Búsqueda Tabú

1.1. Tamaño del vecindario $N(S0)$

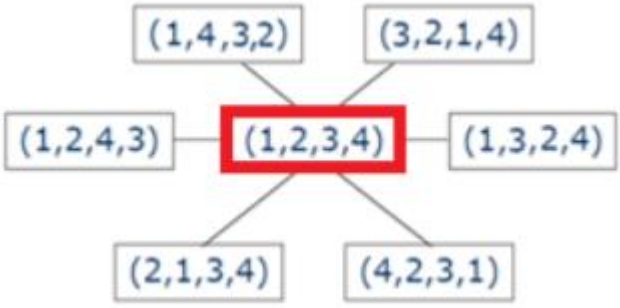
Justifica cuál es el tamaño del vecindario $N(S0)$ de una solución dada $S0$ (número de soluciones vecinas alcanzables), para un problema de representación de orden (permutaciones de valores distintos, como en el problema del viajante) de tamaño n, si el operador de generación de vecindario es el operador de inserción. ¿Cuál sería el tamaño de dicho vecindario si el operador fuese el de inversión simple? ¿Qué porcentaje representa en ambos casos el tamaño del vecindario sobre el tamaño total del espacio de búsqueda completo para un n cualquiera?

Dado que se está usando el operador de inserción, habrá $\sum_{i=0}^{n-1} n - i$ posibles soluciones (el 1º elemento se puede intercambiar con todos menos consigo mismo, el 2º con todos menos consigo mismo y el 1º porque ya se probó antes, etc.), ignorando cualquier tipo de lista tabú. Luego, dependiendo del tipo de lista tabú, habrá que restar en función de lo que se esté prohibiendo.

Si la lista tabú se prohíben soluciones directamente, tan sólo hay que restar el número de estas soluciones si se han generado. En caso de prohibirse ciertos movimientos, se excluyen esos movimientos de los posibles intercambios. Y en caso de prohibirse tuplas se excluyen las soluciones que tengan alguna pareja de valores prohibidos.

1.2. Averigua el vecindario

Considerando el vecindario E de la figura, correspondiente a la solución indicada en el centro, se pide indicar cuál es el vecindario E* para una búsqueda tabú cuando el contenido de la lista tabú de movimientos es $\mathbf{LT} = \{(\mathbf{0}, \mathbf{3}), (\mathbf{0}, \mathbf{1})\}$ en los escenarios que se indican.



Cada par en LT representa índices (comenzando en 0) de intercambio de los valores.

Si la lista tabú representa los índices que no se pueden intercambiar, entonces se han de excluir los intercambios que se realizan con esos índices. Por lo tanto, se excluyen los movimientos que intercambian el 1º con el 4º, y el 1º con el 3º. El vecindario válido es: $(1, 4, 3, 2), (1, 2, 4, 3), (2, 1, 3, 4), (1, 3, 2, 4)$.

Cada par en LT representa los índices de origen y destino de una inserción.

El primer paso es ver que soluciones son válidas si se generan por inserción. Es decir, se descartan $(1, 4, 3, 2), (4, 2, 3, 1), (3, 2, 1, 4)$. Por lo tanto, de las restantes, hay que eliminar las que inserten el elemento 0 en la posición 3 y el que inserten el elemento 0 en la posición 1. Por ello, el vecindario de soluciones final es $(1, 2, 4, 3), (1, 3, 2, 4)$.

Cada par en LT representa pares de atributos $\mathbf{LT} = \{(\mathbf{1}, \mathbf{2}), (\mathbf{1}, \mathbf{4})\}$.

Se prohíben los movimientos que generen posiciones cuyos valores 1 y 2 están pegados, al igual que el 1 y el 4. Por ello, el vecindario válido es: $(2, 1, 3, 4), (4, 2, 3, 1), (1, 3, 2, 4)$.

NOTA: *Se está suponiendo que si se prohíbe* $(1,2)$ *no se prohíbe* $(2,1)$*, ya que no se especifica que sean equivalentes.*

Enfriamiento Simulado

1.4. Posibles soluciones

Al aplicar la metaheurística "temple simulado" a un problema de minimización, se ha fijado una solución inicial $S0$ tal que $C(S0) = 400$ y se ha inicializado la temperatura con un valor T0 tal que permite aceptar con una probabilidad del 97% una solución candidata de coste $C = 1200$. Justifica si, en la primera iteración, se aceptan o no las siguientes soluciones en los casos indicados:

Una solución candidata de coste $C = 410$, para $U(0,1] = 0,99$.

No se permite, ya que 0.99 está entre 0.97 y 1, lo cual no está dentro del 97% que sí se permite.

Una solución candidata de coste $C = 395$, para $U(0,1] = 0,98$.

Se permite, en cualquier caso, ya que 395 es mejor que la solución actual.

Una solución candidata de coste $C = 1200$, para $U(0,1] = 0,90$.

Se permite, ya que 1200 es una solución que se permite en el 97% de las veces (está en el límite, pero se permite), y 0.90 está entre 0 y 0.97, por lo que se admite.

Una solución candidata de coste $C = 10000$, para $U(0,1] = 0,7$.

No se permite nunca, ya que 10000 es peor que el máximo de la solución peor permitida.

1.5. Diferentes Temperaturas

Razonar cuál sería el funcionamiento del temple simulado sin enfriamiento para dos valores de temperatura: $T = 0$, $T = +\infty$.

Si la temperatura siempre es 0, entonces nunca se va a permitir aceptar una solución peor para explorar otro campo de posibles soluciones. Por lo tanto, sólo se estarían aceptando soluciones mejores, llegando siempre a una solución local.

Si el valor de la temperatura es infinito, entonces es como decir que siempre se aceptan soluciones peores, por lo que cualquier solución obtenida será aceptada, independientemente de su valor.

Algoritmos Genéticos

1.8. Cruces

Justifica cuál es el resultado del cruce de los individuos siguientes cuando se aplica el operador correspondiente:

Operador de cruce PMX (partially mapped crossover).

P1	7	2	3	9	5	1	6	8	4
P2	6	9	7	8	5	4	3	1	2

H1	3	2	7	8	5	4	6	9	1
H2	6	8	3	9	5	1	7	4	2

Operador de cruce ordenado (*order crossover*).

P1	9	8	7	6	5	4	3	2	1

P2	1	9	2	8	3	7	4	6	5

H1	8	3	7	6	5	4	1	9	2
H2	7	6	2	8	3	4	2	1	9

Operador de cruce cíclico (*cycle crossover*).

P1	8	4	6	1	7	3	5	9	2
P2	9	3	8	6	5	2	7	1	4

H1	8	9	3	6	5	2	7	1	4
H2	9	2	6	1	7	3	5	9	4

1.9. Describir Representación-Operador

Representación Binaria con Operador Clásico y Reparador

La representación del problema se realiza en números binarios de longitud $log_2 n$. Por lo tanto, si en un problema hay 4 ciudades, se necesitarán 2 bits para representar las soluciones, y 4 cadenas (una por ciudad). Por ejemplo, el camino 3-4-2-1 se representaría como $10\ 11\ 01\ 00$ (los índices correspondientes).

El operador de cruce clásico funciona seleccionando un punto de corte aleatorio, e intercambiando las partes. Se ha luego aplicar un algoritmo de reparación clásico, intercambiando los puntos que no correspondan de uno a otro. Por ejemplo, un cruce sería pasar de los caminos $00\ 11\ 01\ 10$ y $00\ 01\ 10\ 11$ a los caminos $00\ 11\ 10\ 01$ y $00\ 01\ 11\ 10$, tras aplicar las reparaciones.

En cuanto a la mutación, lo que se hace es mutar algún bit de la solución. Sin embargo, se ha de tener en cuenta que esto implica volver a reparar la solución obtenida.

Representación de Camino con Cruce Parcialmente Mapeado (PMX)

La representación de Camino es la más natural de todas. Se representa con números indicando las ciudades a visitar. Por ejemplo, el camino 3-4-2-1 sería tal como está escrito $(3\ 4\ 2\ 1)$.

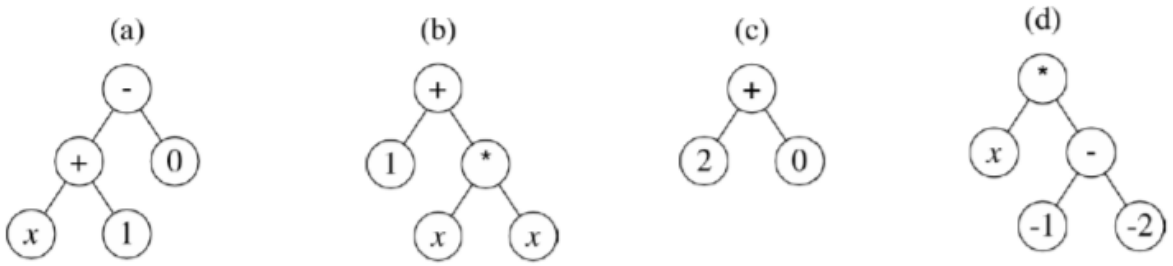
El operador de cruce parcialmente mapeado consiste en partir el camino en 3 subcaminos: uno central que permanecerá fijo, y dos laterales que intercambiaran sus ciudades. Sin embargo, el intercambio se realiza de una manera predefinida: usando la cadena central como mapeo. Es decir, en el ejemplo $(1 \mid 2\ 3 \mid 4)\ (1 \mid 4\ 2 \mid 3)$, se mapearía el elemento en la posición 2 con el 4, y el 3 con el 2. El resultado sería $(4\ 2\ 3\ 1)\ (3\ 4\ 2\ 1)$, rellenando el resto por descarte.

Programación Genética

1.11. Árboles de Expresión

En la diapositiva 6 del tema "programación genética" se muestra un ejemplo de gramática para expresiones aritméticas y una instancia válida de dicha gramática representada en forma de árbol. Revisar dicha diapositiva para recordar la representación de expresiones mediante árboles.

Utilizando una gramática similar, modificada de la del ejemplo, con una única variable $x1 = x2 = x$, las constantes limitadas a un intervalo $N = [-5, 5]$ y los cuatro operadores aritméticos ($+$, $-$, $*$, $/$), se han generado los siguientes cuatro individuos:



a) Se pide, en primer lugar, confirmar que son individuos válidos e indicar, para cada uno de ellos, cuál es la expresión aritmética a que corresponden.

a) $(x + 1) - 0$

b) $1 + (x * x)$

c) $(2 + 0)$

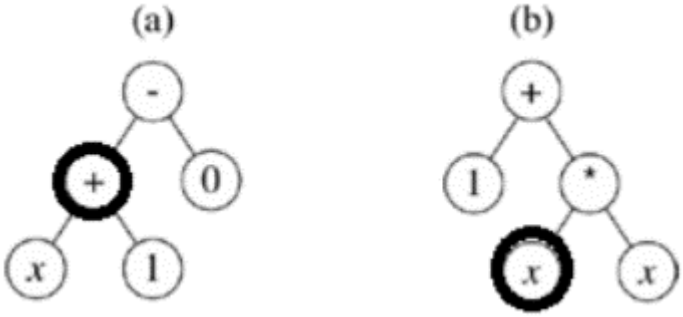
d) $x * (-1 - -2)$

b) Se pretende utilizar este diseño de programación genética para aproximar la función polinómica $f\_obj = x2 + x + 1$ en el intervalo $[-1, 1]$ discretizado $\{-1, -0.9, -0.8, \dots, 0, 0.1, 0.2, \dots, 1\}$. Se pide calcular el *fitness* de cada individuo, sabiendo que se define como el error absoluto entre cada individuo y la función f_obj.

Se ha diseñado un script de Python definiendo las 5 funciones. A continuación, se definieron todos los posibles valores de x y se averiguó el valor real para cada uno de ellos de f_obj. Finalmente, para cada función de prueba, se calculó el resultado y se averiguó el error. Los resultados que se muestran en la siguiente tabla es, para cada función, la suma de cada valor de x (el *fitness* global).

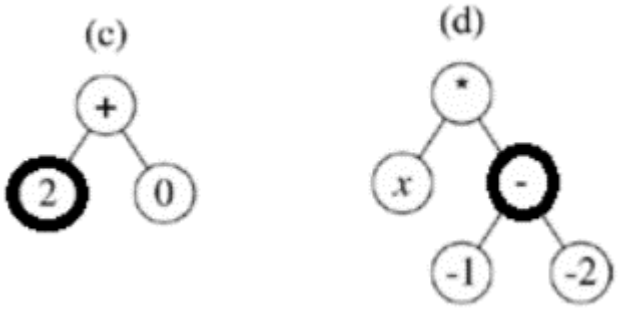
f_a	7.7
f_b	11.0
f_c	18.0
f_d	28.7

c) Indicar cuáles son los dos descendientes que se obtendrían al cruzar los individuos a y b tomando como punto de corte los siguientes. Calcula los *fitness* respectivos de los descendientes resultantes.



$x - 0$	28.7
$1 + ((x + 1) * x)$	0.0

d) Indicar una posible mutación de los individuos c y d, si el punto de mutación es el que se indica a continuación. Y sabiendo que en las expresiones se permiten los cuatro operadores aritméticos ($+$, $-$, $*$, $/$), además de la variable x y las constantes N. Calcular los fitness respectivos de los individuos mutados.



$x * 2$	28.7

Algoritmos de enjambre de partículas

1.15. Tres Partículas

Consideremos un ejemplo de un sistema PSO formado por tres partículas de $V_{max} = 10$ que se mueven en el plano (espacio de dos dimensiones real, $R2$). Tomando la expresión de ajuste de la velocidad descrita en la diapositiva 41, se pide calcular la posición siguiente de las partículas tras una iteración, en el siguiente escenario:

- **Por facilidad, tomaremos 0.25 como valor para los números aleatorios y $\phi 1 = \phi 2 = 2$.**
- **Posición de las partículas:**
  - $\mathbf{x1} = (5, 5)$
  - $\mathbf{x2} = (8, 3)$
  - $\mathbf{x3} = (6, 7)$
- **Mejores posiciones de cada partícula individual:**
  - $\mathbf{pBest1} = (5, 5)$
  - $\mathbf{pBest2} = (7, 3)$
  - $\mathbf{pBest3} = (5, 6)$
- **Mejor posición social:**
  - $\mathbf{lBest} = (5, 5)$
- **Velocidades:**
  - $\mathbf{v1} = (2, 2)$
  - $\mathbf{v2} = (3, 3)$
  - $\mathbf{v3} = (4, 4)$

En algunos modelos se incluye un factor de inercia ω que modula el término de la velocidad actual, de modo que la expresión de ajuste es: $v_{id} = \omega \cdot v_{id} + \cdots$. Indicar cuáles serían las posiciones si la inercia fuese 0.1. ¿Qué ventajas e inconvenientes puede tener un valor alto de inercia?

La expresión de la velocidad es la siguiente, incluyendo la modificación de la inercia: $v_{id} = \boldsymbol{\omega} * v_{id} + \phi * rnd() * (pBest_{id} - x_{id}) + \phi * rnd() * (g_{id} - x_{id})$. Por ello, para cada partícula se haría (tomando rnd como 0.5 siempre):

- $v_1 = 0.1 * (2, 2) + 2 * rnd * \big((5, 5) - (5, 5)\big) + 2 * rnd * \big((5, 5) - (5, 5)\big) = (0.2, 0.2)$
  - $x_1 = (5, 5) + (0.2, 0.2) = (4.8, 4.8)$

- $v_2 = 0.1 * (3,3) + 2 * rnd * \big((7,3) - (8,3)\big) + 2 * rnd * \big((5,5) - (8,3)\big) = (-4, 2.5)$
  - $x_2 = (8,3) + (-4, 2.5) = (4, 5.5)$
- $v_3 = 0.1 * (4,4) + 2 * rnd * \big((5,6) - (6,7)\big) + 2 * rnd * \big((5,5) - (6,7)\big) = (-1.1. -2.1)$
  - $x_3 = (6,7) + (-1.1, -2.1) = (4.9, 4.9)$

El valor de la inercia lo que hace es reducir el efecto de la velocidad actual sobre la nueva velocidad. Poner una inercia muy elevada implica el no haber inercia, pero poner un valor muy bajo es como ignorar la velocidad actual, permitiendo aceleraciones infinitas.

1.16. Similitudes y Diferencias

Explica las similitudes y diferencias entre los Algoritmos de enjambres de partículas y…

Los algoritmos genéticos

Los algoritmos de enjambres de partículas, al igual que en los genéticos, se debe utilizar un conjunto de individuos para generar las soluciones. Sin embargo, en los algoritmos genéticos, simplemente se interactúa a pares (o más) entre ellos sin ningún tipo de guía (aleatorio), a diferencia de los PSO, donde se tiene tanto un factor social como un factor local.

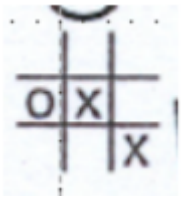
Los algoritmos de Optimización basada en Colonias de Hormigas

Los algoritmos PSO comparten con los ACO que ambos utilizan varios individuos de prueba, pero con una diferencia: en PSO se usa un comportamiento social. Las siguientes decisiones siempre vienen precedidas por una posible mejor posición del entorno (del resto de individuos). Esto en ACO no pasa, ya que por defecto todos los individuos son independientes y no se ven atraídos entre ellos hasta completar alguna solución.

Ejemplos de Búsqueda con Adversario

2.1. Justificar los valores

Justificar los valores que se muestran en la diapositiva 16, para la función de utilidad tras la secuencia de jugadas siguiente:



La imagen en cuestión parte de la jugada 2ª, donde se trata de averiguar el siguiente movimiento de MAX (la celda inferior derecha en teoría está vacía). El árbol mostrado corresponde a los siguientes posibles movimientos de MIN. Los valores que se muestran corresponden a la fórmula $e = P_{MAX} - P_{MIN}$, donde e es la función de utilidad (se trata de maximizar cuando juega MAX y minimizar cuando va MIN) y las P representan el número de filas, columnas y diagonales disponibles para cada jugador con las que podría ganar.

- $4 - 2 = 2$
  - MAX puede ocupar las dos columnas de la derecha, la fila de abajo y una diagonal
  - MIN puede ocupar la primera columna y la primera fila
- $4 - 2 = 2$
  - MAX puede ocupar la última fila, la última columna y ambas diagonales
  - MIN puede ocupar la primera fila y la primera columna

- $3 - 2 = 2$
  - MAX puede ocupar la columna central, la última fila y una diagonal
  - MIN puede ocupar la primera fila y la primera columna
- $5 - 2 = 3$
  - MAX puede ocupar la primera y última fila, la columna central y ambas diagonales
  - MIN puede ocupar la primera y última columna
- $4 - 2 = 2$
  - MAX puede ocupar ambas diagonales, la primera fila y la última columna
  - MIN puede ocupar la primera fila y la primera columna
- $4 - 2 = 2$
  - MAX puede ocupar una diagonal, la primera fila y las dos últimas columnas
  - MIN puede ocupar la primera fila y la primera columna

Por ello, como se trata de una jugada en la que MIN, el objetivo es minimizar el riesgo, por lo que el valor a tomar es 1.

2.2. Explicar justificadamente cuántos nodos se podan (no se exploran) en el ejemplo de la diapositiva 19, teniendo en cuenta que se realiza un recorrido en profundidad. Indicar cuál es la jugada que realizaría MAX.

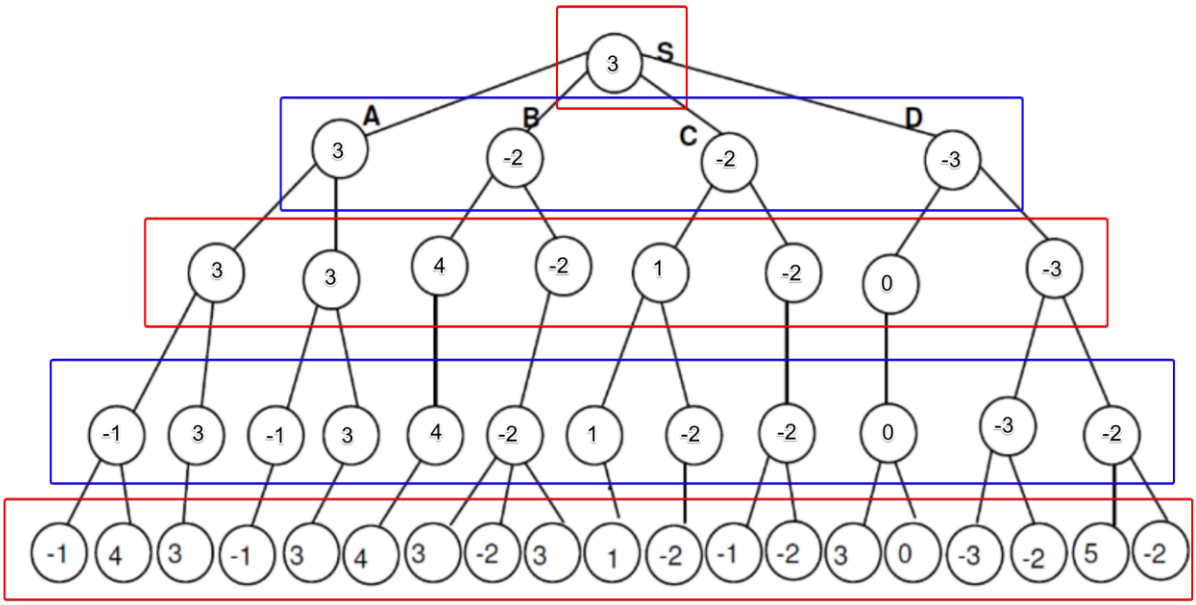
Se ha realizado la poda del subárbol central, a partir del nodo 2 (no se explora ni el central ni el derecho). Esto se debe a que, de los triángulos morados inferiores, se ha de escoger el menor valor, y de los triángulos amarillos el mayor. Con los valores que hemos encontrado hasta el 2, sabemos que el menor valor de la jugada de MIN va a estar entre menos infinito y 2, por lo tanto, si 2 fuese el menor valor, se tomaría este. Pero como en los nodos superiores se ha de escoger el máximo valor, ya se había encontrado un 3 en el subárbol anterior, por lo que es imposible que por muchos más nodos que exploremos de las posibles jugadas de MIN se mejore este resultado. Es decir, no tiene sentido seguir explorando el árbol central, porque el izquierdo siempre nos va a dar una jugada mejor.

Por lo tanto, la jugada que realizaría MAX sería irse por el árbol izquierdo (3), ya que tiene un valor superior al resto de árboles (el central puede estar entre menos infinito y 2, y el derecho es 2).

Ejercicios de Búsqueda con Adversario

2.3. Dado el siguiente árbol de búsqueda, donde se representa la evaluación que se tendría con una determinada función de estimación en los nodos terminales, y teniendo en cuenta un procedimiento de poda alfa-beta en profundidad, se pide:

a) Marcar con "X" y "+" los nodos donde se realizaría una poda alfa y beta respectivamente.



NOTA: *Se muestra el árbol resuelto primero, y en la siguiente página con las podas sin explorar todos los nodos. El nodo D está mal calculado y debería ser -2, pero a efectos prácticos no cambia el valor de S.*

NOTA: *Para el ejercicio, se utilizará la siguiente nomenclatura: los nodos para MAX están en rectángulos rojos, por lo que se denotarán con R. Dependiendo del nivel, se indicará con un número (nodo S es 1, nodos $A - D$ es 2, el siguiente rojo es 3, etc.), y para referenciar cada nodo se indicará con una barra baja seguida de su posición ordinal en la fila de nodos. Por ejemplo, el nodo con valor 0 en el tercer nivel será $R3\_7$.*

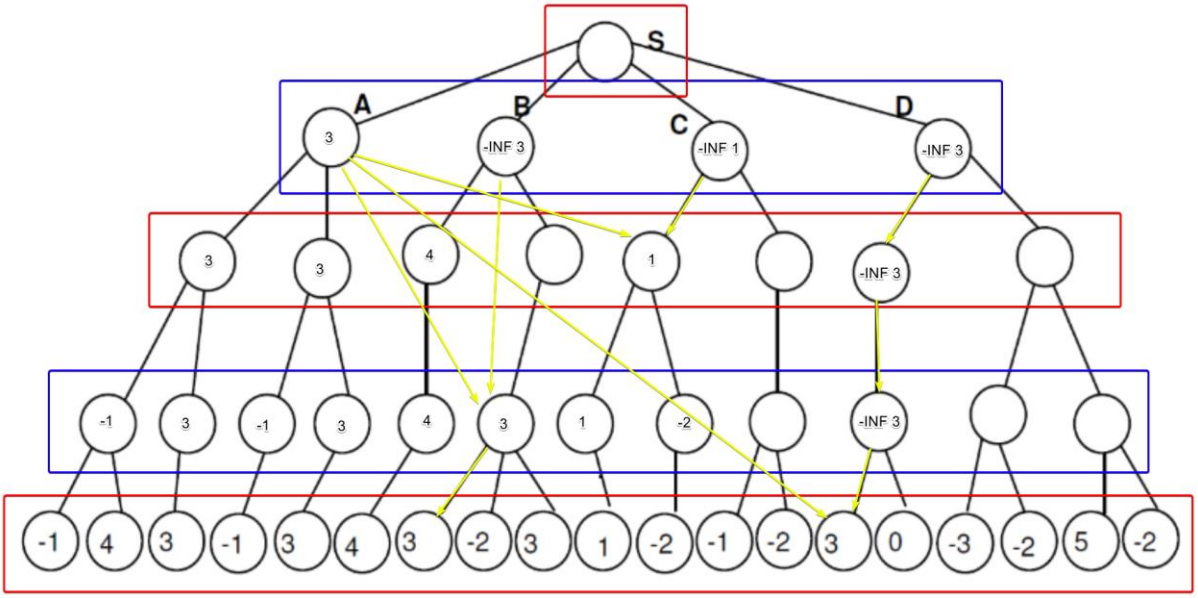
El recorrido del árbol es en profundidad, por lo que es necesario explorar hasta el nodo $R5\_2$, como mínimo. Se explora también el nodo $R5\_3$, por lo que se pone el posible valor de $R3\_1$ a 3. Se comienza a continuación con el árbol de $R3\_2$. Como se encuentra un -1 a partir de $R5\_4$, todavía se puede mejorar, por lo que se explora la sección de $B4\_4$, el cual nos da un 3, por lo que es mejor, por lo que el nodo $B2\_1$ toma el valor de 3.

Se comienza entonces por el nodo $B2\_2$. Del subárbol izquierdo se explora todos sus valores y se obtiene un 4, por lo que todavía se puede empeorar este resultado. Se ha de explorar el siguiente subárbol, y se obtiene un -2. Por lo que el valor de $B2\_2$ es -2. Sin embargo, el nodo $R5\_8$ no se llega a explorar, porque se ha encontrado un 3 el cual es el mismo que el 3 de $B2\_1$, por lo que todo el subárbol de $B2\_2$ se poda ($3 > -2$) ya que por muchos otros valores que se encuentren, se va a tomar uno igual o menor al 3, y ya se tiene un 3 de antes.

A continuación, se explora el subárbol $B2\_3$. Se comienza por la parte izquierda, y se tiene que el valor del nodo $R3\_5$ es un 1, por lo que inmediatamente el árbol se descarta (ya que el menor valor va a estar entre 1 y menso infinito, pero ya se tiene un mejor candidato en $R2\_1$).

Finalmente, el subárbol $B2\_4$, se obtiene del nodo $R5\_14$ un valor de 3, el cual se minimiza, por lo que el máximo valor posible de ese subárbol es 3. Pero ya se tiene un 3 en $B2\_1$, por lo que todo el subárbol se poda.

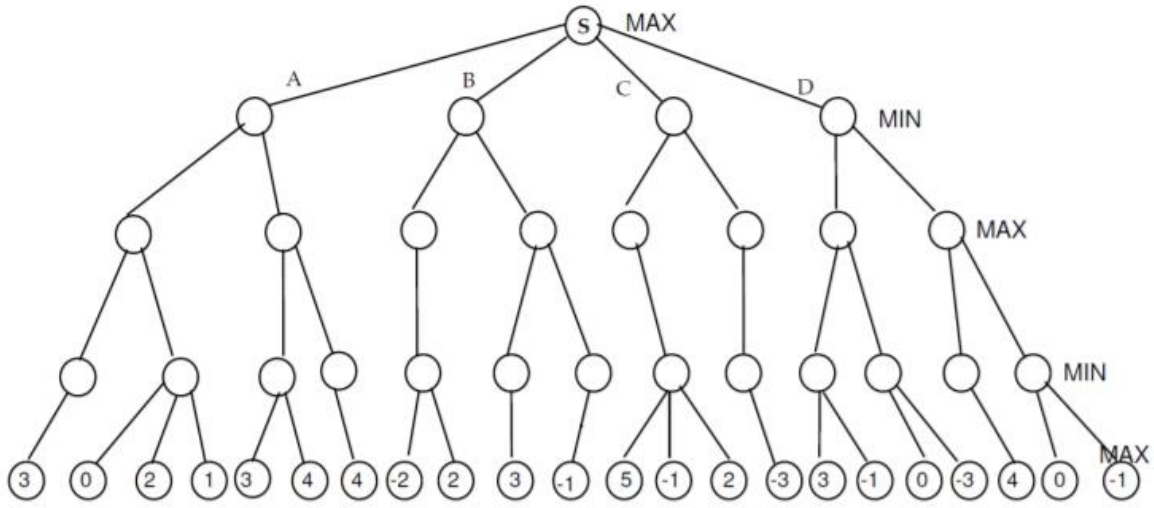
Dicho esto, se realizan podas Alpha en los nodos $B4\_6$, $B2\_3$ y $B4\_10$ tal como se ve en la siguiente figura:



b) Indicar el valor asociado al nodo S y la jugada que realizaría MAX.

El valor final del nodo S es de 3, por lo que la jugada que realizaría MAX es moverse al nodo A ya que se garantiza un resultado exacto de 3 (a diferencia de los nodos B y D, que puede ser igual o peor que 3).

2.4. En el siguiente árbol de búsqueda, que representa el espacio de estados de un juego, se representa en cada nodo terminal la evaluación que se tendría con una determinada función de estimación $f(n)$. Además, se efectúa un procedimiento de poda alfa-beta en profundidad. Se pide:



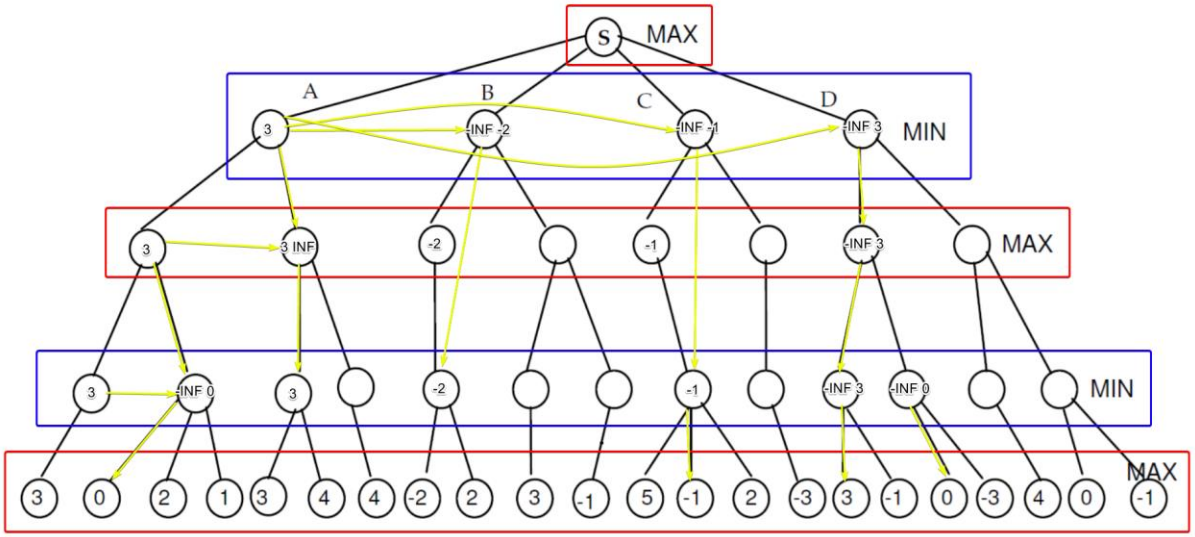
a) Marcar con "+" los nodos donde se realizaría una poda alfa.

b) Marcar con "X" los nodos donde se realizaría una poda beta.

c) Indicar las ramas que explora realmente el procedimiento alfa-beta.

NOTA: *Se usará la misma nomenclatura que para el ejercicio anterior. Se ha realizado la simulación en la imagen siguiente, con los tres apartados a la vez. En amarillo se indican las podas realizadas en base a que nodos se infieren los valores.*

El resultado final tras las podas es que el nodo S tiene valor 3, yendo por el nodo A, tal como se puede ver en la siguiente imagen:



Se empieza por el subárbol del nodo A ($B2\_1$). Se obtiene el primer valor para $R3\_1$ que va a estar comprendido entre 3 e infinito. Cuando se llega al nodo $B4\_2$, se explora el 0, y dado que es un valor inferior al 3 se procede a podar tanto el nodo con valor 2 y con valor 1. En cuanto al otro subárbol $R3\_2$, se explora el subárbol izquierdo $B4\_3$ con valor 3, pero como ya se garantiza que el mínimo valor para $B2\_1$ va a ser 3, se poda y no se explora más.

En cuanto al subárbol del nodo B $B2\_2$, se baja hasta el nodo $R5\_8$, el cual es -2. Y a partir de aquí se sabe que el valor mínimo posible del subárbol $B2\_2$ va a ser -2, pero como el subárbol $B2\_1$ tiene valor 3, se descartan todos los siguientes nodos.

El subárbol del nodo C $B2\_3$ pasa algo similar. Se analiza el nodo $R5\_12$ que vale 5, pero se tiene que seguir analizando ya que $5 > 3$ del nodo $B2\_1$. Se analiza a continuación $R5\_13$ con valor -1, el cual es menor al valor en cuestión, por lo que se descarta todo el subárbol ya que el nodo $B2\_3$ va a valer entre menos infinito y menos uno, el cual es menos que el 3 de $B2\_1$.

Finalmente, el subárbol B2_4 se analiza el nodo $R5\_16$, el cual es 3, siendo igual al nodo $B2\_1$, por lo que ya se garantiza que en ese subárbol no se va a mejorar. Se mira entonces el nodo $R5\_18$, el cual es 0, por lo que el nodo $R3\_7$ va a estar comprendido entre menos infinito y 3, siendo en el mejor caso 3, el cual ya es igual al valor garantizado en el nodo $B2\_1$. Se descarta este árbol también.

d) Razonar si, en algún caso, podría diferir la rama escogida por MAX en caso de usar alfa-beta frente a usar miniMAX.

Sería posible seleccionar un nodo diferente al A en un caso: que el nodo D también valiese exactamente 3. Al haber dos nodos con valores iguales, habría que escoger uno de ellos en función de algún criterio, y es posible que ese criterio determine que se debe continuar por el nodo D en vez del A.