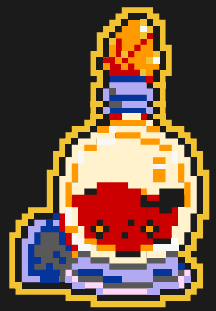
SIA-TP1

Métodos de Búsqueda

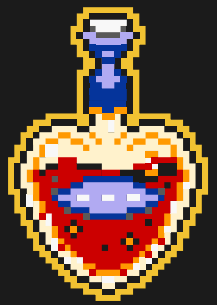
Grupo 6

- Francisco Sendot
- Juan Ignacio Fernández Dinardo
- Lucia Digon
- Martín E. Zahnd



01

8-Puzzle



- Tablero 3x3 con 8 fichas (cada una numerada del 1 al 8) y un espacio vacío
- El objetivo es colocar los números de forma que coincidan con la configuración final utilizando el espacio vacío
- Podemos deslizar cuatro fichas adyacentes (izquierda, derecha, arriba y abajo) en el espacio vacío.

Estado inicial

5	7	3
2	8	
1	6	4

->

Estado Final

1	2	3
8		4
7	6	5

!!!

Definimos un estado del problema como *la ubicación de cada pieza en una celda, incluido el espacio vacío.*

Heuristicas Admisibles

El número de fichas mal colocadas comparando el estado actual y el estado objetivo.

Las fichas 5, 1, 4, 2, 7 y 8 están fuera de lugar, por lo que estimación = 6.

Estado actual

5	7	3
2	8	
1	6	4

Estado Final

1	2	3
8		4
7	6	5

Heuristicas Admisibles

La suma de las distancias de cada ficha a su posición de meta (Manhattan distance)

estimación = 4 + 3 + 0 + 2 + 1 + 2 + 0 + 1 = 13.

Estado actual

5	7	3
2	8	
1	6	4

Estado Final

1	2	3
8		4
7	6	5

Heuristicas Admisibles

Como el anterior, pero tiene en cuenta el número de inversiones de fichas adyacentes directas presentes. Tener muchas inversiones adyacentes sugiere que la configuración actual está lejos de la solución, ya que hay que hacer movimientos adicionales para corregir cada inversión, lo que lo hace más preciso al reflejar el «esfuerzo» necesario para resolver el puzzle.

las fichas 4 y 7 forman una inversión directa, por lo que la estimación = 6 + 2 × 1 = 8

Estado actual

1	2	3
7	8	5
4	6	

Estado Final

1	2	3
4	5	6
7	8	

Algoritmos !!!

Pros

A*	IDA*
óptimo y completo con heurística admisible	óptimo y completo con heurística admisible
Generalmente más rápido gracias a la expansión controlada de los nodos	Uso de memoria mucho menor
Adecuado para problemas en los que la memoria no es una limitación.	Mejor para grandes problemas o entornos de memoria limitada

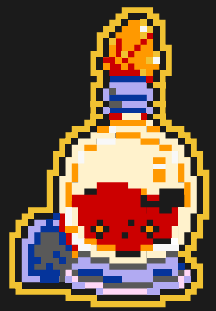
Algoritmos !!!

Contras

A*	IDA*
Alto uso de memoria, almacenando todos los nodos generados	Más lento debido a las búsquedas repetidas a profundidades cada vez mayores
Puede ser prohibitivo para problemas grandes	Puede requerir más tiempo de cálculo que A*

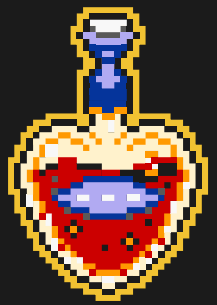
Algoritmos !!!

Si se dispone de suficiente memoria y se desea buscar la solucion mas rapida posible, A* puede ser una muy buena opción. Si se dispone de una cantidad limitada de memoria y se necesitan controlar los recursos, entonces IDA* es mejor opción



02

Sokoban



- Tablero rectangular con un personaje, cajas y objetivos.
- El objetivo es mover todas las cajas hacia las posiciones objetivo en el tablero.
- El personaje puede moverse en cuatro direcciones: arriba, abajo, izquierda y derecha.
- El personaje solo puede empujar una caja a la vez y únicamente si hay un espacio libre detrás de la caja en la dirección del movimiento.
- El personaje no puede atravesar paredes ni cajas, y solo puede empujar, no tirar de las cajas.
- El juego se resuelve cuando todas las cajas han sido movidas a las posiciones objetivo, utilizando el menor número de movimientos posible.

!!!

Defnimos un estado del problema como *la ubicación del personaje y cada caja.*

* Personaje
# Pared
@ Caja
X Objetivo

Heuristicas Admisibles

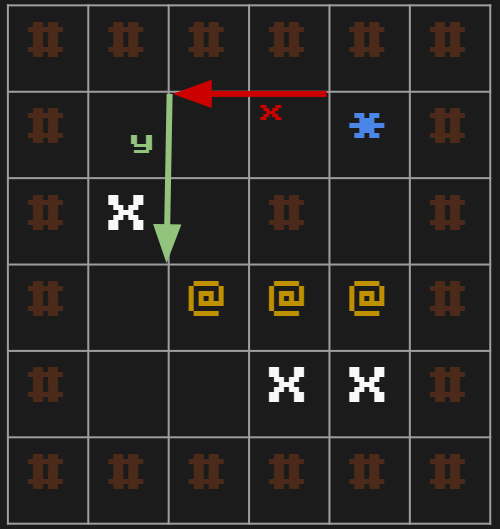
La distancia euclídea calcula la distancia del personaje a una caja y de la caja a cada objetivo.

Aunque la distancia euclidiana da una buena idea de la proximidad de una caja a su objetivo, no considera obstáculos como paredes u otras cajas, por lo que es una heurística admisible pero no siempre la más efectiva para este tipo de problemas !!

Heuristicas Admisibles

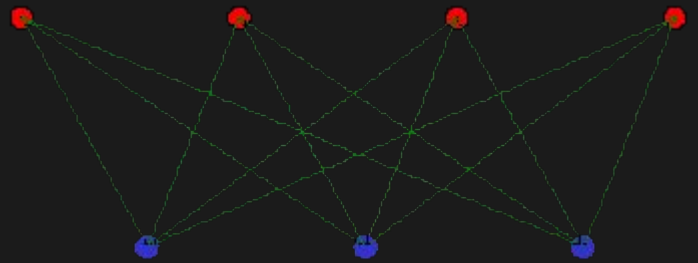
La distancia Manhattan analoga a la euclidea pero calcula distancias ortogonales.

De esta forma, tampoco se consideran obstáculos como las paredes !!



Heuristicas Admisibles

Minimum matching lower bound (MMLB) busca calcular la distancia mínima necesaria para emparejar todas las cajas con los objetivos, considerando que cada caja debe ser emparejada con un objetivo distinto y que este emparejamiento debe ser el más eficiente posible.



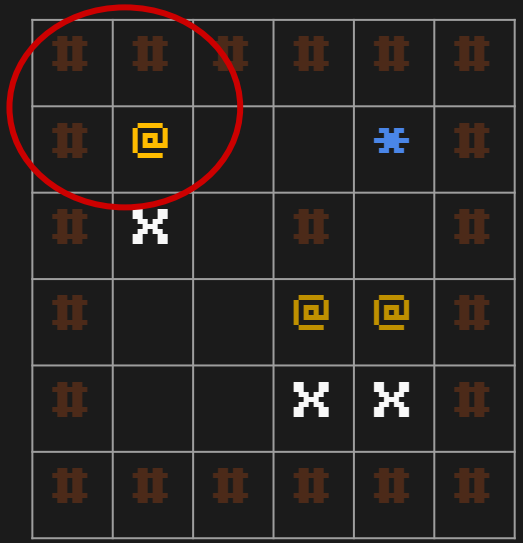
Heuristicas Admisibles

Deadlock busca evitar caer en un deadlock,lo cual hace imposible resolver el nivel.

Para eso, chequea si un estado tiene deadlock, y si lo tiene devuelve un costo muy alto.

No chequea todos los tipos de deadlock, sino que únicamente el caso en que una caja se en una esquina.

deadlock



Heuristicas Admisibles

Combinación de heurísticas

01

MMLB + deadlock

02

MMLB + euclidean

03

Euclidean + deadlock

Métodos desinformados

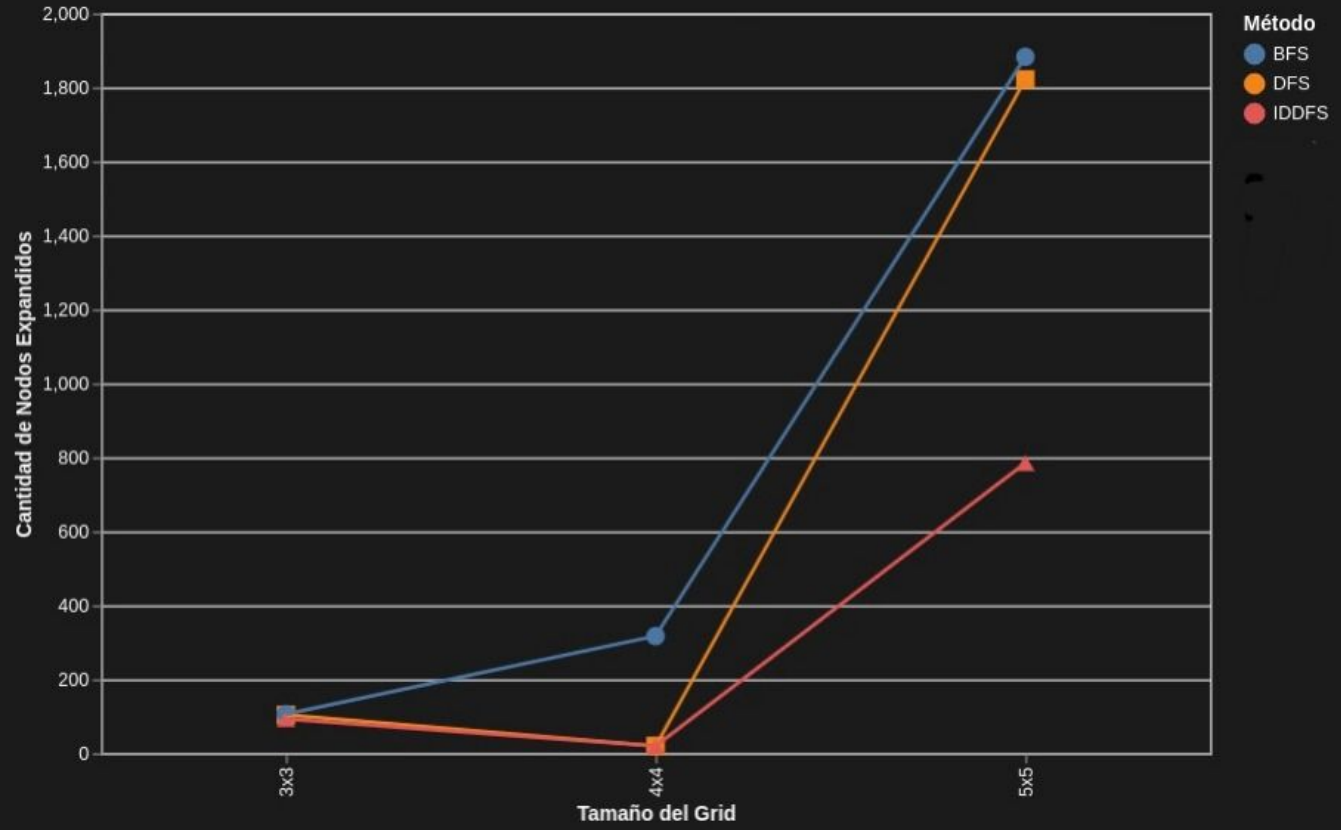
2 cajas

3x3

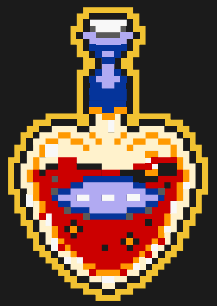
4x4

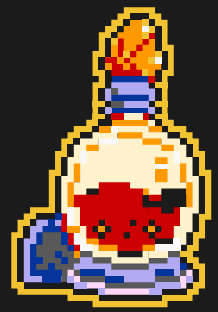
5x5

Resultado: Nodos expandidos

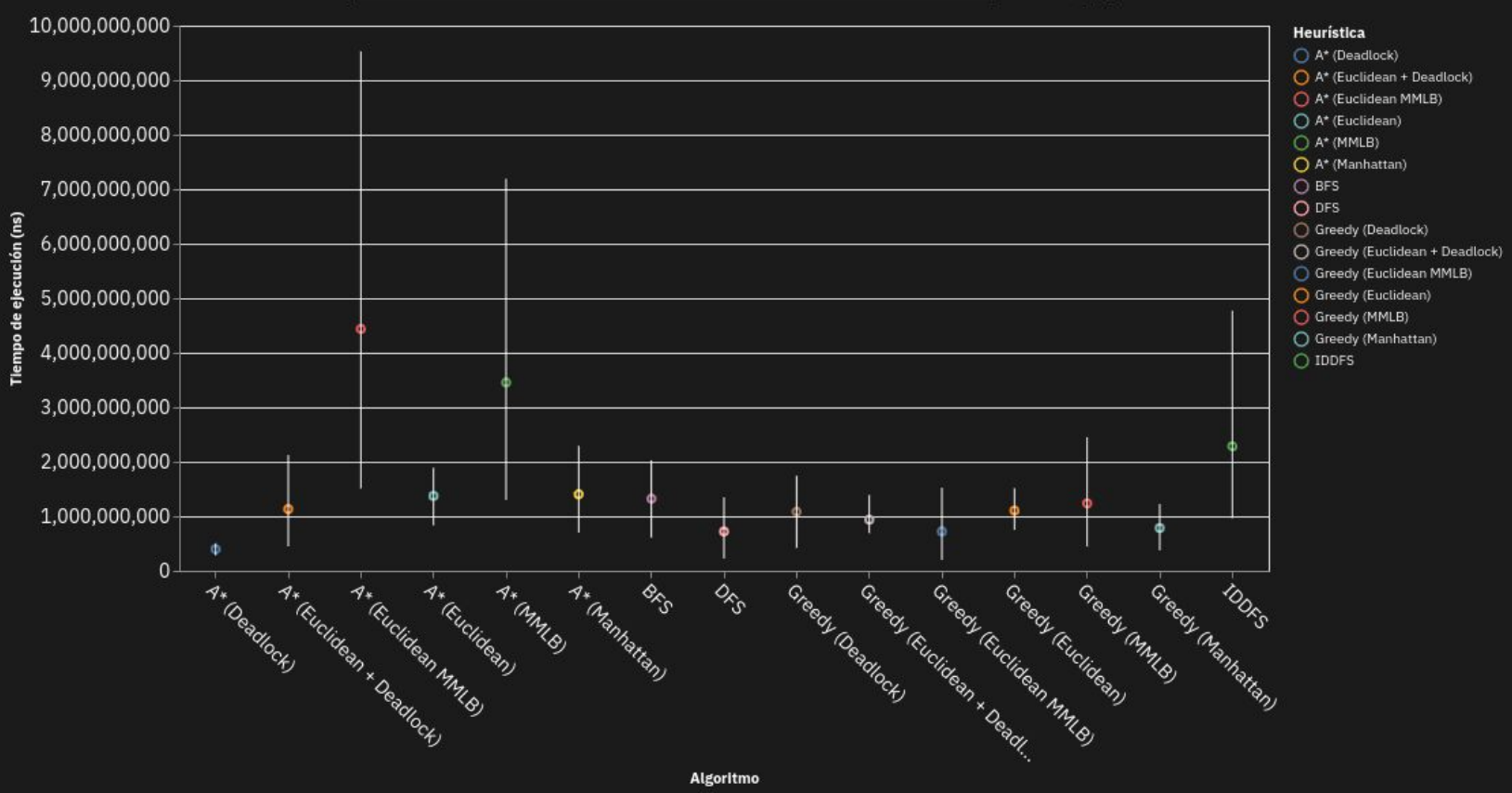


Nivel 53

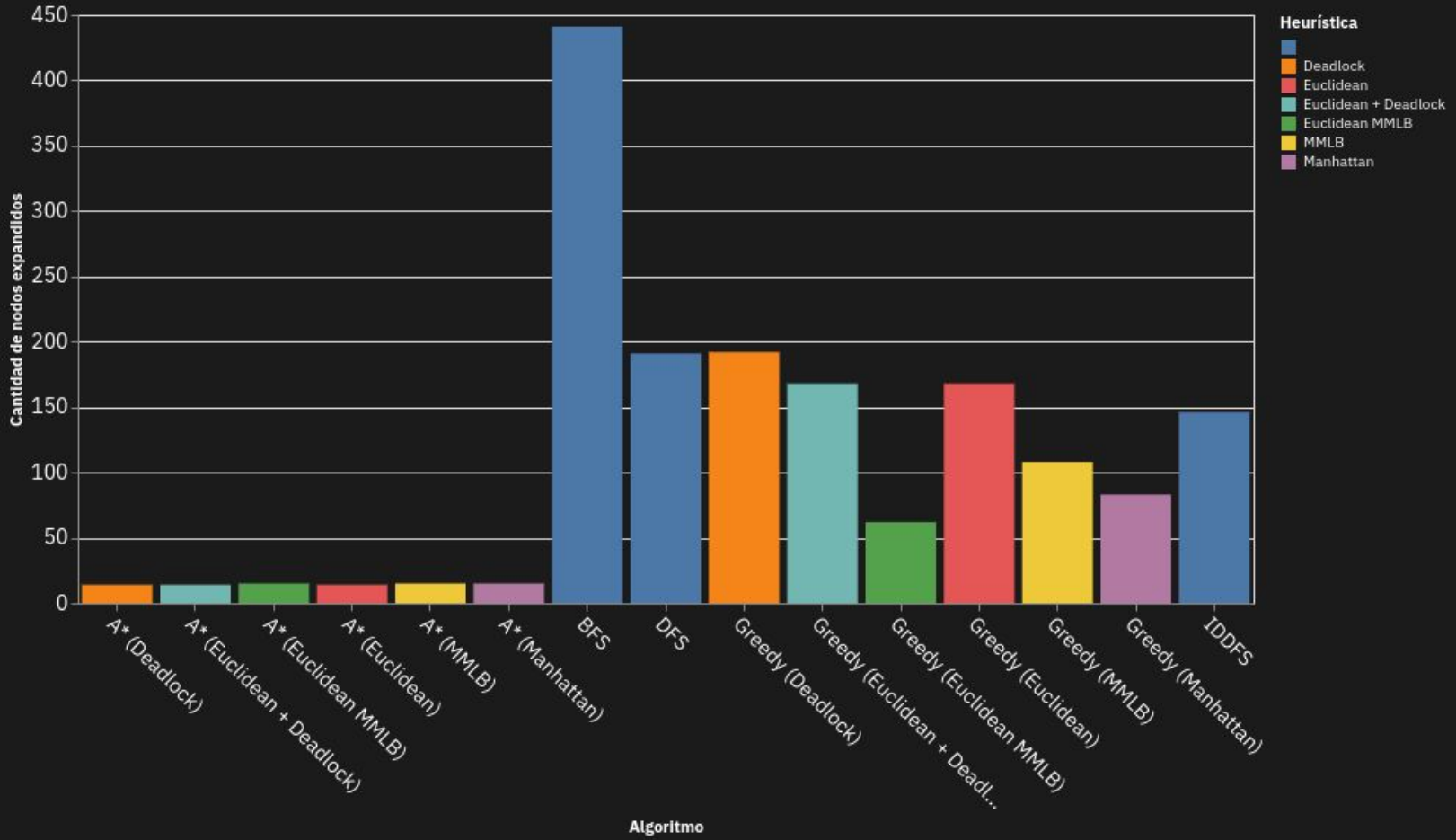




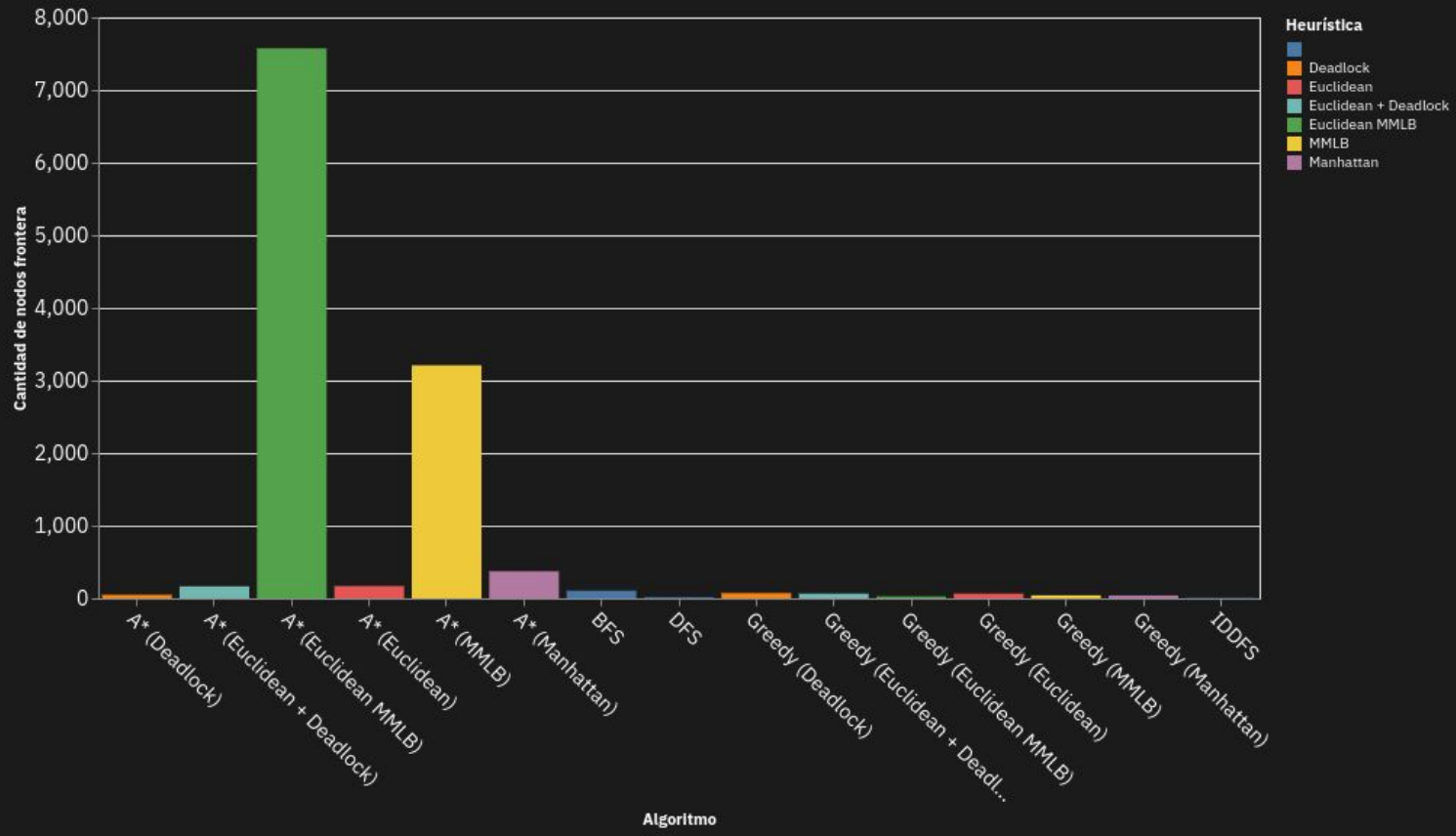
Nivel 53: Tiempo de ejecución promedio



Nivel 53: Nodos expandidos



Nivel 53: Nodos frontera



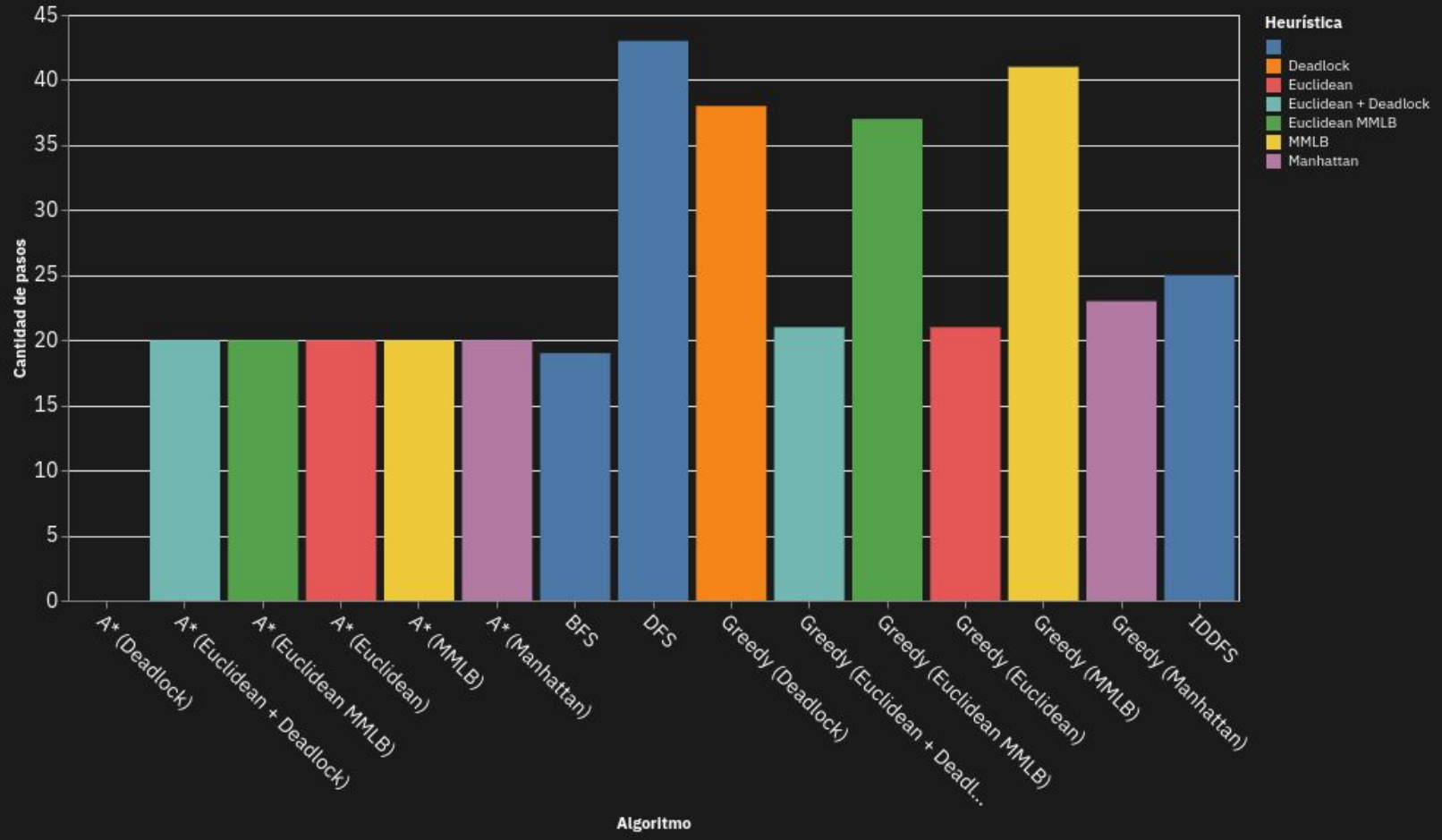
Nivel 53: Nodos explorados vs. nodos frontera

Greedy	Exp	Fr
Deadlock	192	72
Euclidean	168	65
Euclidean + Deadlock	168	65
Manhattan	83	38
MMLB	108	41
Euclidean MMLB	62	26

A*	Exp	Fr
Deadlock	14	49
Euclidean	14	169
Euclidean + Deadlock	14	165
Manhattan	15	372
MMLB	15	3209
Euclidean MMLB	15	7575

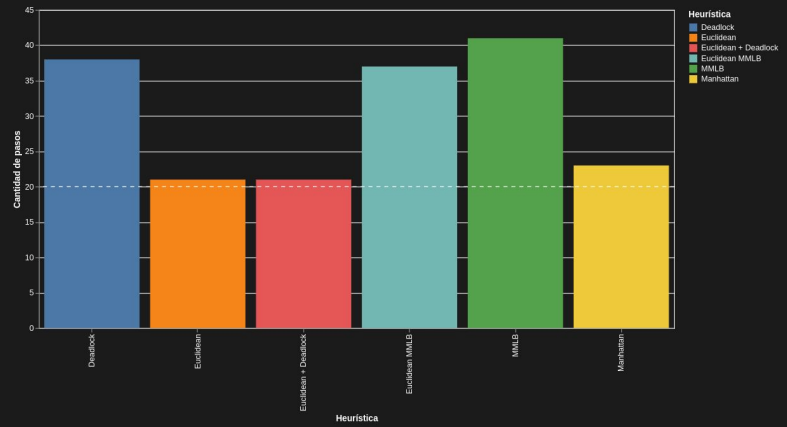
Método	Exp	Fr
BFS	441	105
DFS	191	16
IDDFS	146	8

Nivel 53: Cantidad de pasos

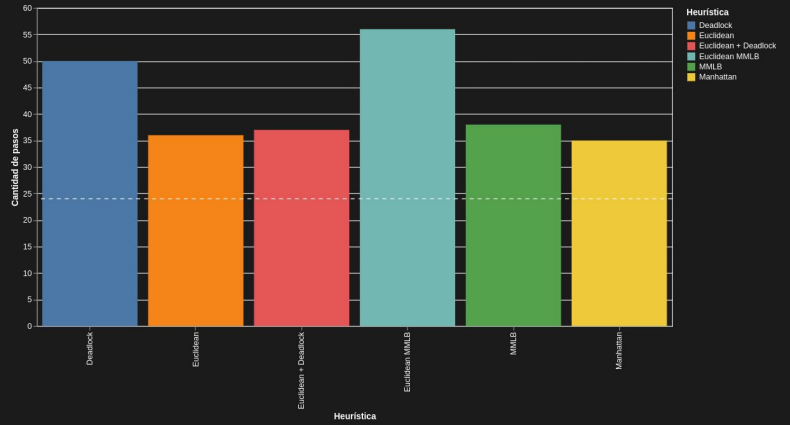


Cantidad de pasos

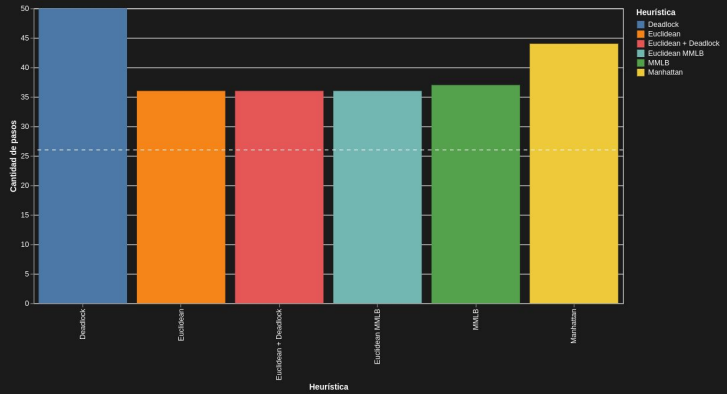
Nivel 53



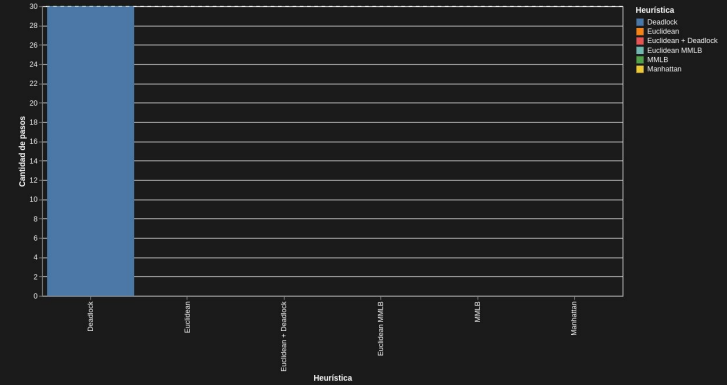
Nivel 40



Nivel 67



Nivel 14

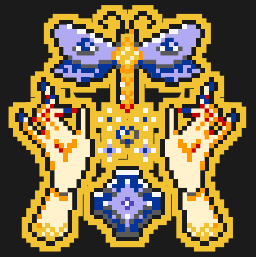


CONCLUSIONES

- No hay una bala de plata para todos los casos
- Las heurísticas influyen muchísimo en el resultado final cuando se usan métodos informados
- Los algoritmos DFS y Greedy nos permiten conseguir una solución en forma breve, pero seguramente no optima
- IDDFS es menos costoso que DFS, pero si se elige mal la profundidad puede no encontrar solución
- A* es más robusto y garantiza una solución más óptima que Greedy, siempre y cuando las heurísticas sean adecuadas
- BFS encuentra soluciones óptimas, pero es muy costoso en cuestión de memoria. Para mapas grandes es una opción que no se puede tener en cuenta

Bibliografia 

- GeeksforGeeks. 8 Puzzle Problem using Branch and Bound. GeeksforGeeks. Available in: https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/
- Kumar, Prateek. Solving 8-Puzzle using A Algorithm*. Good Audience. Available in: https://blog.goodaudience.com/solving-8-puzzle-using-a-algorithm-7b509c331288
- Marshall, J. Heuristic Search. Sarah Lawrence College. Available in: http://science.slc.edu/~jmarshall/courses/2005/fall/cs151/lectures/heuristic-search/
- Apunte de la catedra
- Virkkala, T. (2009). Solving Sokoban (Master's Thesis). Retrieved from https://weetu.net/Timo-Virkkala-Solving-Sokoban-Masters-Thesis.pdf



GRACIAS