

Estrategias de paralelización del algoritmo A* en entornos de memoria compartida

Autor: Miguel Ángel Rodríguez García

Tutor: A. Javier Cuenca Muñoz

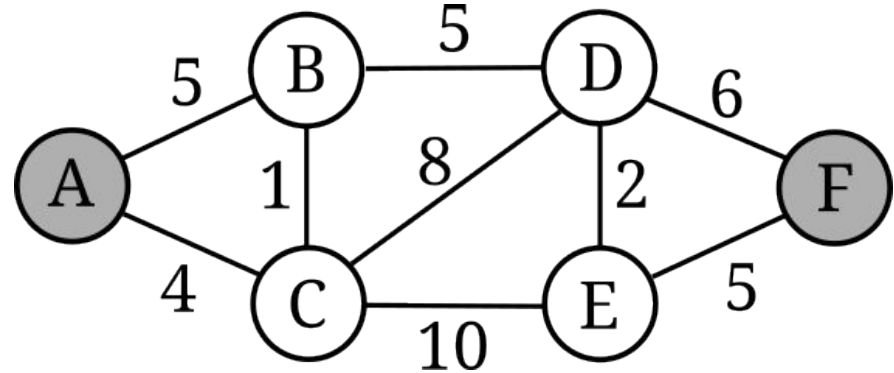
Cotutor: José Matías Cutillas Lozano



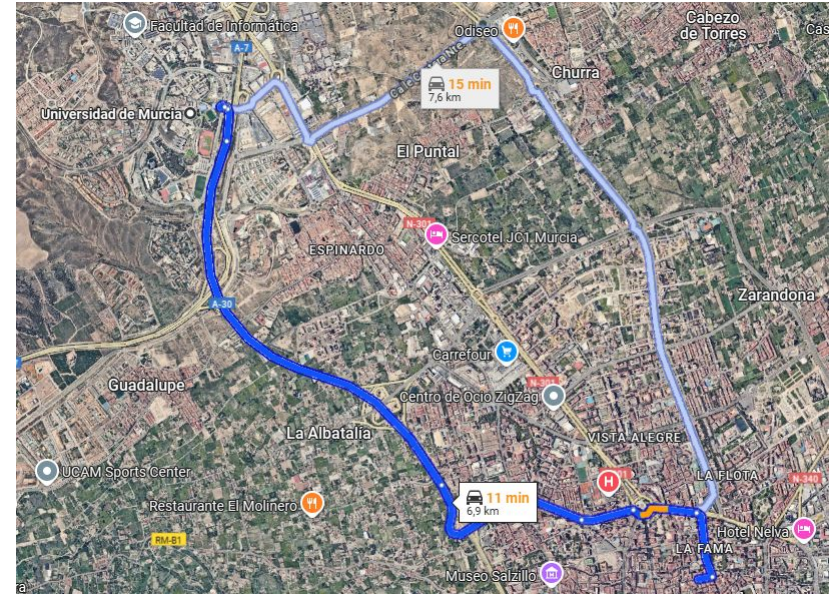
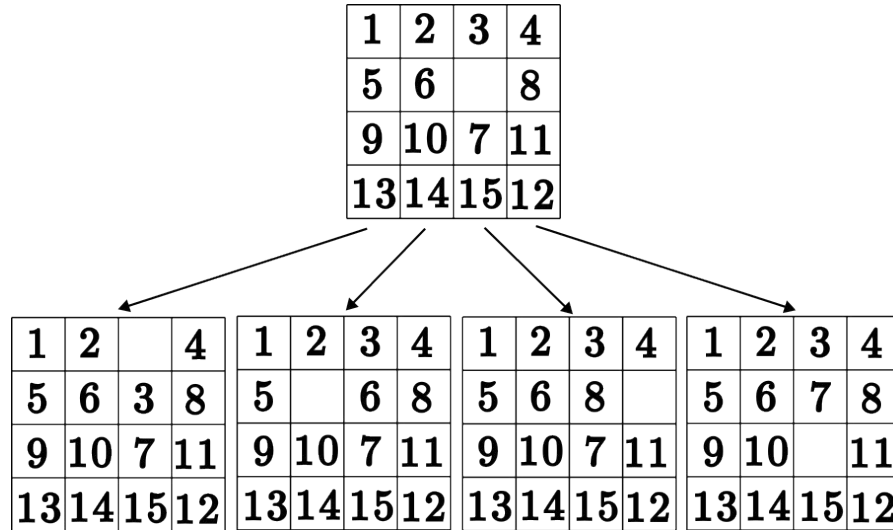
UNIVERSIDAD
DE MURCIA

El problema del camino más corto

Dado un grafo (dirigido o no) con sus aristas ponderadas por pesos, encontrar el camino entre dos vértices de manera que la suma de los pesos sea mínima.



Representación de problemas mediante grafos



Algoritmos no informados vs informados

Los algoritmos **no informados** exploran el espacio de búsqueda de manera sistemática, sin saber *a priori* si el siguiente estado es mejor o peor que el actual.

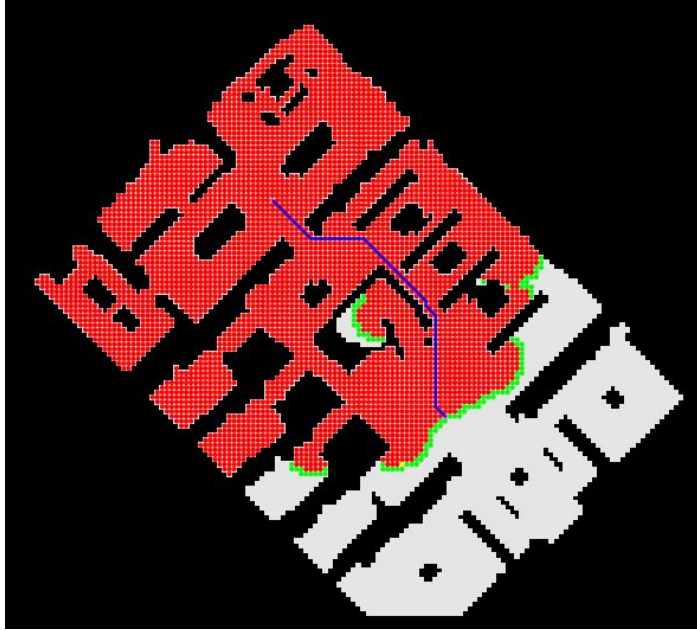
- Búsqueda de coste uniforme
- Dijkstra

Los algoritmos **informados** cuentan con una **función de evaluación** para guiar la búsqueda. Esta función permite evaluar los estados para determinar cuál de ellos es más prometedor.

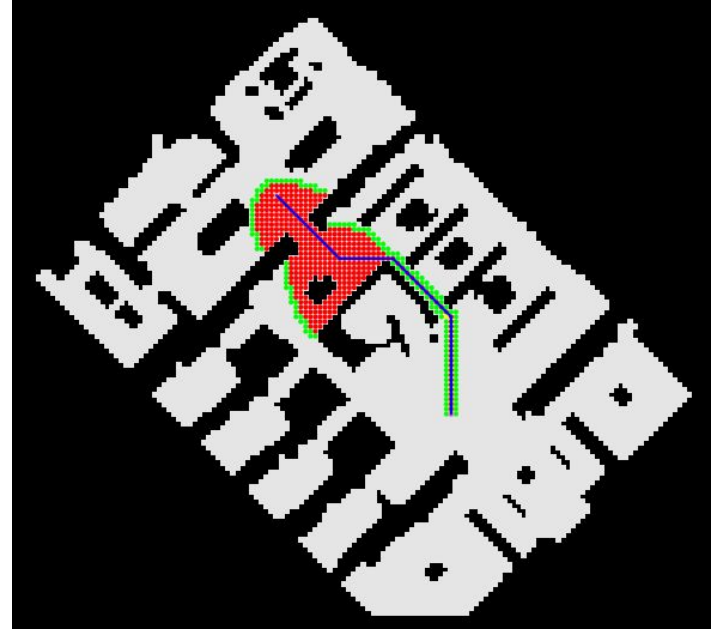
- Algoritmos voraces
- Ascenso de colinas
- A*

Algoritmos no informados vs informados

Dijkstra



A*



Algoritmo A*

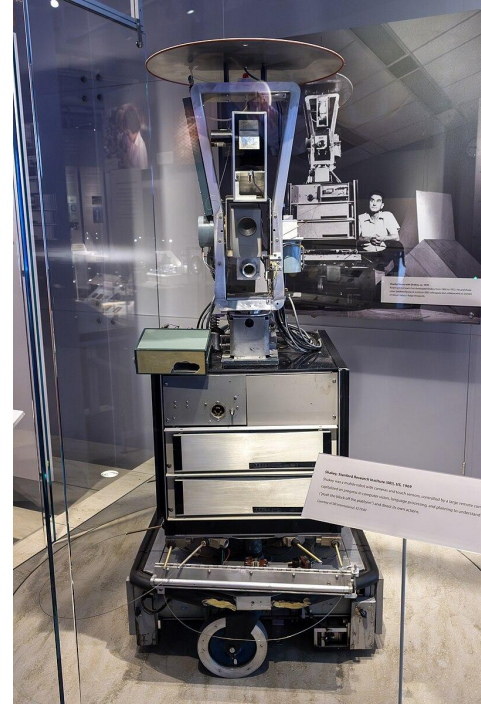
Propuesto en 1968 por **Peter Hart**, **Nils Nilsson** y **Bertram Raphael** como parte del proyecto Shakey.

Combinan la búsqueda de coste uniforme con la búsqueda primero el mejor

$$f(n) = g(n) + h(n)$$

Coste acumulado
desde el inicio

Coste hasta el
objetivo



Algoritmo A*

Para encontrar el camino más corto entre un nodo s y un nodo t , el algoritmo opera de la siguiente manera:

1. Marcar s como abierto y calcular $\hat{f}(s)$.
2. Seleccionar el nodo abierto n con el mínimo valor de $\hat{f}(n)$.
3. Si $n = t$, marcar n como cerrado y terminar el algoritmo.
4. Si no, marcar n como cerrado y para cada uno de sus vecinos n' :
 - a. Si n' no está marcado como cerrado, calcular $\hat{f}(n')$ y marcarlo como abierto.
 - b. Si n' está marcado como cerrado, y el nuevo coste $\hat{f}(n')$ es menor que el que tenía cuando fue cerrado, volver a marcarlo como abierto.

Algoritmo A*

Dos estructuras de datos principales:

- **Lista abierta:** mantiene los nodos que aún no han sido expandidos.
- **Lista cerrada:** mantiene los nodos expandidos.

No se conocen los valores reales de $g(n)$ y $h(n)$, se usan estimaciones.



Es **admisible** si y sólo si $\hat{h}(n) \leq h(n)$ (heurística **admisible**).








Es **óptimo** si y sólo si $\hat{h}(n) \leq c(n, n') + \hat{h}(n')$ (heurística **consistente**).

Ejemplo de A^*

	A	B	C	D	E
1					
2					
3					
4					
5					

Ejemplo de A*

-  Lista abierta
-  Lista cerrada

	A	B	C	D	E
1					
2					
3					
4					
5					

Ejemplo de A*

- Lista abierta
- Lista cerrada

	A	B	C	D	E
1					
2					
3					
4					
5					

Ejemplo de A*

- Lista abierta
- Lista cerrada

	A	B	C	D	E
1	■	←	↙		
2	↑	↖	←	■	
3		■	■	■	
4		■			
5					■

Ejemplo de A*

- Lista abierta
- Lista cerrada

	A	B	C	D	E
1	■	←	↙		
2	↑	↖	←	■	
3		■	■	■	
4		■			
5					■

Ejemplo de A*

- Lista abierta
- Lista cerrada

	A	B	C	D	E
1	■	←	←		
2	↑	↖	←	■	
3		■	■	■	
4		■			
5					■

Ejemplo de A*

- Lista abierta
- Lista cerrada



	A	B	C	D	E
1	■	←	←		
2	↑	↖	←	■	
3	↑	■	■	■	
4		■			
5					■




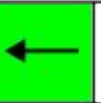







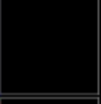





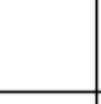
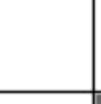






Ejemplo de A*

- Lista abierta
- Lista cerrada

	A	B	C	D	E
1	■	←	←		
2	↑	↖	←	■	
3	↑	■	■	■	
4	↑	■			
5					■

Ejemplo de A*

-  Lista abierta
-  Lista cerrada

	A	B	C	D	E
1					
2					
3					
4					
5					

Ejemplo de A*

- Lista abierta
- Lista cerrada



	A	B	C	D	E
1	■	←	←	←	←
2	↑	↖	←	■	□
3	↑	■	■	■	□
4	↑	■	□	□	□
5	□	□	□	□	■





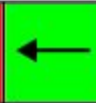








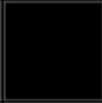







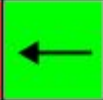



Ejemplo de A*

- Lista abierta
- Lista cerrada

	A	B	C	D	E
1	■	←	←	←	→
2	↑	↖	←	■	□
3	↑	■	■	■	□
4	↑	■	□	□	□
5	→	□	□	□	■

Ejemplo de A*

-  Lista abierta
-  Lista cerrada



	A	B	C	D	E
1					
2					
3					
4					
5					





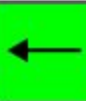











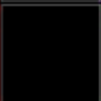
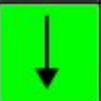







Ejemplo de A*

- Lista abierta
- Lista cerrada



	A	B	C	D	E
1		←	←	←	←
2	↑	↖	←		
3	↑				
4	↑				
5	↑	←	←		





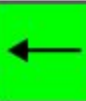











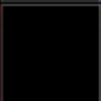
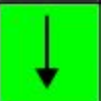






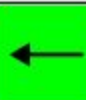
Ejemplo de A*

-  Lista abierta
-  Lista cerrada



	A	B	C	D	E
1					
2					
3					
4					
5					





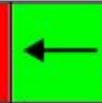







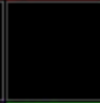









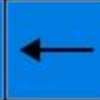

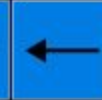
Ejemplo de A*

-  Lista abierta
-  Lista cerrada

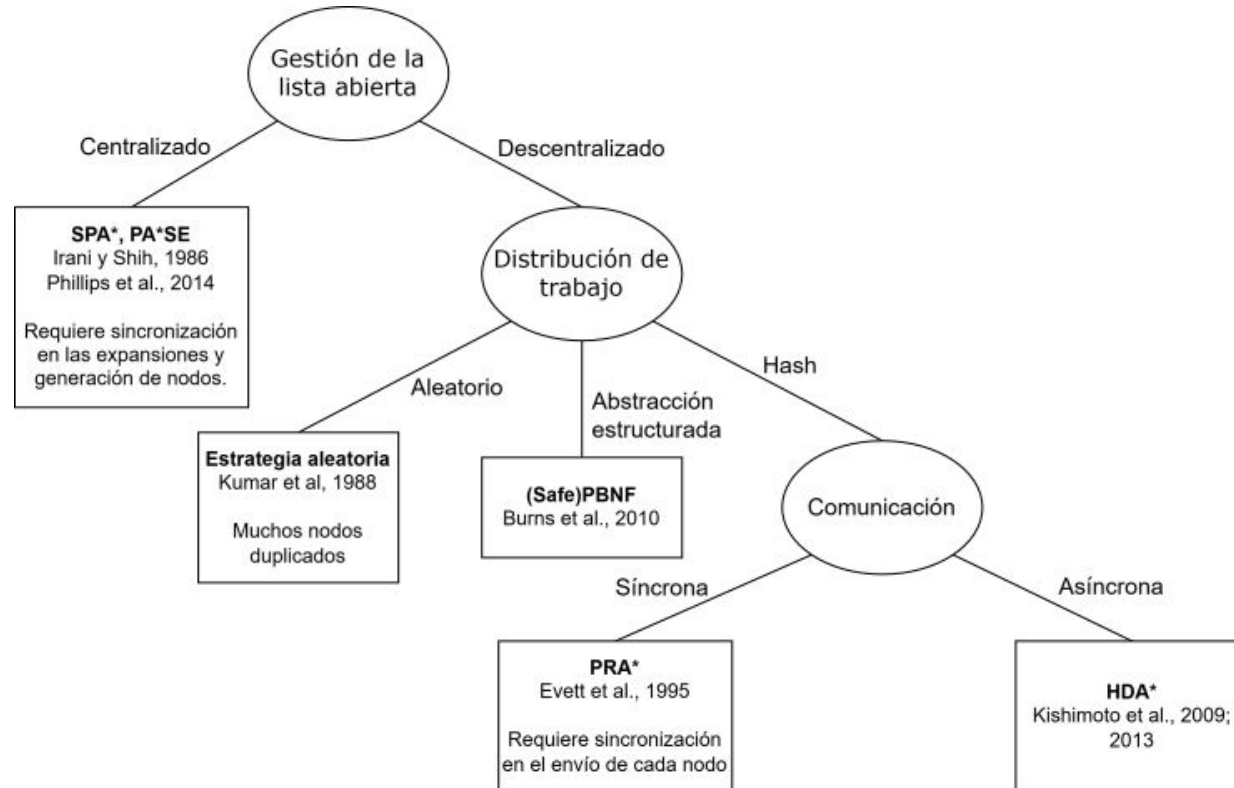
	A	B	C	D	E
1					
2					
3					
4					
5					

Ejemplo de A*

-  Lista abierta
-  Lista cerrada

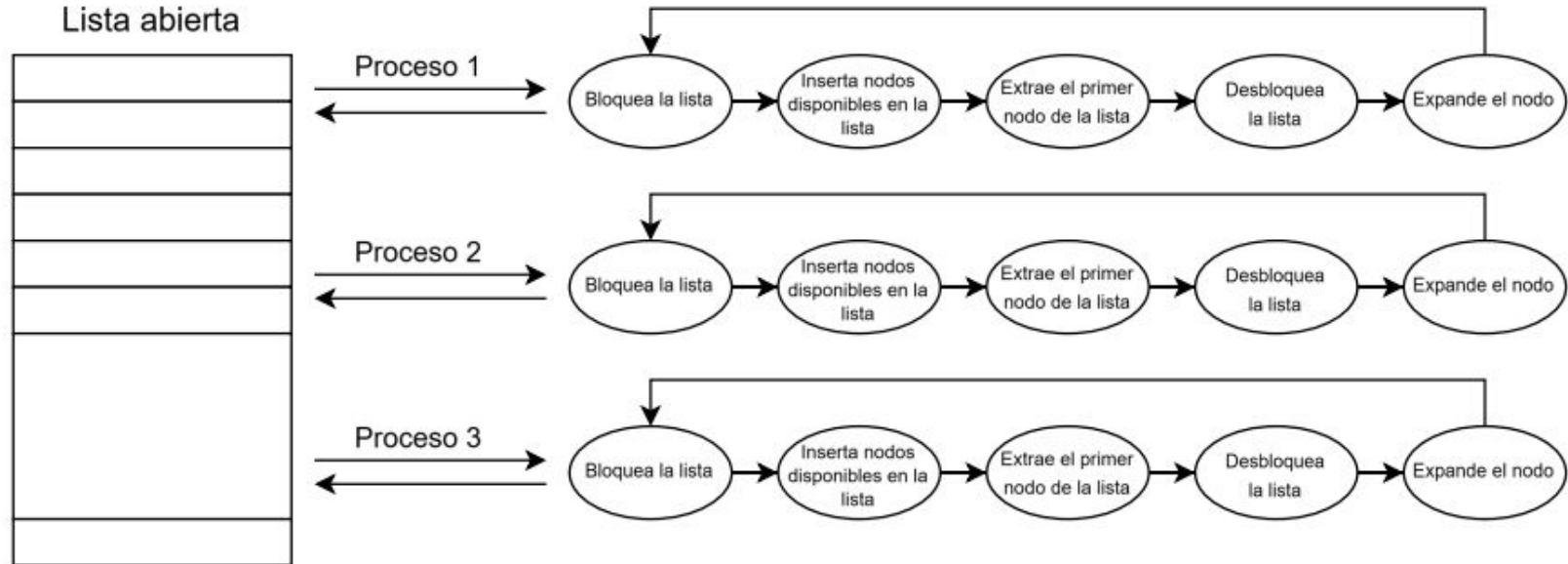
	A	B	C	D	E
1					
2					
3					
4					
5					

Estrategias de paralelización



Enfoque centralizado

La lista abierta es **compartida entre procesos**. Se necesitan mecanismos de **sincronización** para evitar condiciones de carrera.





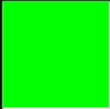


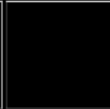



Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada

	A	B	C	D	E
1	■				
2				■	
3		■	■	■	
4		■			
5					■

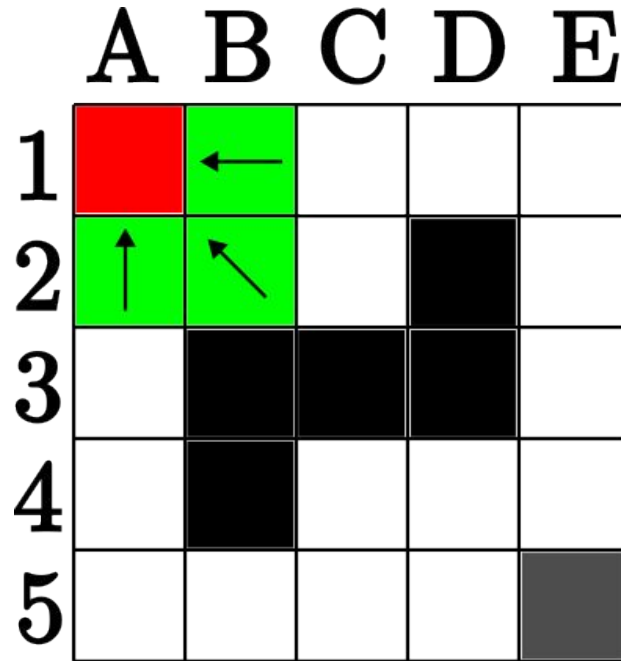
Ejemplo de A* centralizado (dos procesos)

-  Lista abierta
-  Lista cerrada

	A	B	C	D	E
1					
2					
3					
4					
5					

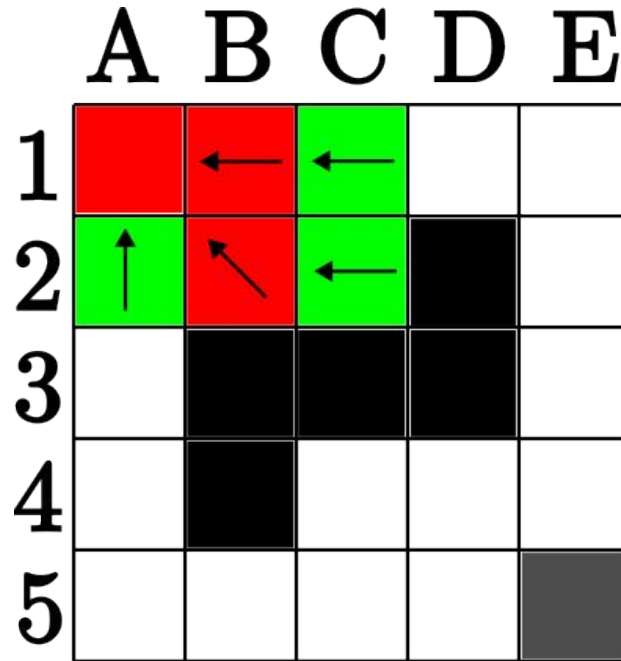
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



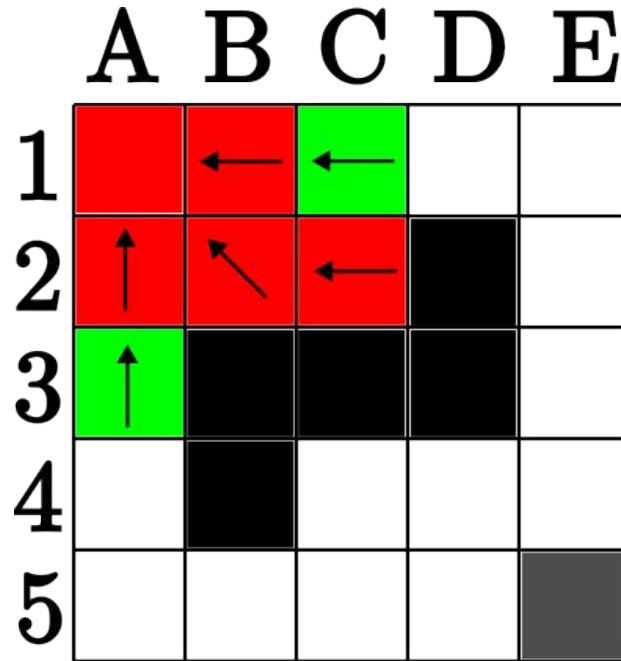
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



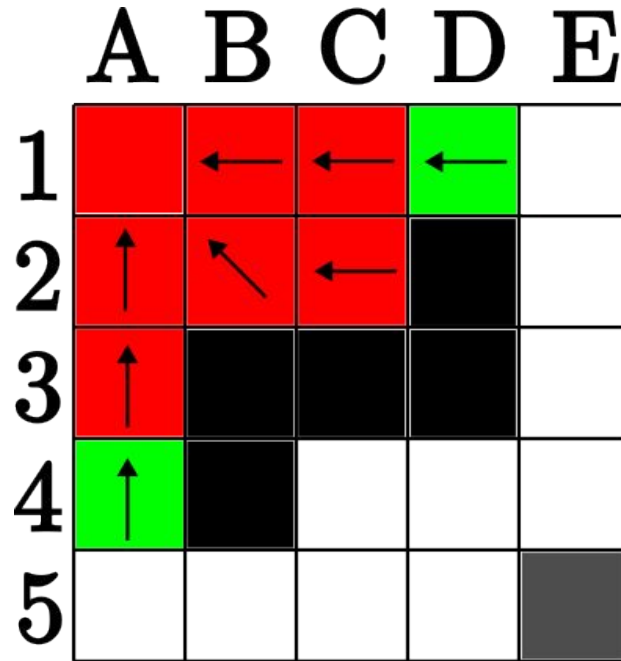
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



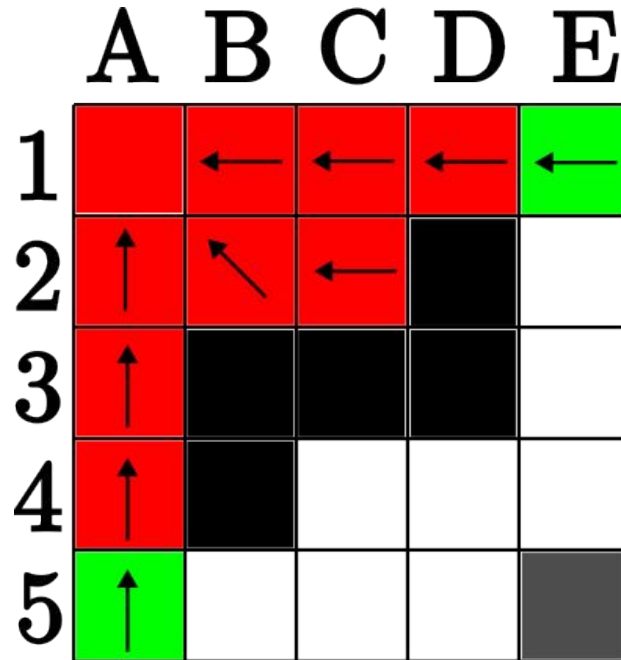
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



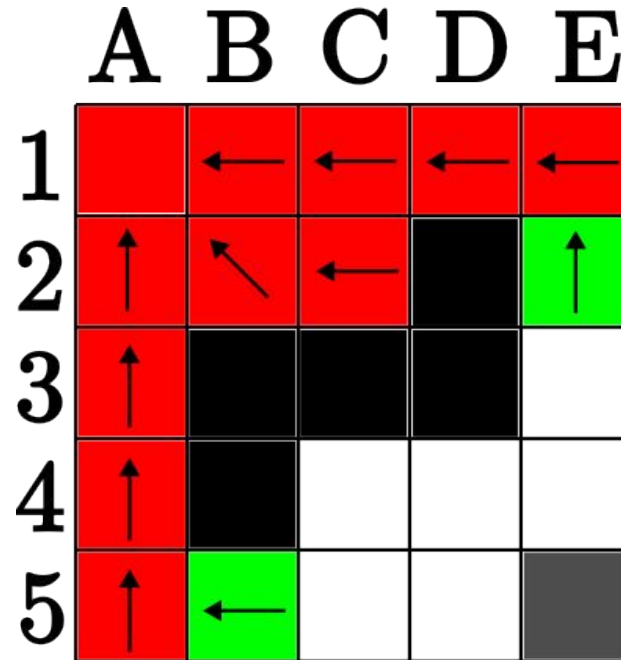
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



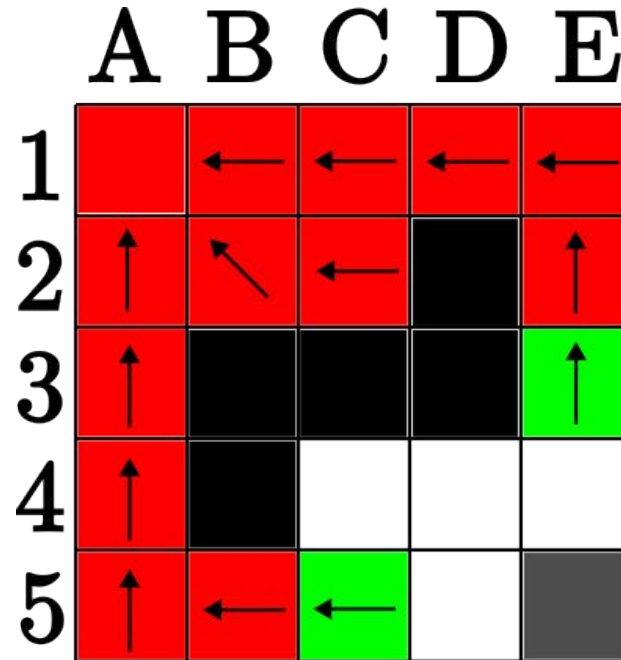
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



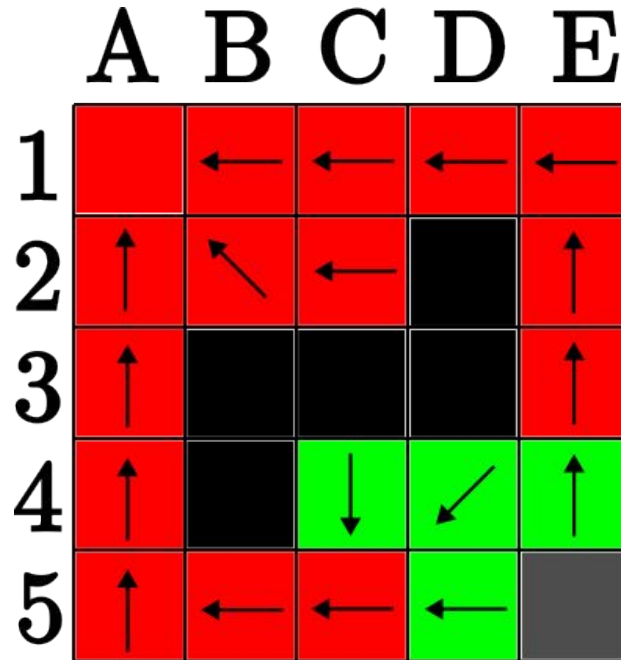
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



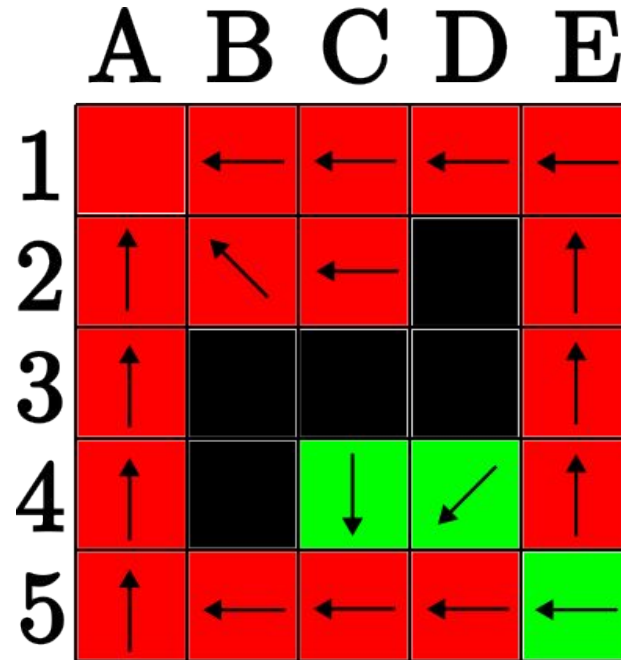
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



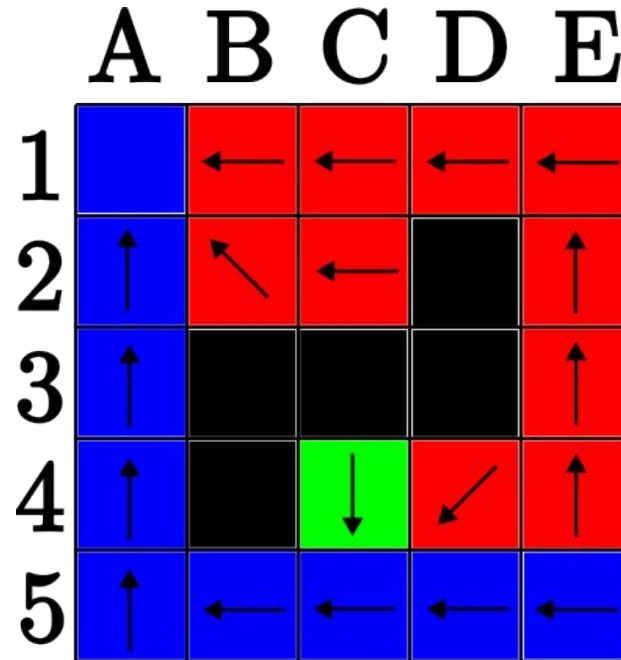
Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



Ejemplo de A* centralizado (dos procesos)

- Lista abierta
- Lista cerrada



Enfoque centralizado

Ventajas:

- Implementación sencilla.

Desventajas:

- Contención en las listas abierta y cerrada.
- Requiere sincronización en las expansiones para mantener la condición de terminación.

Enfoque descentralizado

Cada proceso tiene **su propia lista abierta**.

Al expandir un nodo, se debe calcular a qué proceso enviar cada uno de los vecinos (**distribución de trabajo**):

- De forma aleatoria
- Abstracción estructurada
- Hash

La condición de terminación deja de funcionar: un proceso puede llegar al objetivo por un camino no óptimo.

Enfoque descentralizado.

Hash Distributed A* (HDA*)


Propuesto por Kishimoto, Fukunaga y Botea en 2009. Es un versión paralela de A* simple y escalable.

Características principales:



- Distribución de trabajo basada en **hash**.
- Comunicación entre procesos de manera **asíncrona**.



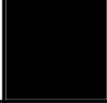
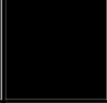



Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada



	A	B	C	D	E
1					
2					
3					
4					
5					

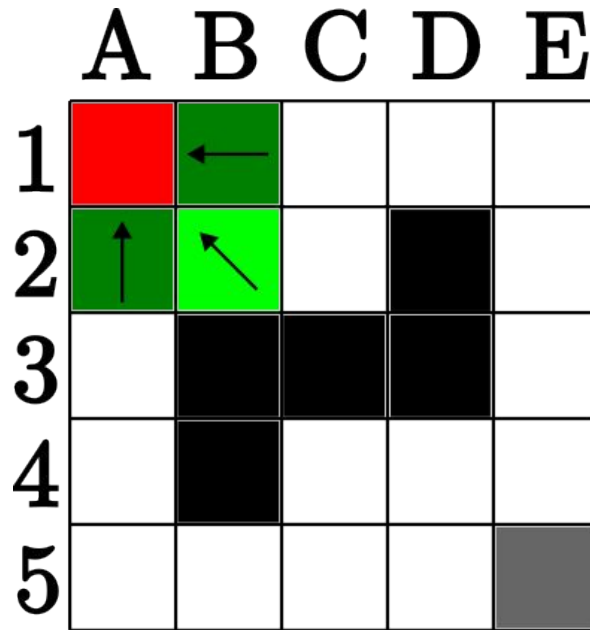
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





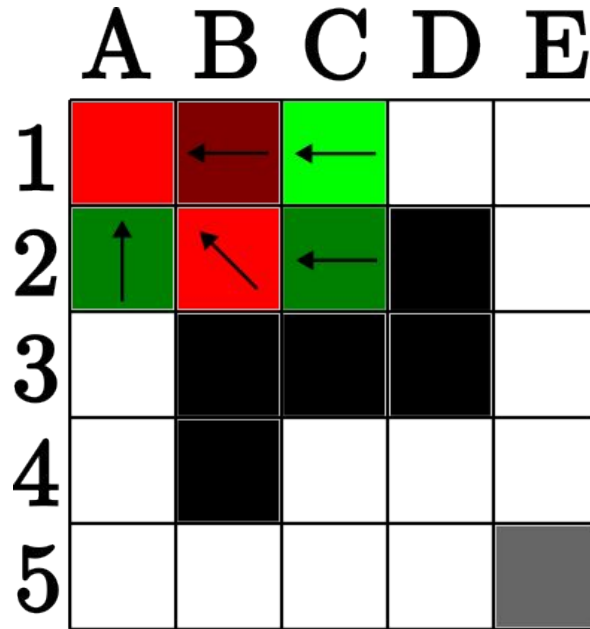
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





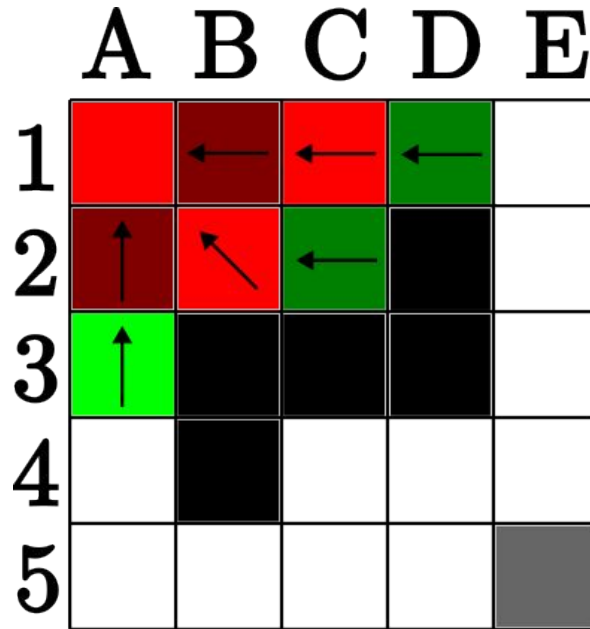
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





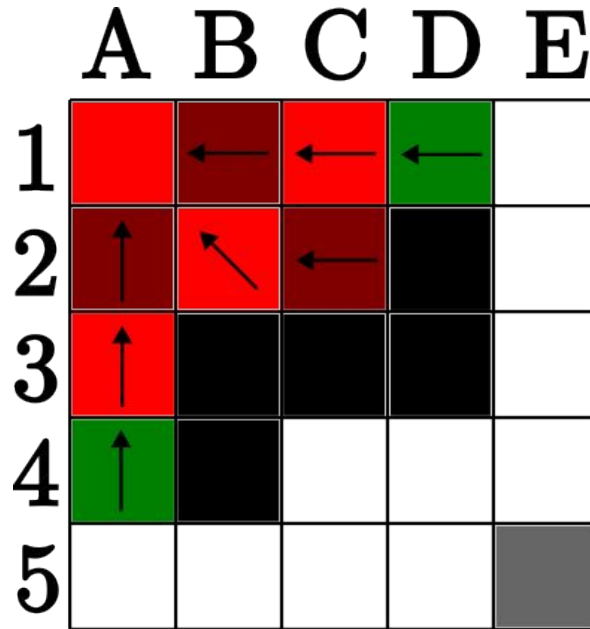
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





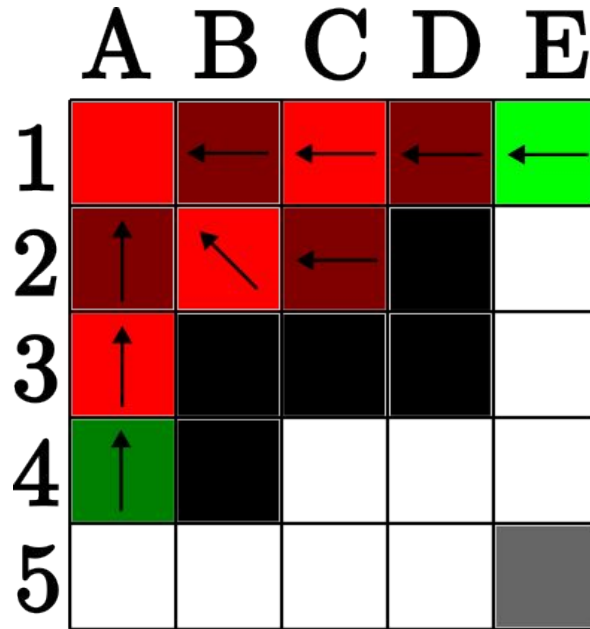
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





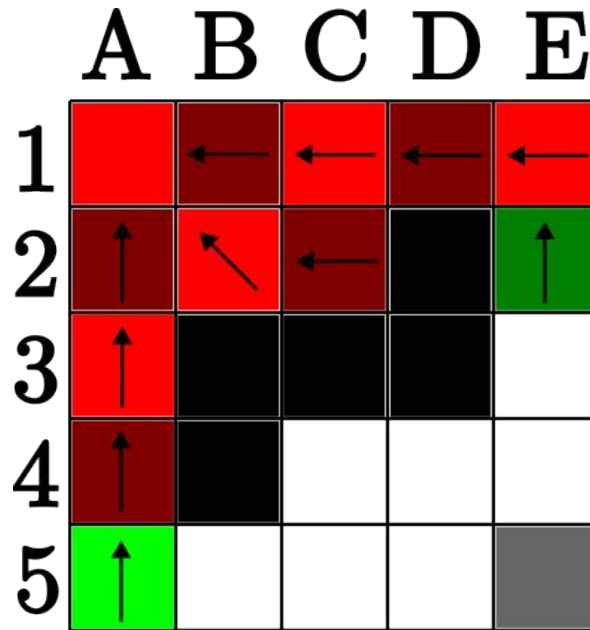
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





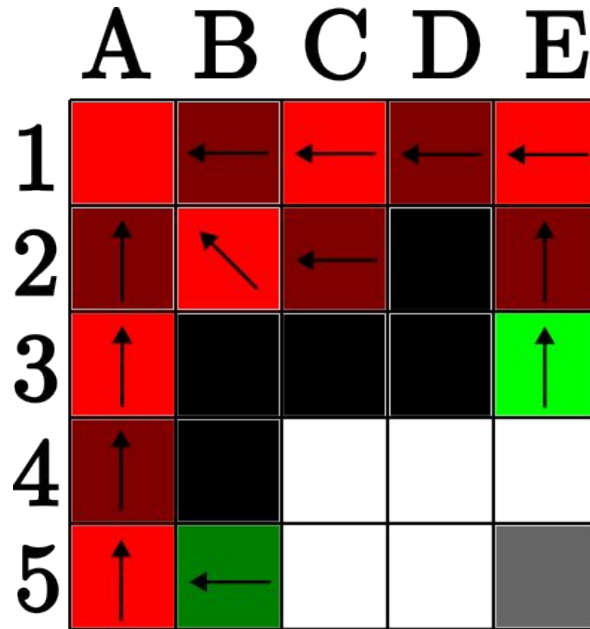
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





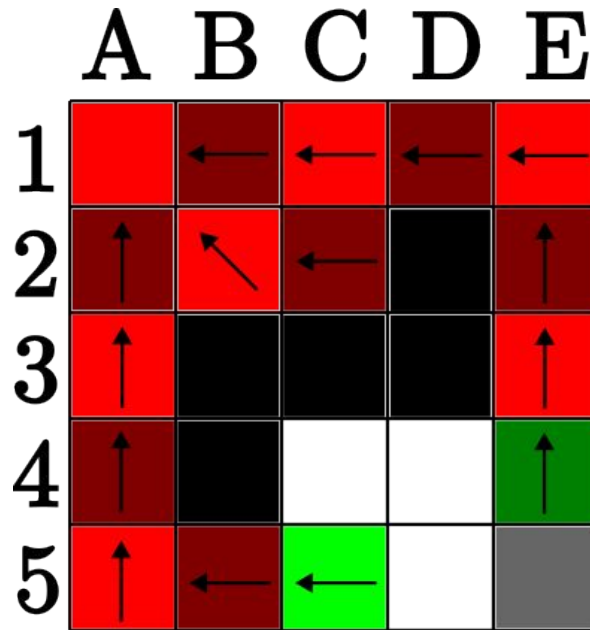
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





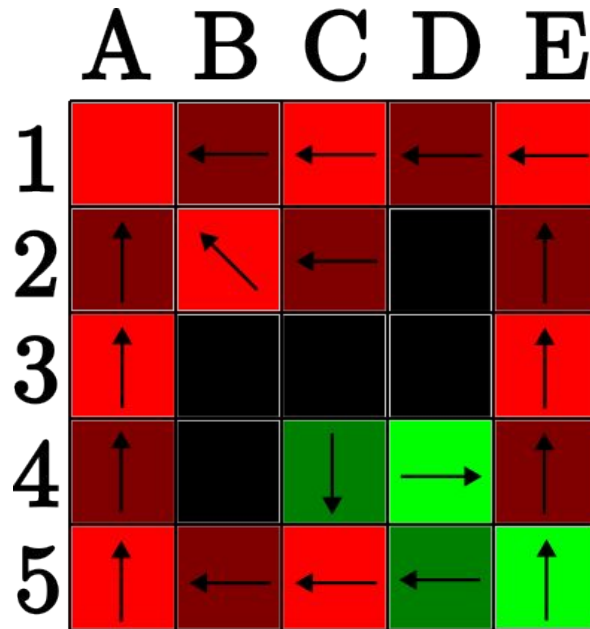
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada



Proceso 2:

-  Lista abierta
-  Lista cerrada





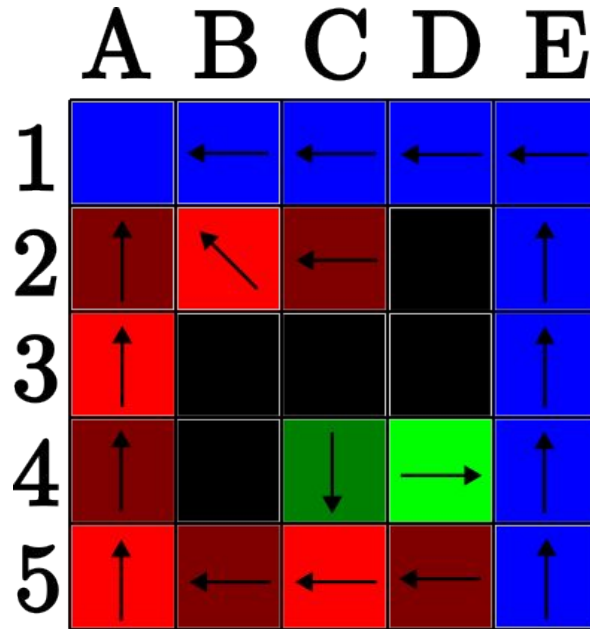
Ejemplo HDA*

Proceso 1:

-  Lista abierta
-  Lista cerrada

Proceso 2:

-  Lista abierta
-  Lista cerrada



Hash Distributed A*

Ventajas:

- Escalable.
- Distribución de trabajo balanceada.
- No hay contención en las listas abiertas ni en la lista cerrada.

Desventajas:

- Es necesario cambiar la detección de terminación.
- El rendimiento depende mucho de la función hash utilizada.

HDA* en memoria compartida

Inicialmente pensado para implementarse en un entorno distribuido.

En este trabajo se ha adaptado a un entorno de memoria compartida.

Simulamos la comunicación asíncrona mediante ***buffers de envío y recepción***.

Se define un parámetro **threshold** para evitar bloqueos innecesarios.

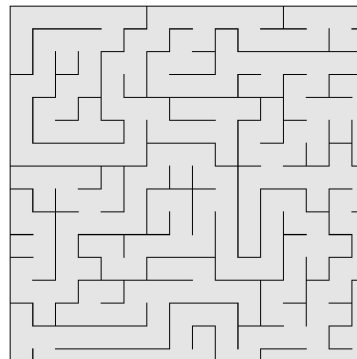
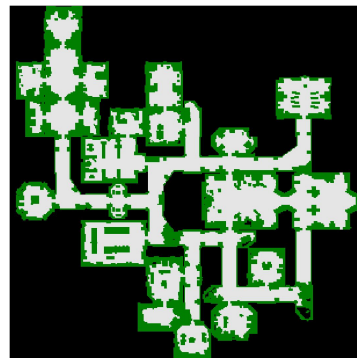
Evaluación de los algoritmos

Tres versiones:

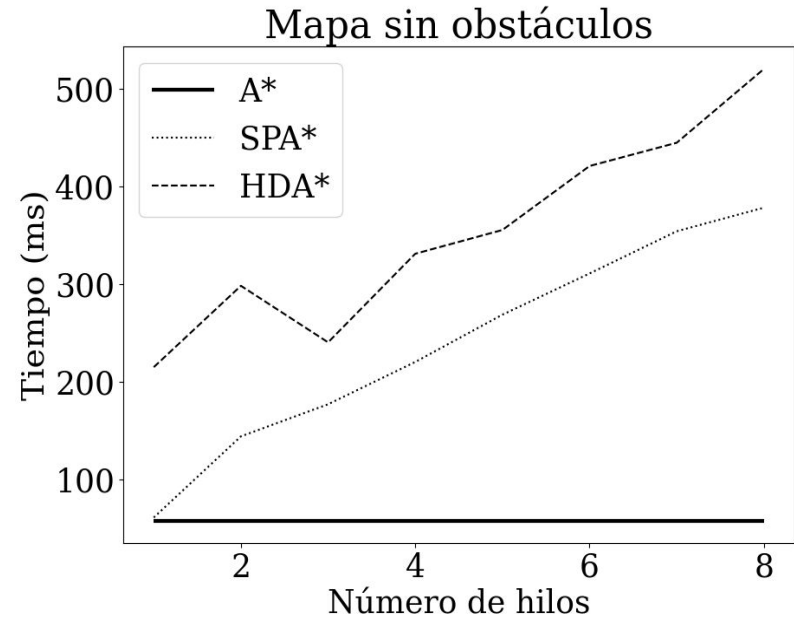
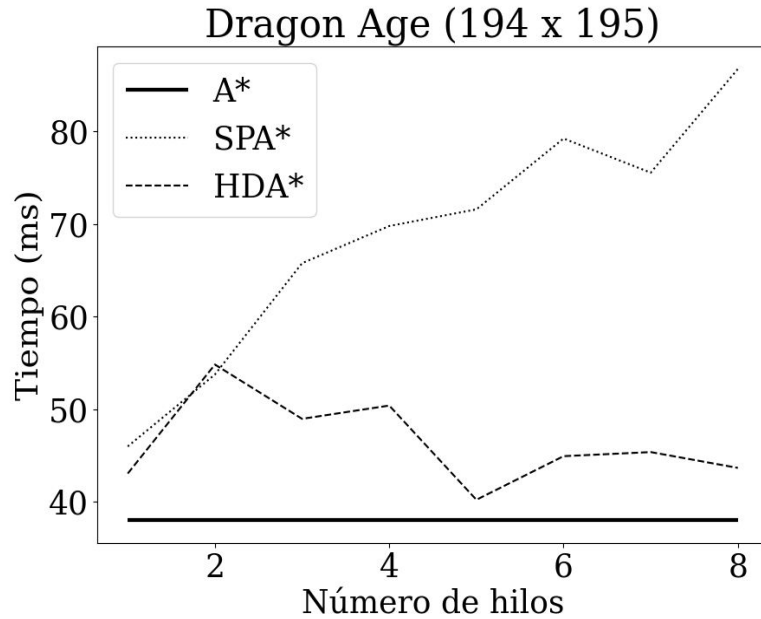
- Secuencial
- Paralela centralizada (SPA*)
- Paralela descentralizada (HDA*)

Implementadas en un entorno de **memoria compartida** utilizando OpenMP para gestionar el paralelismo entre **hilos**.

Mapas obtenidos de <https://movingai.com/>

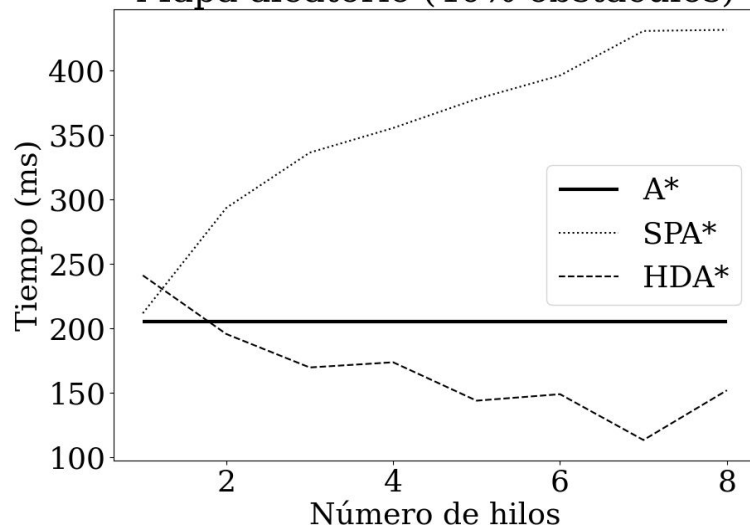


Evaluación de los algoritmos

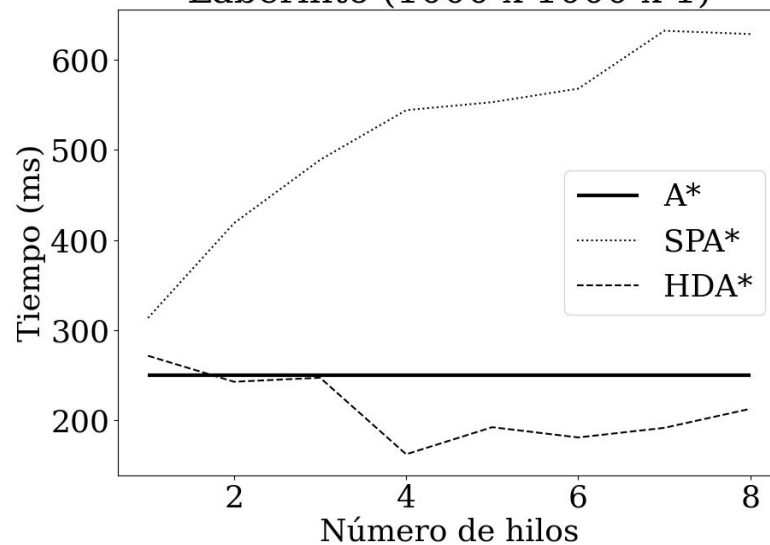


Evaluación de los algoritmos

Mapa aleatorio (40% obstáculos)

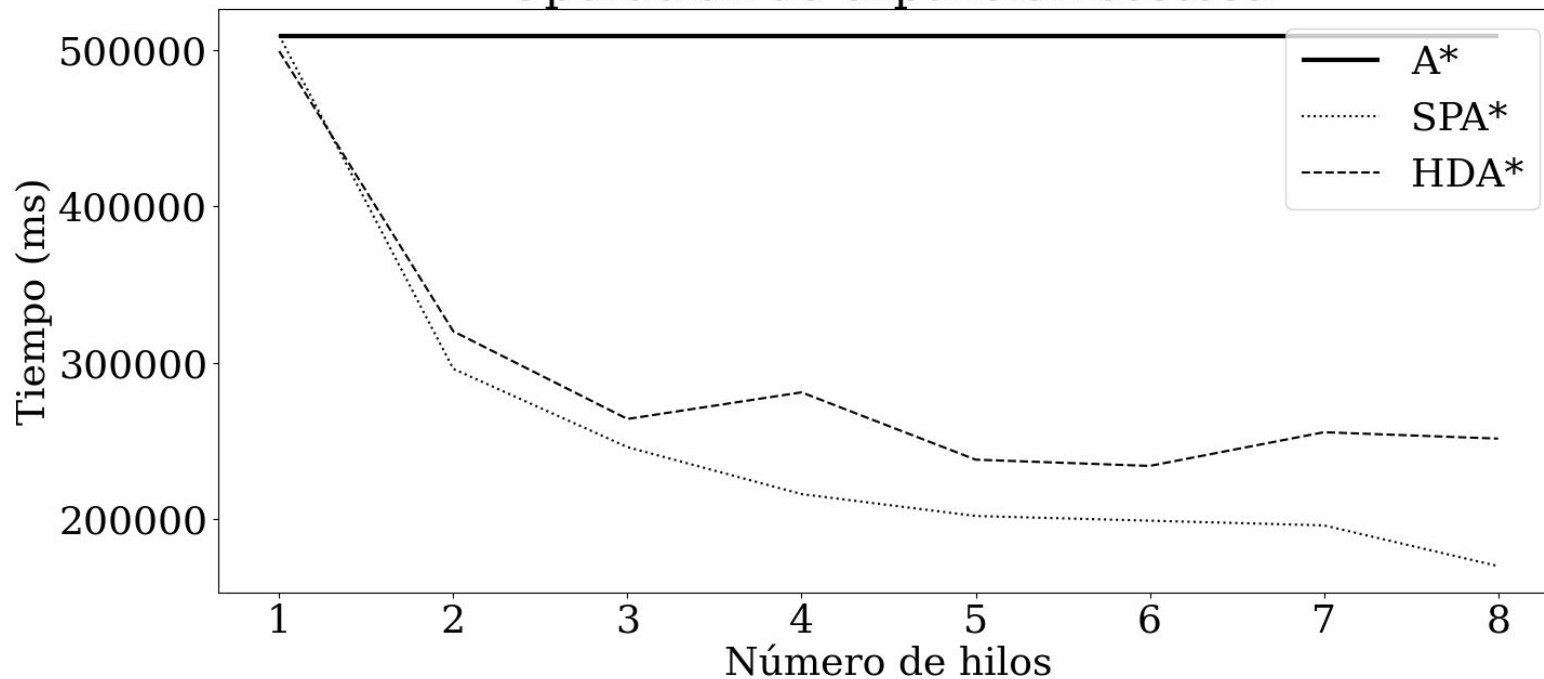


Laberinto (1000 x 1000 x 1)

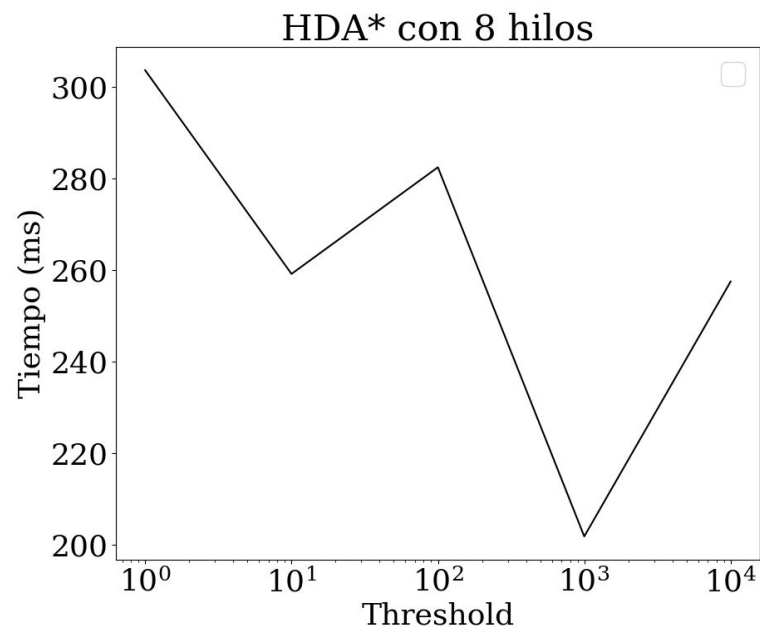
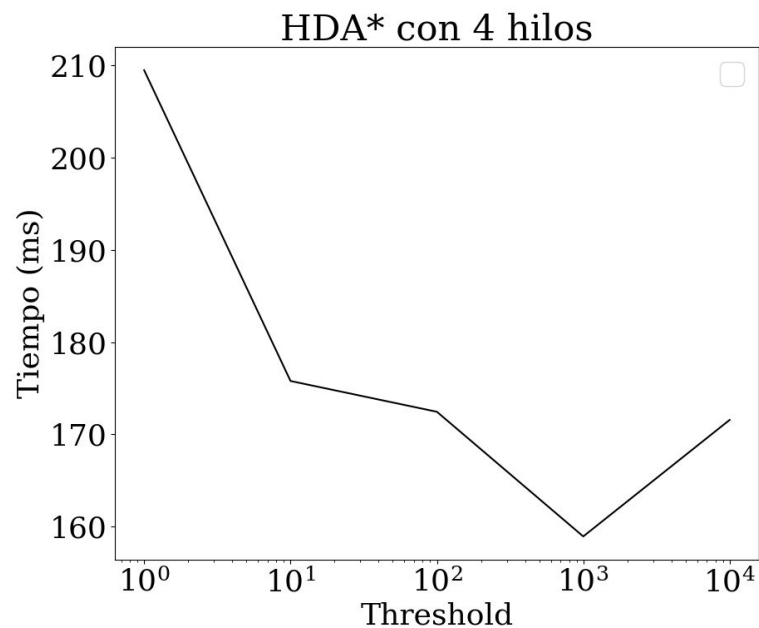


Evaluación de los algoritmos

Operación de expansión costosa



Evaluación de los algoritmos



Conclusiones

- La versión descentralizada consigue mejores resultados en general que la versión centralizada, sobre todo al aumentar la complejidad del problema. También presenta una mejor escalabilidad.
- En problemas en los que la operación de expansión resulte costosa, la estrategia centralizada aún puede ser una alternativa viable.
- En la implementación propuesta de HDA* se detecta un mínimo en el valor del parámetro **threshold**.

Trabajo futuro

- Analizar la eficiencia de las implementaciones propuestas en términos de uso de memoria y expansión de nodos para evaluar cuantitativamente la sobrecarga introducida en la búsqueda.
- Investigar el efecto del parámetro **threshold** en la versión de HDA* en memoria compartida para detectar el mínimo en función de las características del problema.
- Explorar las estrategias descentralizadas en entornos de memoria distribuida utilizando paso de mensajes.
- Estudiar las nuevas propuestas basadas en arquitecturas GPU.

Computación cuántica

Es un área de investigación activa.

No existe como tal una propuesta cuántica del algoritmo A^* .

Se han propuesto algoritmos cuánticos para acelerar la estrategia de backtracking (Montanaro, 2016) utilizando caminatas cuánticas.

Más recientemente (Li, 2025) se ha propuesto un algoritmo cuántico para resolver este problema en grafos soldados de manera eficiente, y se ha probado que ningún algoritmo clásico puede resolverlo en tiempo subexponencial.