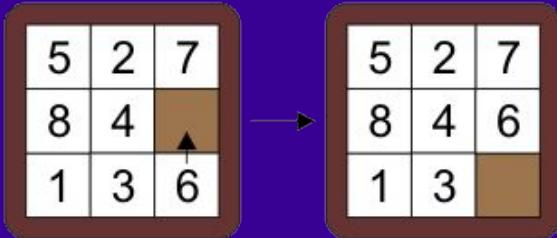
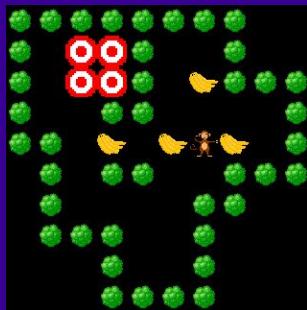
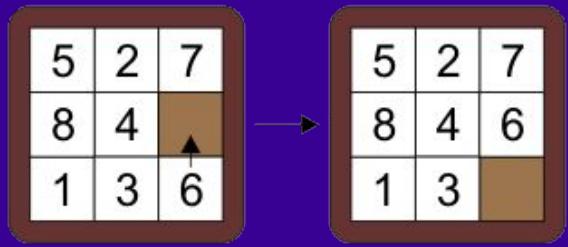


**8 puzzle****Sokoban**

# Grupo 4

Gastón Alasia, 61413  
Juan Segundo Arnaude, 62184  
Bautista Canevaro, 62179  
Matías Wodtke, 62098



# 8-Puzzle

# Cómo se juega?

- El juego consiste en una grilla de 3x3 (9 cuadrados, 8 con un número y uno vacío).
- Solo se puede desplazar en cuatro direcciones (arriba, abajo, izquierda y derecha).
- El objetivo es mover los cuadrados para obtener el estado ganador.

5	2	7
8	4	
1	3	6



5	2	7
8	4	6
1	3	

2	8	3
1	6	4
7		5

Initial State

1	2	3
8		4
7	6	5

Goal State

# Estructura de estado

2	8	3
1	6	4
7		5

Estructura del problema:

Matriz 3x3. El espacio vacío se llena con un 0.

Entonces, guardamos la matriz!!!

# Estructura de estado

2	8	3
1	6	4
7		5

Estructura del problema:

Matriz 3x3. El espacio vacío se llena con un 0.

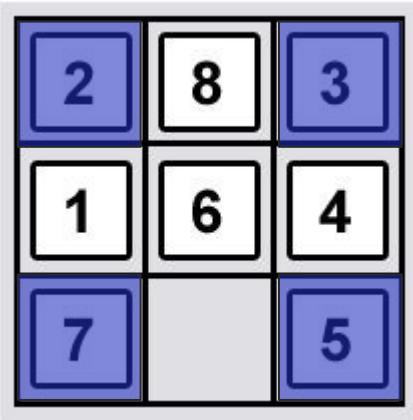
Entonces, guardamos la matriz!!!

~	→	2
	6	8
5	4	3

5		7
4	6	1
3	8	2

3	4	5
8	6	
2	1	7

# Estructura de estado

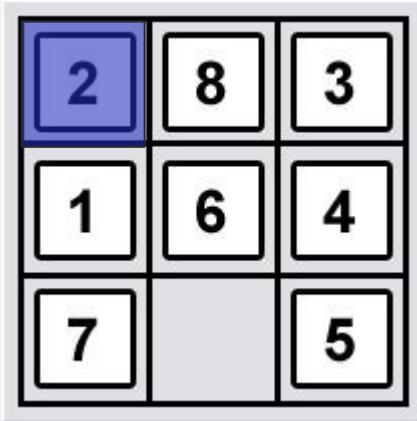


Se toman las 4 esquinas.

Ordenamos las esquinas de manera ascendente:

2 , 3 , 5 , 7

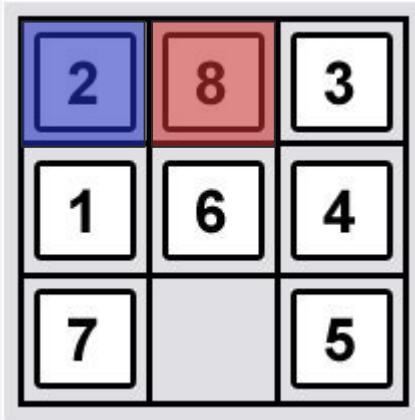
# Estructura de estado



Para cada esquina guardamos el valor propio.

Valor(2)=**2**

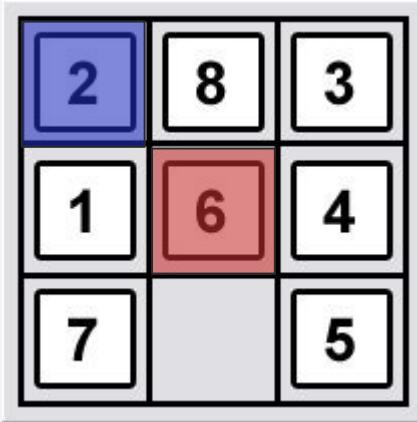
# Estructura de estado



Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2) = 28$$

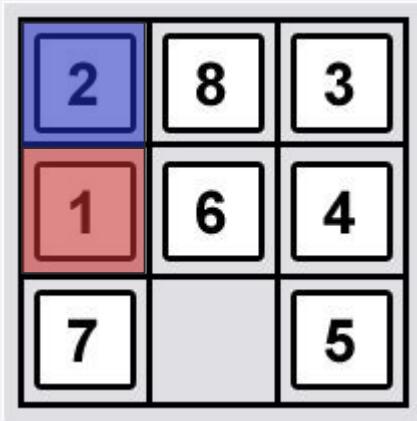
# Estructura de estado



Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2) = 286$$

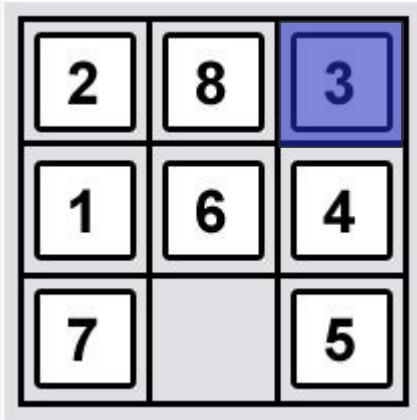
# Estructura de estado



Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2) = 2861$$

# Estructura de estado

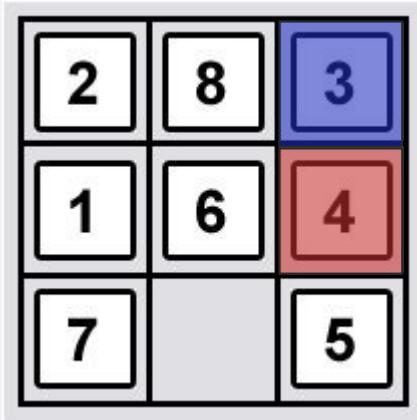


Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2) = 2861$$

$$\text{Valor}(3) = 3$$

# Estructura de estado

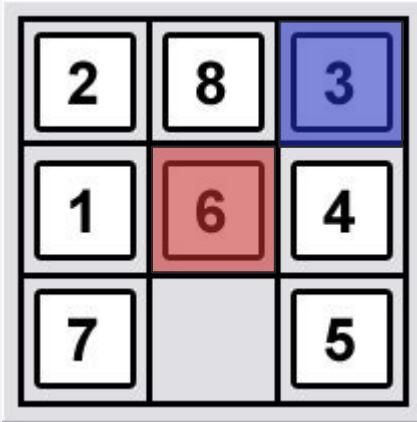


Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2) = 2861$$

$$\text{Valor}(3) = 34$$

# Estructura de estado

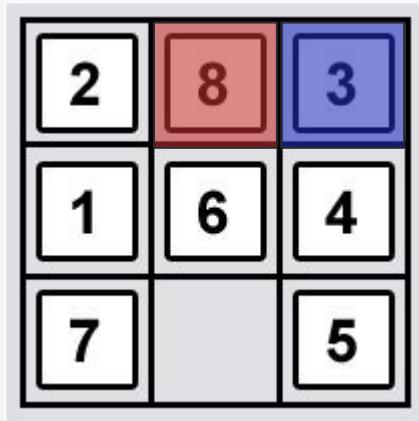


Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2) = 2861$$

$$\text{Valor}(3) = 346$$

# Estructura de estado

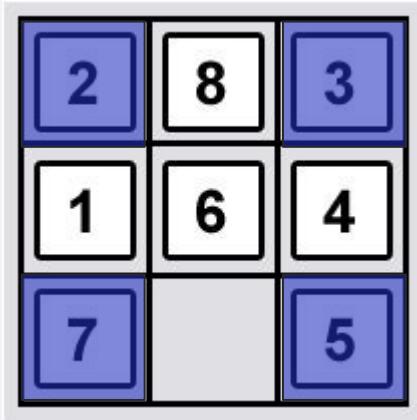


Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2) = 2861$$

$$\text{Valor}(3) = 3468$$

# Estructura de estado



Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2)=2861$$

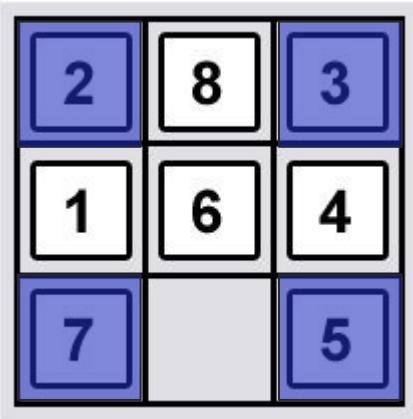
$$\text{Valor}(3)=3468$$

Hacemos lo mismo para las esquinas del 5 y el 7.

$$\text{Valor}(5)=5064$$

$$\text{Valor}(7)=7160$$

# Estructura de estado



Para cada esquina guardamos el valor propio y guardamos el valor de los números que lo rodean con orden horario.

$$\text{Valor}(2)=2861$$

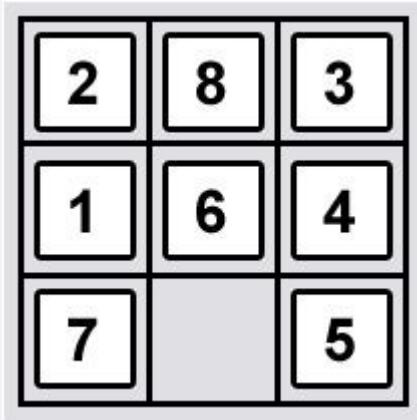
$$\text{Valor}(3)=3468$$

Hacemos lo mismo para las esquinas del 5 y el 7.

$$\text{Valor}(5)=5064$$

$$\text{Valor}(7)=7160$$

# Estructura de estado



Valor(2)= 2861

Valor(3)= 3468

Valor(5)= 5064

Valor(7)= 7160

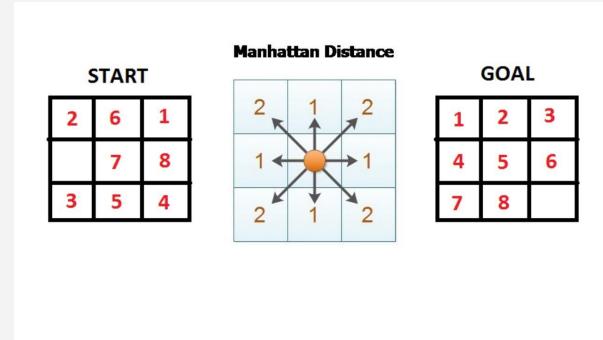
Finalmente unimos todos los resultados:

State = **2861346850647160**

# Heurísticas admisibles posibles

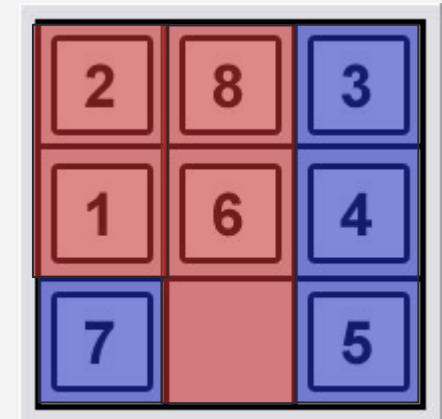
- **Suma de distancias Manhattan**

- Se calcula la distancia del bloque a su posición deseada.
- Es admisible porque cada bloque se debe mover, al menos, su distancia.



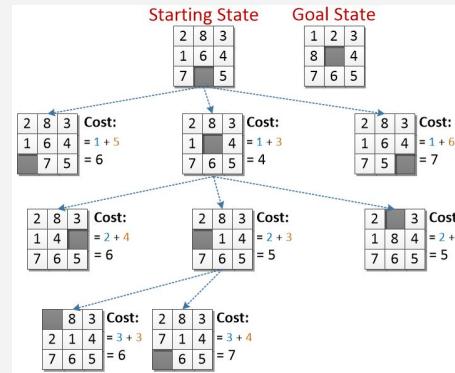
- **Cantidad de bloques en un lugar erróneo**

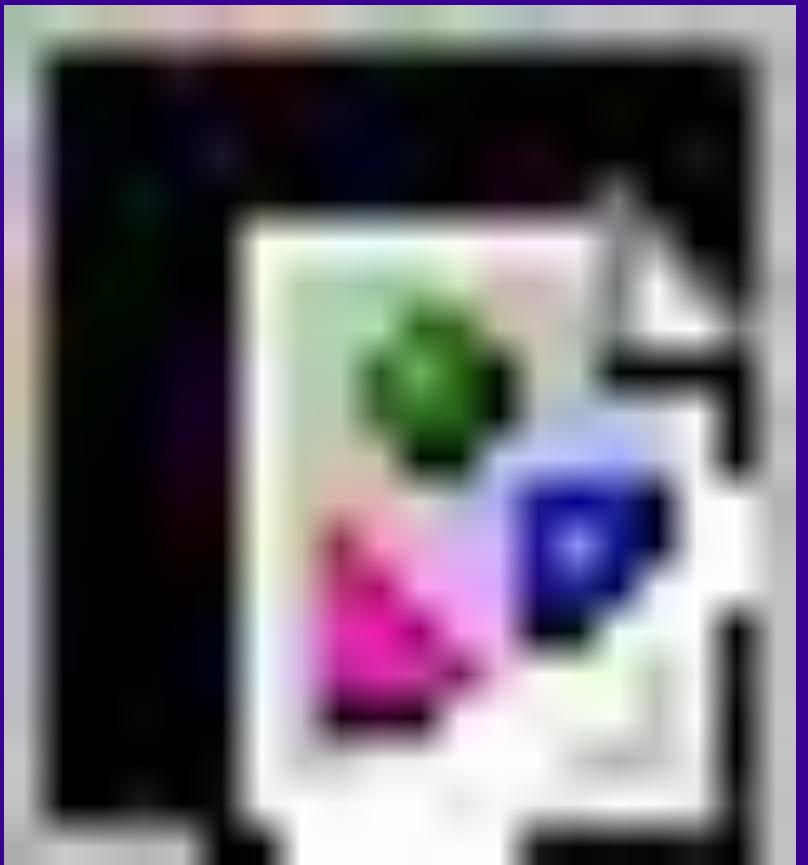
- Se calcula la cantidad de bloques que no se encuentran en su lugar correspondiente.
- Es admisible, pues cada bloque fuera de lugar deberá moverse como mínimo un lugar para llegar a la solución.



# Métodos de Búsqueda

- **Métodos de búsquedas: A\* y Local Greedy**
  - A\* encuentra la solución óptima para cualquier heurística admisible, es rápido pero consume muchos recursos.
  - Local Greedy: es posible que encuentre soluciones no óptimas pero generalmente consume menos recursos que A\*.
- **Heuristica: Manhattan, cantidad de bloques en un lugar erróneo.**
  - También se puede usar cualquier tipo de heurística admisible.





# Sokoban

# Cómo se juega?

- El jugador deberá empujar las bananas hasta dejarlas todas en las posiciones correspondientes para ganar.
- Solo se puede desplazar en cuatro direcciones (arriba, abajo, izquierda y derecha).
- Existe una tabla de posiciones de movimientos.
- Aquel jugador que logre completar el nivel en menor cantidad de movimientos, será quien lidere dicha tabla.

# Estructuras de estado

- Posición del jugador
- Posición de las cajas

# Métodos de búsqueda

- Desinformados
  - BFS
  - DFS
  - IDDFS
- Informados
  - A\*
  - Local Greedy Search
  - Global Greedy Search

# Criterio de elección de niveles

- Se decidió analizar 3 niveles distintos ya que consideramos que se cubren la gran mayoría de los escenarios usando dicha cantidad.
- Se optó por tener variedad en la dificultad de los niveles con el fin de obtener conclusiones precisas.
- Estos niveles son:
  - Nivel 1 (difícil)
  - Nivel 7 (intermedio)
  - Nivel 4 (fácil)

# Niveles analizados

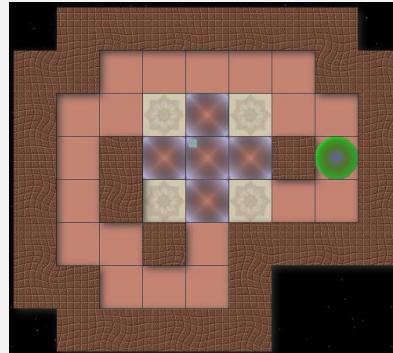
## Nivel 01



Nivel sencillo sin muchas opciones y con solamente 3 cajas

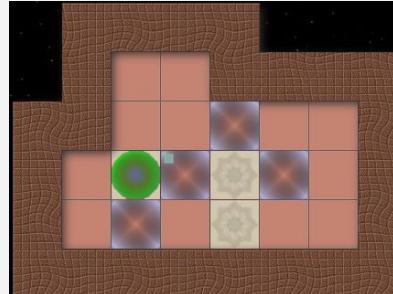
Buena cantidad de espacio para moverse (abre más posibilidades) pero con las cajas lejos de los objetivos

## Nivel 3



Espacios limitados de movimiento pero con una gran cantidad de cajas en relación al tamaño del mapa

## Nivel 7



# Métodos de búsqueda desinformados

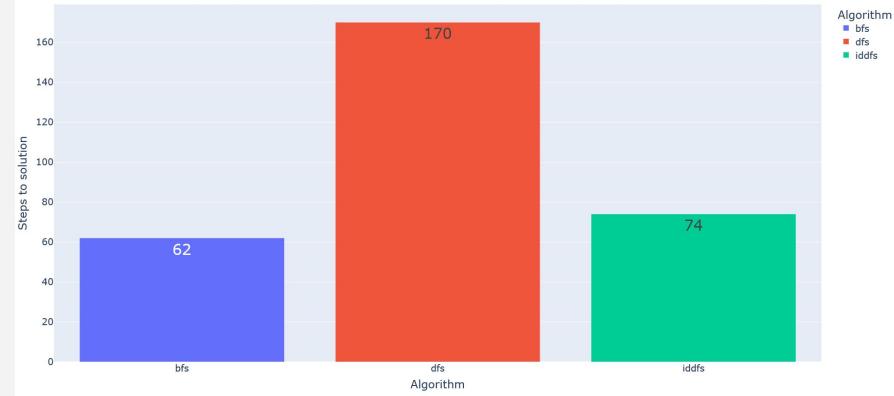
Se plantean las siguientes preguntas disparadoras con el objetivo de analizar los parámetros más relevantes de cada método:

- ¿Qué tan óptima es la solución de cada algoritmo?
- ¿Cuán costoso es cada algoritmo?

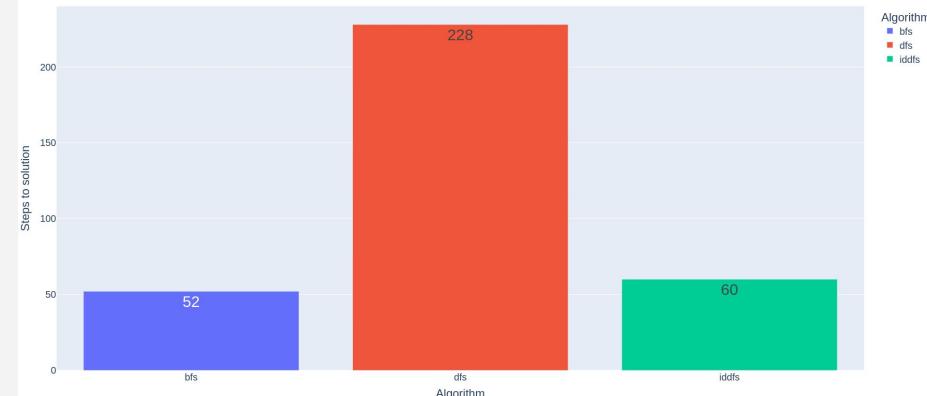


# Optimización en desinformados

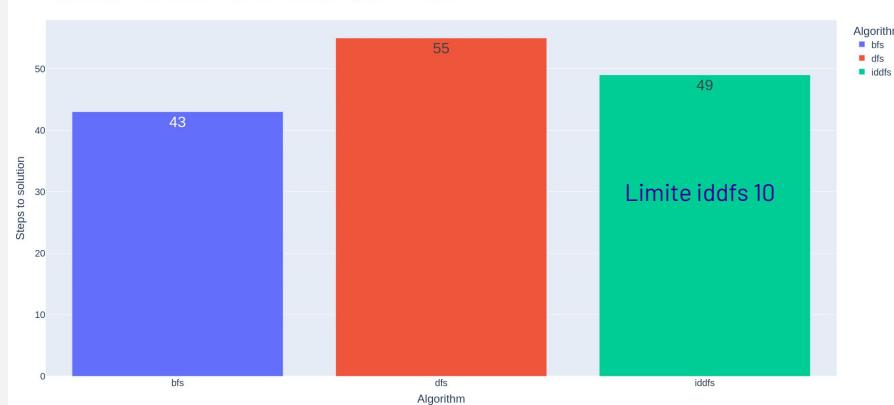
Optimality of Solutions Across Uninformed Algorithms - Level lv01



Optimality of Solutions Across Uninformed Algorithms - Level lv3

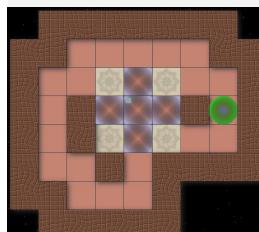


Optimality of Solutions Across Uninformed Algorithms - Level lv7

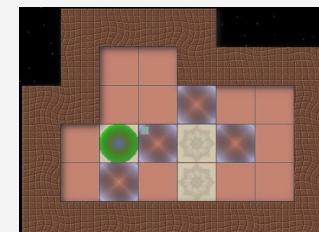


Recordemos los niveles...

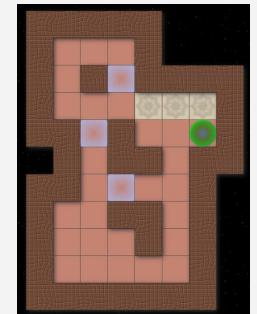
Nivel 3



Nivel 7

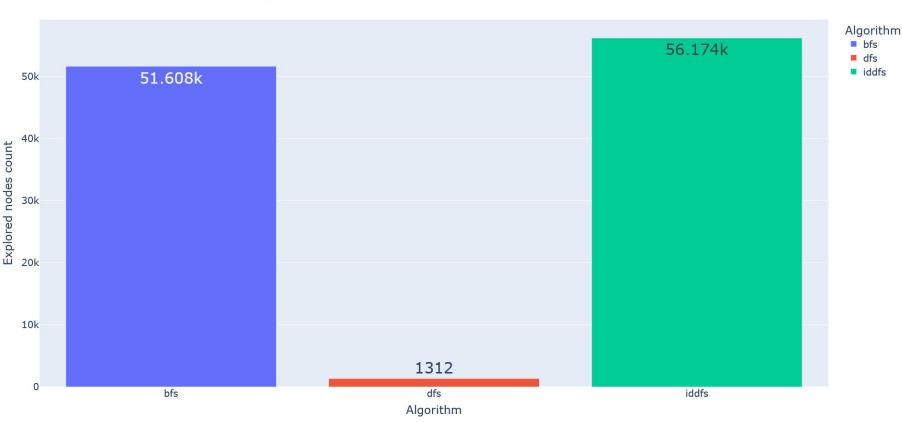


Nivel 01

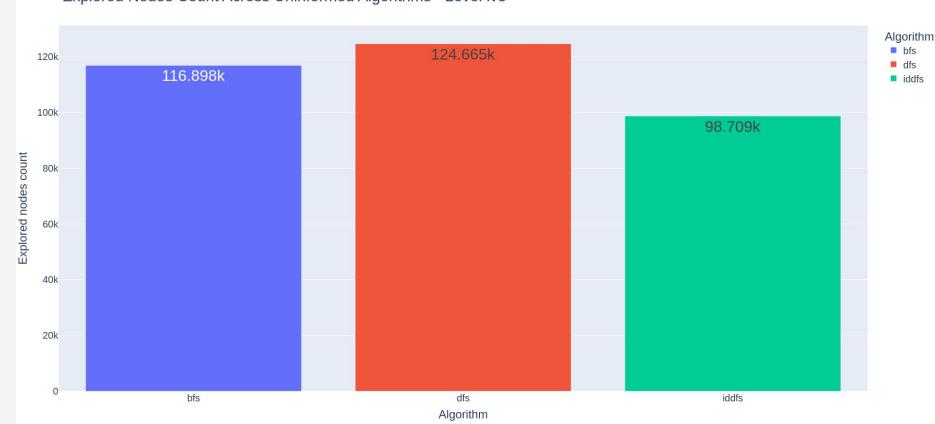


# Nodos explorados en desinformados

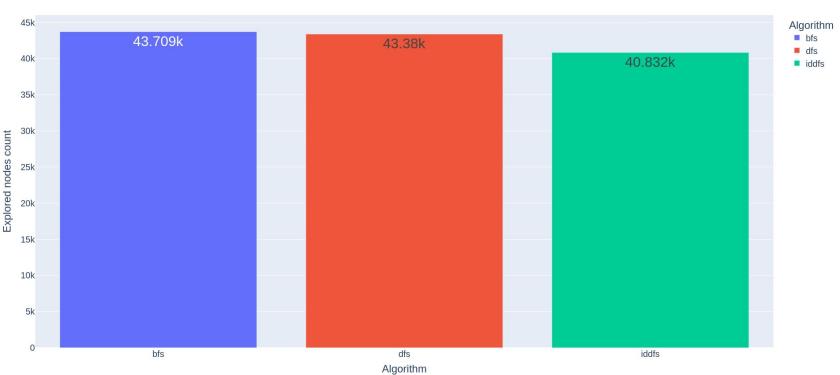
Explored Nodes Count Across Uninformed Algorithms - Level lv01



Explored Nodes Count Across Uninformed Algorithms - Level lv3



Explored Nodes Count Across Uninformed Algorithms - Level lv7



Recordemos los niveles...

Nivel 3



Nivel 7



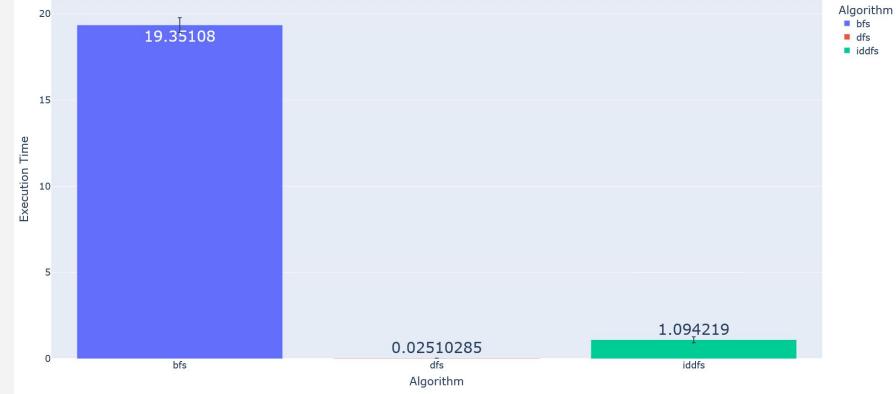
Nivel 01



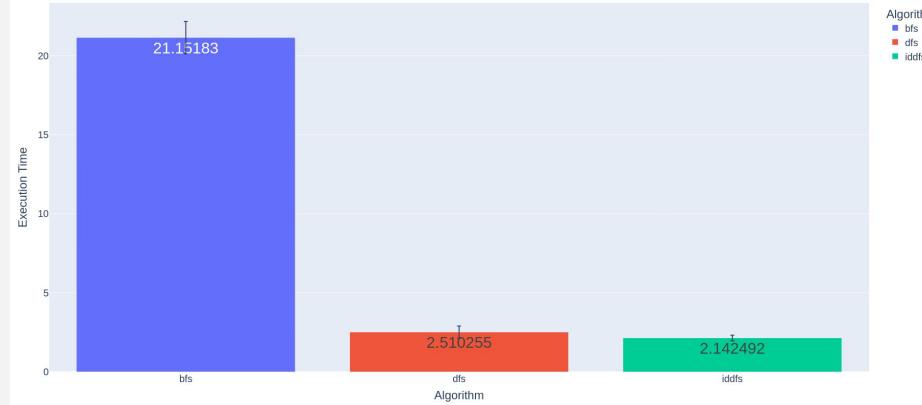
# • Tiempos de ejecución en desinformados

Se utilizaron 10 iteraciones para generar el data set

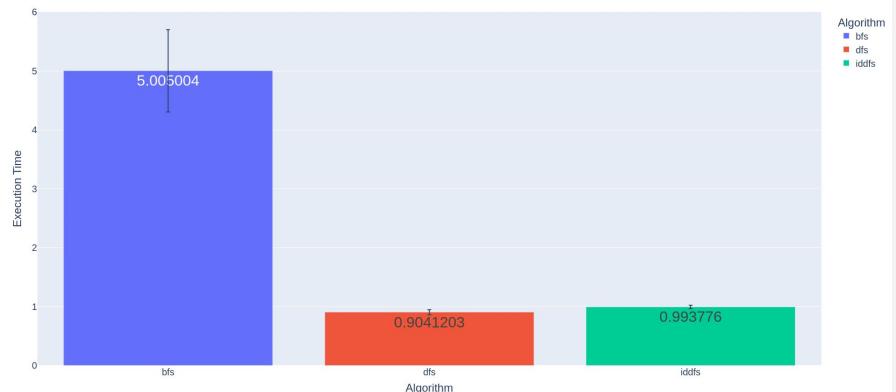
Execution Time Across Uninformed Algorithms- Level lv01



Execution Time Across Uninformed Algorithms- Level lv3

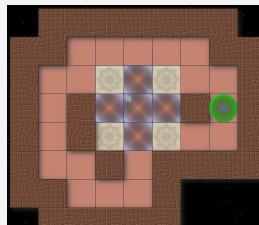


Execution Time Across Uninformed Algorithms- Level lv7

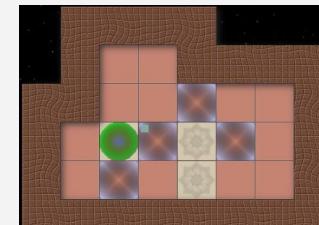


Recordemos los niveles...

Nivel 3



Nivel 7



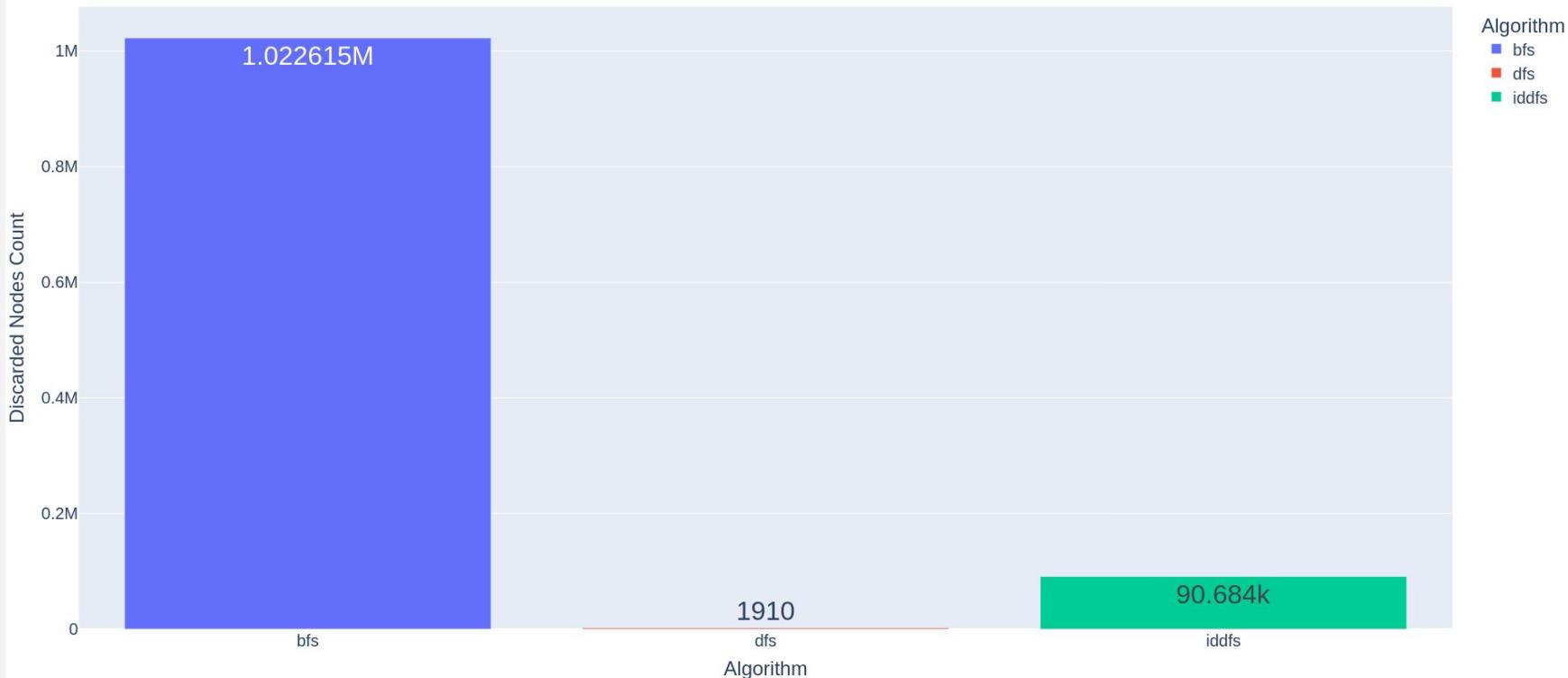
Nivel 01





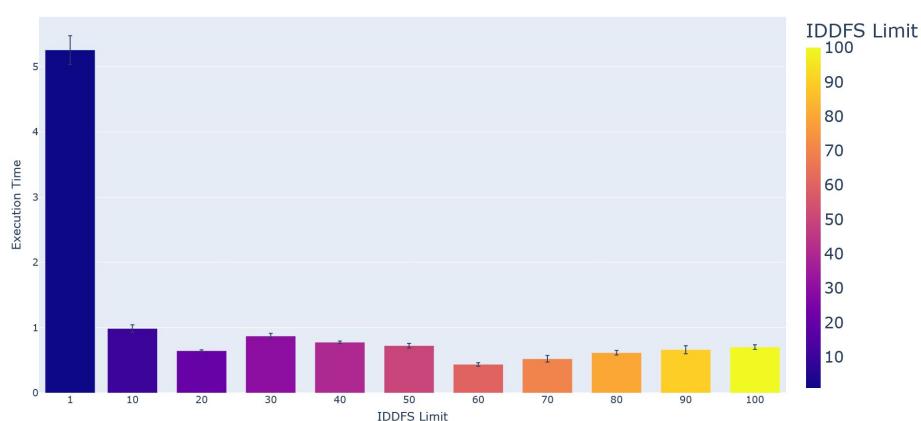
Si bfs e iddfs exploran una cantidad de nodos similar entonces:  
¿Porqué tienen tiempos de ejecución que difieren en ordenes de magnitud?

Discarded Nodes Count Across Algorithms - Level lv01

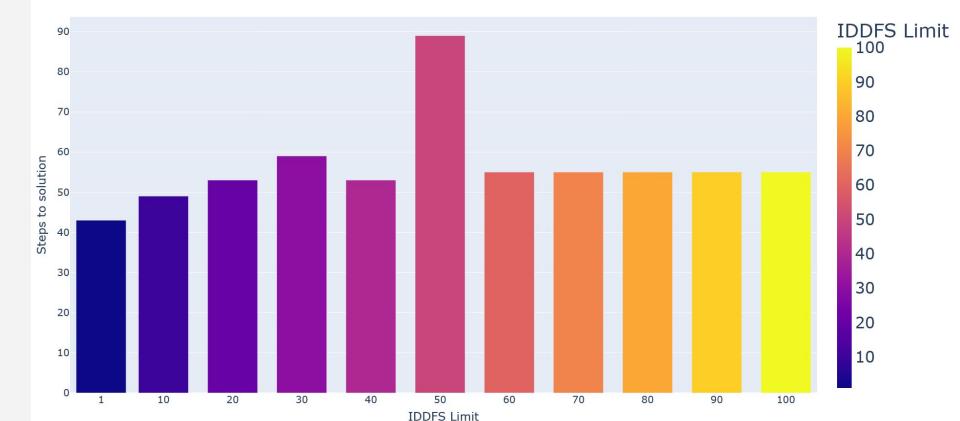


# Iddfs utilizando distintos límites

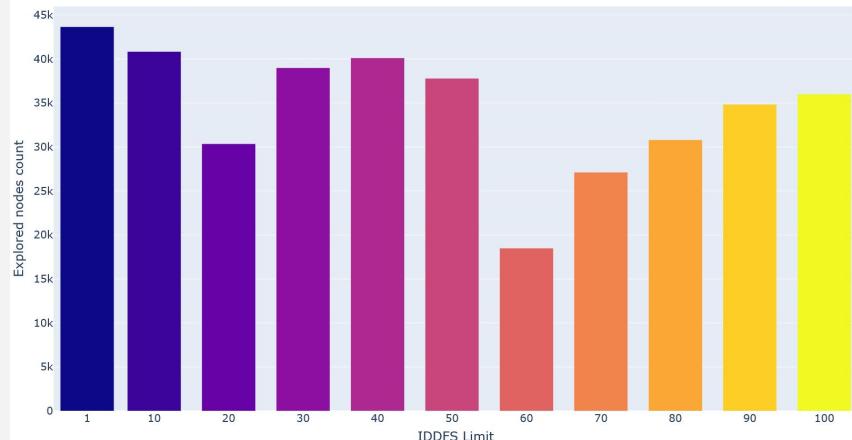
Execution Time Across IDDFS Limits - Level lv7



Optimality of Solutions Across IDDFS Limits - Level lv7



Explored Nodes Count Across IDDFS Limits - Level lv7



# Métodos de búsqueda informados

En primer lugar veamos que heurísticas se analizan

- Boxes in goals
- Euclidean
- Euclidean non admissible
- Manhattan
- Manhattan non admissible

# Heurística Boxes in goals

- Se calcula mediante la cantidad de cajas que hay en las banderas.

# • Heuristica Euclidean

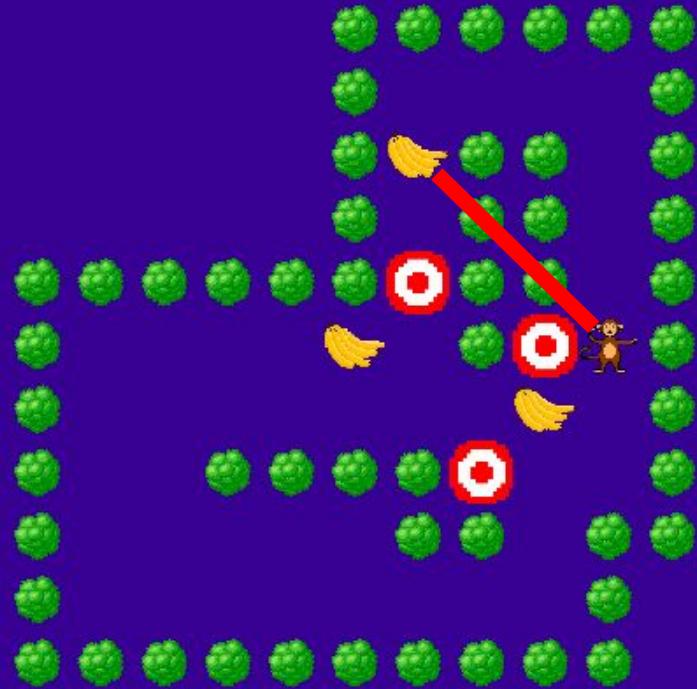
Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.



Se utiliza distancia euclidea.

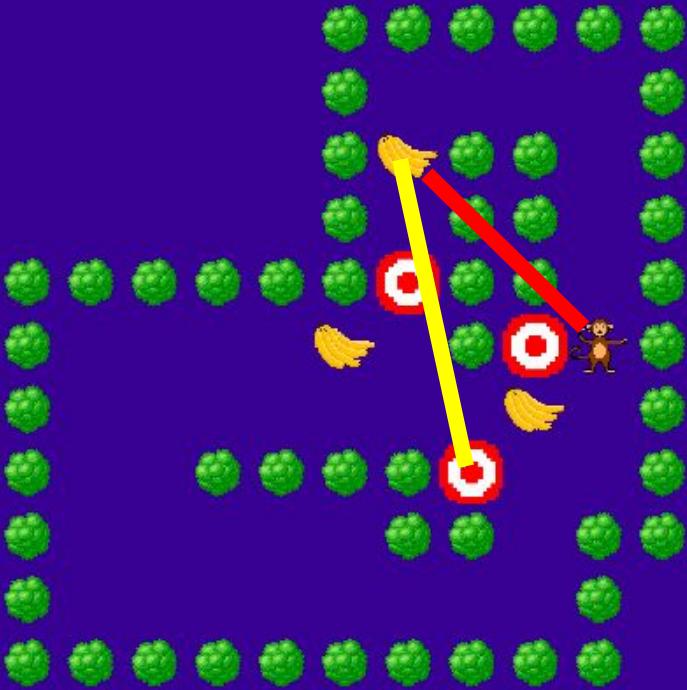


# • Heuristica Euclidean

Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.

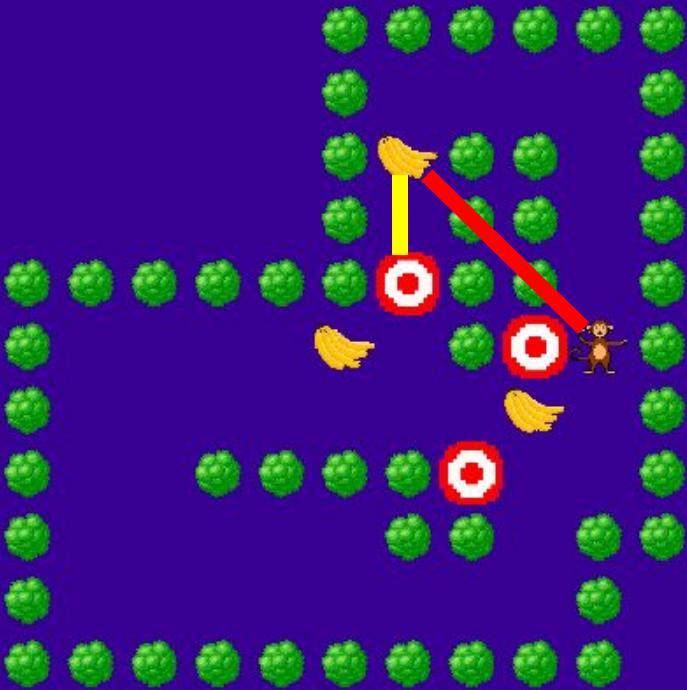


# • Heuristica Euclidean

Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.

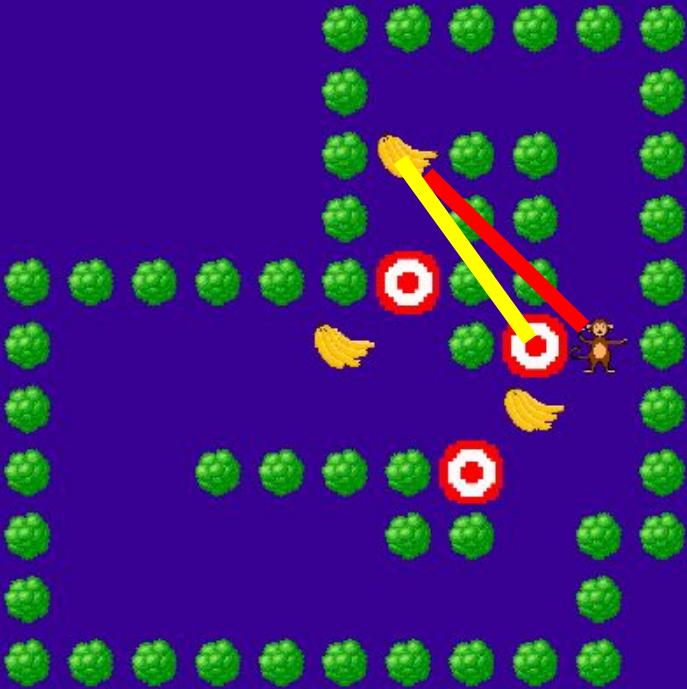


# • Heuristica Euclidean

Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.

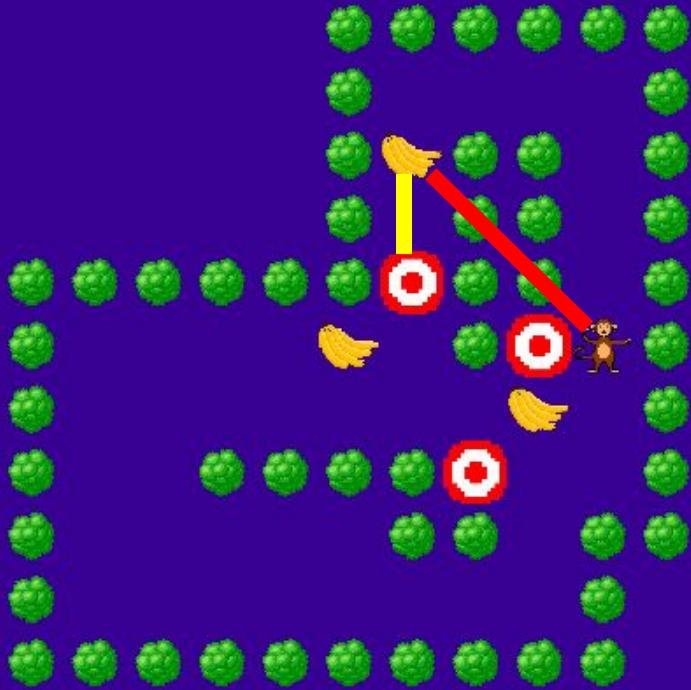


# • Heuristica Euclidean

Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

El costo total de la heurística será finalmente la suma de los costos de las cajas MÍNIMOS



# • Heuristica Euclidean non admissible

Equivalente a la heurística euclidean pero sin considerar la distancia del jugador a la caja.

El costo de cada caja sería:

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.



Se utiliza distancia euclidea.



# • Heuristica Manhattan

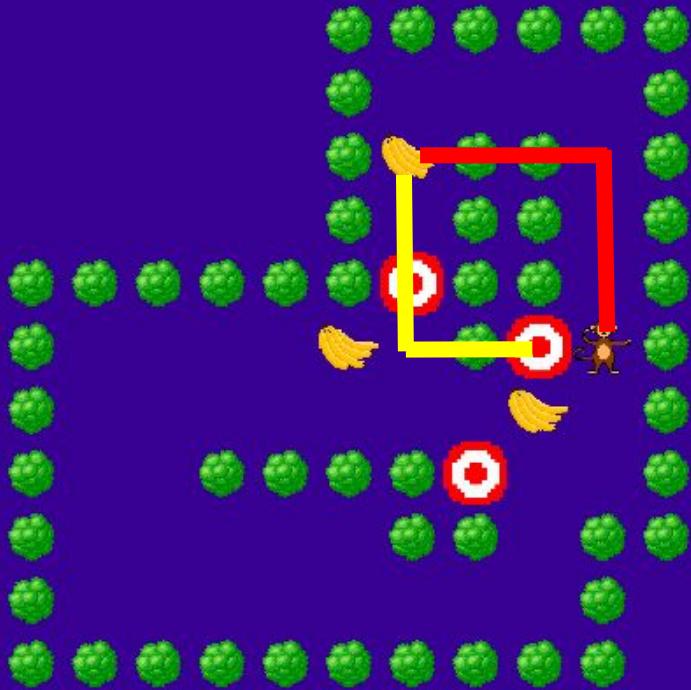
Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.



Se utiliza distancia euclídea.

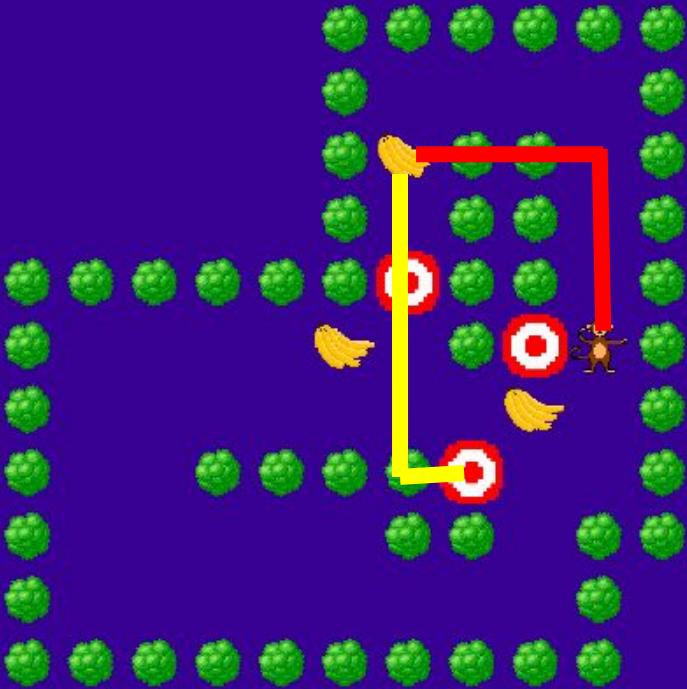


# • Heuristica Manhattan

Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.

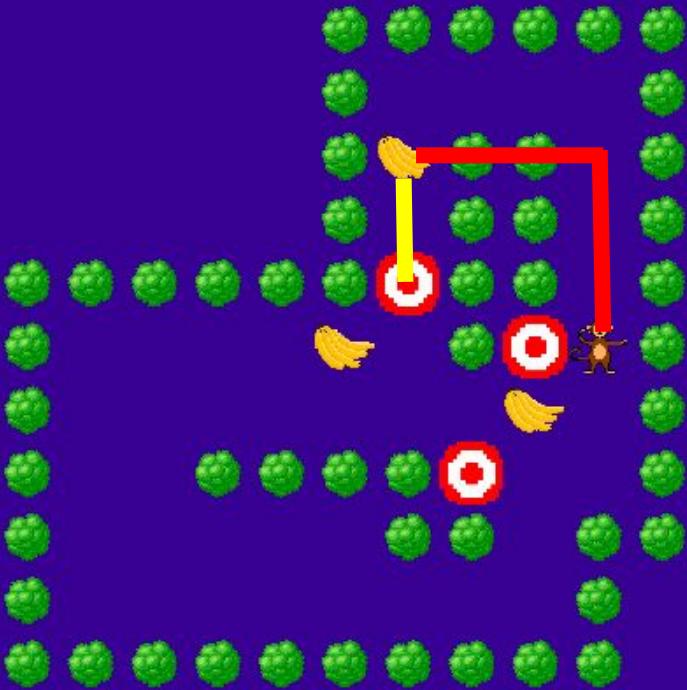


# • Heuristica Manhattan

Para cada caja se calcula el costo.

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{player}) + \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.



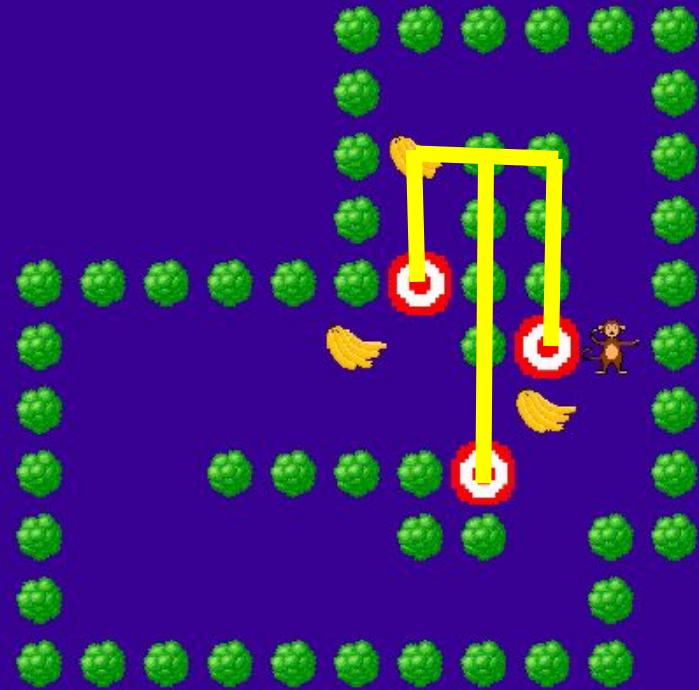
# • Heuristica Manhattan non admissible

Equivalente a la heurística manhattan pero sin considerar la distancia del jugador a la caja.

El costo de cada caja sería:

$$\text{costo}(\text{caja}) = \text{distancia}(\text{caja}, \text{goal})$$

Elegimos la combinación caja goal con costo mínimo.



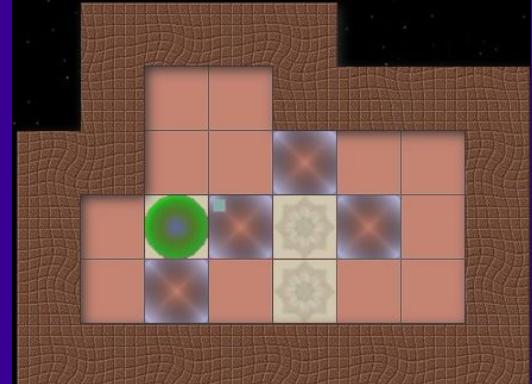
# • Comparación admisibles vs no admisibles

Manhattan => Solución en 43 pasos

Manhattan\_non\_admissible => Solución en 45 pasos

Euclidean => Solución en 43 pasos

Euclidean\_non\_admissible => Solución 45 pasos



Nivel 7

Pasos óptimos: 43

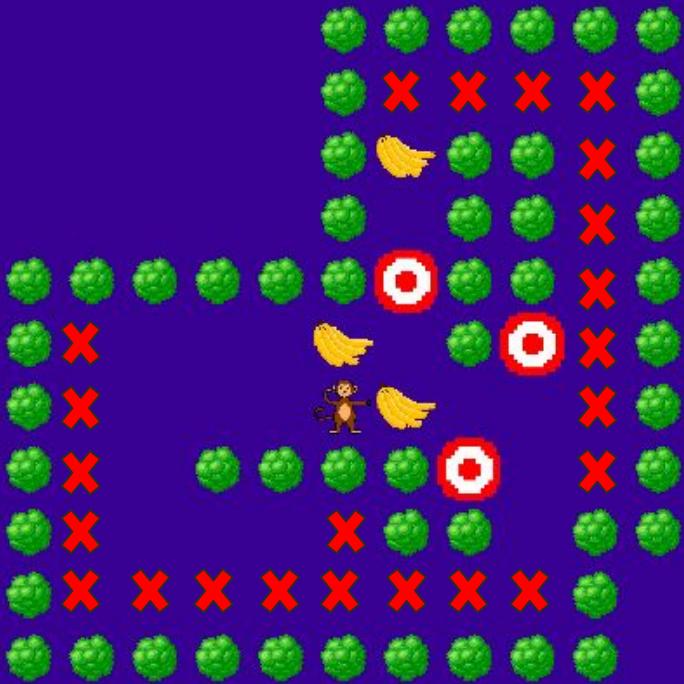
Algoritmo: A\*

# Heurística Box stuck



Primero en todas las esquinas donde no hay goals se marca como posición inaccesible.

¿Por qué?



# Heurística Box stuck

Primero en todas las esquinas donde no hay goals se marca como posición inaccesible para las cajas.

¿Por qué?

Luego se extienden estas posiciones inaccesibles.

Cuando se mueve el personaje nos fijamos si alguna caja esta en las posiciones inaccesibles.



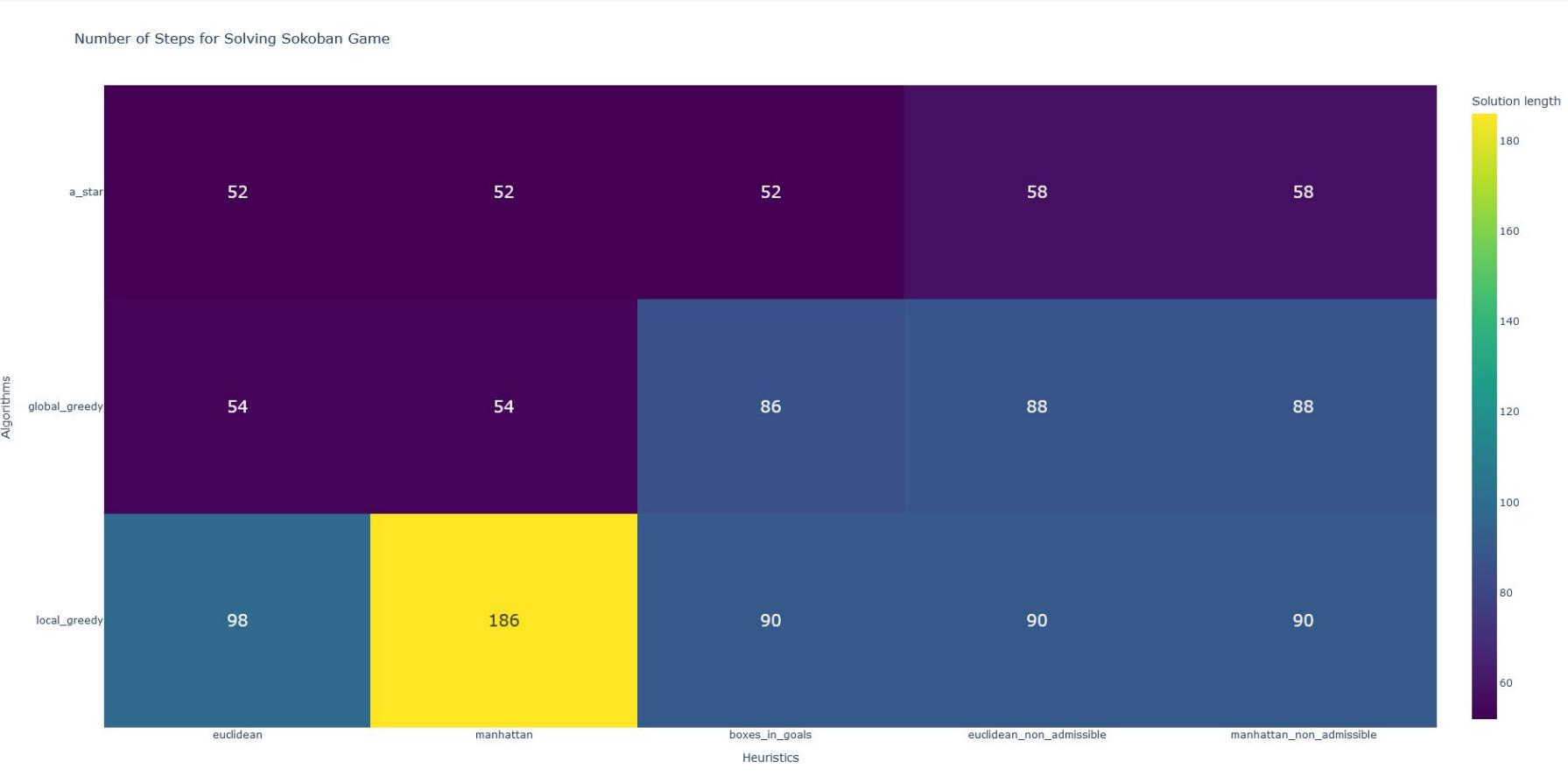
# Análisis

Nuevamente, se proponen preguntas disparadoras del análisis:

- ¿Qué tan óptima es la solución de cada algoritmo cuando aplicamos distintas heurísticas?
- ¿Cuán costoso es cada algoritmo al aplicar distintas heurísticas?
- Para tener una muestra significativa, se realizaron todos los análisis con 10 iteraciones.

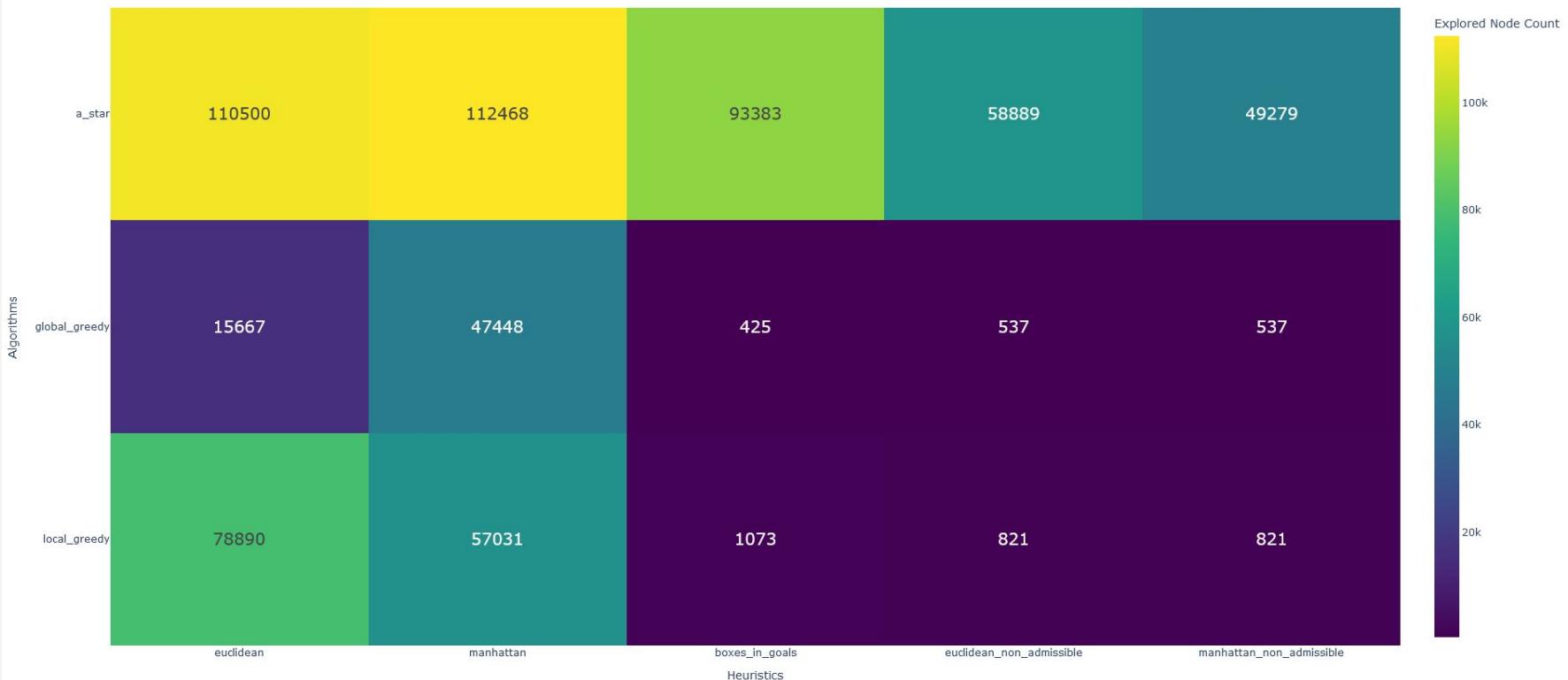


# Optimización en informados - lv3

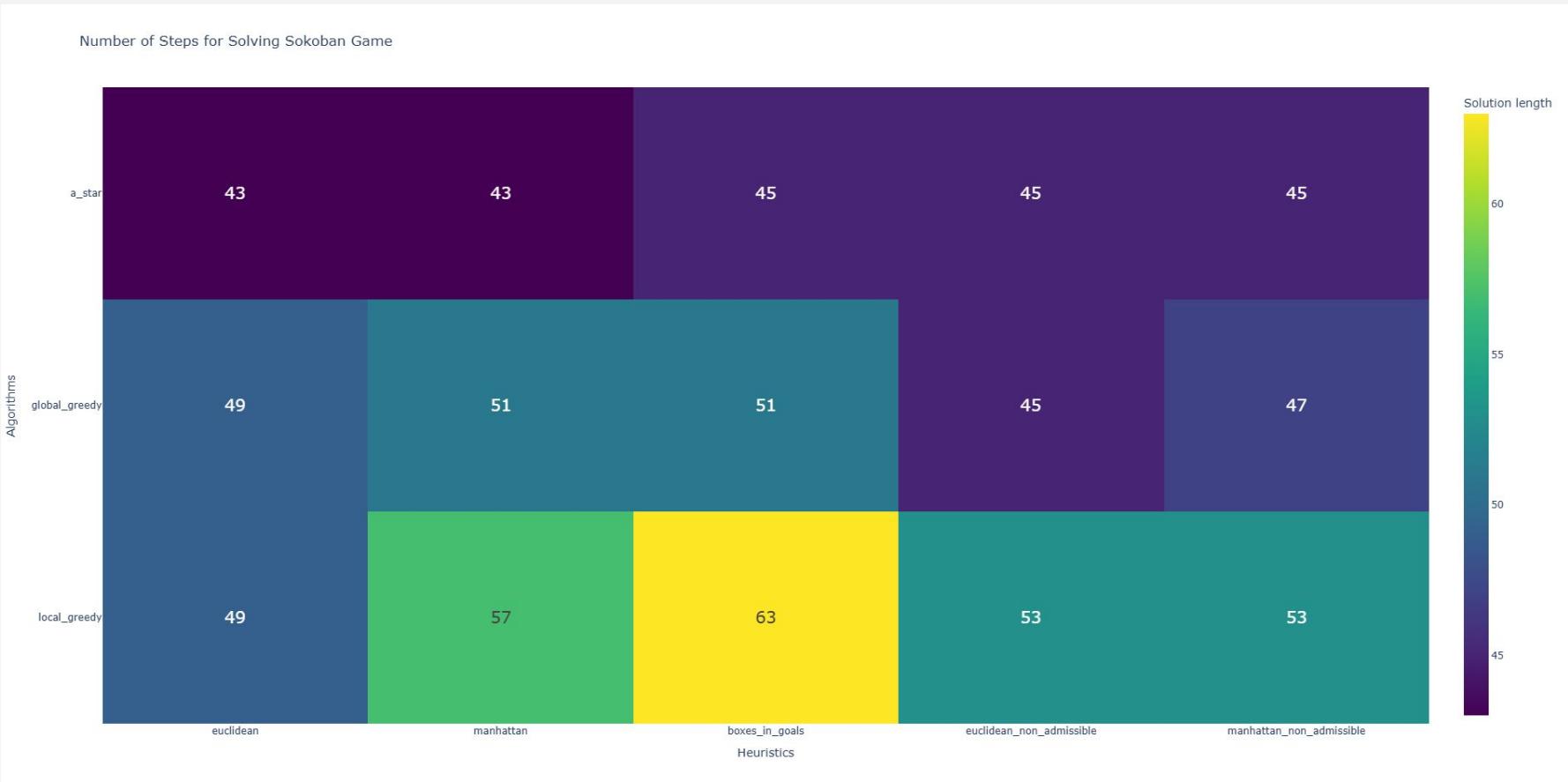


# Nodos explorados en informados - lv3

Explored Node Count for each Algorithm and Heuristic combination



# Optimización en informados - lv7



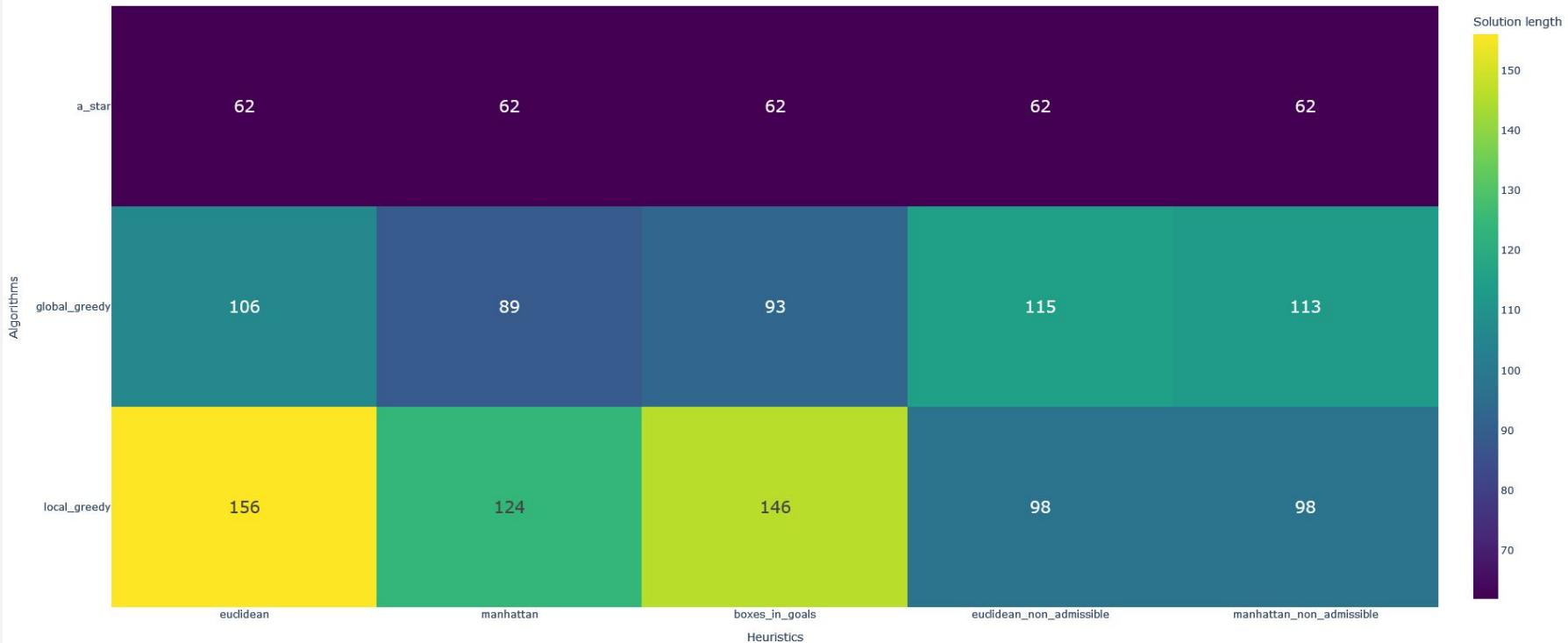
# • Nodos explorados en informados - lv7





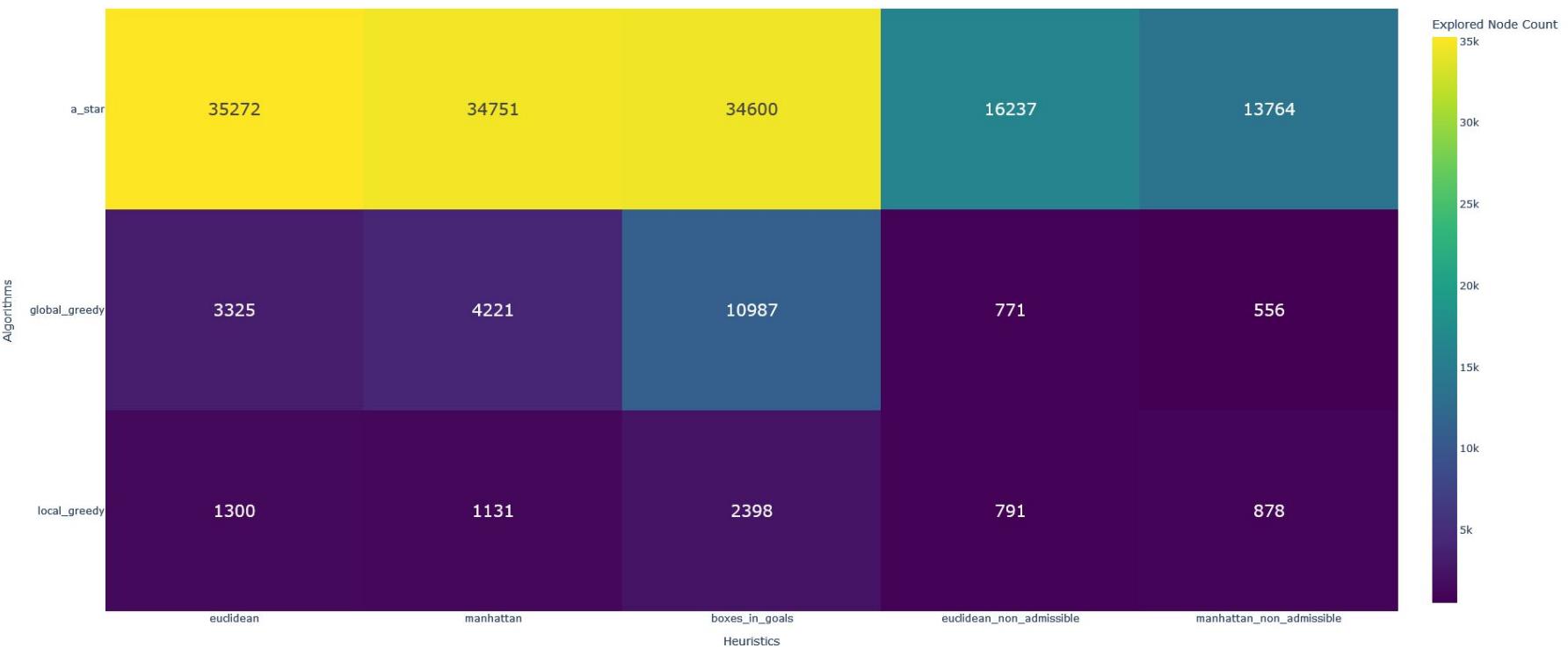
# Optimización en informados - lv01

Number of Steps for Solving Sokoban Game

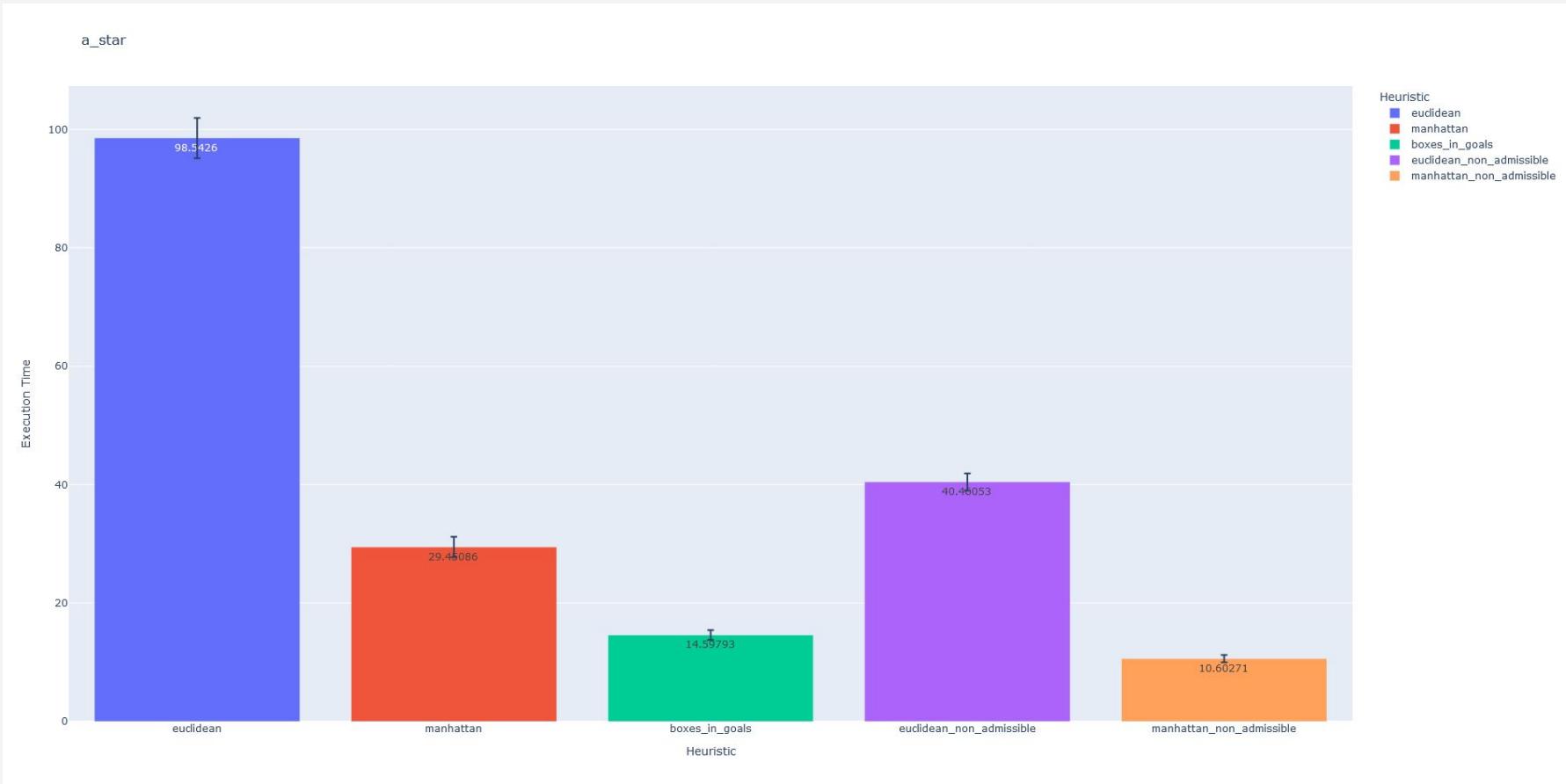


# • Nodos explorados en informados - lv01

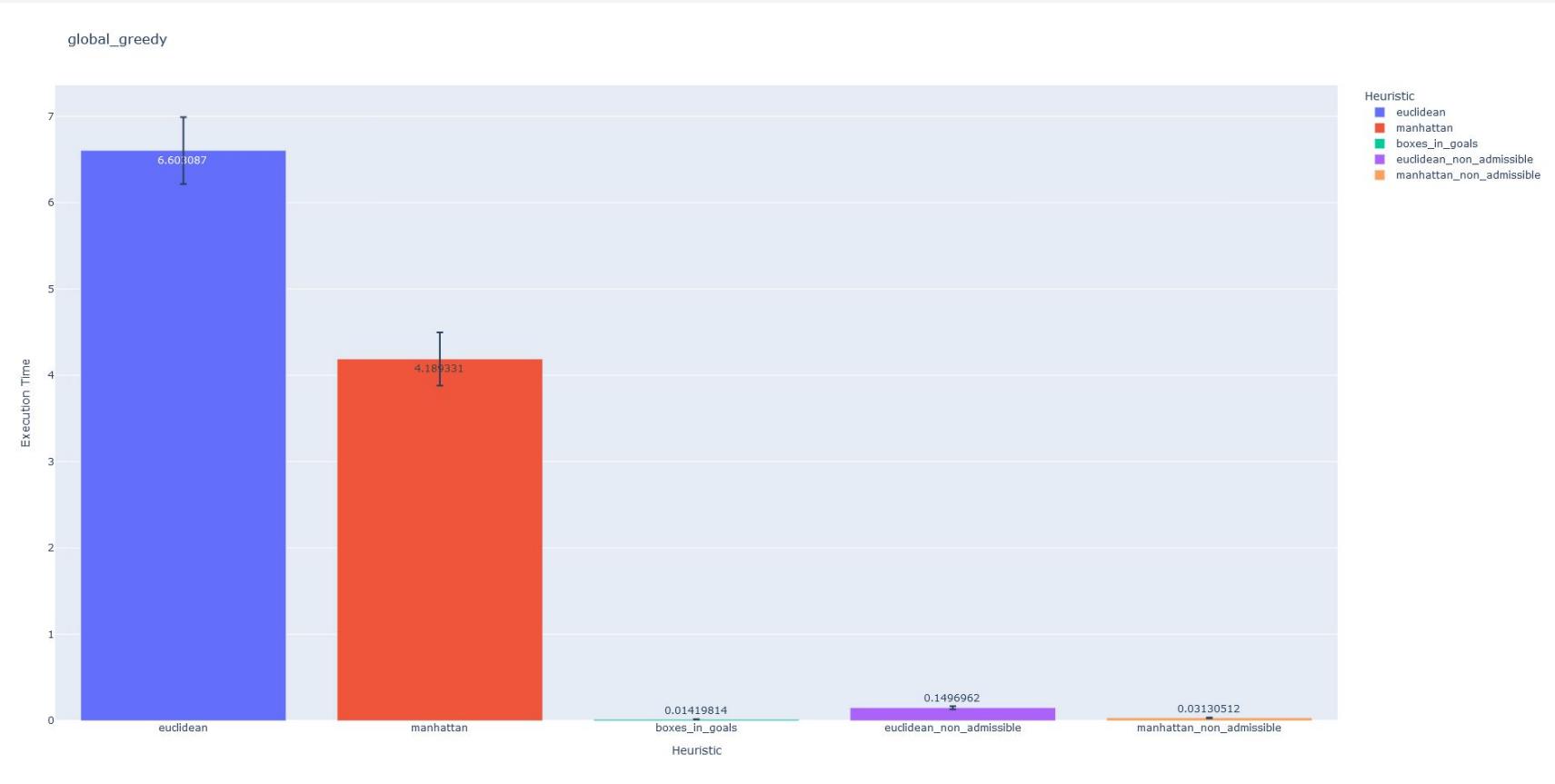
Explored Node Count for each Algorithm and Heuristic combination



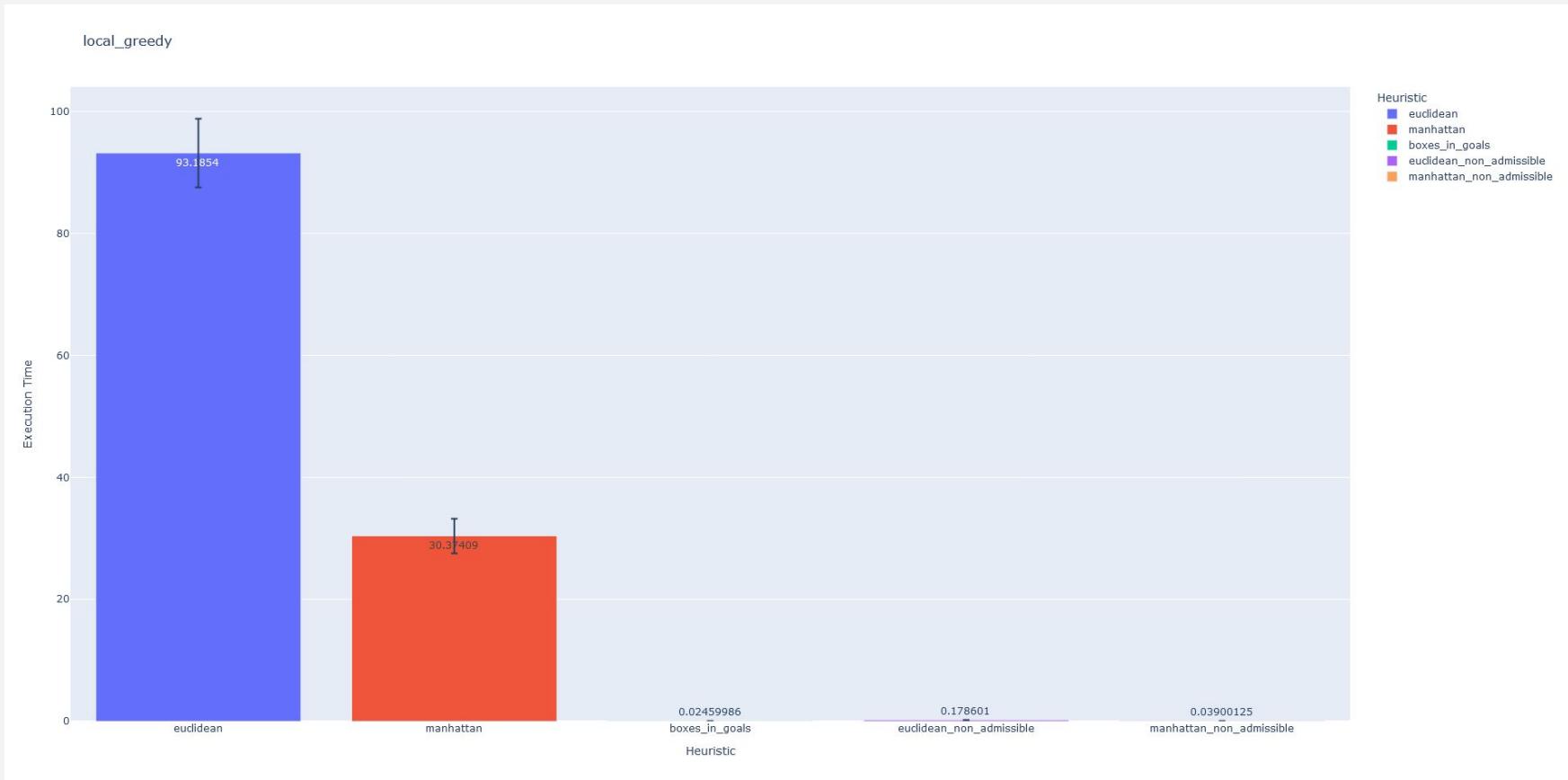
# •Tiempos de ejecución en informados - lv3



# Tiempos de ejecución en informados - lv3

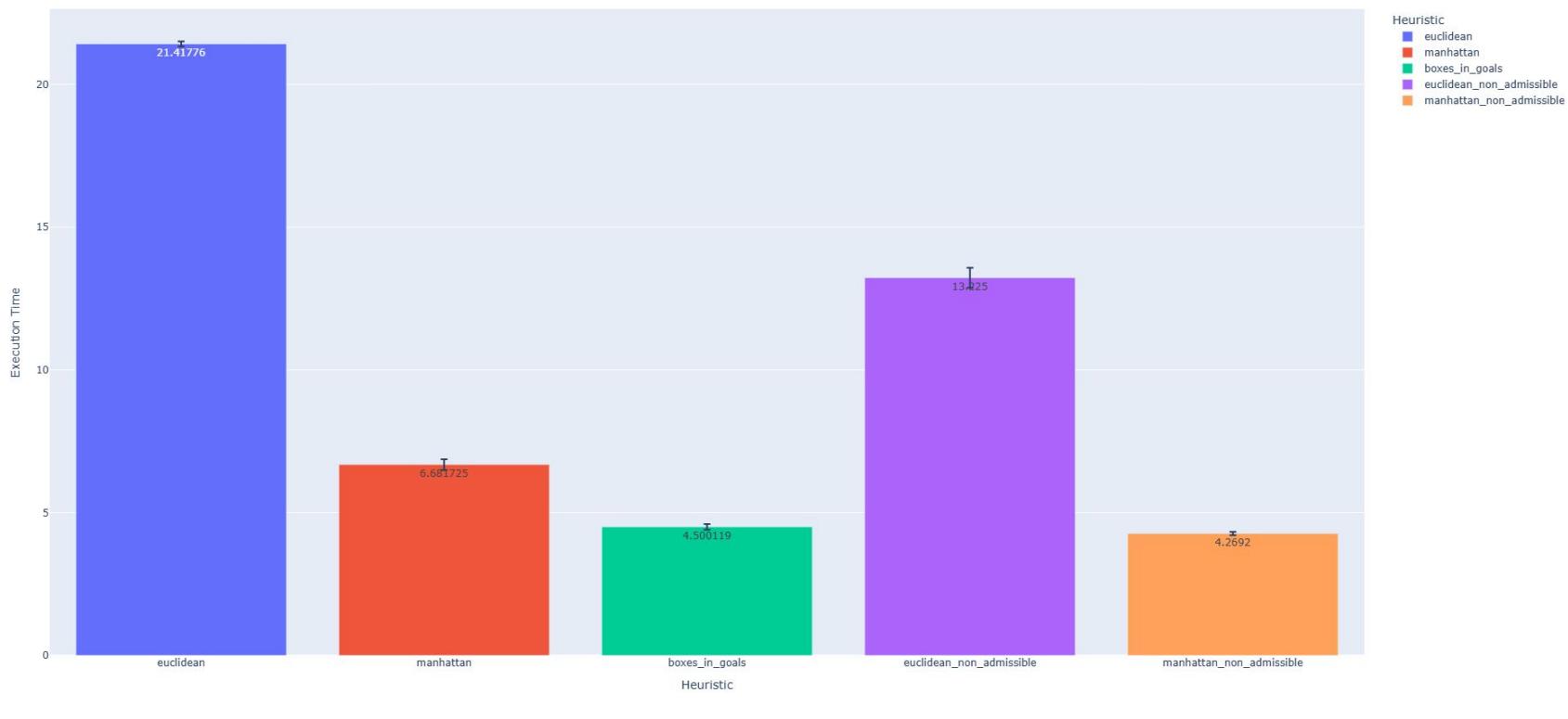


# Tiempos de ejecución en informados - lv3

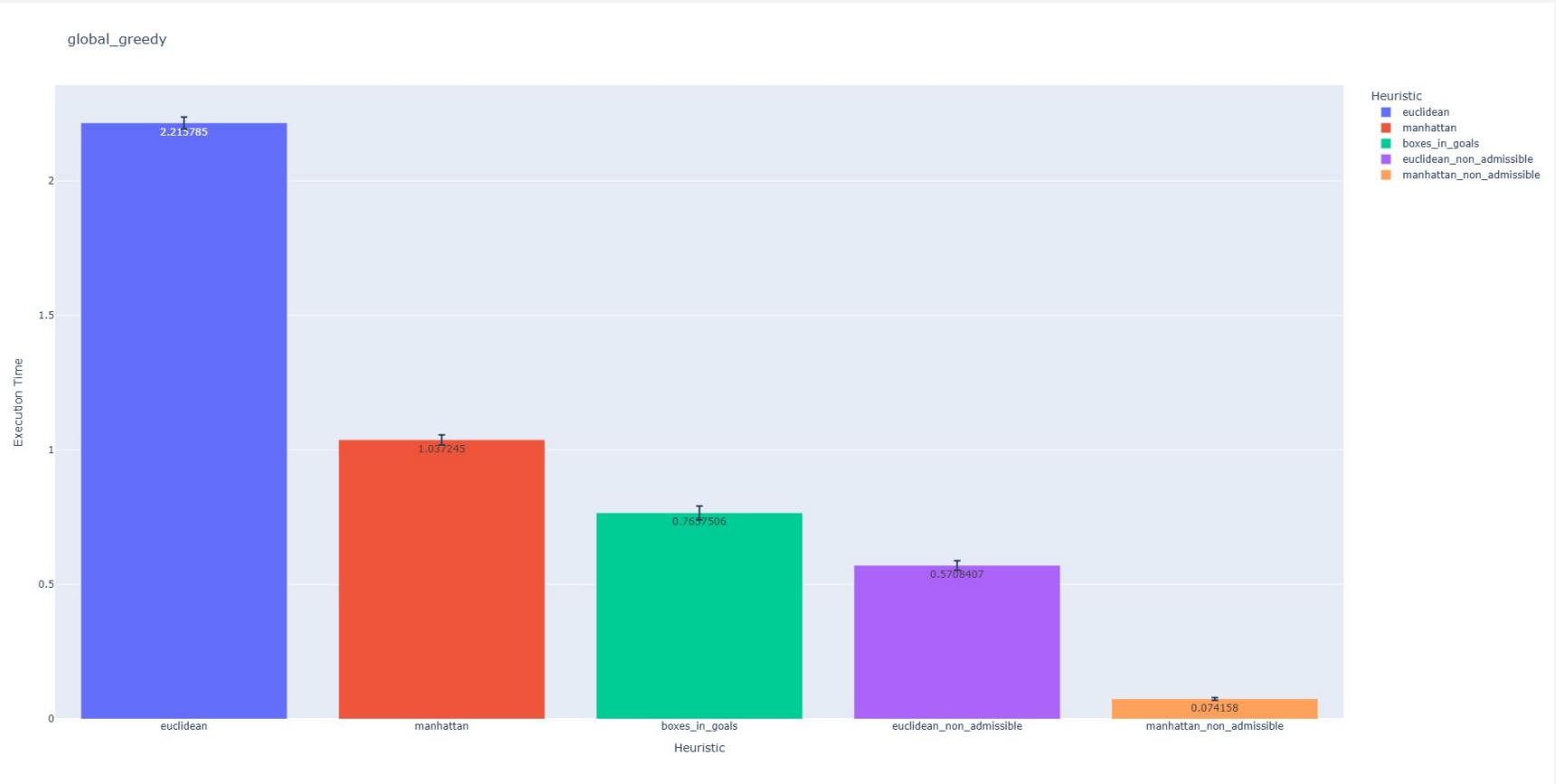


# Tiempos de ejecución en informados - lv7

a\_star

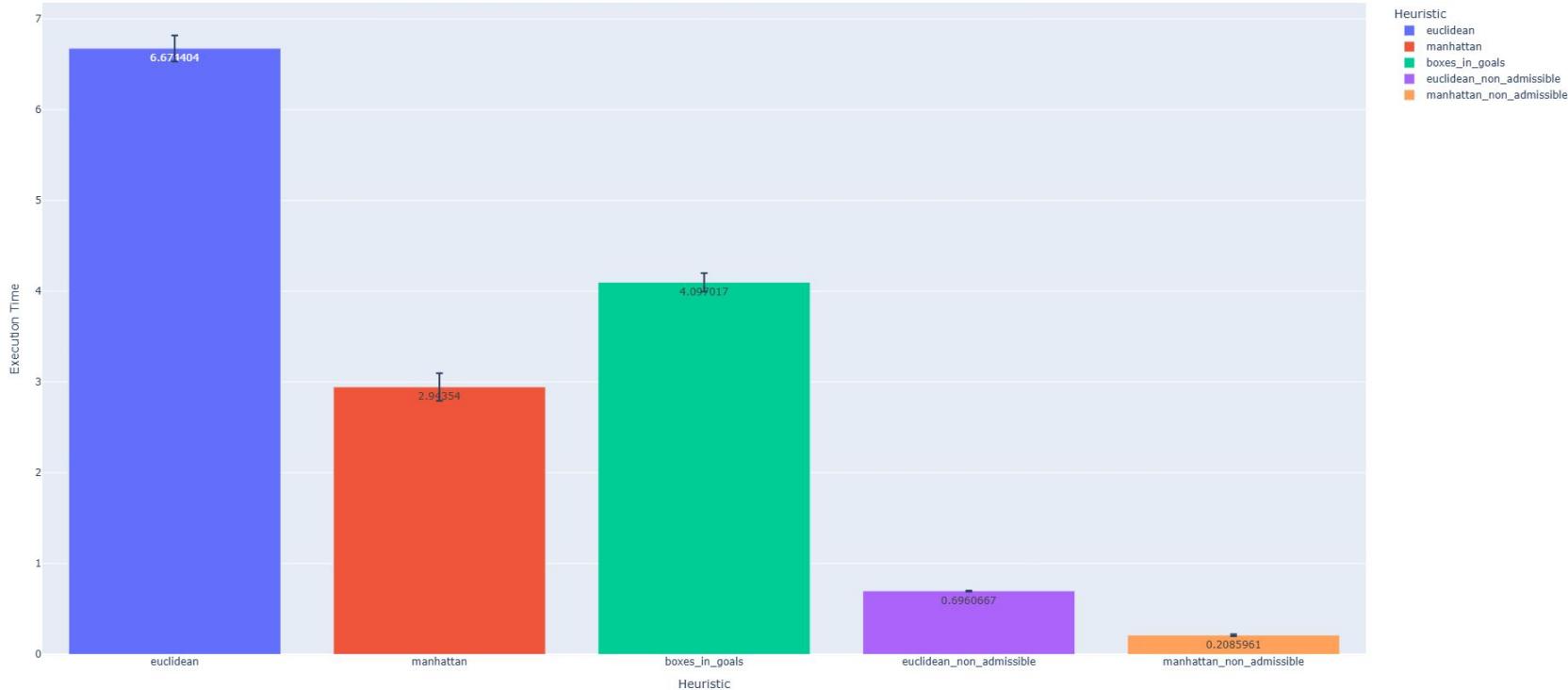


# Tiempos de ejecución en informados - lv7

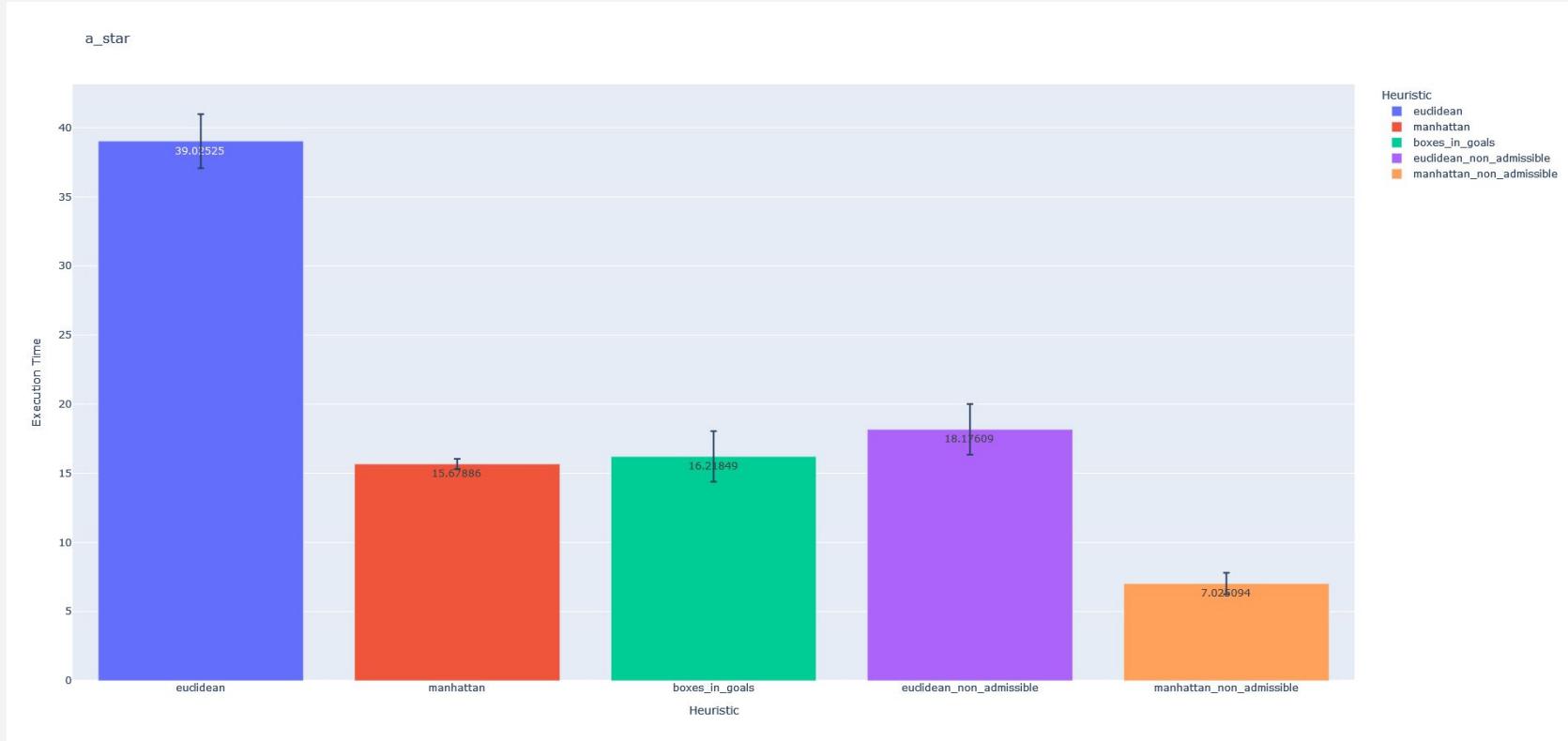


# Tiempos de ejecución en informados - lv7

local\_greedy

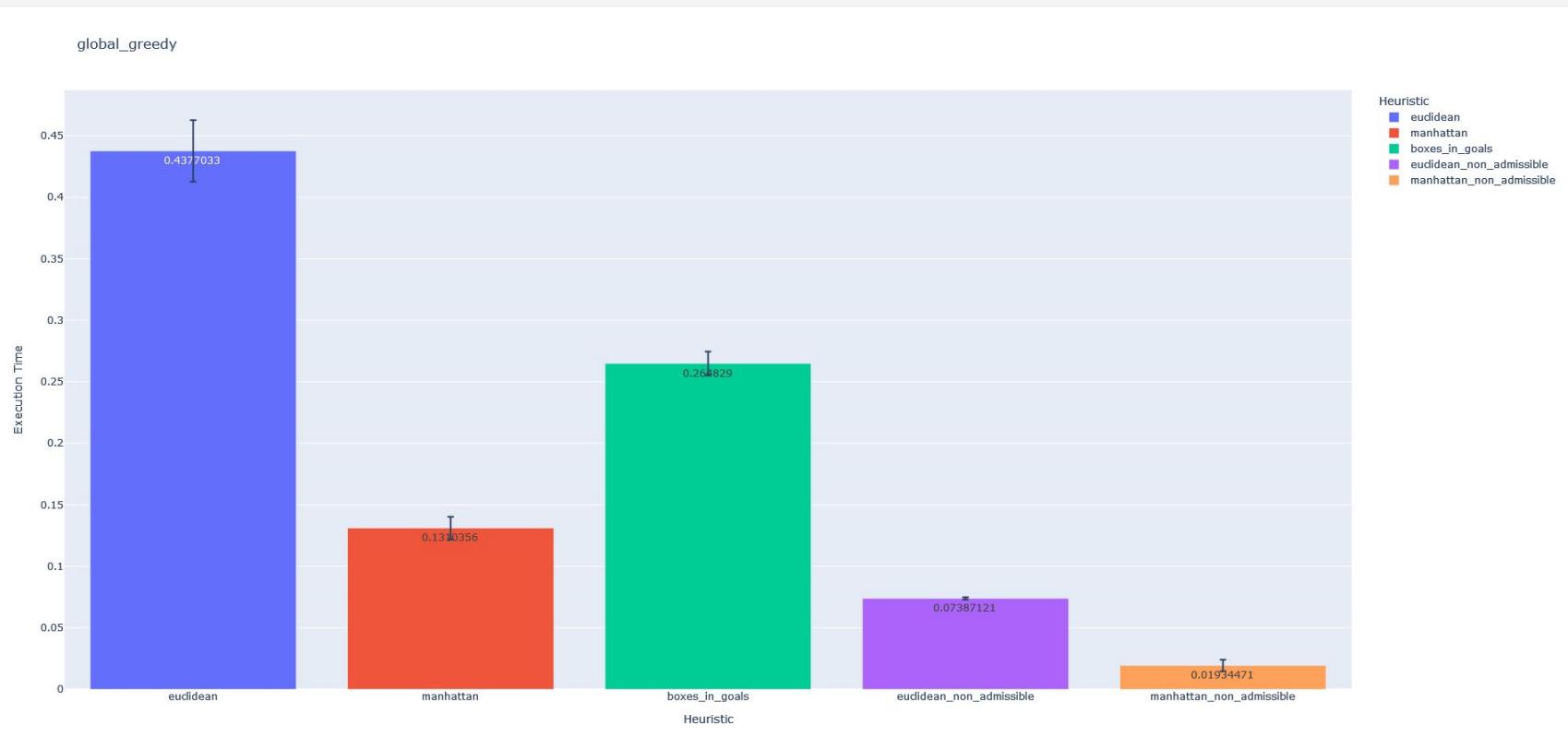


# Tiempos de ejecución en informados - lv01

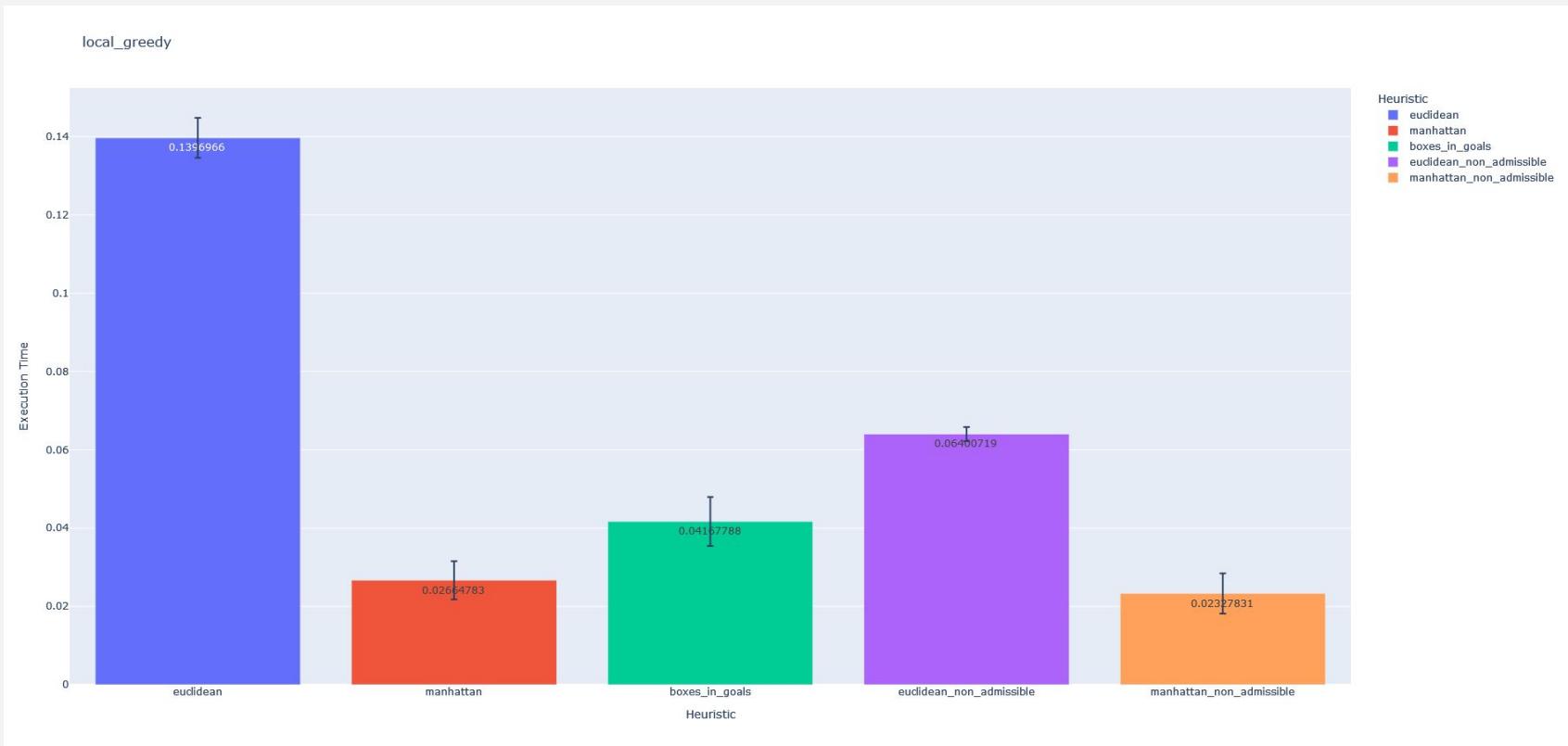


# Tiempos de ejecución en informados - lv01

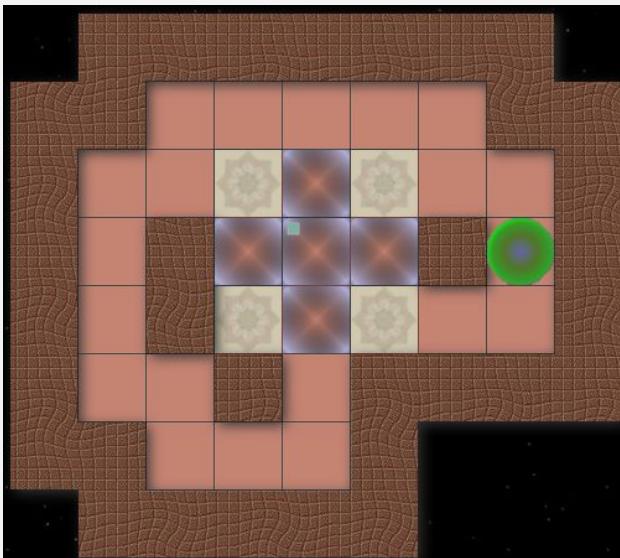
global\_greedy



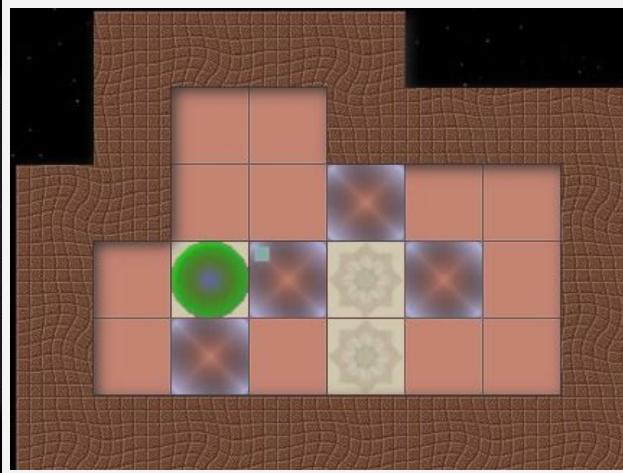
# Tiempos de ejecución en informados - lv01



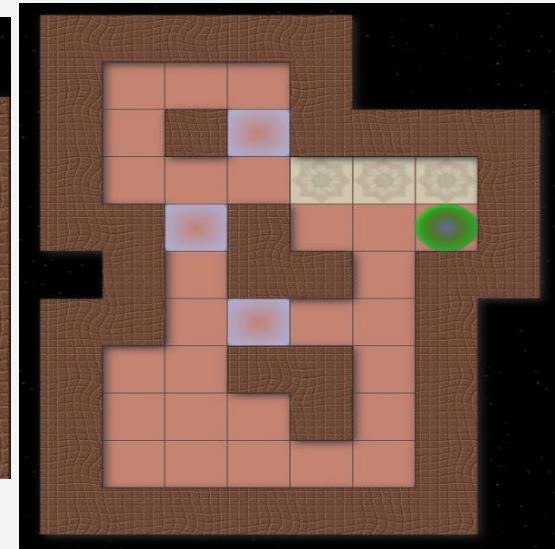
# Comparación con Box Stuck



Nivel 3



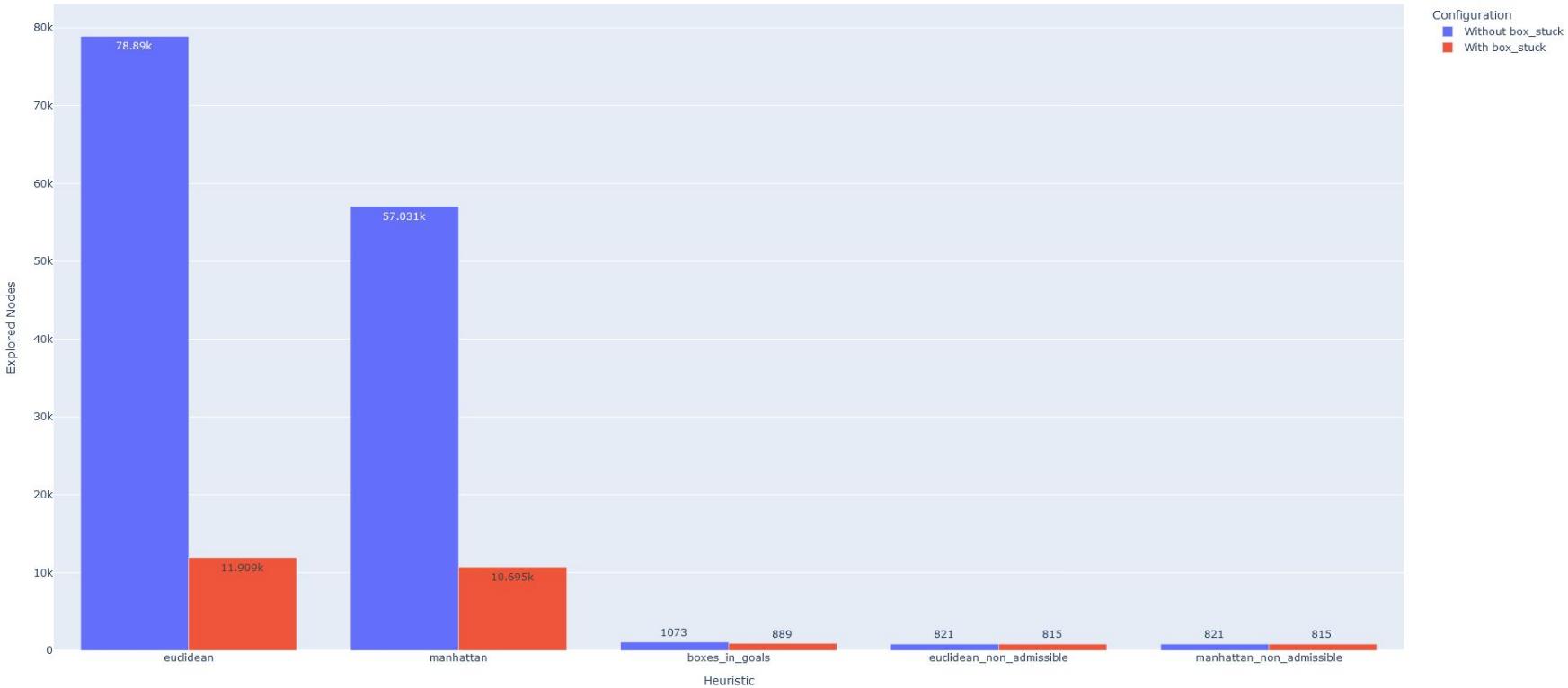
Nivel 7



Nivel 01

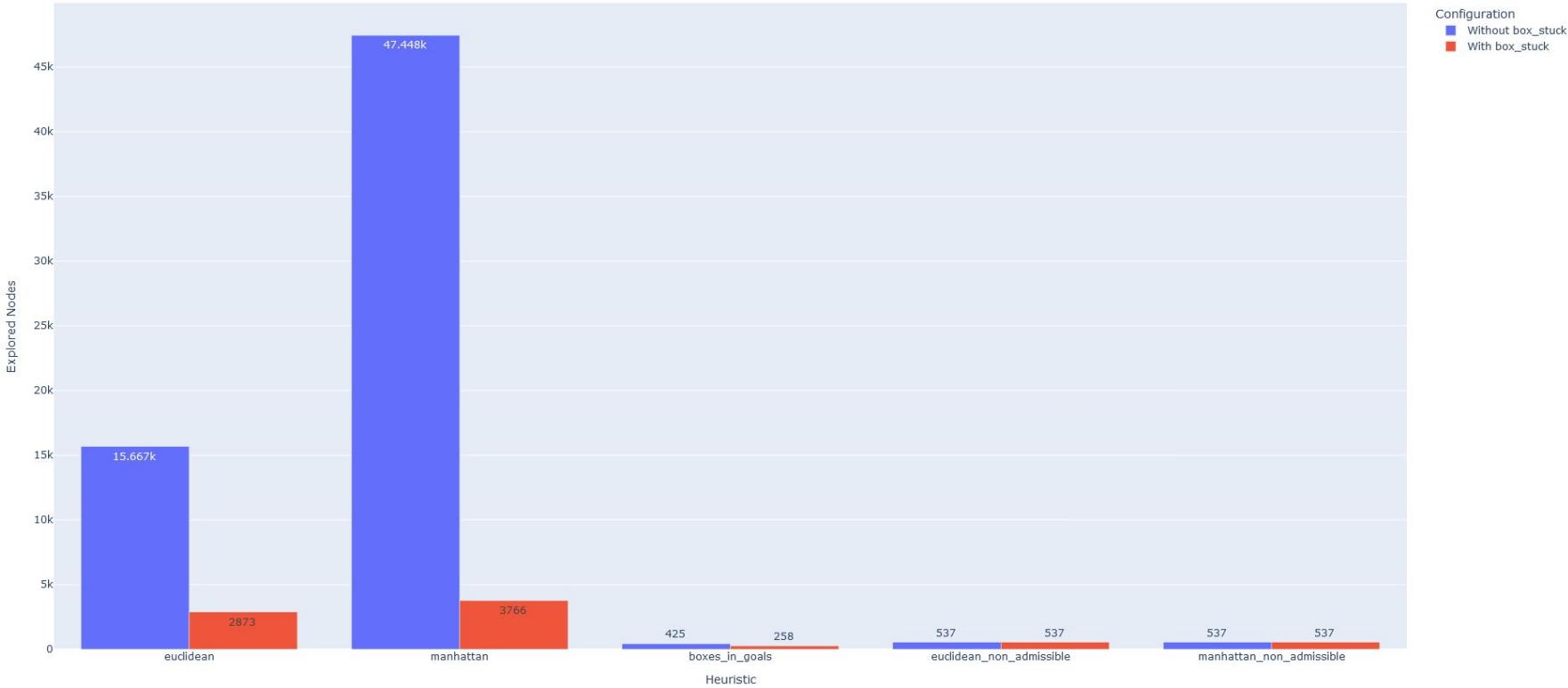
# Nivel 3 con Local Greedy

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in local\_greedy - Level lv3



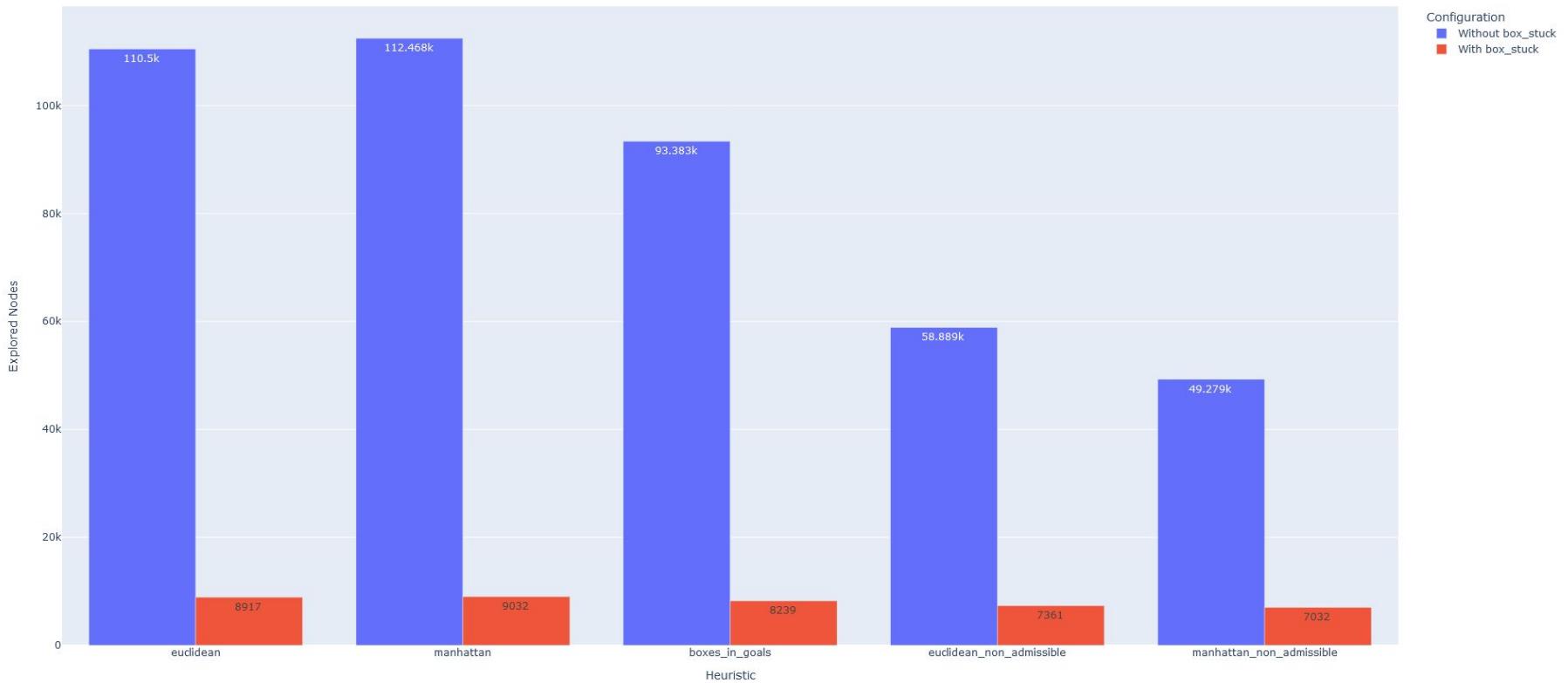
# Nivel 3 con Global Greedy

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in global\_greedy - Level lv3



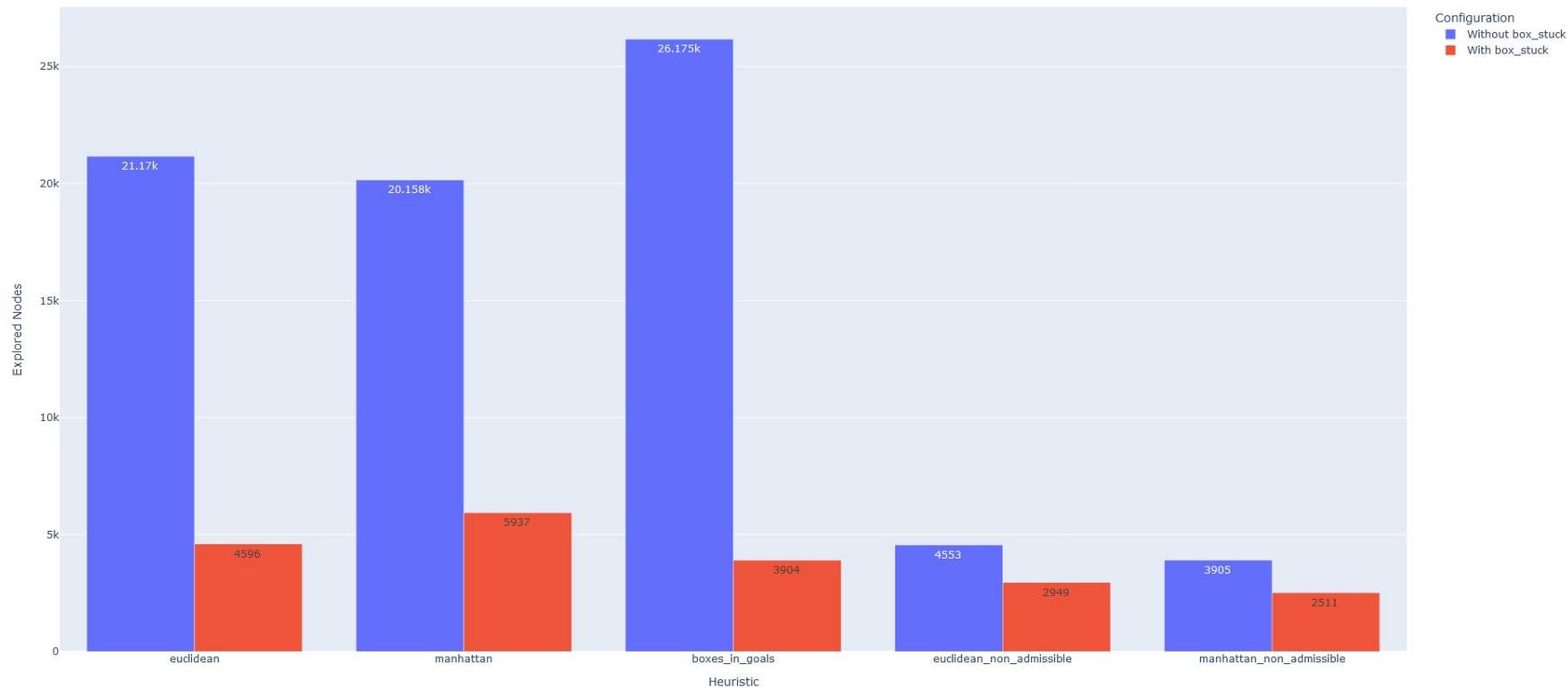
# Nivel 3 con A Star

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in a\_star - Level lv3



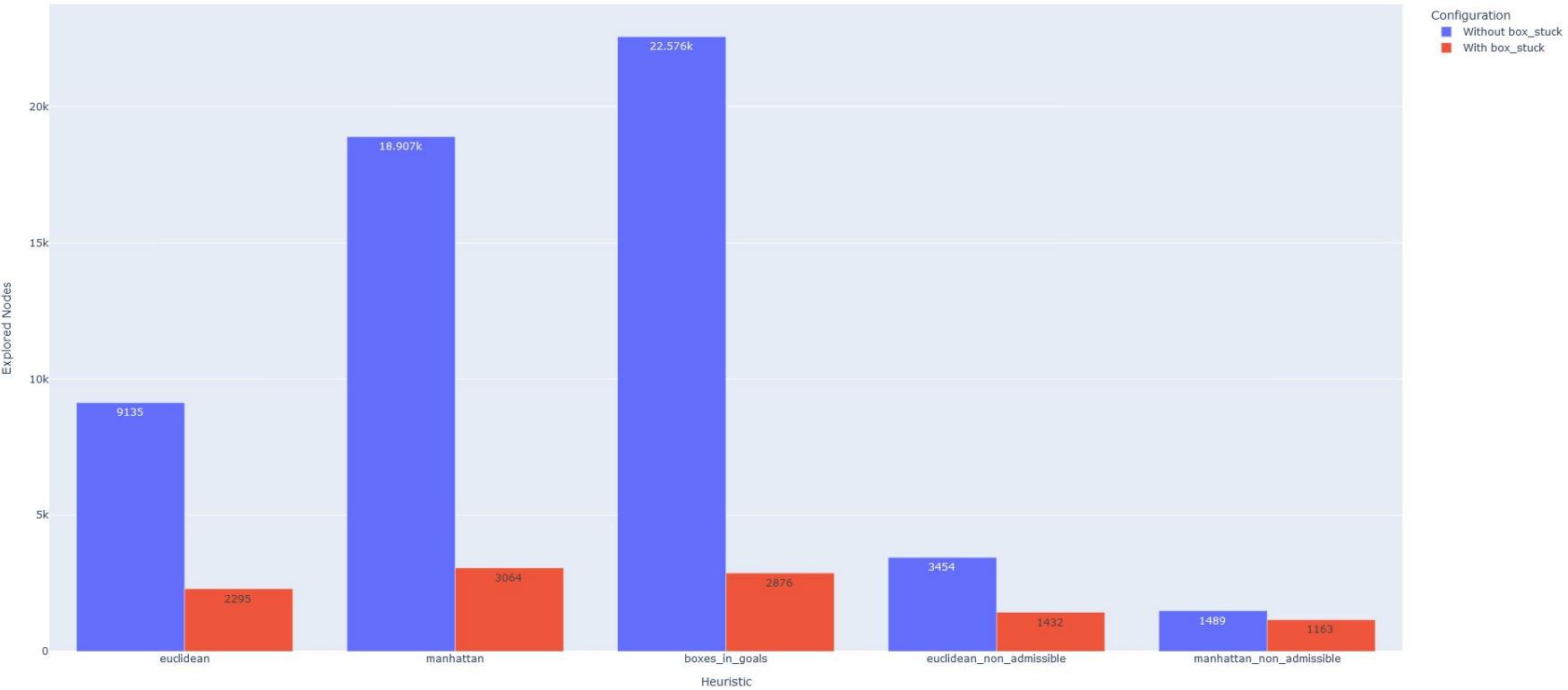
# Nivel 7 con Local Greedy

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in local\_greedy - Level lv7



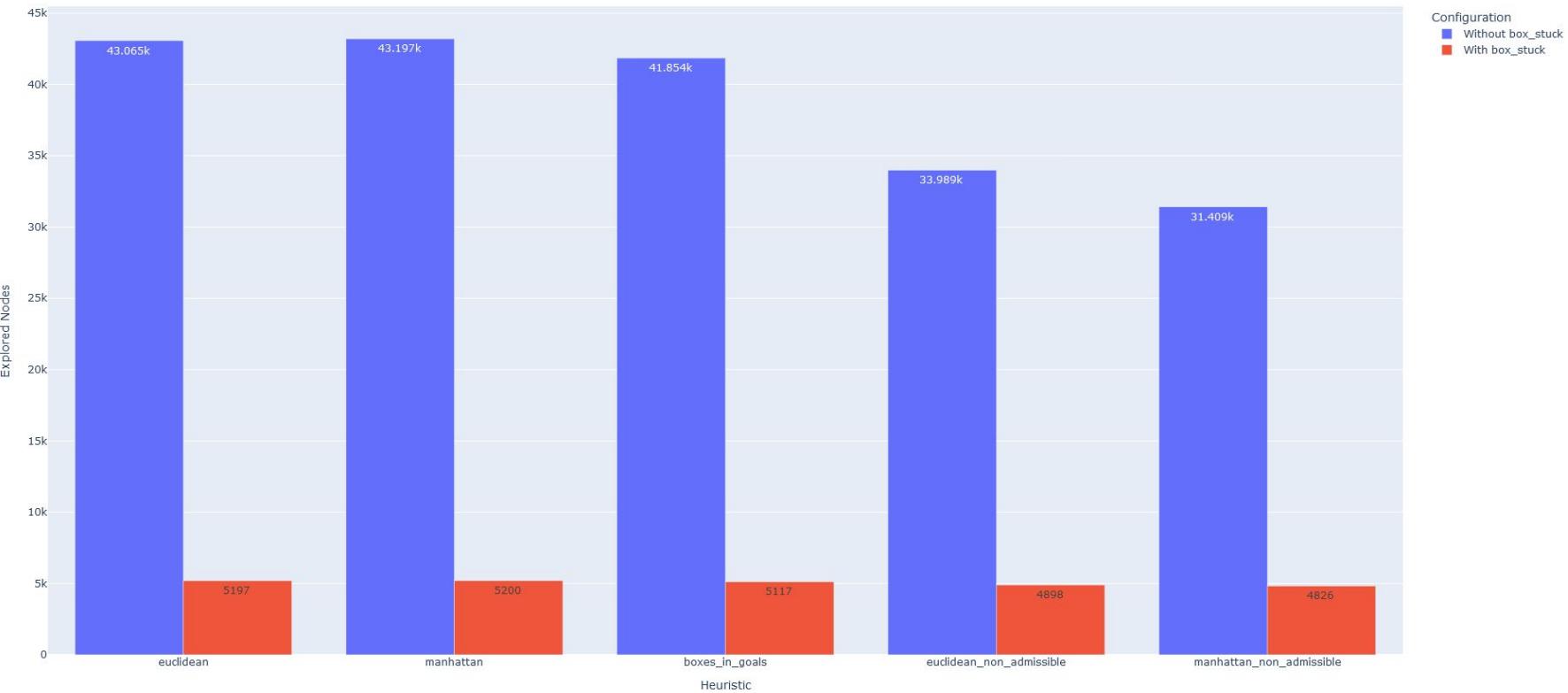
# Nivel 7 con Global Greedy

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in global\_greedy - Level lv7



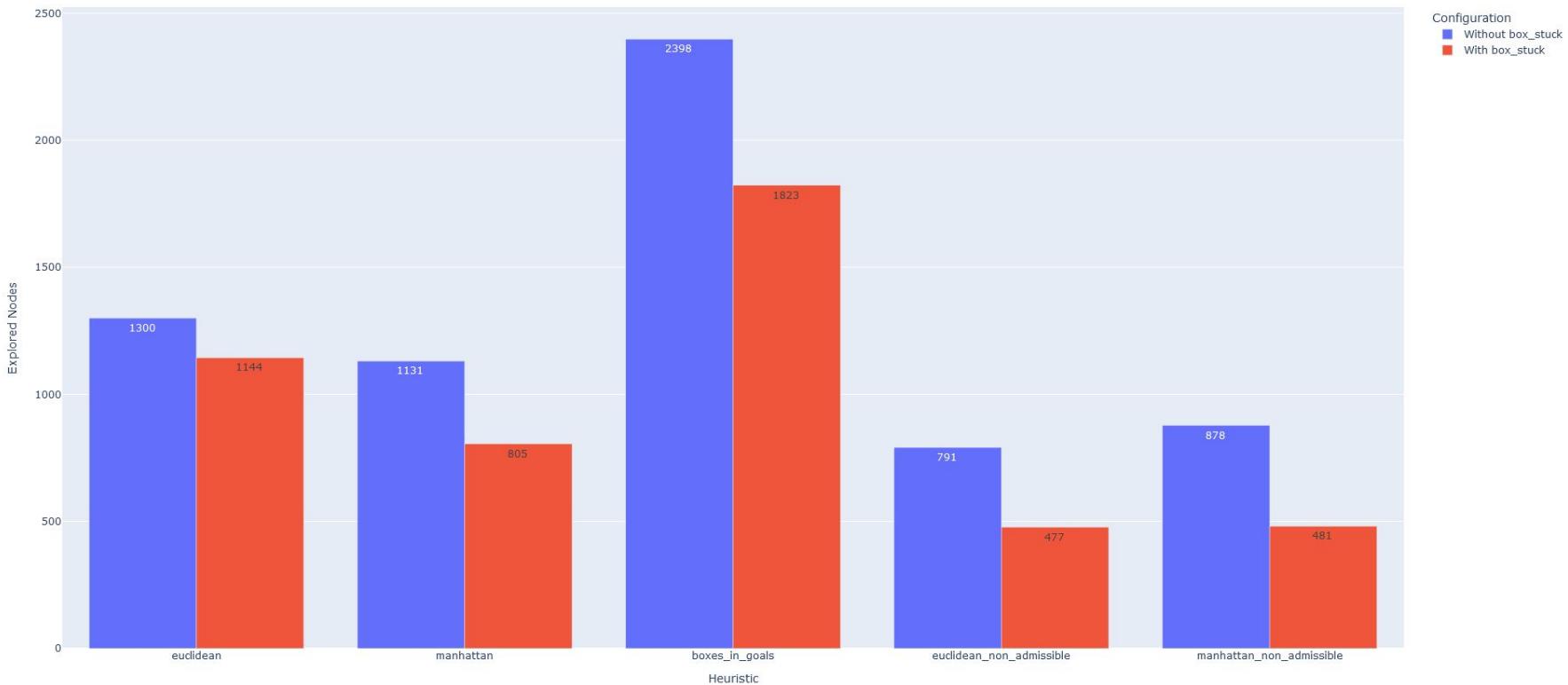
# Nivel 7 con A Star

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in a\_star - Level lv7



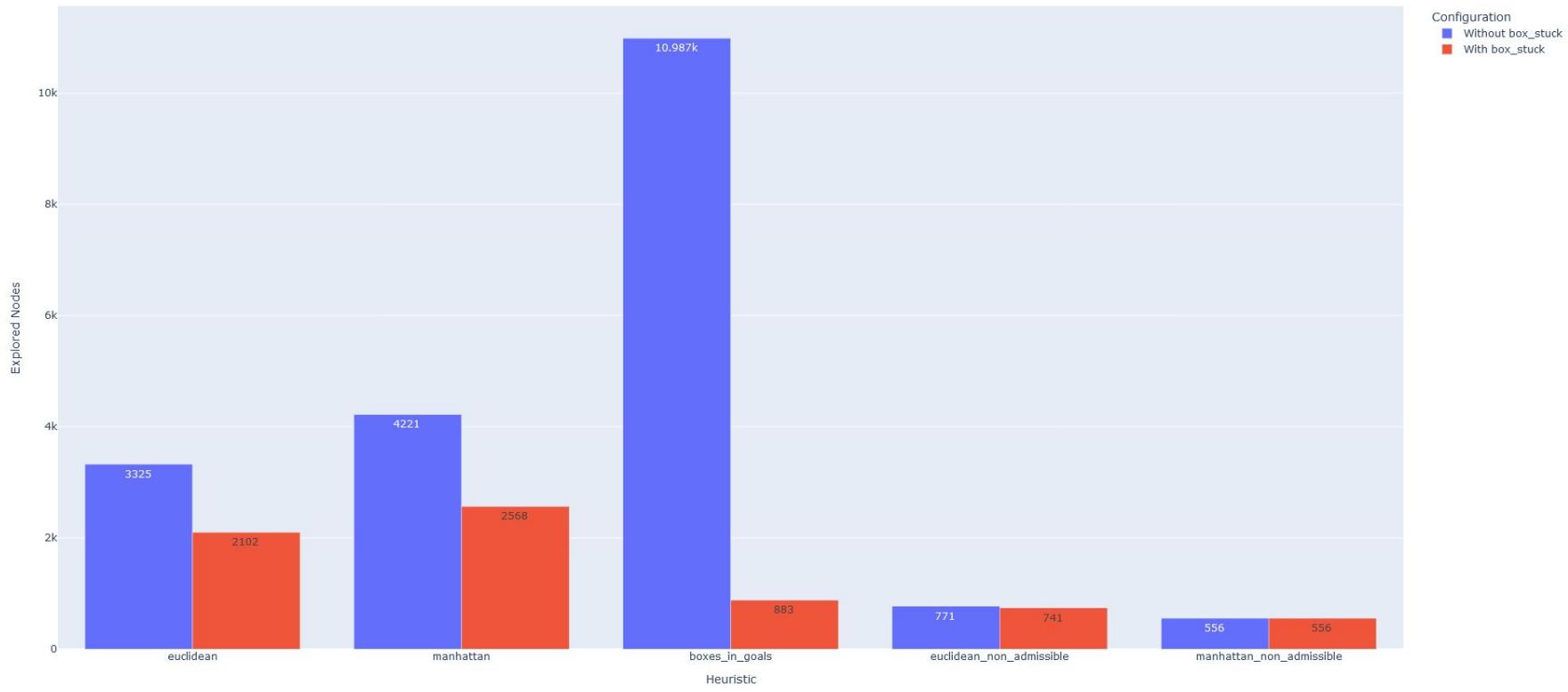
# Nivel 01 con Local Greedy

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in local\_greedy - Level lv01



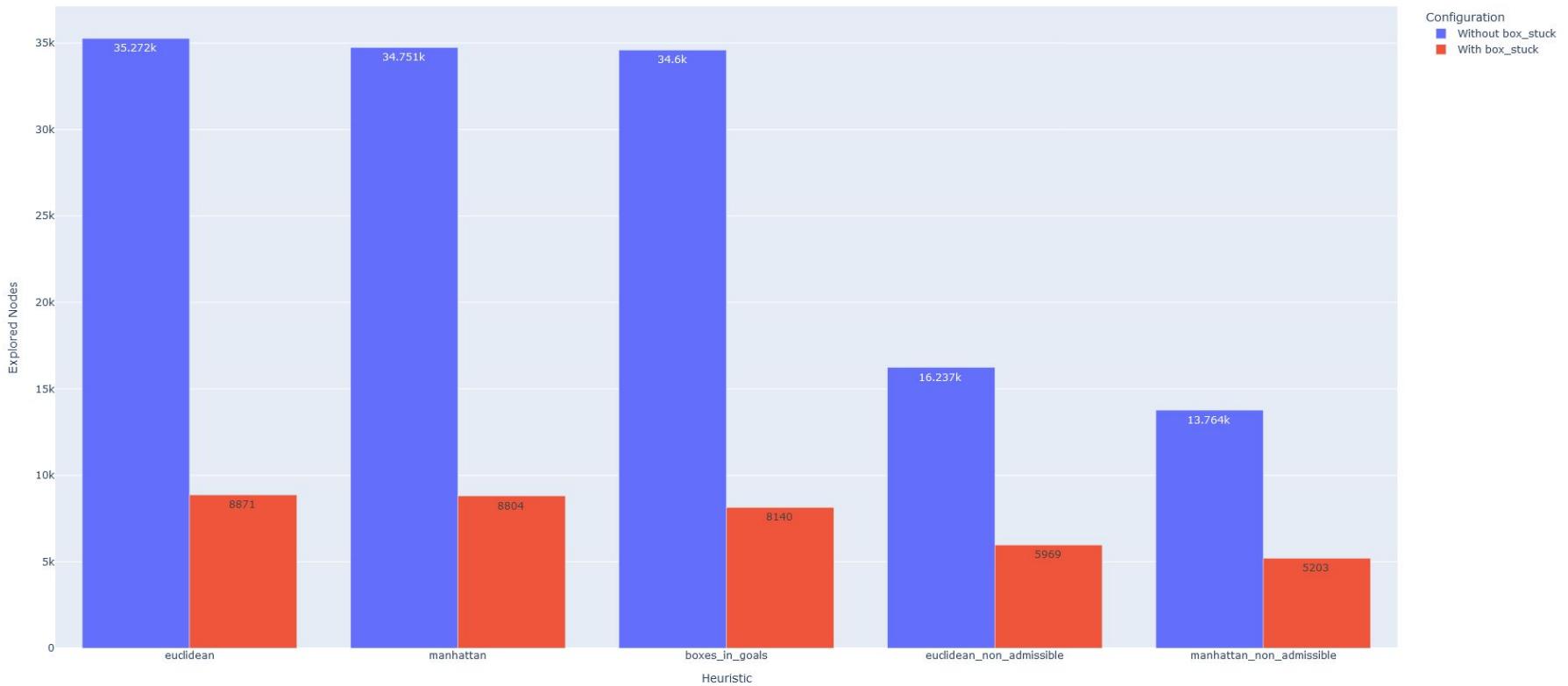
# Nivel 01 con Global Greedy

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in global\_greedy - Level lv01



# Nivel 01 con A Star

Explored Nodes by Informed Algorithms per Heuristic using box\_stuck in a\_star - Level lv01



# Conclusiones

- La elección de heurística + algoritmos ¿Buscamos solución rápida o perfecta?
- En el caso del Sokoban los resultados dependen mucho del tipo de nivel; cantidad de paredes, espacio libre, cajas.
- Con buenas heurísticas generalmente se tiene que: algoritmos informados > desinformados

**¡ Muchas  
gracias !**