

TP 1 - S.I.A - Grupo 1

Integrantes:

Burgos, Jose (61525)

Matilla, Juan Ignacio (60459)

Curti, Pedro (61616)

Panighini, Franco (61258)

8 puzzle

Nuestro estado

Estructura de estado:

- Una **matriz de 3x3** donde cada elemento de la matriz representaría una posición en el tablero, y el valor almacenado en esa posición corresponde al número de la ficha. Y el cero representa la posición vacía

estado = [

[5, 7, 3],

[8, 2, 0],

[1, 6, 4]

]

Estado goal:

solución = [

[1, 2, 3],

[8, 0, 4],

[7, 6, 5]

]

Manhattan Distance

- Distancia de Manhattan: $|x_0 - x_1| + |y_0 - y_1|$
- **Admisible**, pues no sobreestima el costo para llegar al estado final, ya que solo considera movimientos correctos, evitando obstáculos potenciales.

Pieza 1:

Posición inicial: (2, 0)

Posición correcta: (0, 0)

Distancia: 2

Pieza 2:

Posición inicial: (1, 1)

Posición correcta: (0, 1)

Distancia: 1

Pieza 3:

Posición inicial: (0, 2)

Posición correcta: (0, 2)

Distancia: 0

...

Total Manhattan Distance: 11

Tablero Inicial

5	7	3
8	2	
1	6	4

Tablero Solución

1	2	3
8		4
7	6	5

Hamming Distance

- Cuenta la cantidad de piezas que están en un lugar equivocado.
- **Admissible**, pues obtenemos una solución en una menor cantidad de movimientos, ya que cada pieza necesita por lo menos un movimiento para estar en su posición correcta.

Pieza 1:

Posición errónea: (2, 0)

Posición correcta: (0, 0)

+1

Pieza 2:

Posición errónea: (1, 1)

Posición correcta: (0, 1)

+1

Pieza 3:

Posición inicial: (0, 2)

Posición correcta: (0, 2)

+0

...

Total Hamming Distance: 5

Tablero Inicial

5	7	3
8	2	
1	6	4

Tablero Solución

1	2	3
8		4
7	6	5

Metodo de búsqueda elegido

A*:

- Descartariamos las opciones desinformadas ya que al utilizar métodos informados se llega **soluciones más eficientes**
- Dentro de los informados elegimos A*, ya que en comparación con Greedy, este es **óptimo si la heurística elegida es admisible** (Teorema 3.3.2 apunte) y el costo mínimo de mover una pieza es 1 (mayor a cero)

Heurística elegida

Distancia de Manhattan:

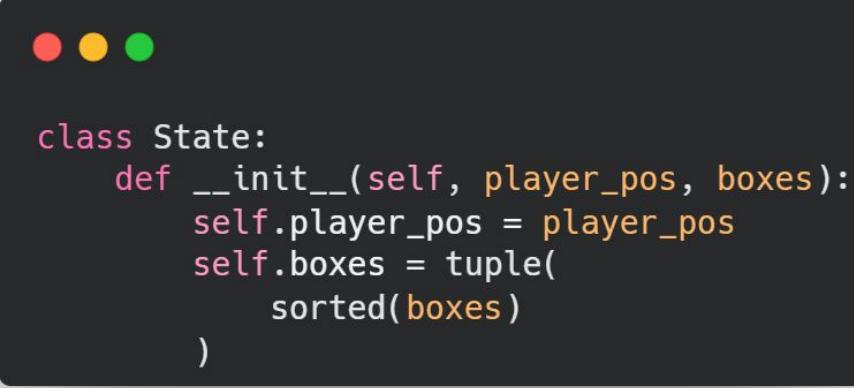
- Ambas heurísticas son no triviales, pues no son funciones constantes, proveen un estimado de la cantidad de pasos o movimientos para llegar al estado correcto, *goal*, sin embargo *Manhattan Distance* es más precisa ya que tiene en cuenta la posición exacta de cada pieza.

Sokoban

Nuestro estado

Una **clase** que contiene:

- Posición del jugador.
- Posición de las cajas.



```
class State:  
    def __init__(self, player_pos, boxes):  
        self.player_pos = player_pos  
        self.boxes = tuple(  
            sorted(boxes)  
        )
```

Manejo de estados repetidos

Utilizamos un set() en el algoritmo que almacena los estados que vamos visitando y los hashCode de estados para un manejo eficiente:

```
def search(self):
    ...
    visited = set()
    ...
    ...

    neighbours = self.get_neighbours(current_node.state)
    for neighbour in neighbours:
        if neighbour not in visited:
            visited.add(neighbour)
            ...
        ...
    ...
}
```

Optimizaciones en métodos informados

Implementamos una clase comparable para utilizar una PriorityQueue

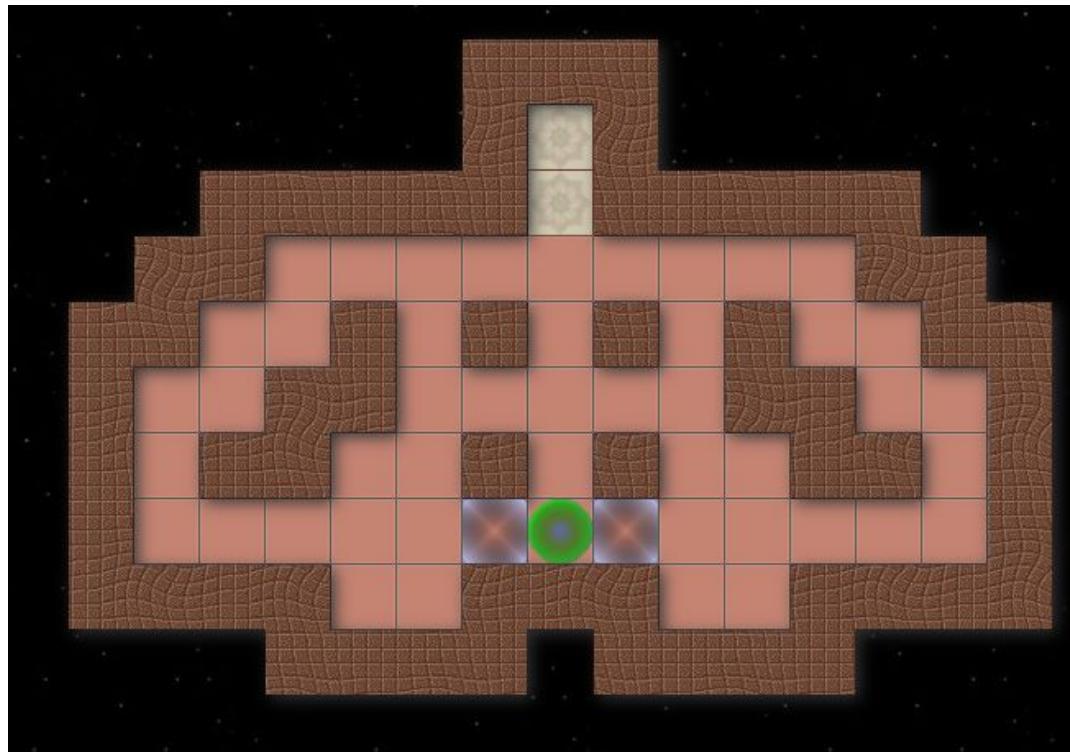
```
class PrioritizedNodeTuple:
    def __init__(self, f_value : float, node : Node):
        self.f_value = f_value
        self.node = node

    def __lt__(self, other):
        return(self.f_value < other.f_value
              or (self.f_value == other.f_value and self.node.h_value < other.node.h_value))

    def __eq__(self, other):
        return self.f_value == other.f_value
```

B.E.S vs. D.E.S

Mapa: soko01



B.F.S

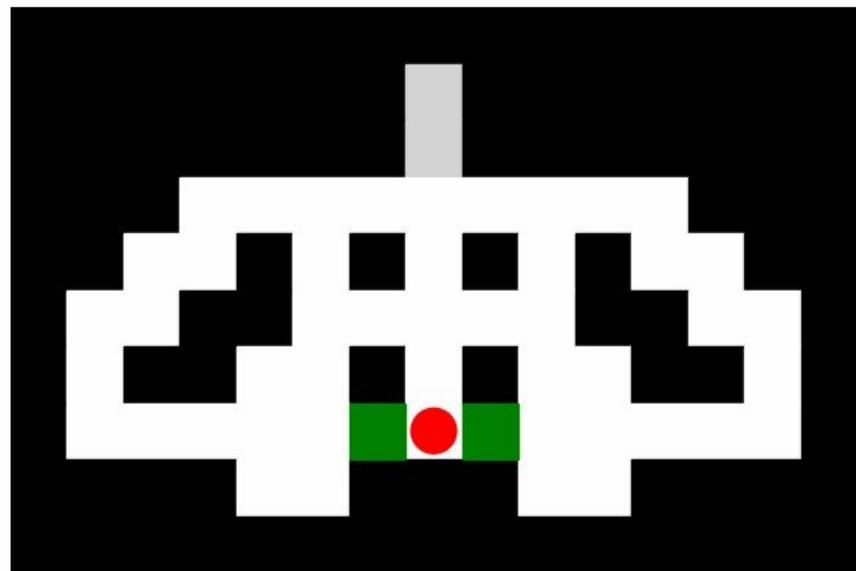
Tiempo: $0.57 \pm 0.07\text{s}$ (15 runs)

Nodos Expandidos: 30376

Nodos Frontera Totales: 19

Costo (longitud de solución): 78 (Óptimo)

Solución:



D.F.S

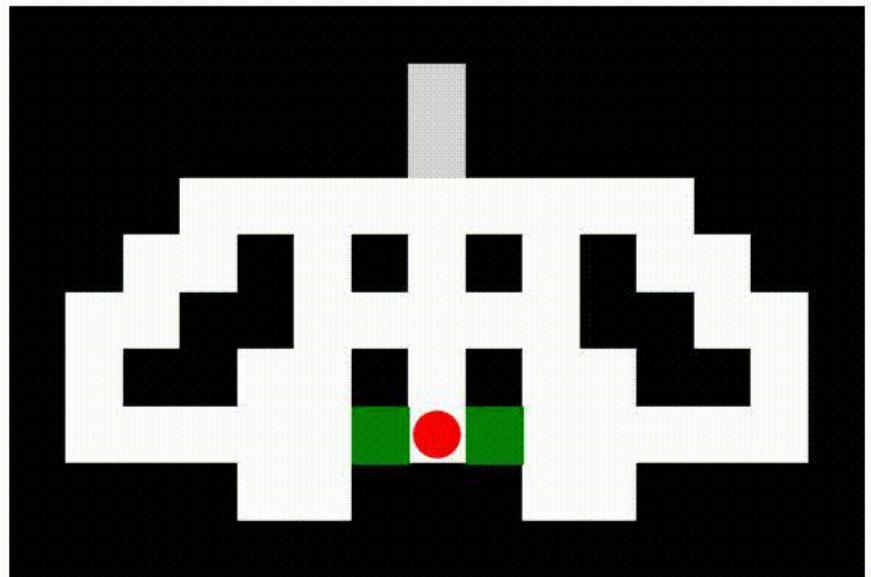
Tiempo: 0.24 ± 0.05 s (15 runs)

Nodos Expandidos: 13536

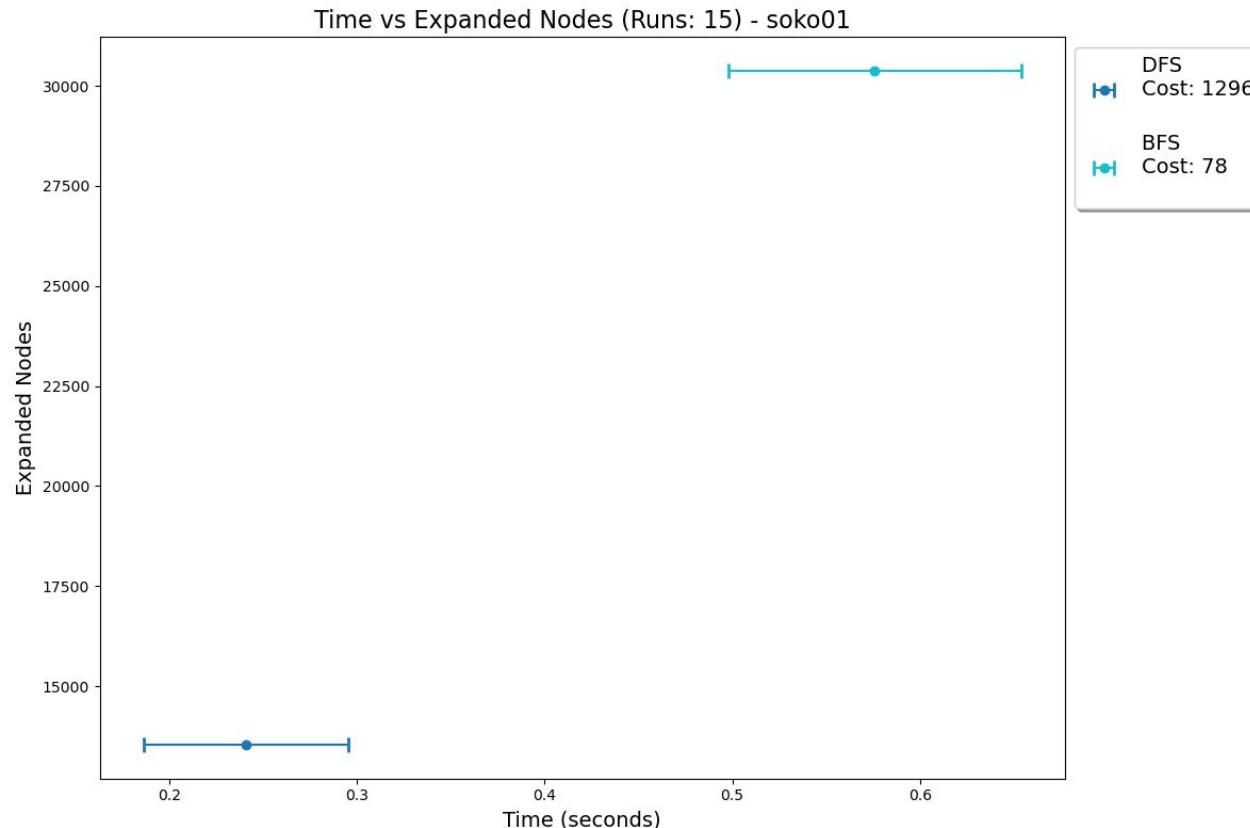
Nodos Frontera Totales: 527

Costo (longitud de solución): 1296

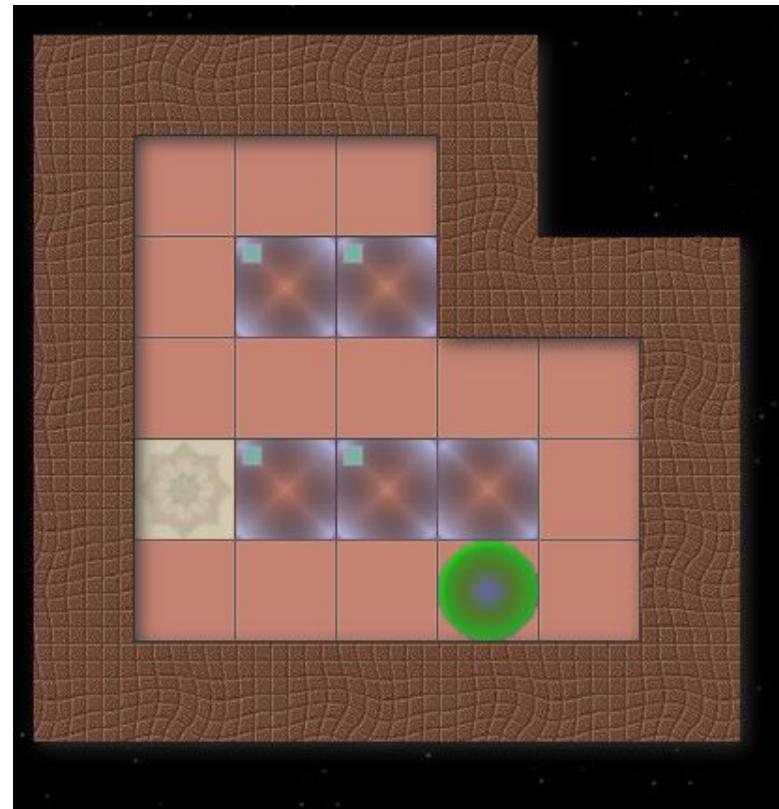
Solución:



Comparación de tiempo y memoria



Mapa: sokoA01



B.F.S

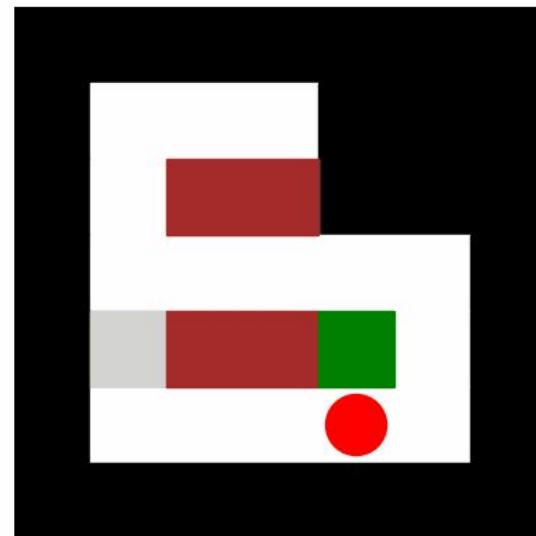
Tiempo: $6.77 \pm 0.26s$ (15 runs)

Nodos Expandidos: 299268

Nodos Frontera: 1732

Costo (longitud de solución): 43 (Óptimo)

Solución:



D.F.S

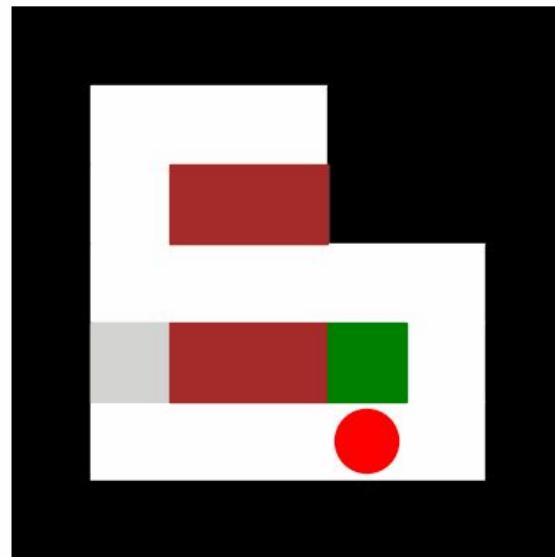
Tiempo: $6.38 \pm 0.45\text{s}$ (15 runs)

Nodos Expandidos: 304867

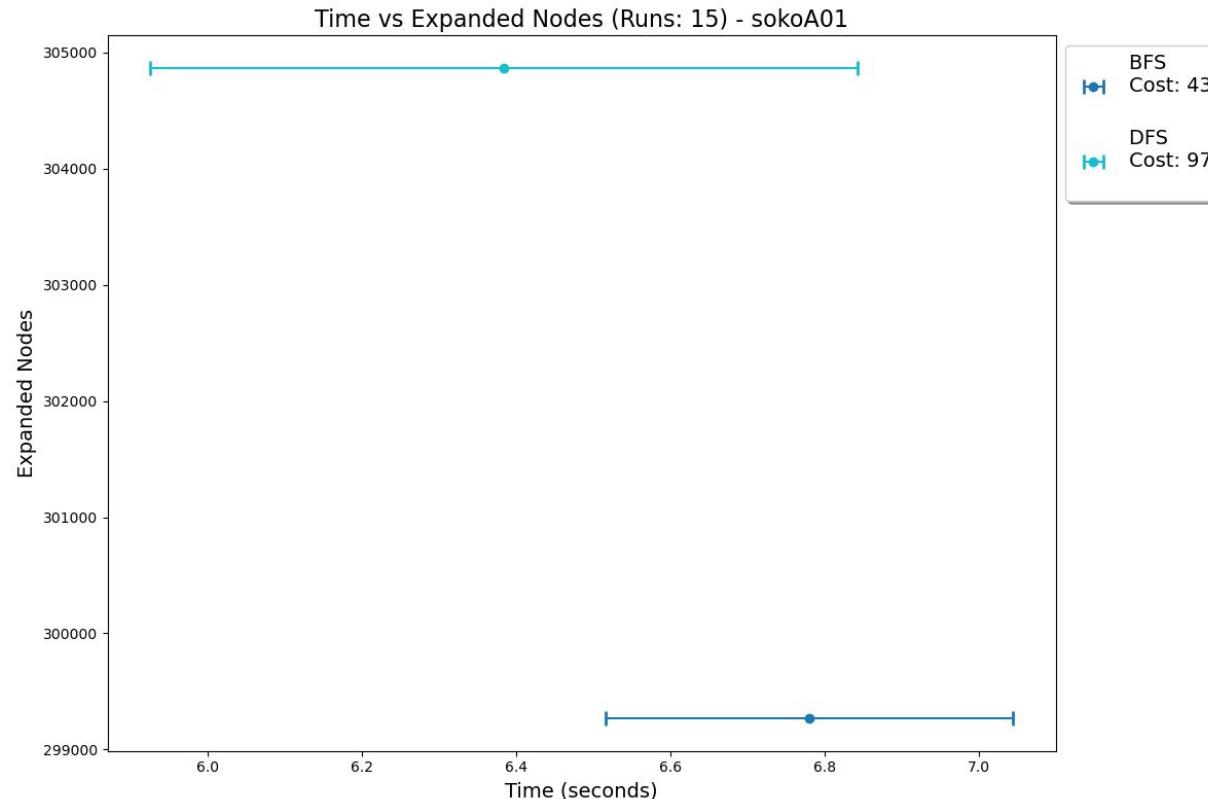
Nodos Frontera: 42

Costo (longitud de solución): 97

Solución:



Comparación de tiempo y memoria



BFS vs DFS

- BFS generalmente tarda más tiempo y usa más memoria que DFS ya que busca la solución óptima y DFS busca la primera que encuentre.
- DFS al dar la primera solución que encuentra puede llegar a ser muy mala.
- BFS siempre dará la solución óptima.

A* vs Greedy

¿Porqué Global Greedy?

Elige el nodo de menor costo de manera global gracias a una cola de prioridad que tiene en cuenta todos los nodos visitados



```
queue = PriorityQueue()
```

Heurística Admisible

Mapa: soko01 con Manhattan



Greedy Search con Manhattan

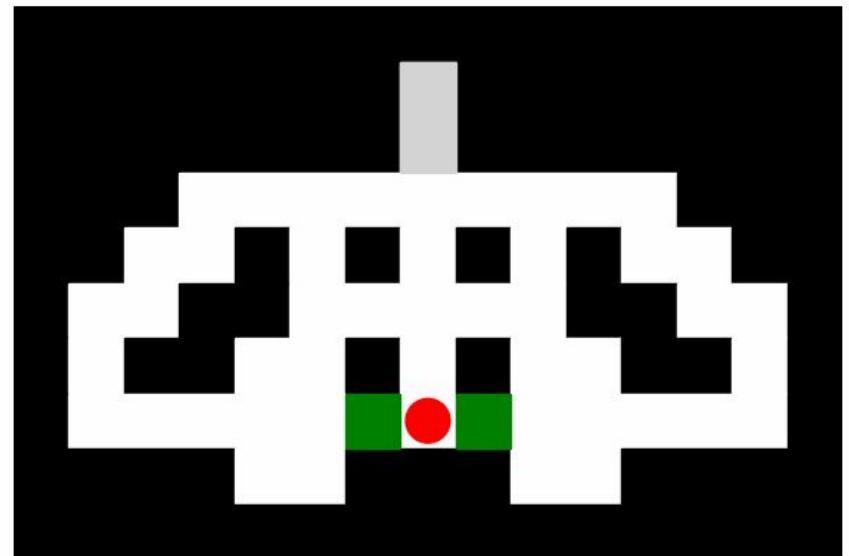
Tiempo: 0.018 ± 0.005 s (15 runs)

Nodos Expandidos: 771

Nodos Frontera: 67

Costo (longitud de solución): 104

Solución:



A* Search con Manhattan

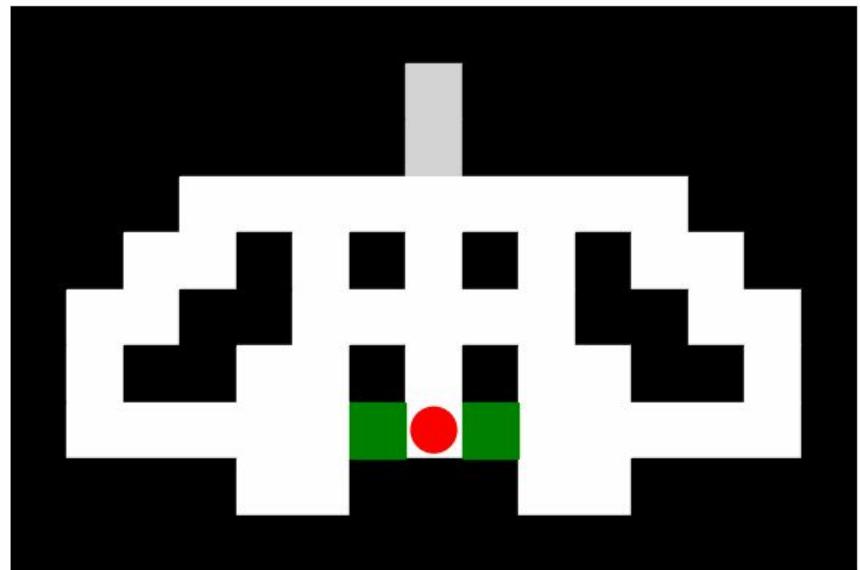
Tiempo: 0.96 ± 0.07 s (15 runs)

Nodos Expandidos: 30297

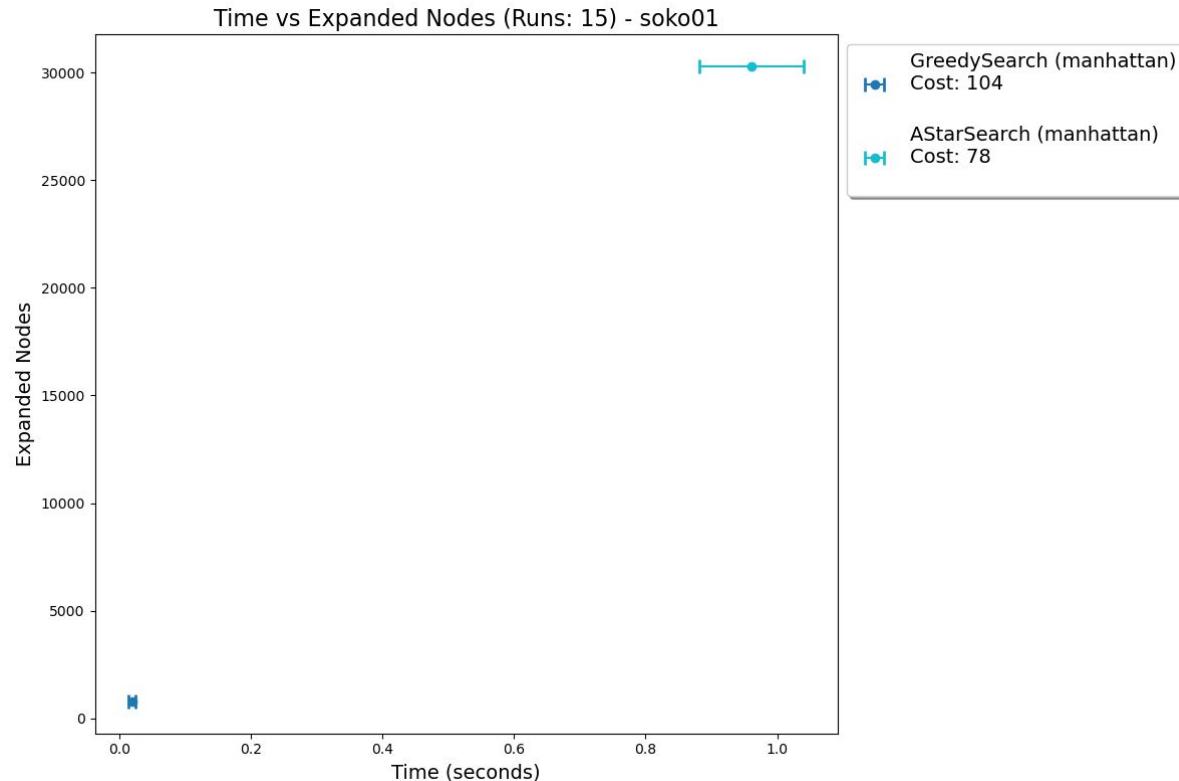
Nodos Frontera Totales: 42

Costo (longitud de solución): 78 (Óptimo)

Solución:



Comparación de tiempo y espacio



Heurística No admissible

Mapa: soko01 con Weighted Manhattan



Greedy Search con Weighted Manhattan

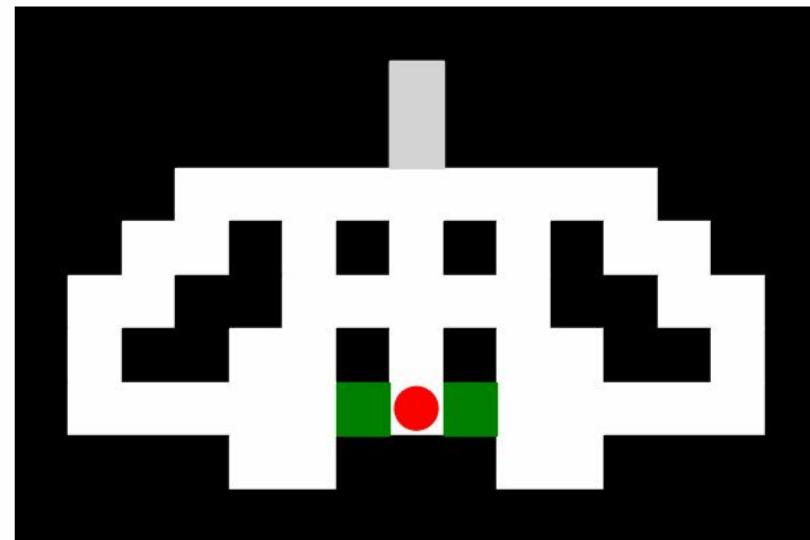
Tiempo: $0.146 \pm 0.019s$ (15 runs)

Nodos Expandidos: 3910

Nodos Frontera: 992

Costo (longitud de solución): 118

Solución:



A* Search con Weighted Manhattan

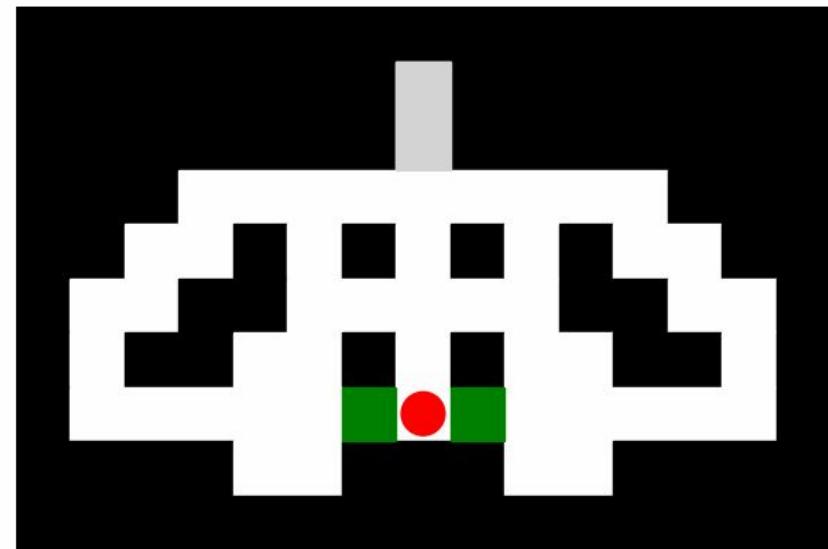
Tiempo: $0.597 \pm 0.036s$ (15 runs)

Nodos Expandidos: 15578

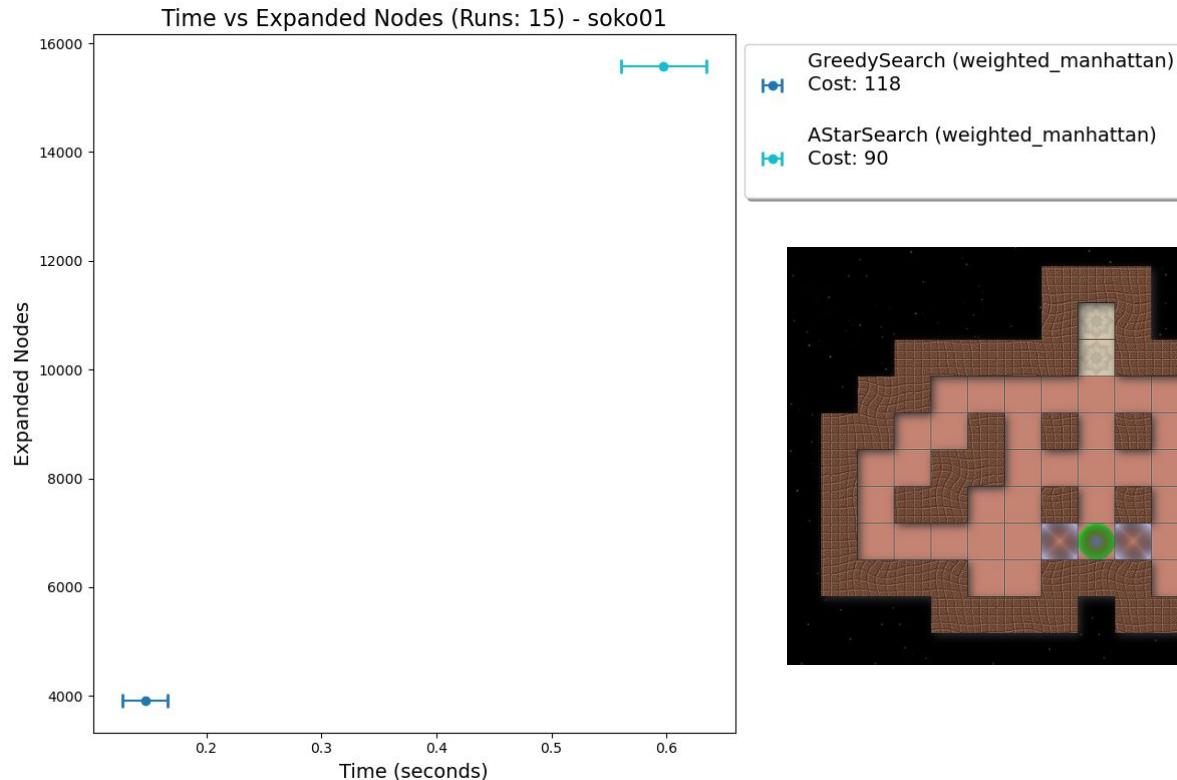
Nodos Frontera: 2155

Costo (longitud de solución): 90

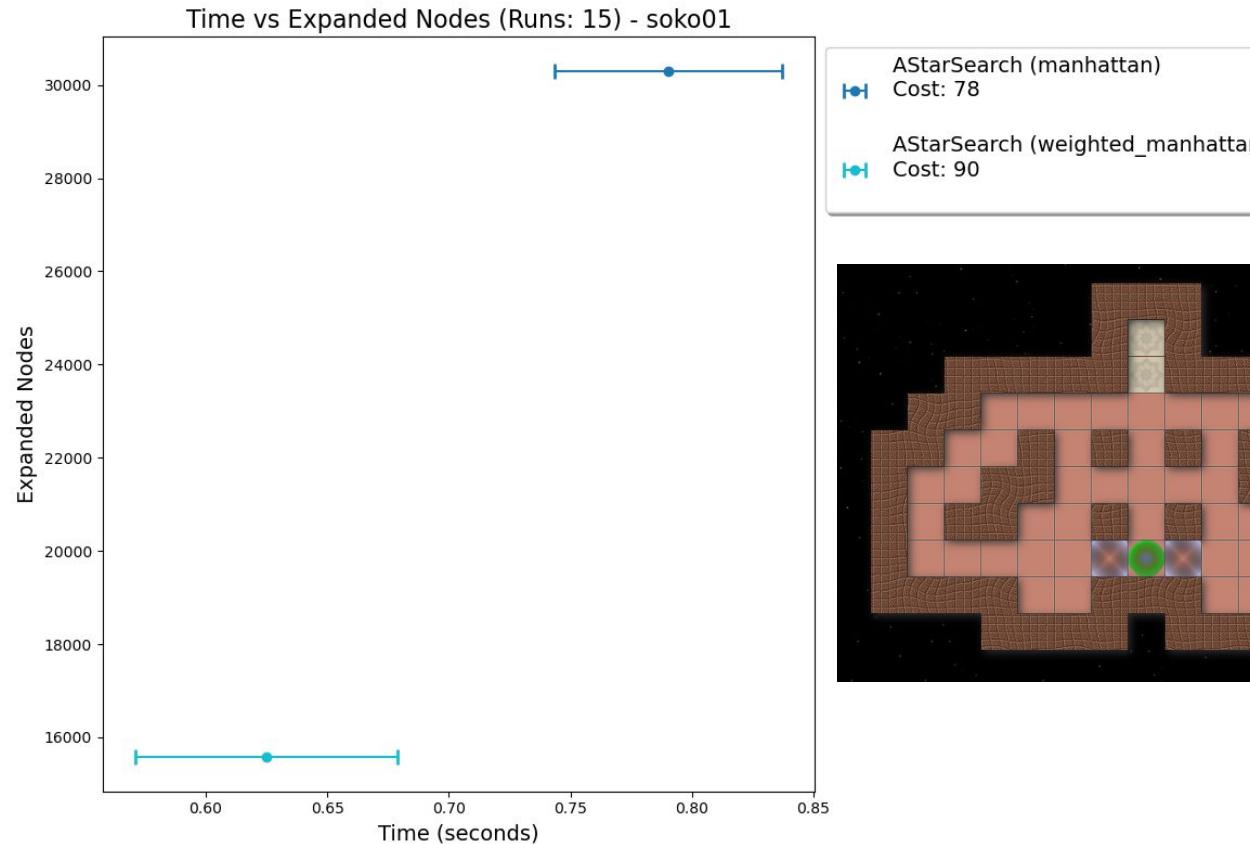
Solución:



Comparación de tiempo y espacio

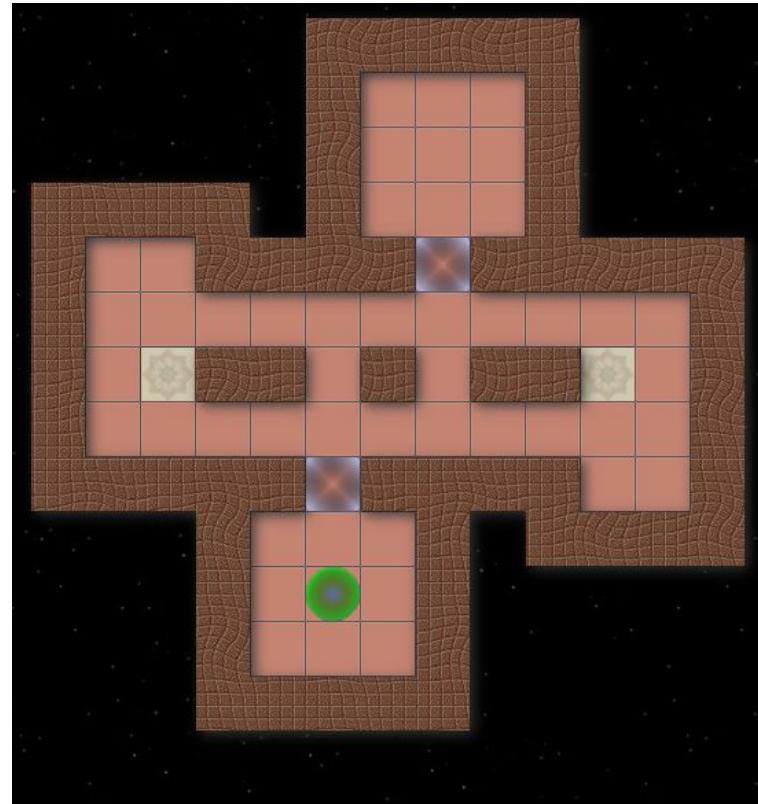


Comparación A* admisible y no admisible



Heurística Admisible

Mapa: sokoll con Manhattan



Greedy Search con Manhattan

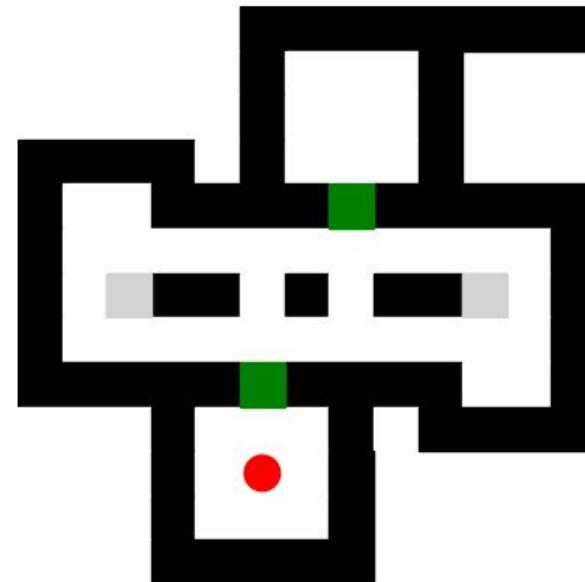
Tiempo: $0.0343 \pm 0.007s$ (15 runs)

Nodos Expandidos: 1055

Nodos Frontera Totales: 108

Costo (longitud de solución): 64

Solución:



A* Search con Manhattan

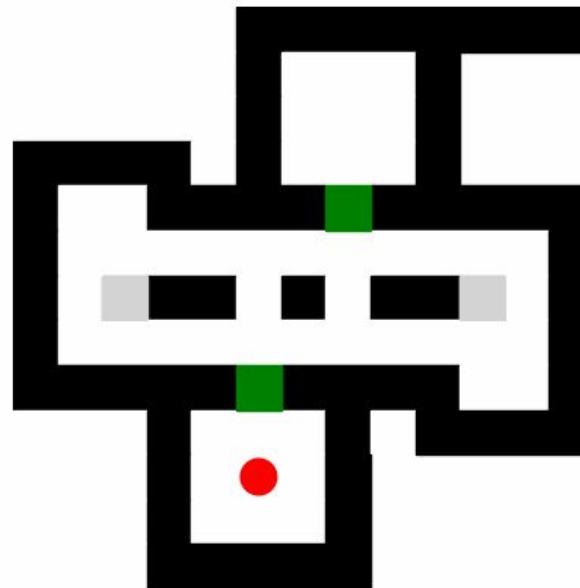
Tiempo: 1.554 ± 0.186 s (15 runs)

Nodos Expandidos: 51422

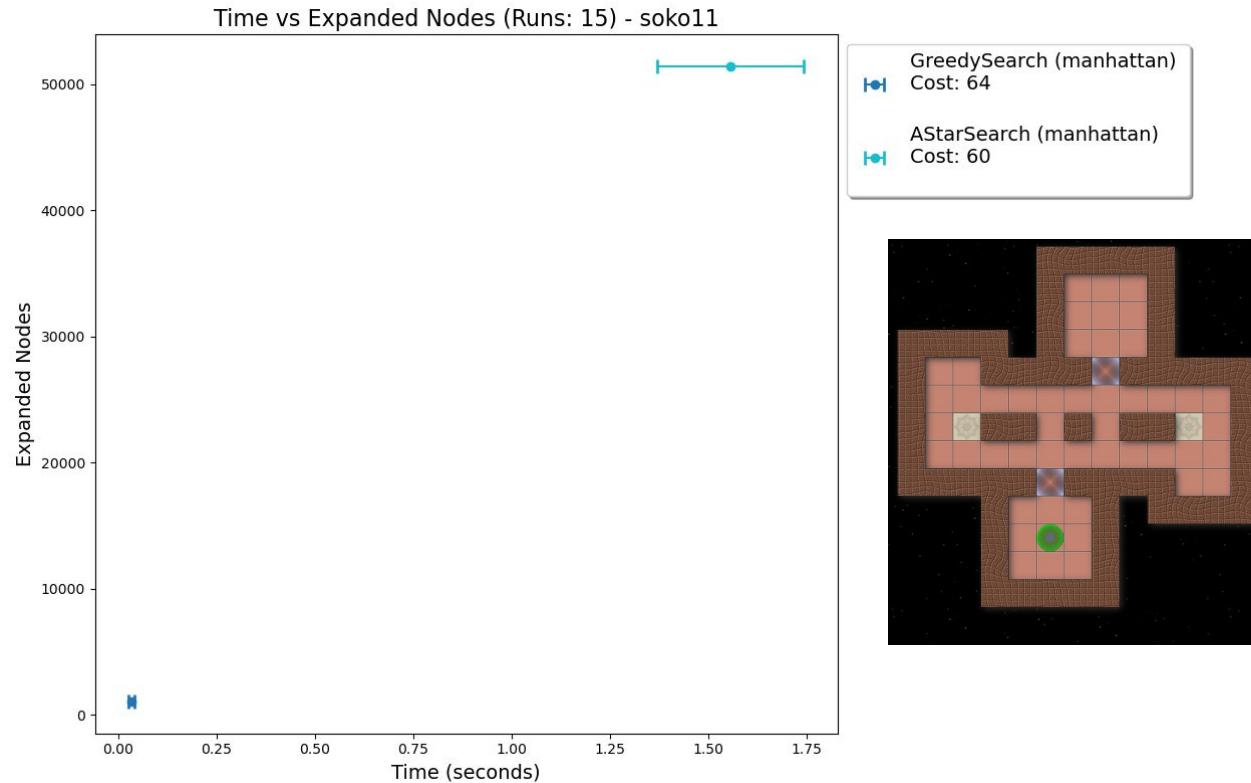
Nodos Frontera Totales: 1673

Costo (longitud de solución): 60 (Óptimo)

Solución:

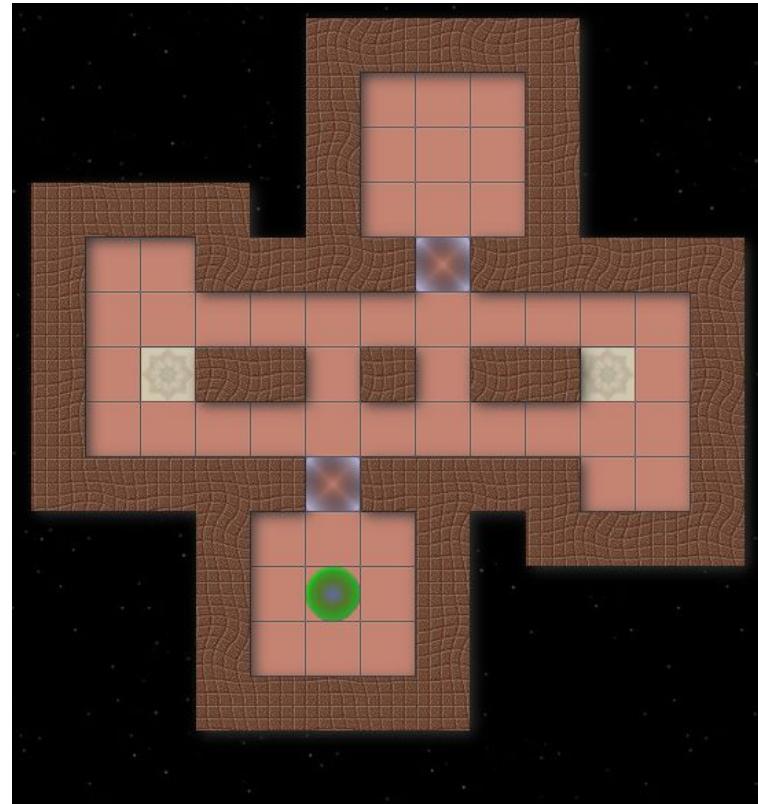


Comparación de tiempo y memoria



Heurística No admissible

Mapa: sokoll con Weighted Manhattan



Greedy Search con Weighted Manhattan

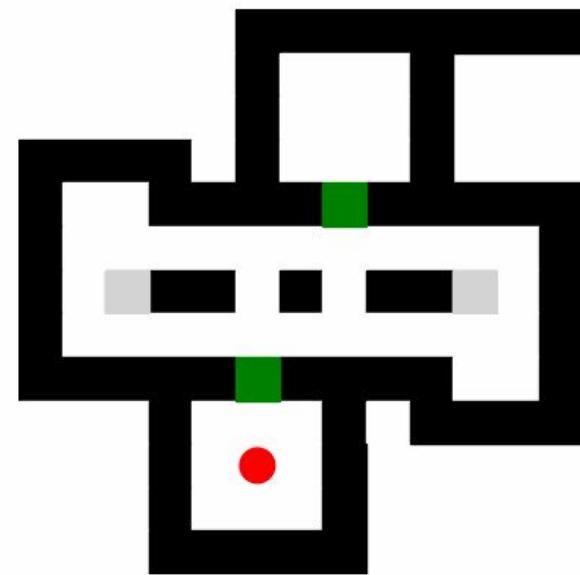
Tiempo: $0.041 \pm 0.002s$ (15 runs)

Nodos Expandidos: 1053

Nodos Frontera Totales: 442

Costo (longitud de solución): 108

Solución:



A* Search con Weighted Manhattan

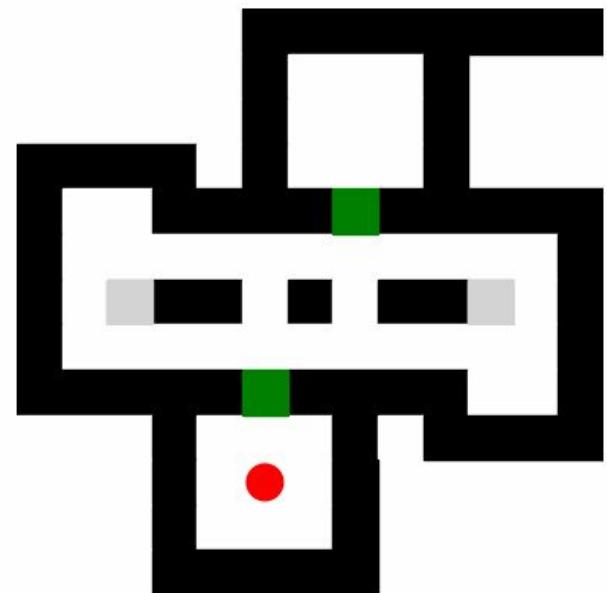
Tiempo: $0.795 \pm 0.051\text{s}$ (15 runs)

Nodos Expandidos: 18672

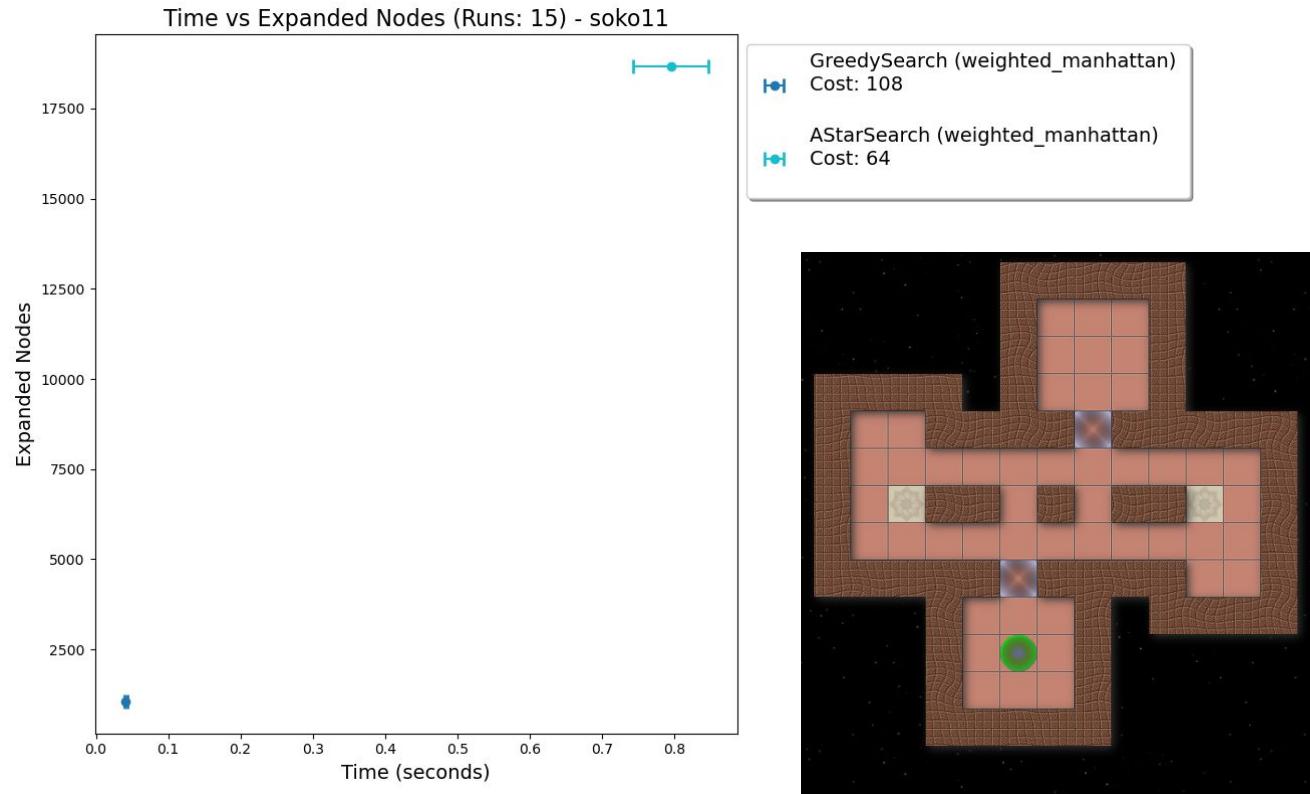
Nodos Frontera Totales: 4292

Costo (longitud de solución): 64

Solución:



Comparación de tiempo y espacio



Global Greedy Search vs A* Search

- Greedy es más rápido y utiliza menos memoria pero normalmente da soluciones largas
- A* utiliza más recursos de tiempo y memoria pero garantiza la solución óptima si la heurística es admisible
- A* con buenas heurísticas no admisibles optimiza tiempos y memoria y puede dar una solución mejor que greedy

Heurísticas

Manhattan Distance

Características:

- Admissible.
- $d = |x_1 - x_0| + |y_1 - y_0|$.

Minimum Matching Distance

Características:

- Admissible.
- Matriz de peso.
- Manhattan Distance.
- Hungarian Method.

Weighted Manhattan Distance

Características:

- No es admisible.
- Penaliza:
 - Cajas juntas.
 - Distancia del jugador a las cajas.
 - Distancia de las cajas a las paredes.
- Manhattan Distance.

Box In Target / Hamming Distance

Características:

- Admisible.
- Retorna cantidad de cajas.
- Resta uno por cada caja ubicada en target.

Combined

Características:

- Depende la composición.
- Combina dos heurísticas con un peso determinado:
- $n * h1() + (1-n) h2()$.

Blocked

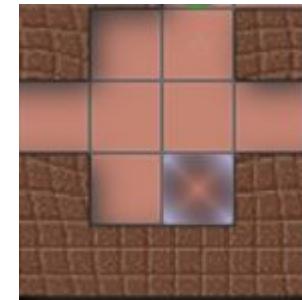
Características:

- Admisible (según con que se combine).
- Si detecta un estado muerto retorna el valor máximo posible por un float sino el valor de la heurística secundaria elegida.

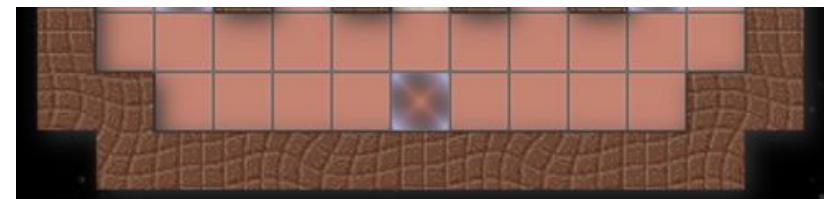
Deadlocks

Deadlocks - Casos

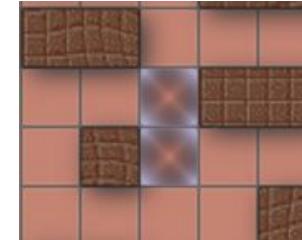
1. Box in corner



2. Box in U-shape (caso 1 extendido)

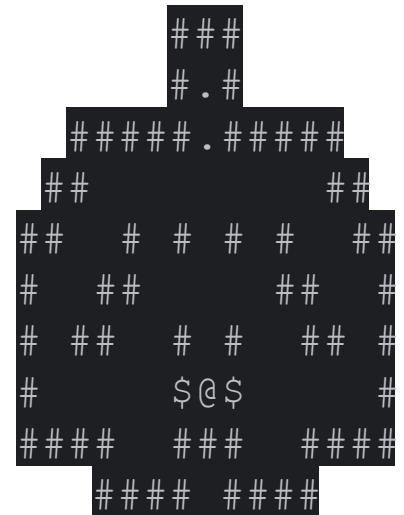
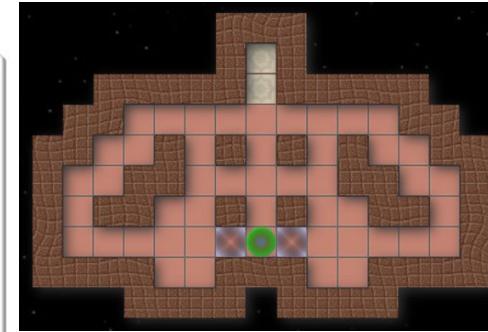
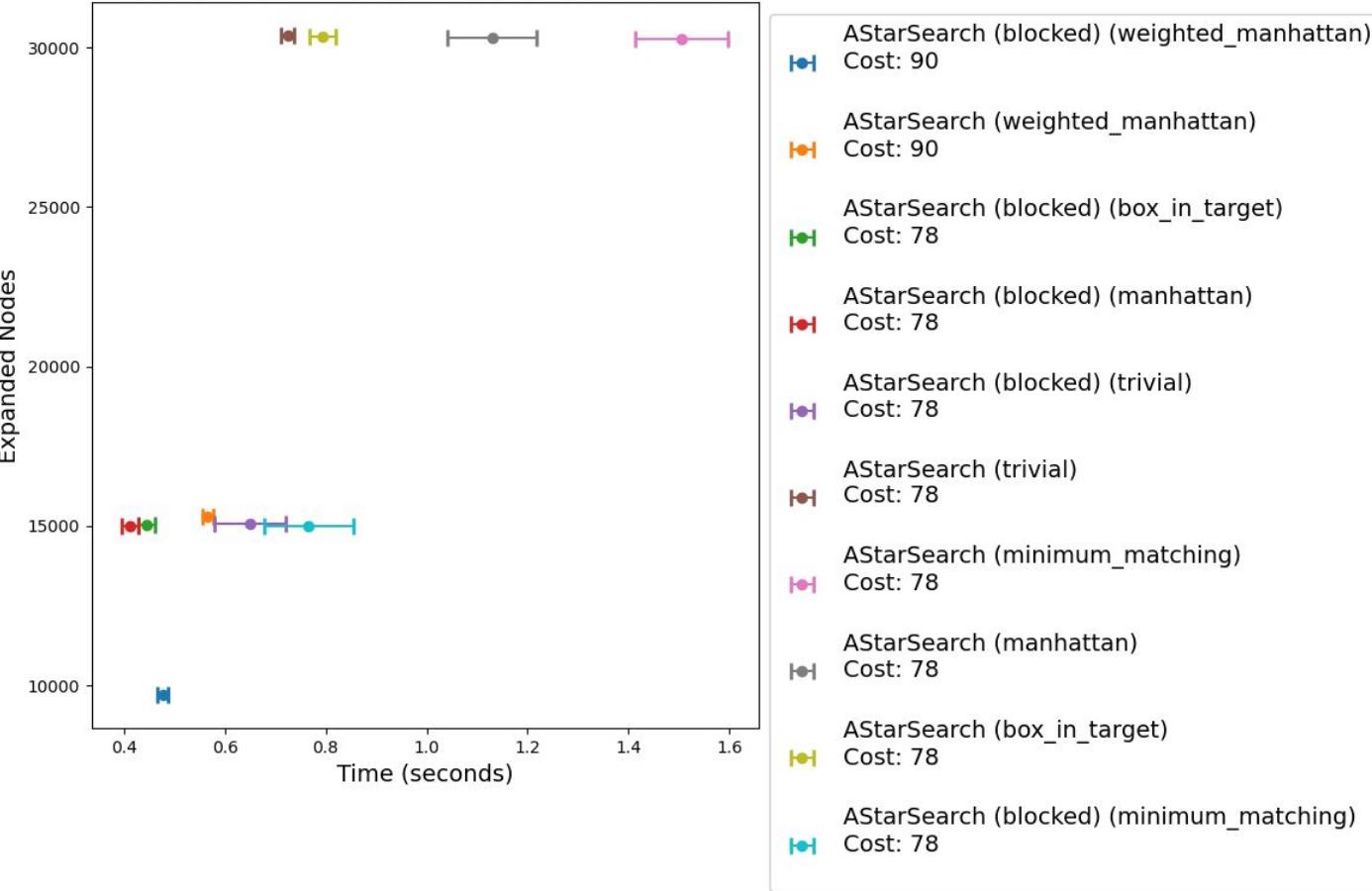


3. Adjacent boxes



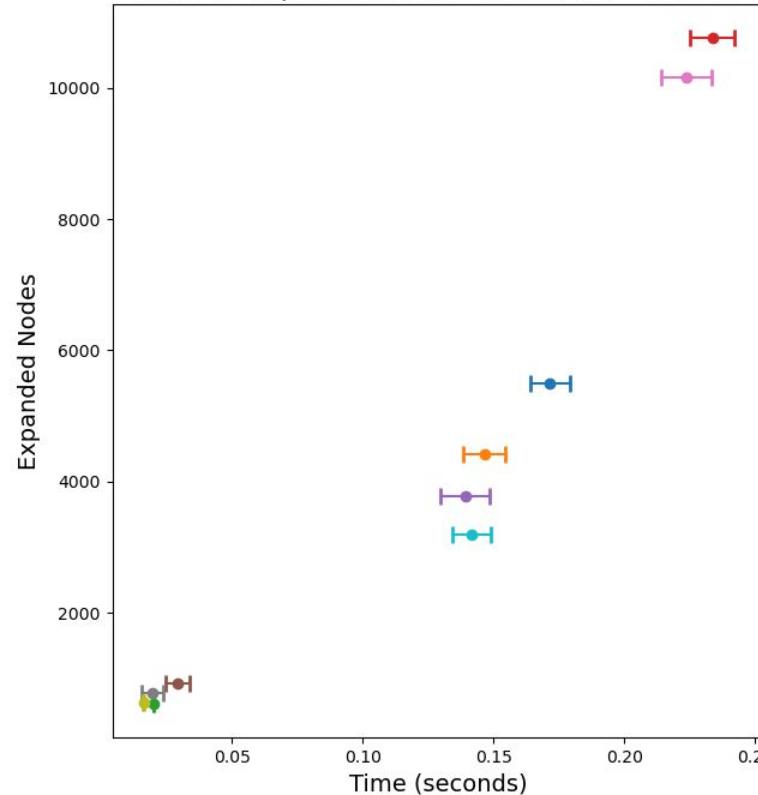
Deadlocks - Heurística Blocked

Time vs Expanded Nodes (Runs: 10) - soko01

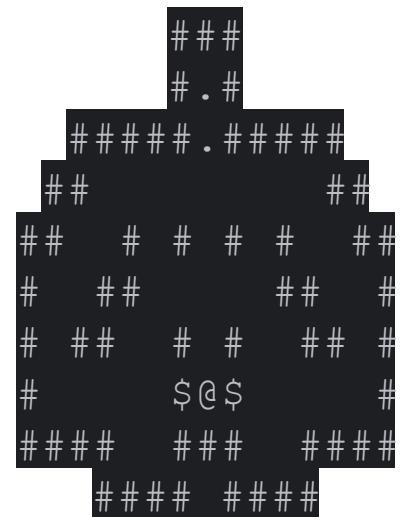
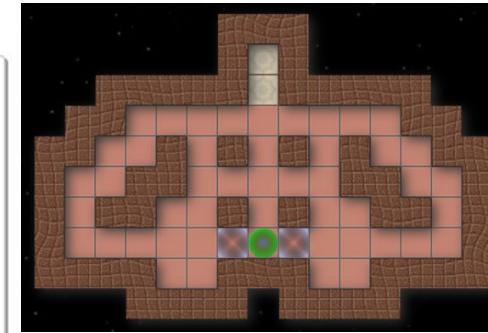


Deadlocks - Heurística Blocked

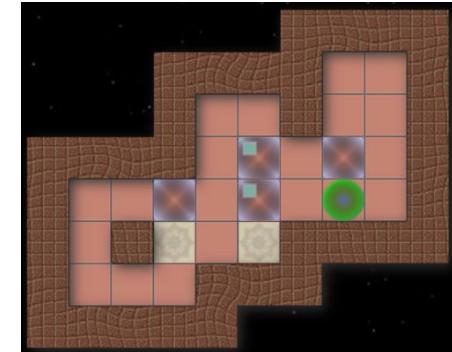
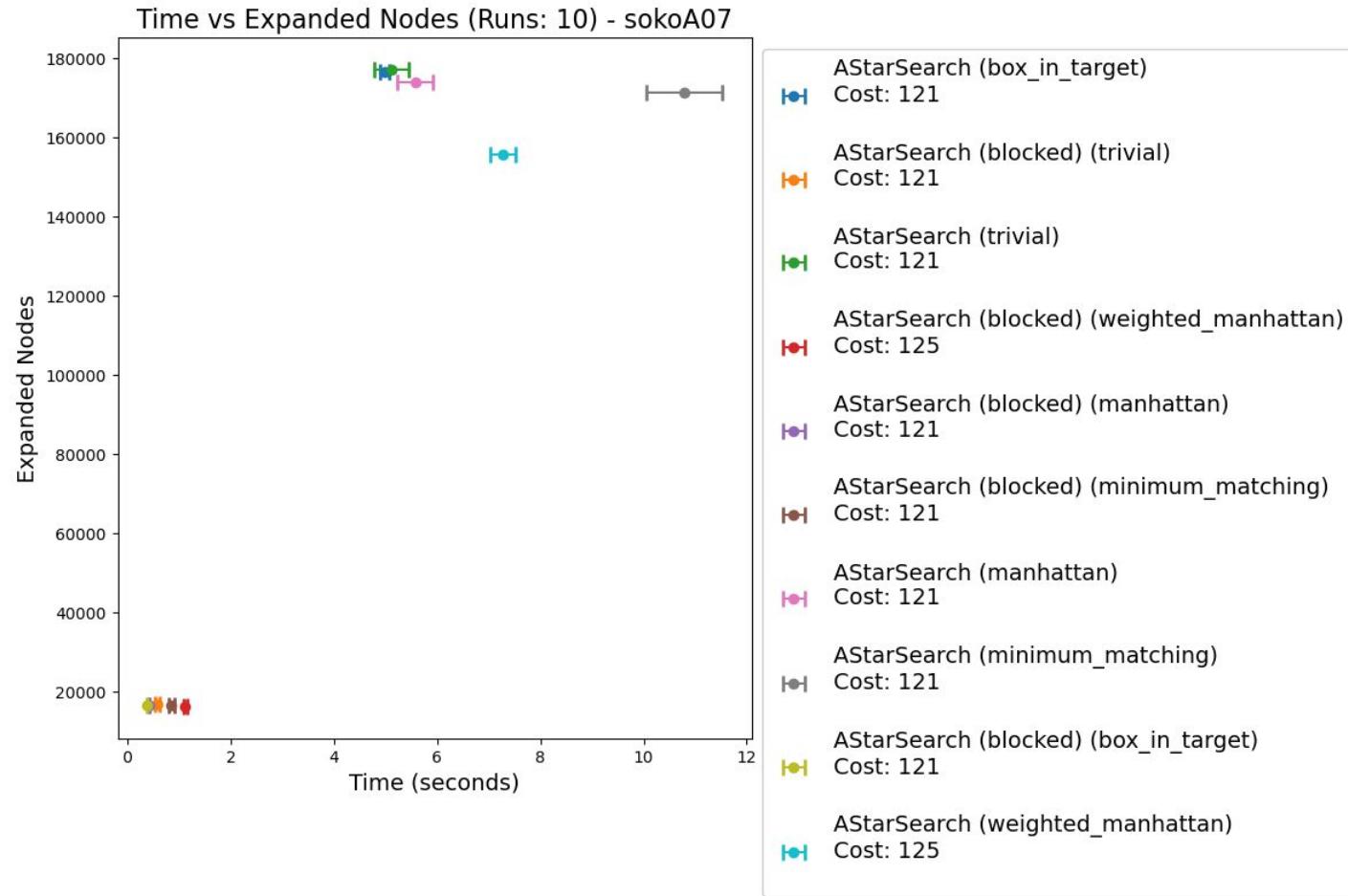
Time vs Expanded Nodes (Runs: 10) - soko01



- GreedySearch (blocked) (trivial)
Cost: 362
- GreedySearch (blocked) (box_in_target)
Cost: 442
- GreedySearch (blocked) (minimum_matching)
Cost: 96
- GreedySearch (box_in_target)
Cost: 448
- GreedySearch (weighted_manhattan)
Cost: 112
- GreedySearch (blocked) (manhattan)
Cost: 114
- GreedySearch (trivial)
Cost: 604
- GreedySearch (manhattan)
Cost: 104
- GreedySearch (minimum_matching)
Cost: 104
- GreedySearch (blocked) (weighted_manhattan)
Cost: 122

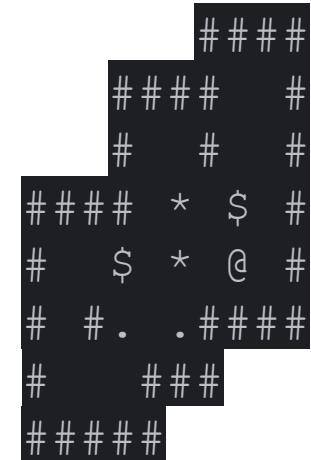
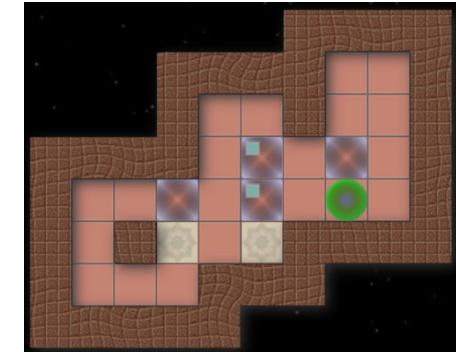
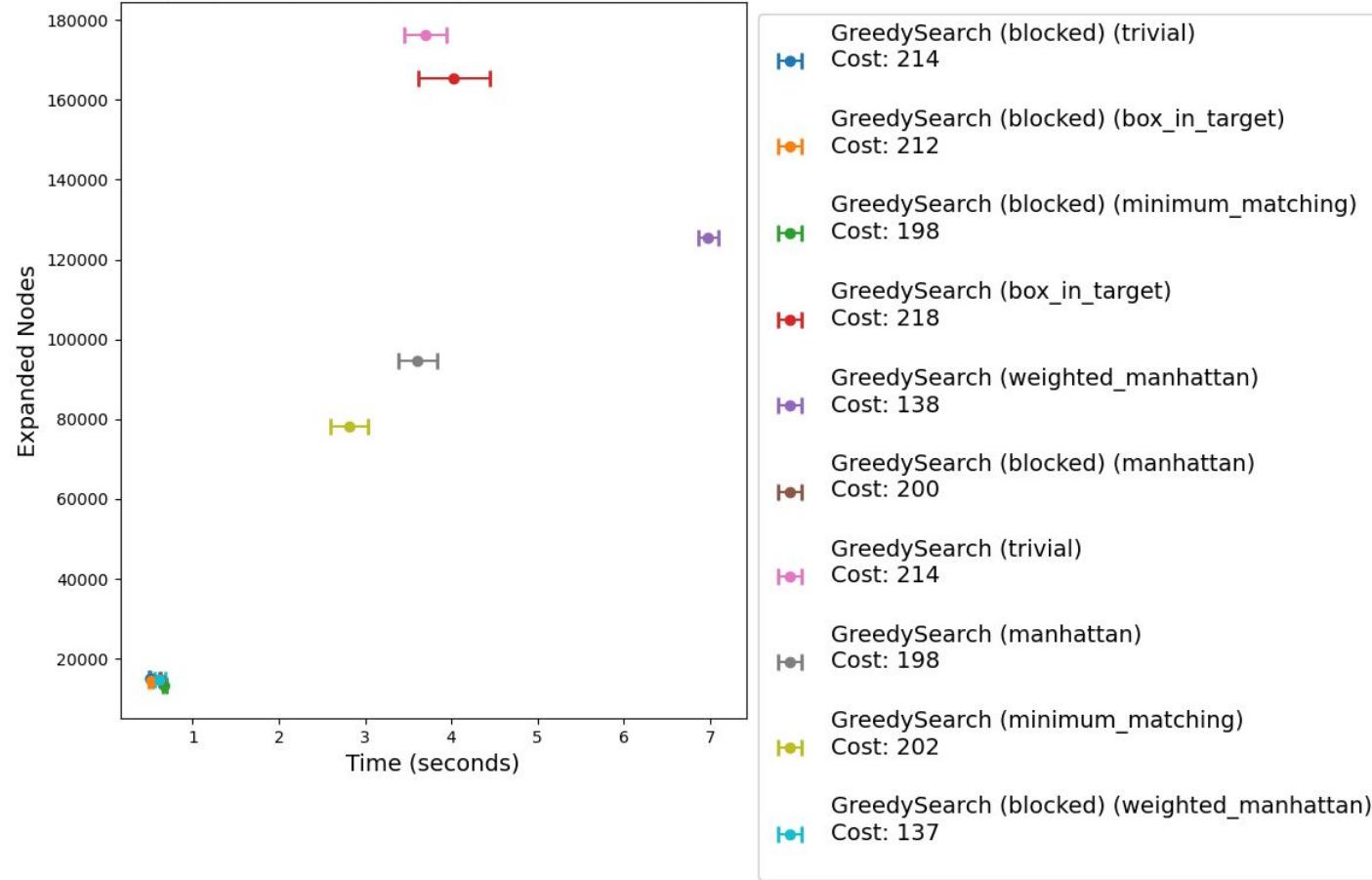


Deadlocks - Heurística Blocked



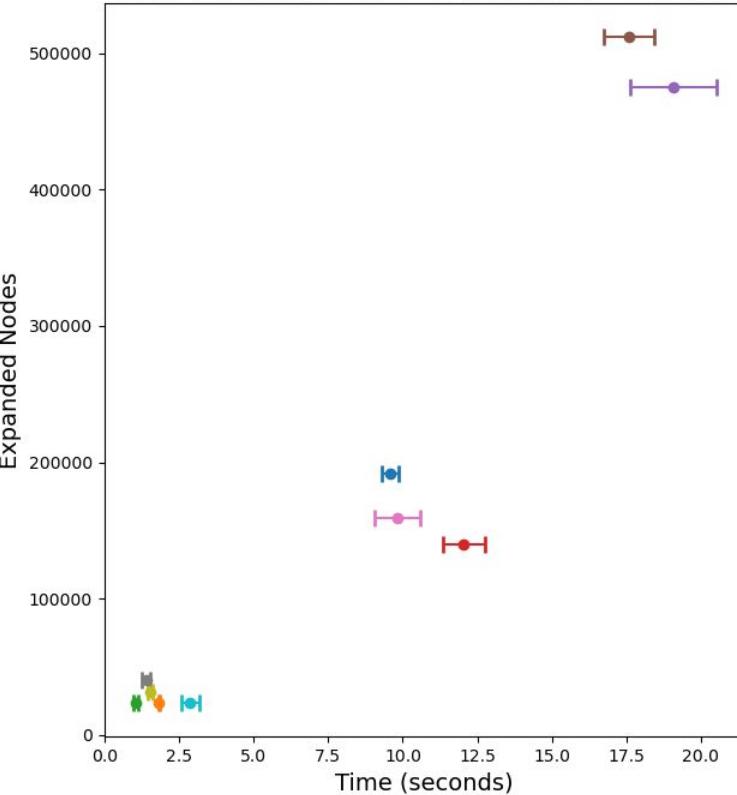
Deadlocks - Heurística Blocked

Time vs Expanded Nodes (Runs: 10) - sokoA07

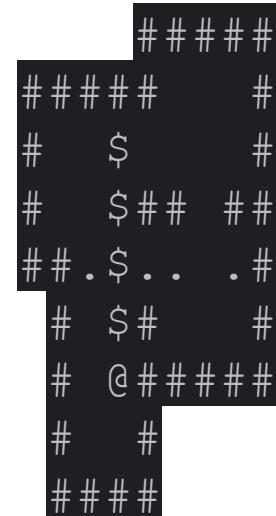
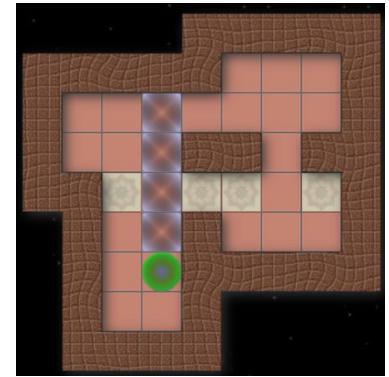


Deadlocks - Heurística Blocked

Time vs Expanded Nodes (Runs: 10) - sokoA12

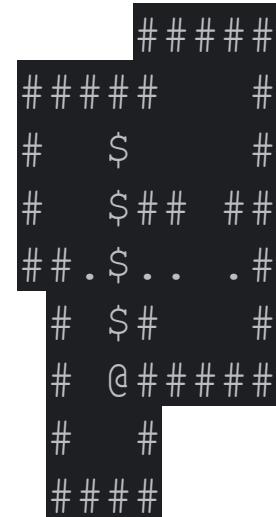
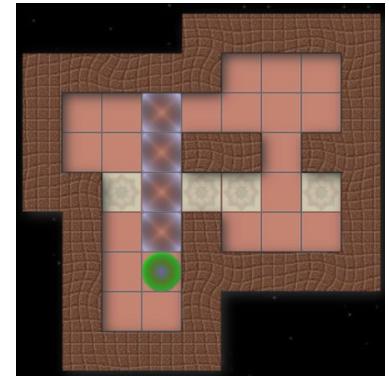
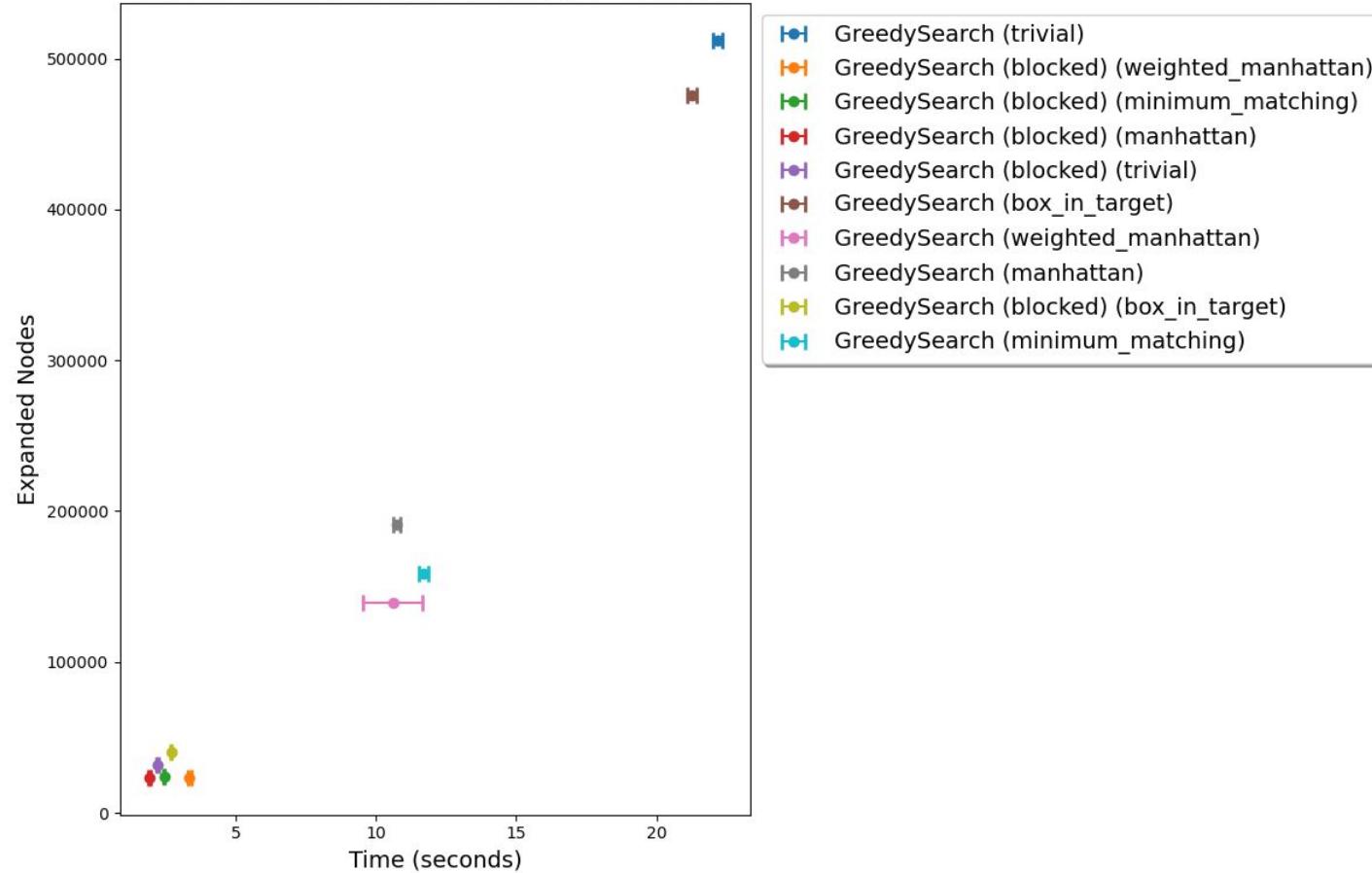


- GreedySearch (manhattan)
Cost: 258
- GreedySearch (blocked) (minimum_matching)
Cost: 220
- GreedySearch (blocked) (manhattan)
Cost: 188
- GreedySearch (weighted_manhattan)
Cost: 222
- GreedySearch (box_in_target)
Cost: 378
- GreedySearch (trivial)
Cost: 562
- GreedySearch (minimum_matching)
Cost: 176
- GreedySearch (blocked) (box_in_target)
Cost: 194
- GreedySearch (blocked) (trivial)
Cost: 180
- GreedySearch (blocked) (weighted_manhattan)
Cost: 222



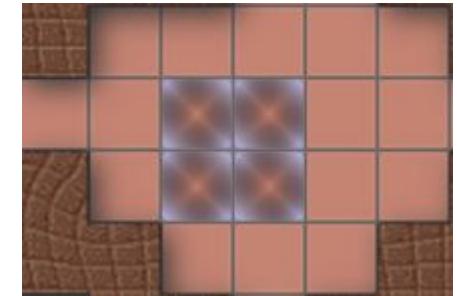
Deadlocks - Heurística Blocked

Time vs Expanded Nodes (Runs: 10) - sokoA12

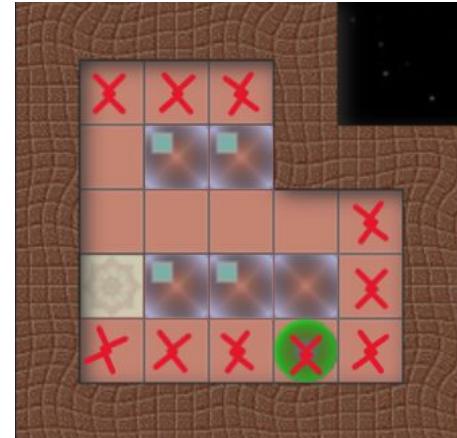


Deadlocks - Mejoras posibles

- Cuarto caso de deadlock: 4 boxes in a square
- A* Optimizado vs. Heurística Blocked



- Precalcuло de deadlocks de cada board
- Deadlocks precalculados vs. Heurística Blocked



Comparación de Heurísticas

Categorización de mapas

- Cantidad de cajas
- Cantidad de objetivos
- Tamaño total del mapa

Mapas *Easy*

soko01

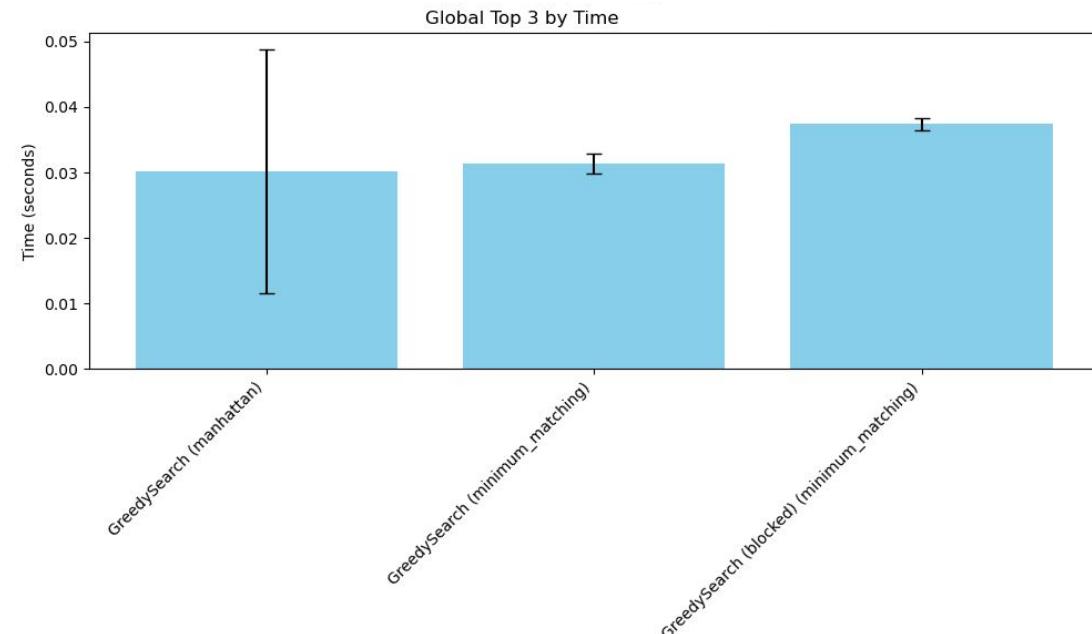
- **2 cajas**
- **caminos sencillos**
- **baja obstaculización**

```
###  
#. #  
#####. #####  
##      ##  
##  #  #  #  #  #  
#  ##  ##  ##  #  
#  ##  #  #  ##  #  
#      $@$      #  
####  ####  ####  
####  ####
```

Mejor tiempo

- **Método:** Global Greedy Search
- **Heurística:** manhattan
- **Tiempo:** 0.0302 ± 0.0186 s
- **Costo de la solución:** 104

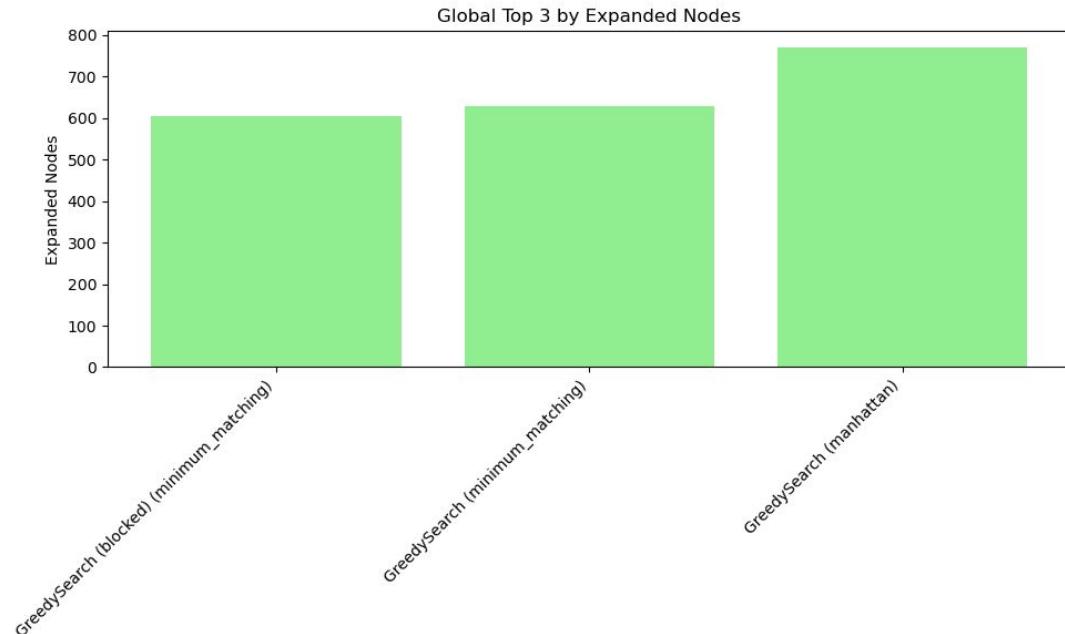
```
###  
#. #  
#####.#####  
## ## ## ##  
## # # # # #  
# # # # # # #  
# # # # # # #  
# $@$ #  
#### # # # ####  
#####
```



Mejor espacio

- **Método:** Global Greedy Search
- **Heurística:** blocked + minimum matching
- **Tiempo:** 605 bloques expandidos
- **Costo de la solución:** 96

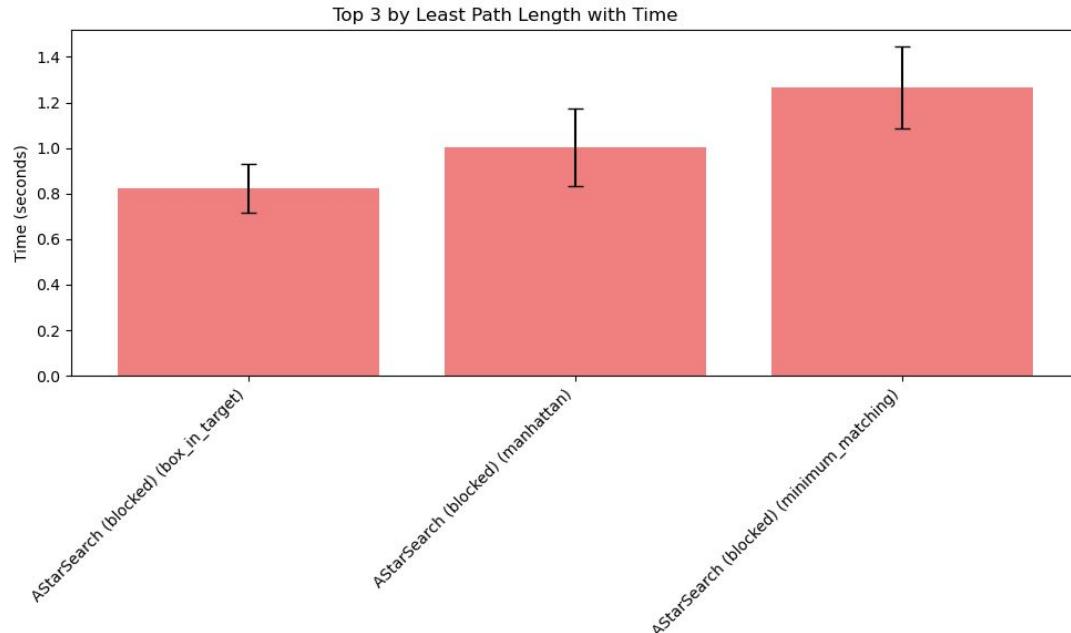
```
###  
#. #  
#####.#####  
##      ##  
##  #  #  #  #  ##  
#  ##  ##  ##  #  
#  ##  #  #  ##  #  
#  $@$      #  
#####  #####  #####  
####  ####
```



Mejor tiempo para el camino óptimo

- **Método:** A* Search
- **Heurística:** blocked + box in target
- **Tiempo:** 0.823 ± 0.108 s
- **Costo de la solución:** 78

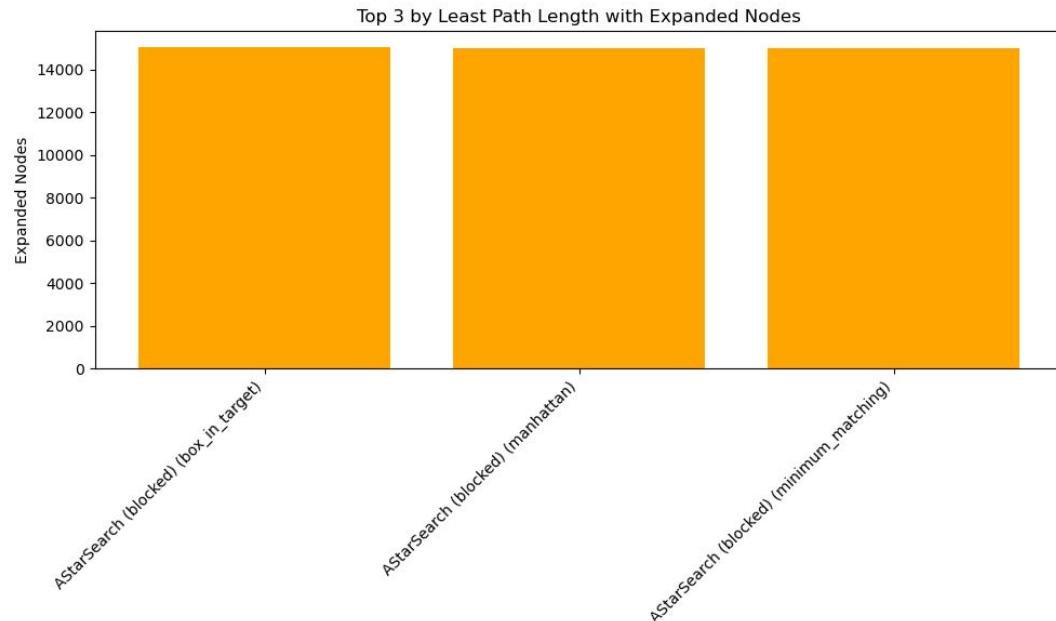
```
###  
.#.  
#####.#####  
##      ##  
##  # # # # #  
#  ##      ## #  
#  ## # # # # #  
#  $@$      #  
#### # # # ####  
#### # # # #
```



Mejor espacio para el camino óptimo

- **Método:** A* Search
- **Heurística:** blocked + minimum matching
- **Tiempo:** 14981 bloques expandidos
- **Costo de la solución:** 78

```
###  
#. #  
#####.#####  
##      ##  
##  #  #  #  #  ##  
#  ##  ##  #  
#  ##  #  #  ##  #  
#  $@$  #  
#####  #####  
####  ####
```



Mapas *Medium*

sokoA07

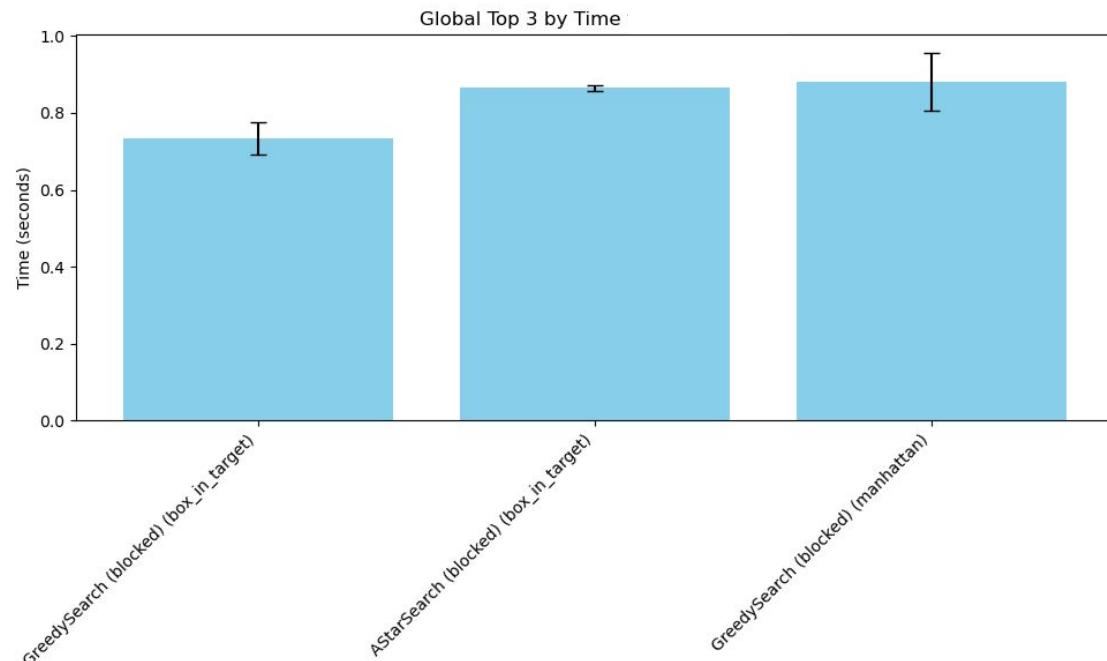
- **4 cajas**
- **espacio escaso**
- **obstaculización notable**

```
#####
##### #
# # #
##### * $ #
# $ * @ #
# #. .#####
#     ####
#####
```

Mejor tiempo

- **Método:** Global Greedy Search
- **Heurística:** blocked + box in target
- **Tiempo:** 0.7339 ± 0.0425 s
- **Costo de la solución:** 212

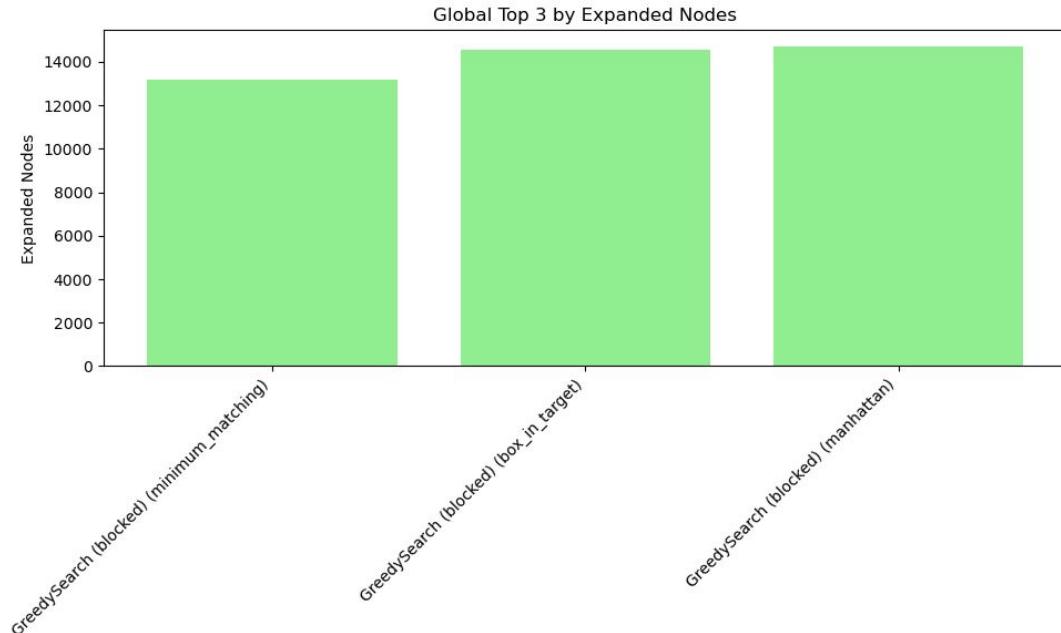
```
#####
# ## #
#   # #
##### * $ #
# $ * @ #
# #. .####
#   ####
#####
```



Mejor espacio

- **Método:** Global Greedy Search
- **Heurística:** blocked + minimum matching
- **Tiempo:** 13161 bloques expandidos
- **Costo de la solución:** 198

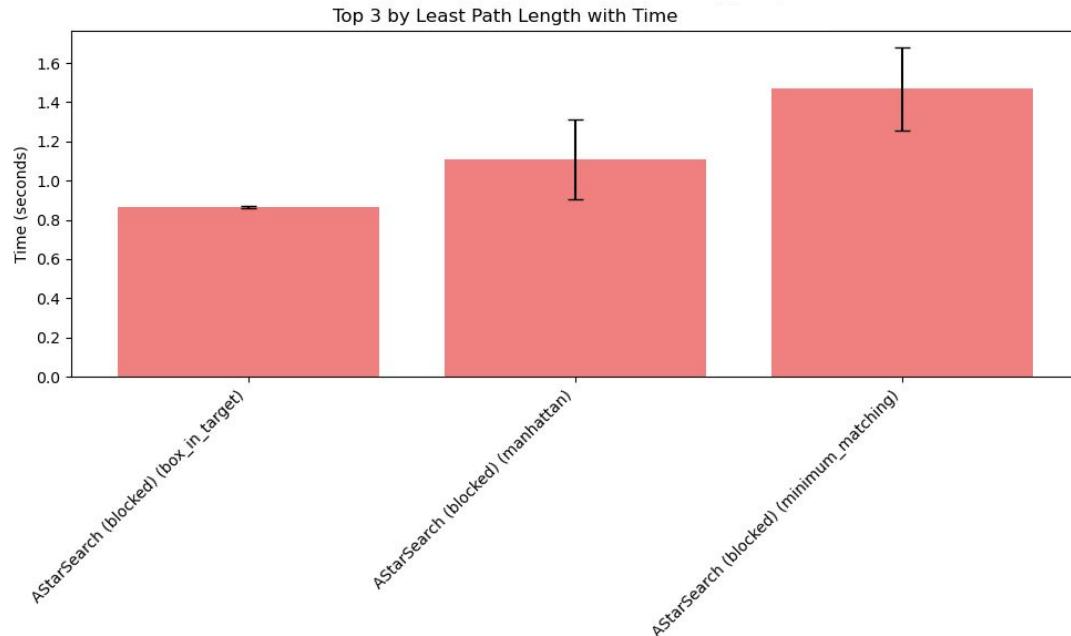
```
#####
# # #
# # #
#   # #
#####
* $ #
# $ * @ #
# #. . #####
# #####
#####
```



Mejor tiempo para el camino óptimo

- **Método:** A* Search
- **Heurística:** blocked + box in target
- **Tiempo:** 0.8645 ± 0.0067 s
- **Costo de la solución:** 121

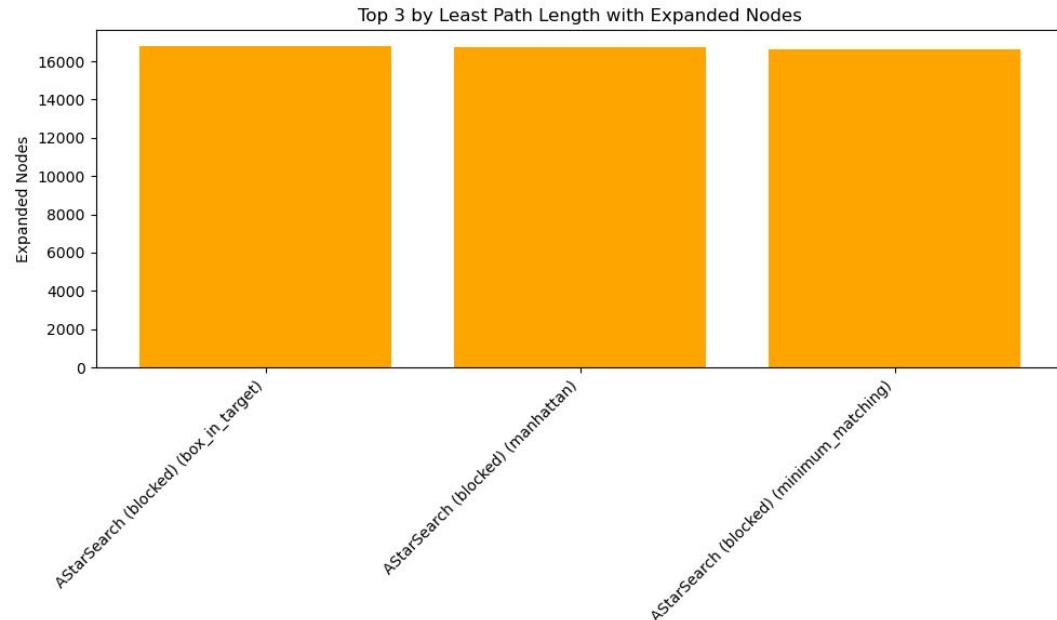
```
#####
# # #
# # #
# # #
##### * $ #
# $ * @ #
# #. . #####
# #####
#####
```



Mejor espacio para el camino óptimo

- **Método:** A* Search
- **Heurística:** blocked + minimum matching
- **Tiempo:** 16598 bloques expandidos
- **Costo de la solución:** 121

```
####  
#### #  
# # # #  
#### * $ #  
# $ * @ #  
# #. .###  
# ####  
####
```



Mapas *Hard*

sokoAll

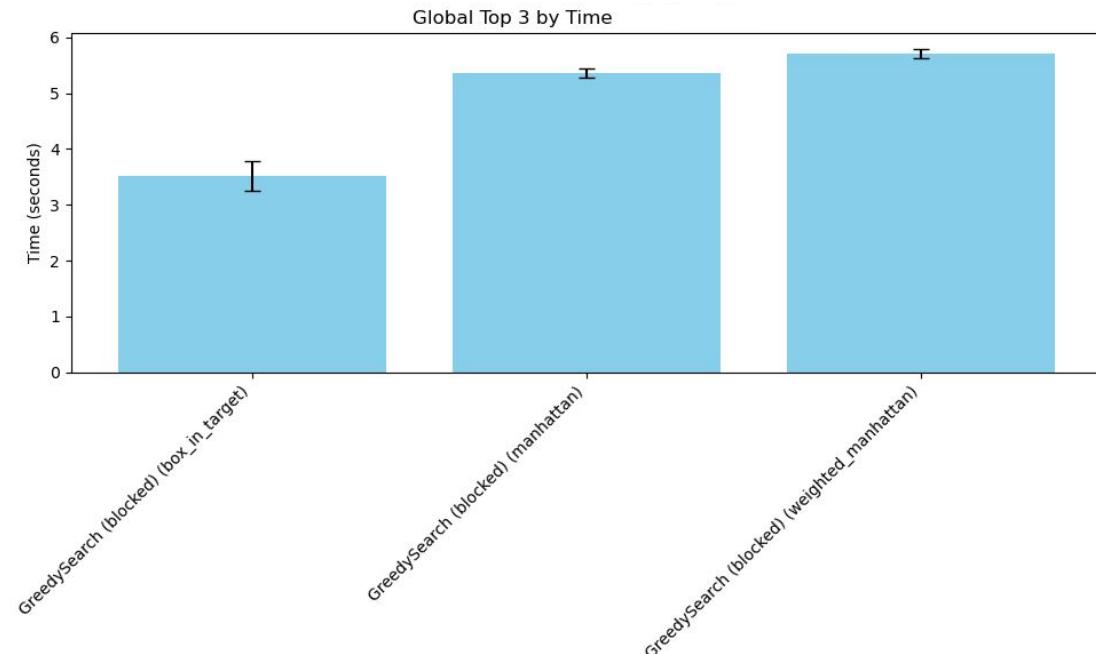
- **5 cajas**
- **espacio escaso, sobre todo cerca de los objetivos**
- **muchas obstaculizaciones**

```
#####
## .#####
##@.$  #
#.$.#  #
##.$#  #
#.$.## $#
#      ##
#  ##  ##
#####
```

Mejor tiempo

- **Método:** Global Greedy Search
- **Heurística:** blocked + box in target
- **Tiempo:** 3.5123 ± 0.2674 s
- **Costo de la solución:** 336

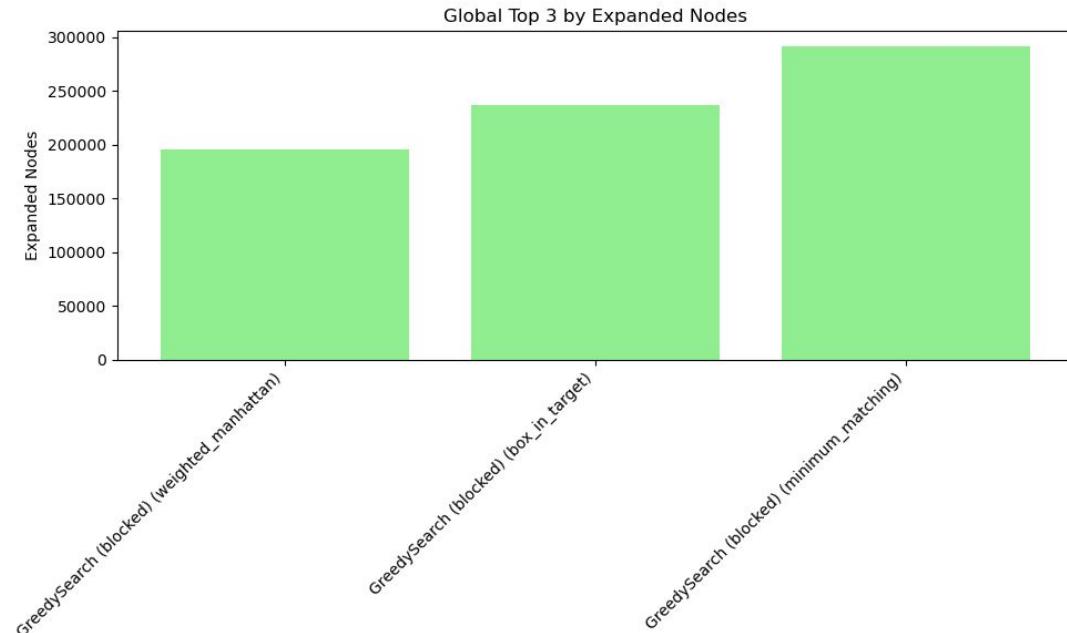
```
#####  
## . #####  
##@.$ #  
# . $ # #  
##. $ # #  
. $## $ #  
# ## ##  
#### ####
```



Mejor espacio

- **Método:** Global Greedy Search
- **Heurística:** blocked + weighted manhattan
- **Tiempo:** 236273 bloques expandidos
- **Costo de la solución:** 320

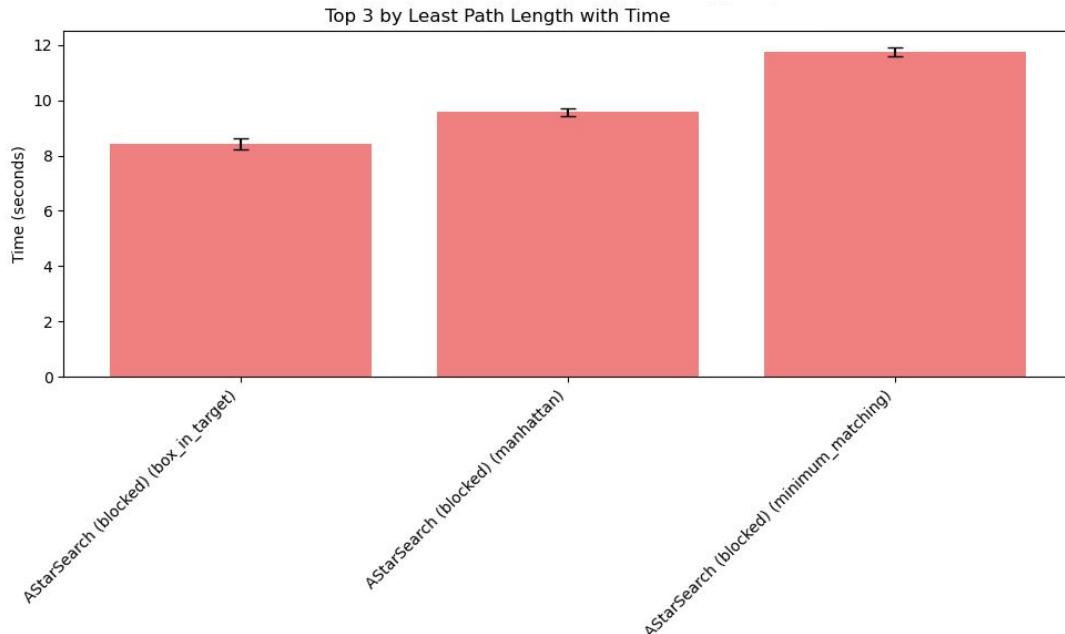
```
####  
## . #####  
##@ . $ #  
# . $ # #  
##. $ # #  
#. $## $ #  
# # # #  
# # # # #
```



Mejor tiempo para el camino óptimo

- **Método:** A* Search
- **Heurística:** blocked + box in target
- **Tiempo:** 8.4275 ± 0.1846 s
- **Costo de la solución:** 194

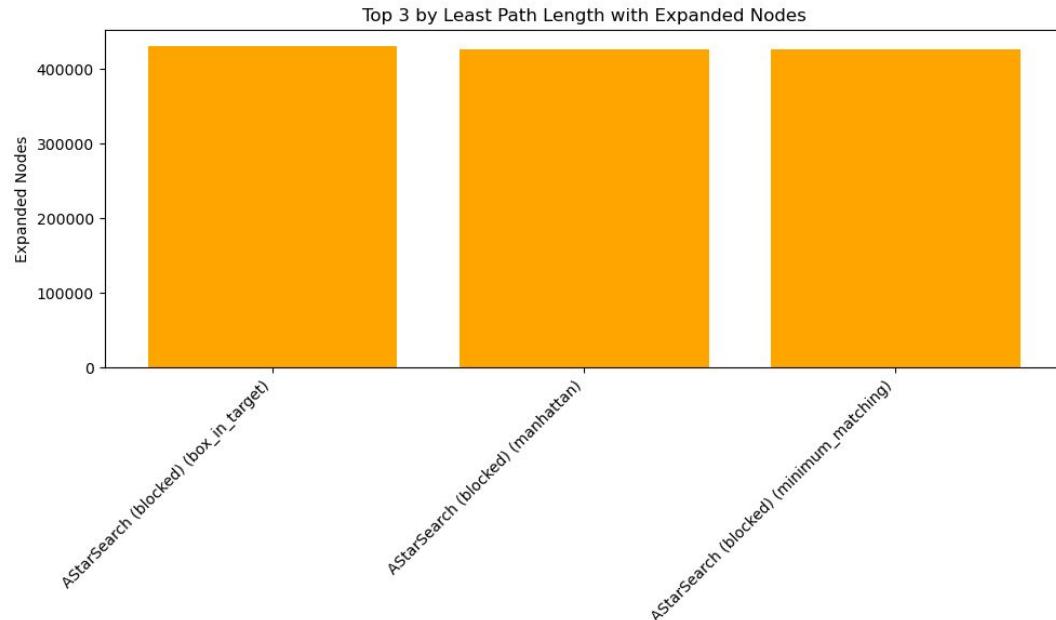
```
####  
## . ####  
##@ . $ #  
# . $ # #  
##. $ # #  
. $## $ #  
# # # #  
# # # # #  
#####
```



Mejor espacio para el camino óptimo

- **Método:** A* Search
- **Heurística:** blocked + minimum matching
- **Tiempo:** 426471 bloques expandidos
- **Costo de la solución:** 194

```
####  
## . ####  
##@. $ #  
# . $ # #  
##. $ # #  
. $## $ #  
# ## ##  
# ## ##
```

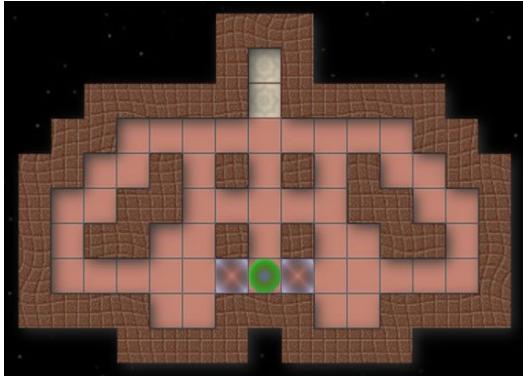


Conclusiones

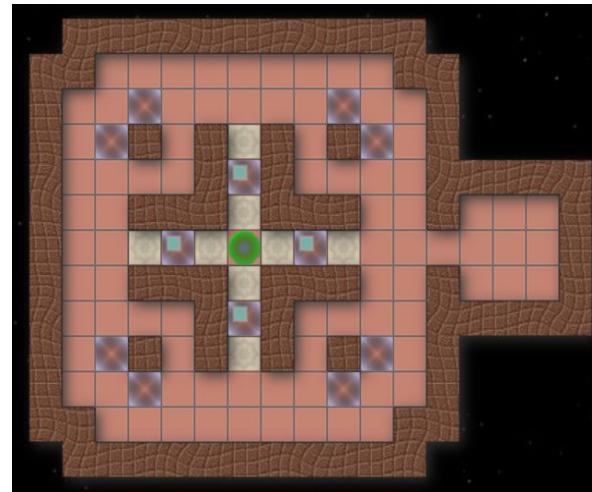
1. Métodos **informados** tienen mejor performance que los métodos **no informados**
2. **A*** genera el mejor camino posible, pero utiliza más tiempo y recursos
Greedy utiliza menos tiempo y recursos, pero, generalmente no encuentra el mejor camino posible
3. La heurística que mejor funcionó para reducir el tiempo y el uso de recursos fue **Blocked**

Anexo - Mapas

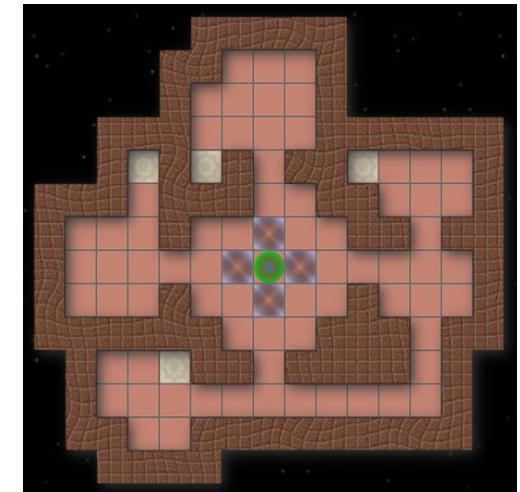
Soko 01



Soko 02

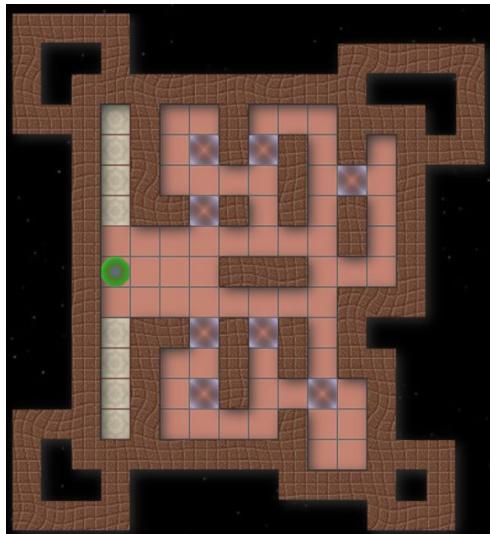


Soko 03



Anexo - Mapas

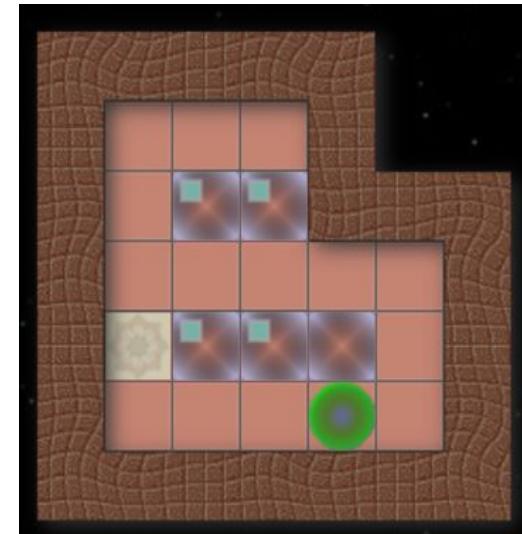
Soko 08



Soko 09

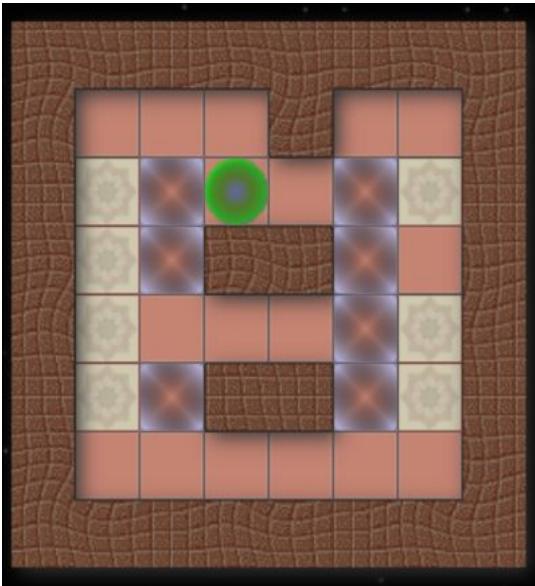


Soko A01

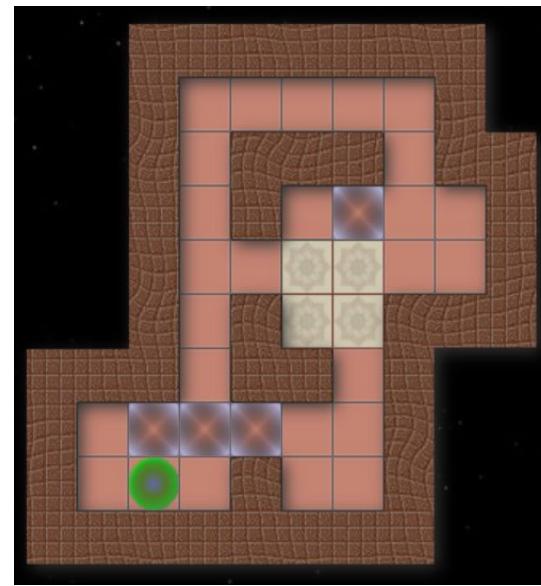


Anexo - Mapas

Soko A02



Soko A03



Soko A04



Anexo - Mapas

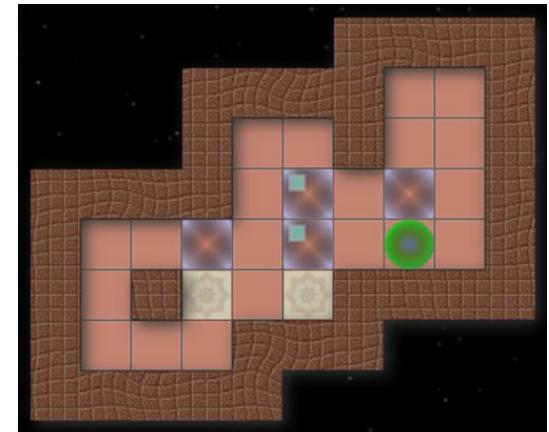
Soko A05



Soko A06



Soko A07

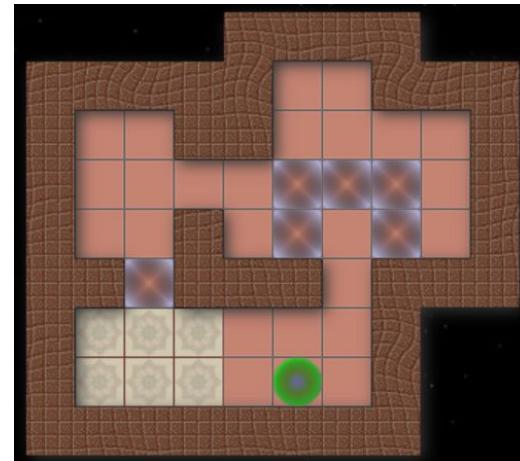


Anexo - Mapas

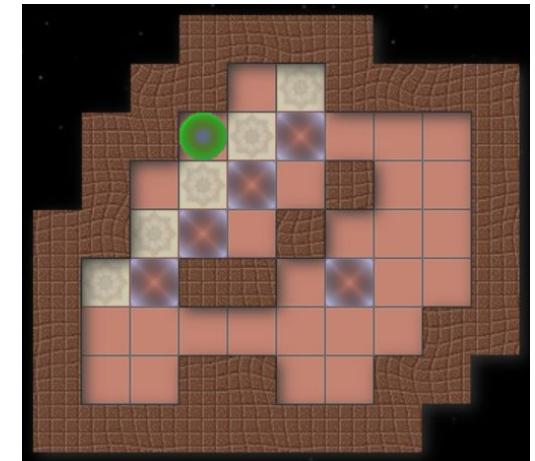
Soko A08



Soko A10

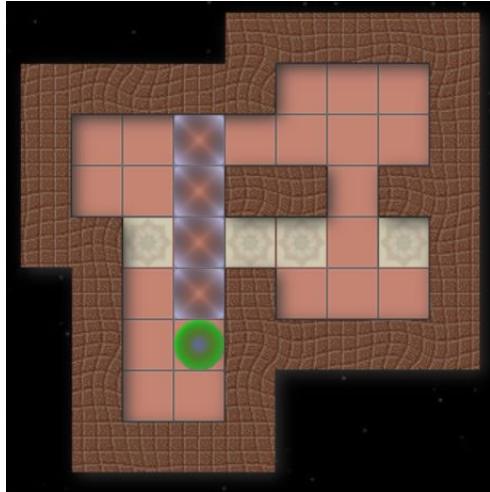


Soko A11

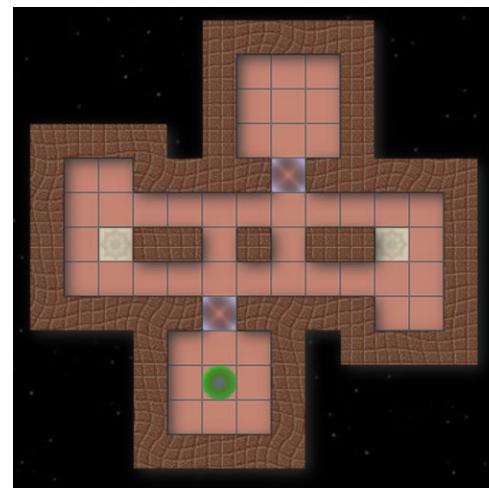


Anexo - Mapas

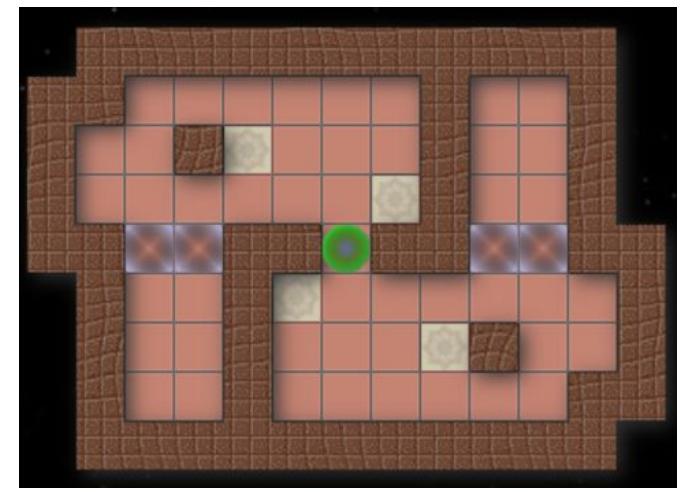
Soko A12



Soko 11



Soko 12



Anexo - Mapas

Soko 00

```
# # # # #
# @ $ . #
# # # # #
```

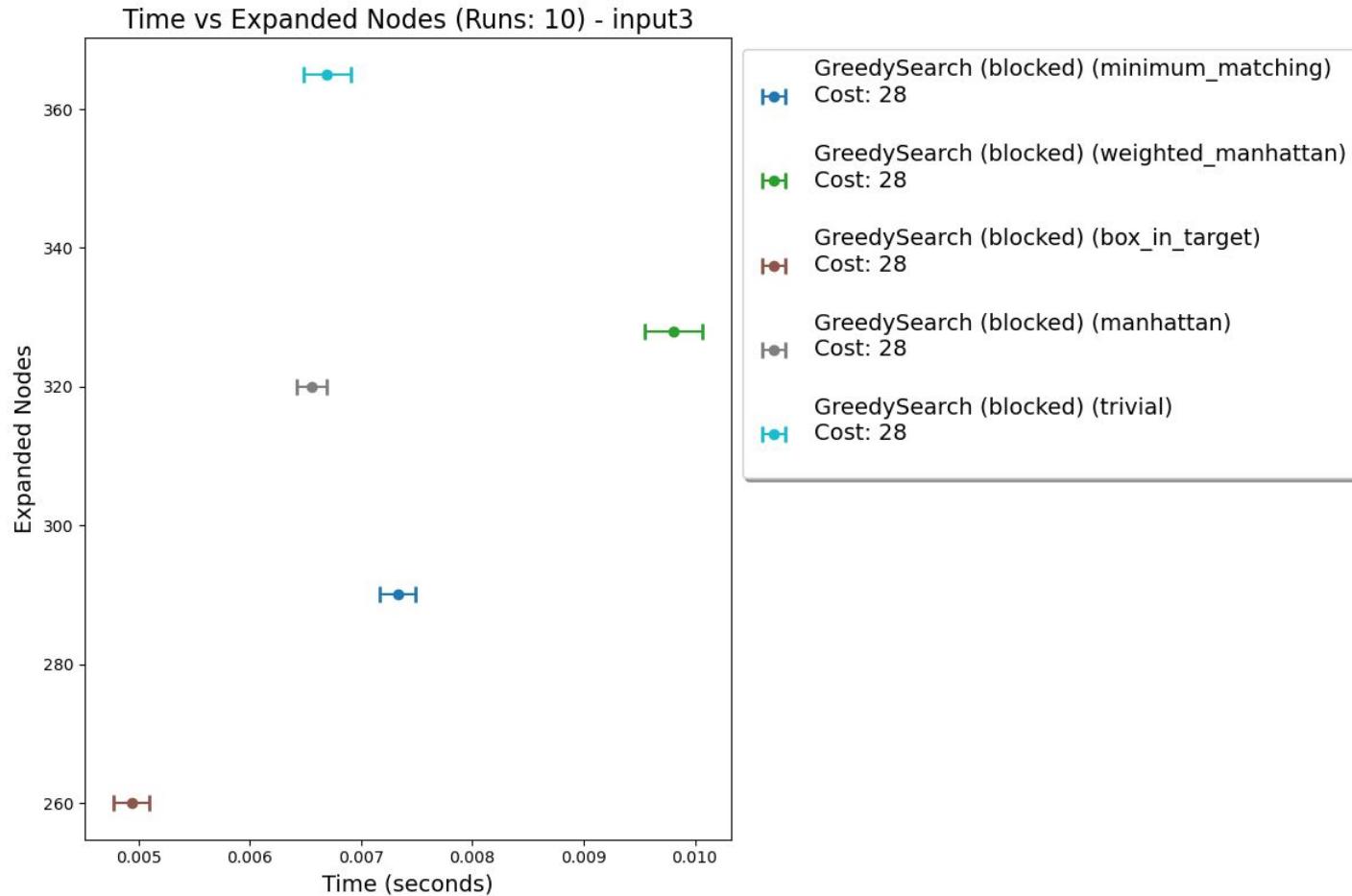
Input 3

```
# # # # # #
# . # @ #
# $$ $ #
# . . #
# # # # # #
```

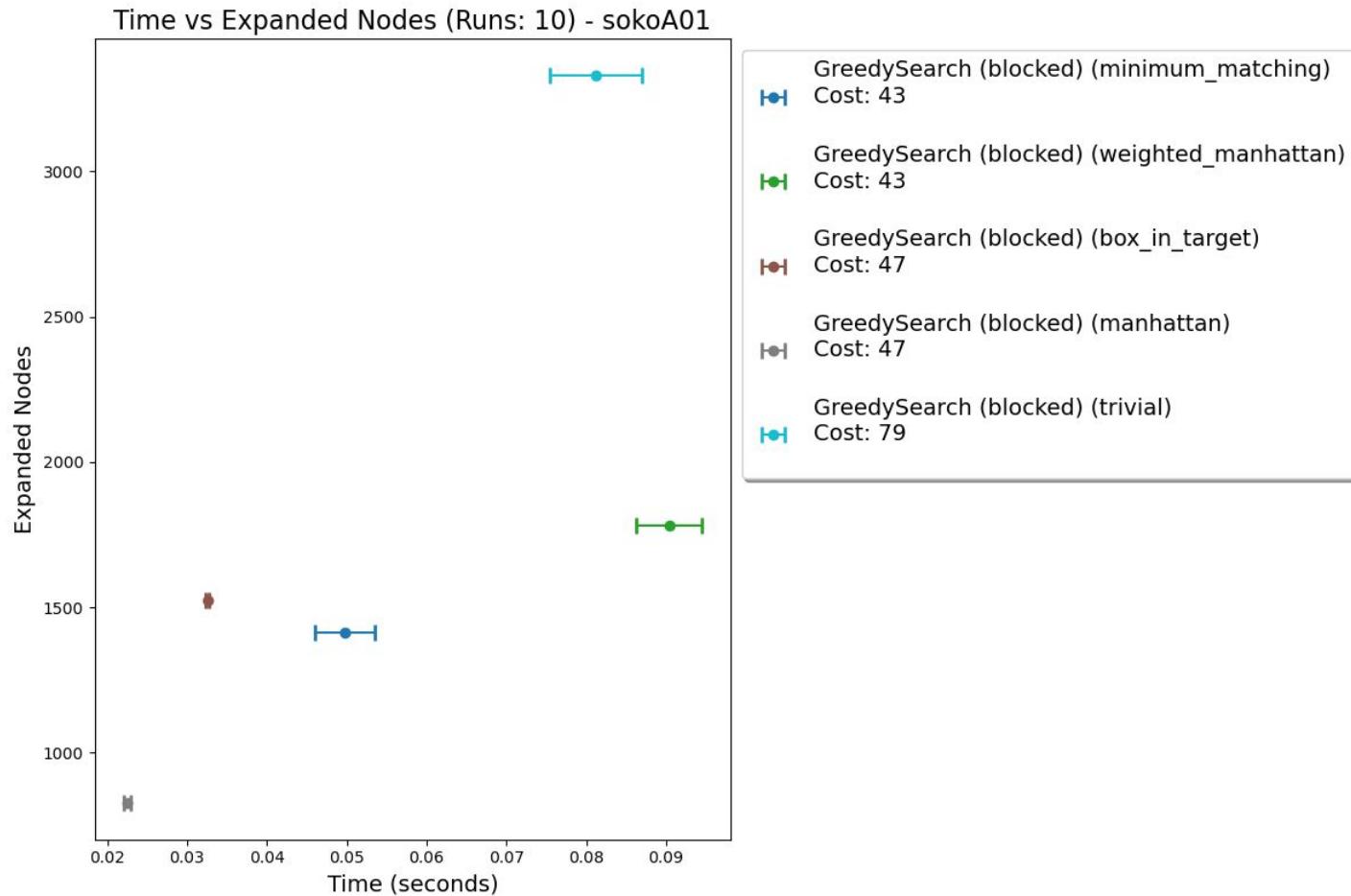
Input 4

```
# # # # # #
# # # # #
# . # #
# . $ $ #
# . $$ #
# . # @ #
# # # # # #
```

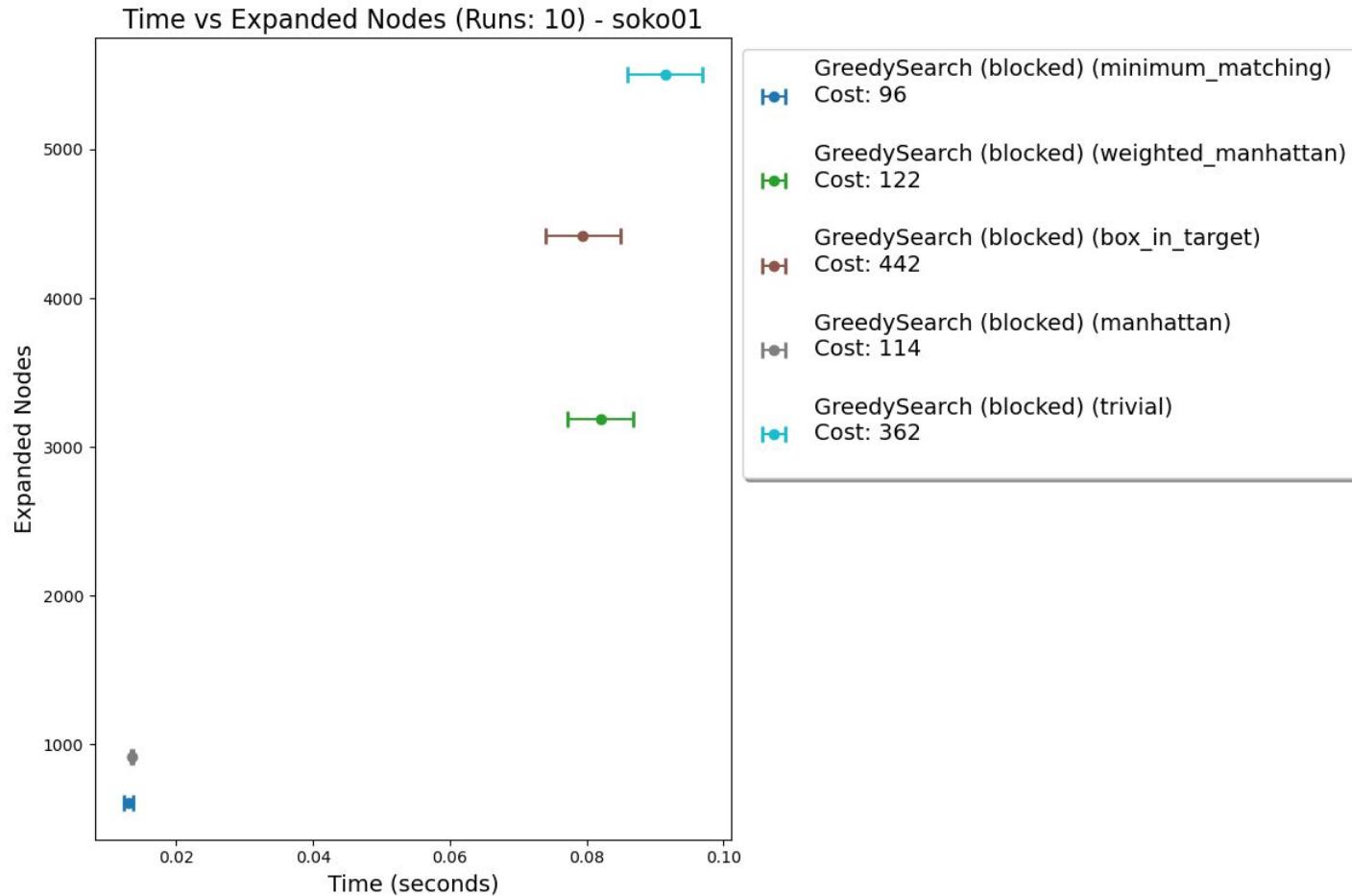
Anexo - Resultados



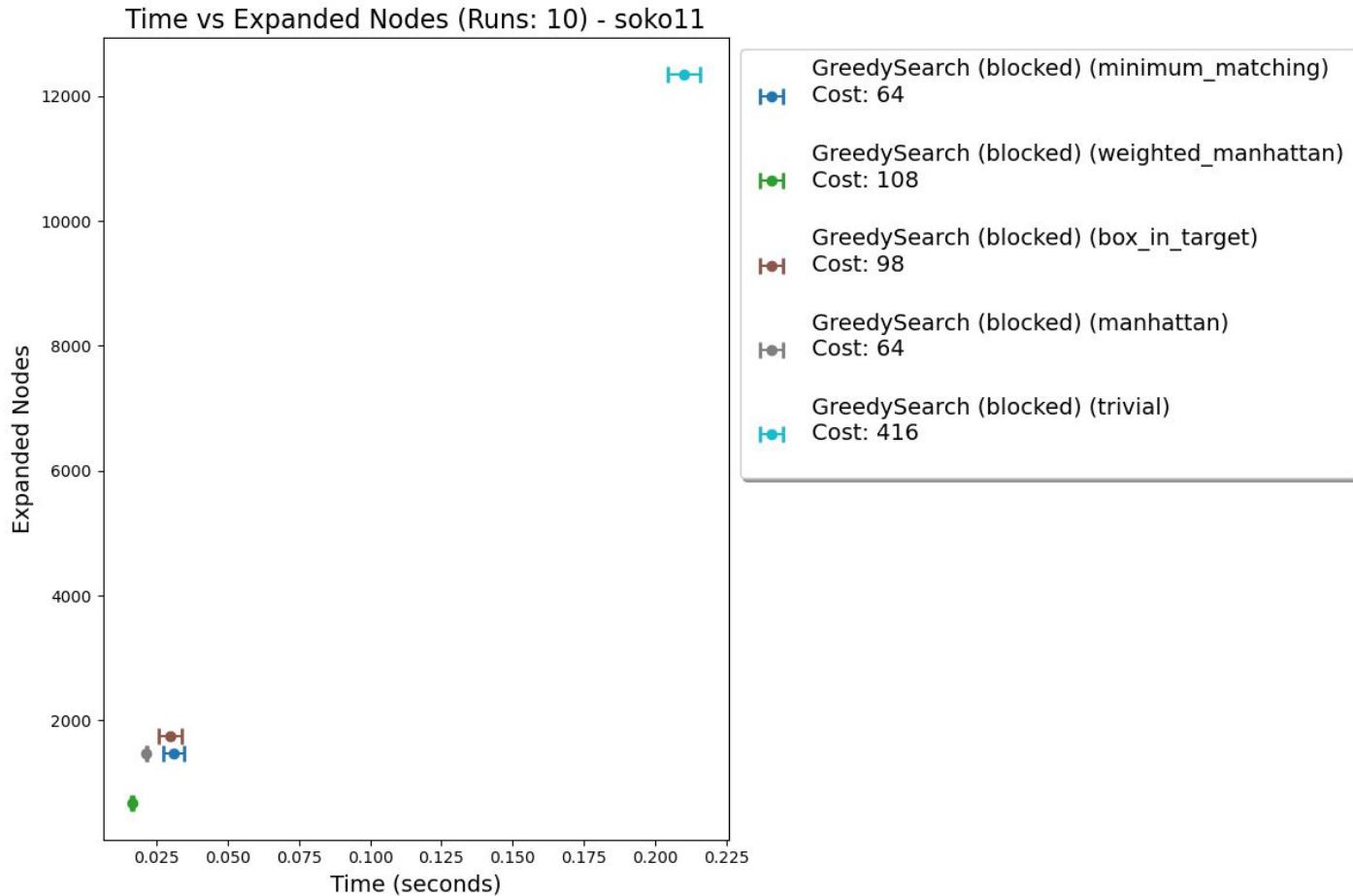
Anexo - Resultados



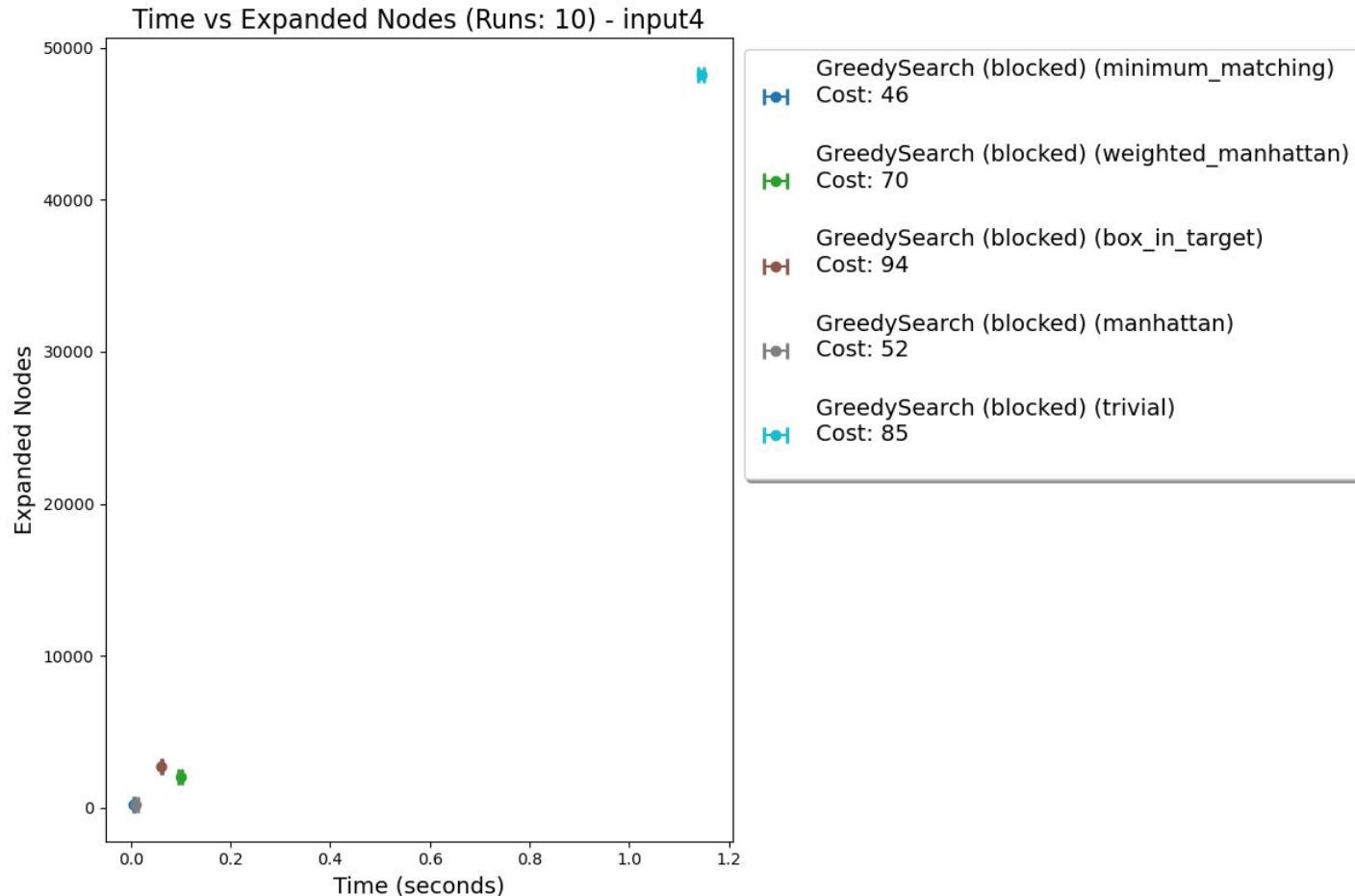
Anexo - Resultados



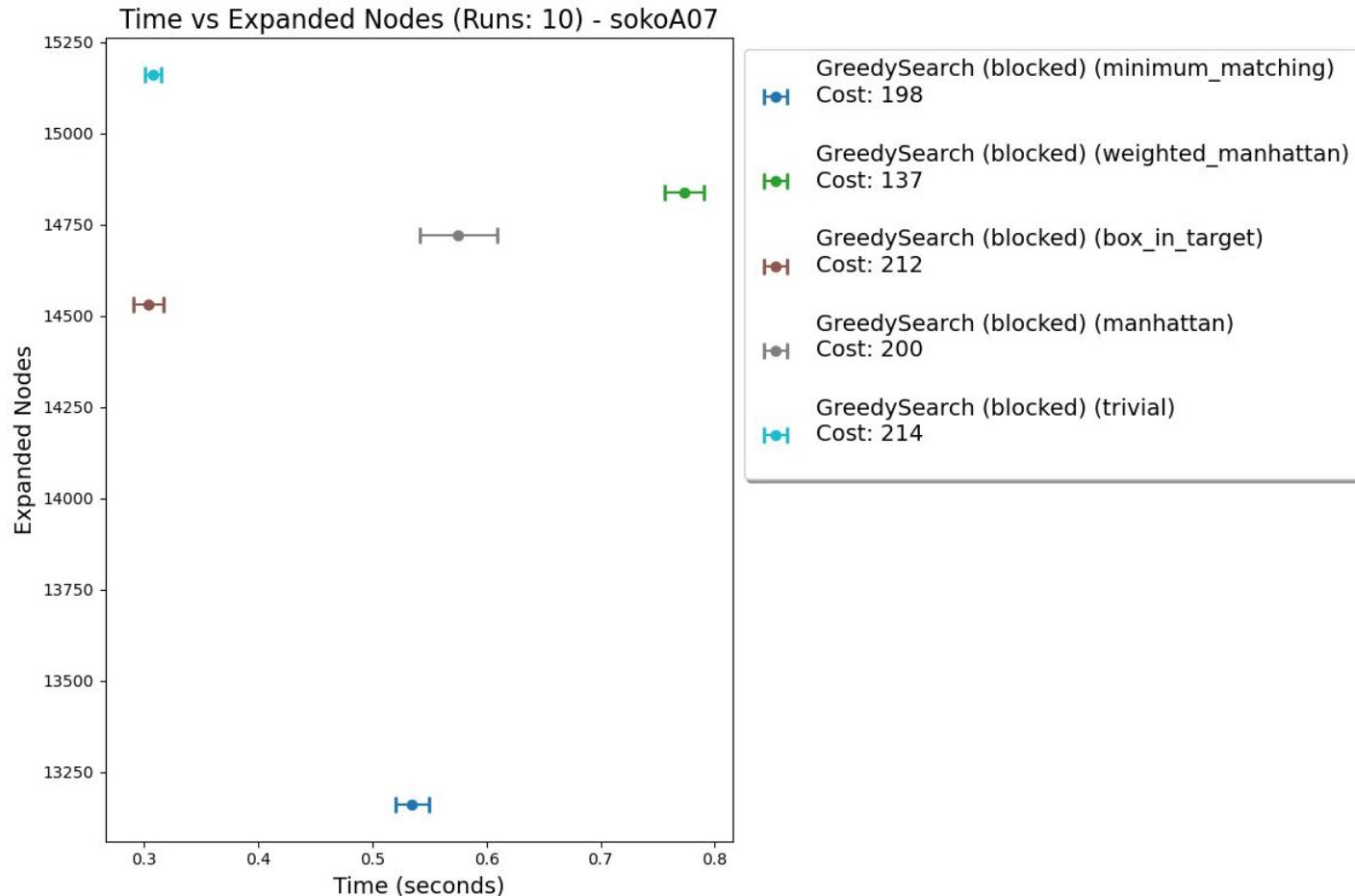
Anexo - Resultados



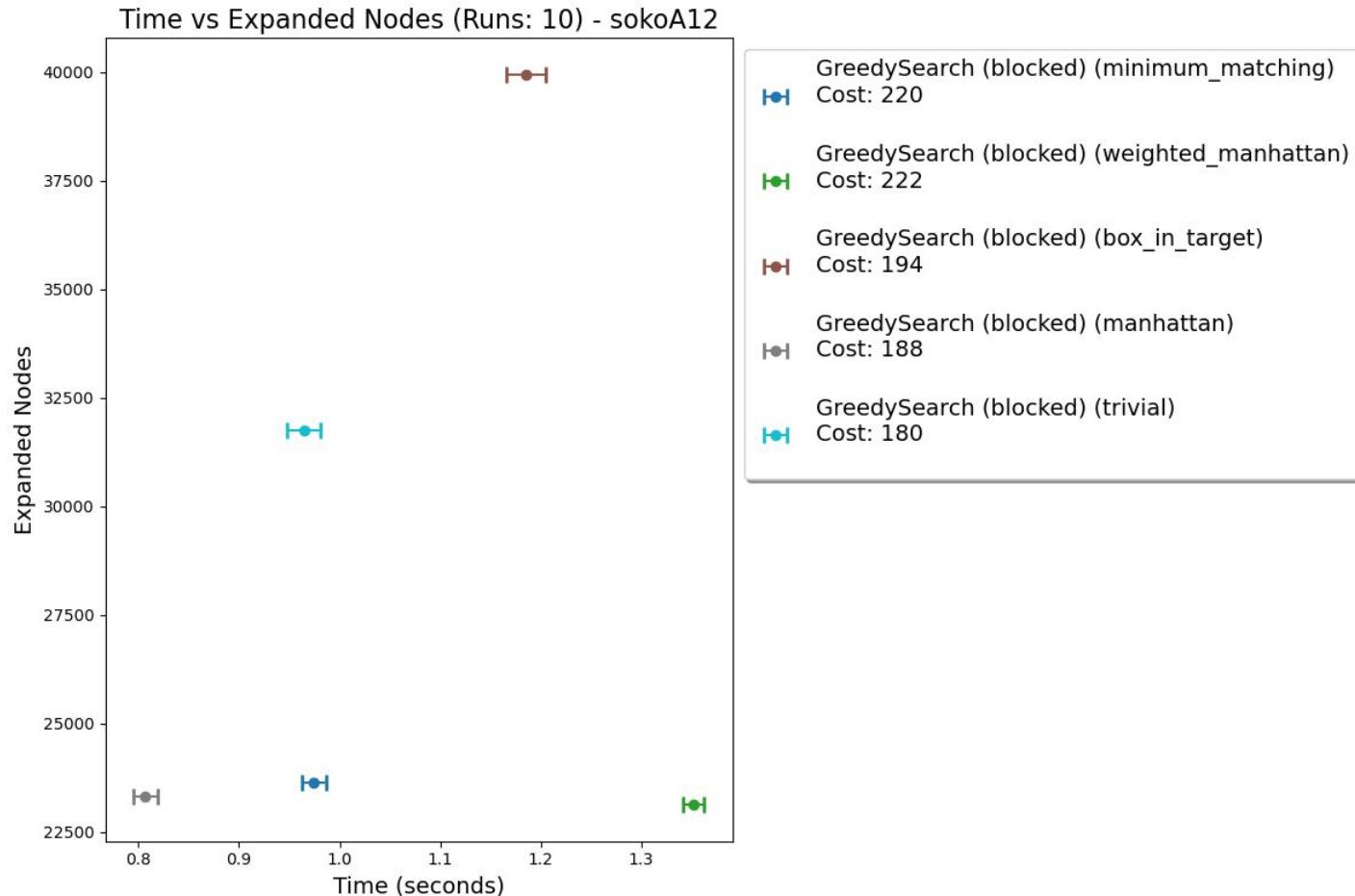
Anexo - Resultados



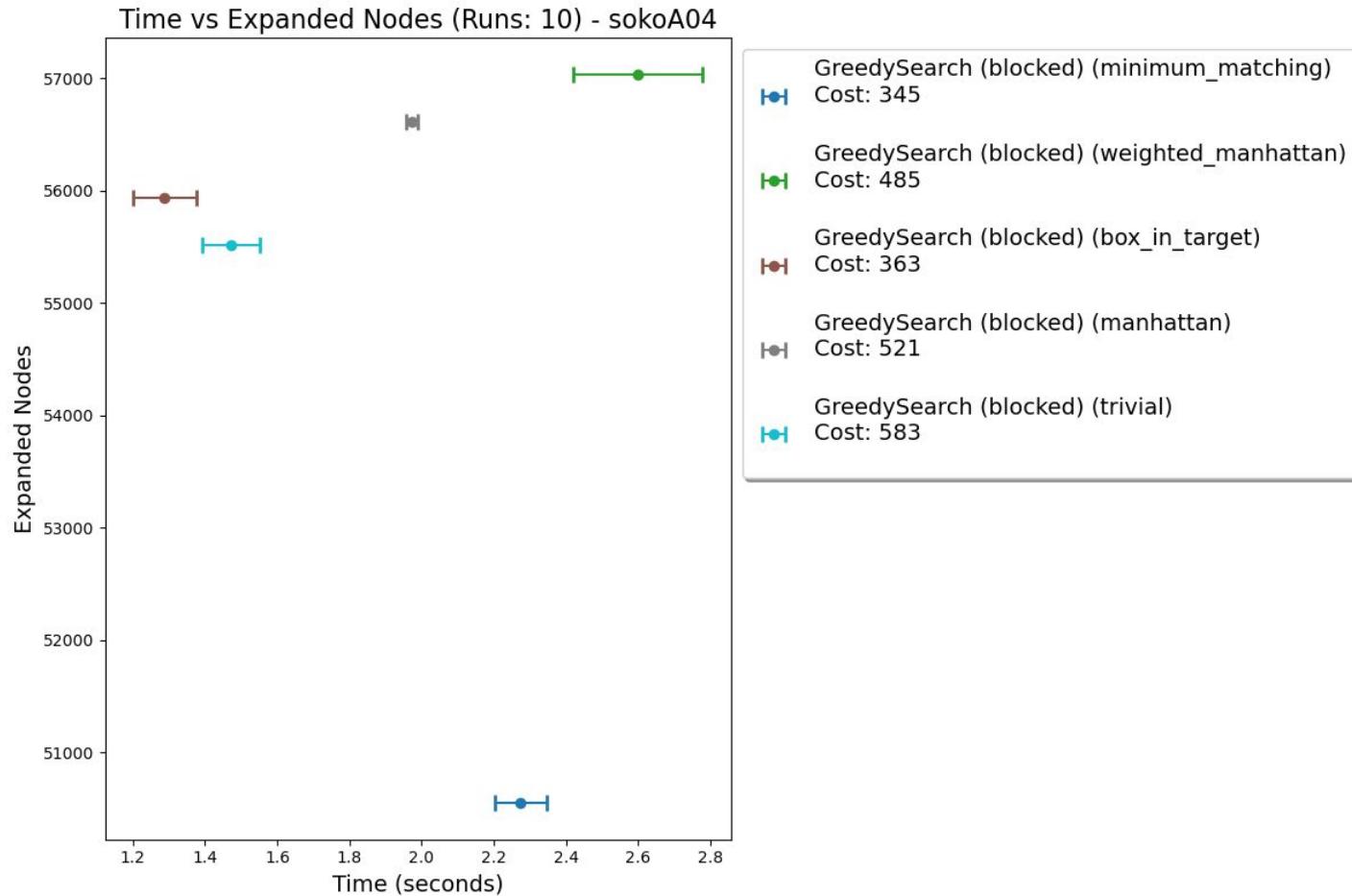
Anexo - Resultados



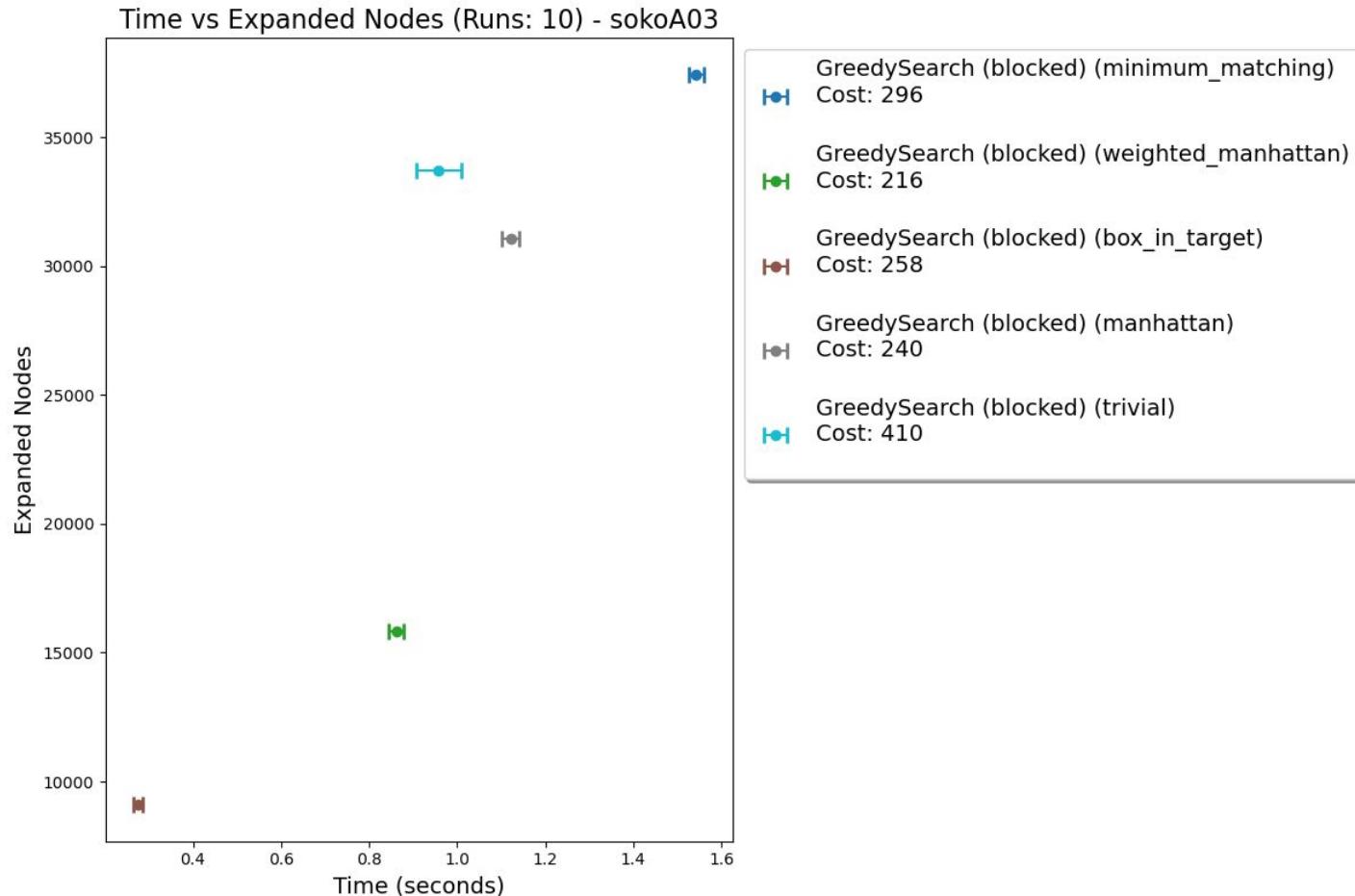
Anexo - Resultados



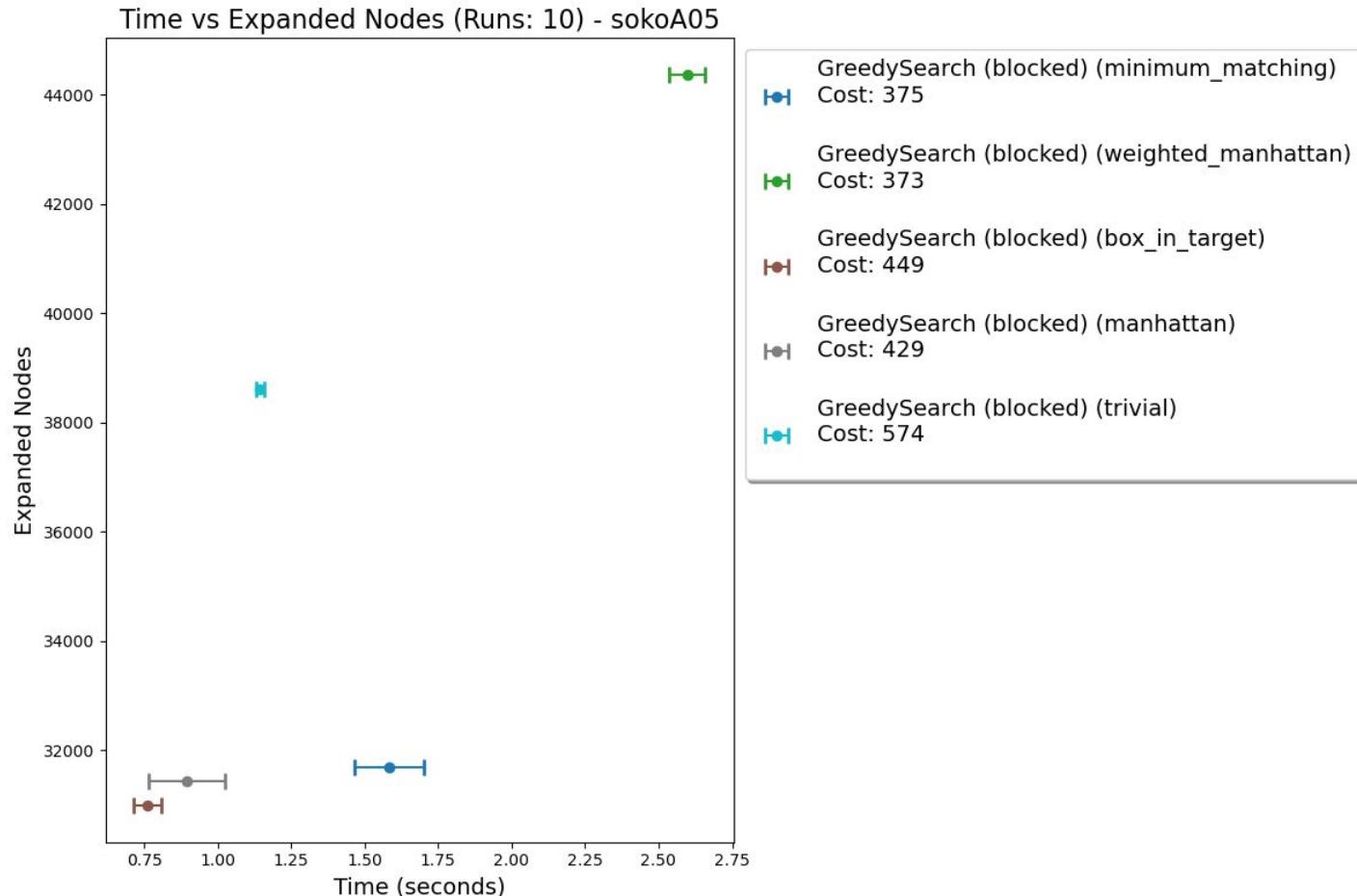
Anexo - Resultados



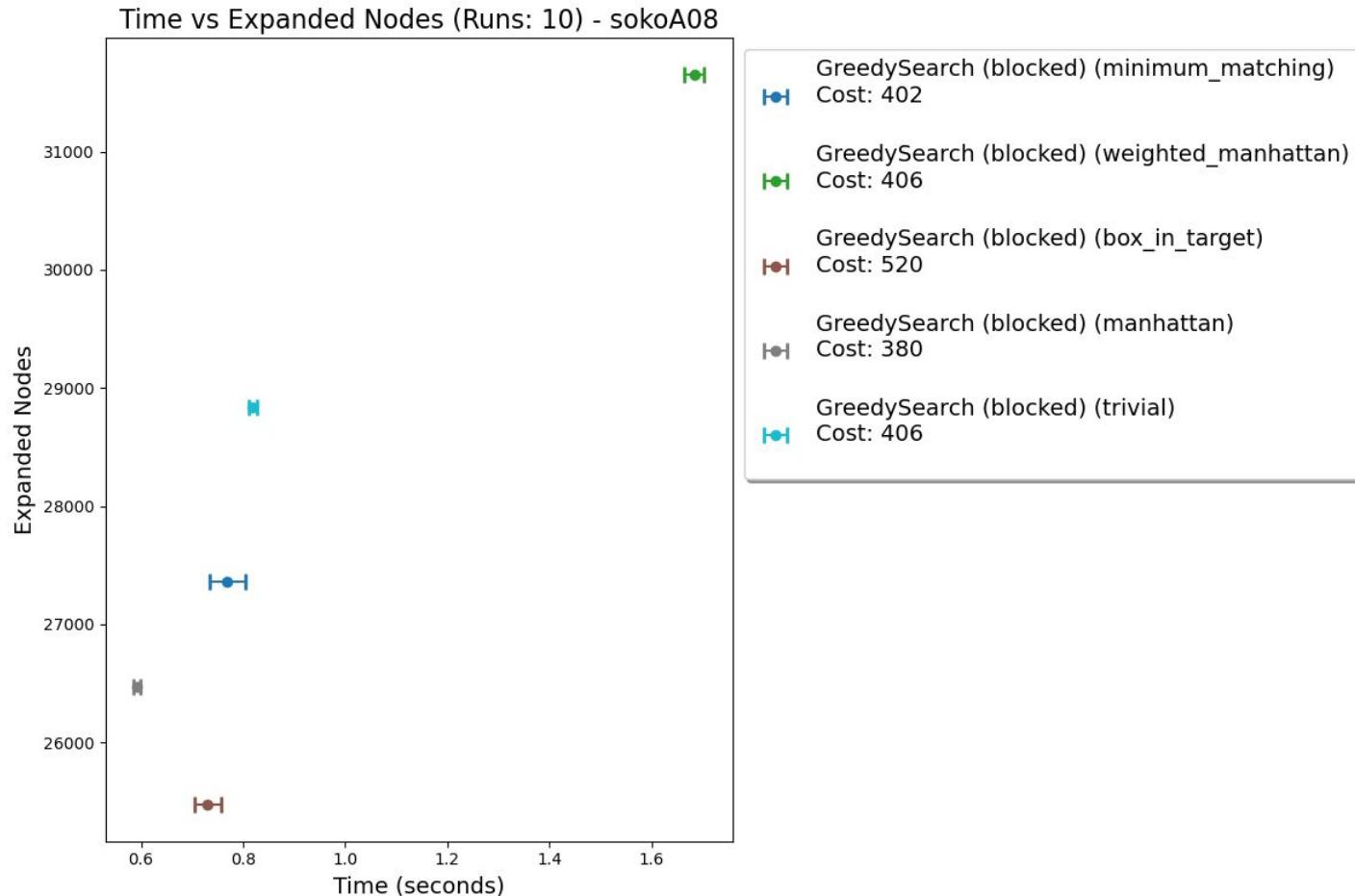
Anexo - Resultados



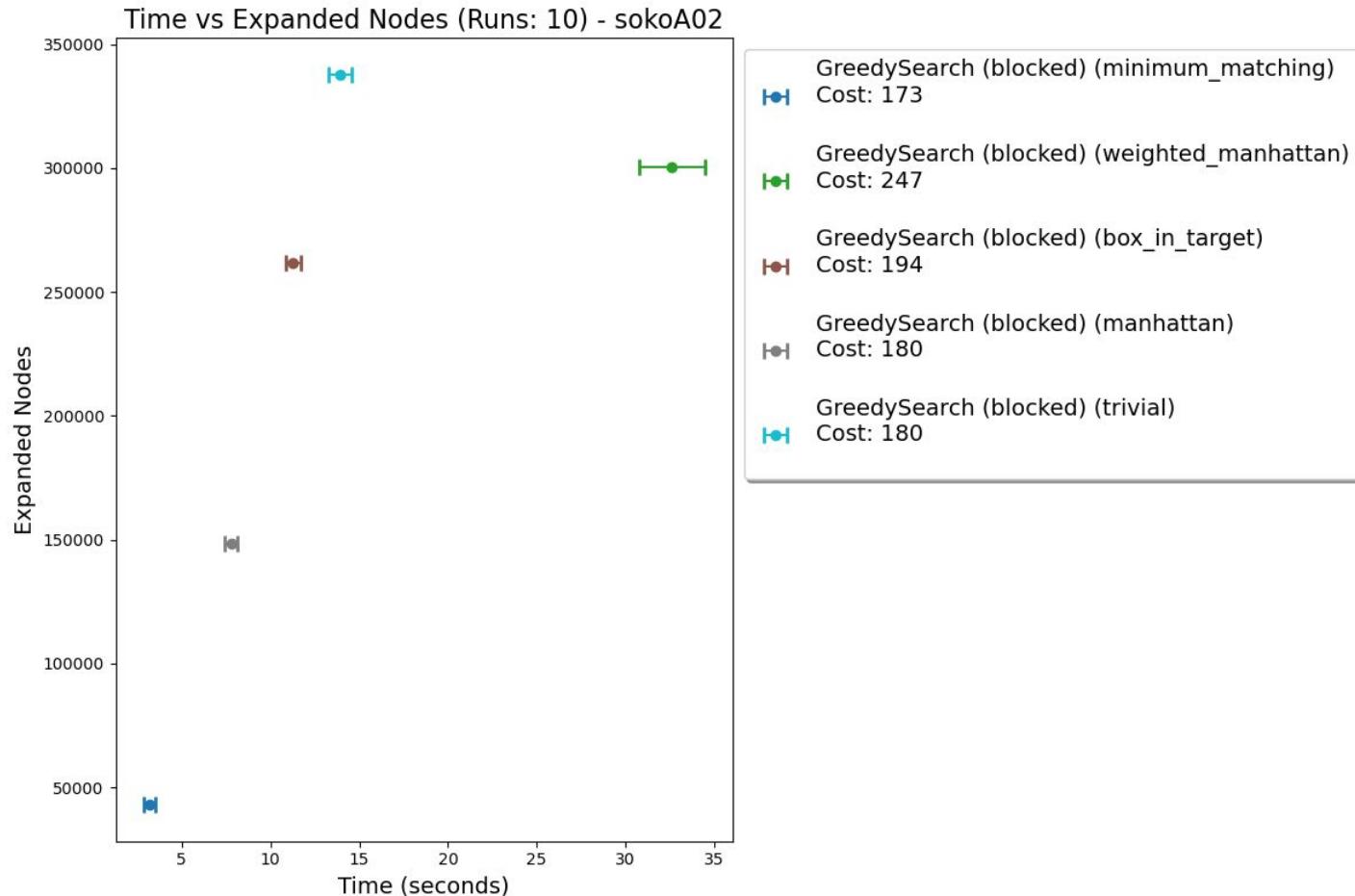
Anexo - Resultados



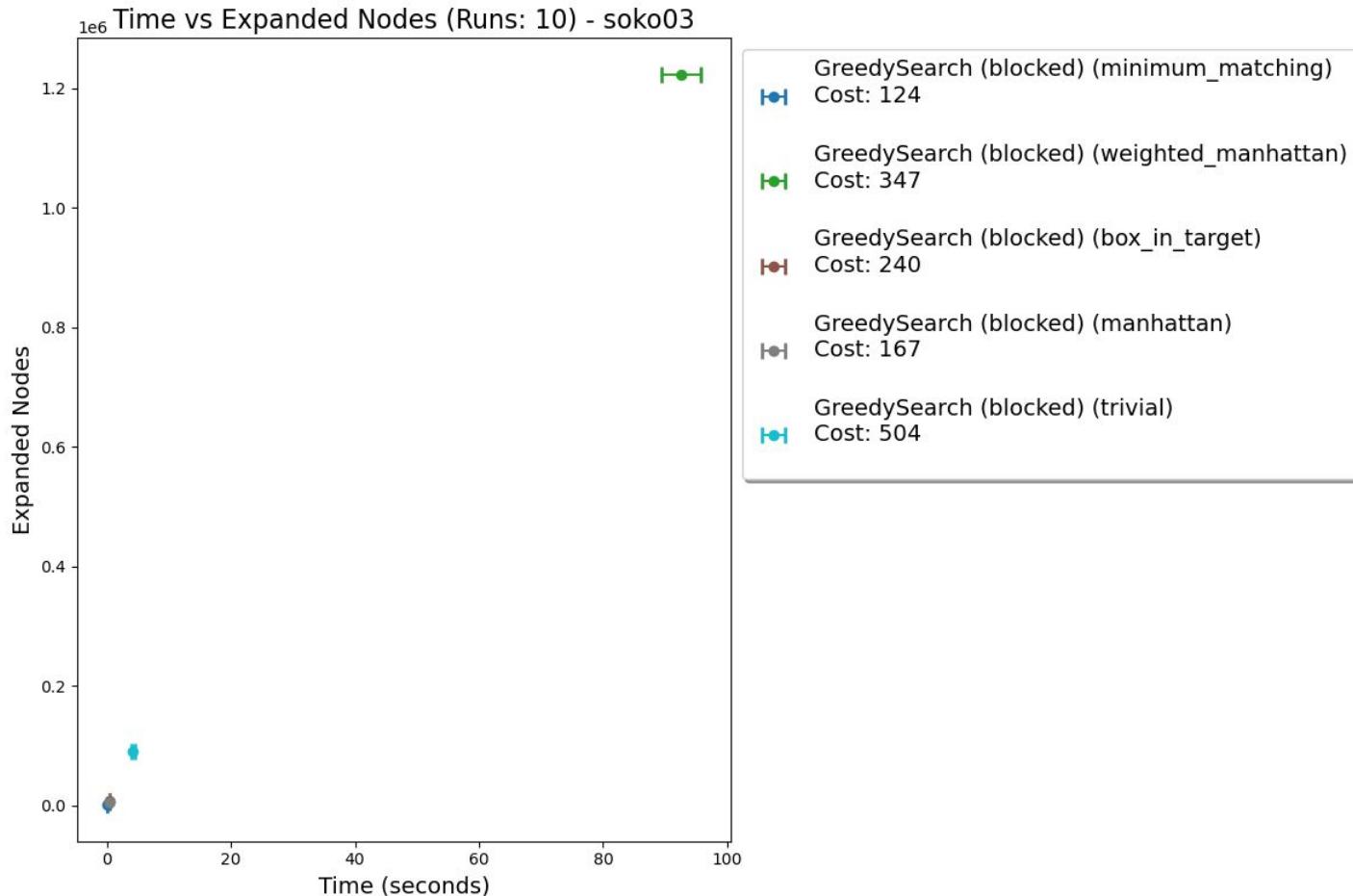
Anexo - Resultados



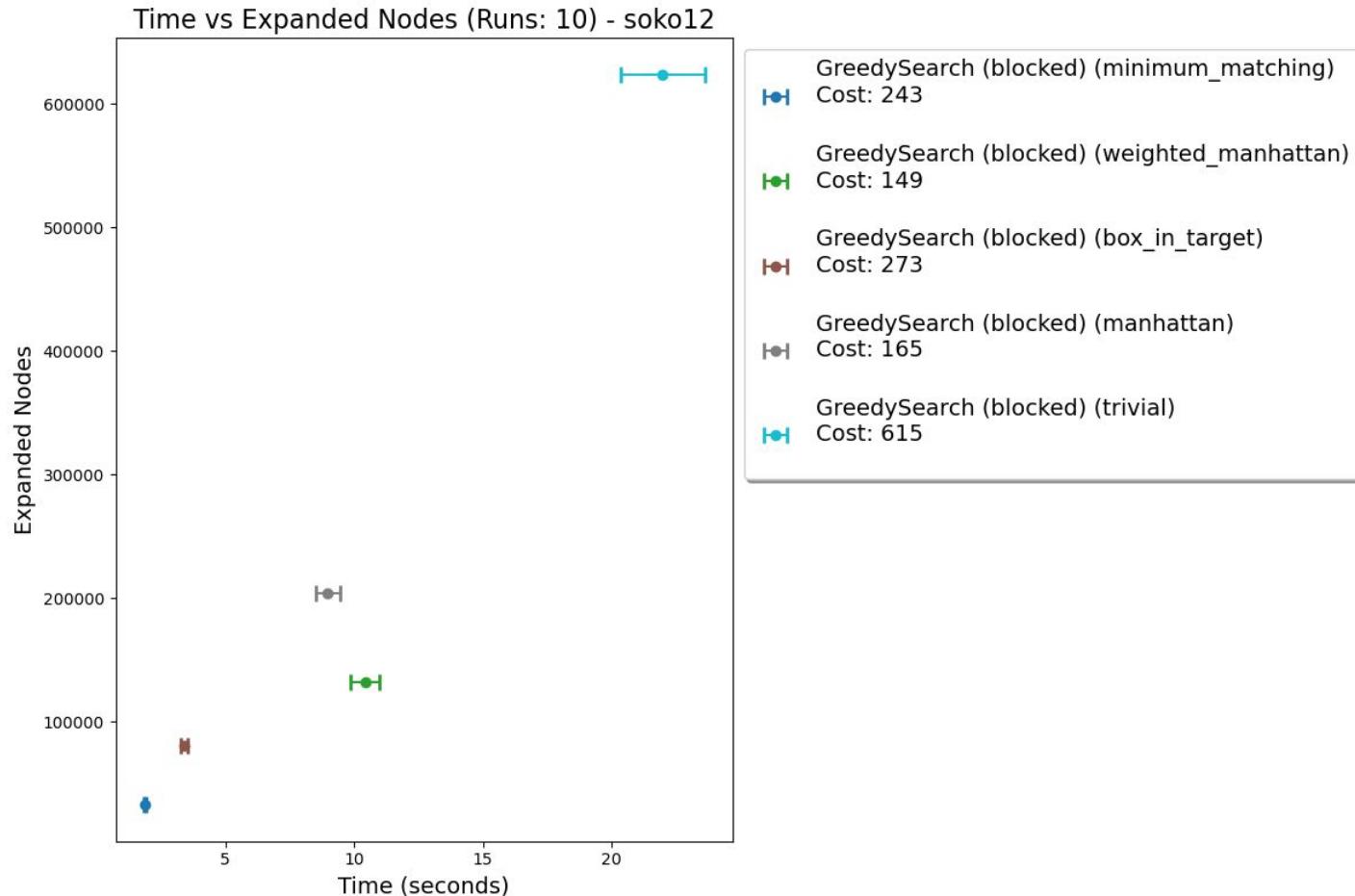
Anexo - Resultados



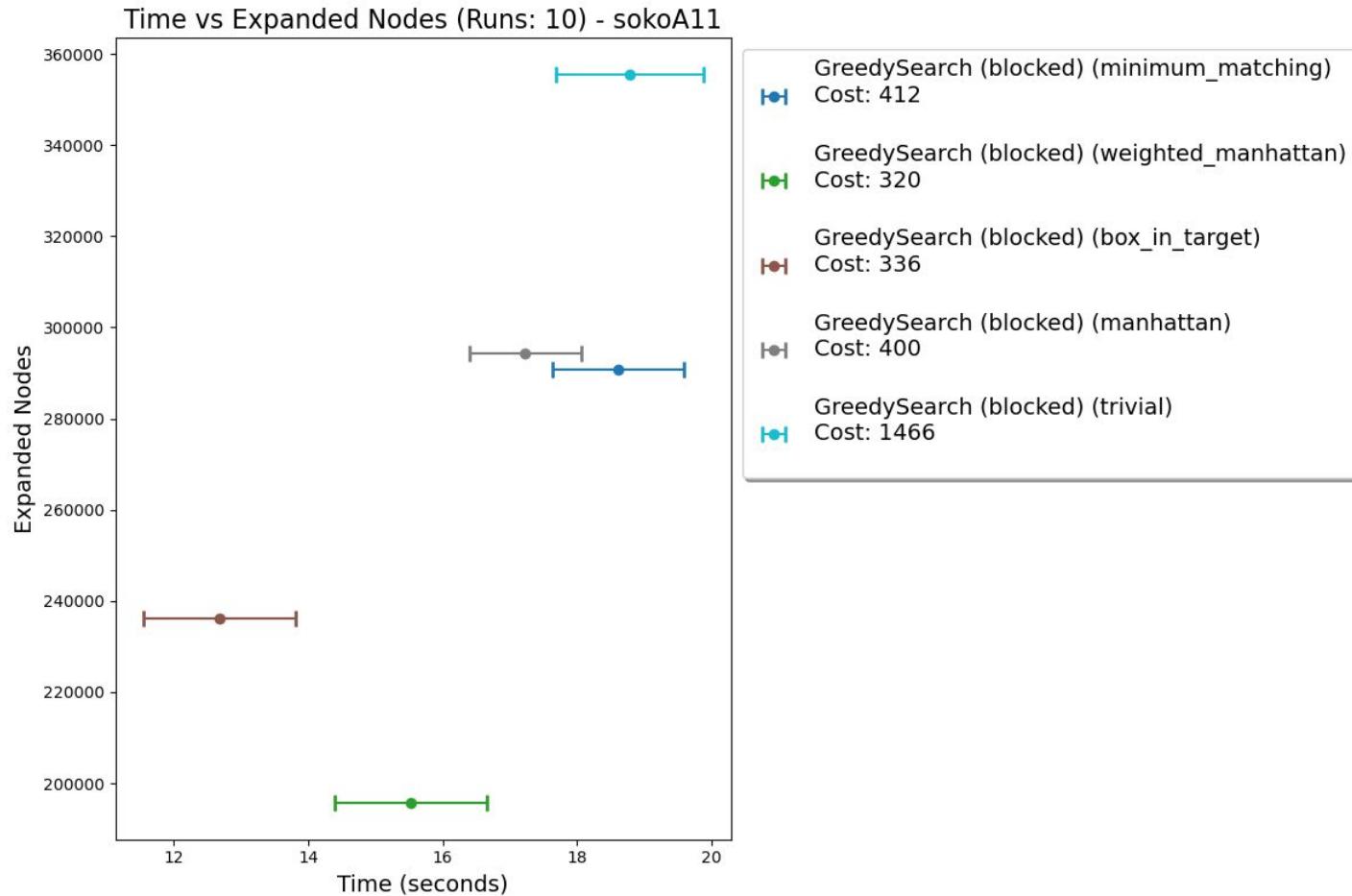
Anexo - Resultados



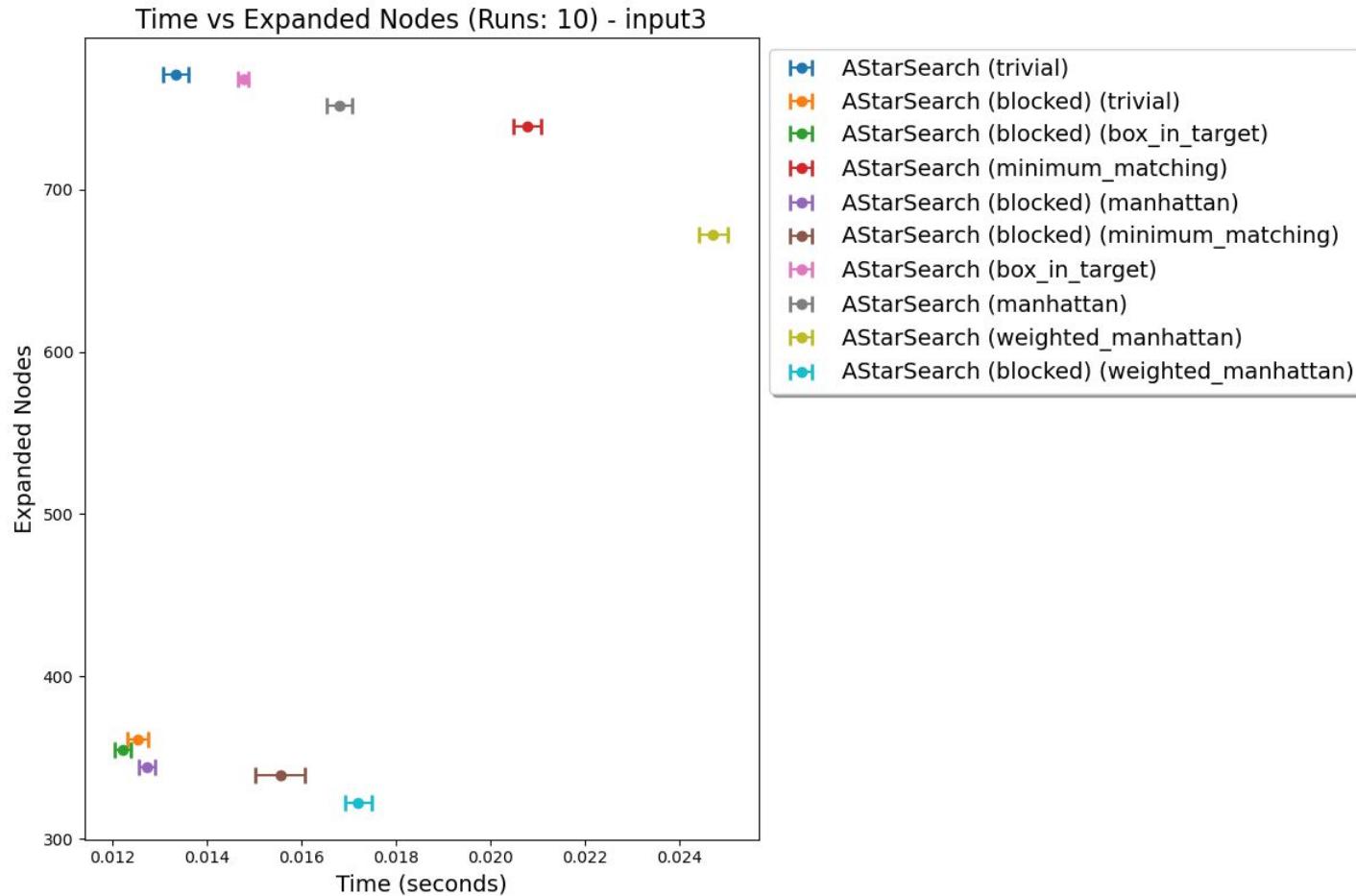
Anexo - Resultados



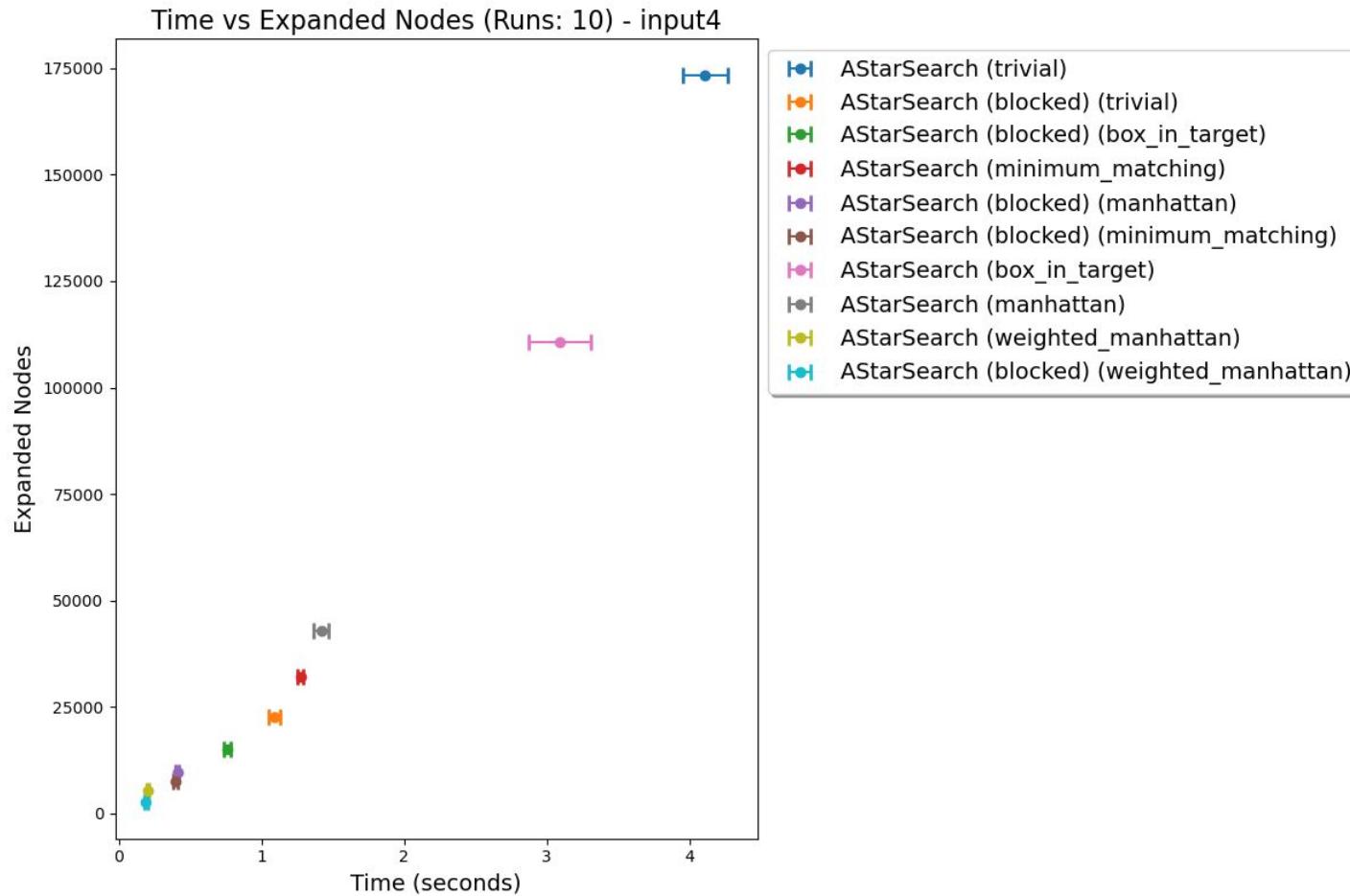
Anexo - Resultados



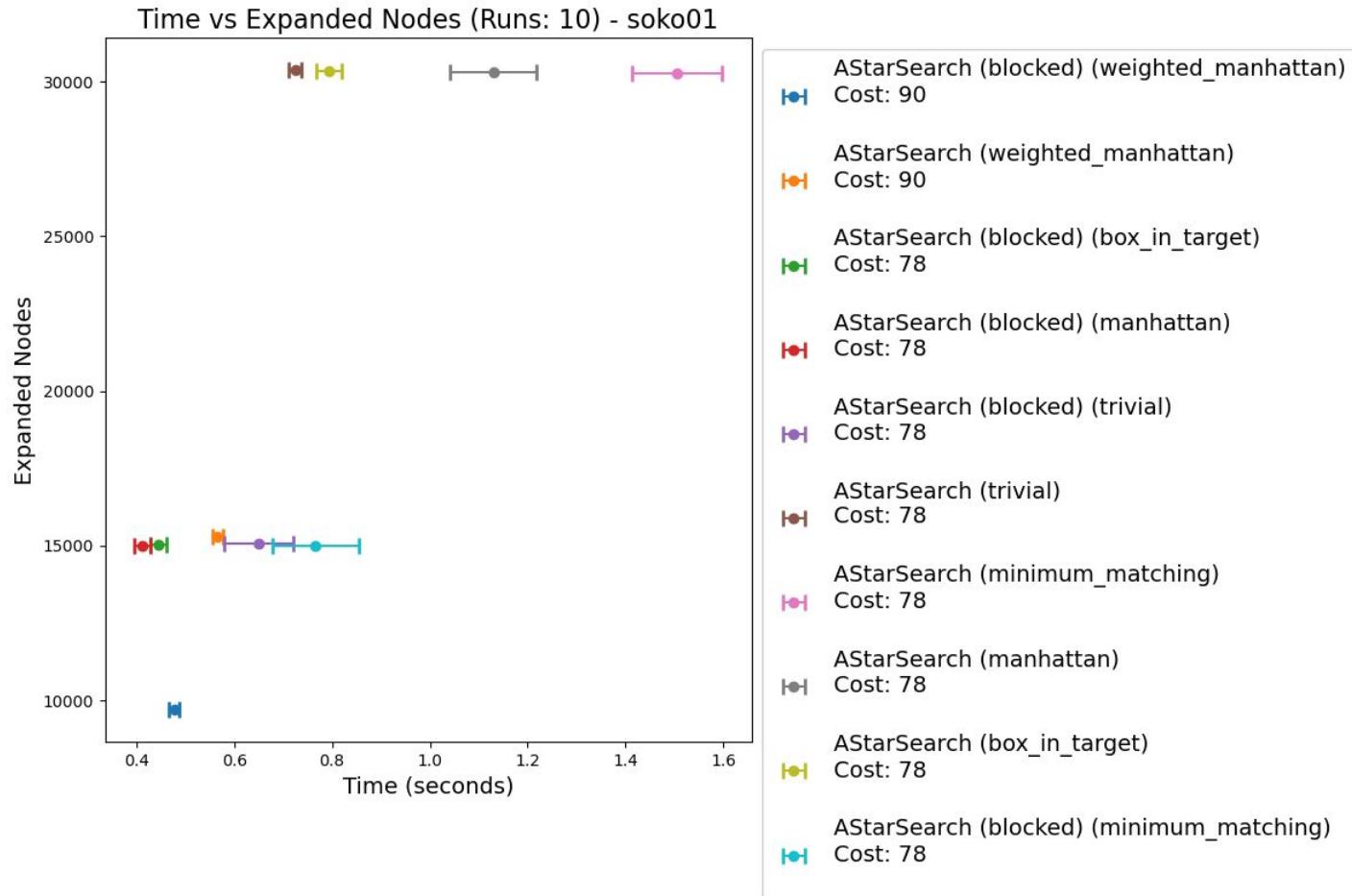
Anexo - Resultados



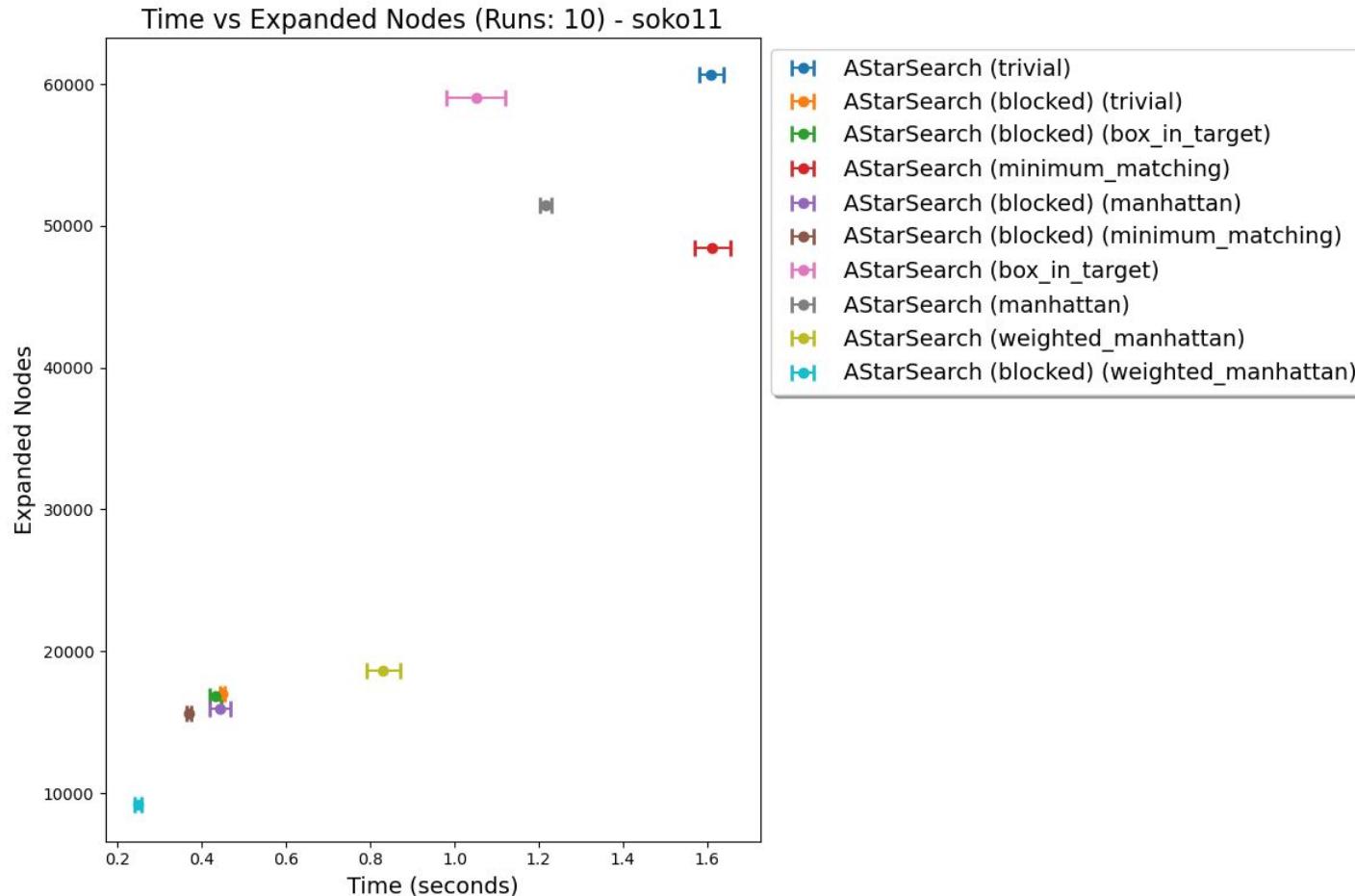
Anexo - Resultados



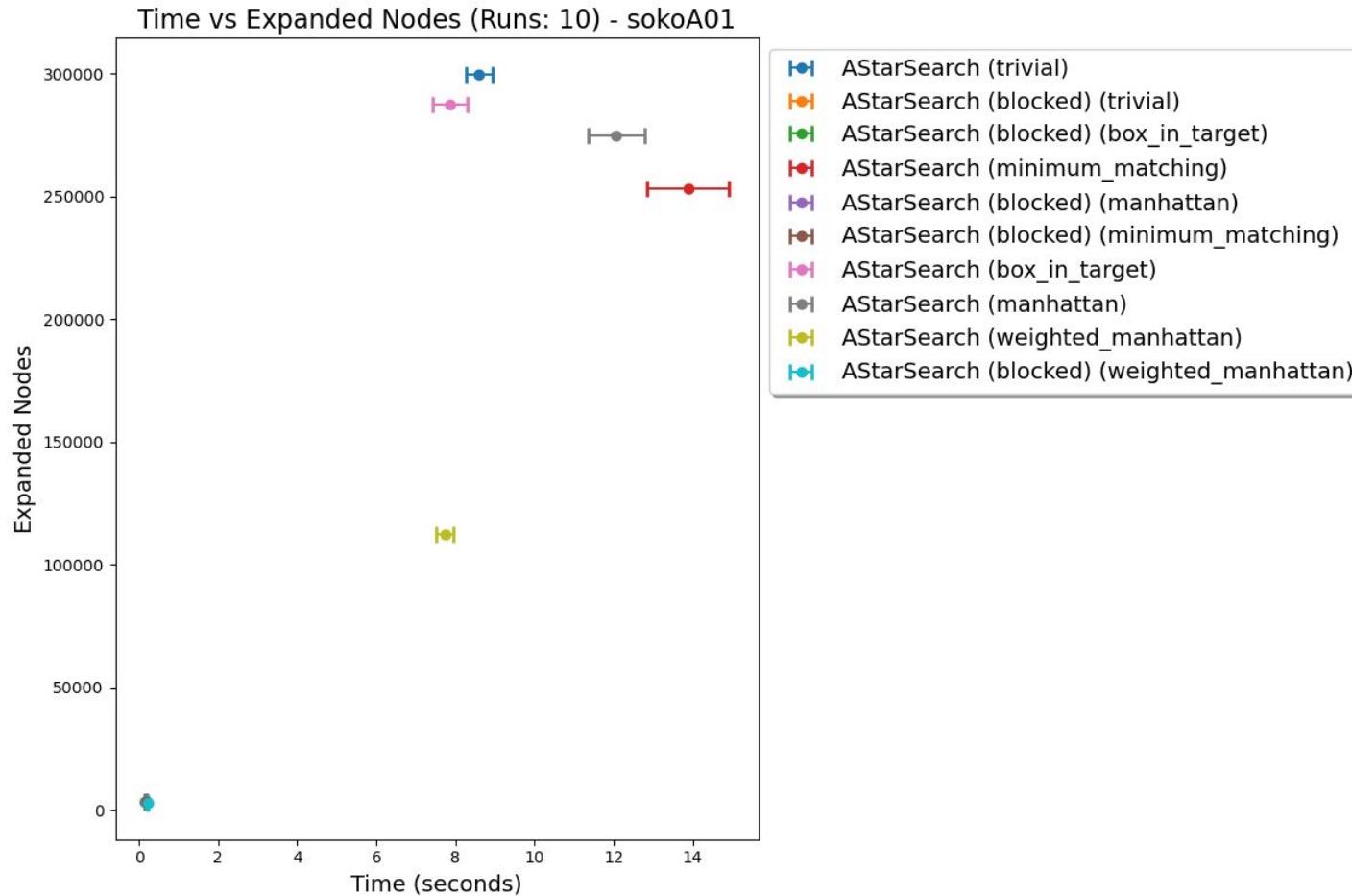
Anexo - Resultados



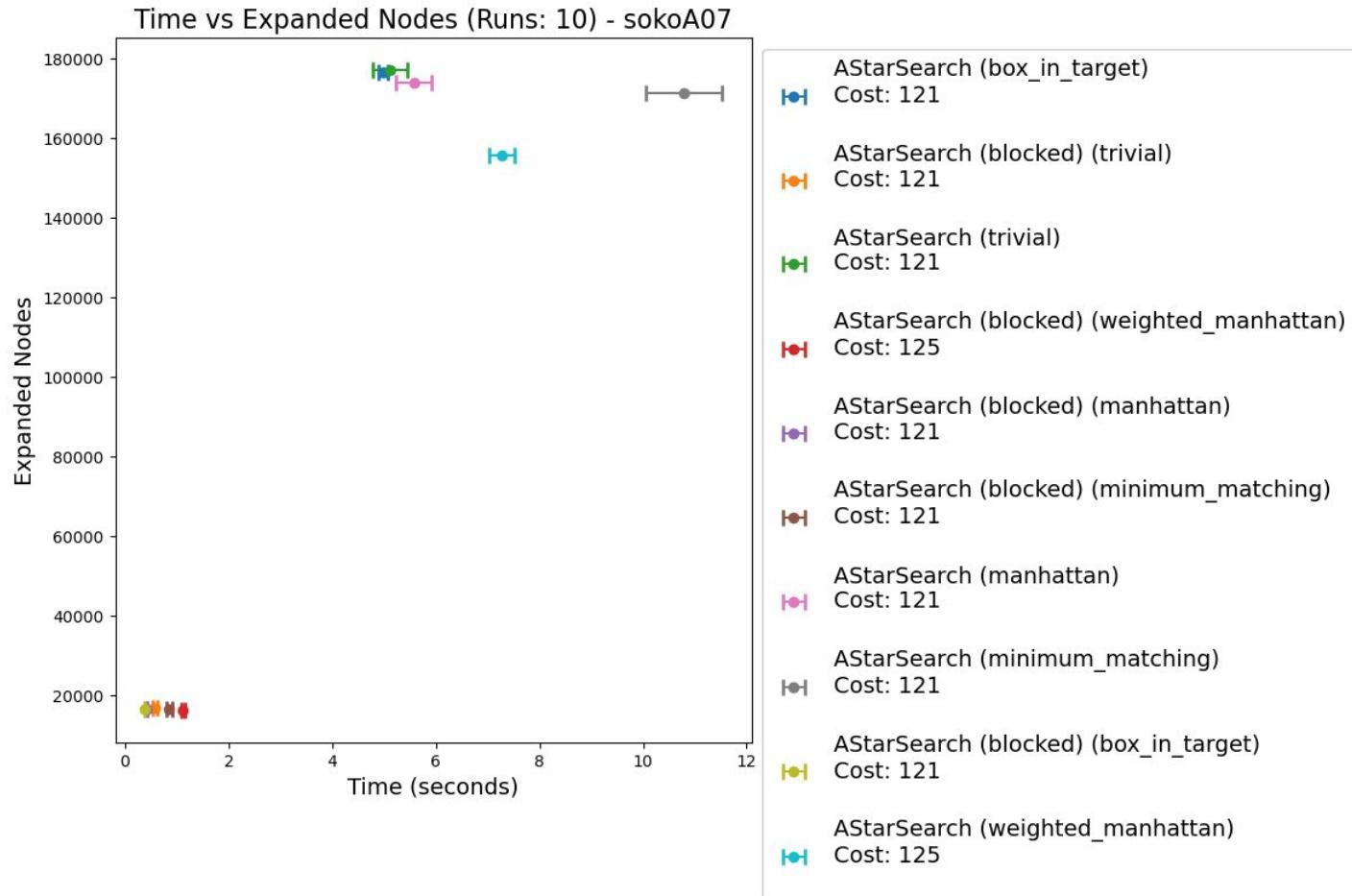
Anexo - Resultados



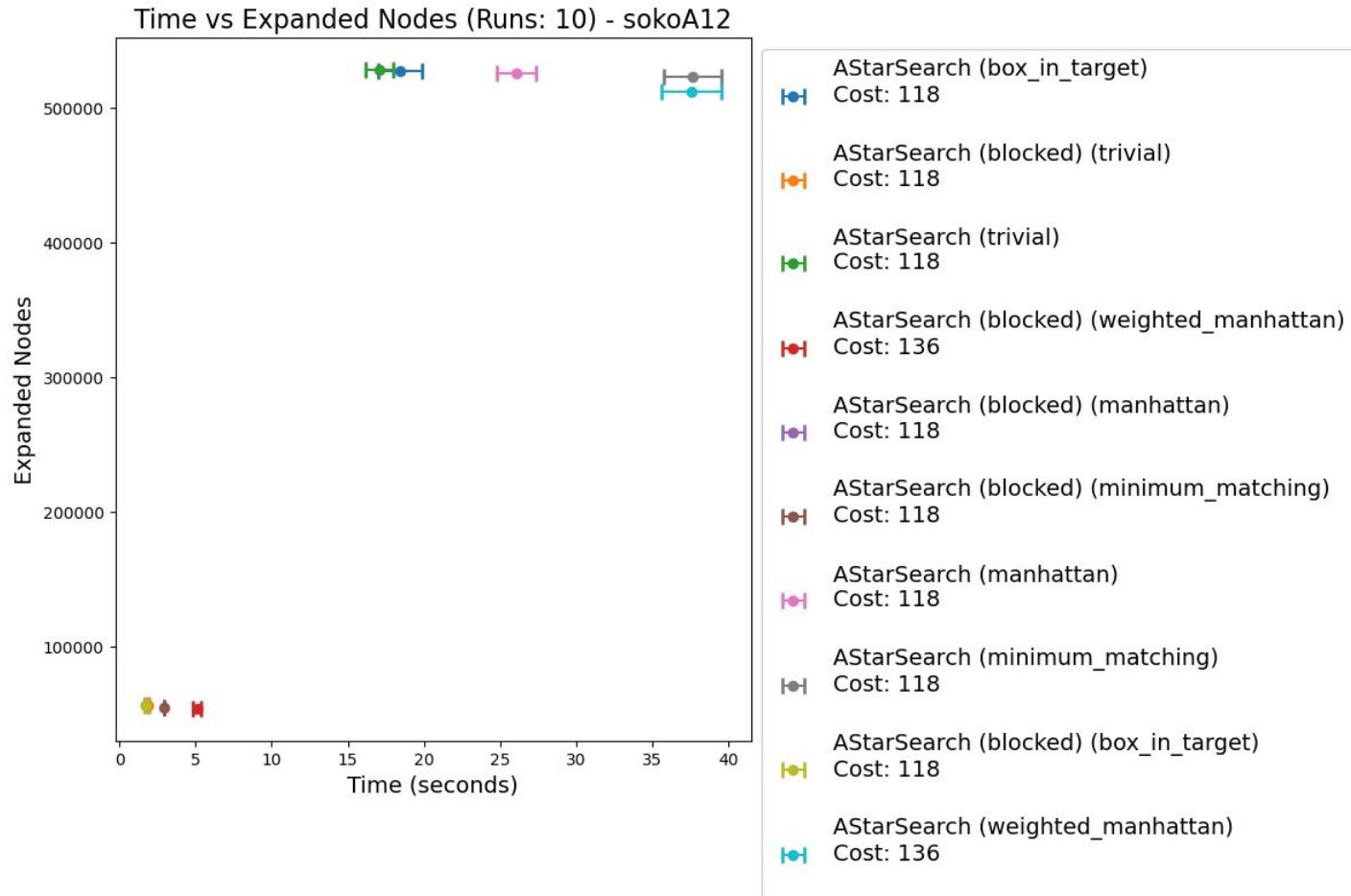
Anexo - Resultados



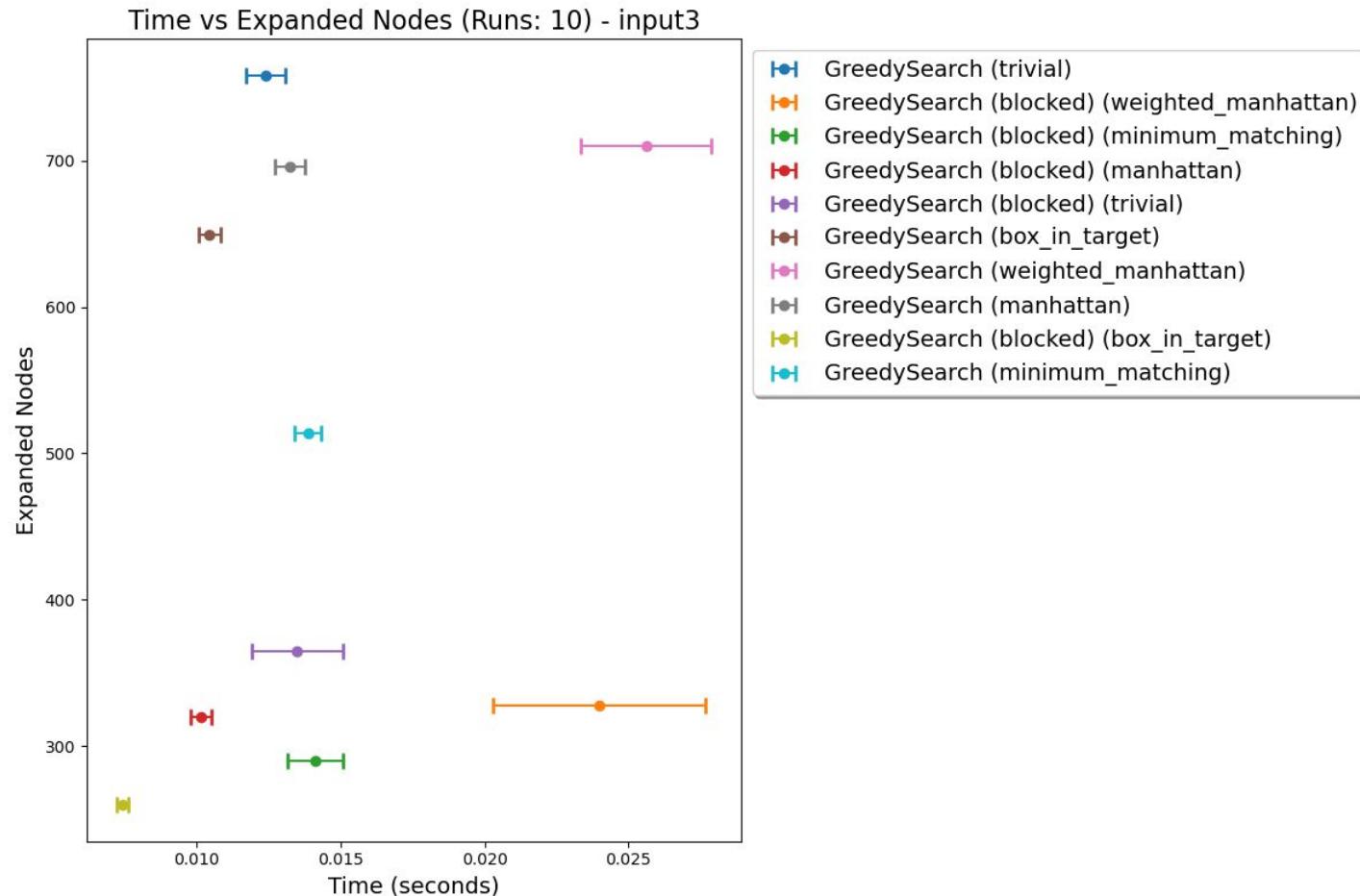
Anexo - Resultados



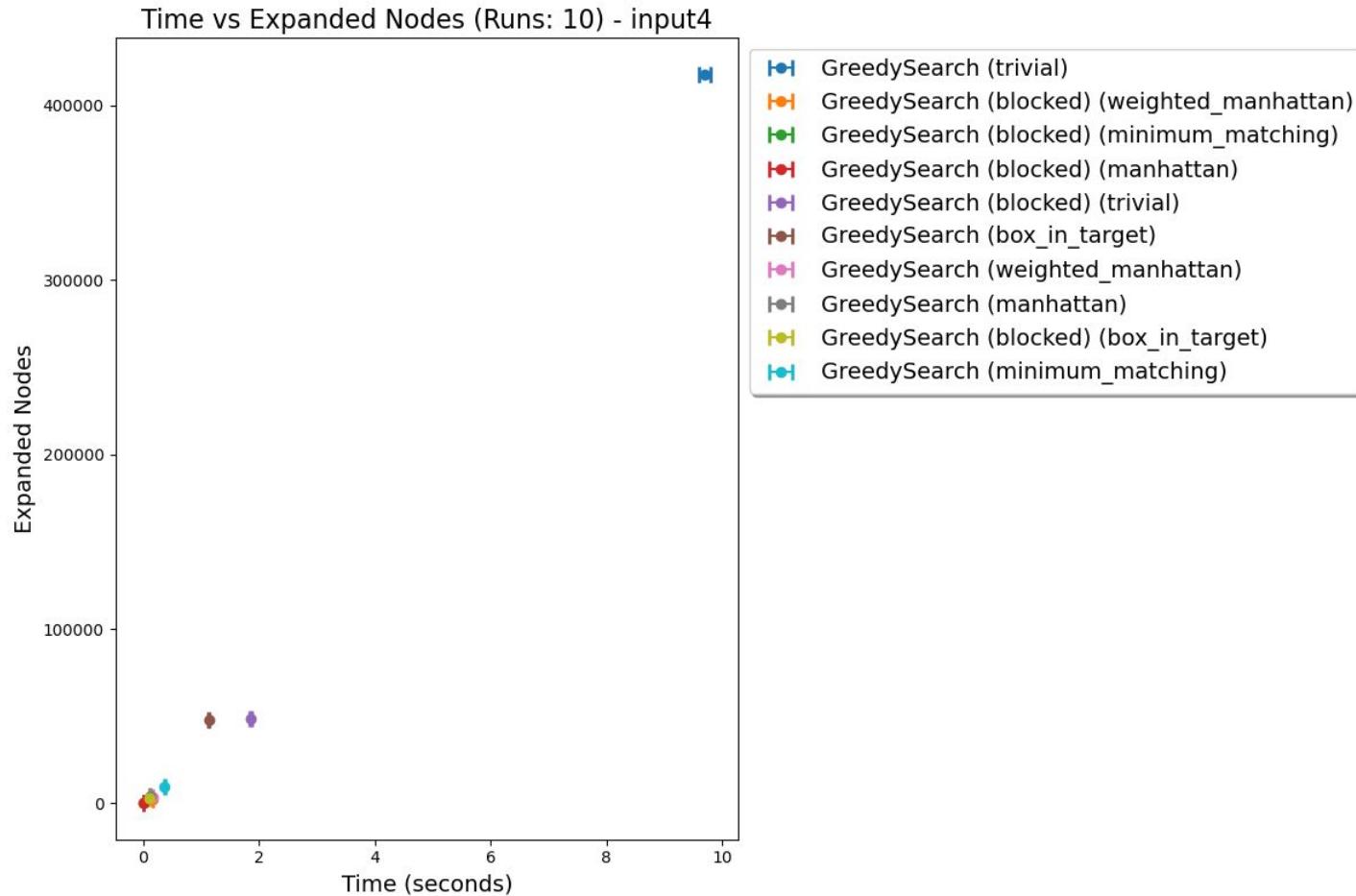
Anexo - Resultados



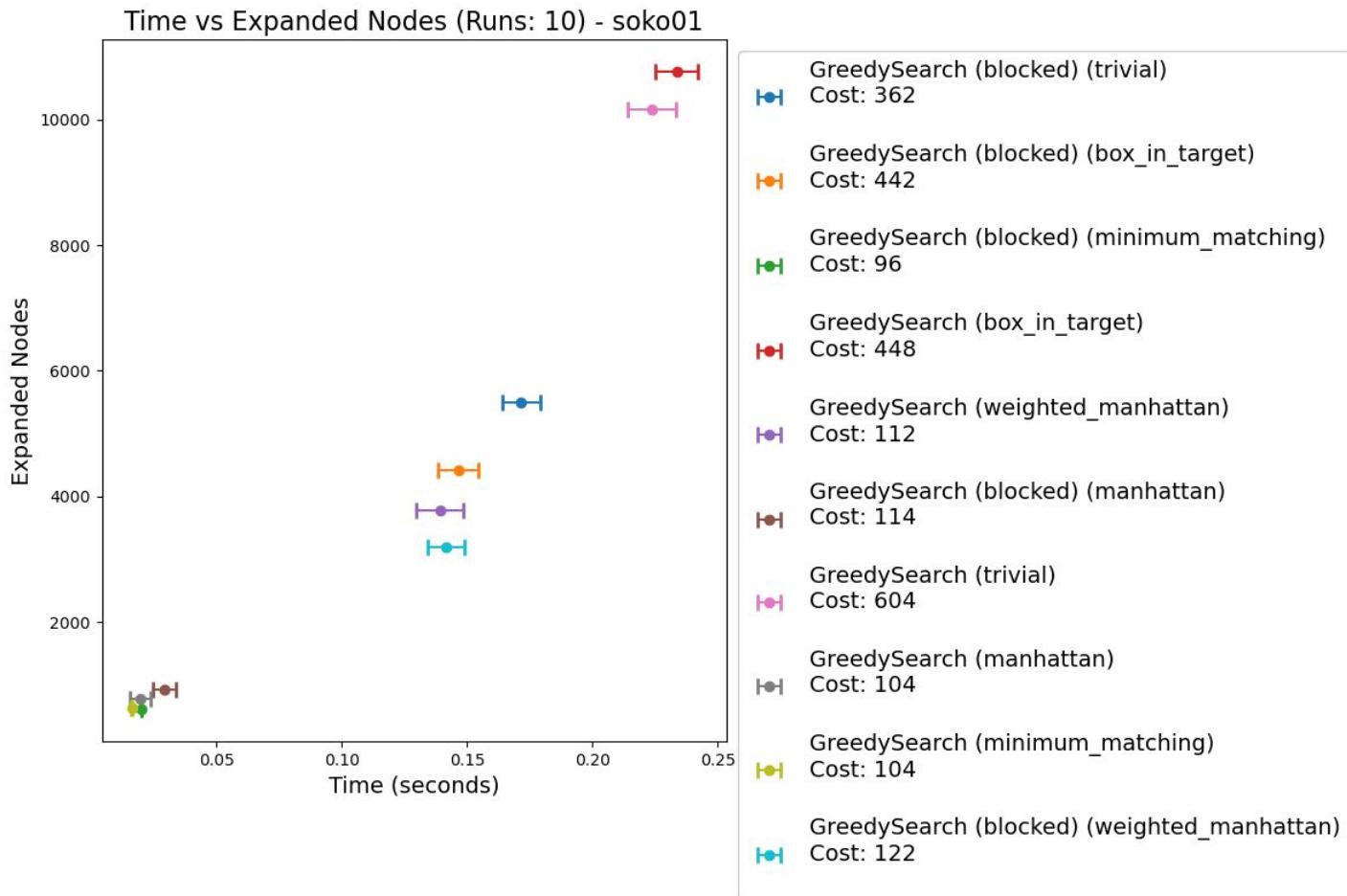
Anexo - Resultados



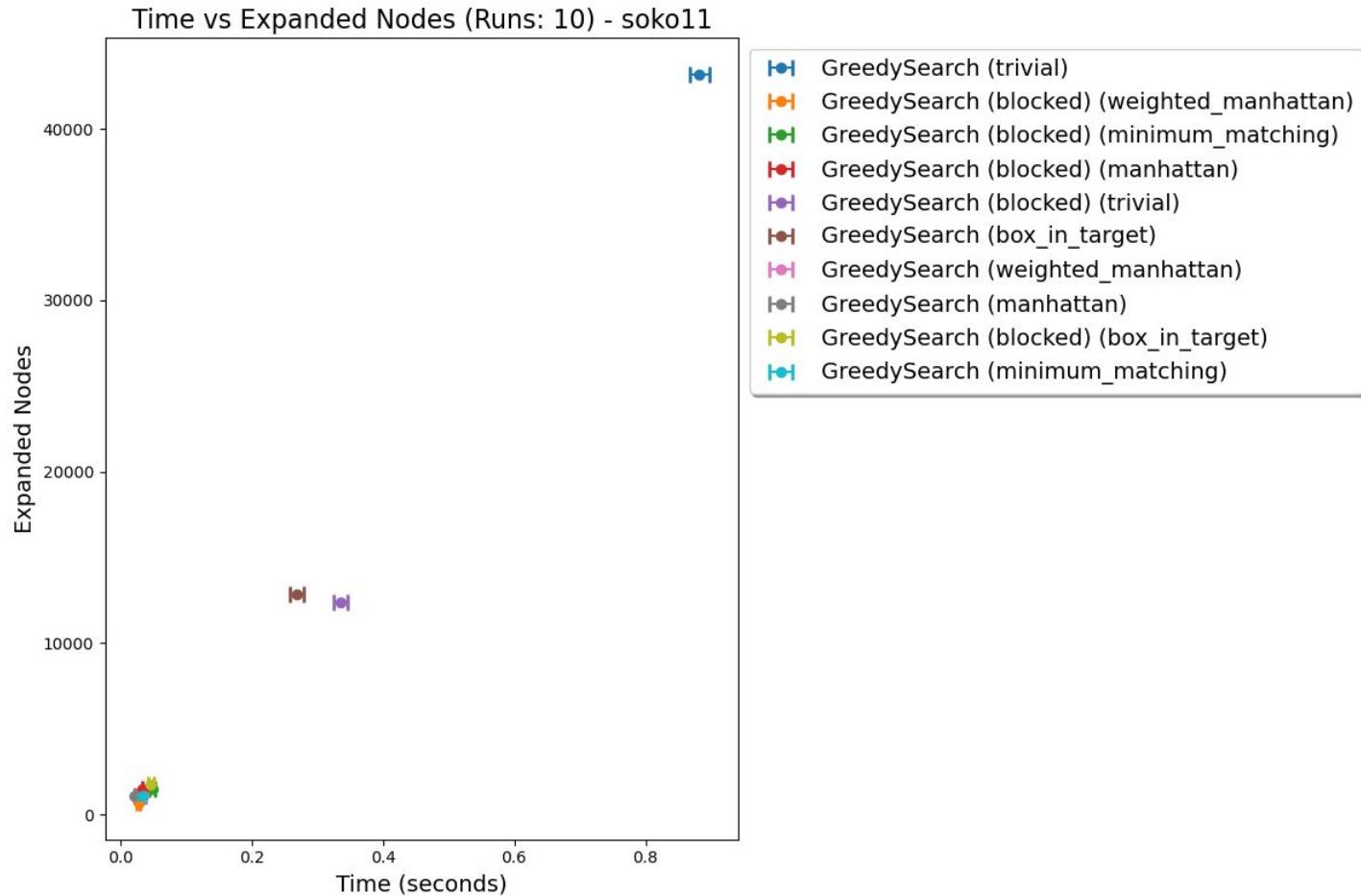
Anexo - Resultados



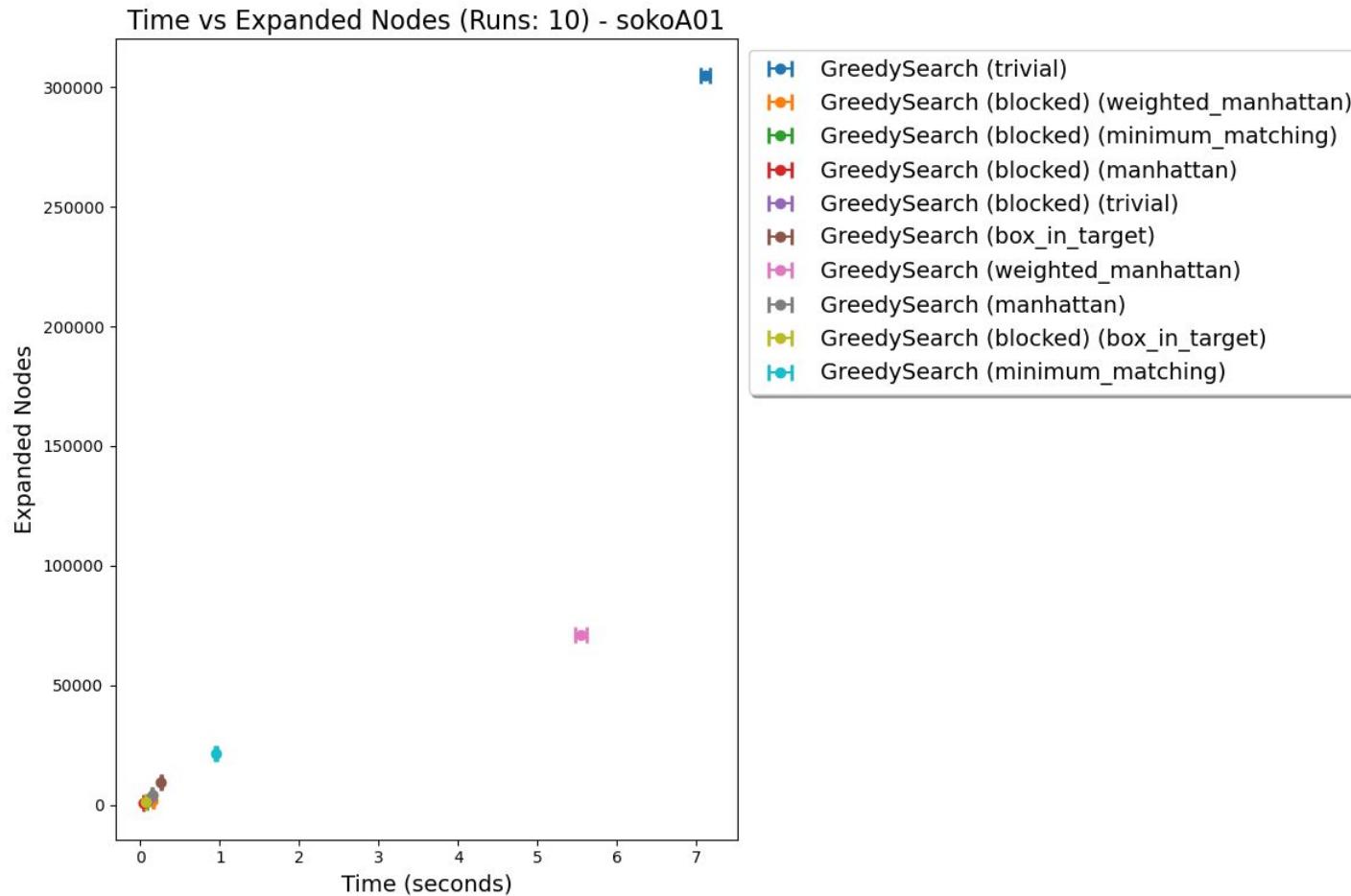
Anexo - Resultados



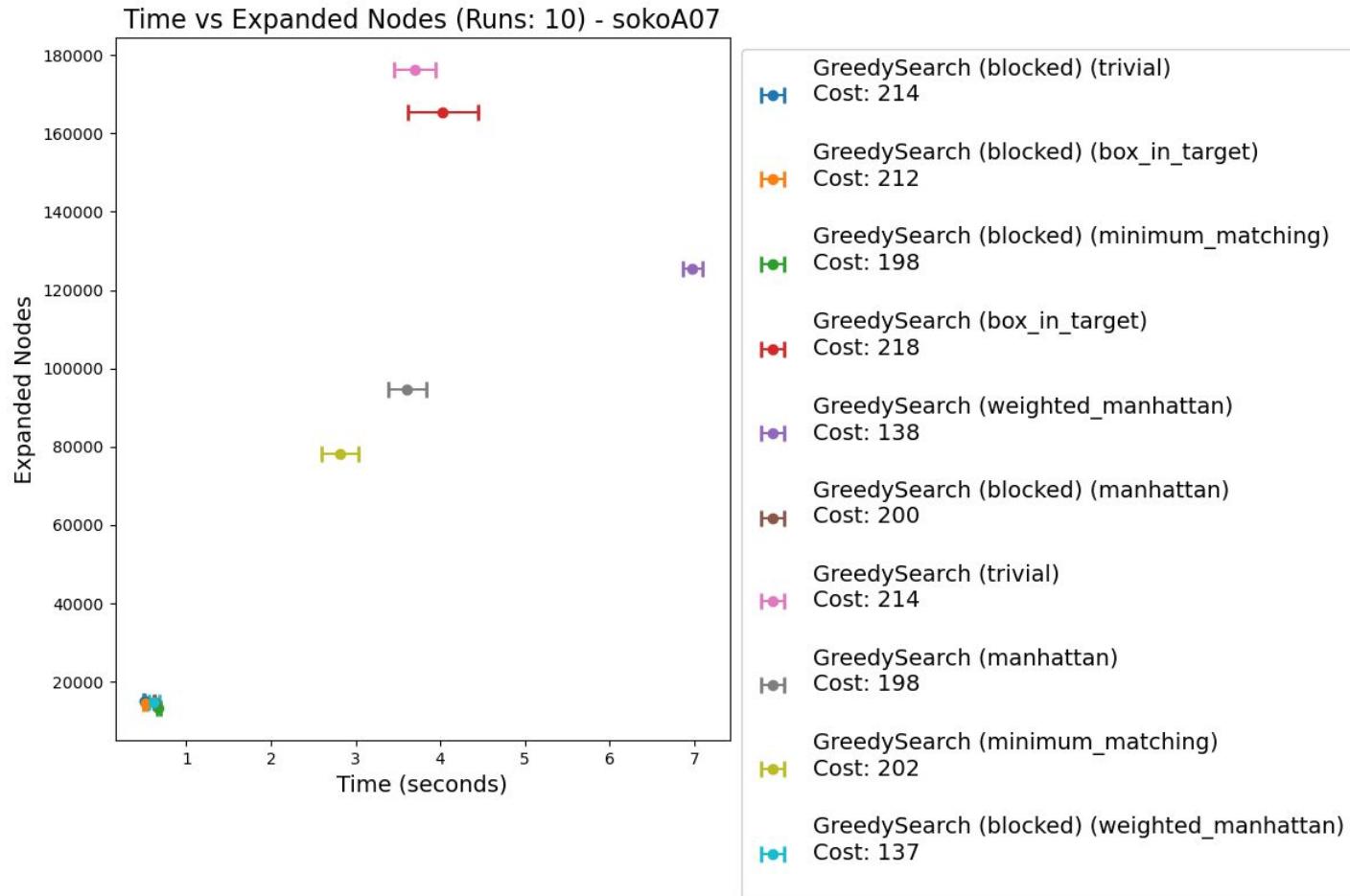
Anexo - Resultados



Anexo - Resultados



Anexo - Resultados



Anexo - Resultados

