



UNIVERSIDAD  
DE GRANADA



# Técnicas de los Sistemas Inteligentes

Grado en Informática

**Curso 2021-22. Práctica 1**

**Búsqueda en tiempo real: RTA\***

Jesús Giráldez Crú y Pablo Mesejo Santiago

Departamento de Ciencias de la  
Computación e Inteligencia Artificial

<http://decsai.ugr.es>

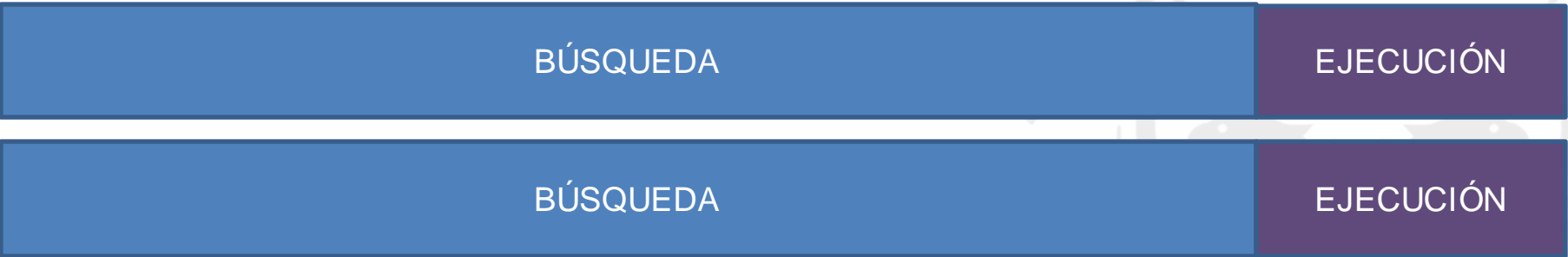
	Algoritmos	Características
<b>Búsqueda no informada</b>	BFS, DFS	Al ser una búsqueda no informada (sin información heurística), son más ineficientes y/o no garantizan que se encuentre el camino óptimo.
<b>Búsqueda heurística (offline)</b>	A*, IDA*	Siempre encuentran el camino óptimo y lo hacen de forma más eficiente que la búsqueda no informada (debido al uso de información heurística).
<b>Búsqueda heurística online (incremental o en tiempo real)</b>	D*, RTA*	Aunque no garantizan el camino óptimo, permiten enfrentarse a problemas más complejos (entornos desconocidos y/o dinámicos, movimientos rápidos, etc).

	Algoritmos	¿Funciona en entornos desconocidos?	¿Es capaz de realizar movimientos rápidos (tiempo corto)?
<b>Búsqueda heurística offline</b>	A*, IDA*	<b>NO</b>	<b>NO</b>
<b>Búsqueda incremental</b>	D*, LPA*, D*Lite	<b>SÍ</b>	<b>NO</b>
<b>Búsqueda en tiempo real</b>	RTA*, LRTA*	<b>SÍ</b>	<b>SÍ</b>

## Búsqueda offline



## Búsqueda incremental



■ ■ ■

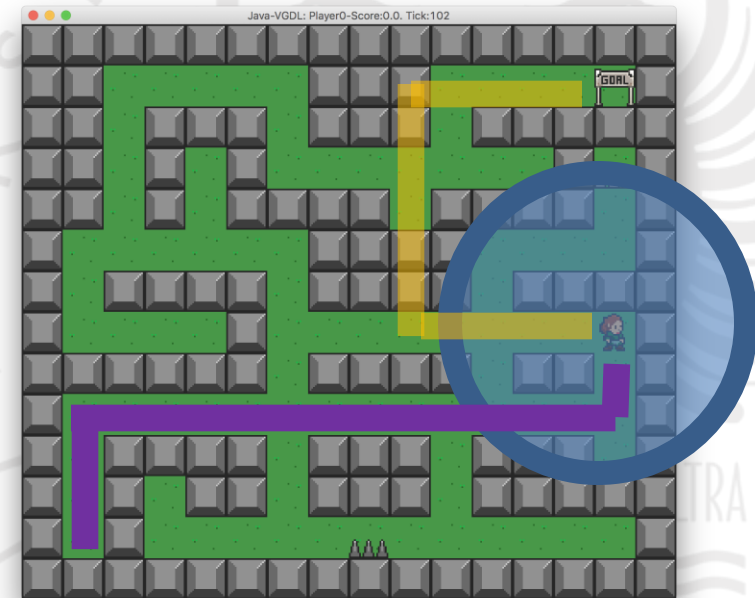
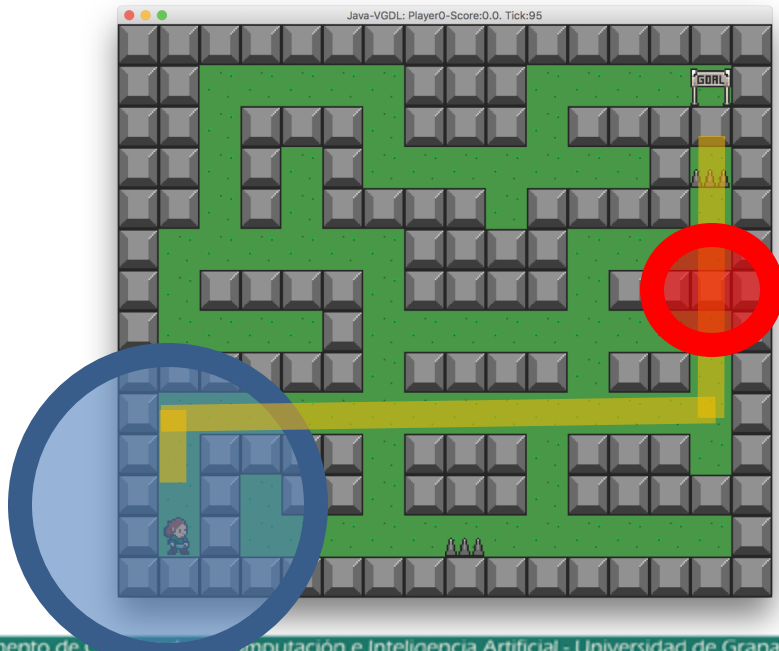
## Búsqueda en tiempo real



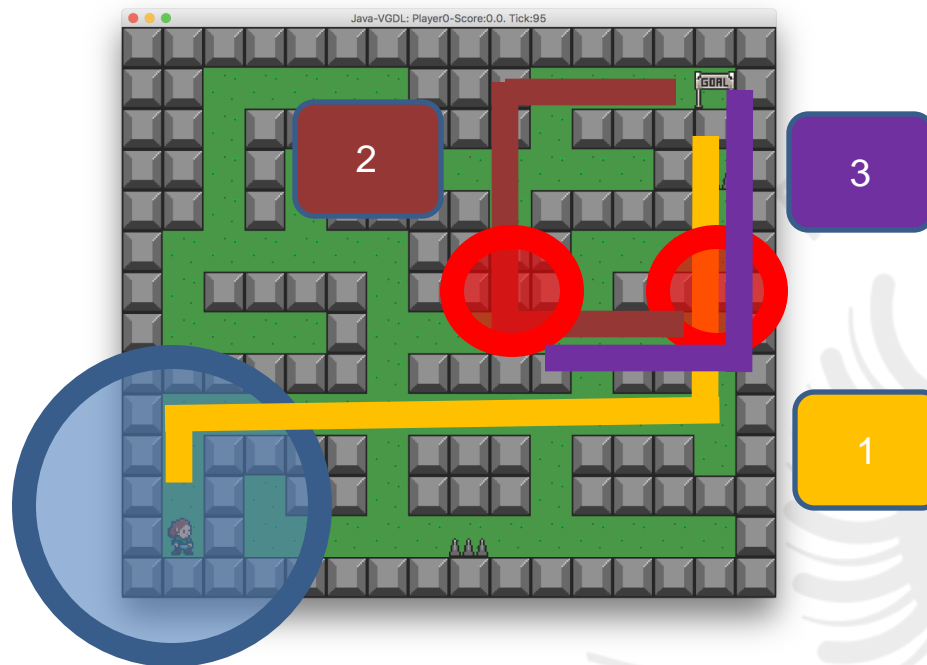
■ ■ ■

- La búsqueda heurística clásica (offline) requiere **información completa del entorno**
- Con esta información, primero se realiza la búsqueda de la ruta (normalmente óptima) –proceso más costoso– y posteriormente se ejecuta –proceso más rápido–.
- Pero en algunos contextos, el entorno sólo se conoce parcialmente:
  - **Información desconocida.** Ej: alcance de los sensores del robot
  - **Información dinámica:** Ej: obstáculos que se mueven por el entorno
- En estos casos, es **inviable calcular UN único plan** (como se hace en la búsqueda heurística clásica), dado que éste puede sufrir modificaciones
- Es necesario **replanificar!**

- La **búsqueda incremental** aborda el problema de la información incompleta o dinámica
  - Zona de visibilidad del agente** (ZVA): área abarcada por los sensores
  - Hipótesis del mundo desconocido**: todo terreno desconocido es navegable.
- Ejemplo: un método incremental basado en A\*. Calcular un plan, intentar ejecutar, y, en caso de obstáculo, recalcular



- Problema: ¿cómo **actualizar la información heurística** a medida que el agente se mueve por el mapa?
  - Si cada llamada a A\* es “independiente”, no se actualiza
  - El agente puede caer en **ciclos infinitos**



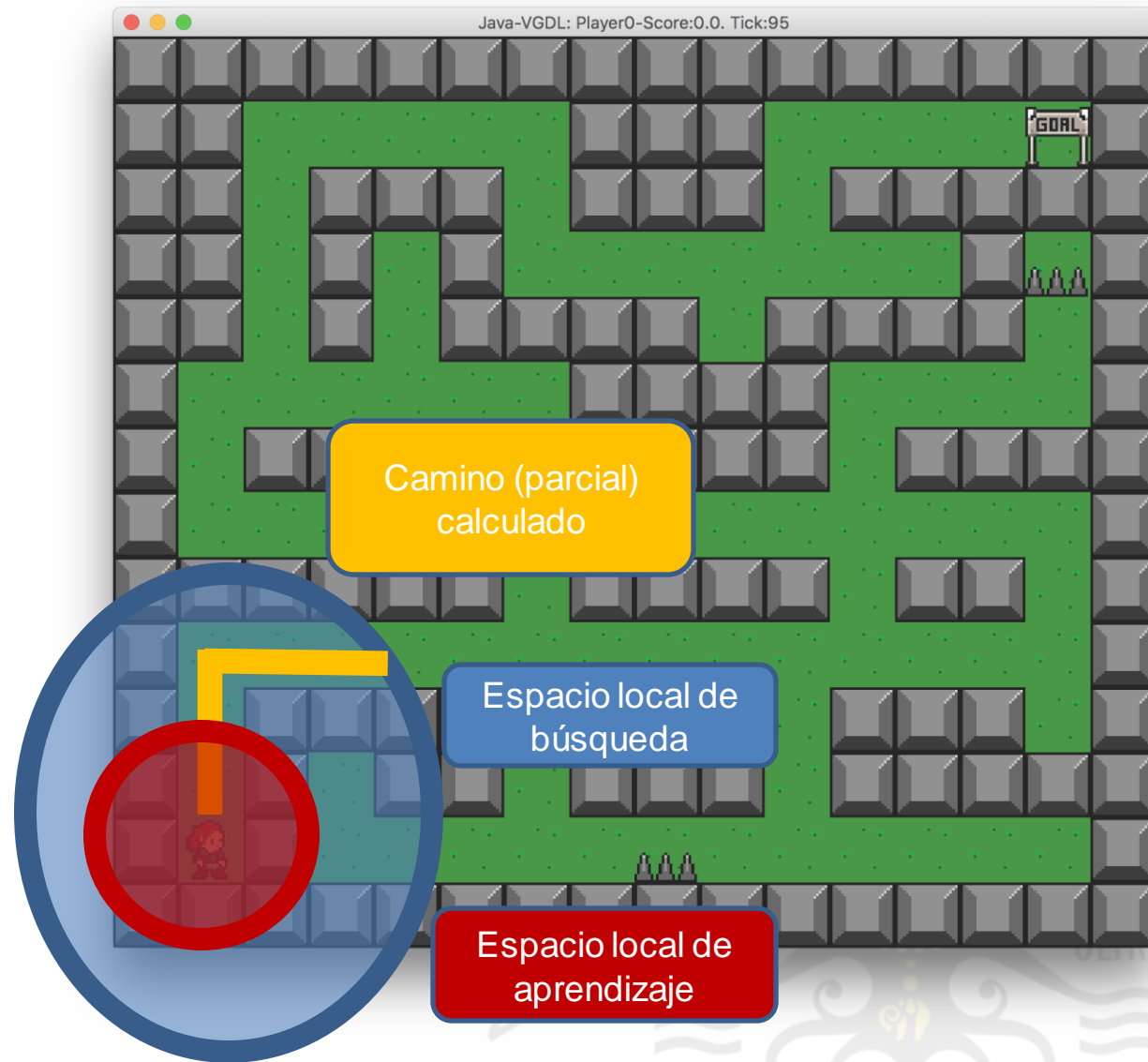
- La **búsqueda incremental** aborda este problema

- Algoritmos de **búsqueda incremental**: D\*, LPA\*, D\*Lite, ...
- La idea general es mantener, de la forma más eficiente posible, **información actualizada sobre  $h(n)$** , a medida que el agente aprende nueva información sobre el mapa en los distintos intentos de ejecutar la ruta
- Inconveniente: **en el peor caso**, la búsqueda de un camino es tan **costosa** como en búsqueda offline
  - Los tiempos de planificación (búsqueda) pueden ser muy altos
  - Por tanto, **no sirve para realizar movimientos rápidos**
- Ante la necesidad de movimientos/decisiones rápidas aparece la **búsqueda en tiempo real**

LPA\*: [Koenig, Likhachev, Furcy. 2004. Lifelong Planning A\*. Artificial Intelligence 155 (1–2): 93-146]  
 D\*: [Stentz. 1994. Optimal and Efficient Path Planning for Partially-Known Environments. Proc. of ICRA]  
 D\*Lite: [Koenig, Likhachev. 2005. Fast Replanning for Navigation in Unknown Terrain. Transactions on Robotics 21 (3)]

## Dos conceptos clave:

- **Espacio local de búsqueda:** el agente calcula rutas "cortas" hasta la mejor frontera
- **Espacio local de aprendizaje:** se actualiza la heurística de cara a movimientos futuros
- La **diferencia** con respecto a la **búsqueda local** (GBFS) es precisamente la regla de aprendizaje





## Real-Time A\* (RTA\*)

- En cada paso, moverse al nodo del espacio local de búsqueda seleccionado por la estrategia de movimiento, y actualizar el espacio local de aprendizaje según la regla de aprendizaje

- Espacio local de búsqueda: los vecinos del nodo actual  $x$
- Estrategia de movimiento: mover al mejor vecino  $z$ :

$$z = \operatorname{argmin}_{(y \in \operatorname{Succ}(x))} \{c(x, y) + h(y)\}$$

- Espacio local de aprendizaje: nodo actual
- Regla de aprendizaje:

$$h(x) = \max(h(x), 2^{\circ} \min(c(x, y) + h(y)))$$

En cada iteración se comprueba si el nodo expandido es el nodo objetivo

RTA\*: [Korf. 1990. Real-time Heuristic Search. *Artificial Intelligence* **42**, 189-211]

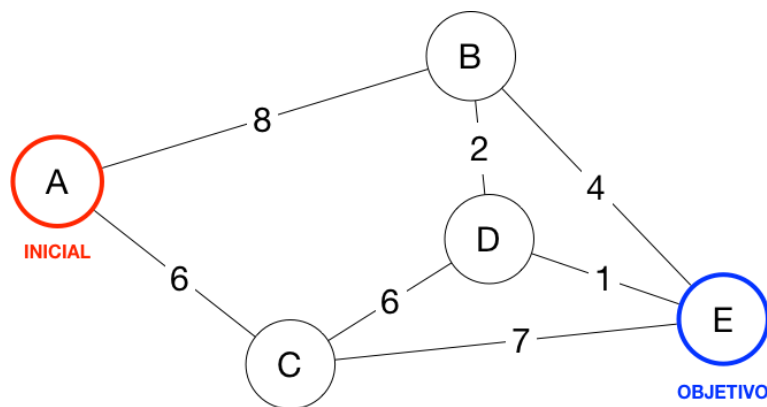
- Los valores de  $h(n)$  son dinámicos, se van aprendiendo a lo largo de la ejecución del algoritmo
- Cuando se visita un nodo por primera vez, se inicializa su valor con alguna función heurística
  - En nuestro caso: distancia Manhattan al objetivo
- Este valor se guarda en una tabla hash
  - Estructura de datos eficiente para acceder y actualizar valores
- Cuando un nodo pertenece al espacio local de aprendizaje, se actualiza su  $h(n)$ 
  - Simplemente se modifica el valor de este nodo en la tabla hash

Consumo de memoria:  
tamaño de la tabla hash  
(número de nodos visitados)

- Espacio local de búsqueda: vecinos del nodo
- Estrategia de movimiento: mejor vecino
  - ARRIBA:  $c(\text{ACTUAL}, \text{ARRIBA}) + h(\text{ARRIBA}) = 1 + 23 = 24$
  - ABAJO: Muro
  - IZQUIERDA: Muro
  - DERECHA: Muro
- Espacio local de aprendizaje: nodo actual
- Regla de aprendizaje:
  - $h(\text{ACTUAL}) = 24$

Como no hay segundo mínimo, elegimos el primero





	A	B	C	D	E
h(n)	10	4	5	1	0

It.	Nodo actual	c(x,y)+h(y) en vecinos del nodo actual					Sig. nodo	Nuevo valor h(actual)				
		y=A	y=B	y=C	y=D	y=E		h(A)	h(B)	h(C)	h(D)	h(E)
0	A	-	12	11	-	-	C	12	=	=	=	=
1	C	17	-	-	7	7	D	=	=	7	=	=
2	D	-	6	13	-	1	E	=	=	=	6	=
3	E	Fin. Camino: E – D – C – A						11	4	7	6	0

Otro algoritmo de búsqueda en tiempo real (**NO usado** en esta práctica): LRTA\*

- La única diferencia entre RTA\* y LRTA\* es la regla de aprendizaje:
  - En RTA\* se usa el segundo mínimo
  - En LRTA\* se usa el (primer) mínimo
- Ambas permiten al agente escapar de mínimos locales en  $h(n)$ 
  - Zonas prometedoras según la  $h(n)$  pero que no llevan a la solución
- RTA\* puede sobreestimar  $h(n)$  por lo que escapa de mínimos locales más rápido
  - En una ejecución, es más rápido
- LRTA\* aproxima mejor  $h(n)$  a  $h^*(n)$  por lo que es más útil si se requieren múltiples ejecuciones en el mismo mapa
  - En múltiples ejecuciones, converge más rápido al camino óptimo

LRTA\*: [Korf. 1990. Real-time Heuristic Search. *Artificial Intelligence* **42**, 189-211]