FORMULARIO EXAMEN FINAL INTELIGENCIA ARTIFICIAL:

BLOQUE 2: BÚSQUEDA

Elementos de un problema de búsqueda: Espacio de estados (estados alcanzables), función sucesora (siguiente acción a realizar), estado inicial y test objetivo.

Tamaño del espacio de estados: T = Posiciones * 2^Objetivos * Dimensiones * Orientaciones

Búsqueda Primero en Profundidad / Depth-First Search (BPP / DFS): Expande el nodo más profundo.

Búsqueda Primero en Anchura / Breadth-First Search (BPA / BFS): Expande el nodo menos profundo.

Profundidad iterativa: Ejecuta con el espacio de la BPP y el tiempo de la BPA.

Búsqueda de Coste Uniforme / Uniform Cost Search (BCU / UCS): Expande el nodo más barato.

Completa: ¿Garantiza encontrar una solución si existe?

Óptima: ¿Garantiza encontrar el camino de menor coste?

Nomenclatura: b (factor de ramificación), m (profundidad máxima), s (profundidad de la solución), C* (coste de la solución), ε (coste de las acciones), C*/ε (profundidad efectiva).

Criterio	BPP	BPA	Profundidad iterativa	BCU
Tiempo	O(b^m)	O(b^s)	O(b^s)	O(b^C*/ε)
Espacio	O(b*m)	O(b^s)	O(b*s)	O(b^C*/ε)
¿Completa?	No	Sí (s finita)	Sí (b finita)	Sí (b finita y ε>0)
¿Óptima?	No	Sí (costes iguales)	Sí (costes iguales)	Sí

Base de conocimiento (KB): Sentencias que representan afirmaciones sobre el mundo en un lenguaje formal.

Heurística: Función que estima lo cerca que se está del estado objetivo siguiendo el camino de menor coste.

Búsqueda voraz (greedy): Expandir el nodo más cercano al objetivo (ni óptima ni completa).

Búsqueda A*: Ordena por la suma de los costes y las heurísticas (f(n) = g(n) + h(n), es óptimo para árboles y grafos con heurísticas admisibles y consistentes).

Heurística admisible: Se cumple 0 ≤ h(n) (coste estimado) ≤ h*(n) (coste verdadero).

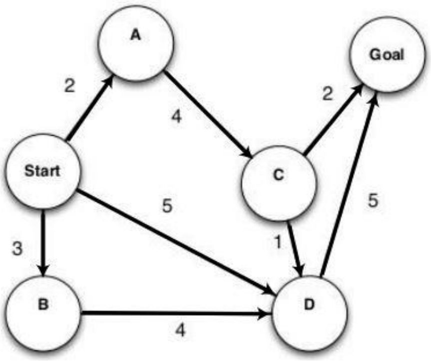
Admisibilidad: Coste ≤ Coste real al objetivo (h(A) ≤ h*(A)).

Consistencia: Coste heurístico del “arco” ≤ Coste real del arco (h(A) - h(C) ≤ c(A,C) o h(A) ≤ c(A,C) + h(C)).

Heurística inadmisible en la búsqueda del grafo A*: Si la función heurística está acotada, entonces la búsqueda en grafos A* visitaría todos los nodos eventualmente, y encontraría un camino hacia el estado objetivo si existe. Una heurística inadmisible no garantiza la optimalidad, ya que puede hacer que el buen objetivo óptimo parezca estar muy lejos, y llevarte a un objetivo subóptimo.

Heurística admisible en la búsqueda en el grafo A*: Aunque las heurísticas admisibles garantizan la optimalidad para la búsqueda en el árbol A*, no garantizan necesariamente la optimalidad para la búsqueda en el grafo A*; sólo se garantiza que devuelvan una solución óptima si también son consistentes. En la búsqueda en el árbol, si la heurística es admisible, la solución será óptima.

Heurística inadmisible vs admisible (ventaja): El tiempo para resolver un problema de búsqueda A* depende de dos factores (número de nodos expandidos y tiempo empleado por nodo). Una heurística inadmisible puede ser más rápida de calcular, lo que conduce a una solución que se obtiene más rápidamente debido al menor tiempo empleado por nodo. También puede ser una estimación más cercana a la función de coste real aunque a veces la sobreestime, con lo que se expanden menos nodos. Perdemos la garantía de optimalidad al utilizar una heurística inadmisible. Pero a veces podemos estar de acuerdo con encontrar una solución subóptima a un problema de búsqueda.



BLOQUE 3: LÓGICA

Conectores lógicos: Implicación ($\alpha \models \beta$, α implica β), negación ($\neg\alpha$, no), conjunción ($\alpha \wedge \beta$, y), disyunción ($\alpha \vee \beta$, o), condicional ($\alpha \Rightarrow \beta$, implica), bicondicional ($\alpha \Leftrightarrow \beta$, si y sólo si).

P	Q	¬P	P ∧ Q	P ∨ Q	P ⇒ Q	P ⇔ Q
Falso	Falso	Verdadero	Falso	Falso	Verdadero	Verdadero
Falso	Verdadero	Verdadero	Falso	Verdadero	Verdadero	Falso
Verdadero	Falso	Falso	Falso	Verdadero	Falso	Falso
Verdadero	Verdadero	Falso	Verdadero	Verdadero	Verdadero	Verdadero

Conversión a CNF: $P \Leftrightarrow Q$ se convierte en $((P \Rightarrow Q) \wedge (Q \Rightarrow P))$; y $P \Rightarrow Q$ se convierte en $(\neg P \vee Q)$, donde cada sentencia es una conjunción de cláusulas y cada cláusula es una disyunción de literales $((\vee) \wedge (\vee) \wedge \ldots)$.

Conmutatividad de ∧ Conmutatividad de ∨ Asociatividad de ∧ Asociatividad de ∨ Eliminar doble negación Contraposición	$(\alpha \wedge \beta) = (\beta \wedge \alpha)$ $(\alpha \vee \beta) = (\beta \vee \alpha)$ $((\alpha \wedge \beta) \wedge \gamma) = (\alpha \wedge (\beta \wedge \gamma))$ $((\alpha \vee \beta) \vee \gamma) = (\alpha \vee (\beta \vee \gamma))$ $\neg(\neg\alpha) = \alpha$ $(\alpha \Rightarrow \beta) = (\neg\beta \Rightarrow \neg\alpha)$	Eliminar implicación Eliminar bicondicional Ley de Morgan 1 Ley de Morgan 2 Distribuir ∧ respecto a ∨ Distribuir ∨ respecto a ∧	$(\alpha \Rightarrow \beta) = (\neg\alpha \vee \beta)$ $(\alpha \Leftrightarrow \beta) = ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$ $\neg(\alpha \wedge \beta) = (\neg\alpha \vee \neg\beta)$ $\neg(\alpha \vee \beta) = (\neg\alpha \wedge \neg\beta)$ $(\alpha \wedge (\beta \vee \gamma) = ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$ $(\alpha \vee (\beta \wedge \gamma) = ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$

Tipos de sentencias: Válida (V en todos los mundos), satisfacible (V en al menos un mundo), insatisfacible (F en todos los mundos, y establece que $\alpha \models \beta = \alpha \wedge \neg\beta$).

Sentencias complejas: $\forall x$ Knows(x, BFF(x)) (todos), $\exists x$ Knows(x, BFF(x)) (un, alguno).

Instanciación universal: $\forall x$ King(x) ∧ Greedy(x) se convierte a King(John) ∧ Greedy(John)).

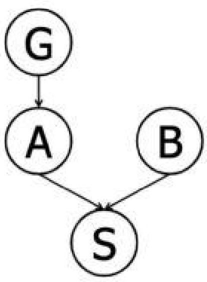
Instanciación existencial: $\exists x$ Crown(x) ∧ OnHead(x,John) se convierte a Crown(C) ∧ OnHead(C,John)).

Unificación: Encontrar sustituciones para que expresiones lógicas diferentes parezcan idénticas.

BLOQUE 4: INCERTIDUMBRE

$P(G)$	
$+g$	0.1
$-g$	0.9

$P(A\|G)$		
$+g$	$+a$	1.0
$+g$	$-a$	0.0
$-g$	$+a$	0.1
$-g$	$-a$	0.9



$P(B)$	
$+b$	0.4
$-b$	0.6

$P(S\|A,B)$			
$+a$	$+b$	$+s$	1.0
$+a$	$+b$	$-s$	0.0
$+a$	$-b$	$+s$	0.9
$+a$	$-b$	$-s$	0.1
$-a$	$+b$	$+s$	0.8
$-a$	$+b$	$-s$	0.2
$-a$	$-b$	$+s$	0.1
$-a$	$-b$	$-s$	0.9

Distribución conjunta: Producto de distribuciones conjuntas de cada variable.

Ejemplo: $P(X1,..,Xn) = \prod_i P(Xi \mid Padres(Xi)) \Rightarrow P(+g,+a,+b,+s) = P(+s|+a,+b)*P(+a|+g)*P(+b)*P(+g)$

Distribución marginal: Subtablas que eliminan variables.

Ejemplo: $P(X=x) = \sum_y P(X=x, Y=y) \Rightarrow P(+a) = P(+a|+g)*P(+g) + P(+a|-g)*P(-g)$

Variables independientes: La distribución conjunta se factoriza en un producto de dos distribuciones simples.

Ejemplo: $P(x|y) = P(x) \Rightarrow P(+a|+b) = P(+a)$

Probabilidad condicional: Probabilidad de a dado b.

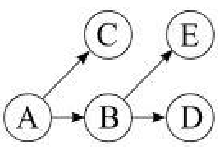
Ejemplo: $P(a|b) = P(a, b) / P(b) \Rightarrow P(+a|+b,+s) = P(+a,+b,+s)/P(+b,+s) = P(+a,+b,+s)/P(+a,+b,+s)+P(-a,+b,+s)$

Teorema de Bayes: Halla la probabilidad condicionada aplicando la regla del producto.

Ejemplo: $P(a|b) = P(b|a)*P(a) \Rightarrow P(+g|+a) = P(+a|+g)*P(+g)/P(+a)$

Entradas de un factor: Número de valores que puede tomar un nodo y sus padres.

Ejemplos: Nodo B (P(B), nº valores ^ nº nodos = 4^1 = 4), nodo A (P(A|G), nº valores ^ nº nodos = 4^2 = 16), nodo S (P(S|A,B), nº valores ^ nº nodos = 4^3 = 64).



	$P(A)$
$+a$	0.25
$-a$	0.75

$P(B\|A)$	$+b$	$-b$
$+a$	0.5	0.5
$-a$	0.25	0.75

$P(C\|A)$	$+c$	$-c$
$+a$	0.2	0.8
$-a$	0.6	0.4

$P(D\|B)$	$+d$	$-d$
$+b$	0.6	0.4
$-b$	0.8	0.2

$P(E\|B)$	$+e$	$-e$
$+b$	0.25	0.75
$-b$	0.1	0.9

Eliminación de variables (P(Q|E1=e1,.., Ek=ek)): Calcula la probabilidad posterior de un conjunto de variables de interés dado un conjunto de evidencias.

Factor (ejemplo): Para eliminar la variable B, $f(A,D,E) = \sum_B P(B|A)*P(D|B)*P(E|B)$

Condicionada de eliminación (ejemplo): Teniendo f(A,-c) y Z = f(+a,-c) + f(-a,-c) ⇒ P(+a, -c) = f(+a,-c) / Z