Pontificia Universidad
JAVERIANA
Bogotá

Inteligencia Artificial

Solución de Problemas

Juegos MultiJugador

Ing. Enrique González, PhD

Ing. Andrea Rueda, PhD

Departamento de Ingeniería de Sistemas

Presentación Generada a partir de Aportes de Estudiantes de Diferentes Versiones del Curso
Contribución especial del Estudiante MISyC Carlos Iván Rivera

Agenda – Juegos

1 – Conceptos Básicos

- Definiciones y Características

2 – Juegos como Problemas de Búsqueda

- Método MIN-MAX
- Poda Alfa-Beta

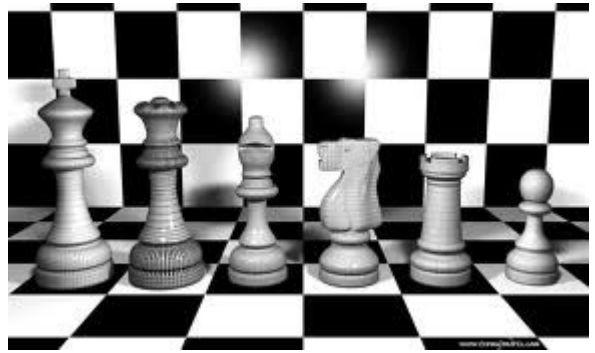
3 – Juegos Variantes Avanzadas

- Juegos en Tiempo Real
  - RTMM
- Juegos de Alta Complejidad
  - Árboles de Montecarlo

1 – Conceptos Básicos

Definiciones

- **Juego**
  - ”A universal form of recreation generally including any activity engaged in for **diversion or amusement** and often establishing a situation that involves a **contest or rivalry**” [Enciclopedia Británica].



Características

- **Reglas fijas** → no ambiguas y bien conocidas
- **Resultados variables y cuantificables** → resultados no controlables por un solo jugador
- **Valorización de los resultados** → resultados mejores/peores que otros
- **Esfuerzo del jugador** → acciones influencian estados
- **Adhesión del jugador al resultado** → se busca alcanzar el mejor resultado

2 – Juegos como Búsqueda

Árbol de Búsqueda

- **Estructura**
  - Desarrollo de las opciones de juego con alternancia de niveles entre los jugadores participantes.
- **Estrategia de Juego**
  - Buscar un camino en el árbol que permita llegar a un nodo “ganador”.

2 – Juegos como Búsqueda

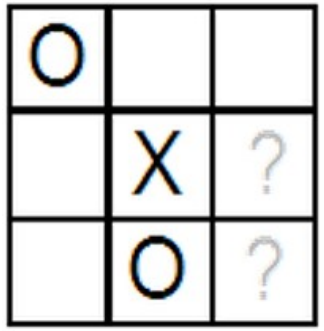
Pontificia Universidad JAVERIANA Bogotá

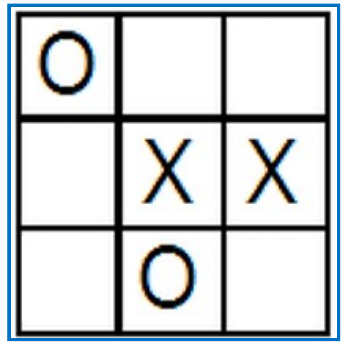
Componentes

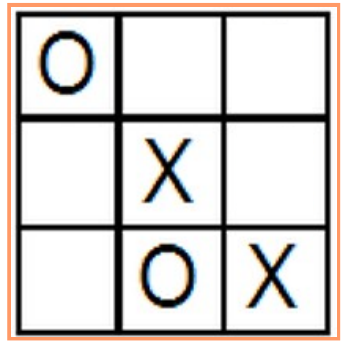
- **Estados / acciones**
  → diferentes situaciones válidas de desarrollo del juego – movimientos “legales”
- **Estado inicial**
  → nodo raíz
- **Función sucesor**
  → operadores de movimiento – cambio de estado – acciones aplicables a partir del estado actual
- **Prueba terminal**
  → identificación de estados terminales – finalización del juego – ganar, perder o empate
- **Función utilidad**
  → función heurística objetivo – medida de rentabilidad de un estado
  - Cercanía a estados ganadores – ej: distancia de Manhattan
  - Posicionamiento estratégico – ej: dominio de una zona importante del tablero

Estrategia MIN-MAX

MAX: Jugador 'X'
Busca valores mayores
desde la perspectiva de X

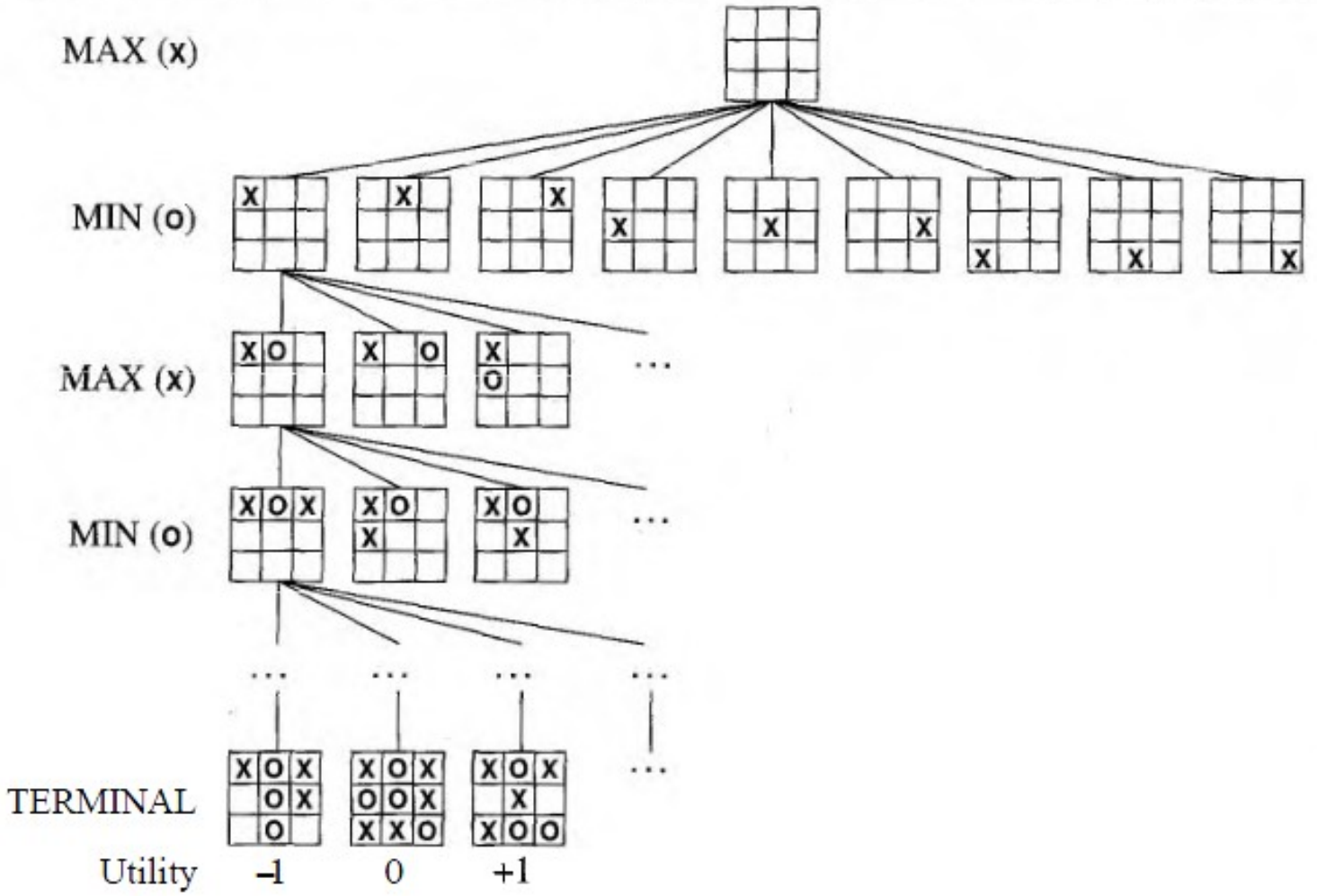






MIN: Jugador 'O'
Busca valores menores
desde la perspectiva de X

Árbol de Juego MIN-MAX



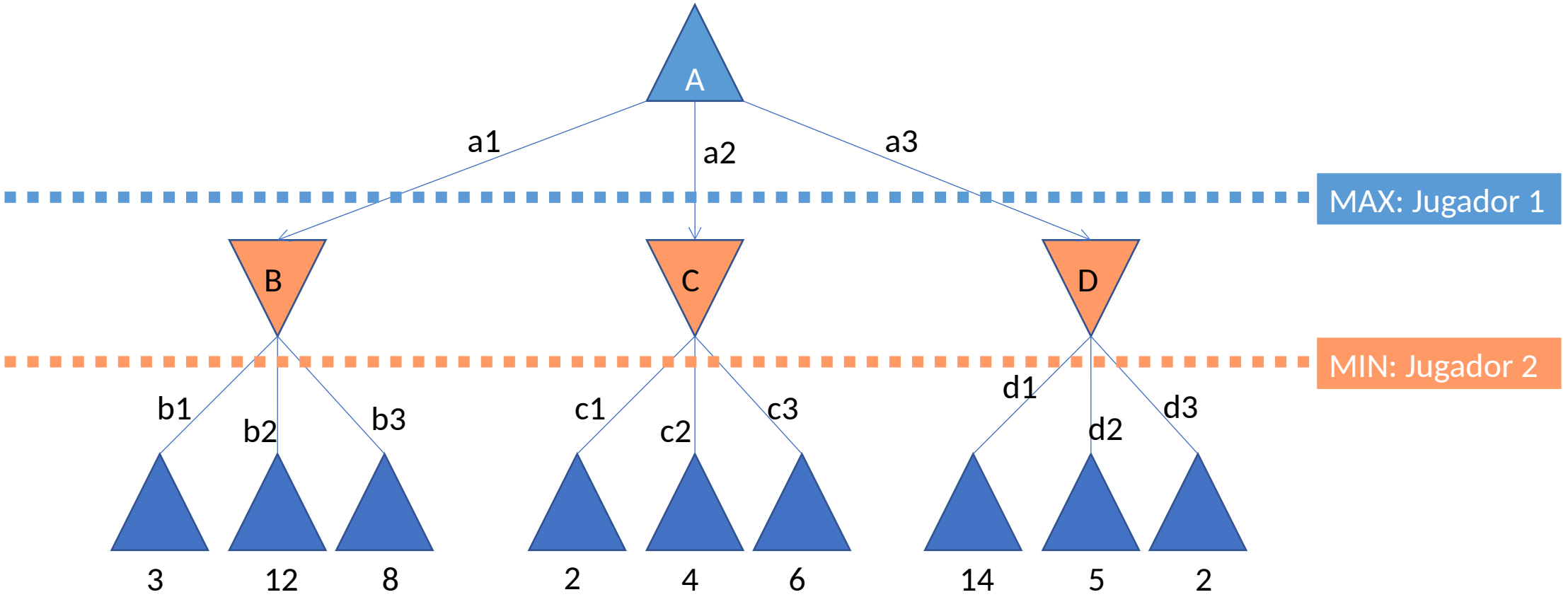
MAX: Jugador 1
Busca valores mayores

MIN: Jugador 2
Busca valores menores

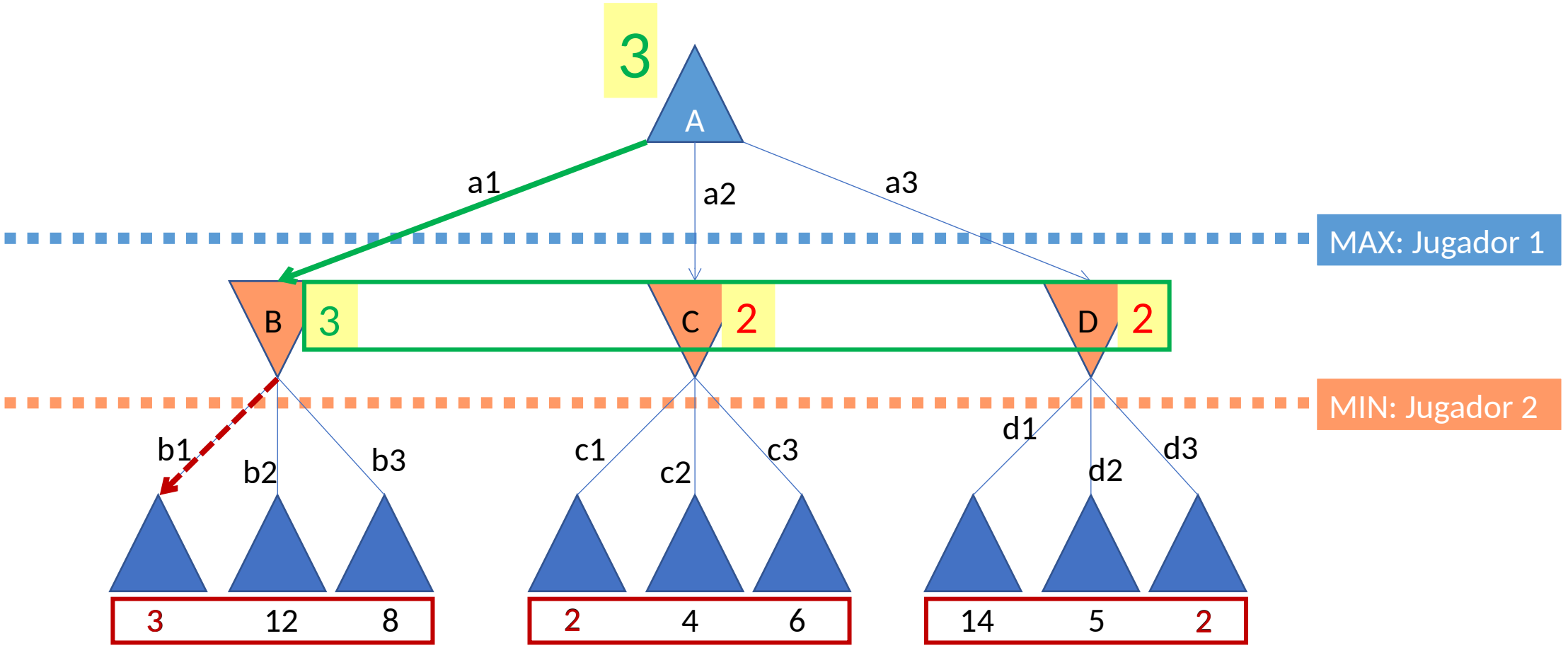
MAX: Jugador 1
Busca valores mayores

MIN: Jugador 2
Busca valores menores

Ejemplo Árbol de Juego MIN-MAX



Ejemplo Árbol de Juego MIN-MAX



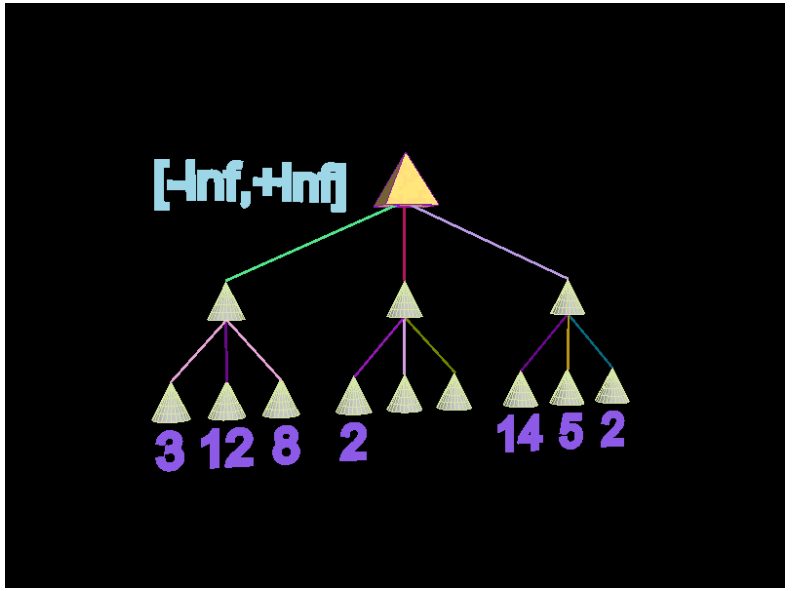
Árbol de Juegos – Poda Alfa-Beta

Pontificia Universidad JAVERIANA Bogotá

Poda del Árbol de Búsqueda

- **¿Por qué es necesaria?**
  - El algoritmo MIN-MAX requiere expandir todo el árbol de búsqueda para obtener el resultado.
  - En la práctica solo es posible expansión limitada
  - Si el árbol es más pequeño → es posible más profundidad

- **Operación**
  - Eliminar búsquedas innecesarias sin cambiar el resultado del juego.
    - Eliminando incluso subárboles enteros
  - Podar si el valor actual ya no puede superar ALFA-BETA.

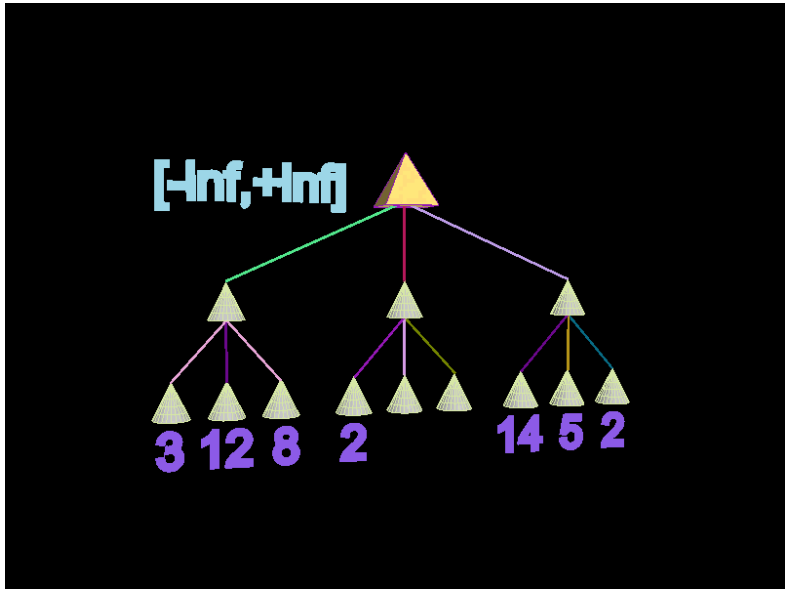


Árbol de Juegos – Poda Alfa-Beta

Pontificia Universidad JAVERIANA Bogotá

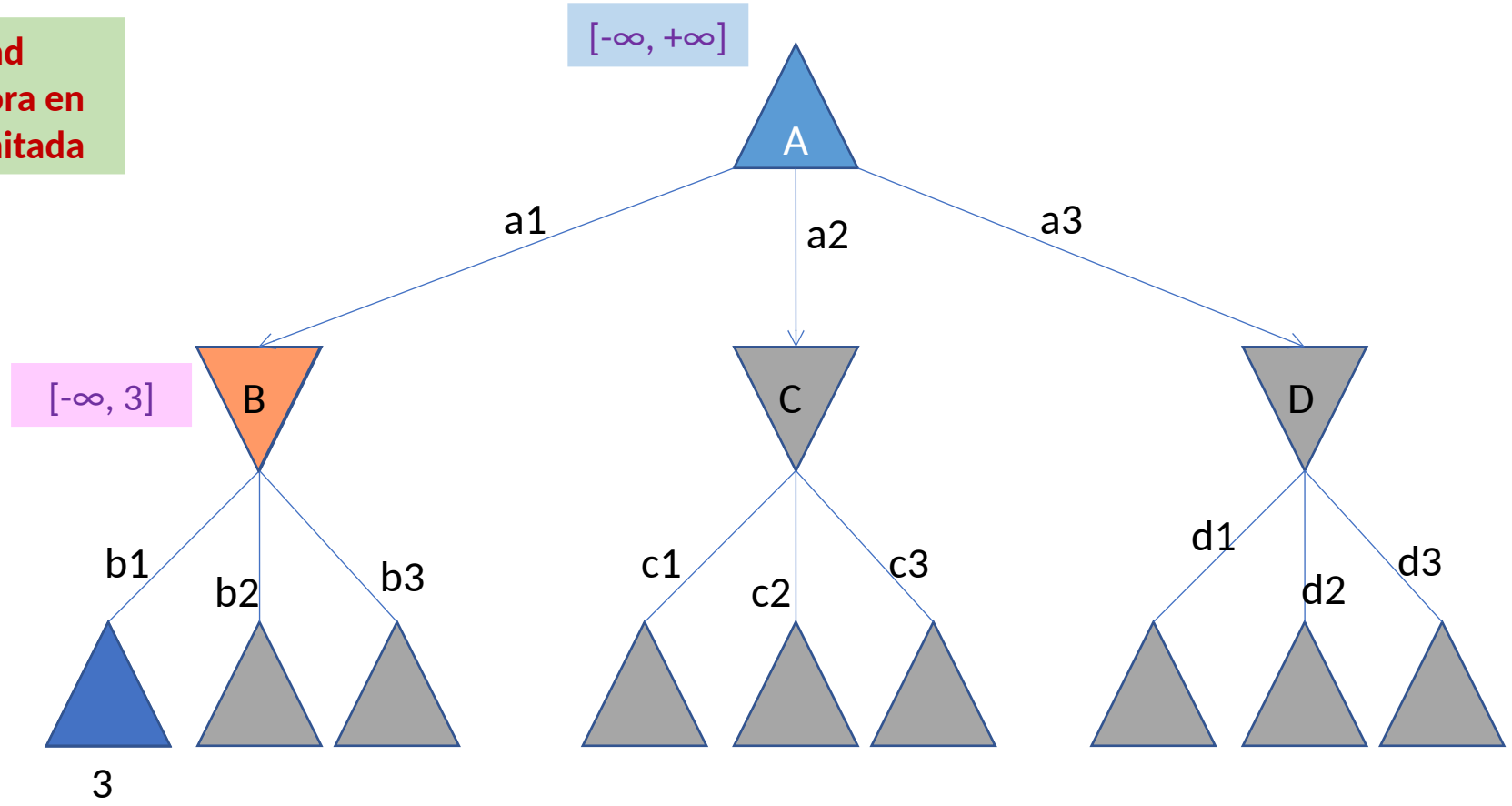
Parámetros de Poda

- **ALFA**
  - Mejor Valor - Más Alto encontrado
  - en puntos de decisión a lo largo de caminos para un nodo MAX
- **BETA**
  - Mejor Valor - Más Bajo encontrado
  - en puntos de decisión a lo largo de caminos para un nodo MIN



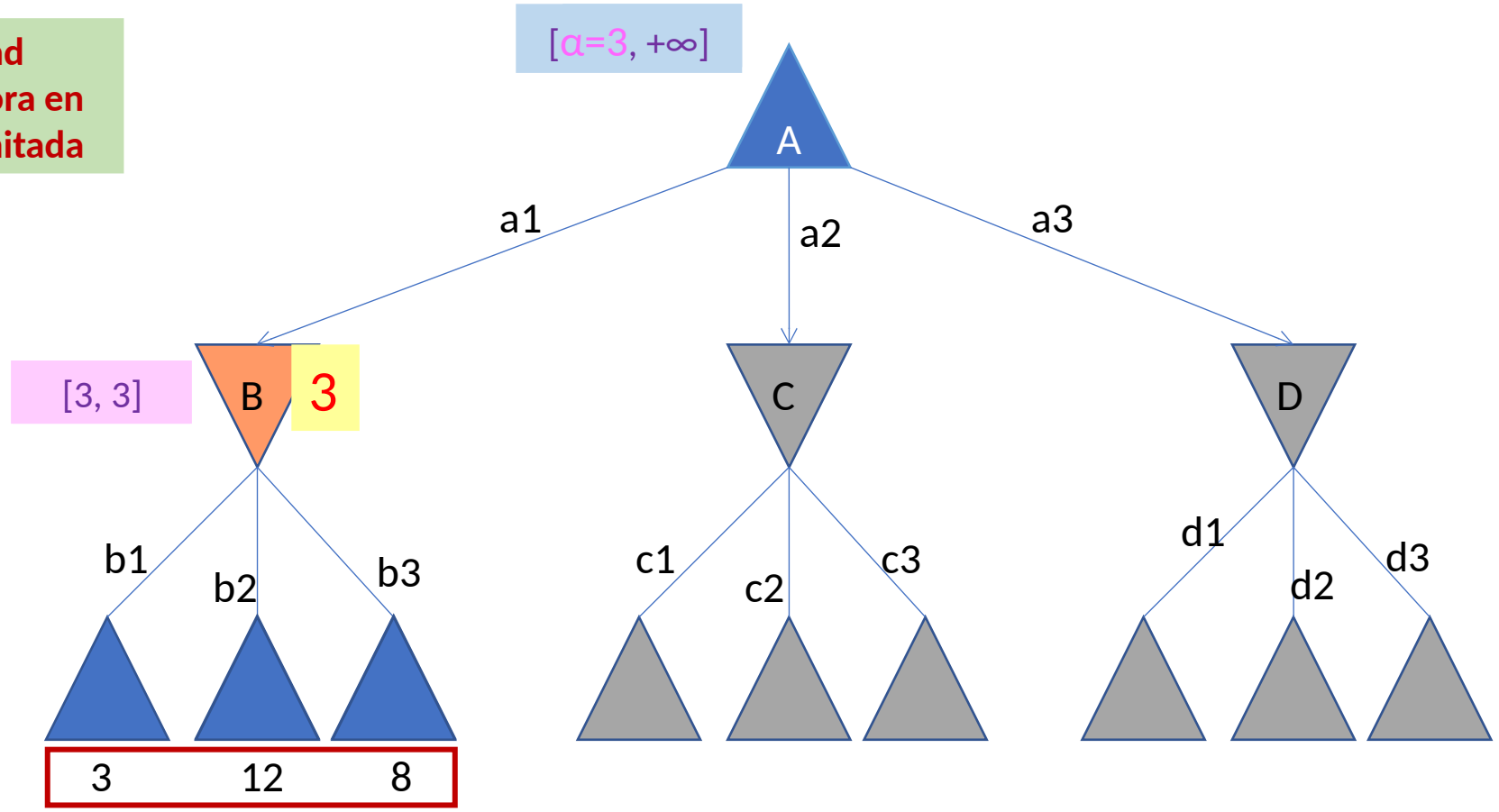
Ejemplo Poda Alfa-Beta

En la Realidad
El Árbol se Explora en Profundidad Limitada



Ejemplo Poda Alfa-Beta

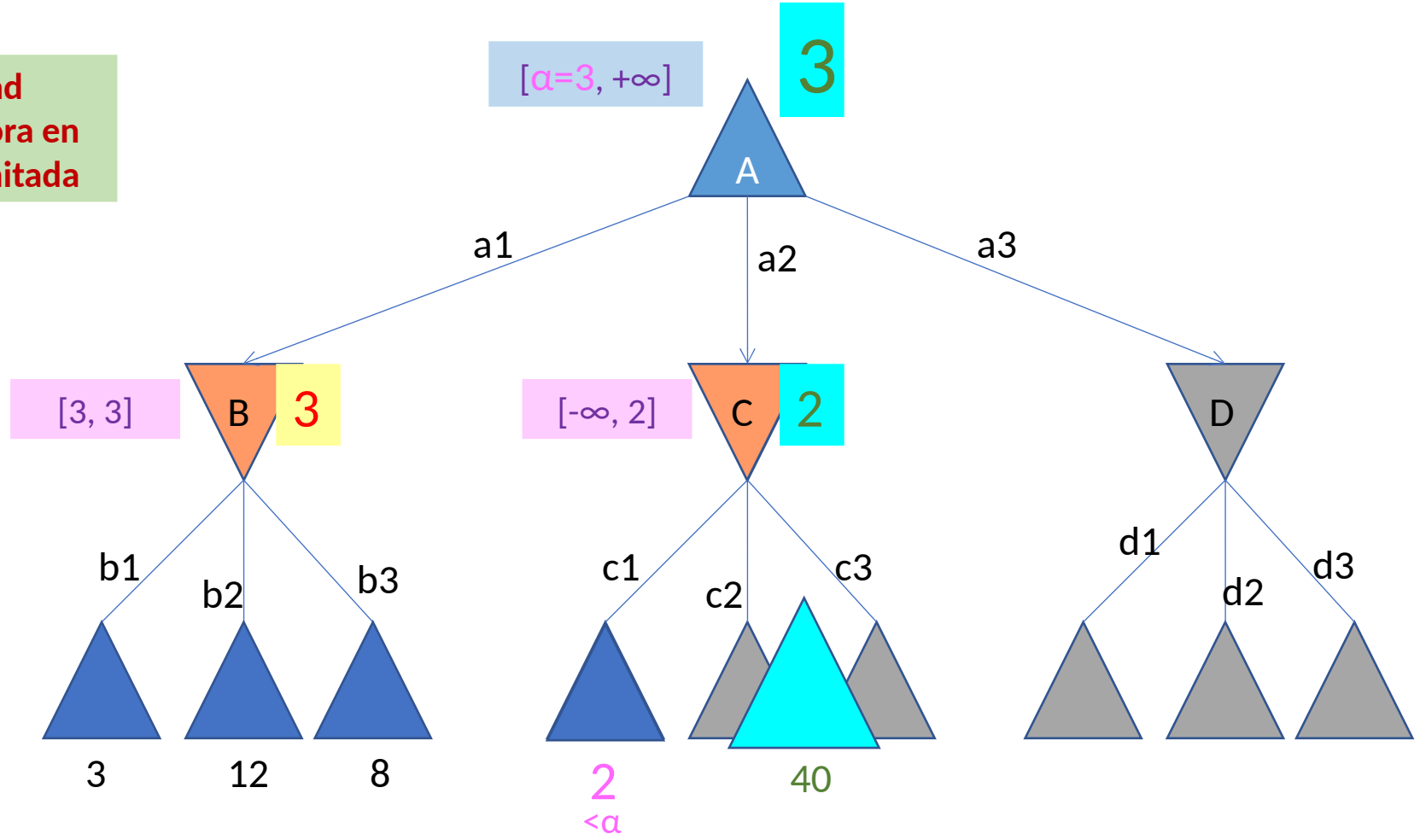
En la Realidad
El Árbol se Explora en
Profundidad Limitada



Ejemplo Poda Alfa-Beta

Pontificia Universidad JAVERIANA Bogotá

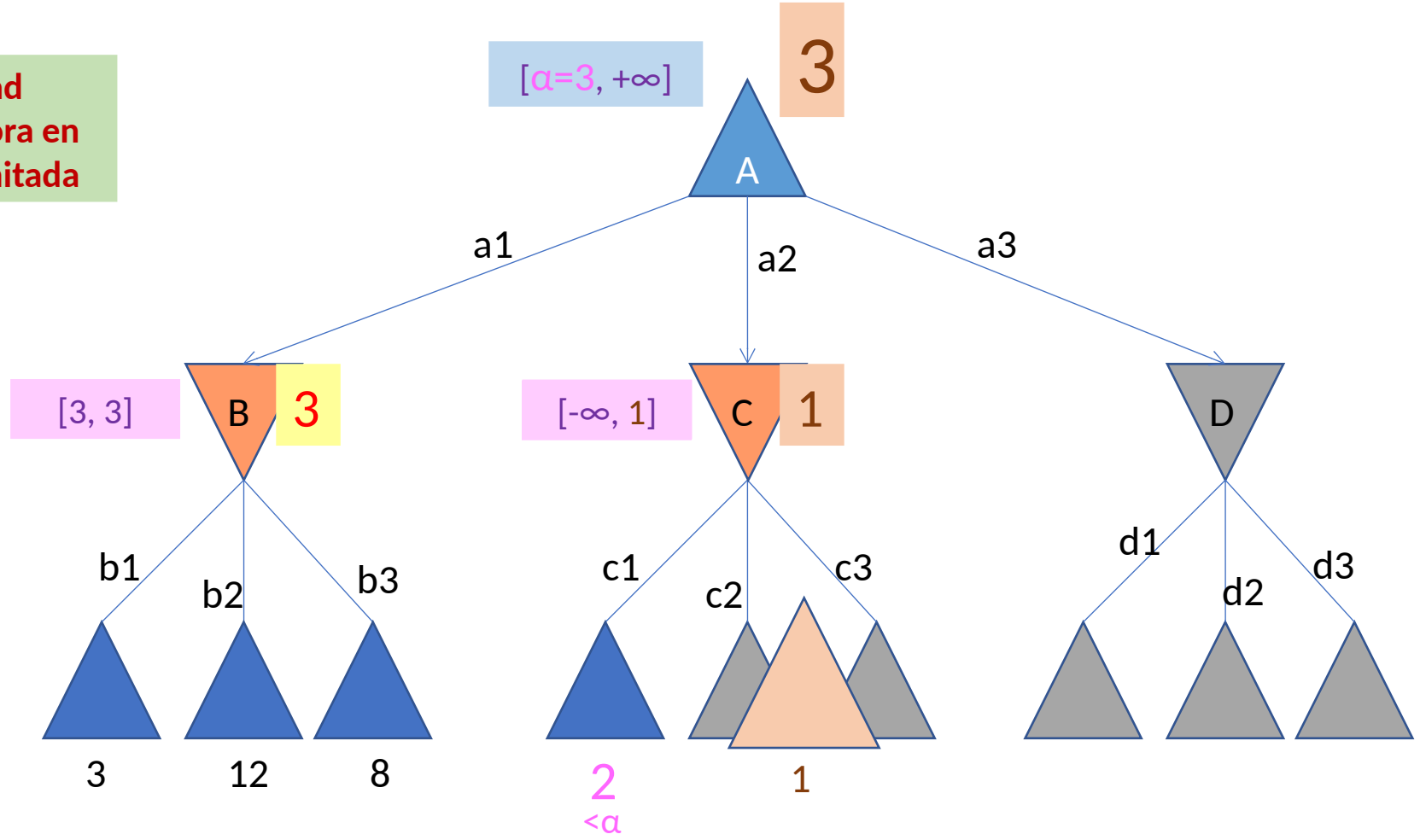
En la Realidad
El Árbol se Explora en Profundidad Limitada



Ejemplo Poda Alfa-Beta

Pontificia Universidad JAVERIANA Bogotá

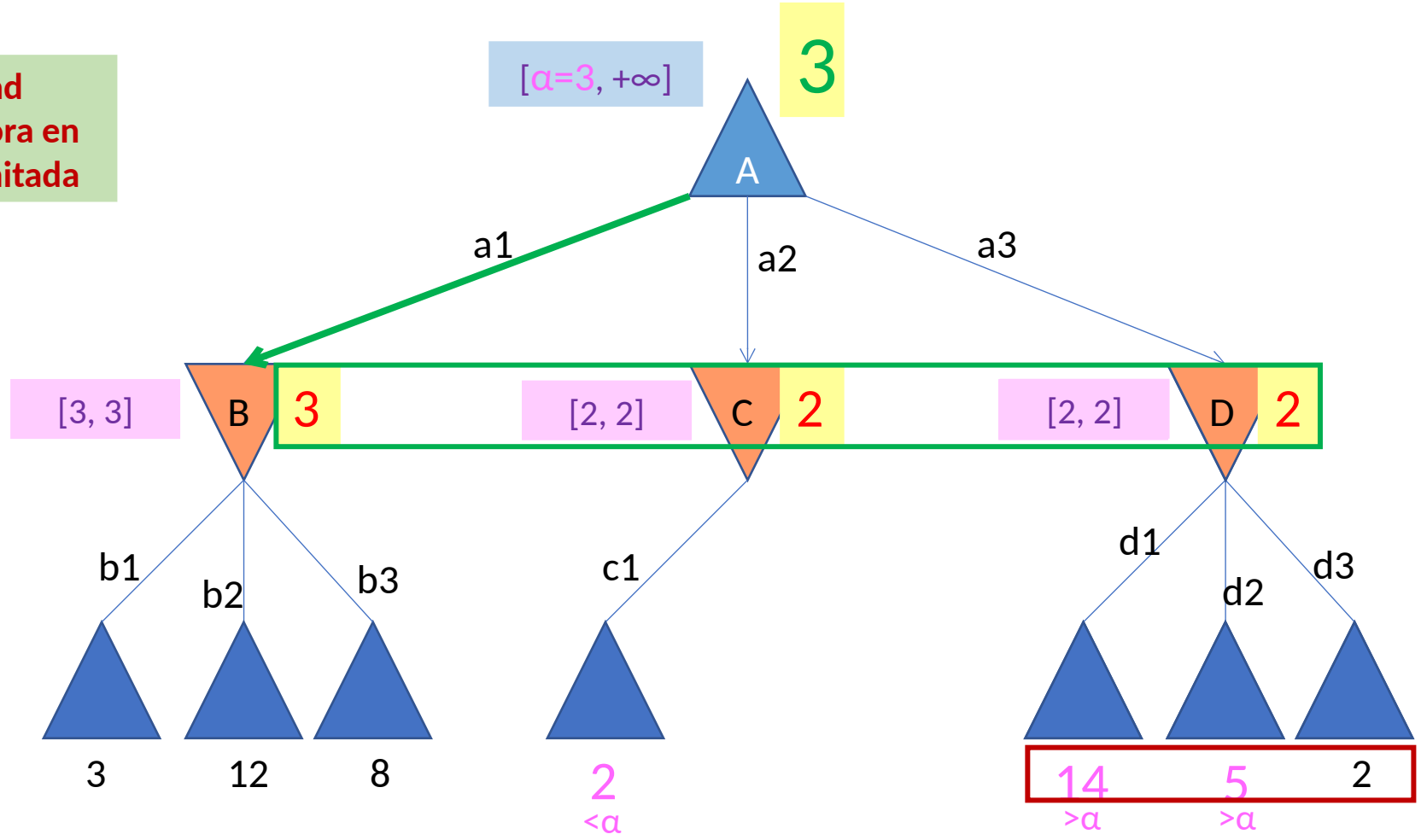
En la Realidad
El Árbol se Explora en Profundidad Limitada



Ejemplo Poda Alfa-Beta

Pontificia Universidad JAVERIANA Bogotá

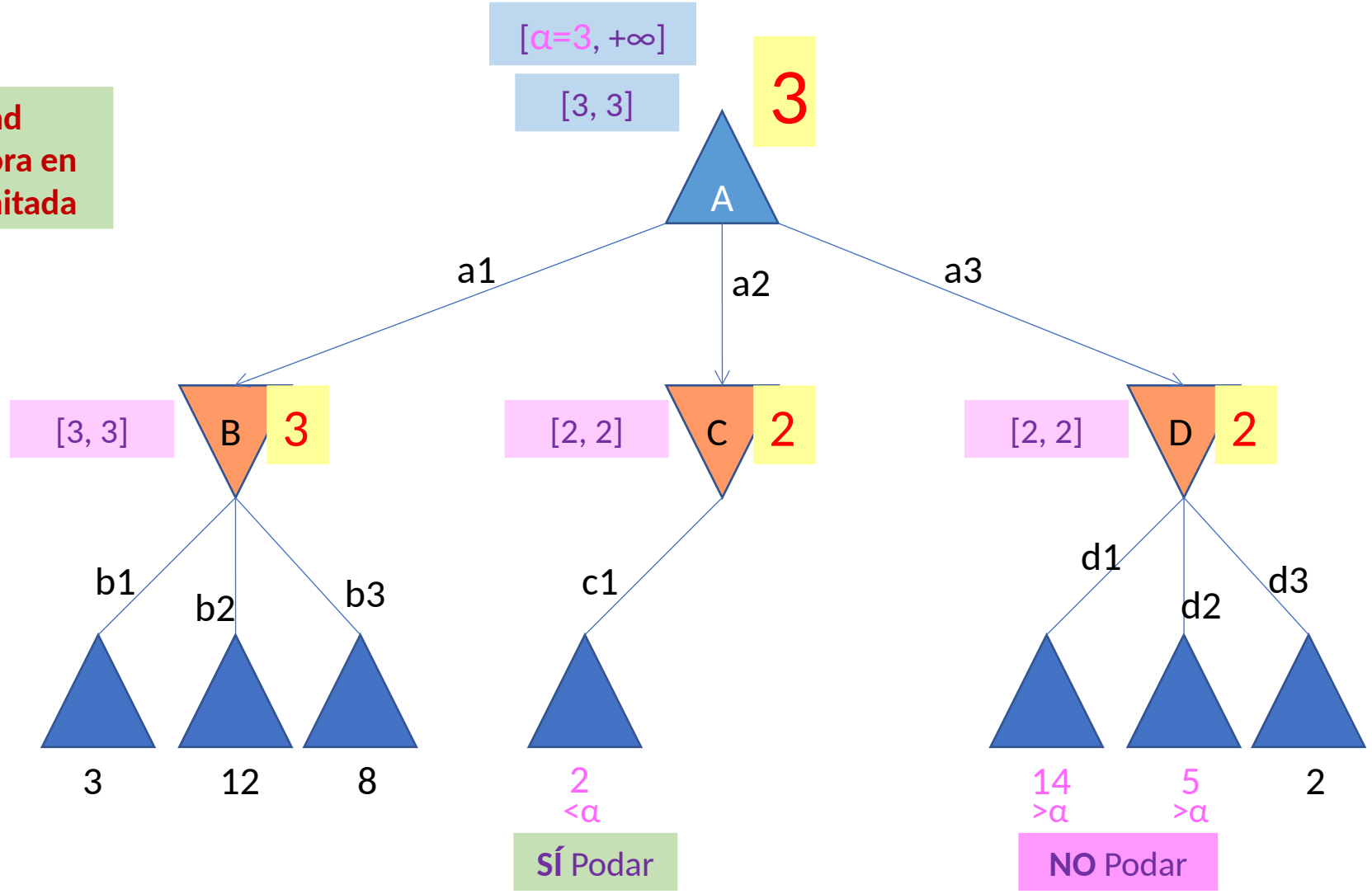
En la Realidad
El Árbol se Explora en Profundidad Limitada



Ejemplo Poda Alfa-Beta

Pontificia Universidad JAVERIANA Bogotá

En la Realidad
El Árbol se Explora en Profundidad Limitada



Juegos en Tiempo Real – RTMM

Problema Tiempo Real

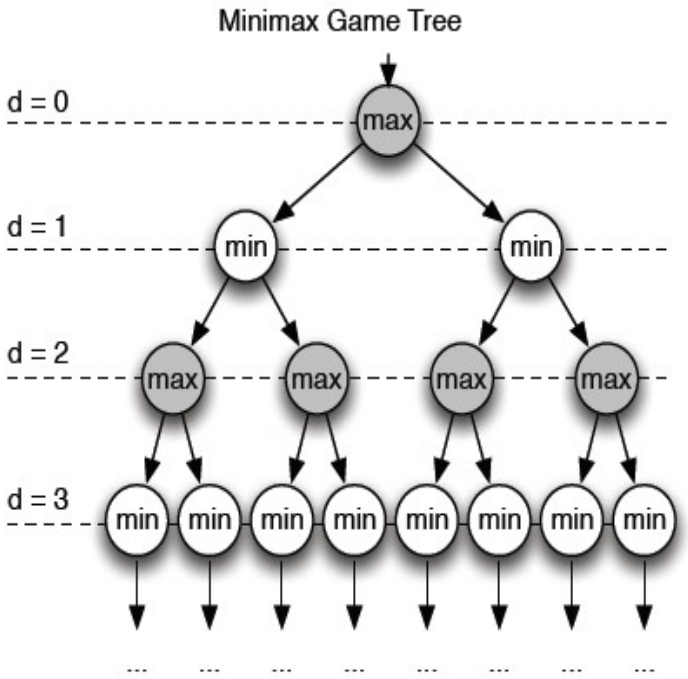
- Cómo representar un árbol de juego cuando los jugadores no juegan por turnos?
- Cómo tratar con el problema que el juego sigue avanzando mientras la IA esta gastando tiempo en la búsqueda del árbol?

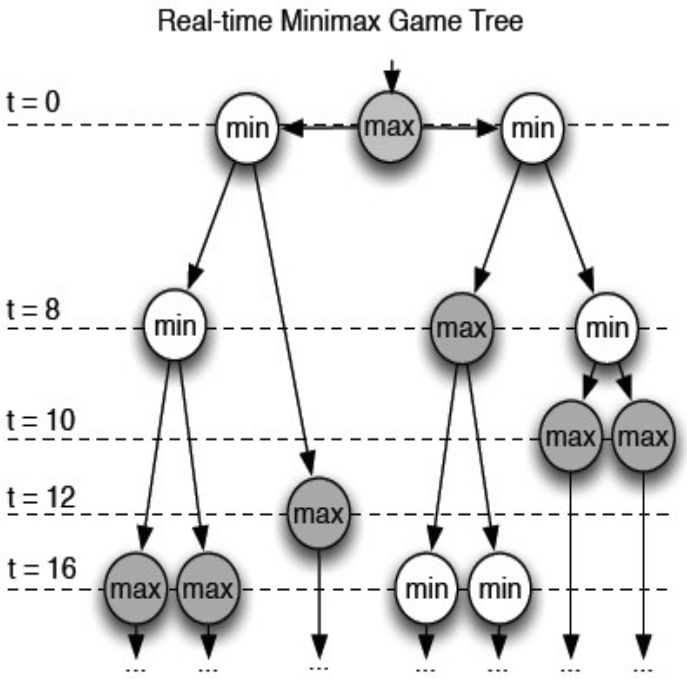
- **Orden**

  por profundidad de los nodos
- **Capas Alternas**

  Min

  Max





- **Orden**

  por tiempo de los nodos
- **Capas Min-Max**

  nodos pueden aparecer en cualquier orden
- **Tiempo**

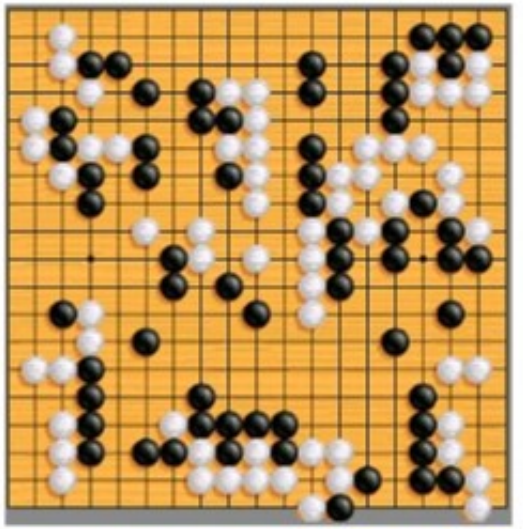
  Independiente de las acciones de los jugadores

Ontañón, S. (2012). Experiments with game tree search in real-time strategy games. *arXiv preprint arXiv:1208.1940*.

Juegos Complejos – MCTS
Árboles de Montecarlo

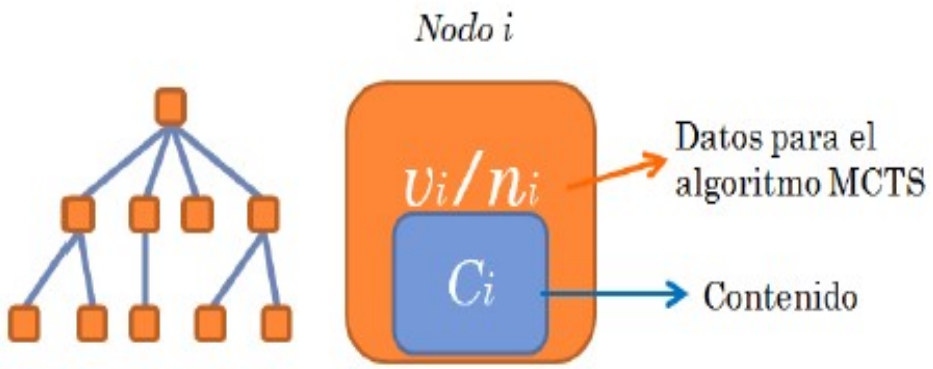
Árbol de Gran Tamaño – Algoritmo MCTS

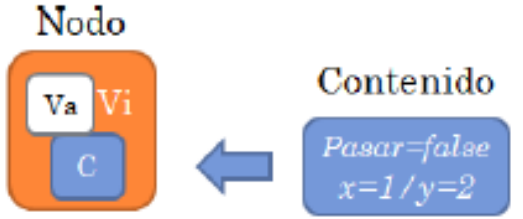
- Cómo buscar en un árbol de gran profundidad, con alto grado de ramificación y alta aleatoriedad?
  - Ej. Juego Go



Estructura del Árbol de Montecarlo

- **Nodo**
  - **Vi** → Valor actual de la posición
  - **Ni** → Contador de visitas
  - **Ci** → Contenido asociado al problema



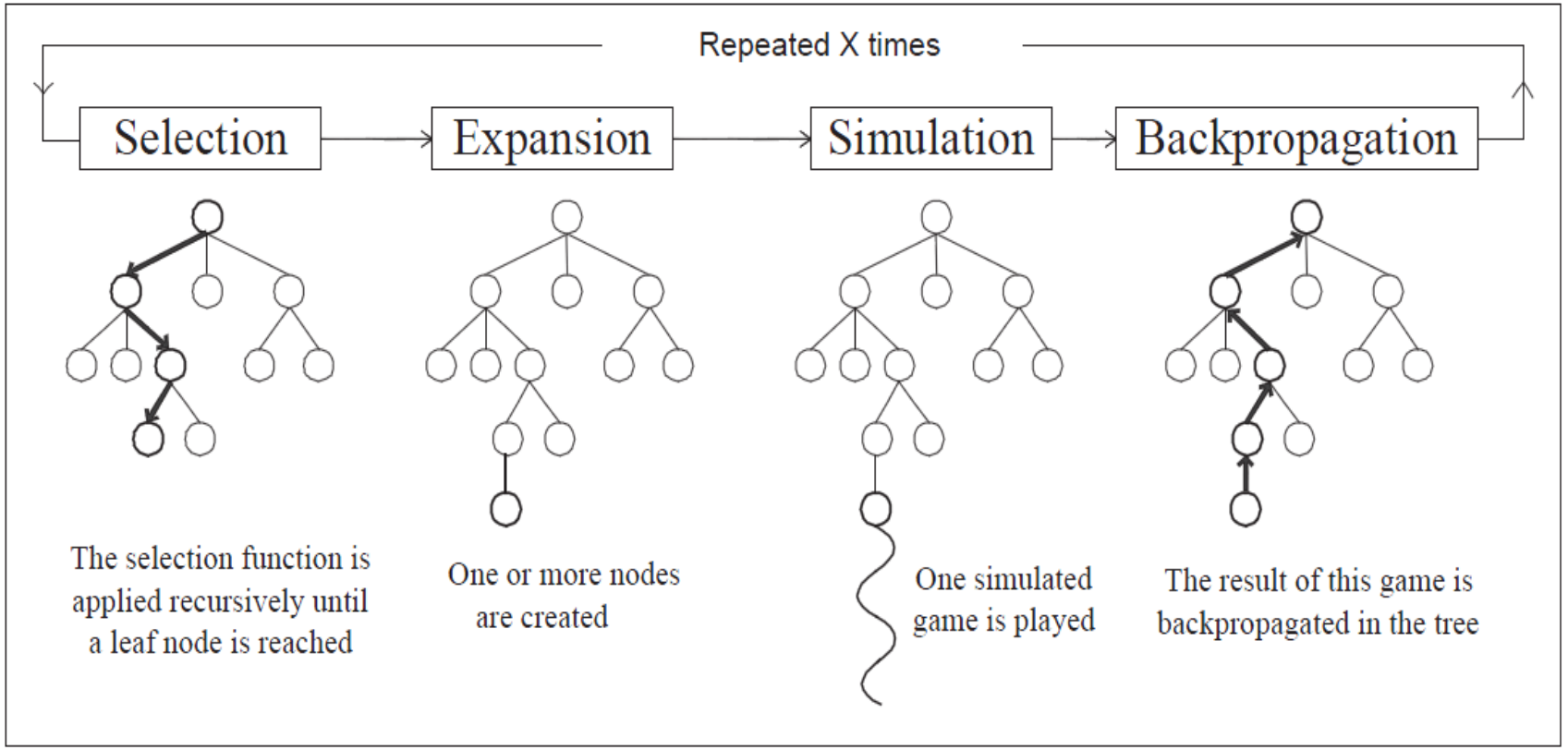


Contenido Asociado al Juego GO

Juegos Complejos – MCTS
Árboles de Montecarlo

Algoritmo MTCS

- Avanzar por una rama (Vi/Ni) hasta una hoja y realizar expansión.
- Simular movimientos aleatorios y propagar MIN-MAX hacia atrás.



Taller - Juegos

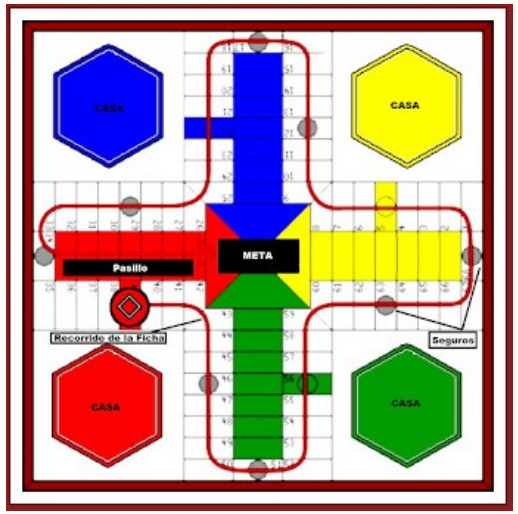
Pontificia Universidad JAVERIANA Bogotá

Juego Parchis

El Parchis es una variante simplificada del juego de Parques colombiano; la principal diferencia es que se juega con un solo dado. Las reglas de juego se pueden consultar en https://enszink.blogspot.com/2019/03/luisa-parchis.html.

Escriba un programa (en lenguaje C, C++ o Phyton) que implemente el algoritmo MIN-MAX para el juego de Parchis.

- Analizar y definir cómo representar un estado.
- Definir una función heurística para valorar un estado.
- Definir una función que genera los posibles sucesores de un estado.
- Desarrollar dos niveles del árbol para el análisis:
    - Aplicar el algoritmo MIN-MAX.



BONUS (para la próxima Clase)

- Algoritmo MIN-MAX que juega con “dos dados de tres caras”. Se debe tener en cuenta que la probabilidad del valor total de los dados no es igual para todos los cinco valores posibles.

Bibliografía

- Rusell N., Inteligencia Artificial: Un Enfoque Moderno, Prentice Hall, 2020.
- Santiago Ontañon, Experiments with Game Tree Search in Real-Time Strategy Games, Drexel University.
- Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michèle Sebag, David Silver, Csaba Szepesvári, and Olivier Teytaud, The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions, communications of the ACM, 2012.
- Guillaume Chaslot, Sander Bakkes, Istvan Szita and Pieter Spronck, Monte-Carlo Tree Search: A New Framework for Game AI, 2008.
- Beatriz Nasarre Embid, Metodo de Monte-Carlo Tree Search (MCTS)para resolver problemas de alta complejidad: Jugador virtual para el juego del Go, Universidad de Zaragoza, 2012.
- E. Rich. Inteligencia Artificial. 1994.

Pontificia Universidad
JAVERIANA
Bogotá

Inteligencia Artificial

Solución de Problemas

Juegos MultiJugador

Ing. Andrea Rueda, PhD – rueda-andrea@javeriana.edu.co
Ing. Enrique González, PhD – egonzal@javeriana.edu.co

Departamento de Ingeniería de Sistemas