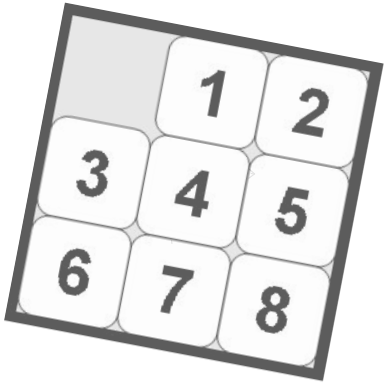
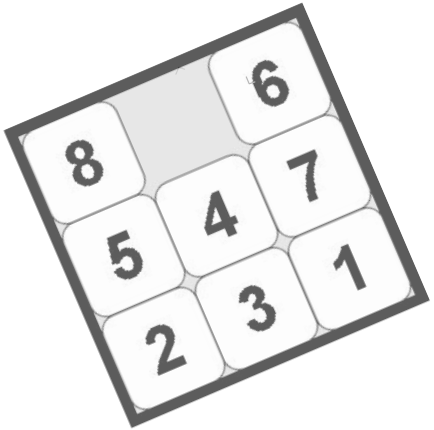
TP1
Métodos de Búsqueda

Grupo 2

Tabla de contenidos



8-Puzzle



Estructuras de estado

Heurísticas

Algoritmos

Approach

- Jugamos para resolver el juego
- Intentamos entender los mecanismos lógicos que seguimos cuando ganamos
- Jugamos siguiendo las reglas que aprendimos en los pasos previos
- Revisamos las reglas cuando no eran suficientes para ganar

Resultado

- Para ganar era necesario ordenar alrededor de la celda central a las piezas.
- Buscamos introducir una pieza al centro y rotar el resto de las celdas alrededor del anillo hasta ubicar la pieza central a donde corresponde.
- Este método nos sirvió para definir heurísticas y estado

Estructura de estados

- Representar las fichas en una clase
- Representar el espacio libre del tablero

Heurísticas

- H1: Suma de distancias Manhattan entre una pieza y donde corresponde estar

  ⇒ Pero si hay que rotar un anillo la solución no es muy precisa

- H2: Cantidad de piezas en un lugar incorrecto

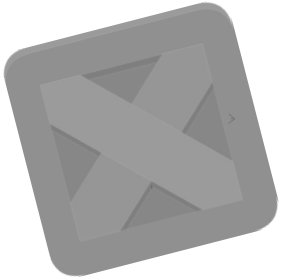
  ⇒ Es más simple pero sigue contradiciendo al método

- H3: Cantidad de piezas desordenadas alrededor del anillo + H1

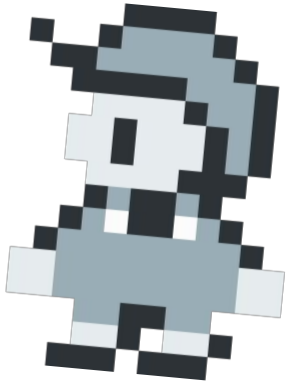
  ⇒ No contradice al proceso que encontramos!

Algoritmos

- A*
- Iterative Deepening A*



Sokoban



Estructuras de estado

Heurísticas

Algoritmos

Estructuras de estado

- Posición del jugador
- Posición de las cajas
- Posición de los “goals”
- Posición de las paredes
- Posición de los deadlocks

Heurísticas

- ***Manhattan Distance***: Suma de las distancias mínimas entre cada caja y un goal, y distancia mínima entre un jugador y una caja.

- ***Min Distance***: Distancia mínima jugador-caja-goal (medida en Manhattan).

- ***Bipartite Heuristic***: Combinación mínima de distancias jugador-caja-goal (medida en Manhattan) considerando todas las cajas.

Algoritmos

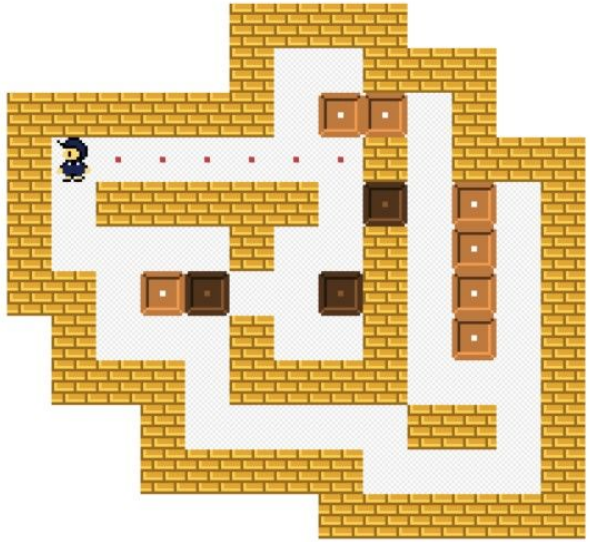
- BFS
- DFS
- Greedy Local
- Greedy Global
- A*

Deadlocks

- *Definición*
  Puntos en los que si se pone una caja, no hay forma de poder llevarla a un *“goal”.*

- *Primera solución:* ***Esquinas***
  Hallar las esquinas del mapa y marcarlas como deadlocks.

  Excepción: Si hay un goal en la esquina, no se marca como deadlock.

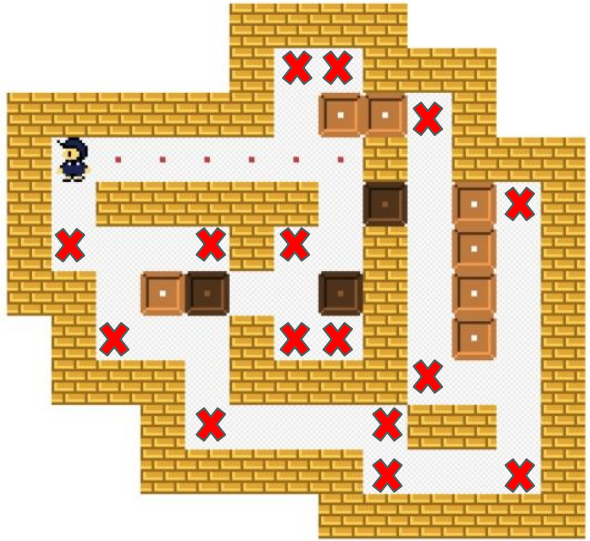


Deadlocks

- *Definición*
  Puntos en los que si se pone una caja, no hay forma de poder llevarla a un *“goal”.*

- *Primera solución:* ***Esquinas***
  Hallar las esquinas del mapa y marcarlas como deadlocks.

  Excepción: Si hay un goal en la esquina, no se marca como deadlock.

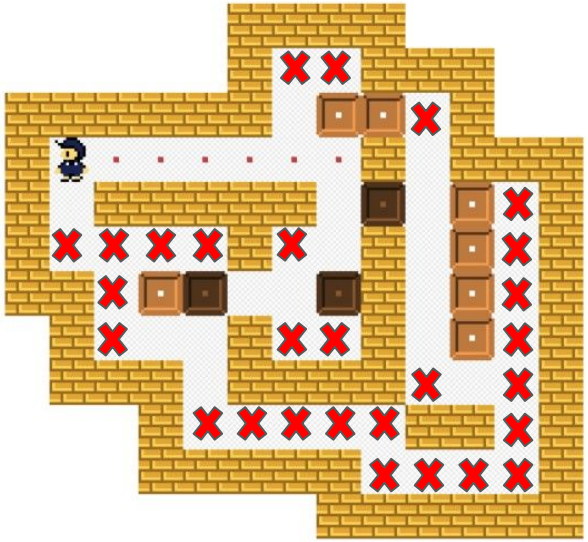


Deadlocks

- *Mejora de la solución:* ***Esquinas y Paredes completas***

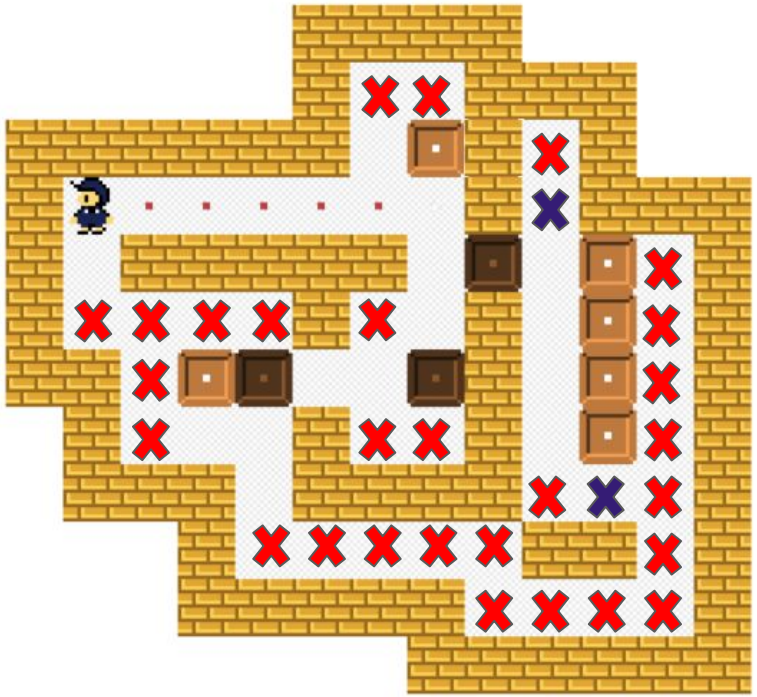
  Se toma una esquina y se mueve en las 4 direcciones hasta encontrar una pared.
  Se detecta si en cada paso hay paredes al costado y se lo marca como deadlock. Si se encuentra que hay un hueco en ambas paredes (no necesariamente enfrentados), se desmarcan todos los deadlocks de esa línea (menos las esquinas).

  Excepción: Si en la línea se encuentra un goal, también se descartan los deadlocks de la línea (menos las esquinas)



Deadlocks

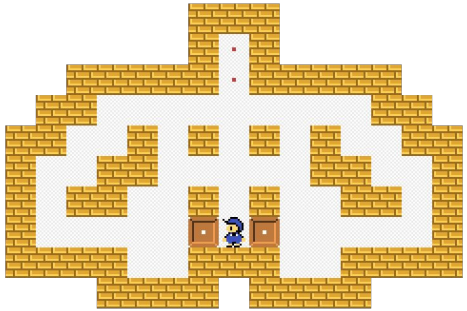
- Deadlocks no detectables por la solución



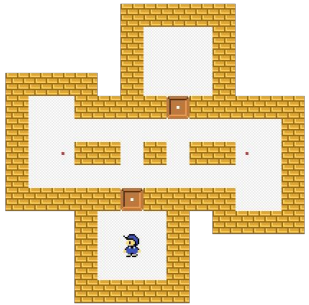
Data analysis

Mapas a utilizar:

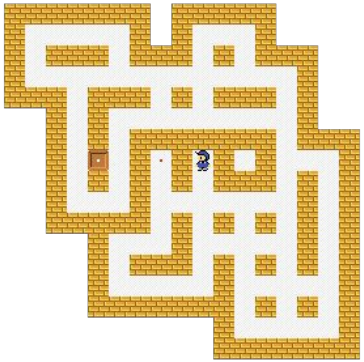
Level 1



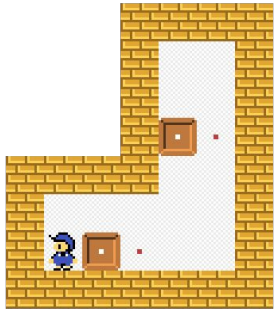
Level 2



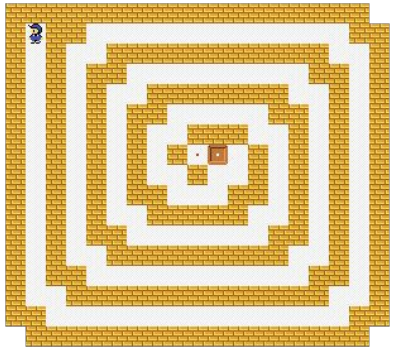
Level 3



Level 4



Level 5



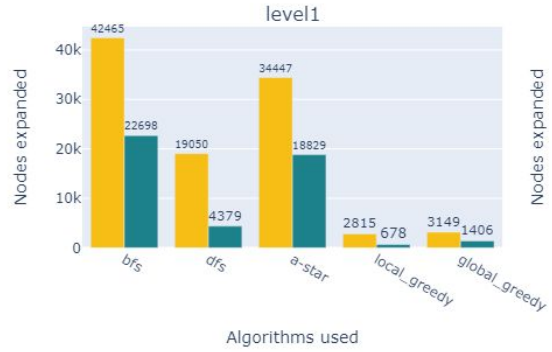
Deadlocks: Comparación de algoritmos

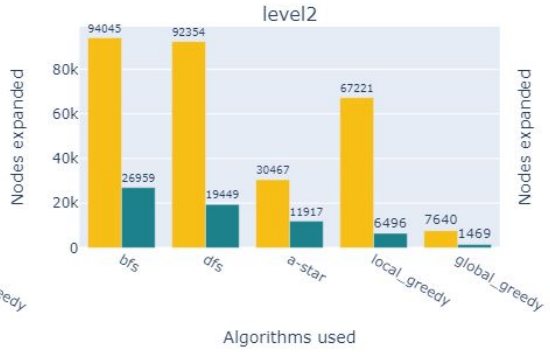
Para obtener estos datos se corrieron todos los algoritmos con y sin deadlocks en cada uno de los mapas. Para aquellos algoritmos con heurísticas se utilizó la de la distancia Manhattan.

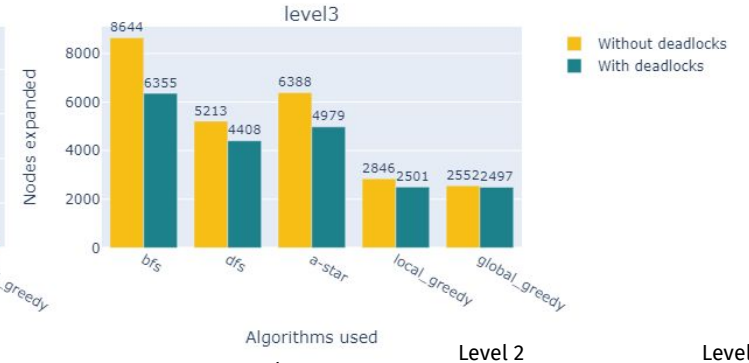
❓ Preguntas:

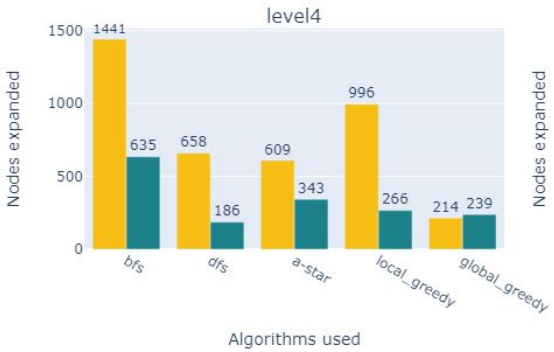
- ¿Vale la pena gastar tiempo en pre-calcular los deadlocks?
- ¿Vale la pena un algoritmo óptimo a expensas de más nodos expandidos?
- ¿Se puede concluir qué algoritmo es mejor?

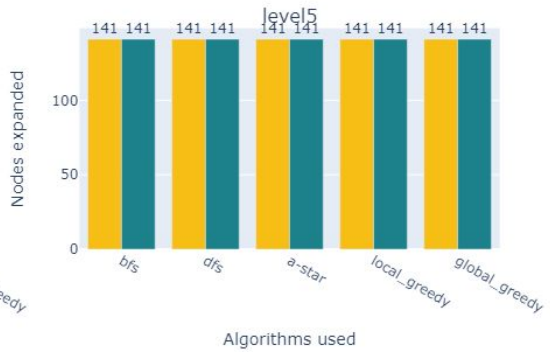
Deadlocks: Comparación de algoritmos

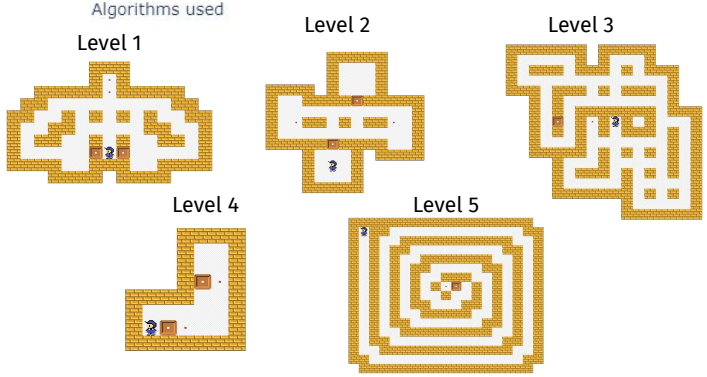






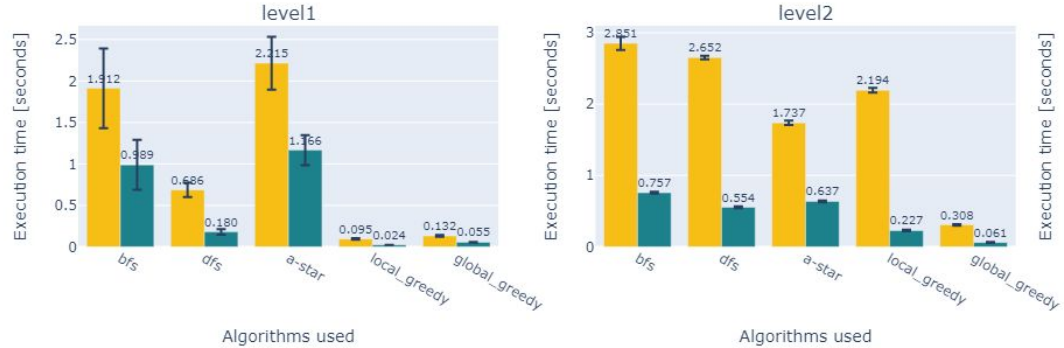


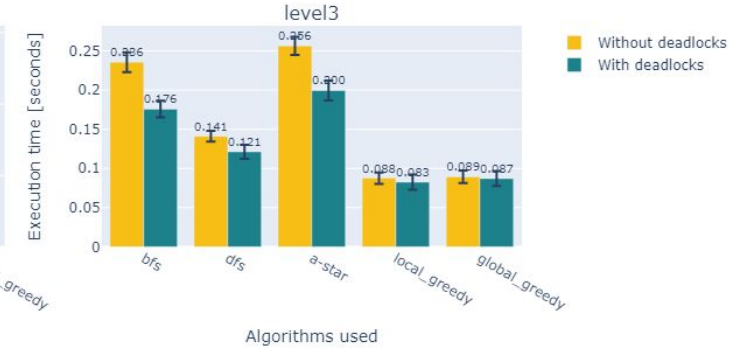


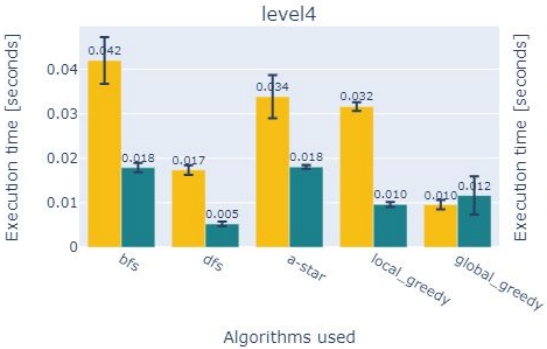


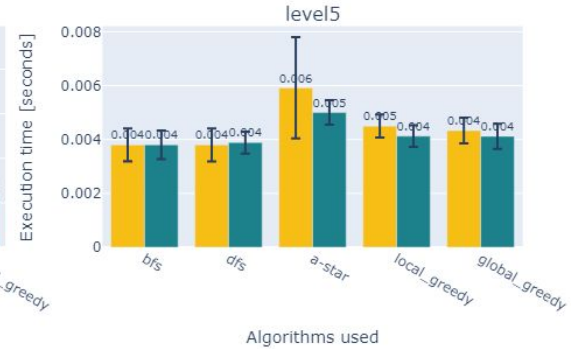
Deadlocks: Comparación de algoritmos

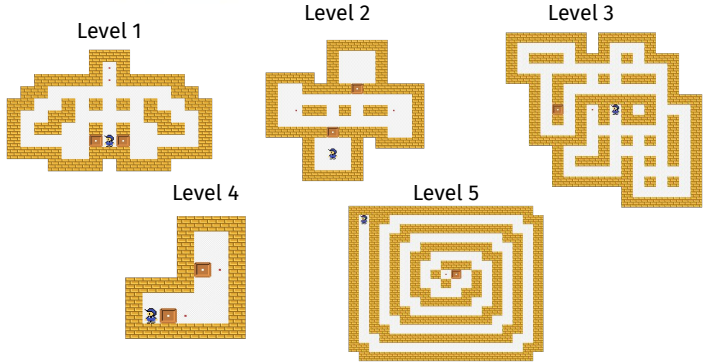
(manhattan)











Heurísticas: Tiempos de ejecución y nodos expandidos (con deadlocks)

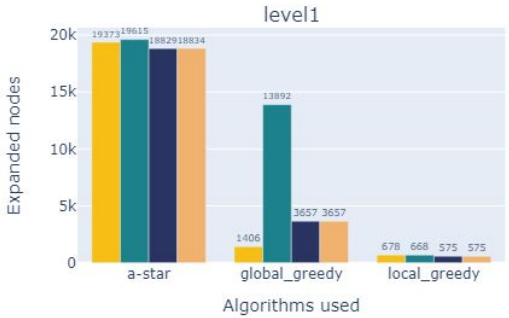
Para obtener estos datos se corrieron solamente los algoritmos que hacen uso de las heurísticas, dado que se buscaba comparar la relación costo-beneficio de usarlos.

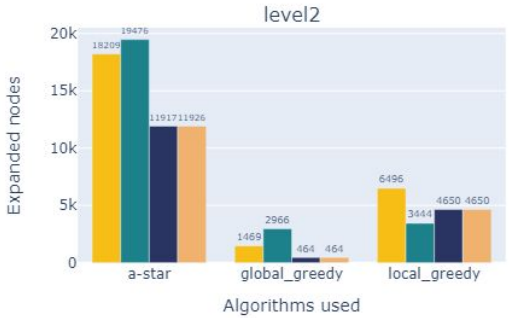
❓ Preguntas:

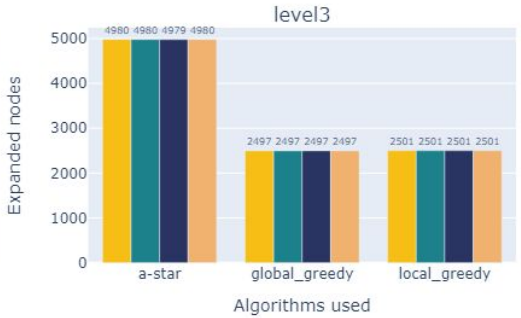
- ¿Existe alguna heurística que beneficia particularmente a un algoritmo?
- ¿El tiempo que tarda en calcular la heurística justifica su uso?
- ¿Qué heurística tarda más en calcularse? ¿Es posible saberlo?

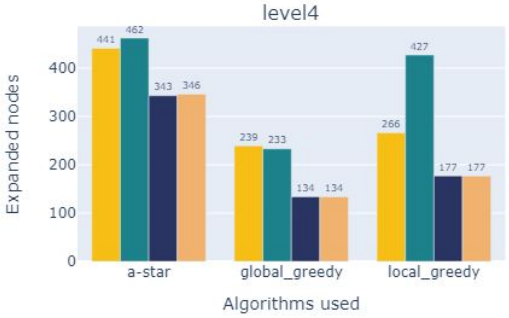
Nodos expandidos

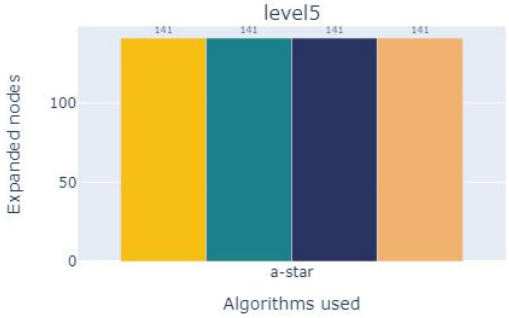
Heurísticas: Nodos expandidos (con deadlocks)

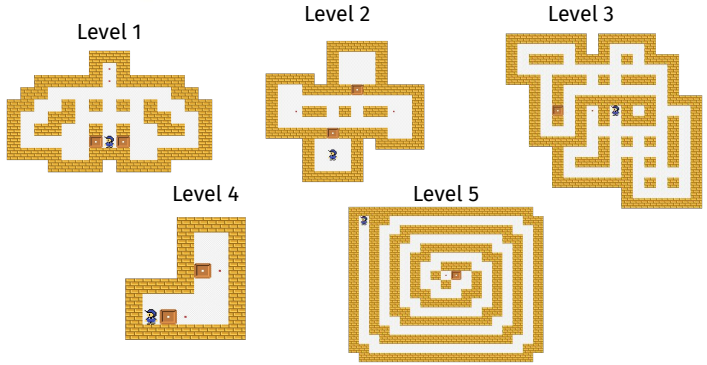






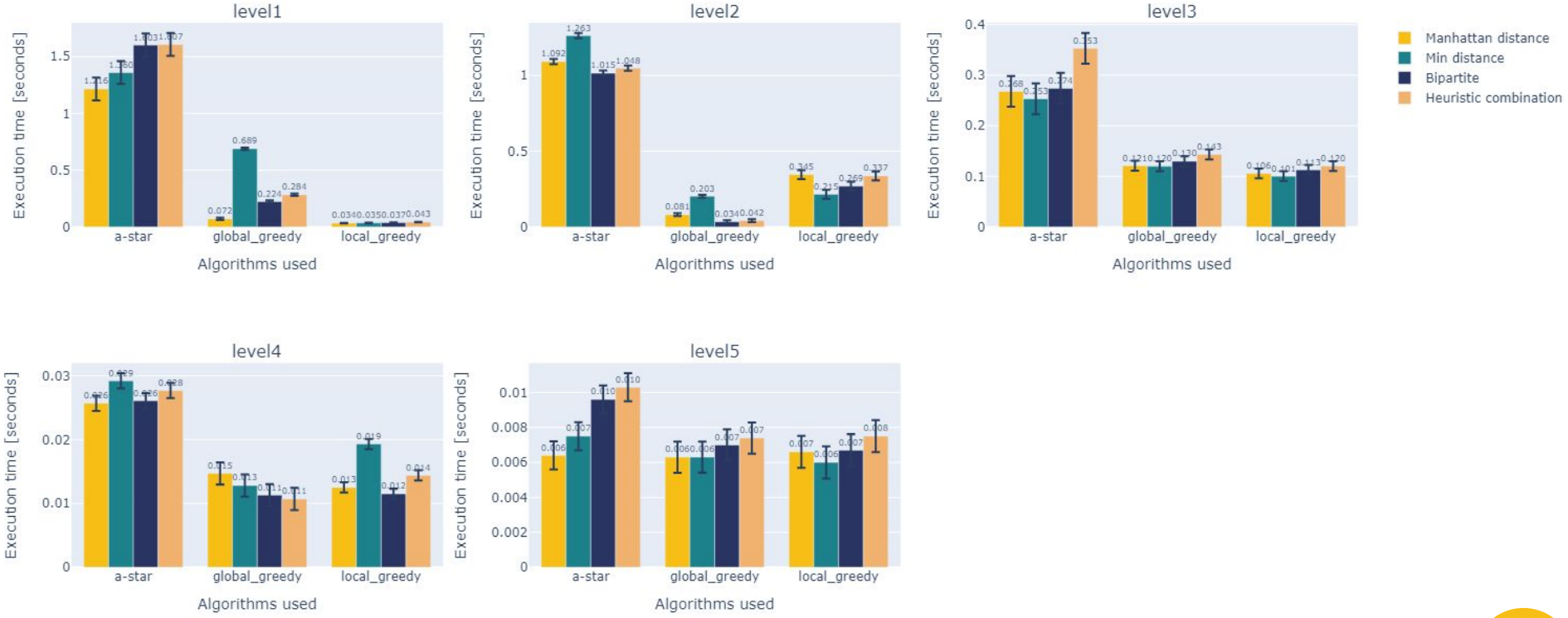






Tiempo de ejecución

Heurísticas: Tiempo de ejecución (con deadlocks)



Heatmap de posiciones de las cajas

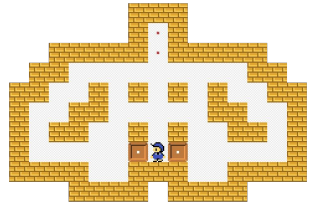
Para obtener estos datos se corrieron todos los algoritmos (con y sin deadlocks) para cada uno de los 5 mapas. En los algoritmos que utilizan heurísticas se utilizaron todas las heurísticas.

❓ Preguntas:

- ¿Sirven los deadlocks?
- ¿A qué algoritmo afecta más la detección de deadlocks? ¿A cuáles menos?
- ¿En qué mapas afecta más la detección de deadlocks? ¿Se puede extraer alguna conclusión de propiedades deseables de un mapa para que la detección de deadlocks tenga un impacto significativo en el rendimiento?

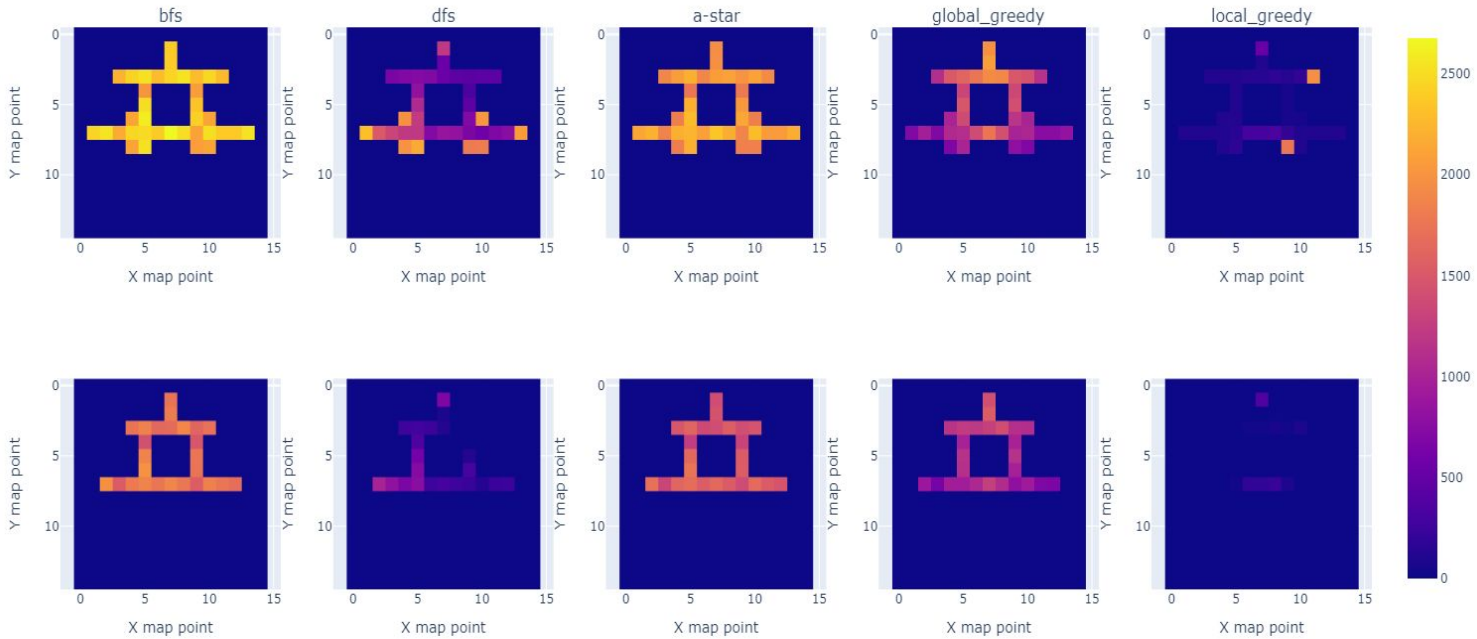
Level 1: MinDistance

Heatmap de posiciones de las cajas



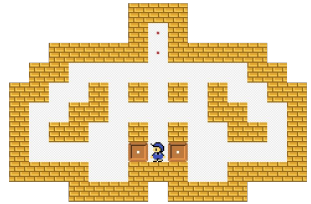
Sin deadlocks

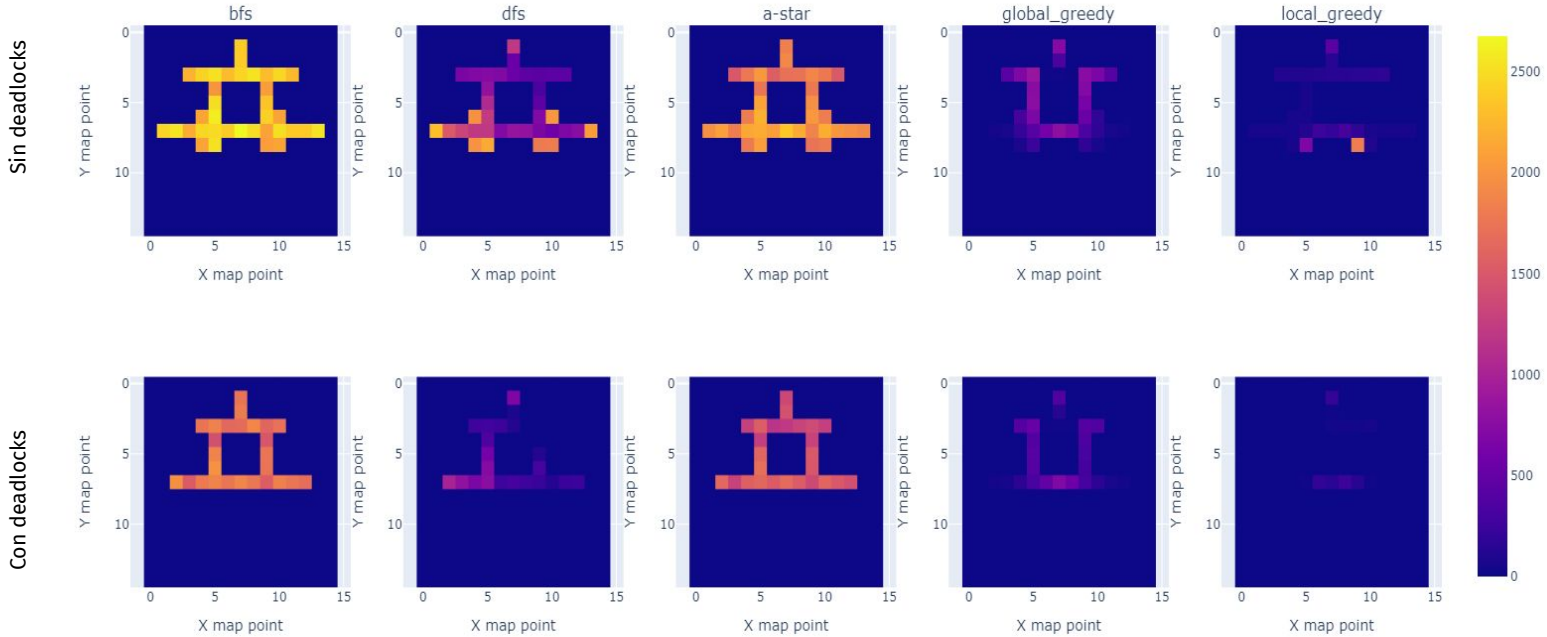
Con deadlocks



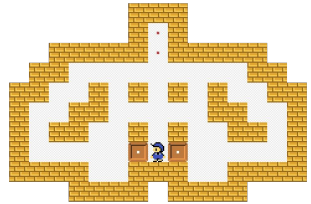
Level 1: Bipartito

Heatmap de posiciones de las cajas

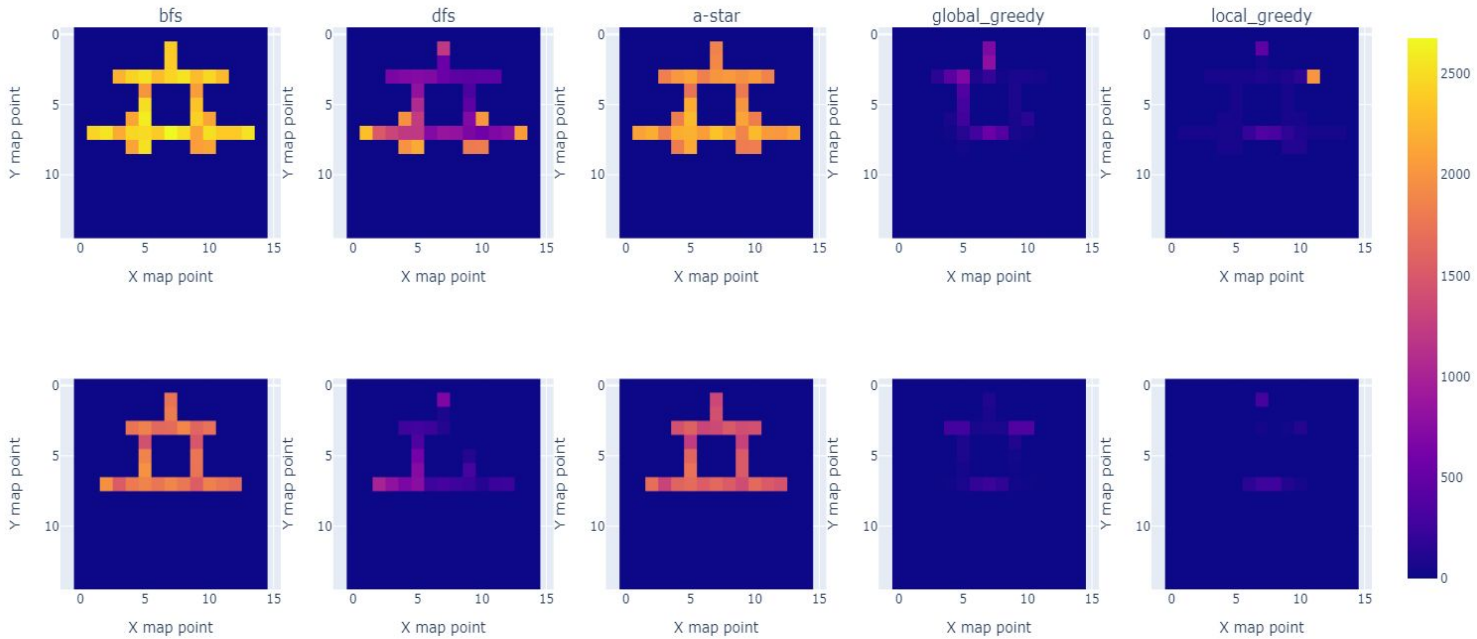




Level 1: Manhattan

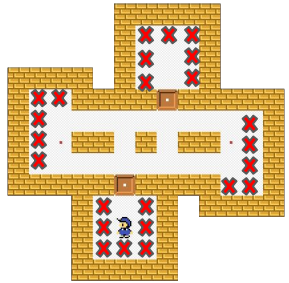


Heatmap de posiciones de las cajas



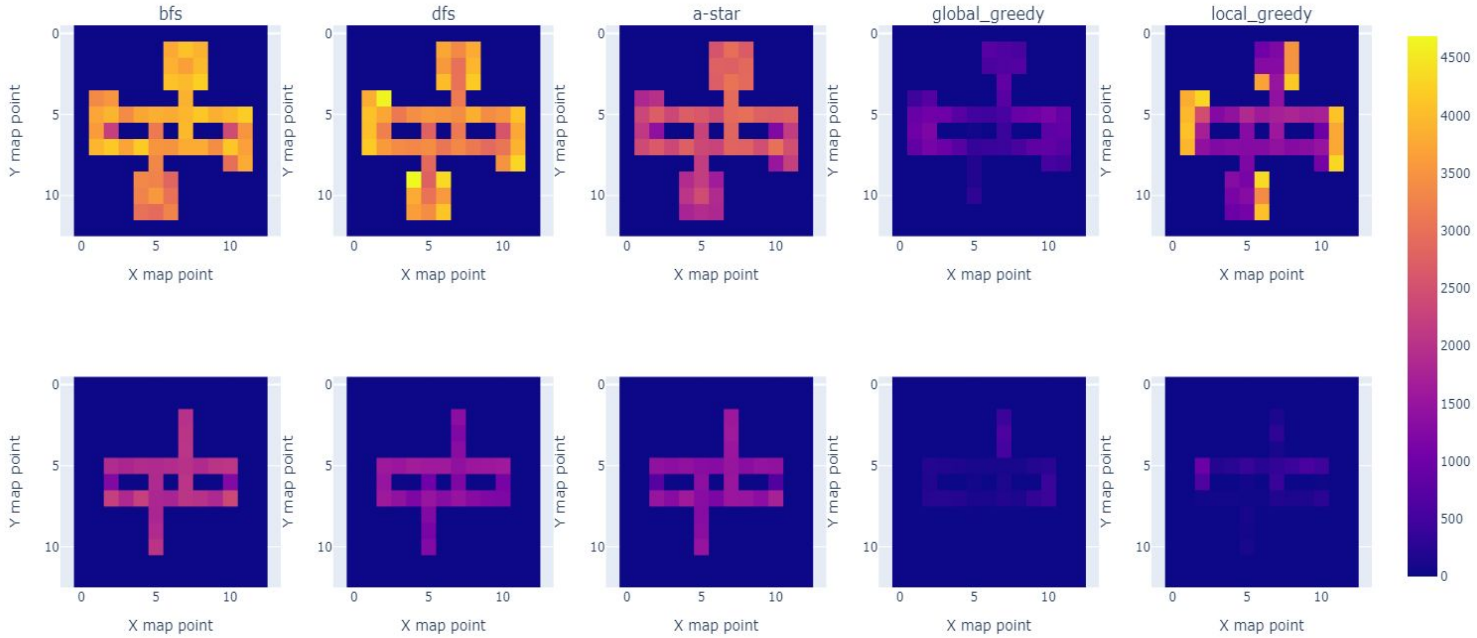
Level 2: MinDistance

Heatmap de posiciones de las cajas



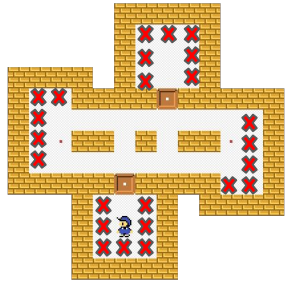
Sin deadlocks

Con deadlocks



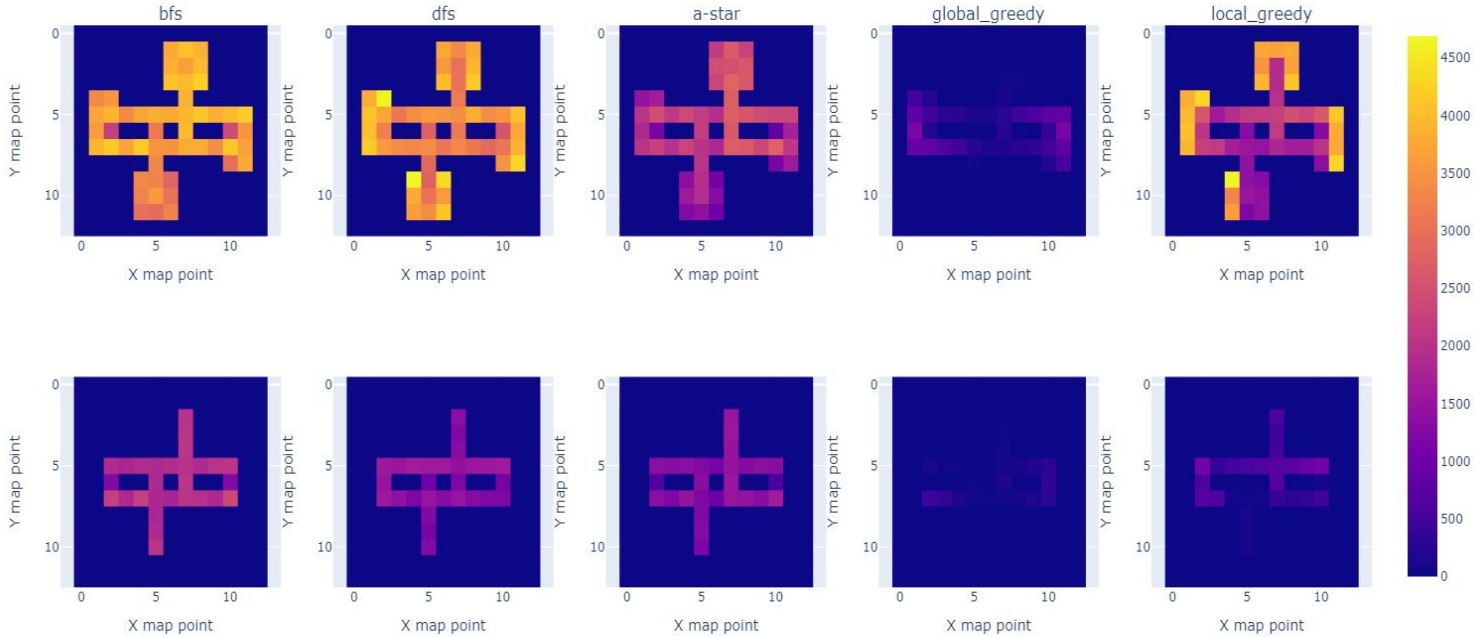
Level 2: Manhattan

Heatmap de posiciones de las cajas



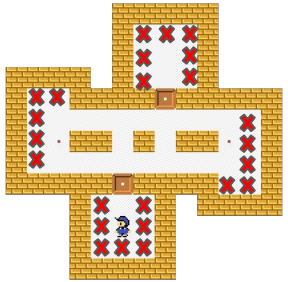
Sin deadlocks

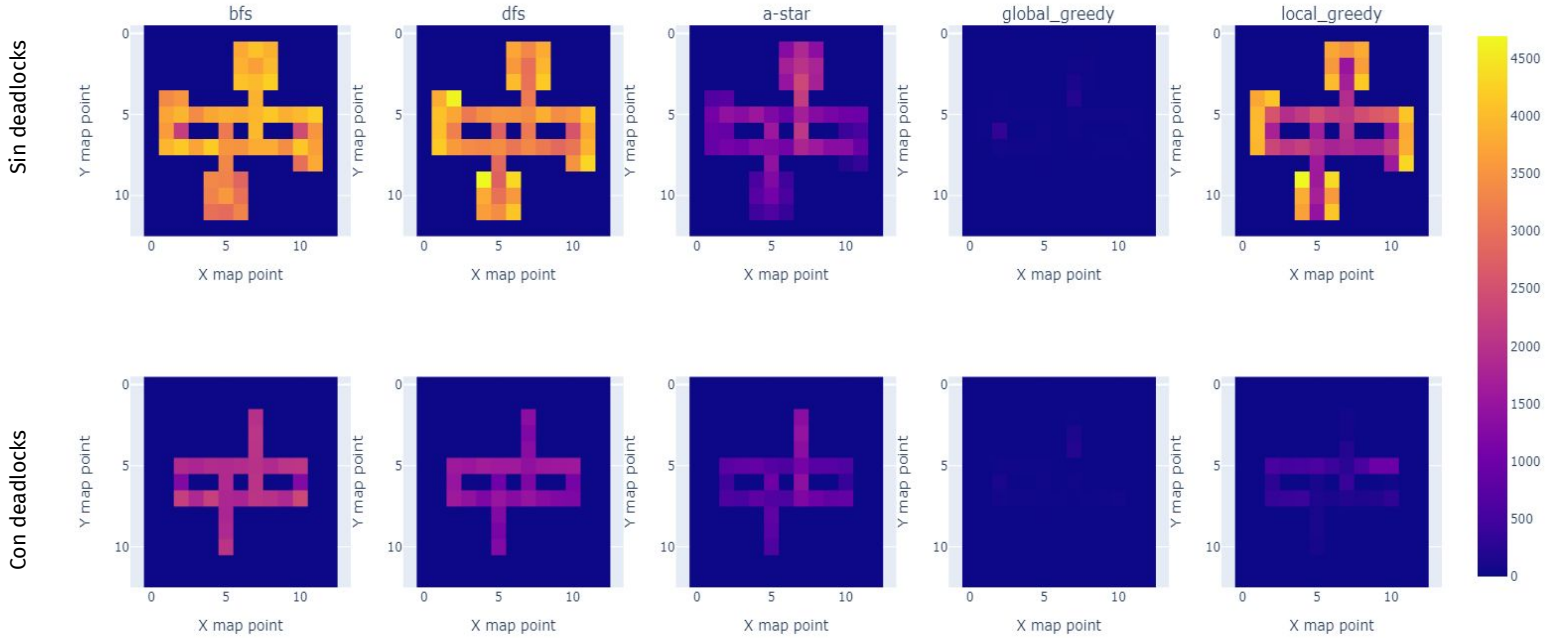
Con deadlocks



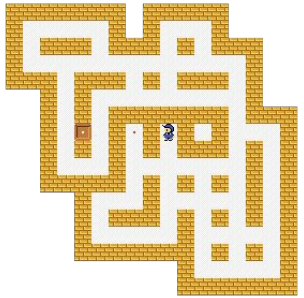
Level 2: Bipartito

Heatmap de posiciones de las cajas



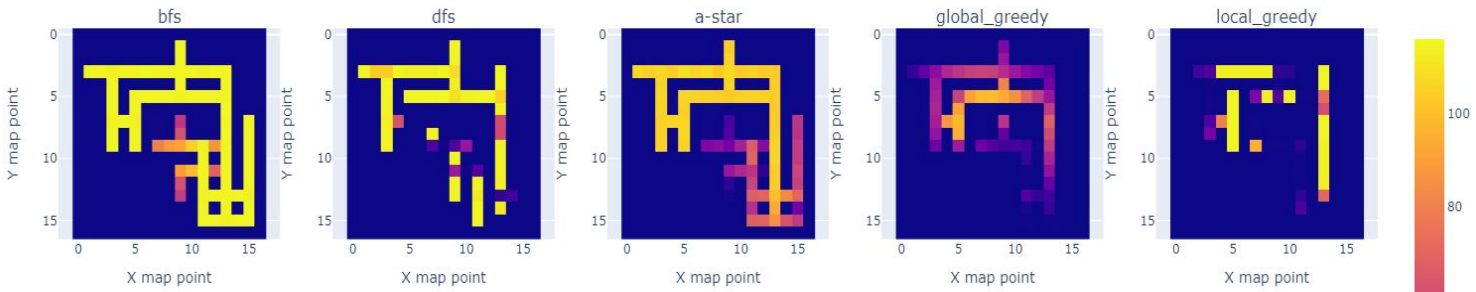


Level 3: Manhattan, MinDistance, Bipartito

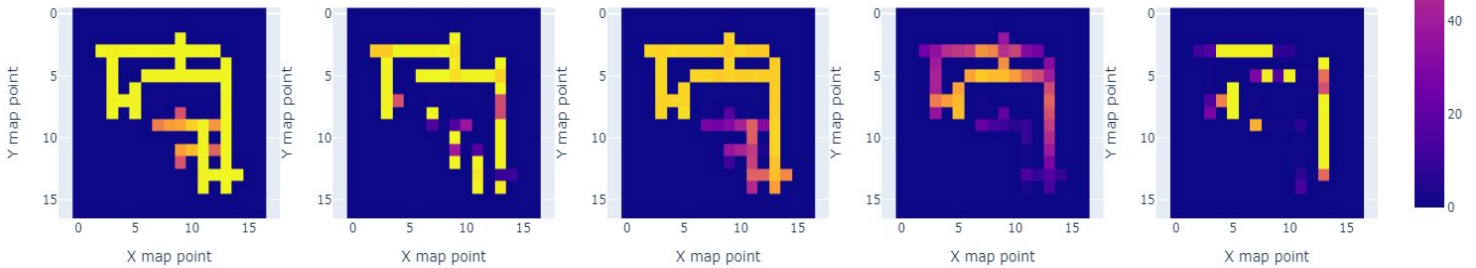


Heatmap de posiciones de las cajas

Sin deadlocks

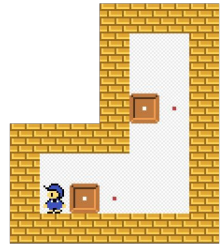


Con deadlocks



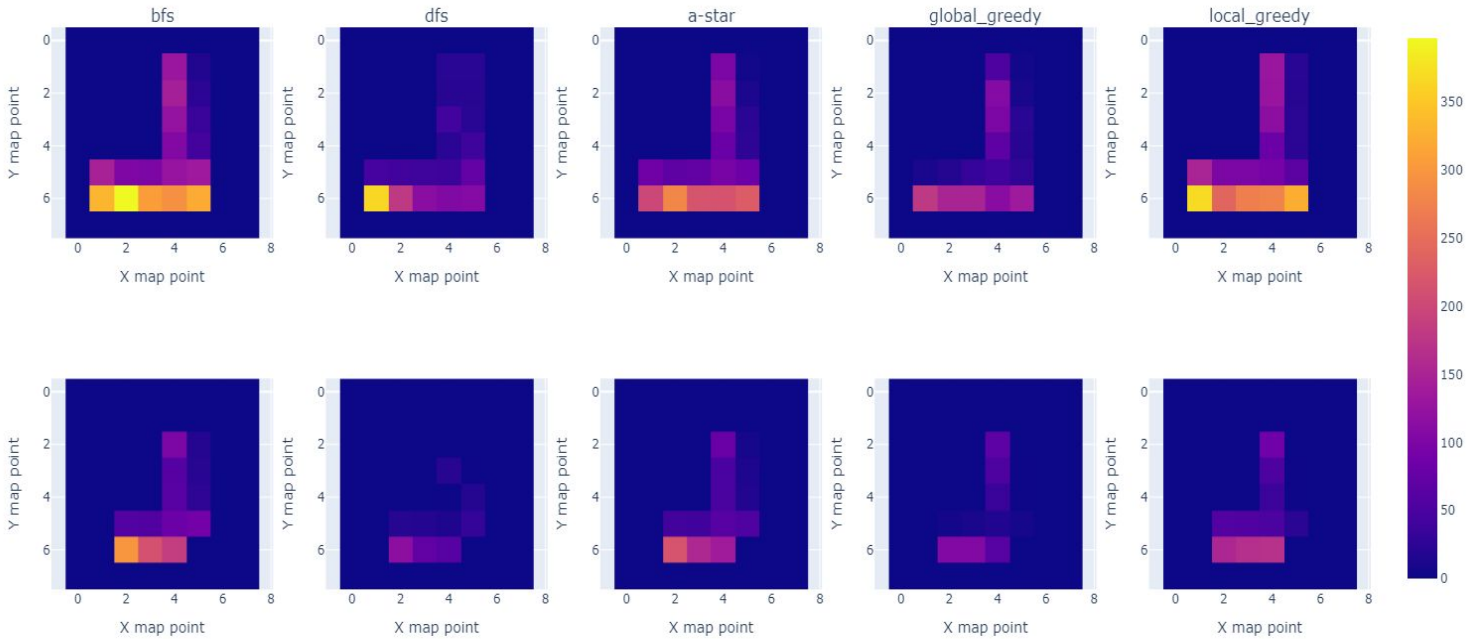
Level 4: MinDistance

Heatmap de posiciones de las cajas



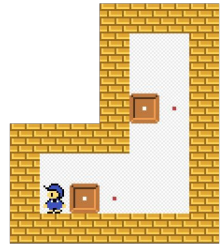
Sin deadlocks

Con deadlocks



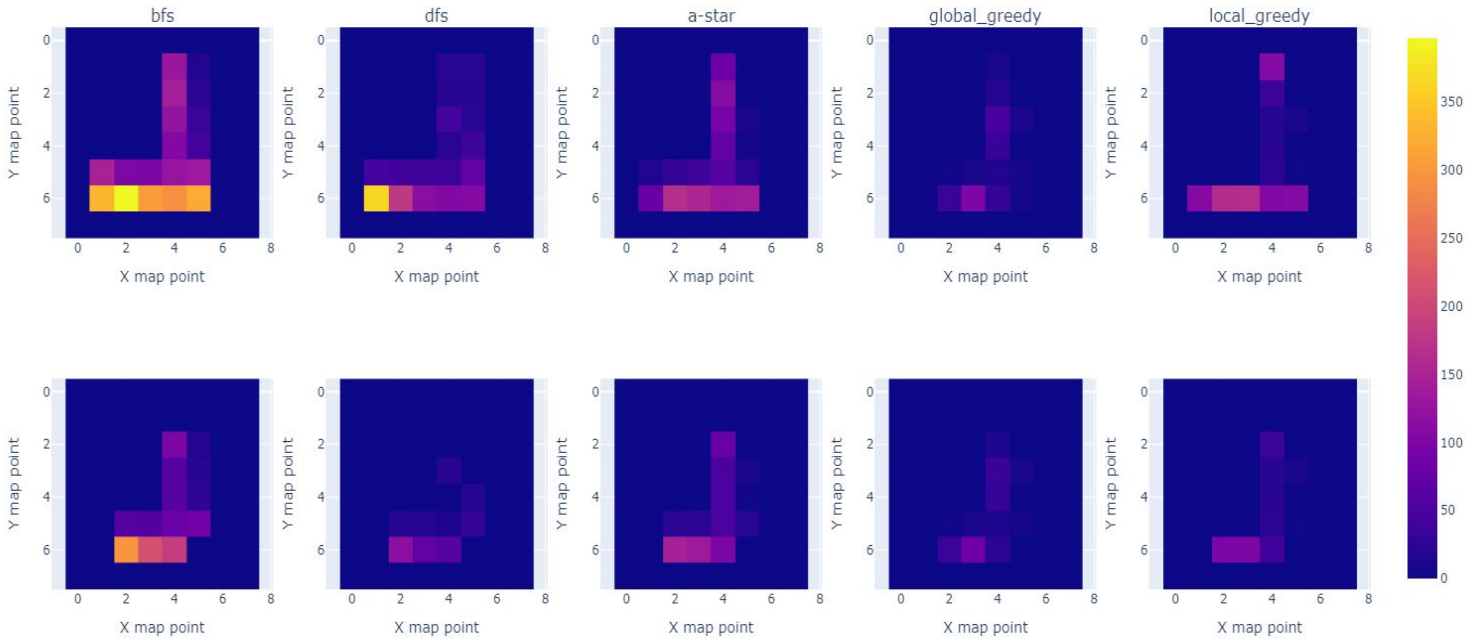
Level 4: Bipartito

Heatmap de posiciones de las cajas



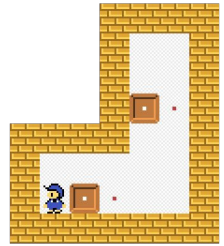
Sin deadlocks

Con deadlocks



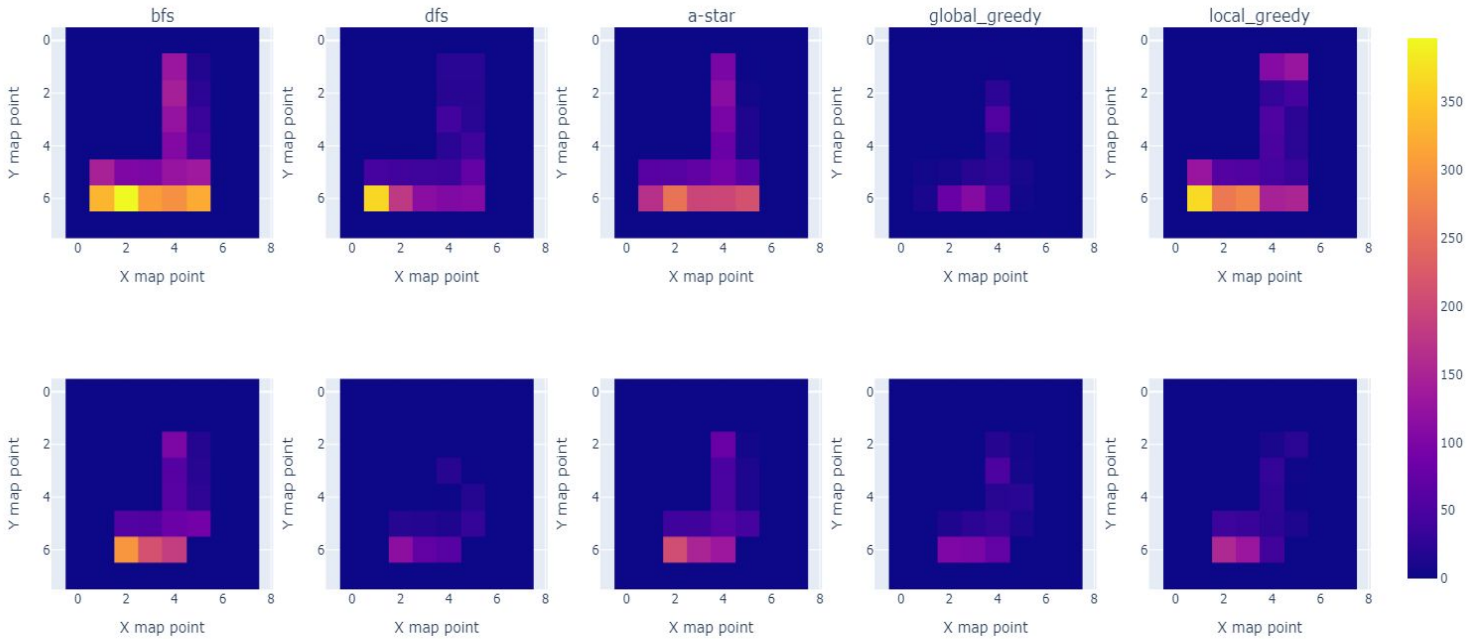
Level 4: Manhattan

Heatmap de posiciones de las cajas



Sin deadlocks

Con deadlocks



Heatmap de posiciones del jugador

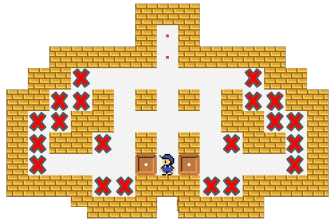
Para obtener estos datos se corrieron todos los algoritmos (con y sin deadlocks) para cada uno de los 5 mapas. Se decidió utilizar la heurística de MinDistance de manera arbitraria para poder realizar comparaciones.

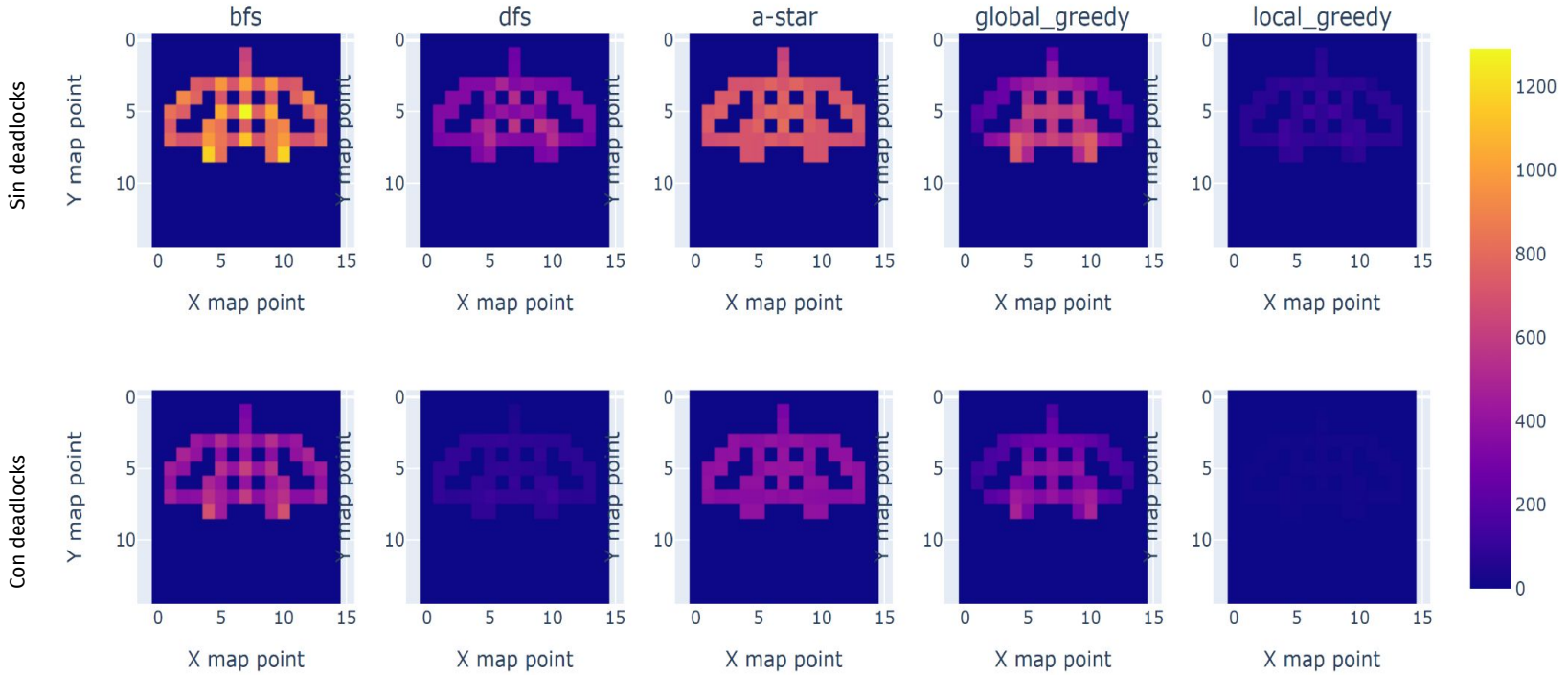
❓ Preguntas:

- ¿A qué algoritmo afecta más la detección de deadlocks? ¿A cuáles menos?
- ¿Sirven los deadlocks?
- ¿En qué mapas afecta más la detección de deadlocks? ¿Se puede extraer alguna conclusión de propiedades deseables de un mapa para que la detección de deadlocks tenga un impacto significativo en el rendimiento?

Level 1: MinDistance

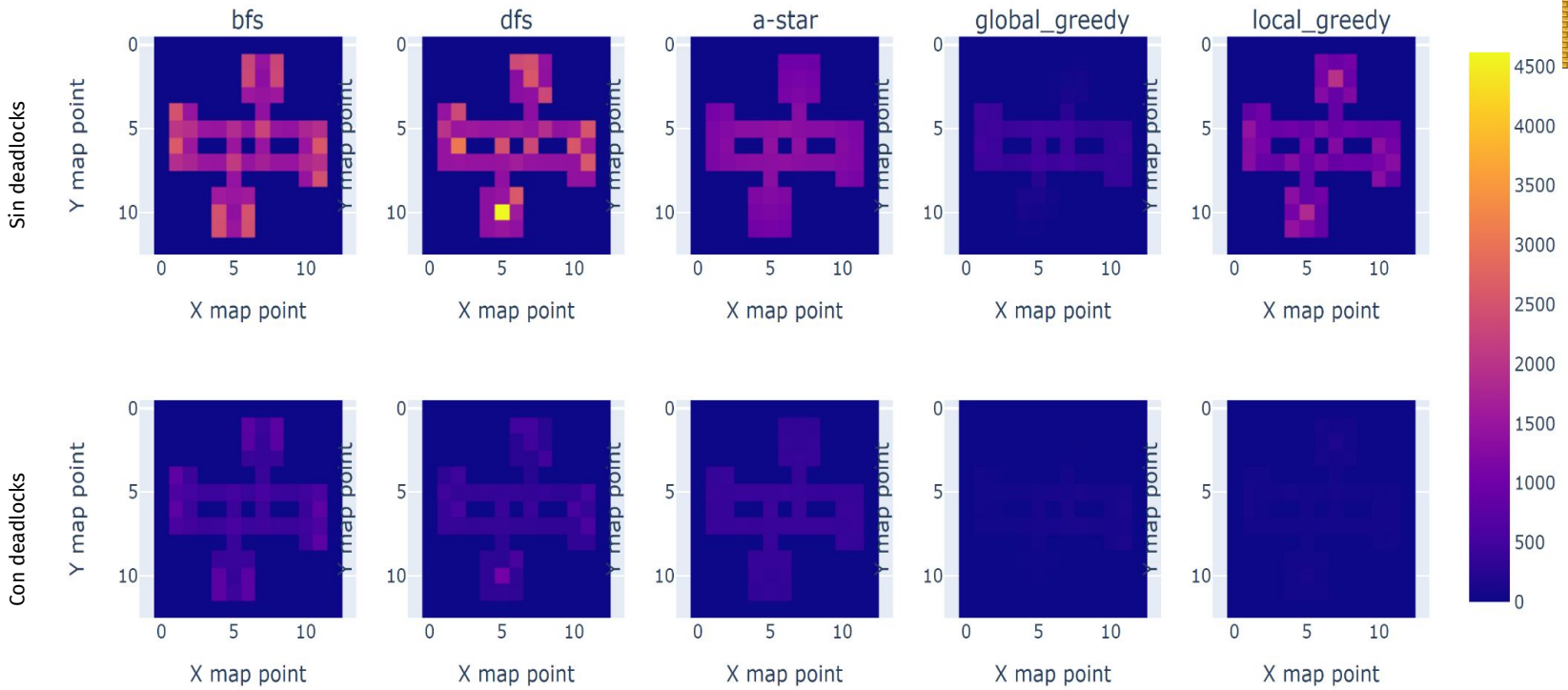
Heatmap de posiciones del jugador

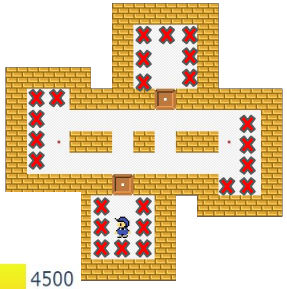




Level 2: MinDistance

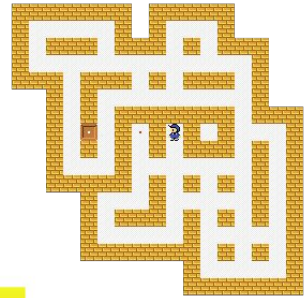
Heatmap de posiciones del jugador

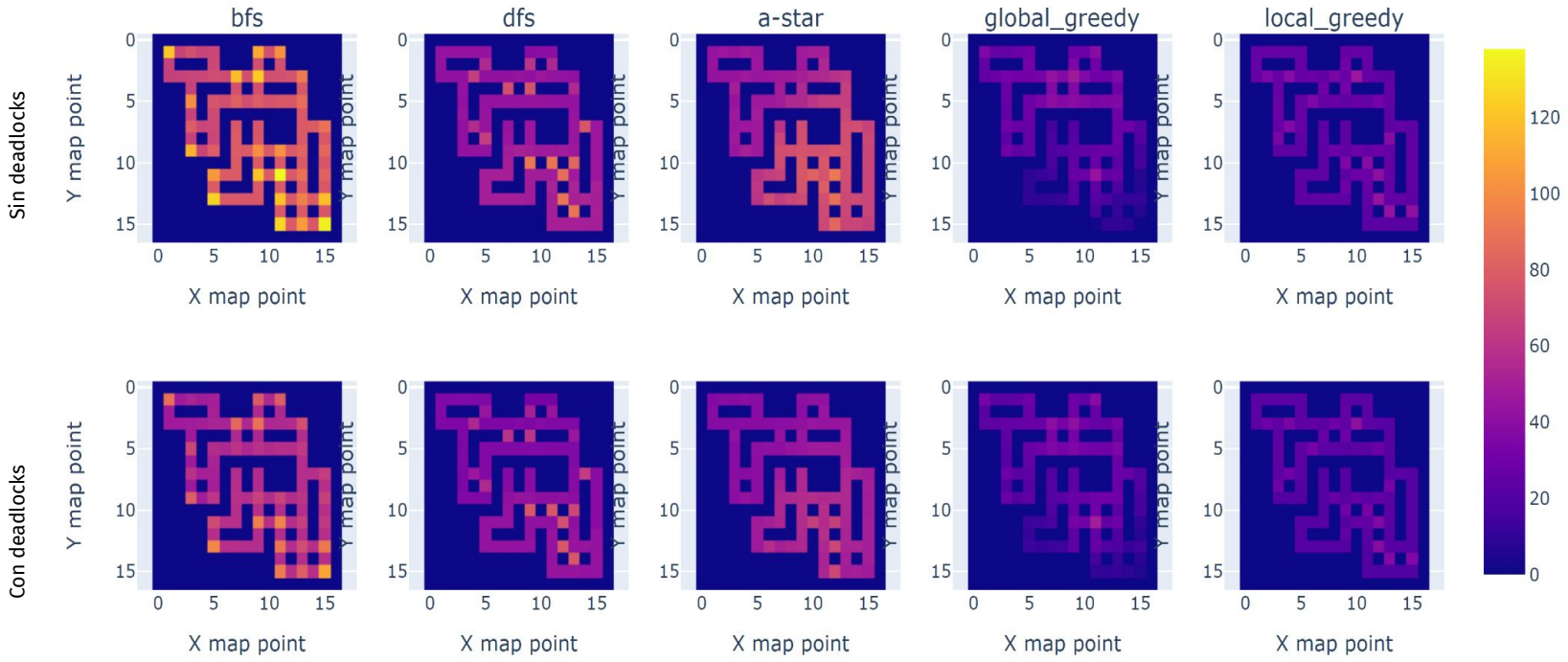




Level 3: MinDistance

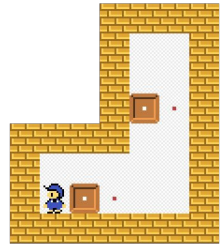
Heatmap de posiciones del jugador

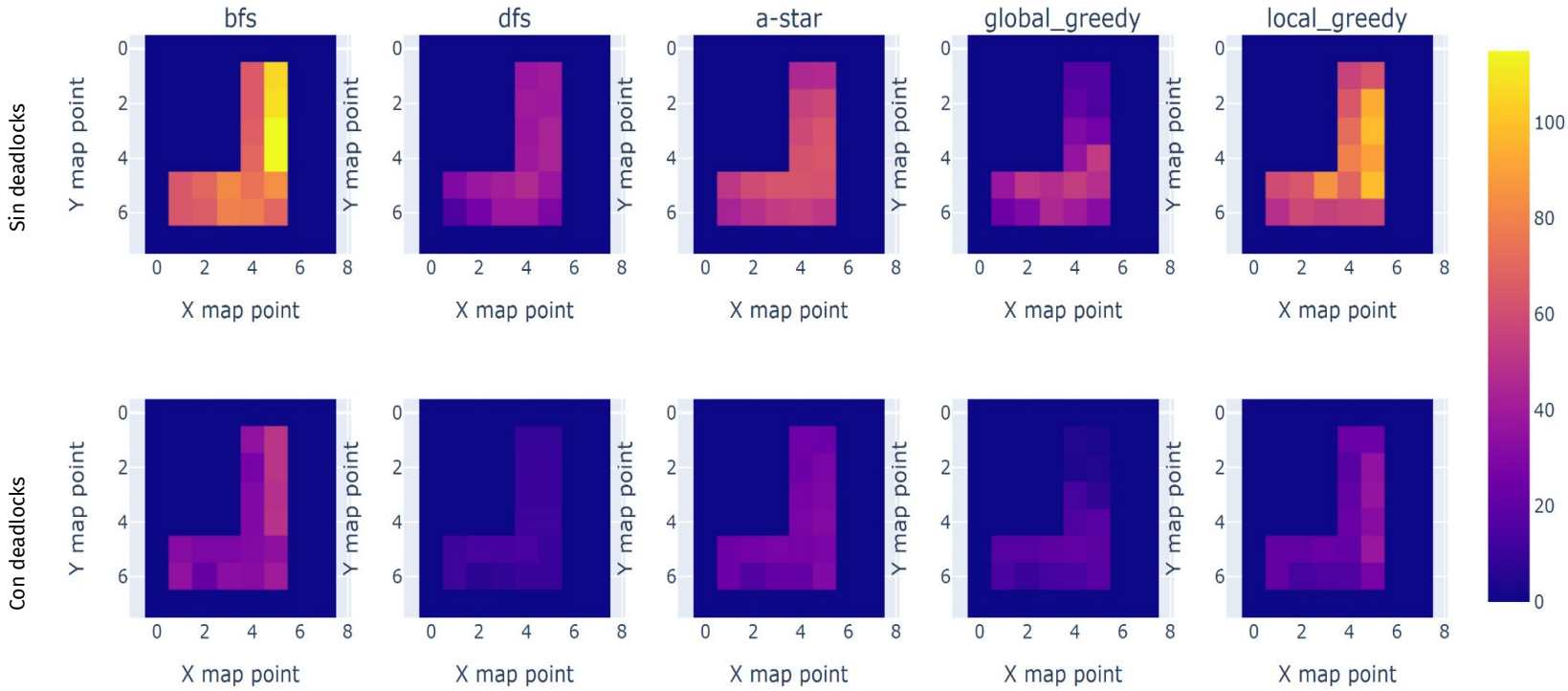




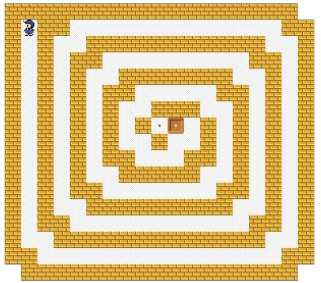
Level 4: MinDistance

Heatmap de posiciones del jugador



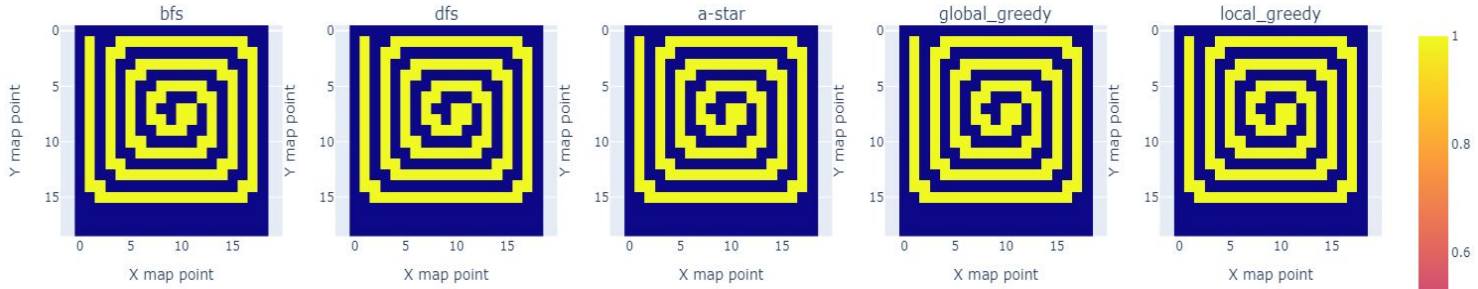


Level 5: Manhattan, MinDistance, Bipartito

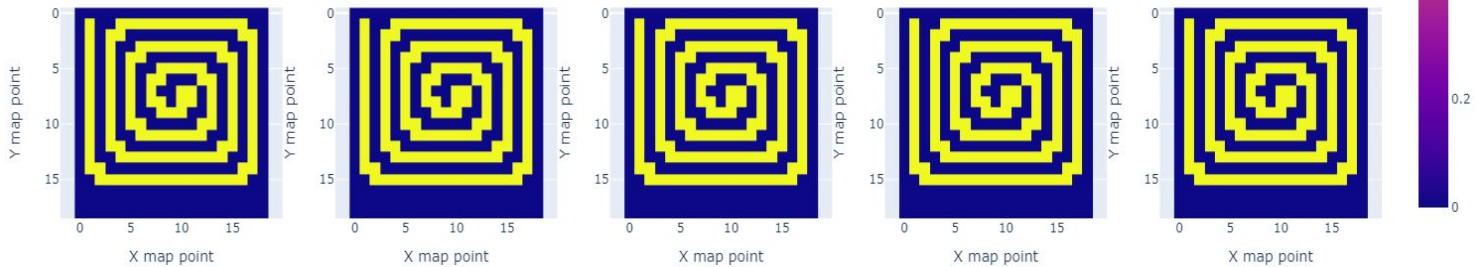


Heatmap de posiciones del jugador

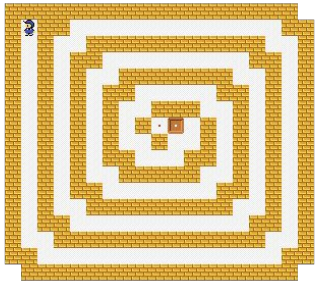
Sin deadlocks



Con deadlocks



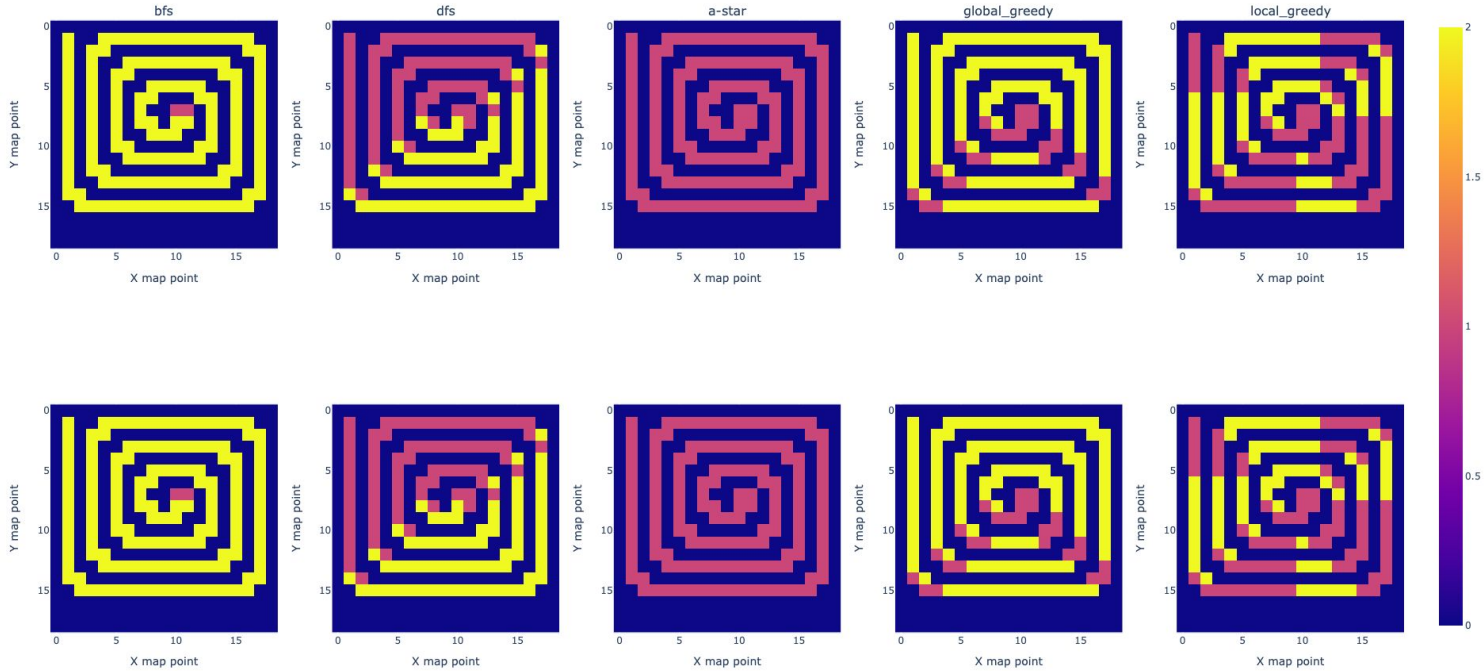
Level 5: Manhattan, MinDistance, Bipartito (sin chequeo de nodos visitados)



Heatmap de posiciones del jugador

Sin deadlocks

Con deadlocks



¡Muchas gracias!