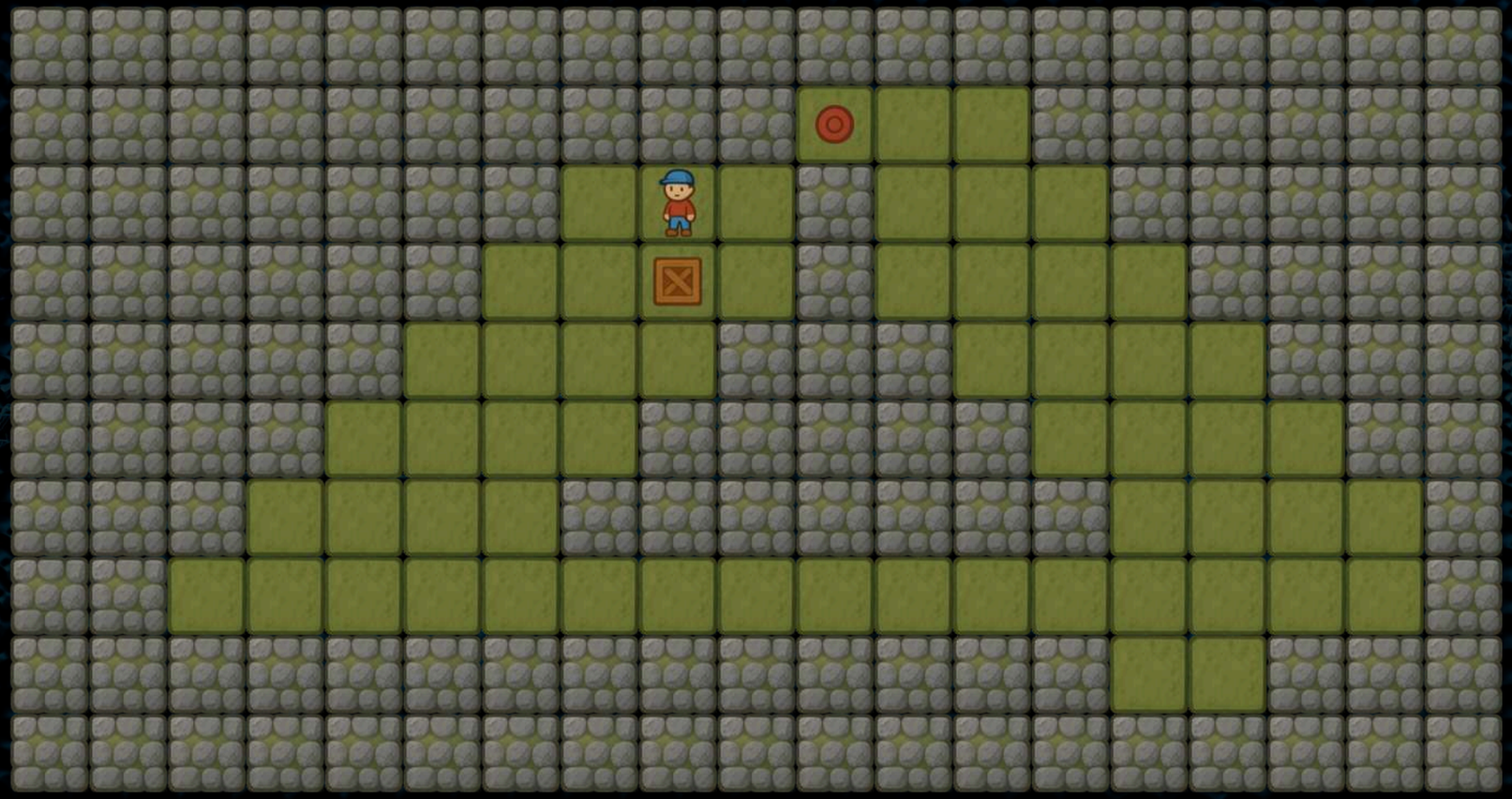
Algorithm Performance

ALGORITMO	NODOS EXPANDIDOS	NODOS RESTANTES EN FRONT.	NODOS MÁXIMO EN FRONT.
BFS	213379	2421	5791
DFS	3301	159	174
G-EUCLIDEAN	4329	1351	1352
G-MANHATTAN	5959	1903	1904
G-NO-CORNERS	3878	1248	1249
G-NO-DEADLOCKS	3878	1248	1249
A*-EUCLIDEAN	65783	773	3574
A*-MANHATTAN	52310	1244	3479
A*-NO-CORNERS	25653	908	1819
A*-NO-DEADLOCKS	25653	908	1819

Algorithm Performance



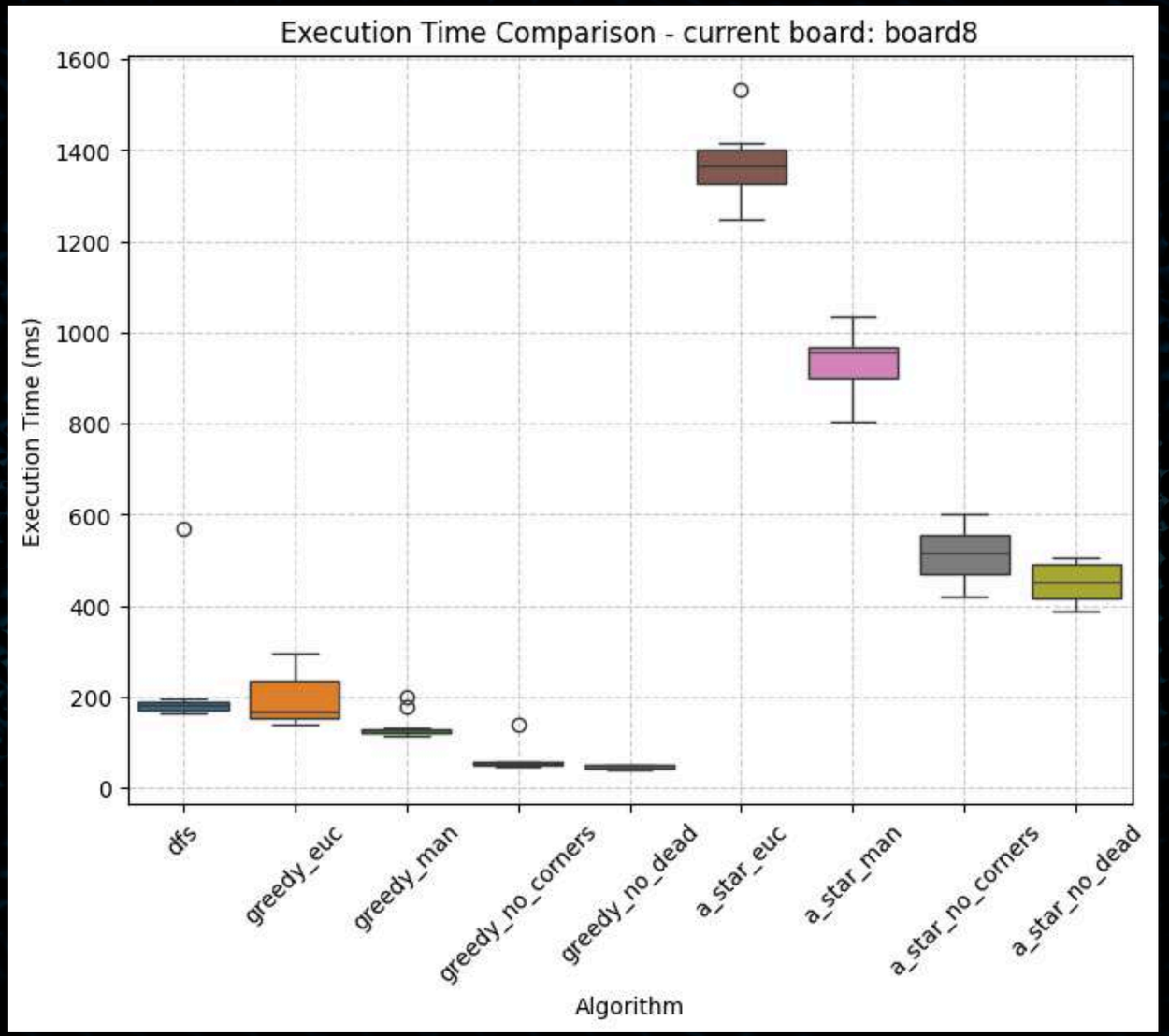
Tablero 8



Resultados:

ALGORITMO	COSTO DE SOLUCIÓN	TIEMPO DE EJECUCIÓN MEDIO
BFS	56	1054.92 ms
DFS	118	185.77 ms
G-EUCLIDEAN	56	167.19 ms
G-MANHATTAN	56	123.95 ms
G-NO-CORNERS	56	59.64 ms
G-NO-DEADLOCKS	56	49.48 ms
A*-EUCLIDEAN	56	1152.59 ms
A*-MANHATTAN	56	839.02 ms
A*-NO-CORNERS	56	434.65 ms
A*-NO-DEADLOCKS	56	403.20 ms

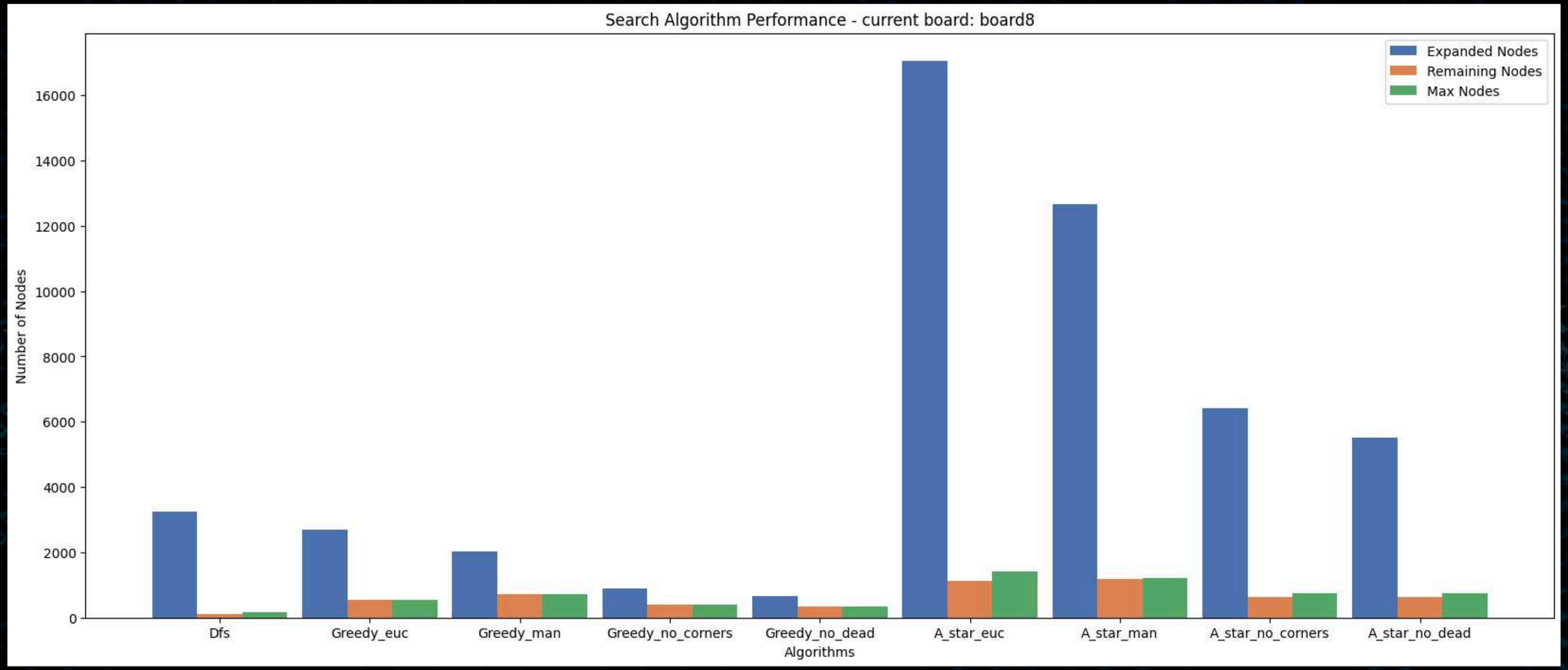
Desvíos de datos



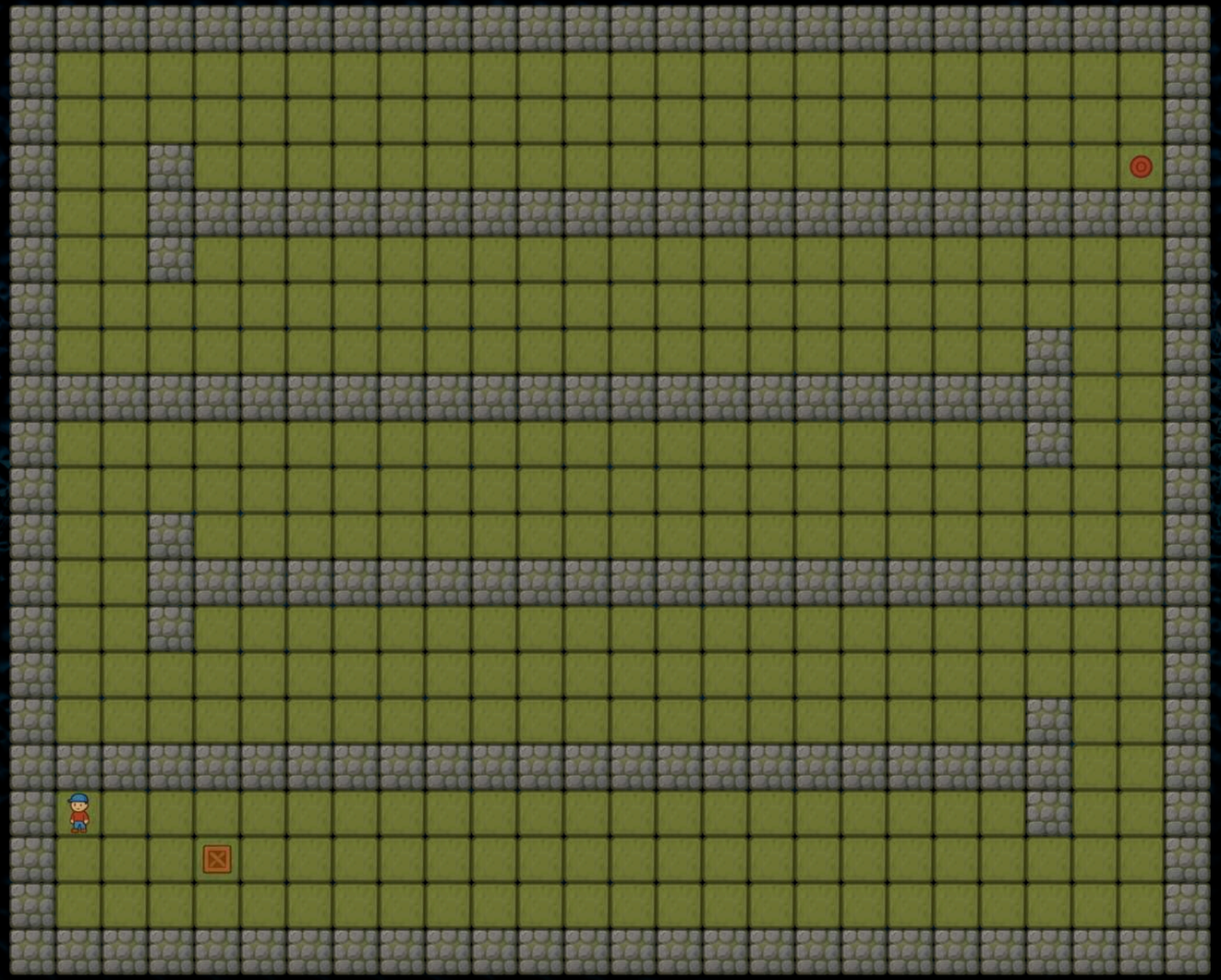
Algorithm Performance

ALGORITMO	NODOS EXPANDIDOS	NODOS RESTANTES EN FRONT.	NODOS MÁXIMO EN FRONT.
BFS	23671	738	925
DFS	3232	98	149
G-EUCLIDEAN	2692	552	553
G-MANHATTAN	2016	726	727
G-NO-CORNERS	877	396	397
G-NO-DEADLOCKS	650	334	335
A*-EUCLIDEAN	17052	1128	1408
A*-MANHATTAN	12666	1185	1208
A*-NO-CORNERS	6407	623	737
A*-NO-DEADLOCKS	5497	622	736

Algorithm Performance



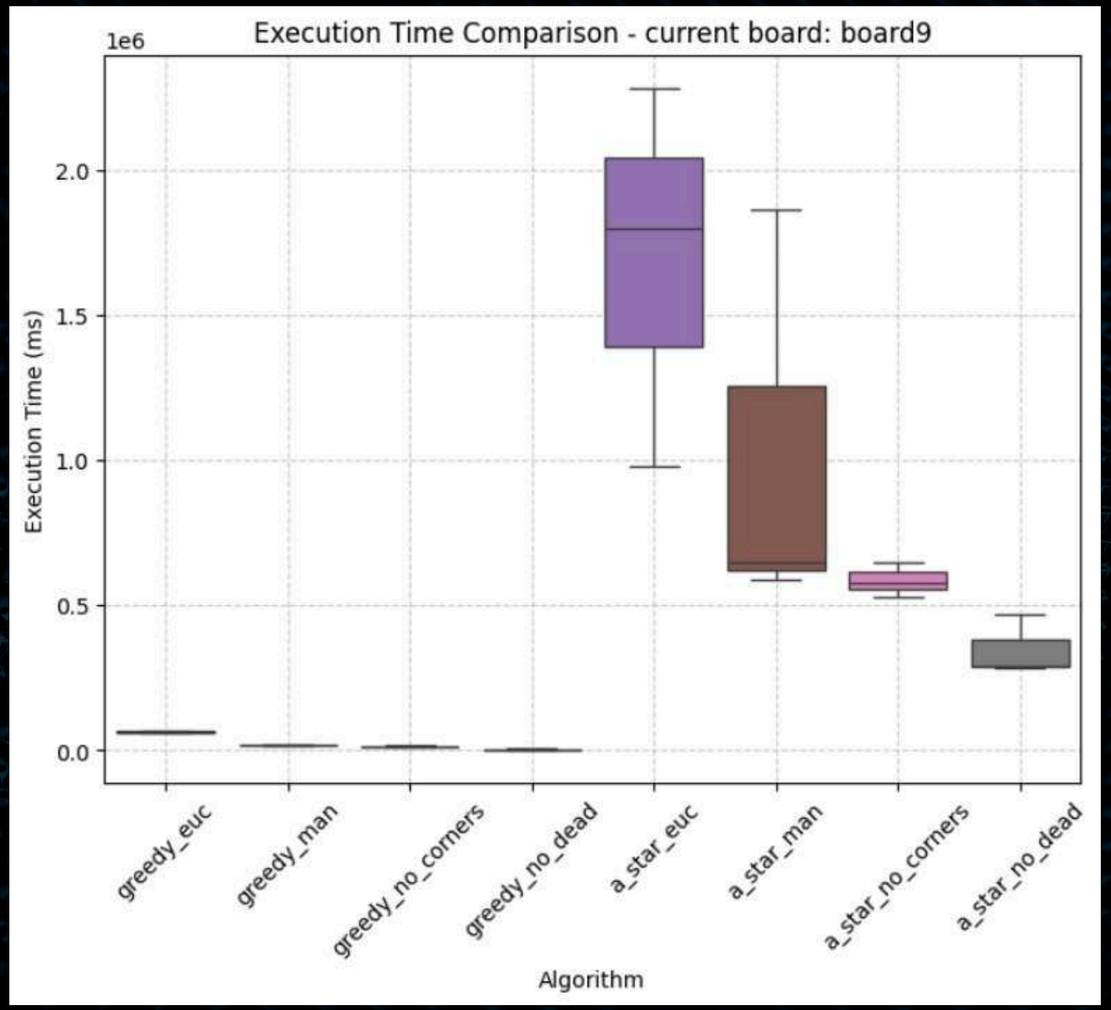
Tablero 9

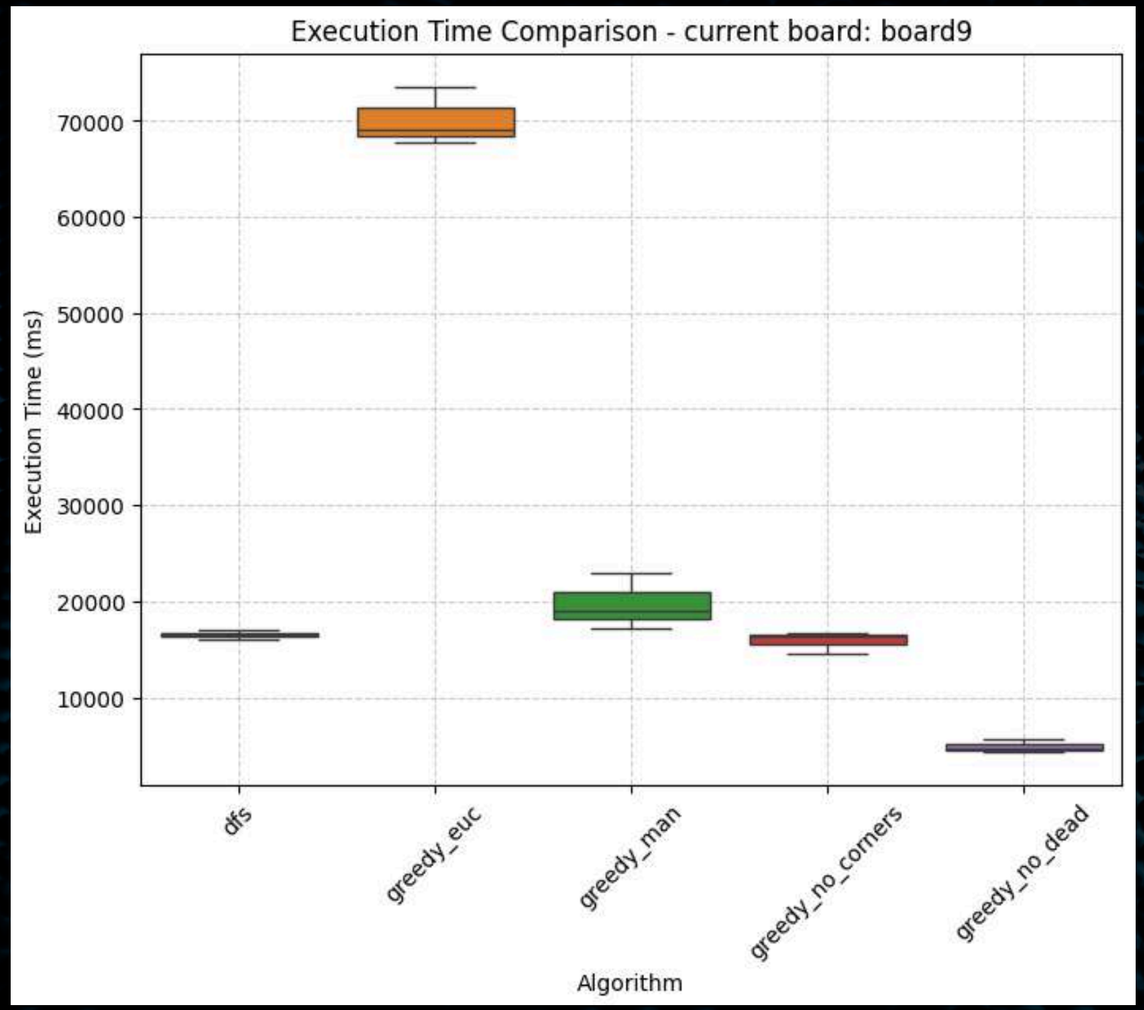


Resultados:

ALGORITMO	COSTO DE SOLUCIÓN	TIEMPO DE EJECUCIÓN MEDIO
BFS	142	1186037.28 ms
DFS	182	17720.64 ms
G-EUCLIDEAN	142	65320.42 ms
G-MANHATTAN	142	18682.88 ms
G-NO-CORNERS	142	15510.20 ms
G-NO-DEADLOCKS	142	4578.87 ms
A*-EUCLIDEAN	142	1686042.55 ms
A*-MANHATTAN	142	1032395.71 ms
A*-NO-CORNERS	142	584817.24 ms
A*-NO-DEADLOCKS	142	349340.68 ms

Desvíos de datos



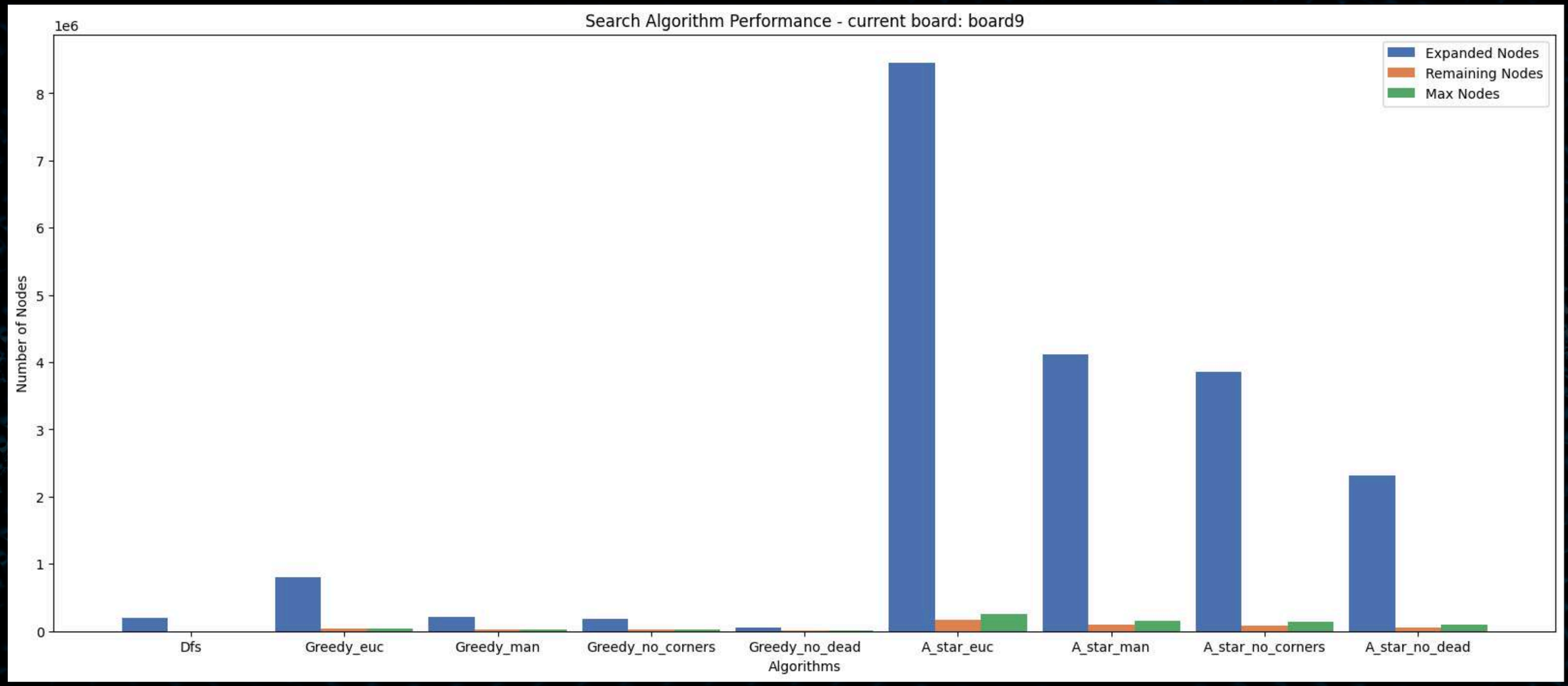


Nota: observar que la escala del eje “Execution Time” está multiplicada por 10^6

Algorithm Performance

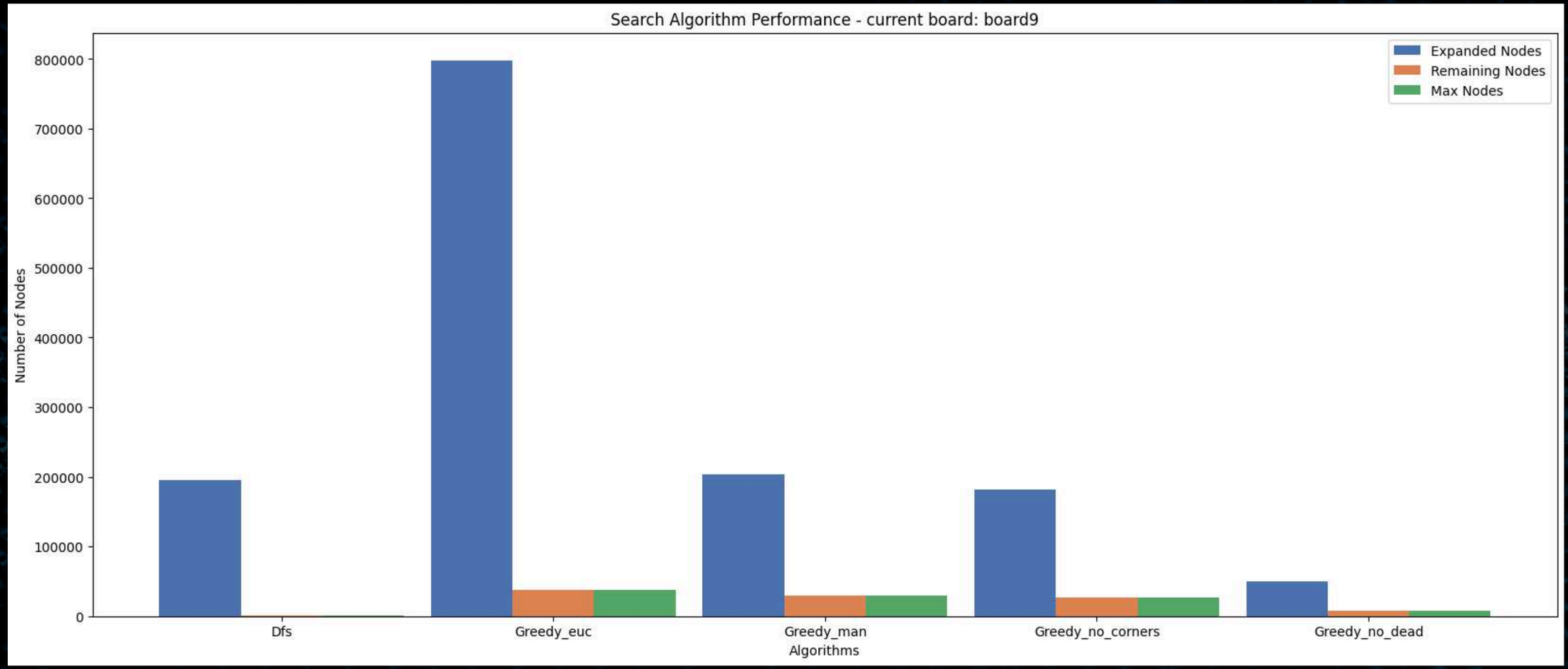
ALGORITMO	NODOS EXPANDIDOS	NODOS RESTANTES EN FRONT.	NODOS MÁXIMO EN FRONT.
BFS	12359287	87504	282623
DFS	194797	118	344
G-EUCLIDEAN	797926	37279	37674
G-MANHATTAN	203109	28817	29203
G-NO-CORNERS	182033	26873	27010
G-NO-DEADLOCKS	49755	7986	7987
A*-EUCLIDEAN	8451779	162152	257000
A*-MANHATTAN	4116729	94343	154669
A*-NO-CORNERS	3850863	87340	141806
A*-NO-DEADLOCKS	2315185	58024	92836

Algorithm Performance



Nota: observar que la escala del eje “Execution Time” está multiplicada por 10^6

Algorithm Performance



Conclusiones - Elección de heurísticas

Cuando los mapas presentan muchos estados muertos se puede notar que los algoritmos que aplican heurísticas que descartan estos estados son significativamente más rápidos y eficientes, a pesar de la extra complejidad de cómputo de estas heurísticas.

Conclusiones - Soluciones Óptimas vs Soluciones Rápidas

En base a los resultados, podemos hacer una distinción entre:

- Buscar **soluciones óptimas**, es decir, que encuentren la solución de menor costo; sin importar cuanto tarda en encontrarlas. Como en los casos de **BFS** y **A***
- Buscar **soluciones rápidas**, sin importar si son las óptimas. Como en los casos de **Global Greedy Search**, dónde no se realiza tanto backtracking
- Luego tenemos a **DFS** que, por lo analizado, depende mucho del mapa, cantidad de cajas y orden de acciones si va a ser rápido o no, pero definitivamente no será óptimo

Conclusiones - Algoritmos Eficientes vs Algoritmos Ineficientes

Fuertemente ligado a la conclusión anterior, pero también podemos hacer una distinción entre:

- **Algoritmos eficientes**, que encuentra la solución expandiendo y guardando en la frontera una **baja cantidad** de nodos; siendo computacionalmente menos costosos. Como en los casos de **Global Greedy Search**, donde no se realiza tanto backtracking
- **Algoritmos ineficientes**, donde se da el caso opuesto al anterior. Como en los casos de **BFS** y **A***
- Nuevamente tenemos a **DFS** que, depende mucho del mapa, cantidad de cajas y orden de acciones si va a ser eficiente o no

The End