UNIVERSIDAD SIMÓN BOLÍVAR
DPTO. COMPUTACIÓN Y TEC. DE INF.
CI-5437 INTELIGENCIA ARTIFICIAL I

Examen Parcial I

Estudiante
EROS CEDEÑO

Profesor
CARLOS INFANTE

18 de julio de 2023

Contenido General

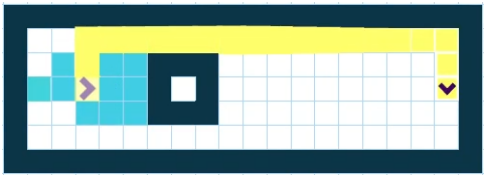


Figura 1: Algoritmo A^*.

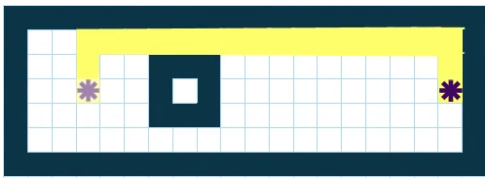


Figura 2: Algoritmo DFS

0. Verdadero o Falso

0.1. Enunciado

(2 pts) – Para cada una de las siguientes oraciones, diga si son Verdaderas o Falsas, y justifique apropiadamente su respuesta (0.5 pts c/u).

A. La cantidad de nodos expandida por el algoritmo de búsqueda en profundidad (DFS) siempre será mayor o igual a la cantidad de nodos expandida por A^*

B. Al ejecutar el algoritmo de consistencia de arco AC3 sobre un CSP con un grafo de restricciones con estructura de árbol, la complejidad en tiempo en el peor caso no será distinta que en un CSP que no presente esta estructura de árbol

C. SAT en lógica proposicional es parcialmente decidible

D. El algoritmo minimax siempre escogerá los mismos movimientos, independientemente de si usamos poda $\alpha - \beta$ o no

0.2. Respuesta

A. En general A^* expande menos nodos que DFS cuando la heurística de A^* es admisible y buena. Sin embargo pueden darse casos en el que la heurística de A^* sea pobre al dirigir la búsqueda, si además los nodos solución son frecuentes y se encuentran a gran profundidad dentro del grafo y dicho grafo es poco denso puede ocurrir que el algoritmo DFS alcance primero un nodo solución que el algoritmo A^*. Es importante señalar que DFS aunque no garantiza que el camino obtenido hasta la solución sea mínimo, si el problema es como se describió anteriormente puede darse el caso en el que DFS expanda menos nodos que A^* en su recorrido. Esto no es común dado que depende mucho del problema a resolver. Sin embargo, la premisa de que 'Siempre' DFS expandirá mas nodos que A^* sin importar el problema, es falsa. Se anexan figuras ilustrativas.

B. Falso. Si se analiza el desempeño del algoritmo AC3 en general se tiene que la complejidad en el peor caso es $O(n^2d^3)$. Sin embargo si se considera que el grafo de restricciones tiene estructura de arbol la complejidad disminuye. El método reduce_arc(X, Y) no cambia su desempeño y se mantiene en $O(d^2)$ con $d = max_i|D_i|$. Como el grafo es un arbol entonces hay a lo sumo n-1 aristas y por lo tanto n-1 inserciones iniciales en la cola (en contraste con los n^2 del caso general). Como hay $O(d)$ valores para X, el arco (Z,X) puede ser reinsertado $O(d)$ veces por lo tanto la cantidad de iteraciones del algoritmo estara acotado por $O(n + nd) = O(nd)$. Y finalmente el tiempo en el peor caso para AC3 sobre arboles es $O(nd^3)$.

C. Falso. SAT en lógica proposicional es decidible. Debido a que existen algoritmos correctos y completos que resuelven el problema en este tipo de lógica. Mediante tablas de la verdad siempre es posible determinar la satisfacibilidad de cualquier formula en lógica proposicional. Sin embargo el problema tiene una complejidad no-polinomial (NP).

D. Verdadero. La poda alfa-beta es una optimización que se realiza sobre el algoritmo minimax para no tener que recorrer ramas de estados donde se sabe de antemano que el resultado no mejora. De aqui que los movimientos que se eligen no se ven afectados porque la elección de los nodos con el valor máximo o mínimo se mantiene.

1. Perros y Hienas

1.1. Enunciado

En alguna ubicación de algún universo, han emprendido un viaje juntos un peculiar grupo: tres perros y tres hienas. Durante su travesía, llegan a un río de aguas turbulentas que se extiende hasta el horizonte, por lo que no queda más remedio que cruzarlo. Por fortuna, a la orilla del río hay una balsa improvisada. Sin embargo, tras un par de pruebas, los cuadrúpedos rápidamente determinan que solamente dos de ellos pueden subir a la balsa a la vez. Además, es necesario que al menos uno de ellos permanezca en la balsa para el viaje de regreso.

Más aún, habiendo empezado esta odisea hace poco tiempo, aún no se ha establecido suficiente confianza entre el grupo. Por lo tanto, como las hienas tienen fama de traicioneras y este grupo de perros es bastante prejuicioso, estos quieren determinar una manera en que los 6 mamíferos puedan cruzar el río (a pesar de los prejuicios, una amistad ha empezado a formarse y no quieren dejar a ninguna hiena atrás) de manera que en ningún momento haya más hienas que perros en algún lado del río.

Para añadir complicación extra, una de las hienas tuvo un desagradable encuentro con un león, debido a lo cuál tiene dificultades de movimiento en sus extremidades y no es capaz de llevarla de un lado a otro si está sola en la balsa.

Dado este planteamiento de problema:

A. (1 pts) – Formule un espacio de estados, según la tupla vista en clases, para representar la situación. Indique el tamaño del espacio de estados según su formulación.

B. (1 pts) – ¿Qué algoritmo de búsqueda usaría para encontrar la solución en el espacio de estados formulado? Justifique su elección, y luego úselo para proponer dicha solución.

C. (1 pts) – Reescriba el problema como un problema de STRIPS (no necesariamente en lenguaje PDDL). Es decir, plantee los estados como colecciones de átomos booleanos y las acciones usando las listas Add, Del y Pre.

1.2. Respuesta

A. Dada la naturaleza del problema se tienen las siguientes observaciones:

- Como no pueden haber mas perros que hienas en ninguna de las dos orillas, entonces en todo momento alguna hiena es la que debe transportar la balsa de regreso.
- Existen 2 hienas transportadoras porque una de ellas esta herida
- Los perros son indistinguibles entre si
- No hay restricciones en cual perros llega primero a la otra orilla ni en cual hiena llegue primero al otro lado siempre y cuando se cumplan las restricciones anteriores.

Sin perdida de generalidad, sean los perros P_1, P_2, P_3 y las hienas H_1, H_2, H_3 y la hiena herida H_3. Se nombran las siguientes convenciones:

- NrPL1: Numero de perros en el lado inicial.
- NrHL1: Numero de hienas en el lado inicial.
- NrPL2: Numero de perros en el otro lado del río.
- NrHL2: Numero de hienas en el otro lado del río.
- IdH: indice de la hiena transportadora (0 si no se ha asignado la hiena transportadora).

Se plantea el siguiente espacio de estados posible.

$$P =< S, A, S_0, S_G, F, c >$$

donde

$$\begin{aligned} S = \{ < NrPL1, NrHL1, NrPL2, NrHL2, IdH > \in (\{0,1,2,3\}^4 \times \{0,1,2\}) : \\ NrPL1 \geq NrHL1 \ \wedge \ NrPL2 \geq NrHL2 \ \wedge \ NrPL1 + NrPL2 = 3 \ \wedge \\ IdH = 0 \Rightarrow NrHL1 + NrHL2 = 3 \ \wedge \\ IdH \neq 0 \Rightarrow NrHL1 + NrHL2 = 2 \\ \} \end{aligned}$$

es el conjunto de estados posibles, por otro lado

$$A = \{EHT1, EHT2, CHT, TP, TH, BT\}$$

sera el conjunto de acciones:

- EHT1: Elegir Hiena 1 como transportadora
- EHT2: Elegir Hiena 2 como transportadora
- CHT: Cambiar Hiena Transportadora
- TP: Transportar Perro
- TH: Transportar Hiena
- BT: Bajar Hiena transportadora al otro lado

El estado inicial S_0 sera:

$$S_0 =< 3, 3, 0, 0, 0 >$$

El estado objetivo S_G es:

$$S_G =< 0, 0, 3, 3, 0 >$$

La funcion de costo sera unitario y la funcion de transicion sera:

$$f(< a, b, c, d, e >, ac) = \begin{cases} < a, b-1, c, d, 1 > & si \quad ac = EHT1 \ \wedge \ e = 0 \\ < a, b-1, c, d, 2 > & si \quad ac = EHT2 \ \wedge \ e = 0 \\ < a, b, c, d, \ 2^e \ mod \ 3 > & si \quad ac = CHT \ \wedge \ b > 0 \\ < a-1, b, c+1, d, e > & si \quad ac = TP \ \wedge \ a > 0 \ \wedge \ e \neq 0 \\ < a, b-1, c, d+1, e > & si \quad ac = TH \ \wedge \ b > 0 \ \wedge \ e \neq 0 \\ < a, b, c, d+1, 0 > & si \quad ac = BT \ \wedge \ e \neq 0 \end{cases}$$

El espacio de estados contiene 14 elementos que se muestran en la figura de los cuales solo uno de ellos es estado objetivo y se encuentra a profundidad de 7.

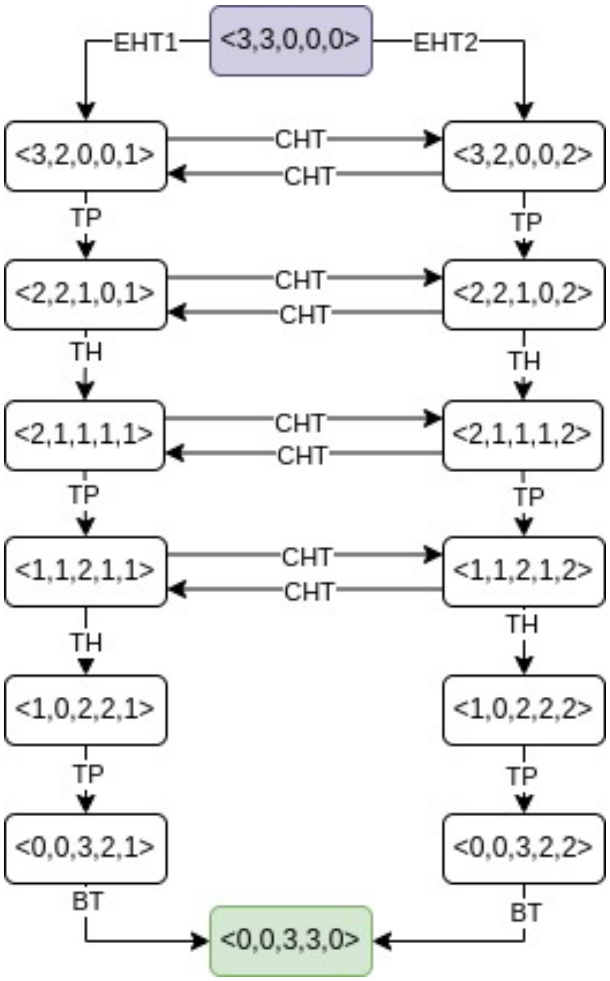


Figura 3: Grafo asociado al modelo de estados P

B. Debido a que las soluciones se encuentran a gran profundidad, aunado a que las soluciones se alcanzan siempre al mismo nivel del grafo entonces considero que un algoritmo con búsqueda en profundidad seria una buena elección. Preferiría DFS con limite en la profundidad maxima. DFS con profundidad de 7 dado que evita el peor caso del DFS que es cuando se recorra el camino mas largo para alcanzar el objetivo (en este caso el camino mas largo seria de tamaño 11). Si este problema se extiende a N perros y N hienas se sabe que el estado objetivo estará a profundidad $2N + 1$ y basta con iterar con DFS con la profundidad máxima de $2N + 1$ para obtener diferentes caminos hasta encontrar el camino óptimo.

C. Determinaremos $P =< F, O, I, G >$. Sea $Animales = \{P_1, P_2, P_3, H_1, H_2, H_3\}$ Y $Lados = \{L_1, L_2\}$. Entonces el conjunto de estados booleanos atómicos esta determinado por el siguiente esquema

$$F = \{AnimalEnLado(?a, ?l), BalsaEnLado(?l), AnimalEnBalsa(?a) | \\ ?a \in Animales, ?l \in Lados\}$$

Por otro lado el estado inicial I:

$$I = \{AnimalesEnLado(?a, L_1) \mid ?a \in Animales\}$$

El estado objetivo:

$$G = \{AnimalesEnLado(?a, L_2) \mid ?a \in Animales\}$$

Luego, los operadores O tienen los siguientes esquemas de acción:

$$Subir(?a \in Animales):$$
$$P: \{AnimalEnLado(?a, L_1), BalsaEnLado(L_1)\},$$
$$A: \{AnimalEnBalsa(?a)\},$$
$$D: \{AnimalEnLado(?a, L_1)\}$$

$$Llevar(?a_1 \in Hienas - \{H_3\}, ?a_2 \in Animales):$$
$$P: \{BalsaEnLado(L_1), AnimalEnBalsa(?a_1), AnimalEnBalsa(?a_2)\}$$
$$A: \{BalsaEnLado(L_2)\},$$
$$D: \{BalsaEnLado(L_1)\}$$

$$Traer(?a_1 \in Hienas - \{H_3\}):$$
$$P: \{BalsaEnLado(L_2), AnimalEnBalsa(?a_1)\}$$
$$A: \{BalsaEnLado(L_1)\},$$
$$D: \{BalsaEnLado(L_2)\}$$

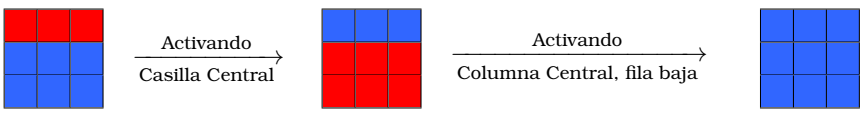
$$Bajar(?a \in Animales):$$
$$P: \{BalsaEnLado(L_2), AnimalEnBalsa(?a)\}$$
$$A: \{AnimalEnLado(?a, L_2)\},$$
$$D: \{AnimalEnBalsa(?a)\}$$

2. Rompecabezas de Colores invertibles

2.1. Enunciado

(3 pts) – Considere que se le presenta una cuadrilla rectangular de tamaño $n \times m$, donde cada uno de las casillas puede tener color rojo o azul. Al seleccionar una casilla cualquiera, esta puede ser activada para cambiar su color y el de todas las casillas adyacentes. El objetivo es lograr que todas las casillas sean de color azul.

Por ejemplo, en el caso particular 3×3:



Para el caso general $n \times m$:

A. (1 pts) – Formule un espacio de estados, según la tupla vista en clases, para representar la situación. Indique el tamaño del espacio de estados según su formulación.

B. (1 pts) – ¿Cuál es el factor de ramificación del problema?

C. (1 pts) – ¿Qué algoritmo de búsqueda usaría para encontrar la solución en el espacio de estados formulado? Justifique su respuesta, incluyendo en su análisis consideraciones respecto a la poda de duplicados y posible uso de heurísticas.

2.2. Respuesta

A. Sea P el espacio de estado definido por la tupla $P =< S,\ A,\ S_0,\ S_G,\ f, c >$ donde un estado S_i es una tupla de 0s y 1s de tamaño $n \times m$ con 1 en las posiciones correspondientes a las casillas de color rojo y 0 en las de color azul. El espacio de estados sera luego $S = \{0,\ 1\}^{n \times m}$, por lo tanto $|S| = 2^{n \times m}$. El conjunto de acciones se define por comprensión como $A = \{SeleccionarCasilla_i : 0 \leq i < nm\}$ donde $SeleccionarCasilla_i$ es la acción de cambiar de color la casilla $i - esima$ y las casillas adyacentes a esta. Un estado inicial S_0 puede ser cualquier elemento en S. El estado objetivo S_G es la tupla de n*m ceros. La funcion de costo sera siempre la funcion unitaria. Y la funcion de transicion dado un estado (tupla) y la accion ($CambiarCasilla_i$) retorna un estado del tablero (tupla) donde los colores de la posicion i-esima asociada a la accion es cambiada de color (de 0 a 1 o viceversa) asi como sus celdas adyacentes.

B. El factor de ramificacion sera $b = nm$ esto debido a que en cada estado es posible ejecutar cualquiera de las $n \times m$ acciones de seleccion de casilla. Y esto para cada nodo a cada nivel.

C. Si se analiza el grafo asociado al modelo de estados se observan varias características notables.

- El grafo asociado presenta estados y ramas repetidas.
- Las acciones son invertibles (seleccionar dos veces la misma casilla retorna al estado anterior a presionarla). Por lo que se puede aplicar poda de padres.
- Los estados objetivos no están a una profundidad fija, una solución se puede encontrarse ya sea en un movimiento o en múltiples movimientos.
- Como la cota de memoria al almacenar los estados posibles es $O(2^{nm})$ y el factor de ramificación es $b = 9$ hay que considerar optimizaciones en memoria.

Dadas las consideraciones anteriores, si no se cuenta con heurísticas adecuadas se puede usar el algoritmo IDDFS con poda parcial de duplicados porque explora nivel por nivel iterando sobre la profundidad del DFS pero sin almacenar en memoria todos los nodos desde la raíz hasta dicho nivel como si haría BFS por ejemplo.

De encontrarse una heurística admisible para el problema se recomienda usar IDA^* dado que la memoria de este algoritmo es lineal y en los casos donde la heurística es consistente es un algoritmo óptimo.

Una posible heurística seria particionar el tablero en rectángulos de menor tamaño, similar a como se hace con el 15-puzzle y tener precalculadas varias PDBs sobre estos subproblemas. La heuristica seria el max del costo estimado de cada PDB sobre el estado en el que se realiza la consulta.

Si no se desea usar PDBs se podria tomar como una heurística la cantidad de rectángulos con área máxima posible en dicho estado, conformados únicamente por celdas rojas. Aunque aun no he logrado demostrar que esta heuristica sea admisible, tengo la hipótesis que así es.

3. Caballos en el Tablero

3.1. Enunciado

(3 pts) – Considere el problema de colocar k caballos en un tablero de ajedrez $n \times n$ sin que dos caballos estén en posición de atacarse (según las reglas convencionales en las que los caballos atacan en ajedrez), dado un número constante $k \leq n^2$

Plantee una formulación como CSP al problema, describiendo variables, dominios y restricciones.

3.2. Respuesta

Se desea colocar k caballos en un tablero $n \times n$ tal que ningún caballo se ataque entre si. En este caso se tendrá una variable por casilla del tablero que sera cierta cuando un caballo ocupe dicha posición y falso caso contrario. Esto es $X = \{Casilla_{i,j} : i, j \in [0, n)\}$ y es facil ver que $|X| = n^2$. Luego, el dominio seria el mismo para cada variable. Dado que son variables booleanas se les asignara a cada una dos valores posibles 1 o 0 si es true o false respectivamente. Por lo tanto $D = \{D_{i,j} = \{0, 1\} : i, j \in [0, n)\}$. Las restricciones indican que para cada dos casillas que compartan rango de ataque no se satisfagan ambas al mismo tiempo (no evalúen ambas a 1). Por utilidad se define el conjunto $Ataque$ como el conjunto de pares de estados en el que dos celdas con caballos se amenazan entre si.

$$
\begin{aligned}
Ataque = \{(Casilla_{i,j}, Casilla_{u,v}) : Casilla_{i,j}, Casilla_{u,v} \in X \wedge \\
(\ (\ (u = i+1 \ \vee \ u = i-1) \ \wedge \ (v = j+2 \ \vee \ v = j-2) \ ) \\
\vee \\
(\ (u = i+2 \ \vee u = i-2) \ \wedge \ (v = j+1 \ \vee \ v = j-1) \ ) \ ) \\
\}
\end{aligned}
$$

Una vez definido el conjunto de pares de celdas amenazadas se puede definir al conjunto de restricciones C de la siguiente manera:

$$C = Ataque \times \{\{(1, 0), (0, 1), (0, 0)\}\}$$

4. Algoritmo en CSP

4.1. Enunciado

(3 pts) – Sea P un problema CSP cualquiera, con su conjunto de variables, dominios y restricciones. Considere que quiere hallar todas las asignaciones posibles que satisfacen el problema.

Describa qué tipo de algoritmo usaría para hallar este conjunto de soluciones. ¿Podemos beneficiarnos del uso de alguna heurística para esto? ¿Por qué?

4.2. Respuesta

Las posibilidades son muy amplias y en general va a depender mucho de la naturaleza del CSP. Pero si se quiere un algoritmo general se puede proponer una combinación entre búsqueda e inferencia para encontrar los estados completos que satisfacen el problema. Inicialmente se puede transformar el CSP en un CSP binario para reducir la aridad de las restricciones.

Luego se realiza la búsqueda sobre el grafo de asignaciones parciales ejecutando una variación del algoritmo de backtracking (que se explicara luego) pero aplicando en cada nodo de la búsqueda el algoritmo AC3 y así mantener la consistencia de arco en el subgrafo a buscar (enfoque MAC). Se prefiere utilizar el enfoque de MAC dado que para problemas difíciles nos asegurara la consistencia de arco en la parte del grafo en que se ejecutara la búsqueda, y si el grafo es muy grande optimiza el uso de memoria y tiempo de ejecución dado que no es necesario hacer el CSP arco consistente antes de empezar la búsqueda.

La variación del algoritmo de backtracking que se plantea seria que la función sea consciente de la profundidad en la que es llamada (agregando otro parámetro a la función) y en el caso base de la función si se encuentra en el nivel mas alto y encontró un estado de satisfacibilidad lo almacena en una lista de estados satisfacibles y continúa en vez de retornar. Al continuar la función se llamara recursivamente con normalidad sin embargo al encontrar un estado satisfacible guardado lo ignora (poda el estado) y continua la búsqueda. De este

modo la búsqueda explorara el grafo e ira encontrando un estado objetivo a la vez y retornara finalmente este conjunto de estados objetivos encontrados cuando la búsqueda no encuentre ningún estado nuevo a agregar.

En cuanto a heurísticas nos podemos beneficiar de heurísticas para el ordenamiento de las variables a seleccionar. La Heuristica MRV (Minimum Remaining Values) junto a la idea de escoger la variable que aparezca en mas restricciones cuando los valores MRV de dos nodos sean iguales es una excelente opción. Con ayuda de estas heurísticas podemos detectar un fallo en la rama lo mas alto posible y así podamos con anticipación las ramas inútiles.

5. Tic Tac Toe

5.1. Enunciado

(4pts) – Considere el juego de la vieja (o tic-tac-toe), con tableros 3×3 donde dos jugadores se turnan para jugar con X_s y O_s. Definimos X_n como la cantidad de columnas, filas o diagonales con n cantidad de X_s. Así mismo, definimos O_n como la cantidad de columnas, filas o diagonales con n cantidad de O_s.

El valor del juego es +1 para cualquier posición con $X_3 = 1$ y -1 para cualquier posición con $O_3 = 1$. Cualquier otra posición terminal tiene un valor de 0. Para evaluar posiciones no-terminales, usaremos la función de evaluación:

$$Eval(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$$

Dado este planteamiento:

A. Dibuje el árbol de juego hasta profundidad 3 (la raíz es la profundidad 1). Es válido aprovechar la simetría de las posiciones para evitar perder tiempo dibujando posiciones equivalentes, pero indíquelo apropiadamente.

B. Indique el valor del juego para todos los nodos generados según la función de evaluación.

C. Use el algoritmo minimax a profundidad 3 para seleccionar el mejor movimiento inicial.

D. Indique los nodos a profundidad 2 y 3 que no serían evaluados si la poda alfa-beta fuera aplicada.

5.2. Respuesta

A. A continuacion el arbol de juego generado hasta profundidad 3. Las Xs se marcan en rojo y las Os en verde.

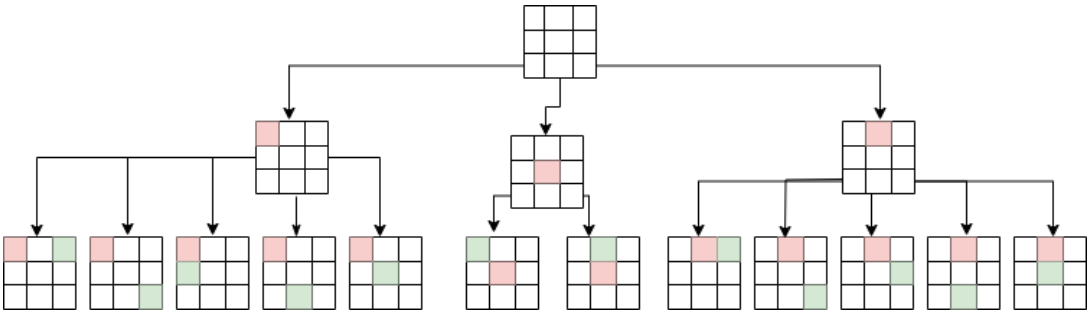
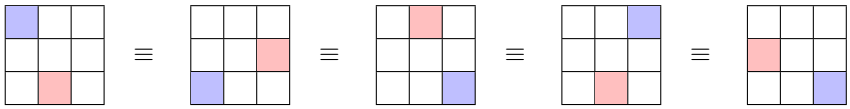


Figura 4: Tic Tac Toe 3x3 hasta profundida 3

Se están considerando equivalentes los estados que mediante rotaciones y reflexividades comparten estados, y se esta tomando uno de los elementos de la clase de equivalencia como representante. Esto se puede hacer porque no afecta el valor del juego Por ejemplo todos los siguientes estados son equivalentes:



B. Si se le aplica la funcion de evaluacion $Eval(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$ sobre cada uno de los estados generados en el grafo anterior se tiene la siguiente figura. Notese que no se esta aplicando minimax solo se esta evaluando estado por estado con la funcion de evaluacion como se pide en el ejercicio.

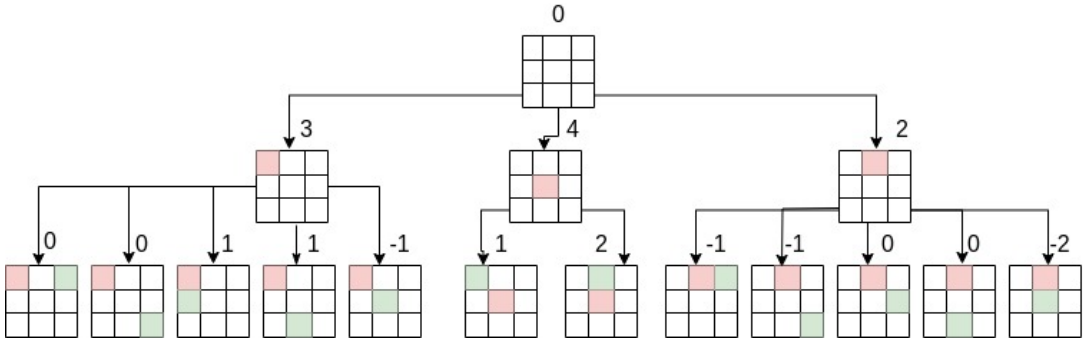


Figura 5: Evaluación de todos los estados generados, según función $Eval(s)$

C. Se utilizo el algoritmo minimax con poda como se observa en la figura 6 para evaluar cual seria el mejor movimiento de inicio para las Xs cuando el tablero esta en blanco. Como se observa en el árbol resultante, el mejor movimiento inicial seria ocupar la casilla central dado que se toma el hijo con mayor valor (porque el jugador X intenta maximizar el valor del tablero de juego).

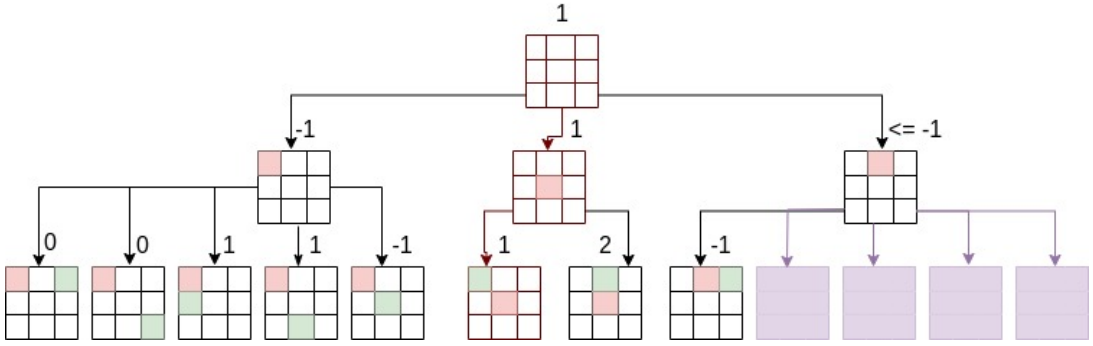


Figura 6: Árbol resultante de ejecutar Minimax con poda alfa beta hasta profundidad 3 con la función de evaluación $Eval(s)$

D. En esta ejecución de minimax con poda alfa beta se podaron 4 nodos en el nivel tres. Estos nodos se colorean de color morado en la figura 6. Se podan porque que el valor obtenido con el primer hijo de este padre ya retorno un costo menor a el máximo costo obtenido en el nivel 2, por lo que no es necesario expandir el resto de hijos y se podan. Hay que resaltar el hecho que el orden en que se expanden los hijos es de izquierda a derecha. Si por ejemplo se hubieran ordenado los nodos en el segundo nivel de mejor manera se podaría aun mas ramas.

6. Forma Clausal y UP

6.1. Enunciado

(2 pts) – Dada la siguiente fórmula lógica:

$$(\neg Z \vee \neg C \vee \neg A \vee D \vee B) \wedge A \wedge (\neg Y \vee B \vee \neg Z) \wedge (\neg Z \vee Y) \wedge (\neg A \vee X \vee \neg Y) \wedge (A \vee \neg X \vee Y \vee \neg D)$$
$$\wedge(\neg B \vee C \vee \neg X) \wedge (\neg B \vee C \vee D) \wedge (\neg A \vee Z)$$

A. (0.5 pts) – Tradúzcala a forma clausal.

B. (1.5 pts) – Aplique propagación unitaria (UP) sobre ella, mostrando los resultados a cada paso. De ser posible, use el resultado para dar un modelo o evaluación de la fórmula.

6.2. Respuesta

A. La representacion en Forma clausal es la siguiente:

$$\Delta = \{\{\neg Z,\ \neg C,\ \neg A,\ D,\ B\},\ \{A\},\ \{\neg Y,\ B,\ \neg Z\},\ \{\neg Z,\ Y\},\ \{\neg A,\ X,\ \neg Y\},\ \{A,\ \neg X,\ Y,\ \neg D\},$$
$$\{\neg B,\ C,\ \neg X\},\ \{\neg B,\ C,\ D\},\ \{\neg A,\ Z\}\}$$

Se aplicara el algoritmo de propagación unitaria sobre Δ inicialmente la única clausula unitaria es $\{A\}$ por lo que se inicia la evaluacion parcial con A.

$$\begin{aligned}
\Delta|A =& \{\{\neg Z,\ \neg C,\ D,\ B\},\ \{\neg Y,\ B,\ \neg Z\},\ \{\neg Z,\ Y\},\ \{\ X,\ \neg Y\},\\
& \{\neg B,\ C,\ \neg X\},\ \{\neg B,\ C,\ D\},\ \{Z\}\}\\
& < \text{Clausula a evaluar: } \{\ \mathsf{Z}\ \} >\\
\Delta|AZ =& \{\{\neg C,\ D,\ B\},\ \{\neg Y,\ B\},\ \{Y\},\ \{\ X,\ \neg Y\},\ \{\neg B,\ C,\ \neg X\},\ \{\neg B,\ C,\ D\}\}\\
& < \text{Clausula a evaluar: } \{\ \mathsf{Y}\ \} >\\
\Delta|AZY =& \{\{\neg C,\ D,\ B\},\ \{B\},\ \{\ X\},\ \{\neg B,\ C,\ \neg X\},\ \{\neg B,\ C,\ D\}\}\\
& < \text{Clausula a evaluar: } \{\ \mathsf{B}\ \} >\\
\Delta|AZYB =& \{\{\ X\},\ \{C,\ \neg X\},\ \{C,\ D\}\}\\
& < \text{Clausula a evaluar: } \{\ \mathsf{X}\ \} >\\
\Delta|AZYBX =& \{\{C\},\ \{C,\ D\}\}\\
& < \text{Clausula a evaluar: } \{\ \mathsf{C}\ \} >\\
\Delta|AZYBXC =& \{\} = \emptyset
\end{aligned}$$

Finalmente el algoritmo de UP retorna la evaluación $\{\ A,\ B,\ C,\ X,\ Y,\ Z\ \}$ y como ademas no queda mas clausulas por satisfacer (dado que en el algoritmo se llego al conjunto vacío) entonces esta evaluación satisface la expresión inicial.

Se prueba a continuación reemplazando los elementos en la evaluación encontrada por True en la expresión inicial.

$$\begin{aligned}
&(\neg True \vee \neg True \vee \neg True \vee D \vee True) \wedge True \wedge (\neg True \vee True \vee \neg True) \wedge (\neg True \vee True)\\
&\wedge (\neg True \vee True \vee \neg True) \wedge (True \vee \neg True \vee True \vee \neg D)\\
&\wedge (\neg True \vee True \vee \neg True) \wedge (\neg True \vee True \vee D) \wedge (\neg True \vee True)\\
&\equiv True
\end{aligned}$$

7. Carrera de vehiculos

7.1. Enunciado

(4 pts) - Suponga tiene n vehículos ocupando las casillas $(0,0)$ hasta $(n-1,0)$ (la primera columna) de una cuadrilla $n \times n$. El objetivo es mover los vehículos a la última columna,

pero en orden inverso. Es decir, el vehículo i $(0 \leq i < n)$, que empezaba en la posición $(i, 0)$ debe terminar en $(n-i-1, n-1)$. En cada paso, cada uno de los vehículos puede moverse una casilla hacia arriba, abajo, izquierda o derecha, o quedarse quietos (es posible que los n vehículos se muevan en el mismo paso). Cuando un vehículo se queda quieto en un paso, un vehículo adyacente (pero no más de uno) puede saltar sobre él. Dos vehículos no pueden estar en la misma casilla.

A. (2 pts) - Suponiendo que el vehículo i esta en (x_i, y_i) y no hay más vehículos en la cuadrilla, escriba una heurística admisible h_i no trivial para la cantidad de movimientos que requiera llevar el vehículo a su objetivo $(n-i-1, n-1)$. Debe demostrar que esta heurística es admisible.

B. (2 pts) - Usando esta h_i, diga cuál de las siguientes heurísticas son admisibles para el problema de mover los n vehículos a sus posiciones. Justifique su respuesta.

- $\Sigma_{i=0}^{n-1} h_i$
- $max_{0 \leq i < n} h_i$
- $min_{0 \leq i < n} h_i$

7.2. Respuesta

A. Un vehículo i, lo mejor que puede acercarse a su casilla objetivo es dando en cada paso un salto por encima de otro vehículo y se posicionarse en la casilla inmediatamente adyacente a este. En palabras simples, el mejor camino es que el vehículo i salte en cada paso avanzando siempre 2 casillas debido al salto. Debido a esta observación que es una idealización del problema original se propone la siguiente heurística h_i.

$$h_i((x_i, y_i)) = \frac{|n-i-1-x_i|}{2} + \frac{|n-1-y_i|}{2}$$

Se demostrara que h_i es consciente del objetivo. Esto es que cuando el vehículo i esta en la posición $(n-i-1, n-1)$ la heurística evalúa a cero.

$$h_i((n-i-1, n-1)) = \frac{|n-i-1-(n-i-1)|}{2} + \frac{|n-1-(n-1)|}{2} = \frac{|0|}{2} + \frac{|0|}{2} = 0$$

Ahora se demostrara que la heuristica es consistente. Para esto hay que definir la funcion de costo c_i como unitaria y además comprender que la función de transición desplaza a un vehículo de una casilla a otra, es decir $f_i((x_i, y_i), accion) = (x_i + a,\ y_i + b)$ donde $a, b \in [-2, 2] \wedge (a = 0 \vee b = 0)$. Por lo tanto basta demostrar que $h_i((x_i, y_i)) \leq c_i((x_i, y_i), accion) + h_i(x_i + a,\ y_i + b)$ con a y b como se describieron anteriormente.

Se asumen las siguientes hipótesis:

$$hip_0 :\ (a = 0 \vee b = 0)$$
$$hip_1 :\ -2 \leq a \leq 2$$
$$hip_2 :\ -2 \leq b \leq 2$$

Y se estudia por casos sobre a y b

Caso 0: si $(hip_3 :\ a = 0 \wedge b = 0)$

$h_i((x_i + a,\ y_i + b)) + c_i((x_i, y_i), accion)$
$= <$ Por definición de la heurística y de la funcion de costos $>$
$\frac{|n-i-1-(x_i+a)|}{2} + \frac{|n-1-(y_i+b)|}{2} + 1$
$= <\ hip_3 :\ a = 0 \wedge b = 0\ >$
$\frac{|n-i-1-x_i)|}{2} + \frac{|n-1-y_i|}{2} + 1$
$\geq <$ Por definición de la heurística $>$
$h_i((x_i,\ y_i)) \rightarrow$ Por lo tanto se satisface en este caso

Caso 1: si $(hip_3 : \ a = 0 \wedge b \neq 0)$

$h_i((x_i + a, \ y_i + b)) + c_i((x_i, y_i), accion)$
$= \ <$ Por definición de la heurística y la funcion de costo $>$
$\frac{|n - i - 1 - (x_i + a)|}{2} + \frac{|n - 1 - (y_i + b)|}{2} + 1$
$= \ <\ hip_3 : \ a = 0 \wedge b \neq 0 \ >$
$\frac{|n - i - 1 - x_i)|}{2} + \frac{|n - 1 - (y_i + b)|}{2} + 1$
$= \ <\ 1 = \frac{|2|}{2} \ >$
$\frac{|n - i - 1 - x_i)|}{2} + \frac{|n - 1 - (y_i + b)|}{2} + \frac{|2|}{2}$
$\geq \ <\ |a + b| \leq |a| + |b| \rightarrow$ desigualdad triangular $>$
$\frac{|n - i - 1 - x_i)|}{2} + \frac{|n - 1 - y_i + (-b + 2)|}{2}$
$\geq \ <\ hip_2 : -2 \leq b \leq 2 \Rightarrow -b + 2 \geq 0$ por lo al ser siempre positivo el termino $(-b + 2)$ si se elimina se obtiene justamente la definición de $h_i((x_i, y_i)) \ >$
$h_i((x_i, \ y_i)) \rightarrow$ Por lo tanto se satisface en este caso

Caso 2: si $(hip_3 : \ a \neq 0 \wedge b = 0)$ Análogo al caso 1

Por lo tanto queda demostrado que $h_i((x_i, y_i)) \leq c_i((x_i, y_i), accion) + h_i(f_i((x_i, \ y_i), accion))$

Luego del análisis por casos queda demostrado que la heurística es consistente. Finalmente, por teorema como la heurística es consciente del objetivo y también es consistente se tiene en consecuencia que la heurística h_i es admisible.

B. Se estudiara ahora posibles heurísticas para el problema completo.

- $\Sigma_{i=0}^{n-1} h_i$: esta heurística es consciente del objetivo dado que en el estado final todos los $h_i((x_i, y_i)) = 0$ por lo que la suma de ellos sera 0. De igual modo es consistente dado que como $h_i((x_i, y_i)) \leq c_i((x_i, y_i), accion) + h_i(f_i((x_i, \ y_i), accion))$ entonces $\Sigma_{i=0}^{n-1}[h_i((x_i, y_i))] \leq \Sigma_{i=0}^{n-1}[c_i((x_i, y_i), accion) + h_i(f_i((x_i, \ y_i), accion))]$. Por definición de la función de costo $\Sigma_{i=0}^{n-1}[c_i((x_i, y_i), accion)] = c(s, accion)$. Por definición de función de transición $\Sigma_{i=0}^{n-1}[f_i((x_i, y_i), accion) = f(s, accion)$. Por lo tanto $h(s) = \Sigma_{i=0}^{n-1}[h_i((x_i, y_i))] \leq c(s, accion) + h(f(s, accion))$ que es la definición de consistencia. Finalmente como h es consciente del objetivo y es consistente entonces es admisible.
- $max_{0 \leq i < n} h_i$: Esta heuristica considera unicamente la heuristica h_i maxima. Partiendo del hecho demostrado de que $h_i((x_i, y_i)) \leq c_i((x_i, y_i), accion) + h_i(f_i((x_i, \ y_i), accion))$ entonces en particular la heuristica maxima satisface que $max_{0 \leq i < n} h_i((x_i, y_i)) \leq \Sigma_{i=0}^{n-1}[\ c_i((x_i, y_i), accion)\ ] + max_{0 \leq i < n} h_i(f_i((x_i, \ y_i), accion))$ porque la función de costos c_i es no negativa. Por lo tanto $max_{0 \leq i < n} h_i((x_i, y_i)) \leq c(s, accion) + max_{0 \leq i < n} h_i(f_i((x_i, \ y_i), accion))$. Si se define a $h(s) = max_{0 \leq i < n} h_i$ entonces por la ultima desigualdad h es consistente. De igual manera como ya se demostro que todo h_i es consciente del objetivo entonces en particular la heuristica con resultado maximo tambien sera consciente del objetivo. Finalmente como h es consistente y consciente del objetivo entonces h es admisible.
- $min_{0 \leq i < n} h_i$: De igual modo que con el inciso anterior como todos los h_i son conscientes del objetivo entonces el h_i con valor mínimo también lo sera por lo tanto la heuristica es admisible. Como todos los h_i son consistentes entonces el minimo tambien lo sera, por lo que se cumple la siguiente desigualdad $min_{0 \leq i < n} h_i \leq \Sigma_{i=0}^{n-1}[\ c_i((x_i, y_i), accion)\ ] \ + \ min_{0 \leq i < n} h_i(f_i((x_i, \ y_i), accion))$. Que sustituyendo las definiciones respectivas lleva a que $min_{0 \leq i < n} h_i \leq c(s, accion) + min_{0 \leq i < n} h_i(f_i((x_i, \ y_i), accion))$ o a lo que es lo mismo que $h(s) \leq c(s, accion) \ + h(f(s, accion))$ que indica que h es consistente. Entonces como h es consistente y consciente del objetivo entonces es admisible.

8. Traduccion a Forma Clausal

8.1. Enunciado

(2 pts) – Traduzca la siguiente fórmula de lógica de primer orden a forma clausal:

$$\forall x\ (Perrito(x) \land \exists y(Persona(y) \land Acaricia(y,x)) \Rightarrow Feliz(x))$$

8.2. Respuesta

Partimos de la expresión en lógica de primer orden y manipulamos hasta obtener la forma buscada.

$$\begin{aligned}
&\forall x\ (Perrito(x) \land \exists y(Persona(y) \land Acaricia(y,x)) \Rightarrow Feliz(x)) \\
&\equiv <\ p \Rightarrow q \equiv \neg p \lor q\ > \\
&\forall x\ (\neg(Perrito(x) \land \exists y(Persona(y) \land Acaricia(y,x))) \lor Feliz(x)) \\
&\equiv <\ \neg(p \land q) \equiv \neg p \lor \neg q\ > \\
&\forall x\ (\neg Perrito(x) \lor \neg\exists y(Persona(y) \land Acaricia(y,x)) \lor Feliz(x)) \\
&\equiv <\ \neg\exists y(P(y)) \equiv \forall y(\neg P(y))\ ;\ \neg(p \land q) \equiv \neg p \lor \neg q\ > \\
&\forall x\ (\neg Perrito(x) \lor \forall y(\neg Persona(y) \lor \neg Acaricia(y,x)) \lor Feliz(x)) \\
&\equiv <\ p \lor q \equiv q \lor p\ ;\ p \lor q \lor r \equiv p \lor (q \lor r) \equiv (p \lor q) \lor r\ > \\
&\forall x\ (\ (\neg Perrito(x)\ \lor\ Feliz(x))\ \lor\ \forall y(\neg Persona(y)\ \lor \neg Acaricia(y,x))) \\
&\equiv <\ p \lor \forall x(P(x)) \equiv \forall x(p \lor P(x))\ > \\
&\forall x\ \forall y(\neg Perrito(x)\ \lor\ Feliz(x)\ \lor\ \neg Persona(y)\ \lor \neg Acaricia(y,x))
\end{aligned}$$

Esto se puede leer como: para todo perro y humano tal que el humano acaricia al perro entonces el perro es feliz. Con esta ultima expresión aplicamos el ultimo paso del algoritmo de traducción a forma clausal y tenemos finalmente:

$$\Delta = \{\ \{\ \neg Perrito(x),\ \neg Persona(y),\ \neg Acaricia(y,x),\ Feliz(x)\ \}\ \}$$

9. Something completely different

La orquesta nacional del principado de Mónaco tiene más miembros que su ejército. Según cifras del 2014, aproximadamente 250 hombres conforman las fuerzas armadas de este principado y más de 300 hacen parte de la orquesta nacional. Esto es increíble.

Ojala los ejércitos se convirtieran en orquestas, quizás así tendríamos mas musica y menos guerra.

10. Referencias

- https://en.wikipedia.org/wiki/Boolean_satisfiability_problem
- https://luismejias21.files.wordpress.com/2017/09/inteligencia-artificial-un-enfoque-moderno-stuart-j-russell.pdf
- https://es.wikipedia.org/wiki/Factor_de_ramificacion
- https://www.youtube.com/watch?v=l-hh51ncgDI
- https://logicajhongalindo.blogspot.com/2014/07/problema-sat-tipos-de-algoritmos.html