APELLIDOS:	PL:
NOMBRE:	DNI:

ESCUELA DE INGENIERÍA INFORMÁTICA

SISTEMAS INTELIGENTES

Examen Final de Teoría. Viernes 22 de enero de 2021.

I.- Búsqueda

1.- [0,5 puntos] Consideremos el siguiente heurístico en el problema de las N-reinas y el espacio de búsqueda completo, es decir el espacio en el que los estados tienen N reinas distribuidas una en cada columna del tablero.

h(n) = número de reinas atacadas en el estado n.

¿Es consistente este heurístico?

RESP: Si consideramos, por ejemplo, un estado en el que están todas las reinas en la primera fila, tenemos que h*(n) = N-1 y h(n) = N. En consecuencia, h no es admisible y por lo tanto tampoco es consistente.

2.- Se trata ahora de resolver el problema de secuenciación de trabajos en una máquina con ventanas de tiempo y minimización del tiempo total con el algoritmo A* y con un algoritmo genético.
El problema consiste en que tenemos un conjunto J de N trabajos que deben realizarse en una máquina, de forma que cada trabajo J_i, con i=1,..,N, está disponible a partir de un instante r_i, tiene una duración de p_i unidades de tiempo y debe finalizar como muy tarde en el instante d_i. La máquina solamente puede procesar un trabajo a la vez, y el objetivo es que el último de los trabajos en finalizar lo haga lo antes posible.

Formalmente, el problema se define así:

Datos

- Un conjunto de trabajos $J = \{ J_1,\ldots, J_N \}$, con $N>0$
- r_i, p_i, d_i , i=1,…,N, tiempo mínimo de inicio, duración y tiempo máximo de fin del trabajo J_i respectivamente

Variables de decisión (lo que tenemos que calcular)

- st_i, i=1,…,N, tiempo de inicio del trabajo i en la máquina

Restricciones

- R1: La máquina solo puede realizar un trabajo a la vez
- R2: La ejecución de un trabajo requiere el uso de la máquina durante todo el tiempo de procesamiento, y una vez iniciada no se puede interrumpir
- R3: El trabajo J_i no puede empezar antes de r_i, es decir $st_i \geq r_i$,
- R4: El trabajo J_i no puede terminar después de d_i, es decir $st_i + p_i, \leq d_i$,

Objetivo

- Minimizar el *makespan* definido como:

$$\max\{ st_i + p_i;\ i=1,\ldots,N \}$$

Ejemplo con tres trabajos

Trabajo	Tiempo mín. de inicio (r_i)	Duración (p_i)	Tiempo máx. de fin (d_i)
J_1	2	3	9
J_2	4	2	7

J_3	0	6	15

Una solución de coste 15 con st_1=6, st_2=4 y st_3=9, no óptima, es la siguiente (los intervalos sombreados representan períodos de inactividad de la máquina):

				J_2	J_1	J_3

Solución con A*

i.- [1 punto] Describir formalmente el espacio de búsqueda, y mostrar el espacio completo para la instancia anterior.

RESP: Una forma posible de resolver este problema (no la única) es ir planificando las tareas en un determinado orden de forma que a cada tarea se le asigna el menor tiempo de inicio posible, siempre después de la anterior tarea planificada y por supuesto teniendo en cuenta el valor de r_i de la tarea. Si se da una situación en la que al planificar la siguiente tarea no se puede cumplir la restricción R4, entonces esta solución parcial no es factible y habrá que considerar otro orden de las tareas.
Con esta idea, los estados se pueden representar mediante tuplas de la forma:

$$(J_u, st_u, J_R)$$

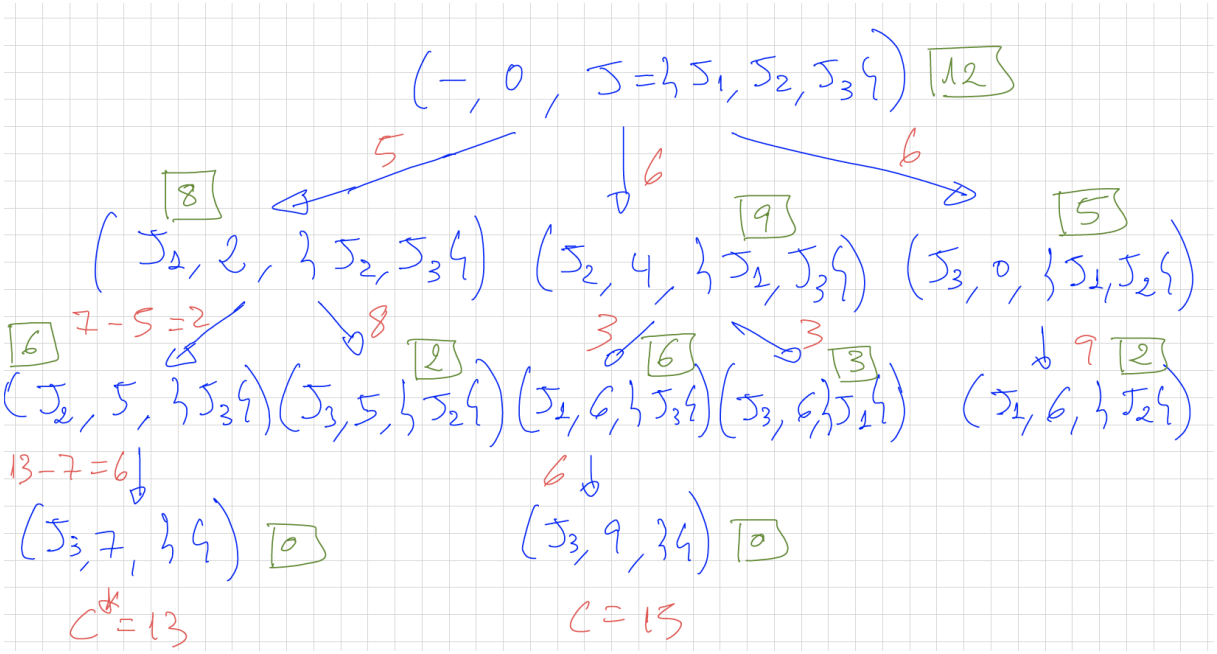
Donde:

- J_u es la última tarea planificada
- st_u es el tiempo de inicio asignado a la tarea J_u
- J_R es el conjunto de tareas que quedan sin planificar

Con esta información y los datos del problema, es posible identificar claramente cuál es el subproblema que nos queda por resolver en un estado: Planificar las tareas de J_R a continuación de las que ya están planificadas, teniendo en cuenta todas las restricciones del problema.

Los sucesores de un estado n, como el anterior, vienen dados por todas las opciones de planificar una tarea J_i de J_R, en el instante $st_i = \max(r_i, st_u+p_u)$ de forma que $st_i + p_i, \leq d_i$.
El coste de la regla que nos permite pasar de un estado n a otro n' es la diferencia entre el tiempo de finalización de las últimas tareas planificadas en los estados n' y n.

El espacio completo para la instancia anterior es el siguiente:



ii.- [1 punto] Diseñar un heurístico no trivial mediante el método de la relajación del problema y mostrar los valores de este heurístico para los nodos del espacio anterior.

RESP: En este caso, una relajación razonable es prescindir de la restricción R4, es decir, las tareas pueden terminar después de su tiempo máximo de fin d_i. El problema relajado que representa un estado se puede resolver el tiempo polinomial planificando las tareas de J_R a partir st_u+p_u de acuerdo con el siguiente algoritmo cuya idea es ir planificando las tareas restantes, eligiendo en cada paso la que antes puede empezar, en caso de empate, da igual:

JR = J_R; Ju = J_u; stu = st_u; du = st_u+p_u;
Mientras JR ≠ ∅ hacer
    Para cada tarea J_i ∈ JR calcular STi = max(r_i,stu+du);
    Calcular STj = min{STi; J_i ∈ JR}
    Planificar Jj tal que st_j = STj;
    stu = st_j; du = st_j+p_j;
    JR = JR \ {J_j};
finMientras;
Devuelve el tiempo de fin de la última tarea planificada.

Los valores de este heurístico para los nodos del espacio anterior se muestran en verde.

Solución con AG

iii.- [1 punto] Proponer un esquema de codificación y decodificación adecuados. Indicar varios cromosomas para el ejemplo anterior y las soluciones potenciales que representan y sus valores de fitness de acuerdo con el algoritmo de decodificación propuesto.

RESP: Siguiendo la misma idea de la pregunta anterior, la codificación natural es mediante permutaciones de las tareas, es decir permutaciones de 1,…,n. Y la decodificación se haría con un algoritmo similar al anterior, sin tener en cuenta los tiempos máximos de fin, concretamente:

Dado un cromosoma C = [1],…,[n]
Planificar la primera tarea de C, $J_{[1]}$, tal que: $st_{[1]} = r_{[1]}$;
Para i de 2 a n hacer

Planificar la tarea $J_{[i]}$ tal que $st_{[i]}=\max(r_{[i]}, st_{[i-1]} + p_{[i-1]})$;
finMientras;

Con este algoritmo, posiblemente algunas tareas no terminarán antes de su tiempo máximo de fin. En este caso, en el cálculo del fitness hay que penalizar al cromosoma, pero de forma proporcional a las restricciones que se violan. Una opción es la siguiente:

Fitness(C) = 1 /
(Tiempo de fin de la tarea J[n] $\times$ (1 + número de tareas que no terminan en su tiempo máximo de fin)).

De esta forma, un cromosoma que representa una solución no factible siempre es peor que otro que representa una solución factible. Además, cuantas más restricciones se violen, peor será el cromosoma. Se podría afinar un poco más teniendo en cuenta la cantidad de tiempo que se exceden las tareas su tiempo máximo de fin.

II.- Representación

3.- [1 punto] Dado el siguiente sistema conjunto de reglas

R1: H,F,E -> D R2: F,C -> I R3: D,G -> I R4: H -> A R5: F -> E R6: I,A ->B R7: G->F R8: A,G->I R9: A,H ->F	• **[0.5 puntos]** Aplicando encadenamiento hacia delante, describe el proceso de inferencia resultante suponiendo más prioritaria la regla con más condiciones en su antecedente (en caso de igualdad, se aplicará el principio de prioridad, aplicándose primero la regla con menor subíndice). La Base de Hechos inicial es BH: {G, H} y la meta es **I** • **[0.5 puntos]** Aplicando encadenamiento hacia detrás, describir el proceso de inferencia resultante suponiendo que la regla más prioritaria es la de subíndice mayor, que la meta a obtener es B y que la base de hechos inicial es BH: {C, H}

a)
BH_0: {G, H}, CC_0= {R4, R7} . Aplicamos R4
BH_1: {A, G, H}, CC_1= {R7, R8, R9}. Aplicamos R8
BH_2: {I, A, G, H}, CC_2= {R6, R7,R9}

b)
BH_0: {C, H}, Aplicando R6, B es cierto si I y A lo son.

Submeta I
CC={R2,R3,R8}. Aplicamos R8. I es cierto si A y G lo son

Submeta G: No se puede verificar

CC={R2,R3}. Aplicamos R3. I es cierto si D y G lo son

Submeta G: No se puede verificar

CC={R2}. Aplicamos R2. I es cierto si F y C lo son. C está en la base de hechos. Intentamos verificar F.

Submeta F: CC={R9}. F es cierto si A y H lo son. H está en la base de hechos.
Submeta A. CC={R4}

4.- Dada la red bayesiana de la figura, utilizando el criterio de D-separación o la condición de Markov

- **[0.5 puntos]** Estudia que variables son condicionalmente independientes dadas otras variables y que variables son independientes si ninguna otra variable está observada.

Cuando no hay variables observadas: Son independientes las que sean causa común de otra: A y B, C y F

B y E son c.i si C o D están observados. En general cualquier par de nodos que sean un subcamino de B->C->D->E son C.I si algún nodo intermedio está observado.

F y E son C.I si D está observado

A y E son C.I si D o C está observado

- **[0.5 puntos]** Calcula P(¬A,¬B,¬C,D,E,F)

P(¬A,¬B,¬C,D,E,F) = P(¬A) * P(¬B) * P(¬C|¬A,¬B) * P(D|¬A,¬C,F) * P(E|D) * P(F) = 0,8 * 0,7 * 0,1 * 0,6 * 0,2 * 0,4 = 0,002688

- **[0.5 puntos]** Calcula un ejemplo de muestra aplicando la ponderación de la verosimilitud para calcular P(B/A, ¬D).

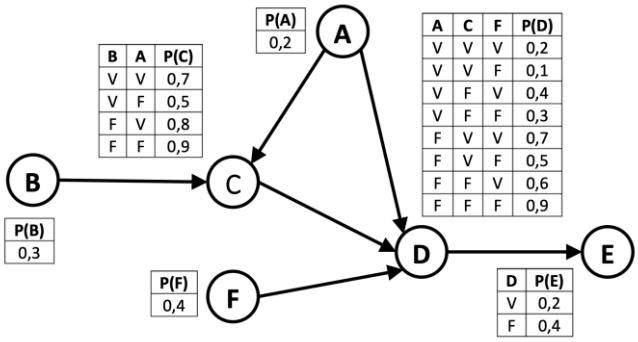
El primer paso es determinar un orden topológico de las variables, por ejemplo: B, A, F, C, D, E. La muestra empieza teniendo un peso w=1.

B: será verdadero aleatoriamente con prob. 0,3. Supongamos que sale B=V.

A: será verdadero obligatoriamente. El nuevo peso será w = 1*0,2 = 0,2.
F: será verdadero aleatoriamente con prob. 0,4. Supongamos que sale F=F. C: será verdadero aleatoriamente con prob. 0,7. Supongamos que sale C=V.

D: será falso obligatoriamente. El nuevo peso será w = 0,2 * 0,9 = 0,18.

E: será verdadero aleatoriamente con prob. 0,4. Supongamos que sale E=V.
Entonces, la posible muestra final que hemos generado es {B=V, A=V, F=F, C=V, D=F, E=V} y su peso es w = 0,18.



III.- Aprendizaje

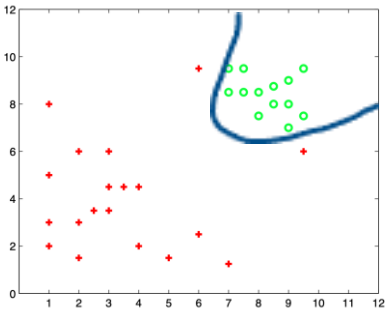
5.- [0.5 puntos] Tenemos un conjunto de datos con 100 instancias para resolver un problema de clasificación binario. Explica **razonadamente** que esquema de validación utilizarías.

6.- [1 punto] Considera una red neuronal con **función de activación lineal**. Tiene dos entradas (a, b), una neurona oculta (c) y una unidad de salida (d). Habrá un total de cinco pesos ($w_{ca}, w_{cb}, w_{c0}, w_{dc}, w_{d0}$) que vamos a inicializar todos a 0.1 y con una tasa de aprendizaje $\eta = 0.3$. Calcula el valor de los pesos utilizando el algoritmo de propagación hacía atrás para la siguiente instancia (a,b,d)=(1,1,0). En este caso: $\delta_k = (y_k - o_k)$ y $\delta_h = \sum_k \delta_k w_{kh}$

Solución: $o_c = 0.1 \cdot 1 + 0.1 \cdot 1 + 0.1 \cdot 1 = 0.3$; $o_d = 0.1 \cdot 1 + 0.1 \cdot 0.3 = 0.13$
$\delta_d = 0 - 0.13 = -0.13$; $\delta_c = -0.13 \cdot 0.1 = -0.013$

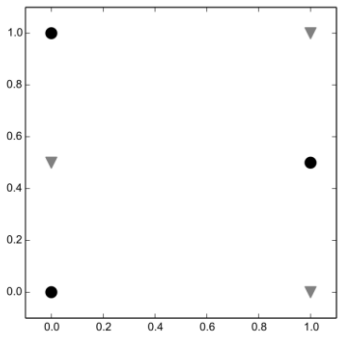
$$w_{d0} = 0.1 + 0.3 \cdot (-0.13) \cdot 1 = 0.061$$
$$w_{dc} = 0.1 + 0.3 \cdot (-0.13) \cdot 0.3 = 0.0883$$
$$w_{c0} = 0.1 + 0.3 \cdot (-0.013) \cdot 1 = 0.0961$$
$$w_{ca} = 0.1 + 0.3 \cdot (-0.013) \cdot 1 = 0.0961$$
$$w_{cb} = 0.1 + 0.3 \cdot (-0.013) \cdot 1 = 0.0961$$

7.- [0.5 puntos] Dado el siguiente conjunto de datos que representa un problema de clasificación binario, pretendemos resolverlo utilizando SVM con kernel polinómico de grado 2. Dibuja una posible frontera de separación cuando el valor de C es tan grande como sea posible y justifica por qué has dibujado esa frontera de decisión.



8.- [0.5 puntos] Dado el siguiente conjunto de datos, indica **razonadamente** cual o cuales de los siguientes métodos podrían resolverlo:

- SVM con Kernel lineal NO
- SVM con kernel radial SI
- Árboles de Decisión SI
- 3-NN SI



9.- [0.5 puntos] Explica las similitudes y diferencias entre la InfoGain (IG) y GainRatio (GR)

Solución: La métrica GrainRatio se define como la relación entre InfoGain de la clase y una variable predictora dividido por el SplitInfo de esta variable. Con esto, la métrica GR consigue paliar el sesgo de IG hacia variables con muchos valores y que conduciría a modelos con sobreajuste.

10.- La siguiente figura muestra un árbol de decisión. Los números de cada nodo representan el número de ejemplos en cada clase de ese nodo. Por ejemplo, en el nodo raíz, los números

16; 16 representan 16 ejemplos de la clase 0 y 16 ejemplos de la clase 1. Para ahorrar espacio,

- **[0.5 puntos]**¿Cuál es la ganancia de información en el nodo raíz?

Solución: H(X)=-2*0.5 $\log_2(0.5)$=1

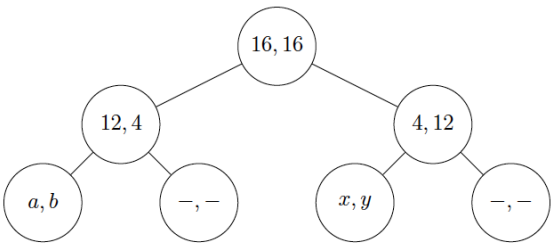
Llamemos a la variable que está en el nodo raíz Y

H(X|Y)=$\sum_{y\in Y} p(y)H(X|Y=y)=\frac{1}{2}\left(\frac{-4}{16}log\frac{4}{16}-\frac{12}{16}log\frac{12}{16}\right)+\frac{1}{2}\left(\frac{-12}{16}log\frac{12}{16}-\frac{4}{16}log\frac{4}{16}\right)=$
$\left(\frac{-1}{4}log\frac{1}{4}-\frac{3}{4}log\frac{3}{4}\right)=\left(\frac{1}{2}-\frac{3}{4}*-0.4\right)=0.8$

Así que IG(X,Y)=1-0.8=0.2

- **[0.5 puntos]**¿Cómo tienen que ser los valores de a y b para que la ganancia de información sea la más pequeña posible?

Solución: Lo valores de a y b deben mantener una relación proporcional con 12 y 4, por ejemplo a=9 y b =3. Esto significa que la variable de este nodo no aporta ninguna información para separar la variable clase, al menos en esta rama.



$P(x_1,\dots,x_i)=\sum_{x_{i+1},\dots,x_n} P(x_1,\dots,x_i,x_{i+1},\dots,x_n)=\sum_{x_{i+1}} P(x_1,\dots,x_i,x_{i+1})$		
$P(x\vee y)\equiv P(X=x\vee Y=y)=\frac{P(x,y)}{P(y)}$	$P(X,Y)=P(X\vee Y)P(Y)=P(Y\vee X)P(X)$	
$P(Y)=\sum_{i=1}^{m} P(Y\|x_i)P(x_i))$	$P(X_1=x_1,\dots,X_n=x_n)=\prod_{i=1}^{n} P(X_i=x_i\vee X_{i+1}=x_{i+1},\dots,X_n=x_n)$	
$P(X\vee Y)=P(X)$	$P(X\vee Y,K)=P(X\vee K)$	$P(X\vee Y,K)\neq P(X\vee K)$
$P(A\vee MB(A),B)=P(A\vee MB(A))$	$P(x_1,\dots,x_n)=\prod_{i=1}^{n} P(X_i=x_i\vee Padres(X_i))$	
	$P(X\vee Padres(X))=P(X\vee Padres(X),NoDescendientes(X))$	
$E(w)=\frac{1}{2}\sum_d\sum_k (y_{kd}-o_{kd})^2$	$\delta_k=o_k(1-o_k)(y_k-o_k)$ $\delta_h=o_h(1-o_h)\sum_k w_{kh}\delta_k$	$\Delta w_i=\eta(y_d-wx_d)x_{di}$ $g(x)=\frac{1}{1-e^{-x}}$
$dist(\vec{x},\vec{y})=\left(\sum_i^k \|x_i-y_i\|^p\right)^{1/p}$	$E(w)=\frac{1}{2}(y_d-wx_d)^2$	$g(\cdot)=\begin{cases}1 si wx>0\\-1 \text{en otro caso}\end{cases}$
$H(X)=-\sum_{i=1}^{k} p_i log_2 p_i$	$H(X\|Y)=\sum_{y\in Y} p(y)H(X\vee Y=y)$	$K(x_i,x_j)=\phi(x_i)\phi(x_j)$ $K(x,y)=(xy+1)^d$
$b=\frac{1}{\|VS\|}\sum_{i\in VS} y_i-wx_i$	$w=\sum_{i\in VS}\alpha_i y_i x_i$	$d=\frac{2}{\|\|w\|\|}$