7.3 GRAFOS PODA ALFA-BETA

Si bien el algoritmo MiniMax, es una importante técnica que se aplica a problemas de juegos, la misma puede ser mejorada considerando:

- Complejidad MiniMax: exponencial, que no se podía reducir a polinómica. En este sentido una de las primeras mejoras que se puede realizar es en lugar de examinar todas las ramas, se proceda a realizar una poda de algunas ramas, bajo algunos criterios que permitan establecer a través de una función de utilidad que esas ramas no son importantes para resolver el problema.
- Se asume el objetivo de calcular el valor correcto de la función MiniMax sin tener que mirar todos los nodos de un árbol.
- Se utiliza la técnica: poda de ramas, que corresponde a los algoritmos de búsqueda en grafos.

La poda Alfa Beta se basa en dos valores:

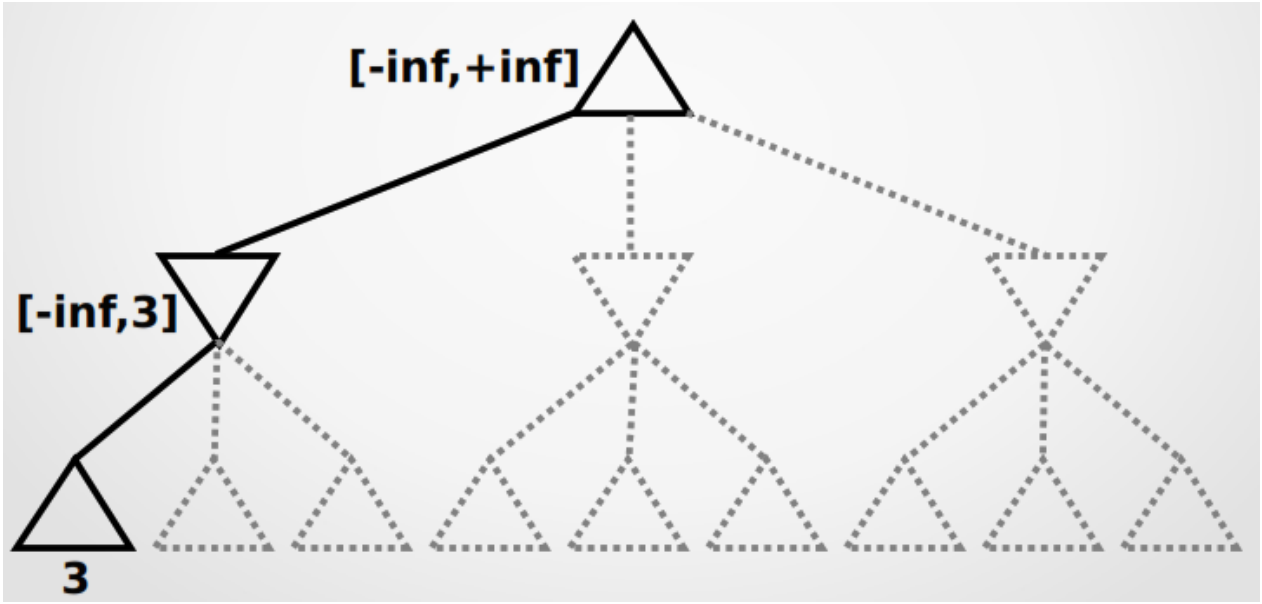
- Alfa: el valor de la mejor acción que se ha encontrado en cualquier punto de la búsqueda para Max.
- Beta: el valor de la mejor acción que se ha encontrado en cualquier punto de la búsqueda para Min.

En principio se empieza con el +infinito y el -infinito, a partir de ello se irán reduciendo estos valores a través de la valoración de la función de utilidad y una respectiva acción que asuma el algoritmo, considerando márgenes de utilidad que permitan definir si una determinada rama se la considera o se la poda.

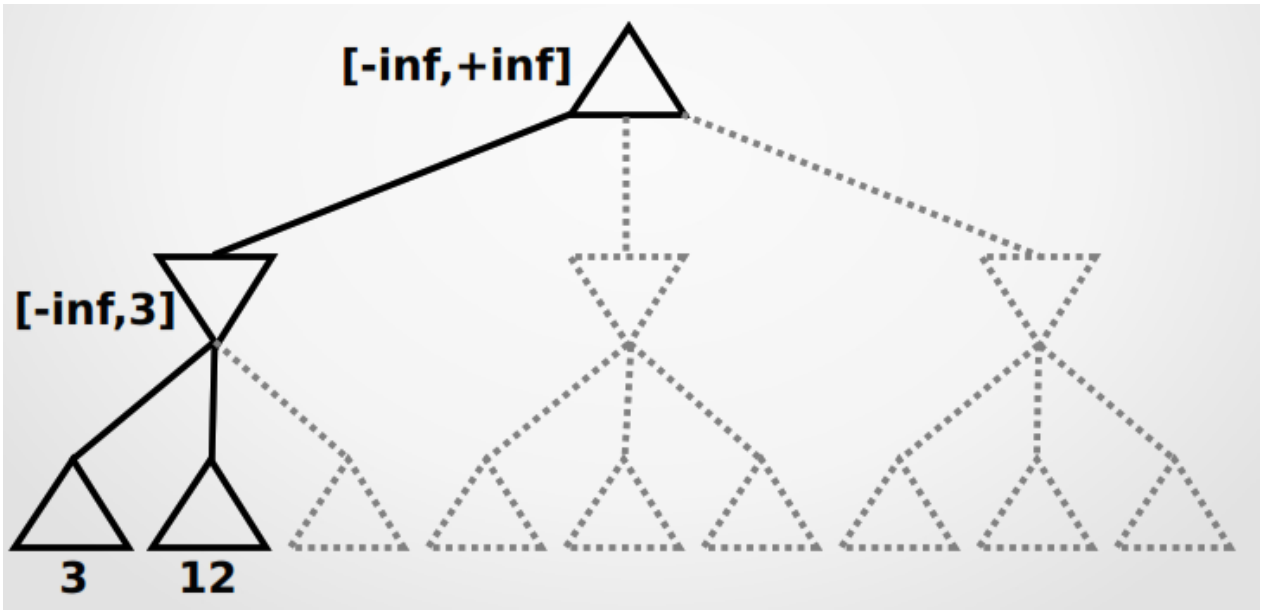
Para explicar con mayor claridad de utilizaran una serie de figuras representativas de un árbol y las distintas acciones que se apliquen y como se aplican las respectivas podas a los valores de utilidad que no ingresen en los rangos que correspondan. El siguiente árbol es el mismo que se utilizo en el ejemplo de MiniMax anteriormente. Los nodos hoja no tienen valores alfa o beta, solo los nodos intermedios poseen estos valores.

Se debe considerar que nos encontramos en el primer nodo y se inicia la exploración por la izquierda, recordemos que el primer nodo (raíz) es un nodo MAX, los nodos del siguiente nivel son nodos MIN y los últimos son nodos hoja. En la siguiente figura se muestran los nodos que se desplegarían en una primera instancia tanto MAX como MIN. Se asocia MAX con alfa y MIN con beta.

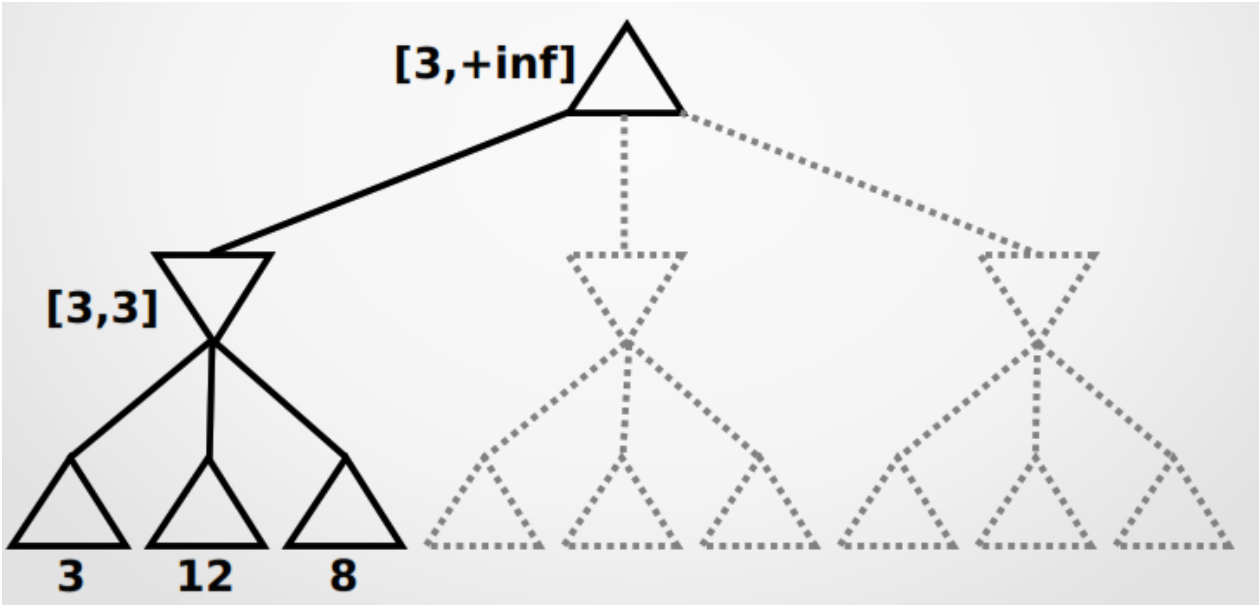
En principio MAX posee el valor de -infinito y MIN el valor de +infinito, en la primera valoración de la función de utilidad se establece un valor de 3 para beta. En base a esta primera asignación para las siguientes expansiones, todo valor que quede fuera del rango de alfa y beta, ya no sirve y por lo tanto ya no se expande, es decir si obtiene un valor mayor a 3, no lo toma y si el valor es menor a 3 si lo hace.



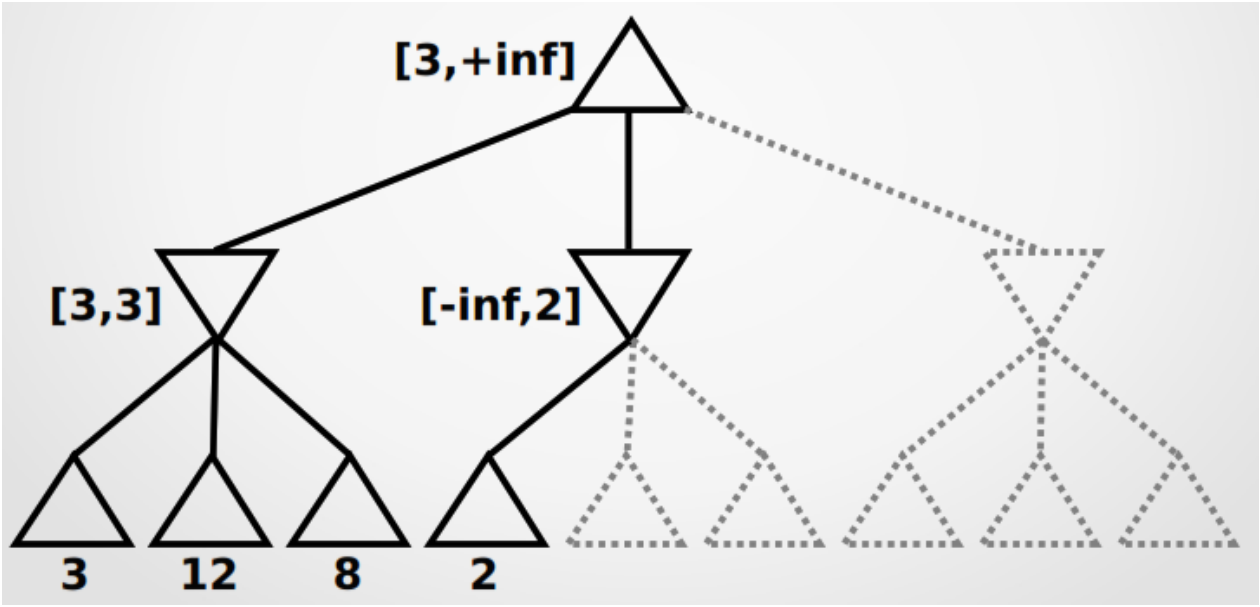
Se realiza el siguiente recorrido



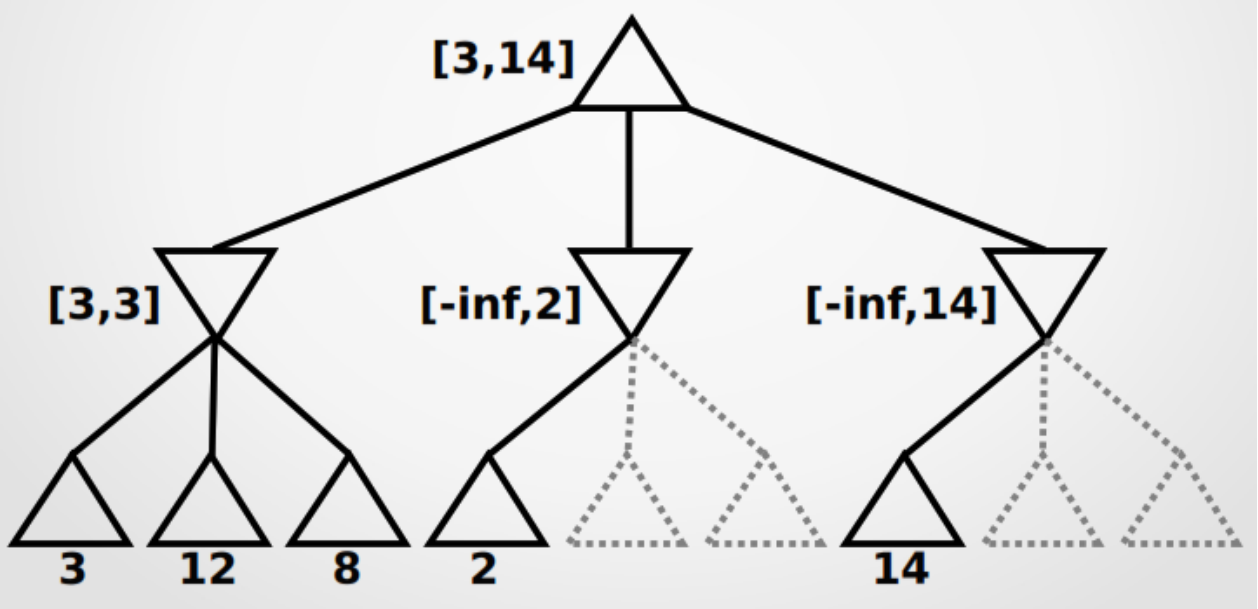
El valor es mayor, por lo que no se la considera. Procediendose a explorar el siguiente nodo.



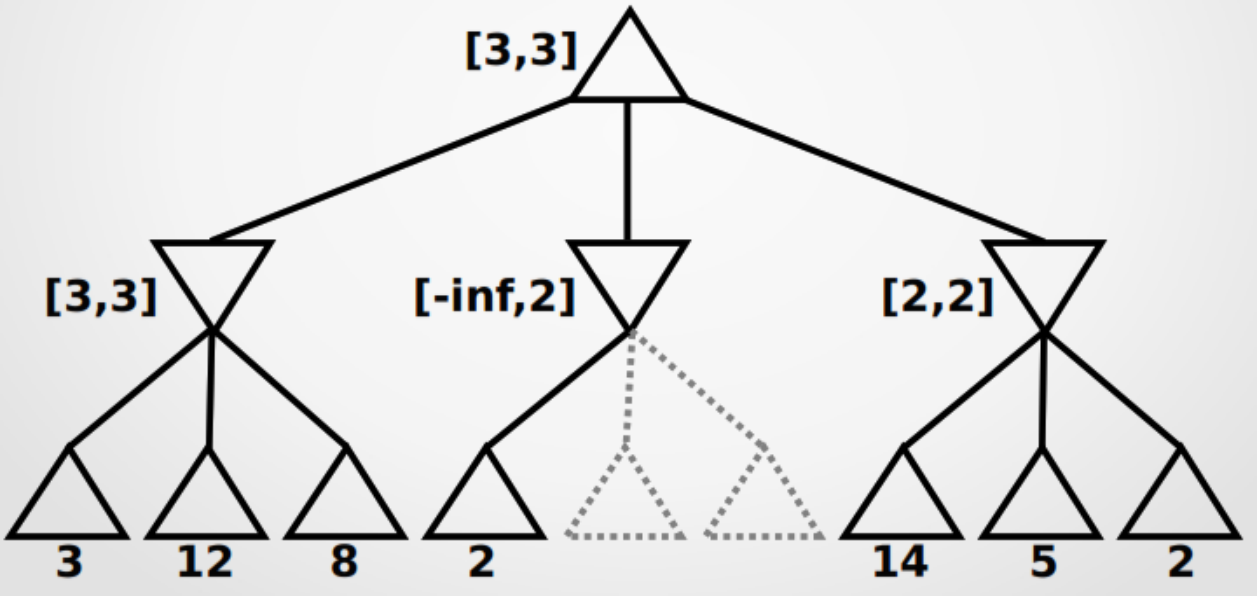
Procediéndose de la manera anterior. Peo como se ha terminado de explorar todas las ramas y el mejor valor es 3, se asigna el mismo a MAX y ese valor se lleva hasta el valor alfa del nodo raíz, y con ese valor de alfa en el nodo raíz, se establece que los valore 0, 1 y 2 no le sirven a MAX, antes de continuar con el siguiente nodo hacia la derecha del nivel 2. Se procede con la exploración del siguiente nodo MIN.



Con esta expansión se determina que ya no se debe explorar las siguientes hojas por cuanto el menor valor que podrá obtener a partir del nodo elegido es 2 y ese valor es menor a 3, por lo que no vale la pena explorar los siguientes, procediéndose a explorar el siguiente y ultimo nodo MIN.



Se identifica un valor 14 que es un buen valor, con lo que se actualizan los valores alfa y beta y se continua con la siguiente hoja. Y como es un valor de beta mejor al que tenía el nodo raíz, se actualiza con este el valor beta del nodo raíz.



Al explorar las siguientes notas, se encuentran valores menores al 14 del primer nodo, por lo que se debe actualizar el valor de beta en lo nodos antecesores, quedando después de explorar los dos nodos restantes el valor de 2 para alfa y 2 para beta. En pocas palabra lo que se busca es almacenar en alfa del nodo raíz el mayor valor de todos los menores generados con la función de utilidad y en la variable beta el menor valor de todos los mayores calculados con la función de utilidad, para reducir la brecha que generan los márgenes de estos valores. Algo importante es el orden en el que se exploren los nodos, si se hubiesen empezado en los dos

últimos nodos MIN por los nodos que generan 2 como el caso del nodo del medio, se hubieran realizado dos podas adicionales en el último nodo MIN.

El algoritmo es muy similar al minimax, excepto la inclusión de los parametros alfa y beta y devuelve adicionalmente a el valor una acción:

```
función ALFA-BETA(problema)
devuelve acción
  inicial ← problema.ESTADO-INICIAL
  (accion, valor) ← VALOR-MAX(problema,
                              inicial,
                              –inifinito,
                              +infinito)
  devolver accion
```

```
funcion VALOR-MAX(problema, estado, alfa, beta)
devuelve (accion, valor)
  si problema.ES-OBJETIVO(estado) entonces
    devolver problema.UTILIDAD(estado)
  mayor-valor  ← –infinito
  mejor-accion ← nulo
  por cada accion en problema.ACCIONES(estado) hacer
    resultado ← problema.RESULTADO(estado, accion)
    utilidad  ← VALOR-MIN(problema, resultado, alfa, beta)
    si utilidad > mayor-valor entonces
      mayor-valor  ← utilidad
      mejor-accion ← accion
    si mayor-valor >= beta entonces
      devolver (mejor-accion, mayor-valor)
    si mayor-valor > alfa entonces
      alfa ← mayor-valor
  devolver (mejor-accion, mayor-valor)
```

```
funcion VALOR-MIN(problema, estado, alfa, beta)
devuelve (accion, valor)
  si problema.ES-OBJETIVO(estado) entonces
    devolver problema.UTILIDAD(estado)
  menor-valor  ← +infinito
  mejor-accion ← nulo
  por cada accion en problema.ACCIONES(estado) hacer
    resultado ← problema.RESULTADO(estado, accion)
    utilidad  ← VALOR-MAX(problema, resultado, alfa, beta)
    si utilidad < menor-valor entonces
      menor-valor  ← utilidad
      mejor-accion ← accion
    si menor-valor <= alfa entonces
      devolver (mejor-accion, mayor-valor)
    si menor-valor > beta entonces
      beta ← menor-valor
  devolver (mejor-accion, menor-valor)
```

En la función VALOR-MAX y VALOR-MIN las últimas cuatro líneas son las que difieren con las del algoritmo MINIMAX, en estas líneas de seudocódigo se actualiza los valores de alfa y beta, para poder ejecutar las respectivas podas.

Las mejoras que se pueden realizar son las siguientes:

- La complejidad logrará una reducción que dependerá del orden.
- Si se logra ordenar todo se puede lograr una complejidad de orden: $O(b^{m/2})$, con una ramificación media: $b^{1/2}$
- Se alcanza doble de profundidad en un mismo tiempo. Sin embargo, esto es casi imposible.
- Porque no existe mucha información en este tipo de problemas se utiliza un orden aleatorio: con lo que se alcanza $O(b^{3m/4})$ de media.

Se pueden realizar mejoras utilizando técnicas de aprendizaje concretas al MINIMAX como:

- Tablas de Transposición:
  - Se guarda algunas permutaciones de conjunto de acciones que llevan al mismo resultado.
  - Similar al conjunto explorado en búsqueda no informada.
  - Ahorra muchos cálculos.
  - Son técnicas de programación dinámica.

Se utiliza aprendizaje por refuerzo para mejorar el rendimiento, que considera:

- Refuerzo, probar primero acciones que dieron buenos resultados en el pasado:
  - Juegos anteriores.

- Búsqueda en profundidad iterativa:
  - Se guarda las mejores acciones de cada iteración.
  - Tiempo límite: siempre una acción.