INTELIGÊNCIA ARTIFICIAL

Parte 8

Busca Competitiva

Prof. Me. Celso Gallão - 2012

1 – Busca Competitiva

1.1 – Jogos:

- **Teoria dos Jogos** é um ramo da matemática aplicada que estuda estratégias envolvendo sistemas multiagentes, os quais escolhem diferentes ações na tentativa de melhorar o resultado.
- Um agente considera as ações de outros agentes e o modo como essas ações afetam seu próprio bem estar.

1 – Busca Competitiva

1.1 – Jogos:

- Ocupam as faculdades intelectuais dos seres humanos desde que surgiu a civilização.
- Jogos são fáceis de representar, pois os agentes se restrigem às regras e ações precisas.
- Em IA, a natureza abstrata dos jogos os tornam atraentes.
  - Por volta de 1950: Konrad Zuse, Claude Shannon, Norbert Wierner e Alan Turing.

1 – Busca Competitiva

1.1 – Jogos:

- **Jogo de Xadrez:** um jogo **competitivo** de **soma zero**, em que um jogador ganha e o outro perde, ou dá empate.
- Valores de utilidade, ao final do jogo, de **soma zero**:
  - *Exemplo: No Xadrez, quem vence (+1), quem perde (-1), e o empate (0).*
- Fator de ramificação médio do Xadrez é 35, e em média 50 movimentos para cada jogador:
  - *Árvore de busca, aproximadamente 35^{100} = 1E+154 nós.*

1 – Busca Competitiva


Prisioneiro A


Prisioneiro B

1.1 – Jogos:

- **O Dilema do Prisioneiro:** um jogo **cooperativo** de **soma não-nula**, em que os dois jogadores podem, juntos, ganhar ou perder, conforme as opções fornecidas:
  - Dois suspeitos, **A** e **B**, são presos pela polícia. Os policiais têm provas insuficientes para os condenar. **A** e **B** estão separados na delegacia e os oficiais propõem o mesmo acordo a ambos.
  - Se **A** dedurar **B**, e **B** não dedurar **A**, então **A** fica **2 ano preso** e **B** pega **10 anos de cadeia**. (e vice-versa)
  - Se ambos ficarem em silêncio, ambos estão **livres**.
  - Se ambos dedurarem um ao outro, cada um ficará **5 anos preso**.

1 – Busca Competitiva


Prisioneiro A


Prisioneiro B

1.1 – Jogos:

- **O Dilema do Prisioneiro:** um jogo **cooperativo** de **soma não-nula**, em que os dois jogadores podem, juntos, ganhar ou perder, conforme as opções fornecidas:

www.oopa.com.br

A	B	CONSEQUÊNCIAS
FALA	NÃO FALA	2 ANOS A 10 ANOS B
NÃO FALA	FALA	10 ANOS A 2 ANOS B
FALA	FALA	5 ANOS
NÃO FALA	NÃO FALA	LIVRES

1 – Busca Competitiva

1.1 – Jogos:

- Jogos podem ser muito difíceis de resolver, justamente pela tamanho da árvore de busca.
- A busca de soluções para jogos fez surgir interessantes técnicas, para tomar boas decisões em tempo aceitável.

1 – Busca Competitiva

1.2 – Decisões Ótimas em Jogos:

- Jogos com 2 jogadores:
  - **MAX** faz o primeiro movimento, buscando maximizar o seu resultado e minimizar o resultado de seu oponente.
  - **MIN** faz o movimento seguinte, buscando o inverso de MAX.

1 – Busca Competitiva

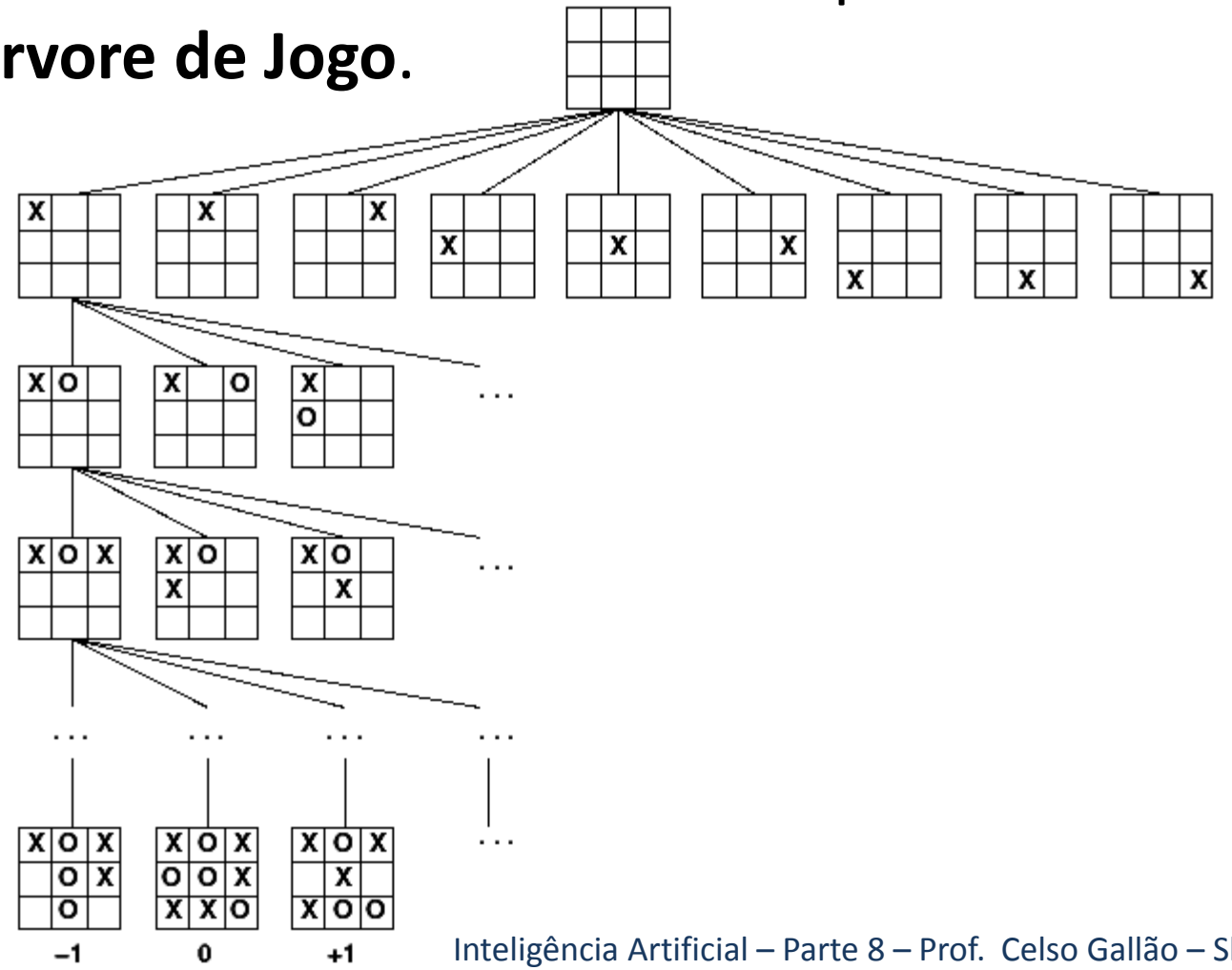
1.2 – Decisões Ótimas em Jogos:

- Um jogo é definido como um problema de busca com os seguintes componentes:
  - **Estado Inicial**: indica a posição e o jogador MAX.
  - **Função Sucessor**: lista de pares (*movimento, estado*).
  - **Teste de Término**: determina quando o jogo termina (estado terminal).
  - **Função Utilidade** (objetivo ou compensação): valor numérico aos estados terminais.

1 – Busca Competitiva

1.2 – Decisões Ótimas em Jogos:

- O estado inicial e os movimentos válidos para cada lado definem a **Árvore de Jogo**.



Árvore de busca parcial para o jogo-da-velha.

1 – Busca Competitiva

1.2 – Decisões Ótimas em Jogos:

- Em um problema de busca trivial (sem ser jogo), a **solução ótima** é uma sequência de movimentos que leva ao melhor estado terminal possível.
- Em um jogo, **MAX** deve encontrar uma estratégia de contingência que especifique seus movimentos, mesmo após a jogada de **MIN**.
- Uma **estratégia ótima**, leva a resultados bons mesmo quando está se enfrentando oponente infalível.

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores:

- A busca MINIMAX é um procedimento de busca competitiva em profundidade.
- A ideia é iniciar na posição corrente e usar o gerador de movimentos plausíveis para gerar o conjunto de possíveis posições sucessoras.
- Depois aplicar a função de avaliação estática a essas posições para que seja atribuído um valor, que representa a qualidade de cada uma dessas posições.

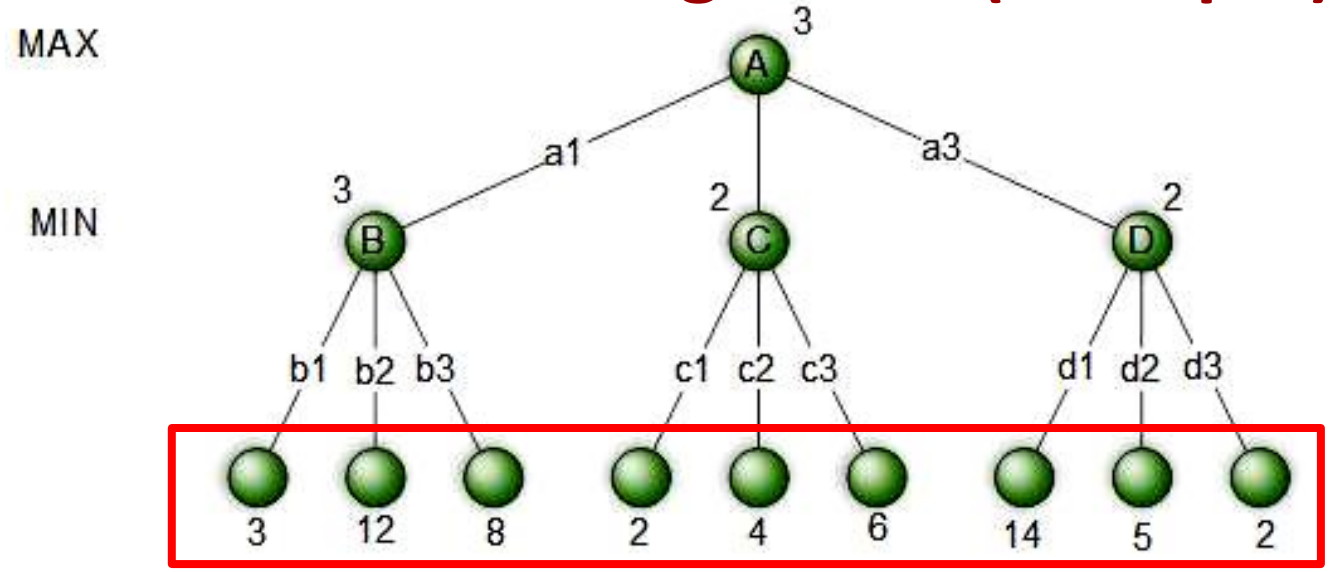
2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores:

- Então, retorna-se os valores para o estado inicial para simplesmente escolher a jogada que resultará no estado mais promissor.
- Assume-se que a função de avaliação estática retorna valores altos para indicar situações boas, portanto a meta é maximizar o valor na próxima posição do tabuleiro.

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores (exemplo):

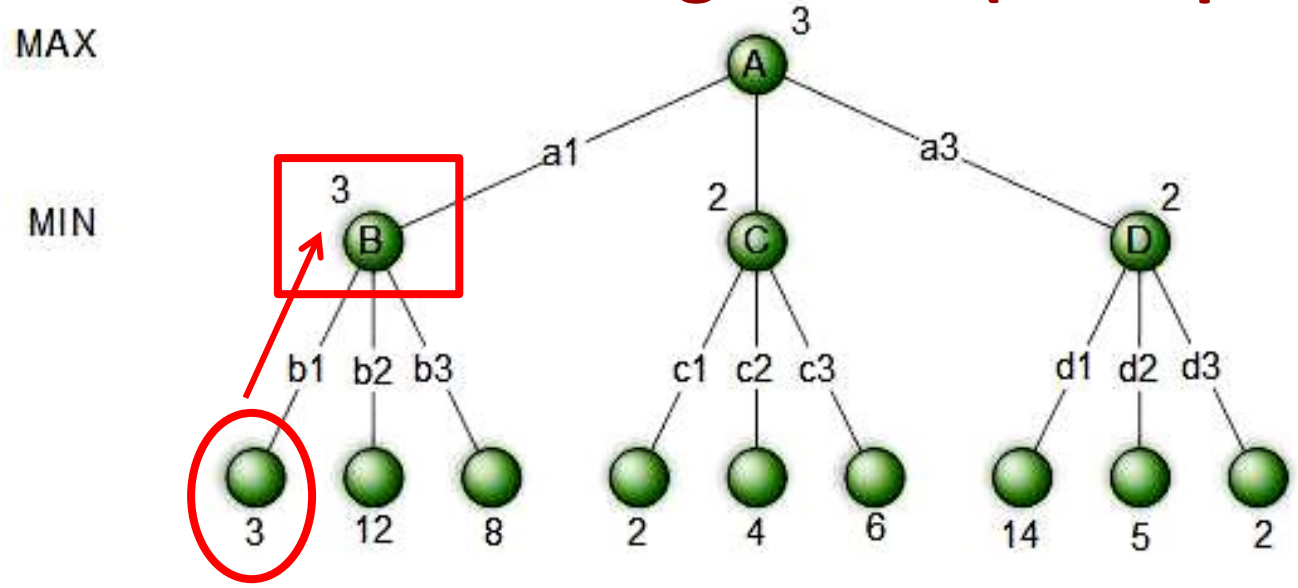


CARREGANDO

- Neste jogo, os valores da função de avaliação dos estados terminais variam de 2 até 14.

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores (exemplo):

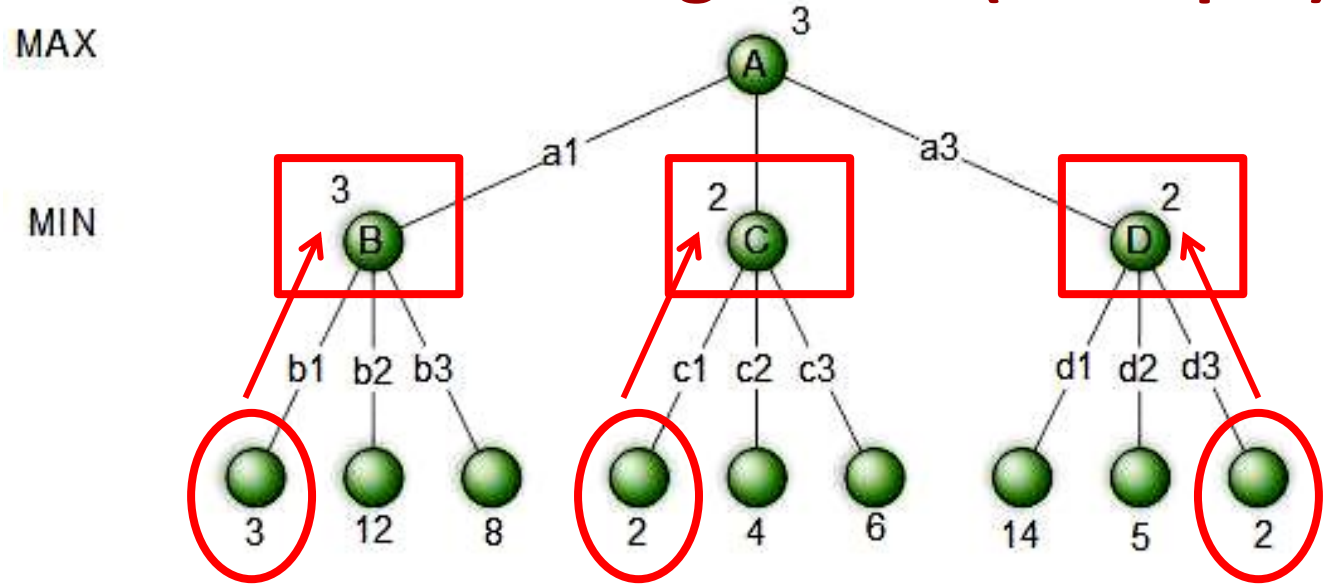


CARREGANDO

- O primeiro nó de **MIN**, identificado por **B**, tem três sucessores com valores **3**, **12** e **8**. **MIN busca sempre o menor valor**, portanto seu valor **MINIMAX = 3**.

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores (exemplo):

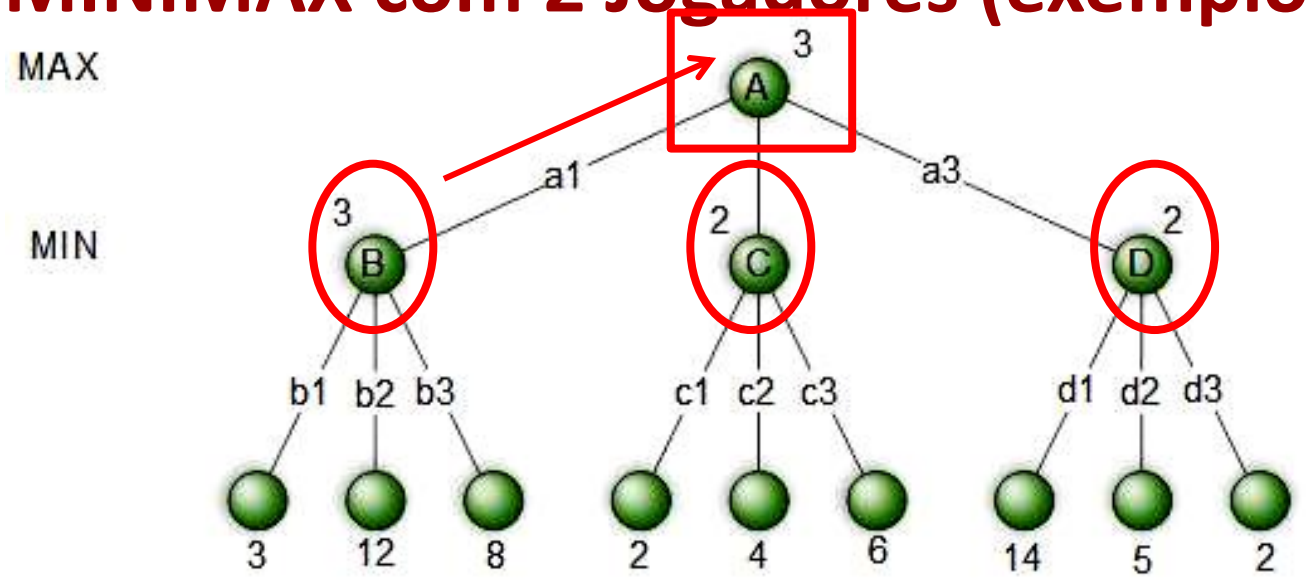


CARREGANDO

- O primeiro nó de **MIN**, identificado por **B**, tem três sucessores com valores **3**, **12** e **8**. **MIN busca sempre o menor valor**, portanto seu valor **MINIMAX = 3**.
- De modo semelhante, os outros dois nós de **MIN** têm valor **MINIMAX = 2**.

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores (exemplo):

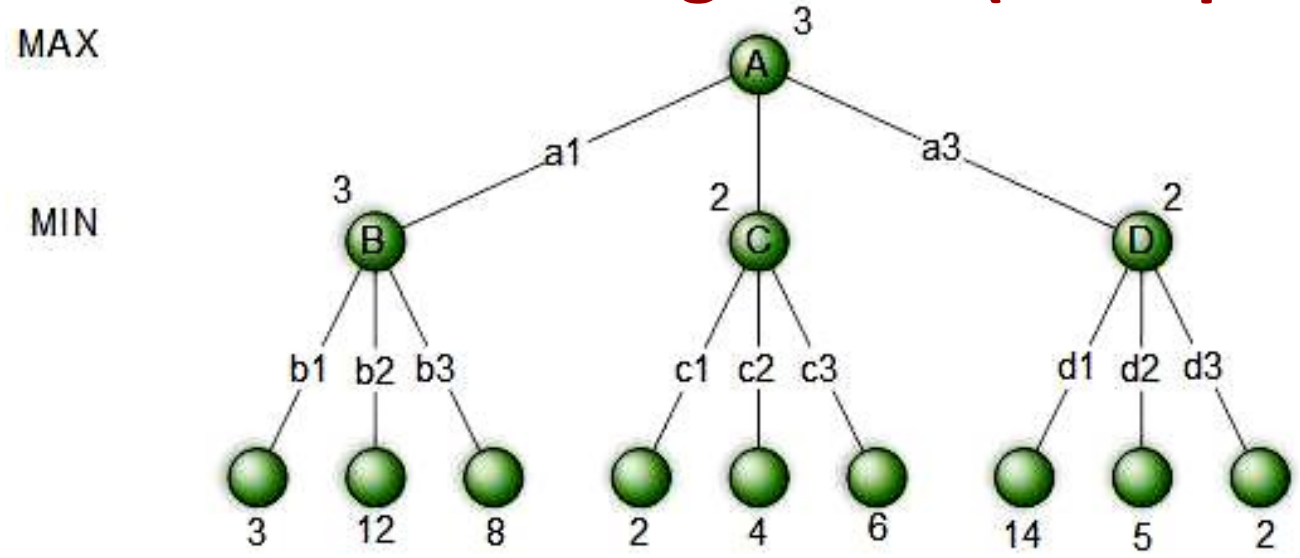


CARREGANDO

- O nó **raiz** é um nó de **MAX**, seus sucessores têm valores MINIMAX **3**, **2** e **2**, cuja origem vem da escolha MIN. **MAX busca sempre o maior valor**, portanto ele tem um valor **MINIMAX = 3**.

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores (exemplo):



CARREGANDO

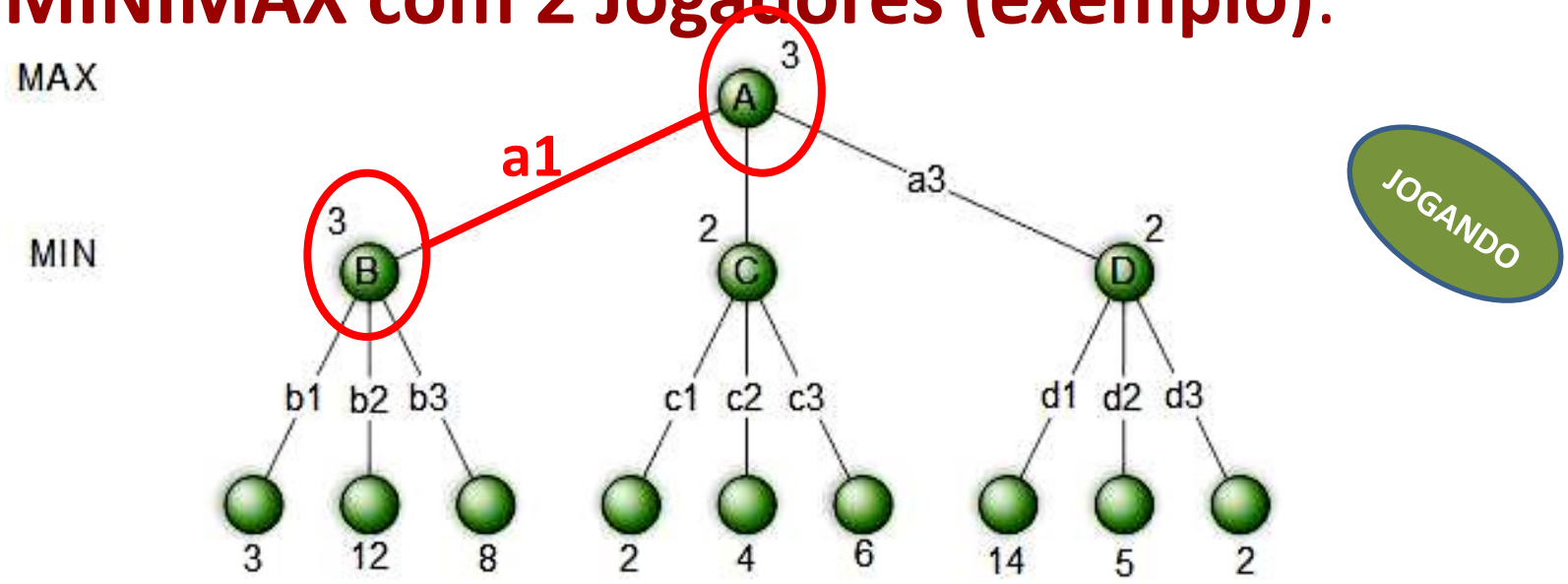
- Conhecendo todos os valores MINIMAX, pode-se calcular o valor MINIMAX, a partir da raiz:

$$\mathbf{MINIMAX(A) = max\{min(3,12,8), min(2,4,6), min(14,5,2)\}}$$

- Conhecendo todos os valores MINIMAX, pode-se montar a **Árvore de Jogo**, a partir da raiz, como temos a seguir...

2 – Busca MINIMAX

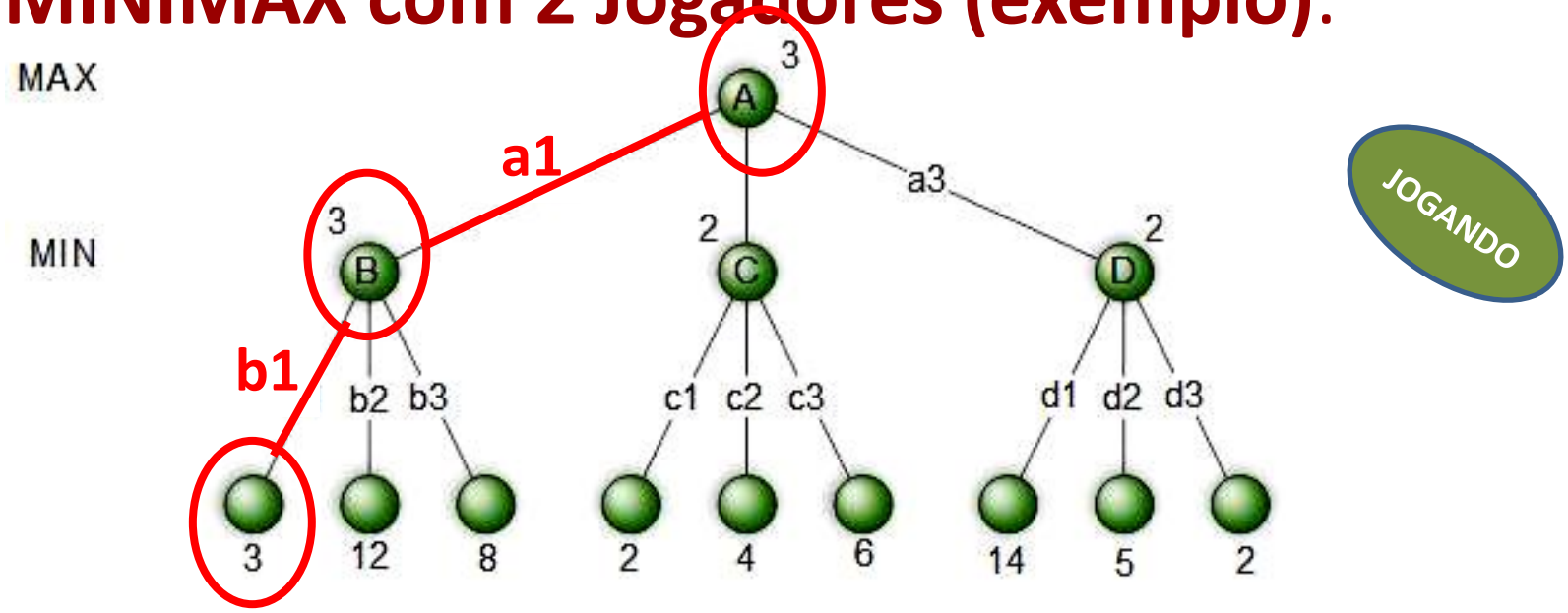
2.1 – MINIMAX com 2 Jogadores (exemplo):



- Identifica-se a decisão MINIMAX na raiz, sendo a ação **a1** a escolha ótima para MAX, porque leva ao sucessor com o mais alto valor MINIMAX.

2 – Busca MINIMAX

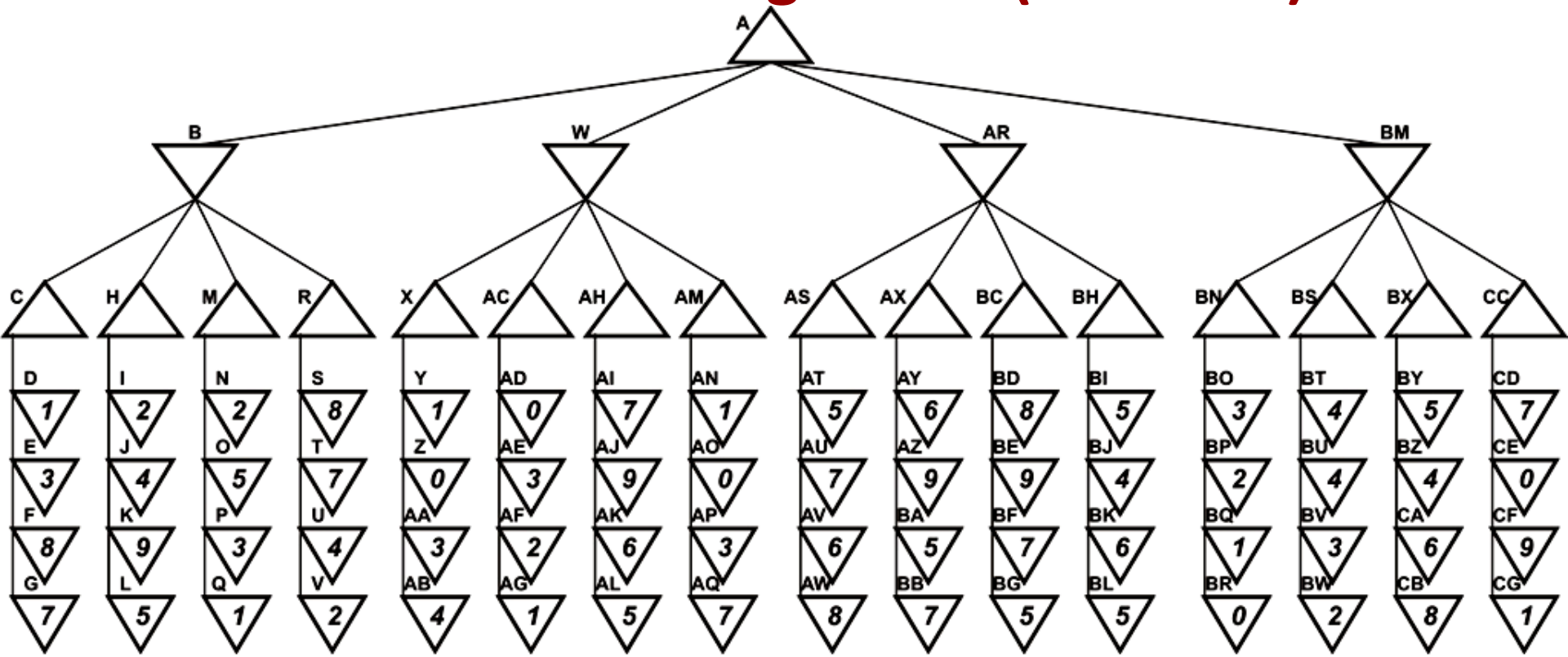
2.1 – MINIMAX com 2 Jogadores (exemplo):



- Assim pode-se identificar a decisão MINIMAX na raiz, a ação **a1** é a escolha ótima para MAX, porque leva ao sucessor com o mais alto valor MINIMAX.
- E a decisão MINIMAX em B, a ação **b1** é a escolha ótima para MIN, porque leva ao sucessor com o mais baixo valor MINIMAX.

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores (exercício):



- Considere um jogo com 2 jogadores, 4 possibilidades de escolha por jogada e 2 rodadas por jogador.
- Utilidade dos estado terminais: [0, 1,...,9]
- Para facilitar o entendimento, MAX = △ e MIN = ▽

2 – Busca MINIMAX

2.1 – MINIMAX com 2 Jogadores (exercício):

1. Calcule o valor MINIMAX para o vértice **W**.
2. Monte a **Árvore de Jogo**, a partir da raiz, considerando que ambos jogadores se utilizam do algoritmo MINIMAX para sua tomada de decisão.
3. Considerando a **Árvore de Jogo** da questão anterior, e que MAX vence com utilidade de [5,...,9] e perde para as demais utilidades, quem vence este jogo?

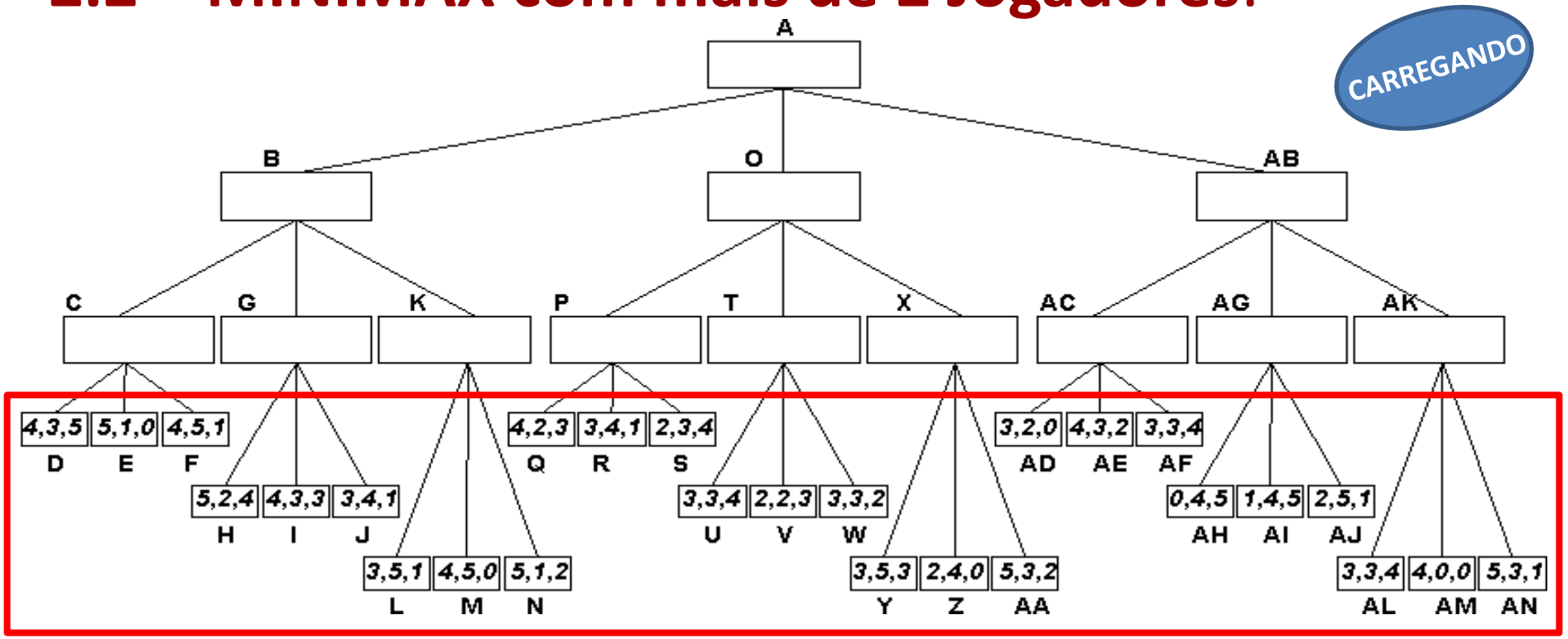
2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

- A ideia é substituir o único valor para cada nó por um vetor de valores, por exemplo:
  - *Em um jogo com 3 jogadores **J1**, **J2** e **J3**, um vetor **(v_{J1}, v_{J2}, v_{J3})** está associado à cada nó.*
- Nos estado terminais, este vetor fornece a utilidade do estado sob o ponto de vista de cada jogador.
- Quando o algoritmo MINIMAX faz a propagação dos valores dos estados terminais até a raiz, ele propaga o maior valor para o "jogador da vez" e o pior valor para seus concorrentes, respeitando os valores de cada jogador expressos no vetor de utilidade.

2 – Busca MINIMAX

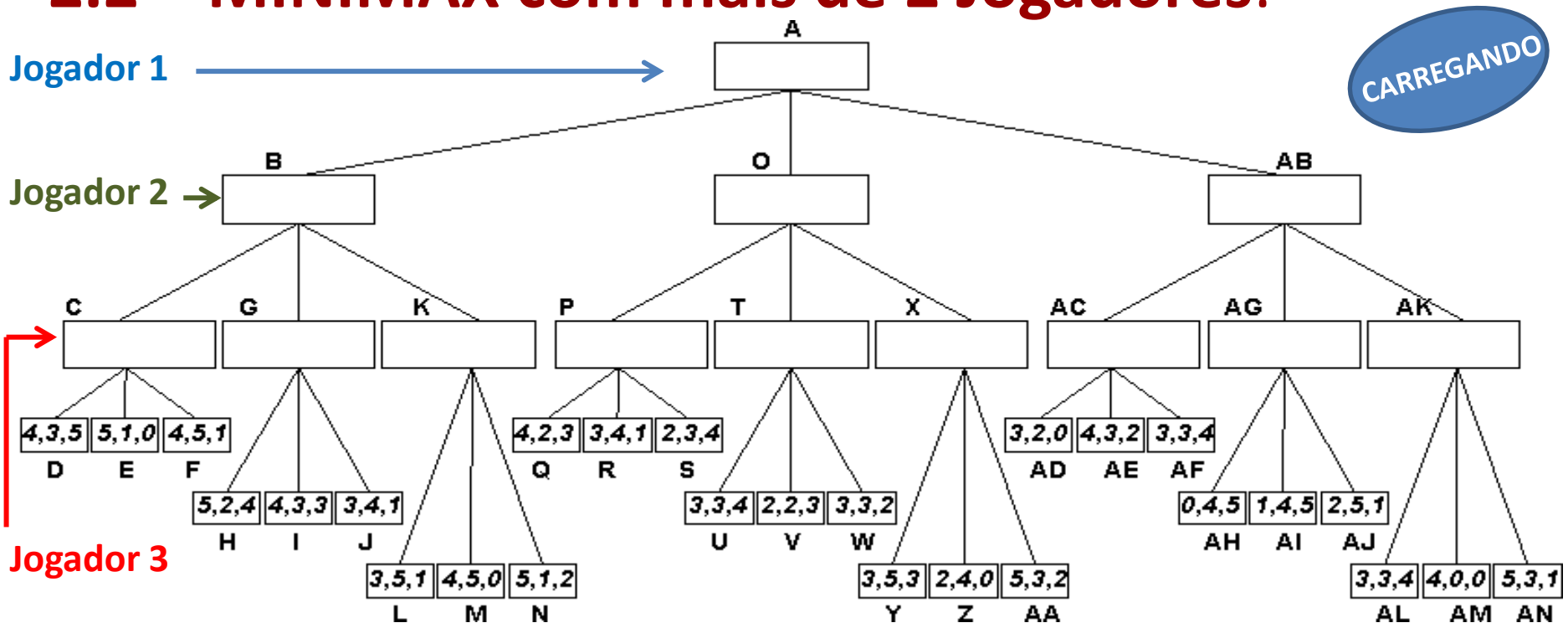
2.2 – MINIMAX com mais de 2 Jogadores:



- Neste jogo, os valores da função de avaliação dos estados terminais variam de 0 até 5.

2 – Busca MINIMAX

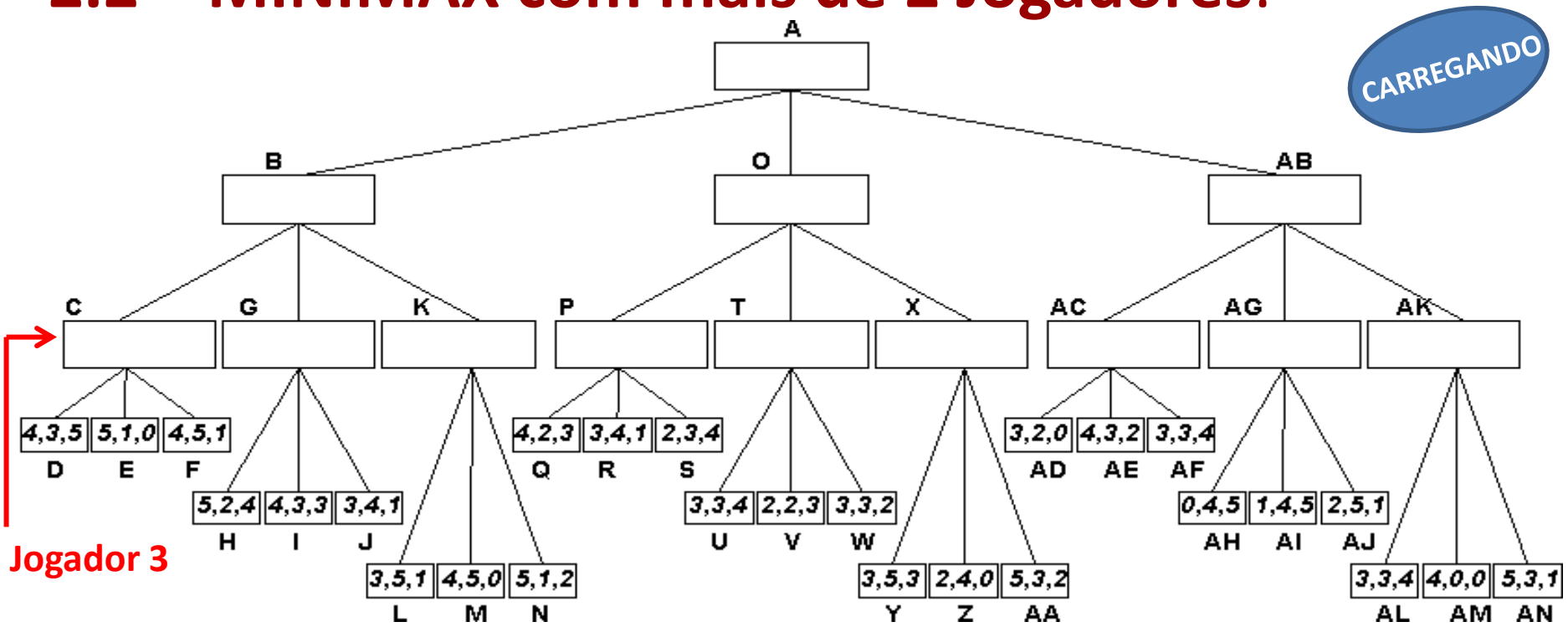
2.2 – MINIMAX com mais de 2 Jogadores:



- Neste jogo, há 3 jogadores.

2 – Busca MINIMAX

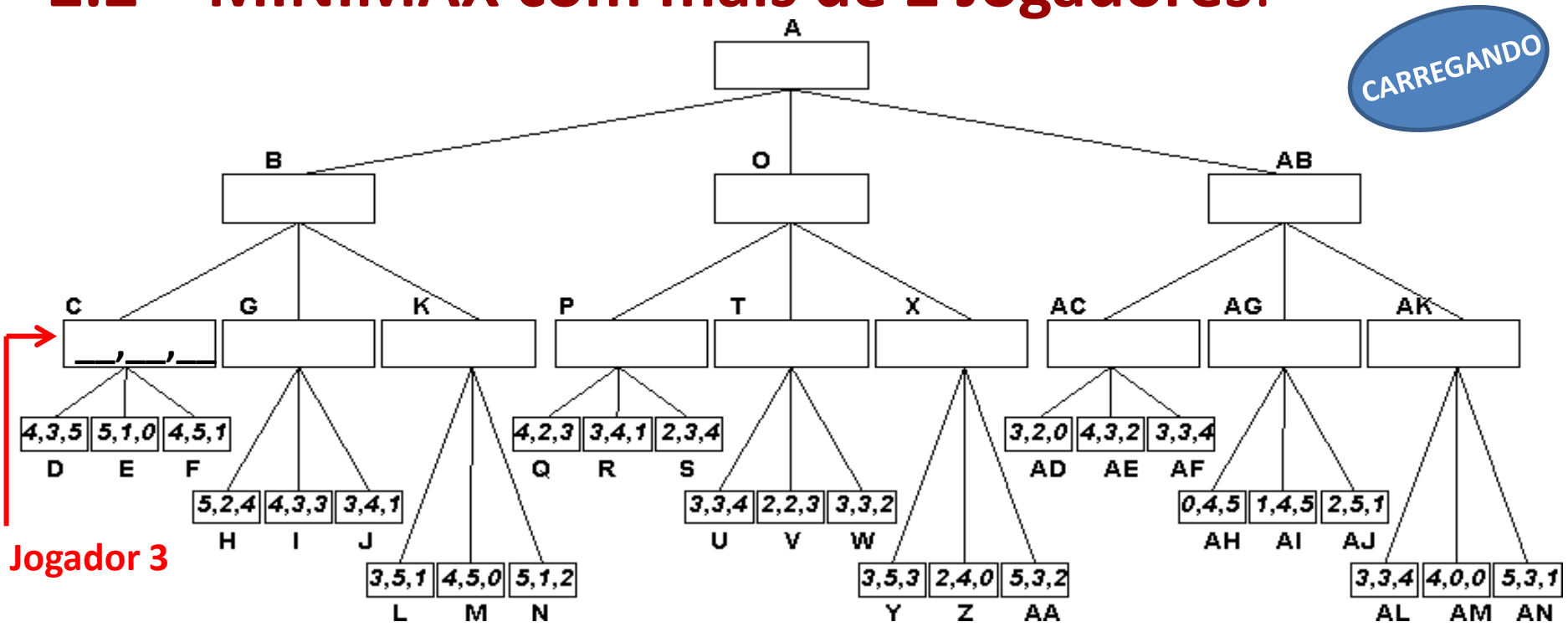
2.2 – MINIMAX com mais de 2 Jogadores:



- Propagando inicialmente com o Jogador 3.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***C***, encontra-se os vetores:

  D$(v_{J1}=4, v_{J2}=3 \text{ e } v_{J3}=5)$;

  E$(v_{J1}=5, v_{J2}=1 \text{ e } v_{J3}=0)$;

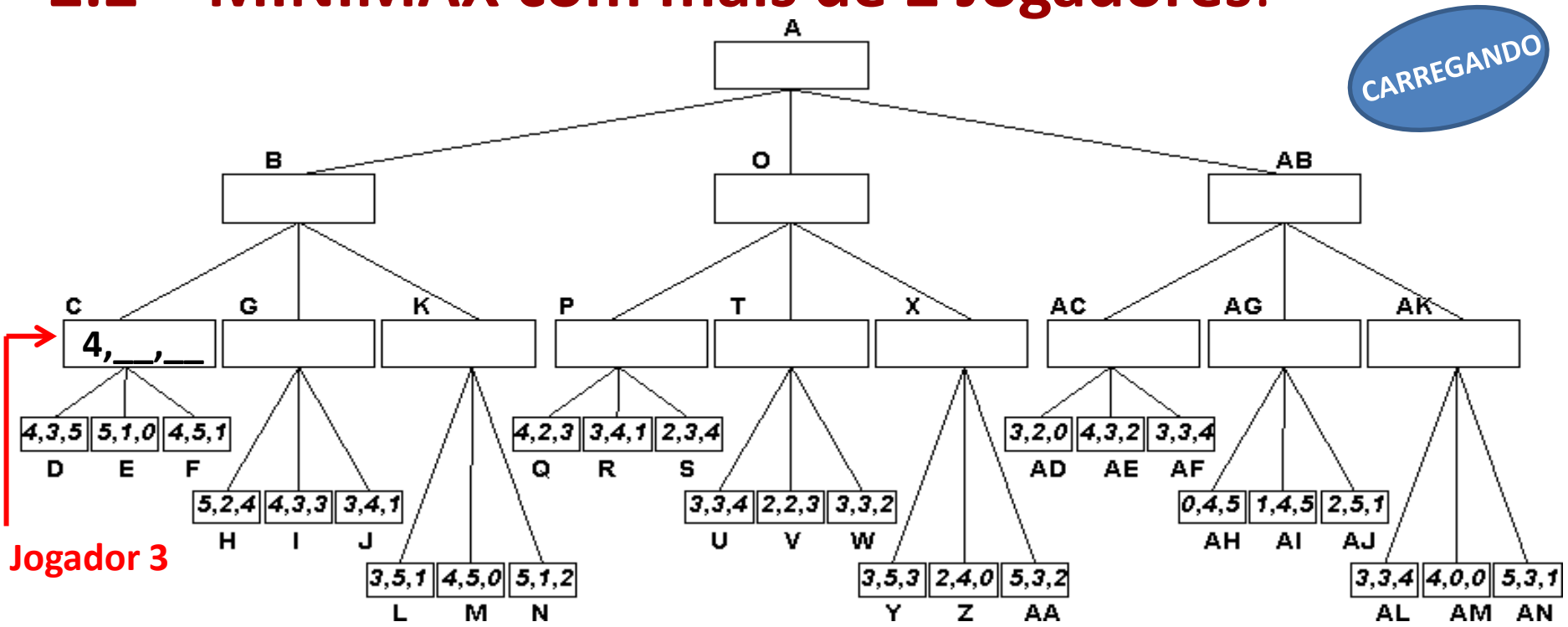
  F$(v_{J1}=4, v_{J2}=5 \text{ e } v_{J3}=1)$;

  PIOR

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***C***, encontra-se os vetores:

  D$(v_{J1}=4, v_{J2}=3 \text{ e } v_{J3}=5)$;

  E$(v_{J1}=5, v_{J2}=1 \text{ e } v_{J3}=0)$;

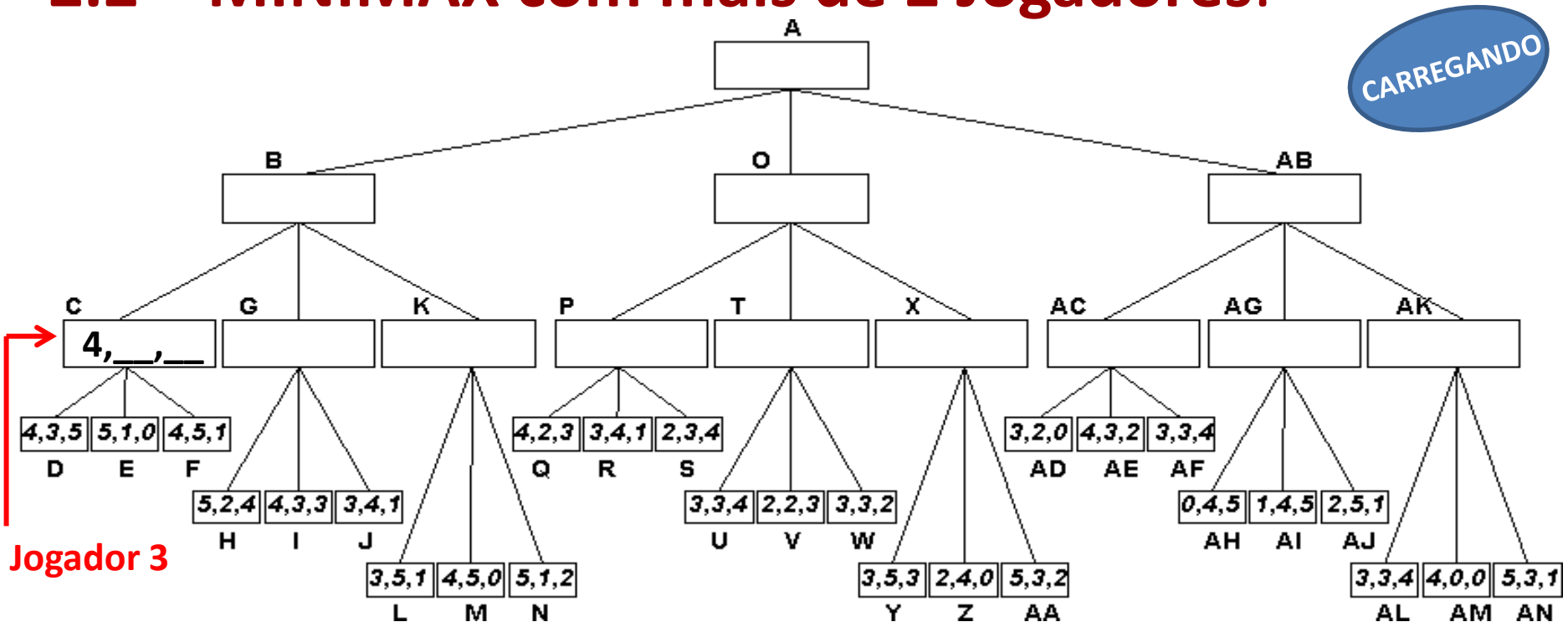
  F$(v_{J1}=4, v_{J2}=5 \text{ e } v_{J3}=1)$;

  PIOR

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***C***, encontra-se os vetores:

  D(v_{J1}=4, v_{J2}=3 e v_{J3}=5);

  E(v_{J1}=5, v_{J2}=1 e v_{J3}=0);

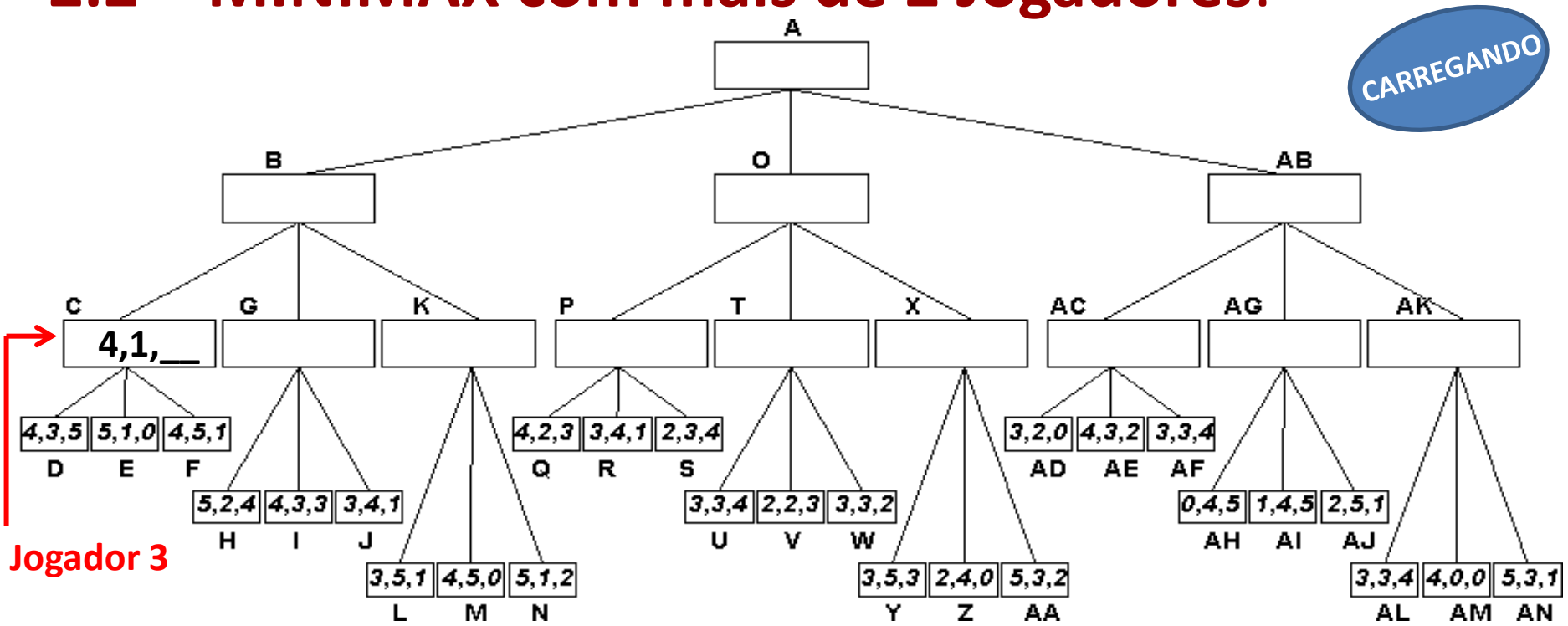
  F(v_{J1}=4, v_{J2}=5 e v_{J3}=1);

  PIOR

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***C***, encontra-se os vetores:

  D(v_{J1}=4, v_{J2}=3 e v_{J3}=5);

  E(v_{J1}=5, v_{J2}=1 e v_{J3}=0);

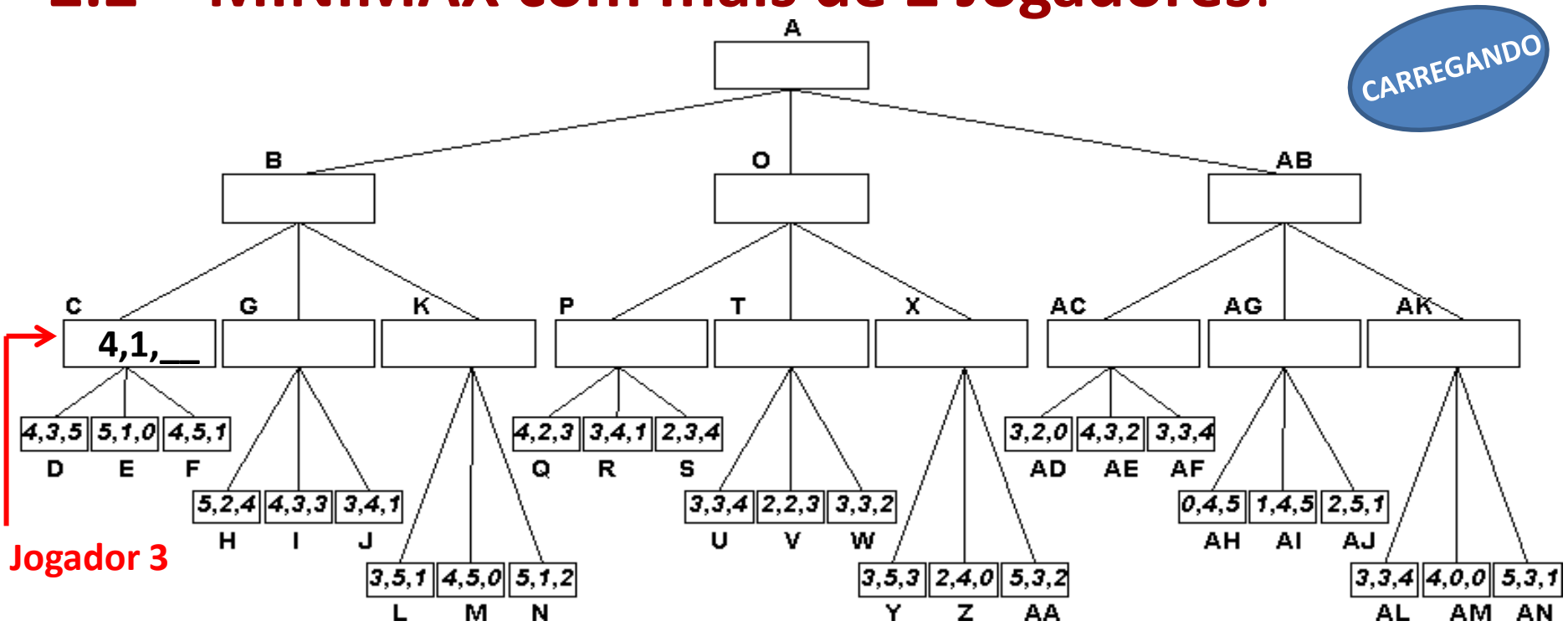
  F(v_{J1}=4, v_{J2}=5 e v_{J3}=1);

  PIOR

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***C***, encontra-se os vetores:

  D(v_{J1}=4, v_{J2}=3 e v_{J3}=5);

  E(v_{J1}=5, v_{J2}=1 e v_{J3}=0);

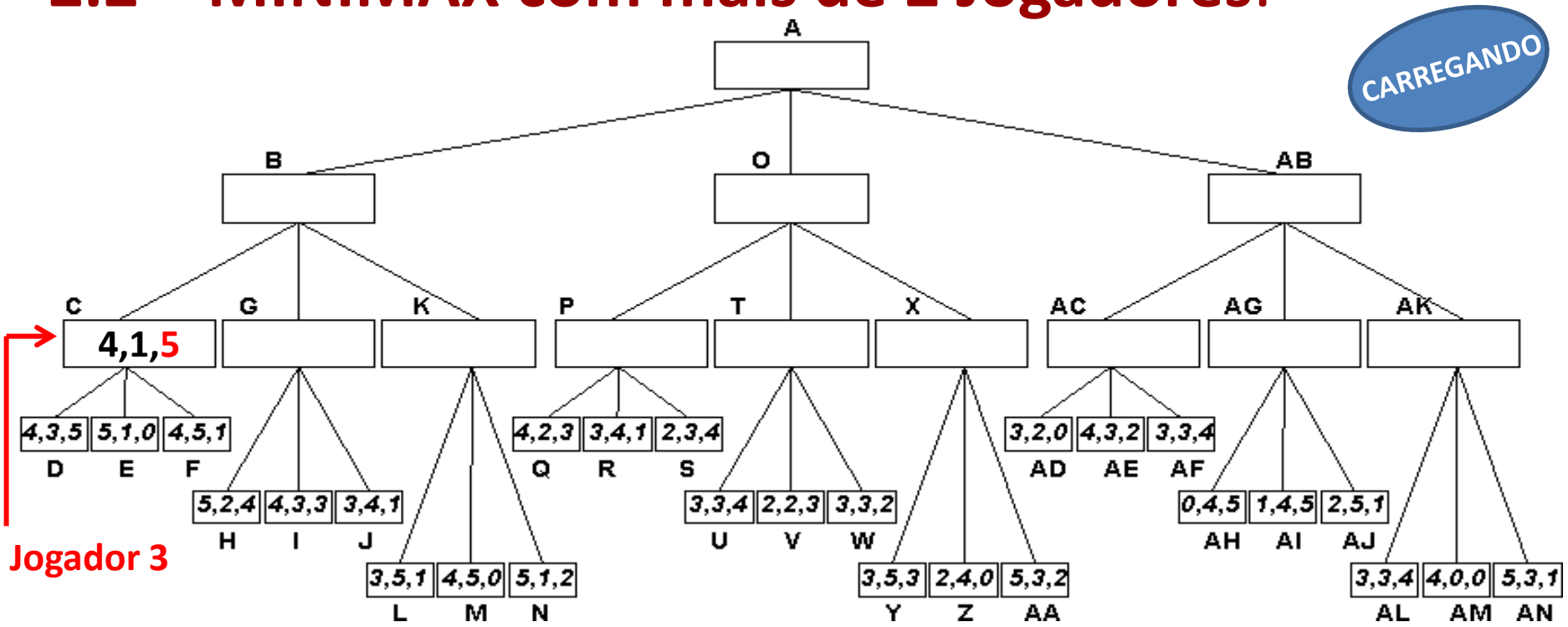
  F(v_{J1}=4, v_{J2}=5 e v_{J3}=1);

  MELHOR

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***C***, encontra-se os vetores:

  D(v_{J1}=4, v_{J2}=3 e v_{J3}=5);

  E(v_{J1}=5, v_{J2}=1 e v_{J3}=0);

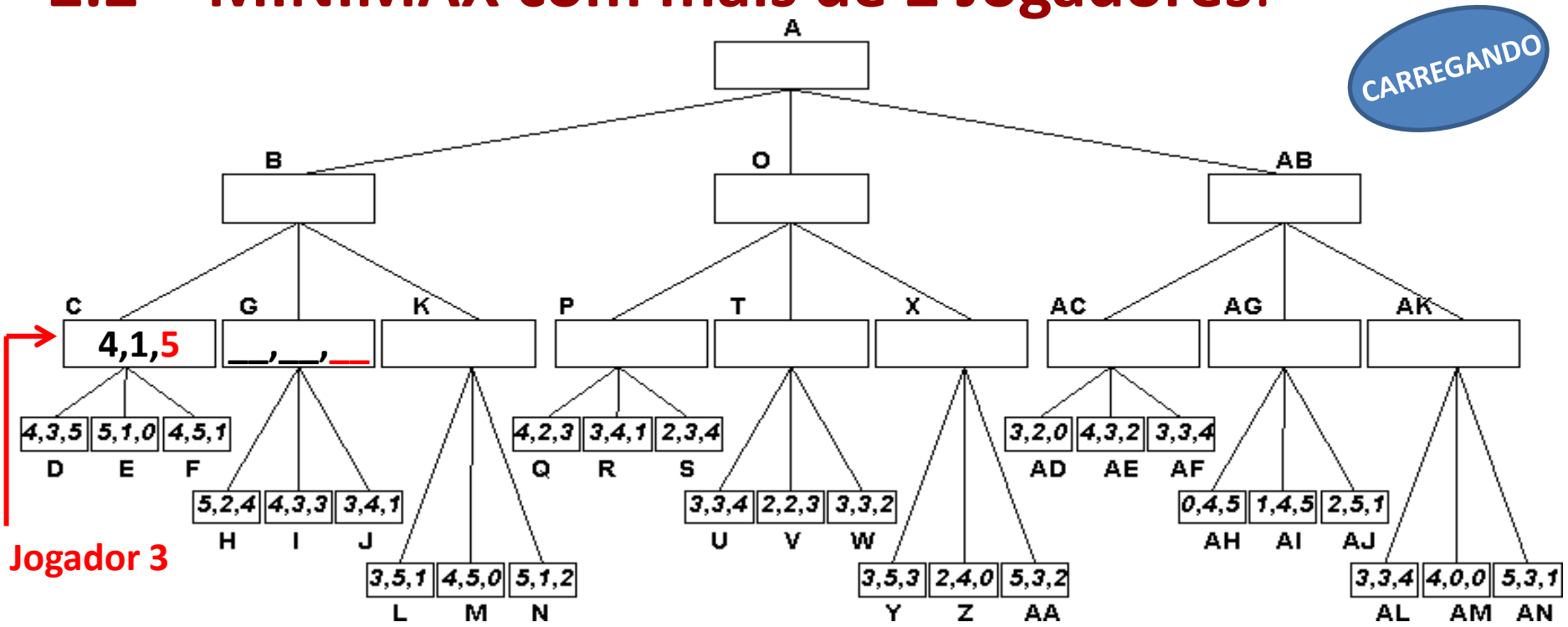
  F(v_{J1}=4, v_{J2}=5 e v_{J3}=1);

  MELHOR

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

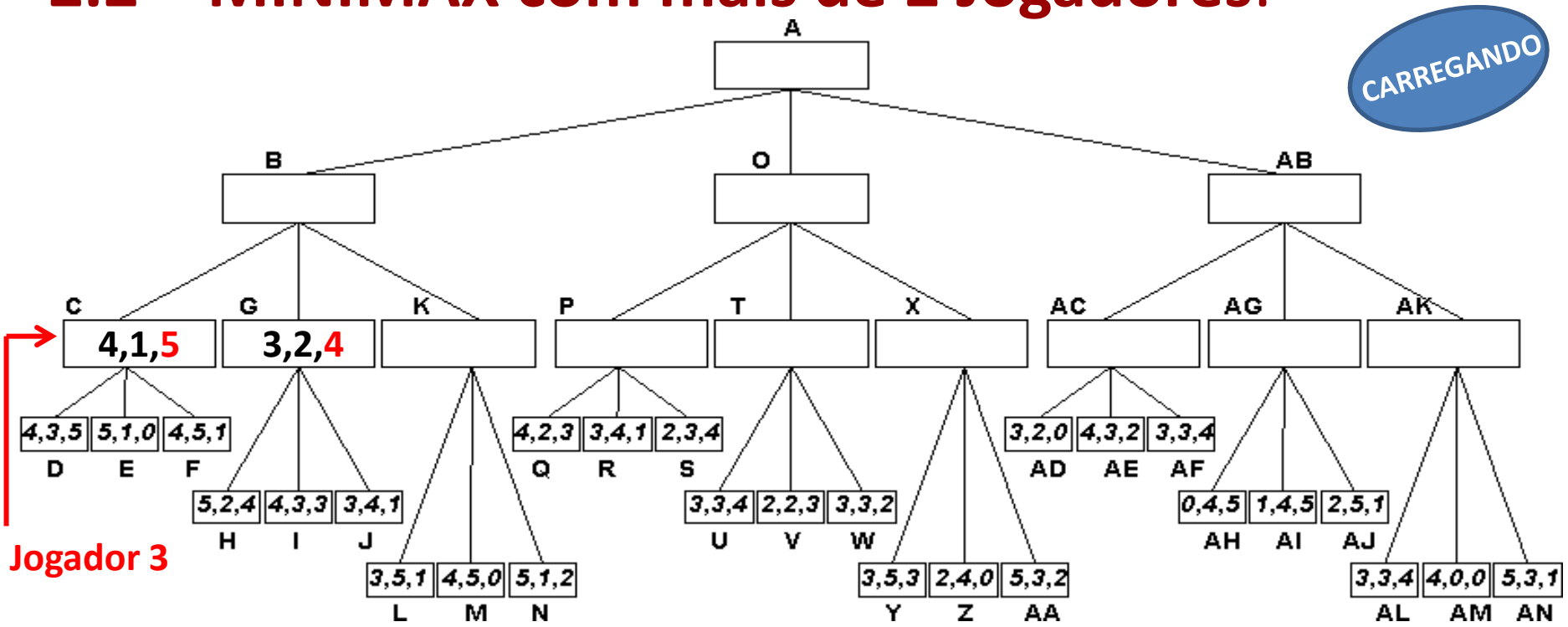


- Analisando o nó ***G***, propaga-se...

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

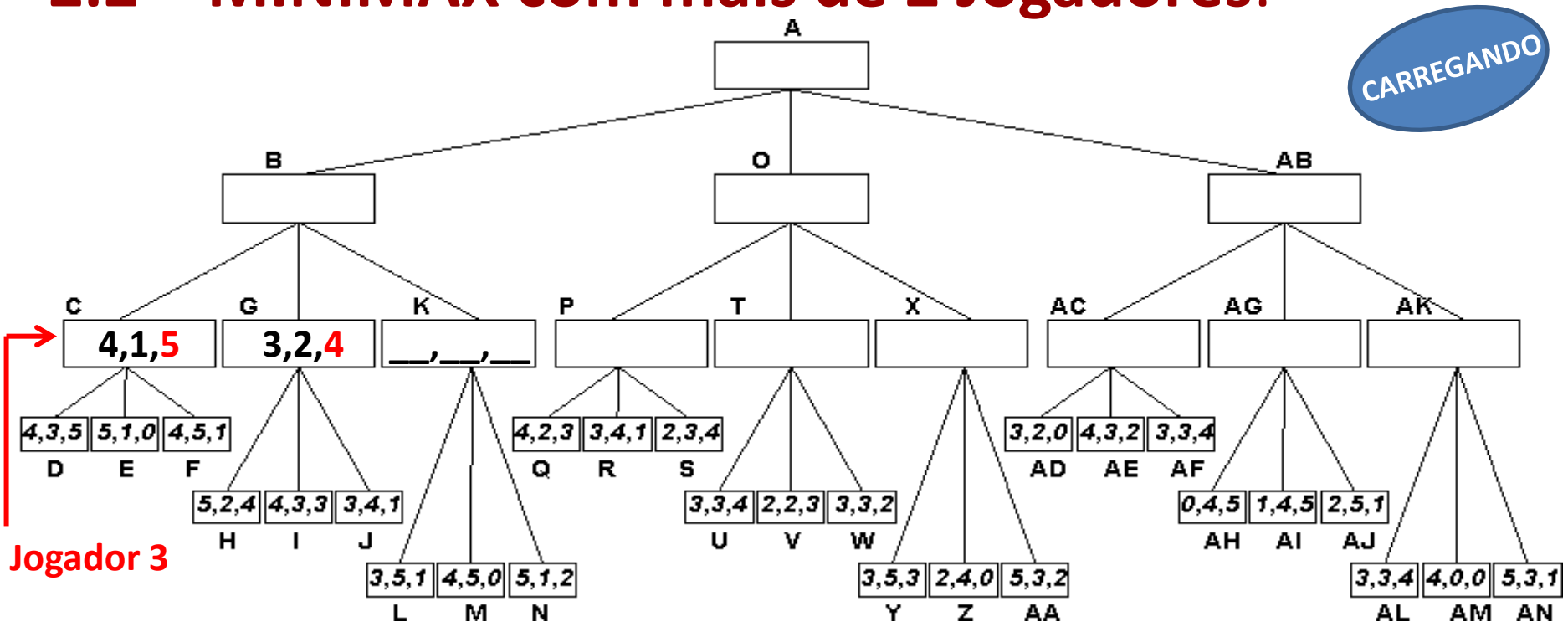


- Analisando o nó ***G***, propaga-se (3,2,4)

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

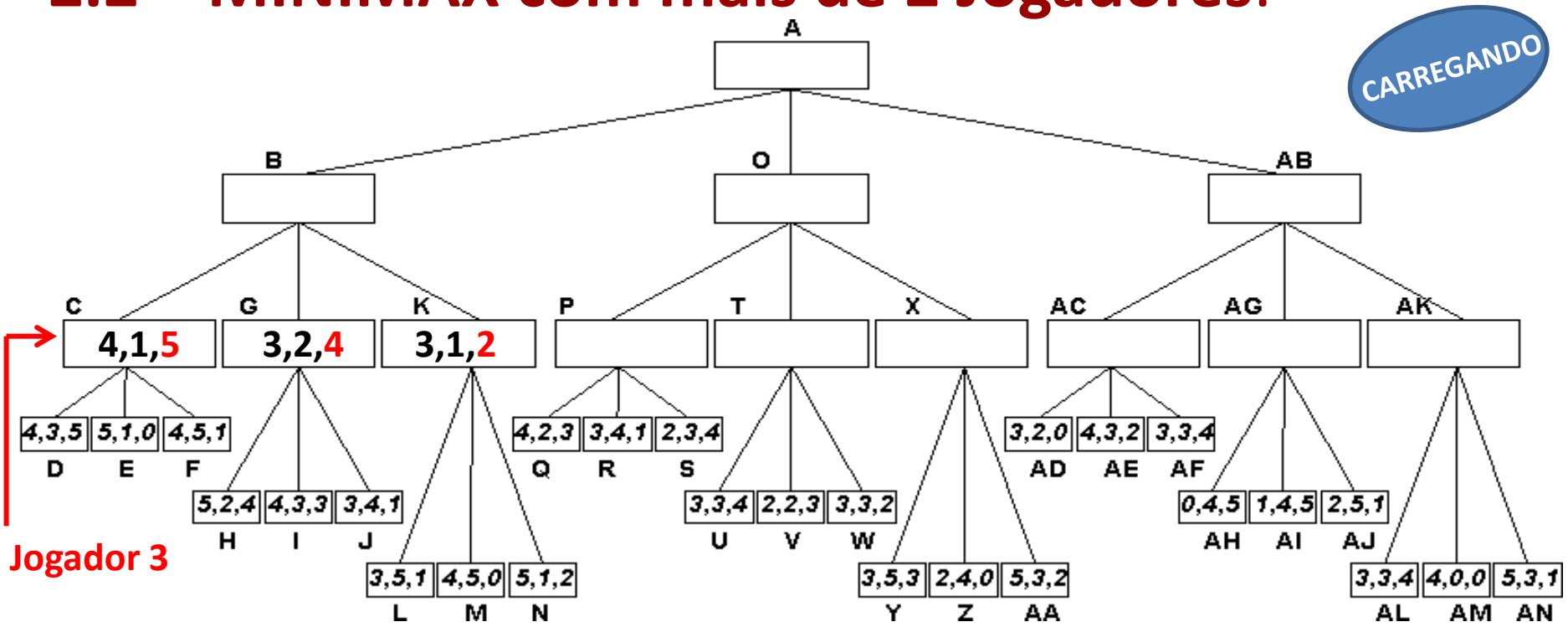


- Analisando o nó ***K***, propaga-se...

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

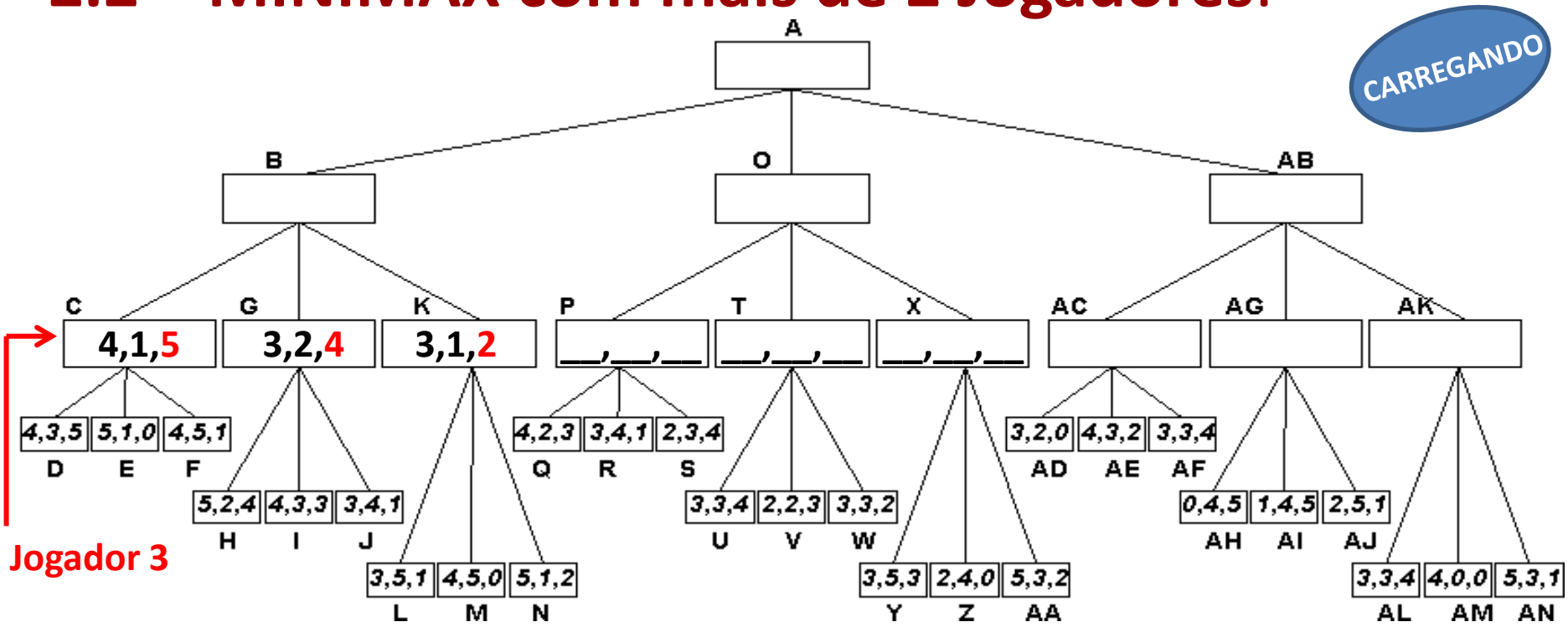


- Analisando o nó ***K***, propaga-se (3,1,2)

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

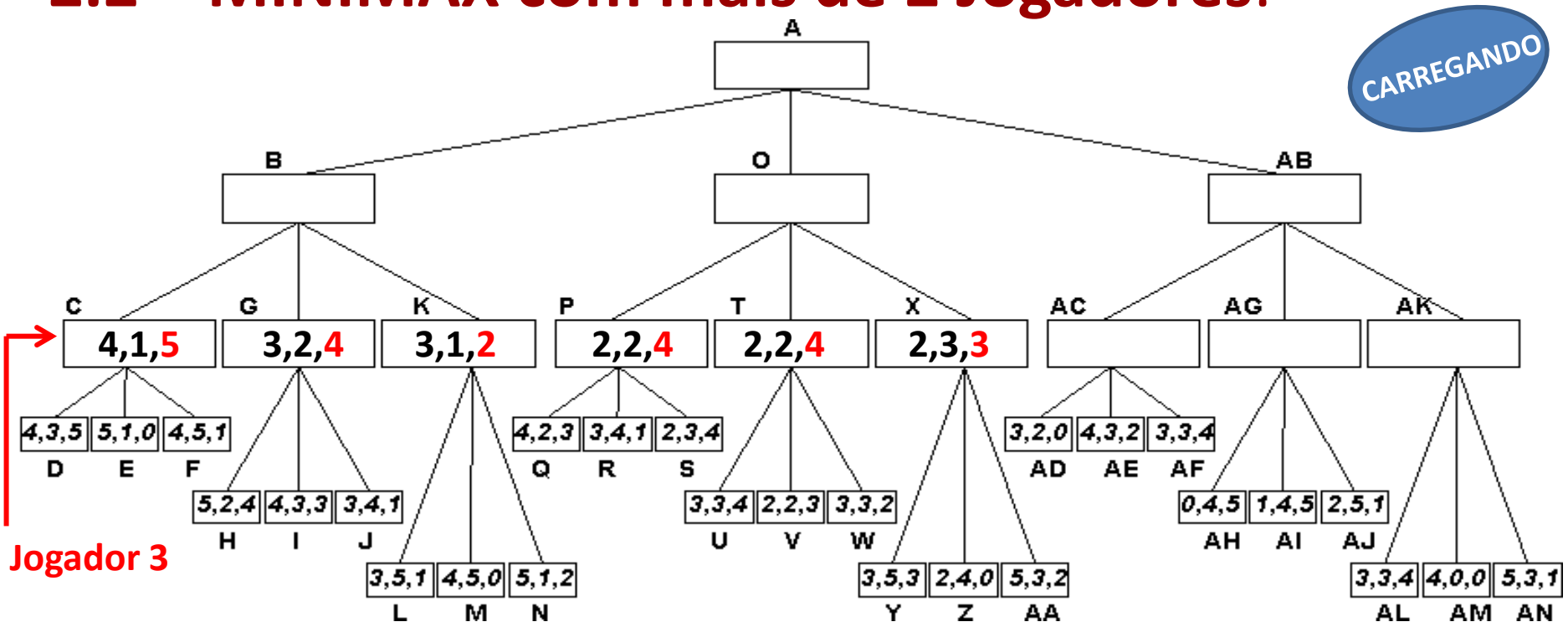


- Analisando os nós ***P***, ***T*** e ***X***, propaga-se...

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

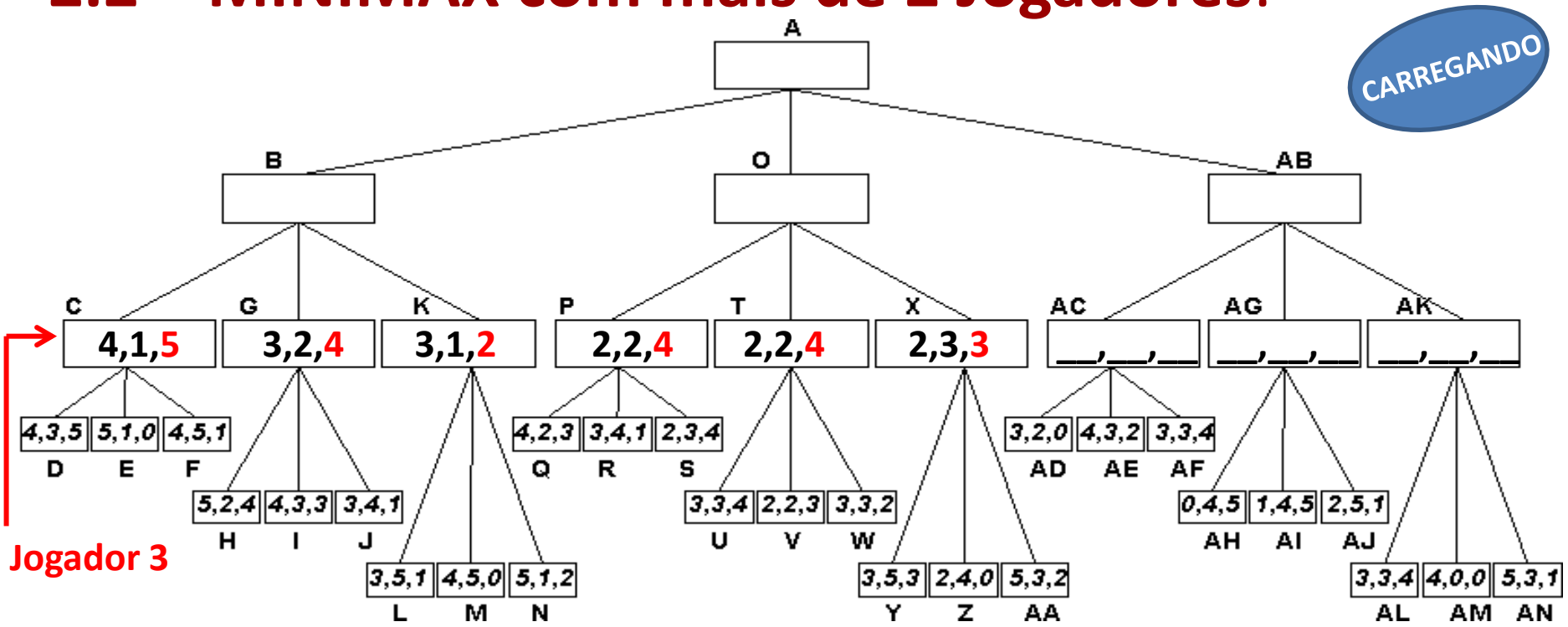


- Analisando os nós ***P***, ***T*** e ***X***, propaga-se...

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

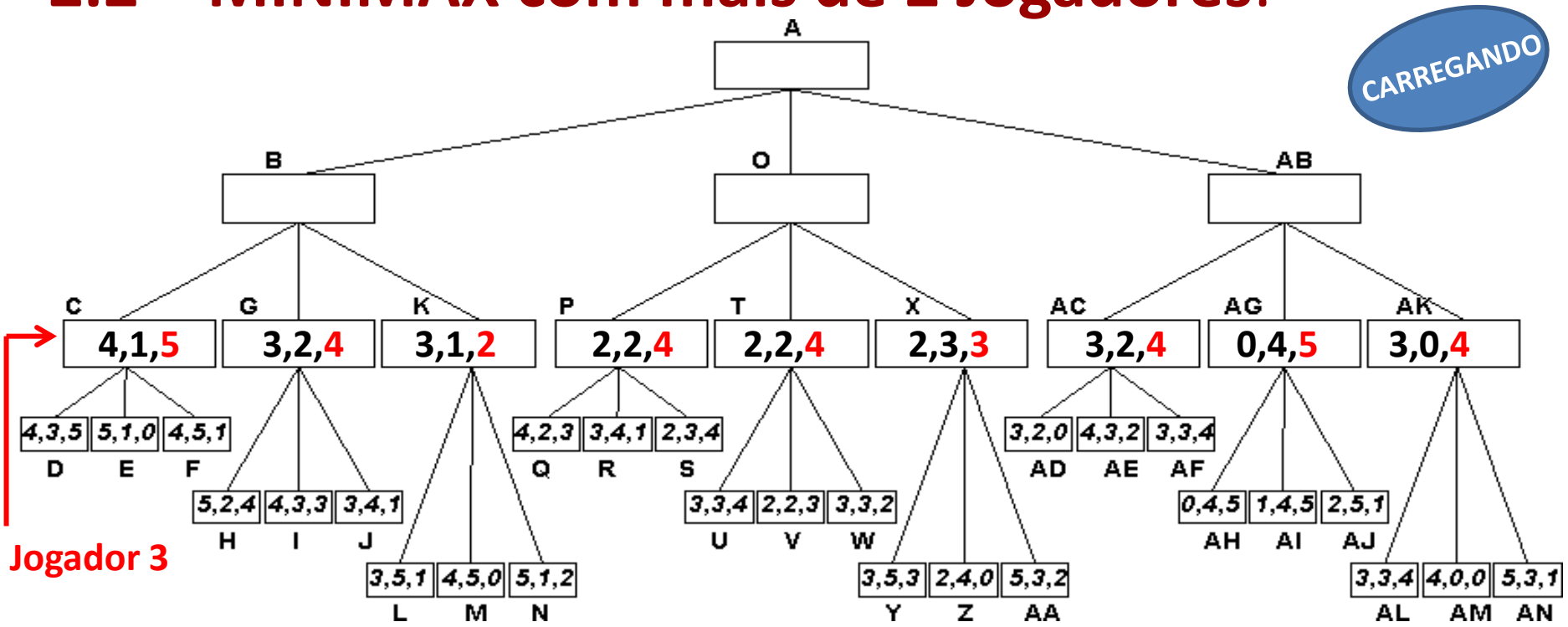


- Analisando os nós ***AC***, ***AG*** e ***AK***, propaga-se...

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

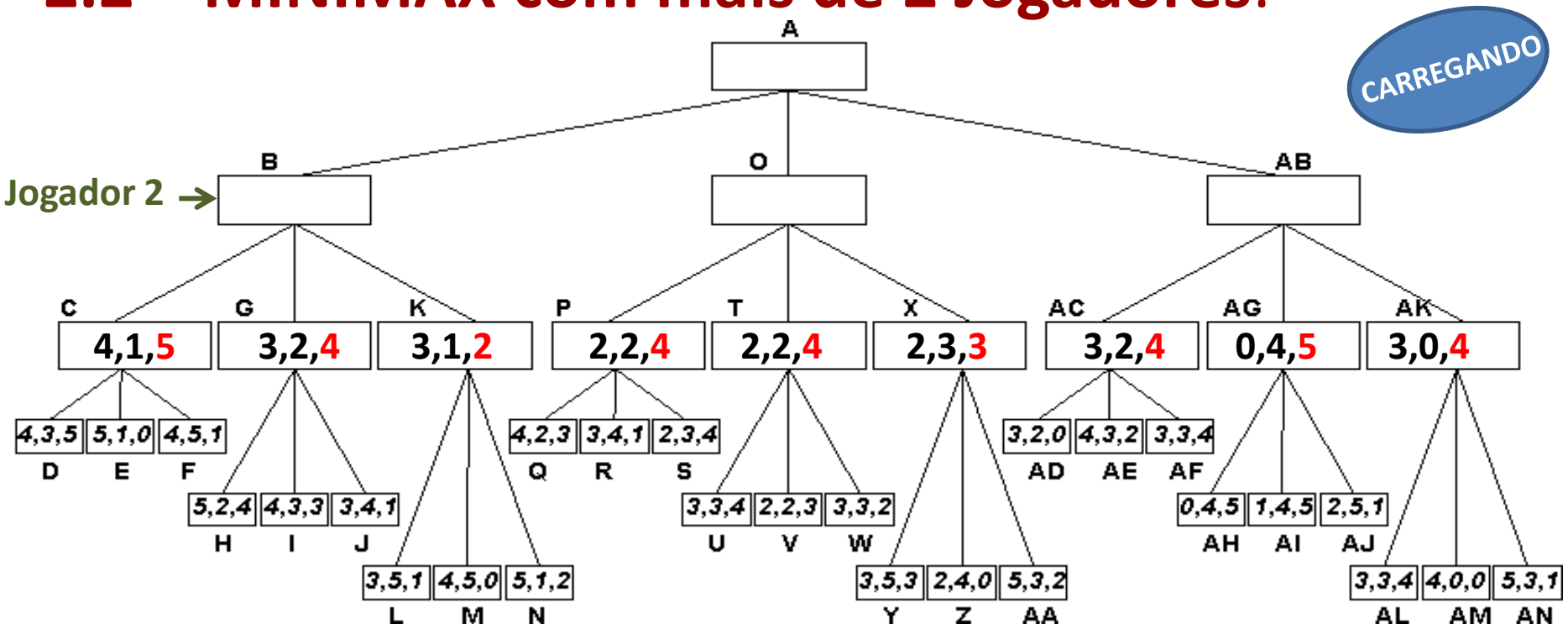


- Analisando os nós ***AC***, ***AG*** e ***AK***, propaga-se...

O Jogador 3 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

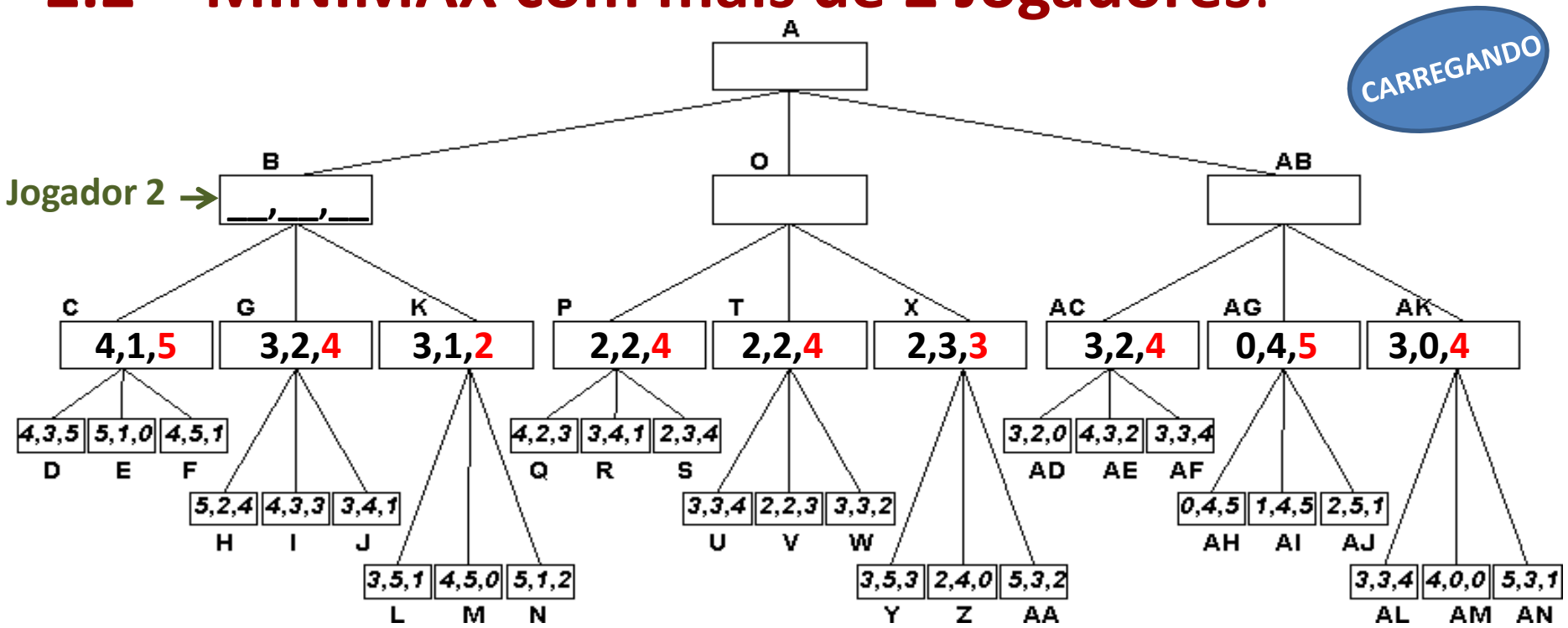
2.2 – MINIMAX com mais de 2 Jogadores:



- Propagando com o Jogador 2.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***B***, encontra-se os vetores:

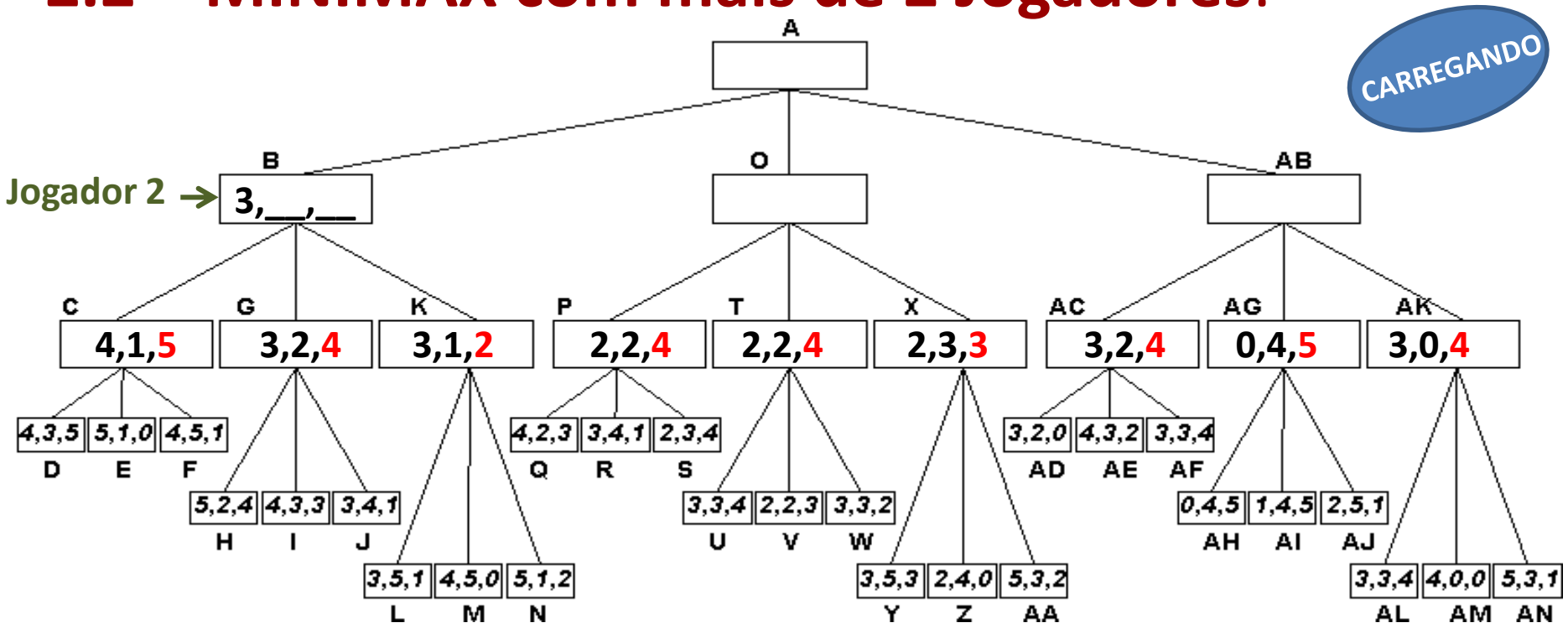
  C(v_{J1}=4, v_{J2}=1 e v_{J3}=5);

  G(v_{J1}=3, v_{J2}=2 e v_{J3}=4);

  K(v_{J1}=3, v_{J2}=1 e v_{J3}=2);

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***B***, encontra-se os vetores:

  C*(v_{J1}=4, v_{J2}=1 e v_{J3}=5)*;

  G*(v_{J1}=3, v_{J2}=2 e v_{J3}=4)*;

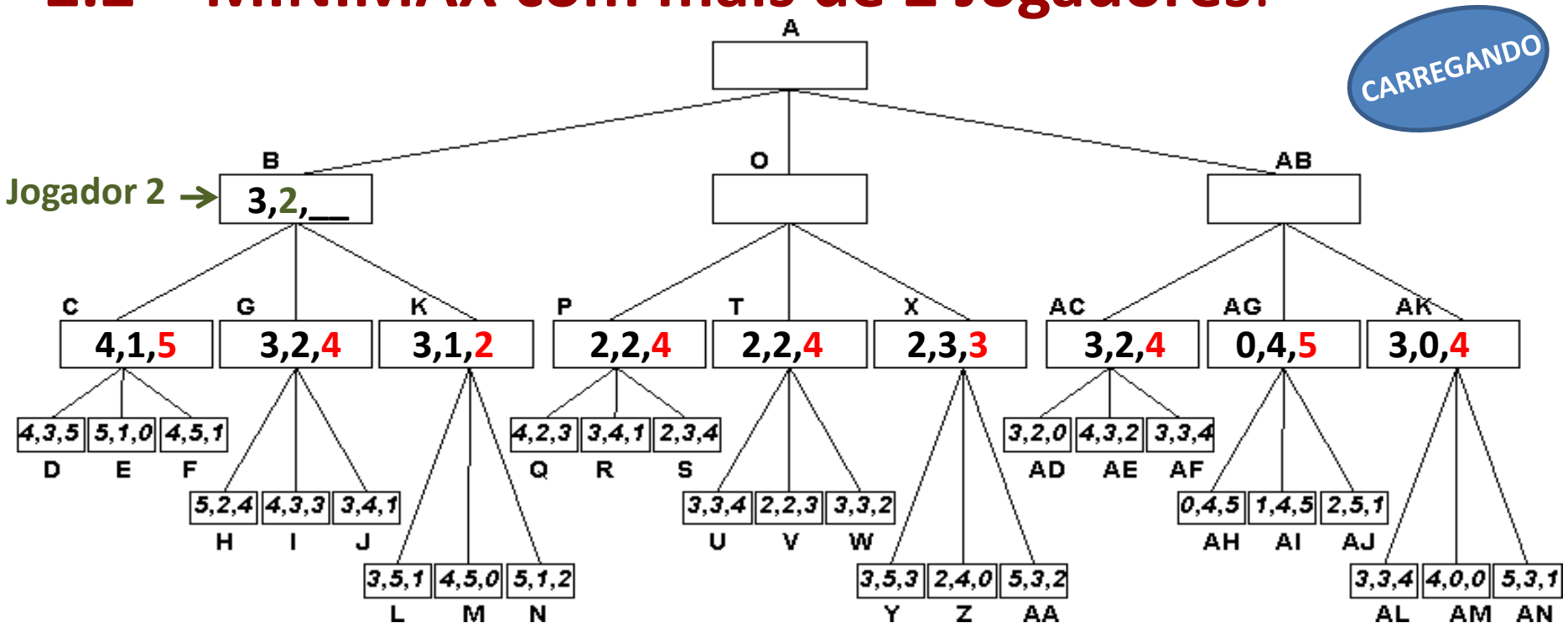
  K*(v_{J1}=3, v_{J2}=1 e v_{J3}=2)*;

  PIOR

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***B***, encontra-se os vetores:

  C(v_{J1}=4, v_{J2}=1 e v_{J3}=5);

  G(v_{J1}=3, v_{J2}=2 e v_{J3}=4);

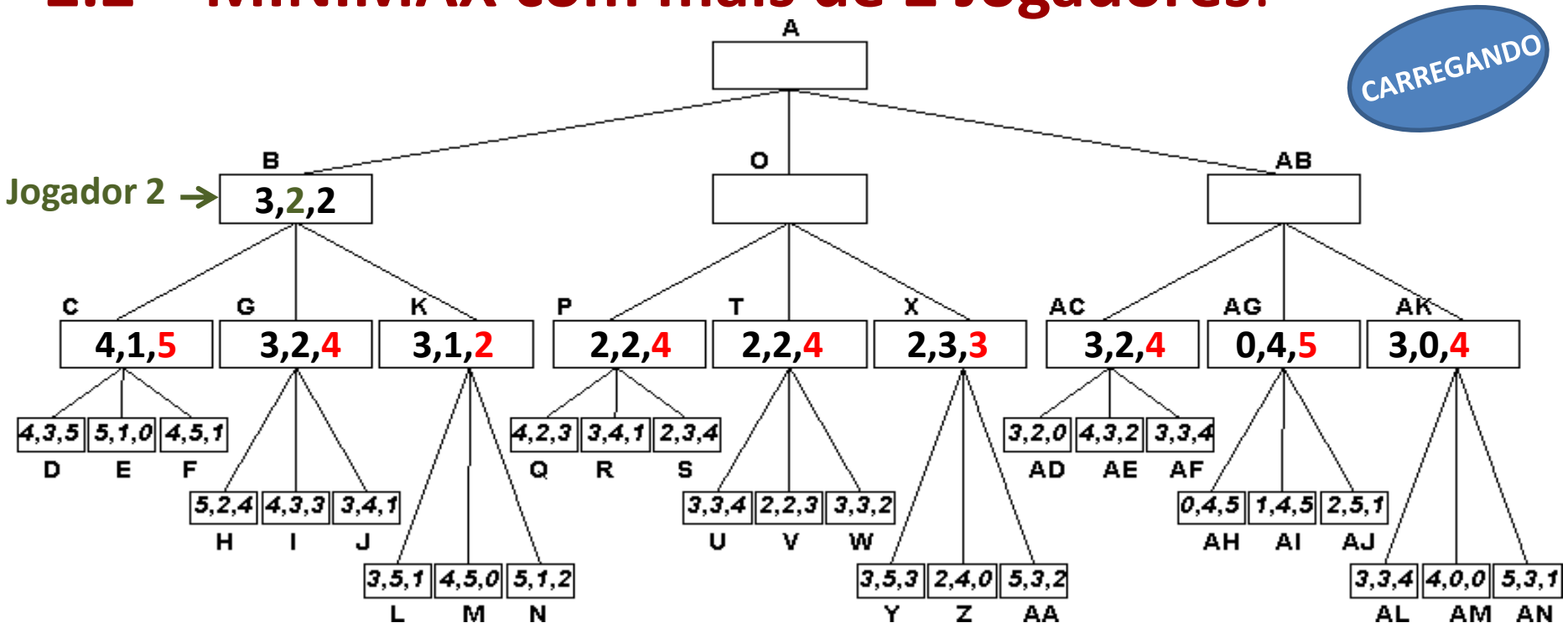
  K(v_{J1}=3, v_{J2}=1 e v_{J3}=2);

  MELHOR

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***B***, encontra-se os vetores:

  C(v_{J1}=4, v_{J2}=1 e v_{J3}=5);

  G(v_{J1}=3, v_{J2}=2 e v_{J3}=4);

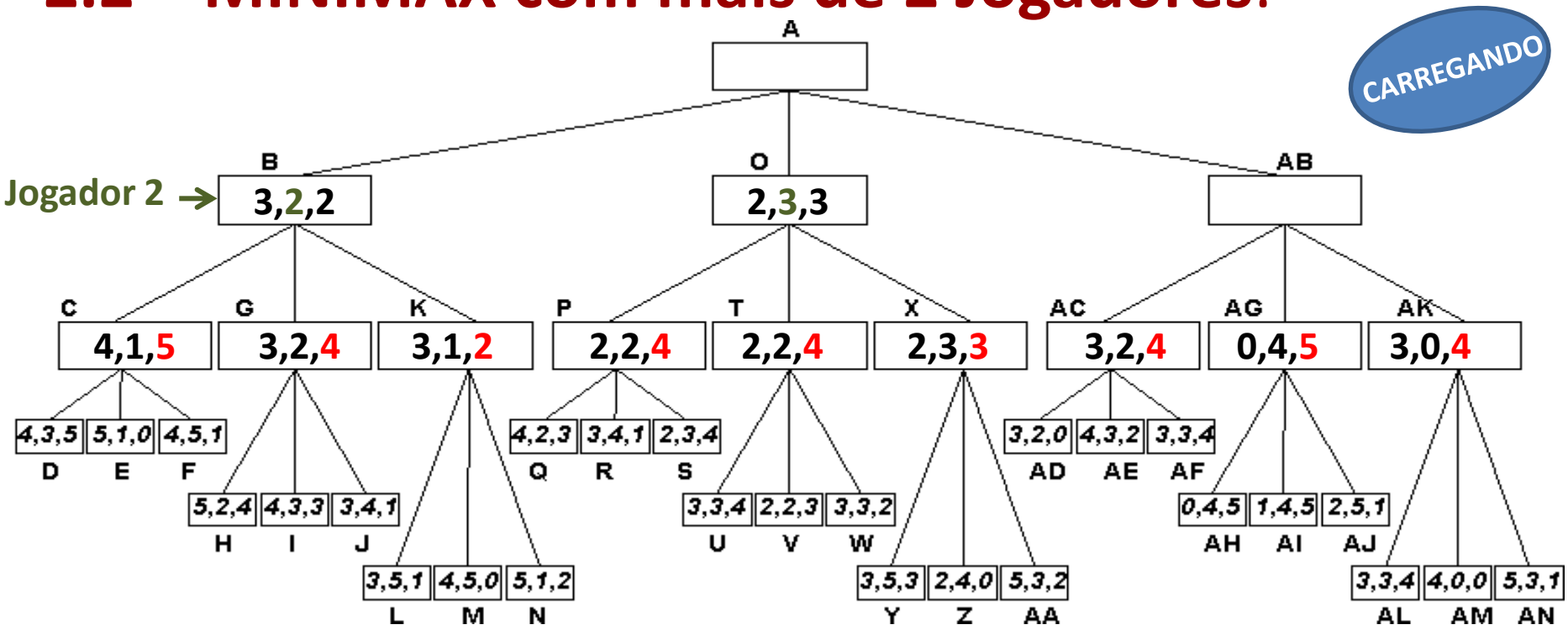
  K(v_{J1}=3, v_{J2}=1 e v_{J3}=2);

  PIOR

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

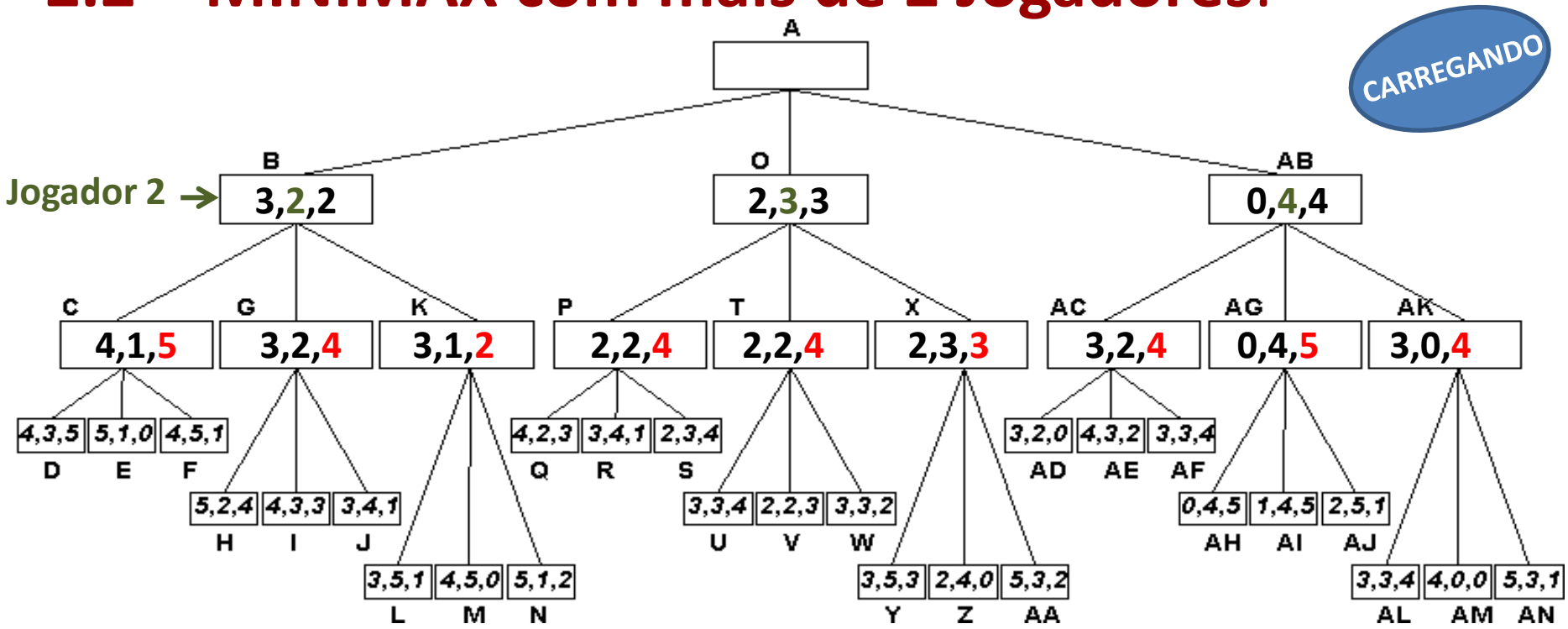


- Analisando o nó ***O***, propaga-se...

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:

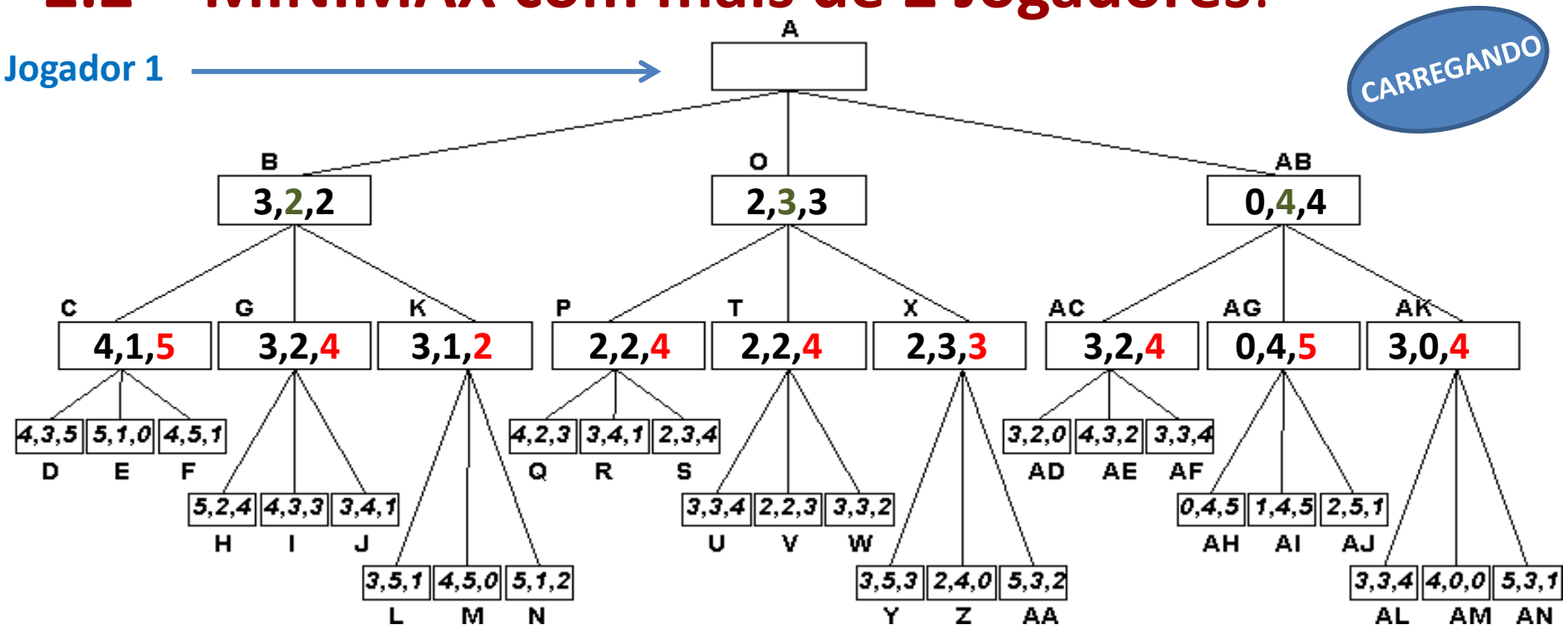


- Analisando o nó ***AB***, propaga-se...

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

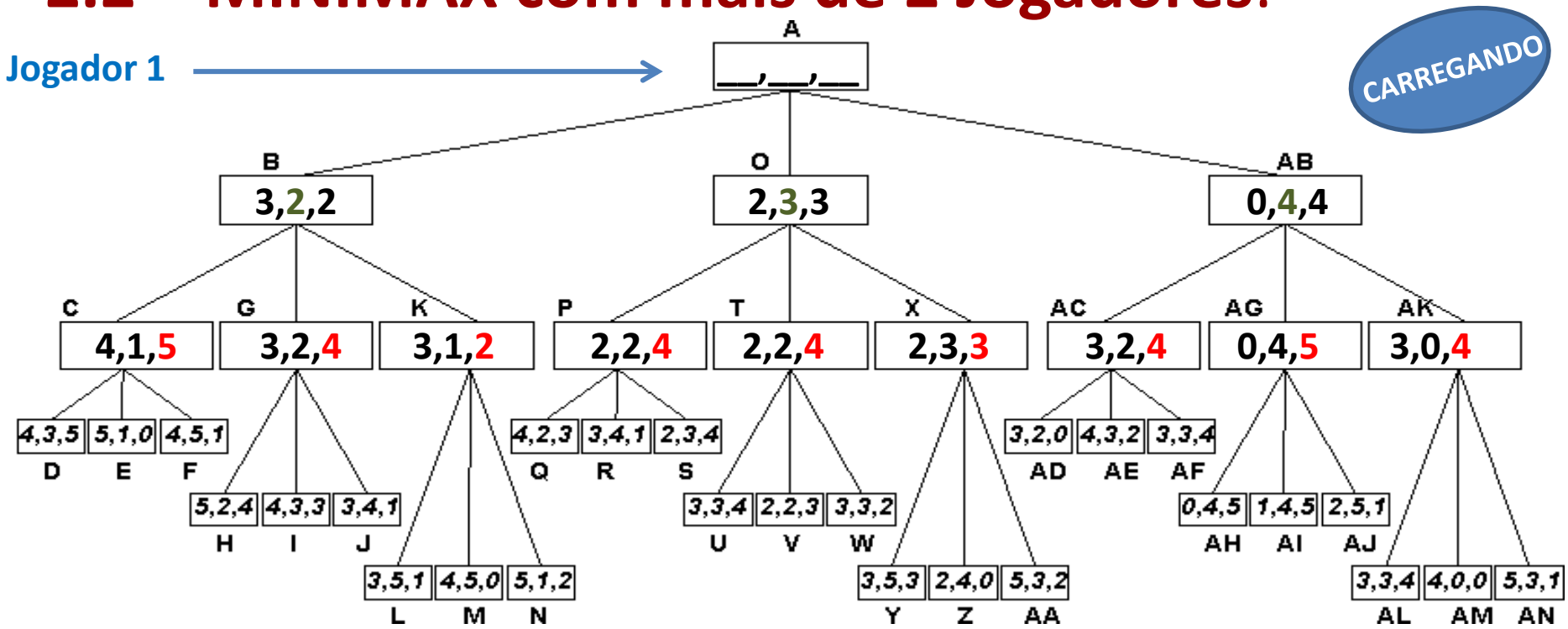
2.2 – MINIMAX com mais de 2 Jogadores:



- Propagando para o Jogador 1.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***A***, encontra-se os vetores:

  B *(v_{J1}=3, v_{J2}=2 e v_{J3}=2);*

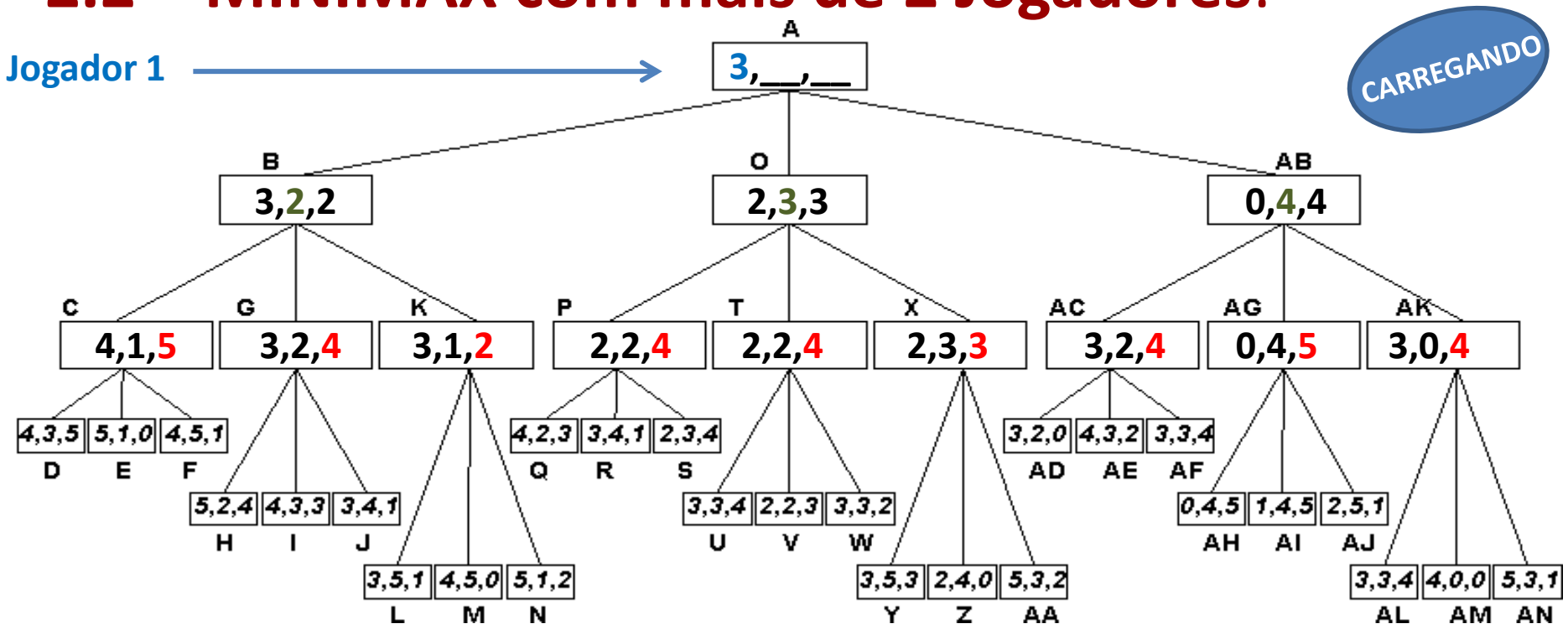
  O *(v_{J1}=2, v_{J2}=3 e v_{J3}=3);*

  AB*(v_{J1}=0, v_{J2}=4 e v_{J3}=4);*

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***A***, encontra-se os vetores:

  B *(v_{J1}=3, v_{J2}=2 e v_{J3}=2);*

  O *(v_{J1}=2, v_{J2}=3 e v_{J3}=3);*

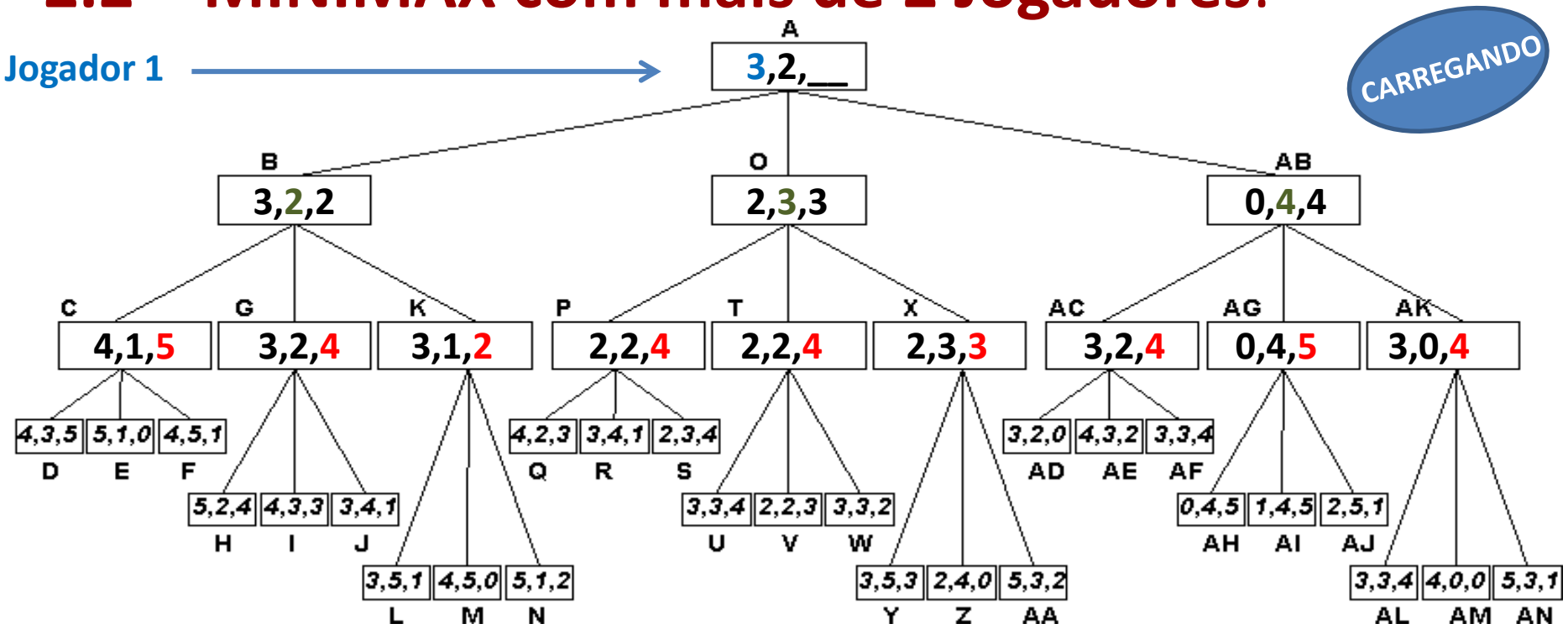
  AB *(v_{J1}=0, v_{J2}=4 e v_{J3}=4);*

  MELHOR

O Jogador 1 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***A***, encontra-se os vetores:

  B *(v_{J1}=3, v_{J2}=2 e v_{J3}=2);*

  O *(v_{J1}=2, v_{J2}=3 e v_{J3}=3);*

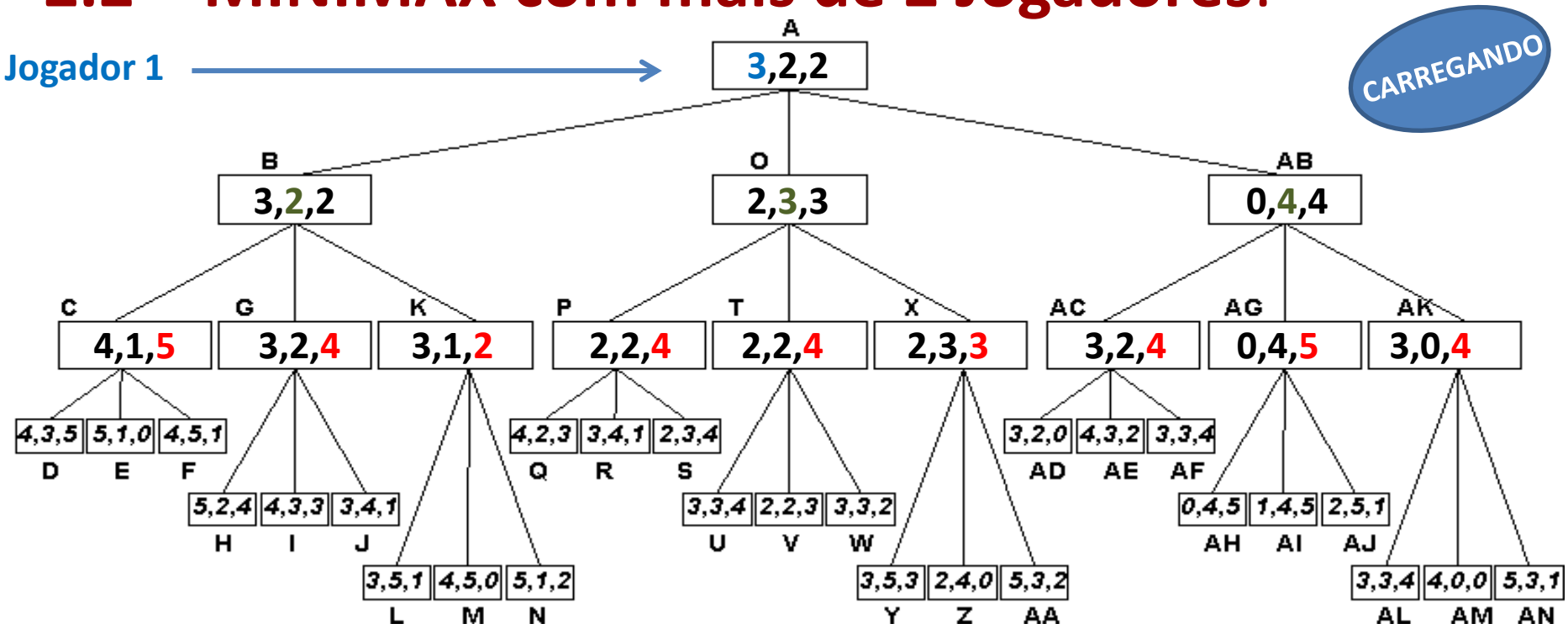
  AB*(v_{J1}=0, v_{J2}=4 e v_{J3}=4);*

  PIOR

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



- Analisando o nó ***A***, encontra-se os vetores:

  B *(v_{J1}=3, v_{J2}=2 e v_{J3}=2)*;

  O *(v_{J1}=2, v_{J2}=3 e v_{J3}=3)*;

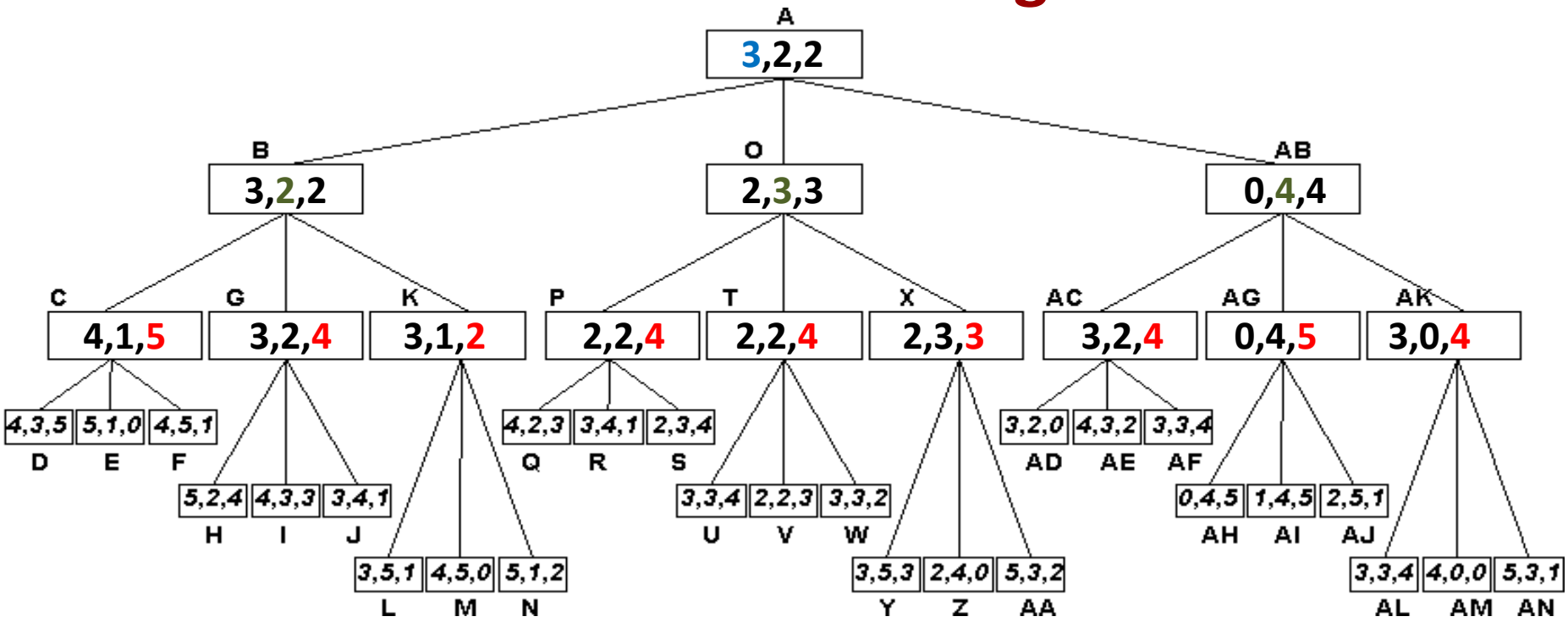
  AB*(v_{J1}=0, v_{J2}=4 e v_{J3}=4)*;

  PIOR

O Jogador 2 irá propagar o melhor para si e o pior para os concorrentes.

2 – Busca MINIMAX

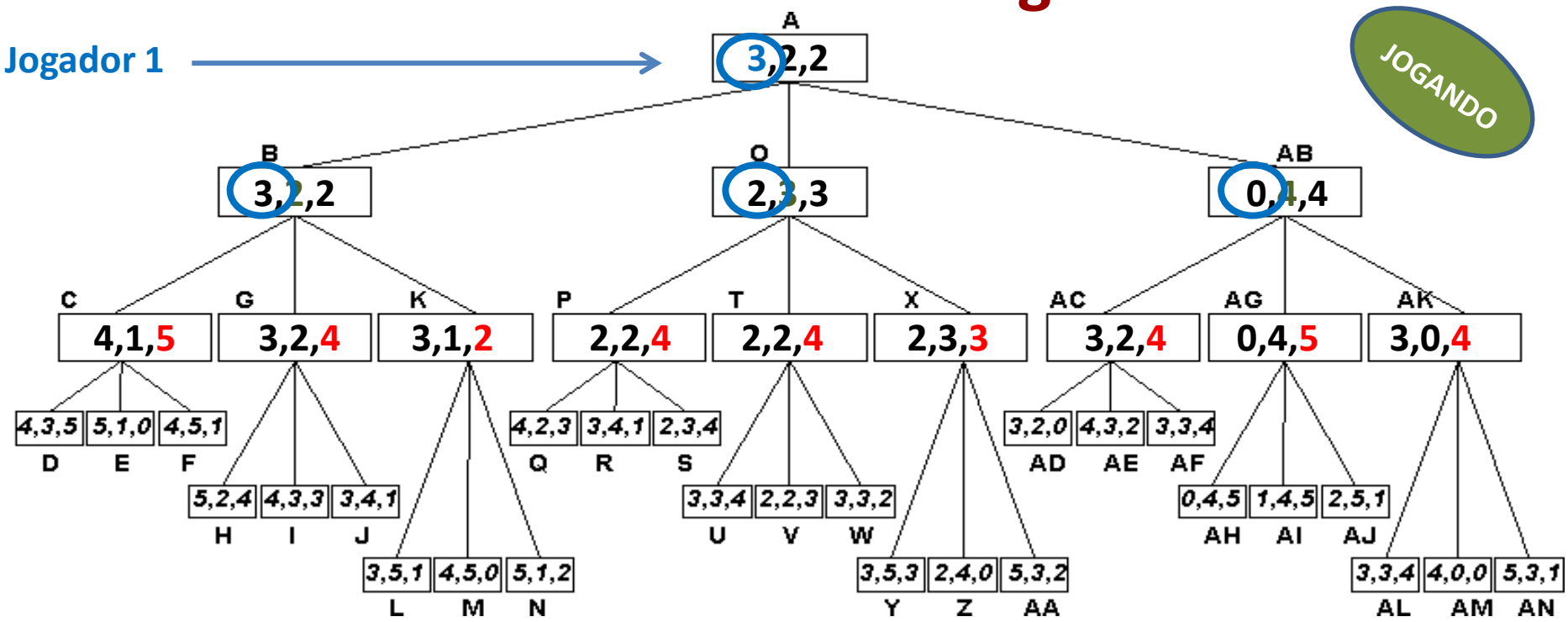
2.2 – MINIMAX com mais de 2 Jogadores:



- Conhecendo todos os valores MINIMAX, pode-se montar a **Árvore de Jogo**, a partir da raiz, como temos a seguir...

2 – Busca MINIMAX

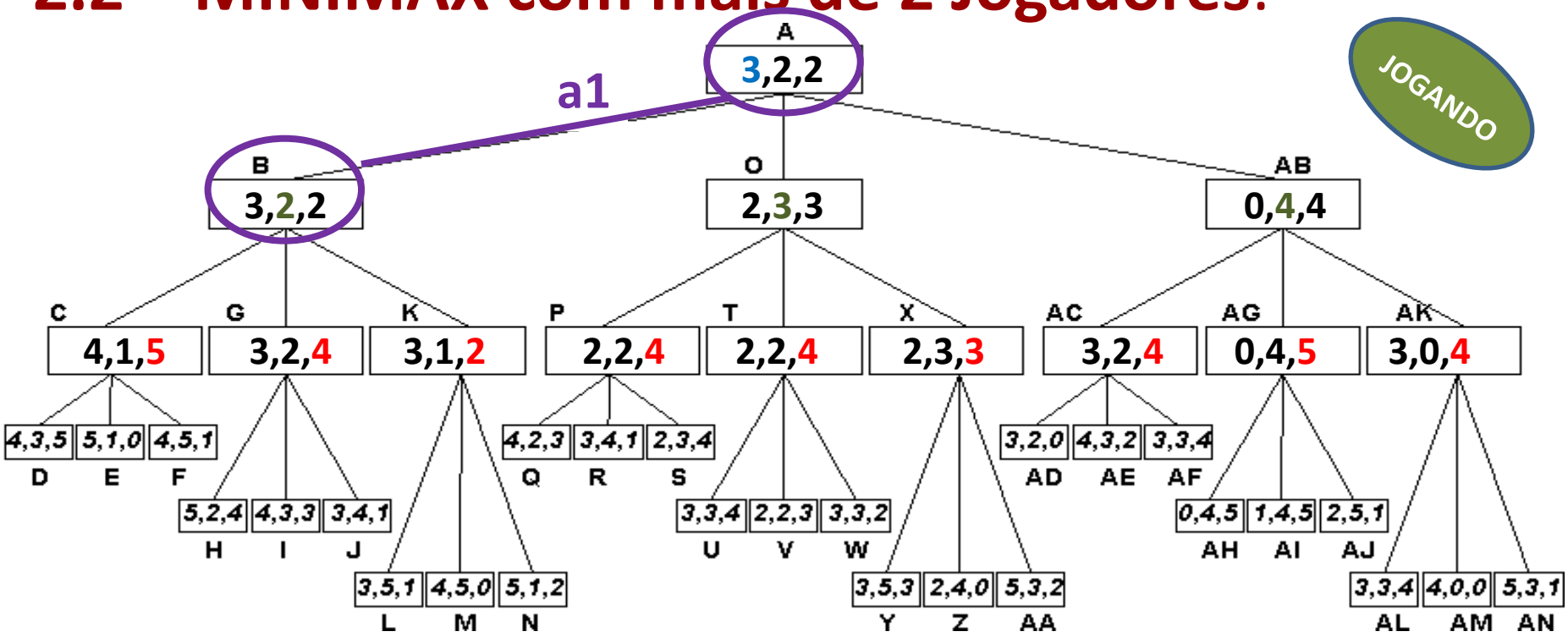
2.2 – MINIMAX com mais de 2 Jogadores:



- Analisa-se o valor MINIMAX para o Jogador 1 através do valor propagado no vetor A.

2 – Busca MINIMAX

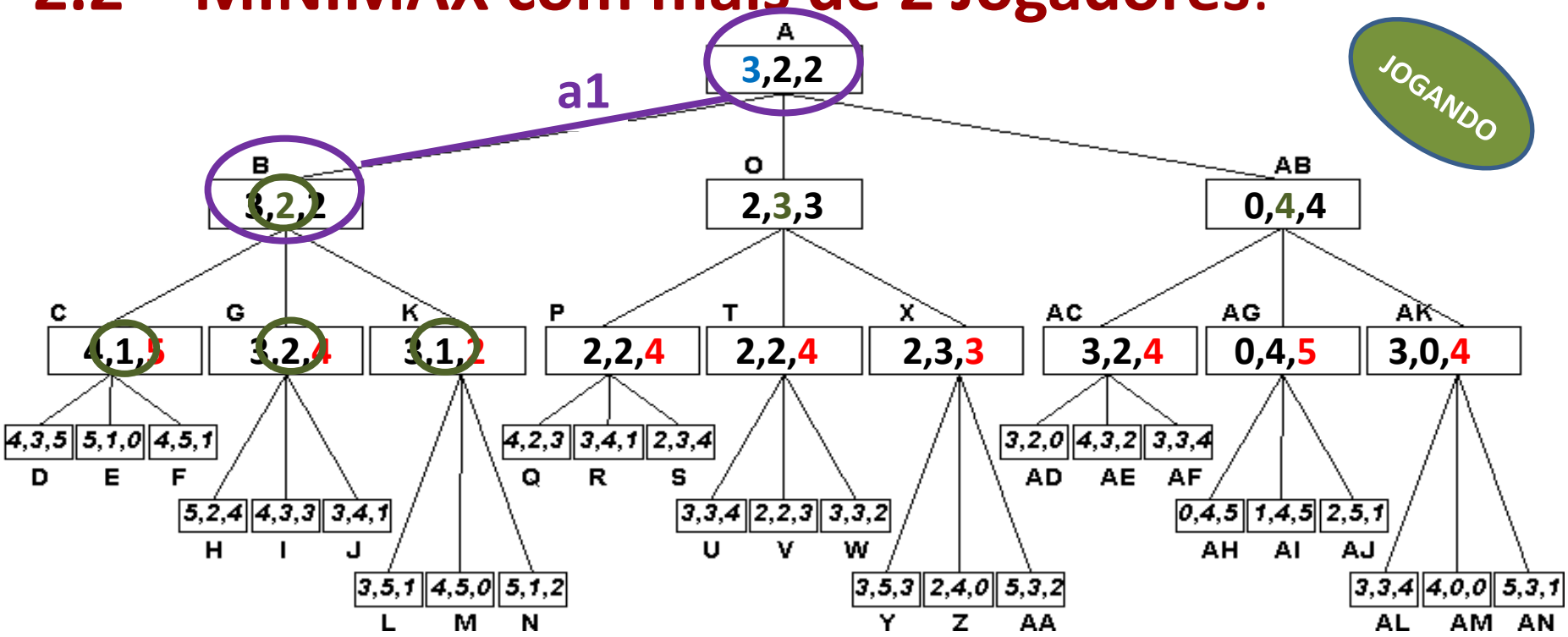
2.2 – MINIMAX com mais de 2 Jogadores:



- Identifica-se a decisão MINIMAX na raiz, sendo a ação **a1** a escolha ótima para o Jogador 1, porque leva ao sucessor com o mais alto valor MINIMAX para o Jogador 1.

2 – Busca MINIMAX

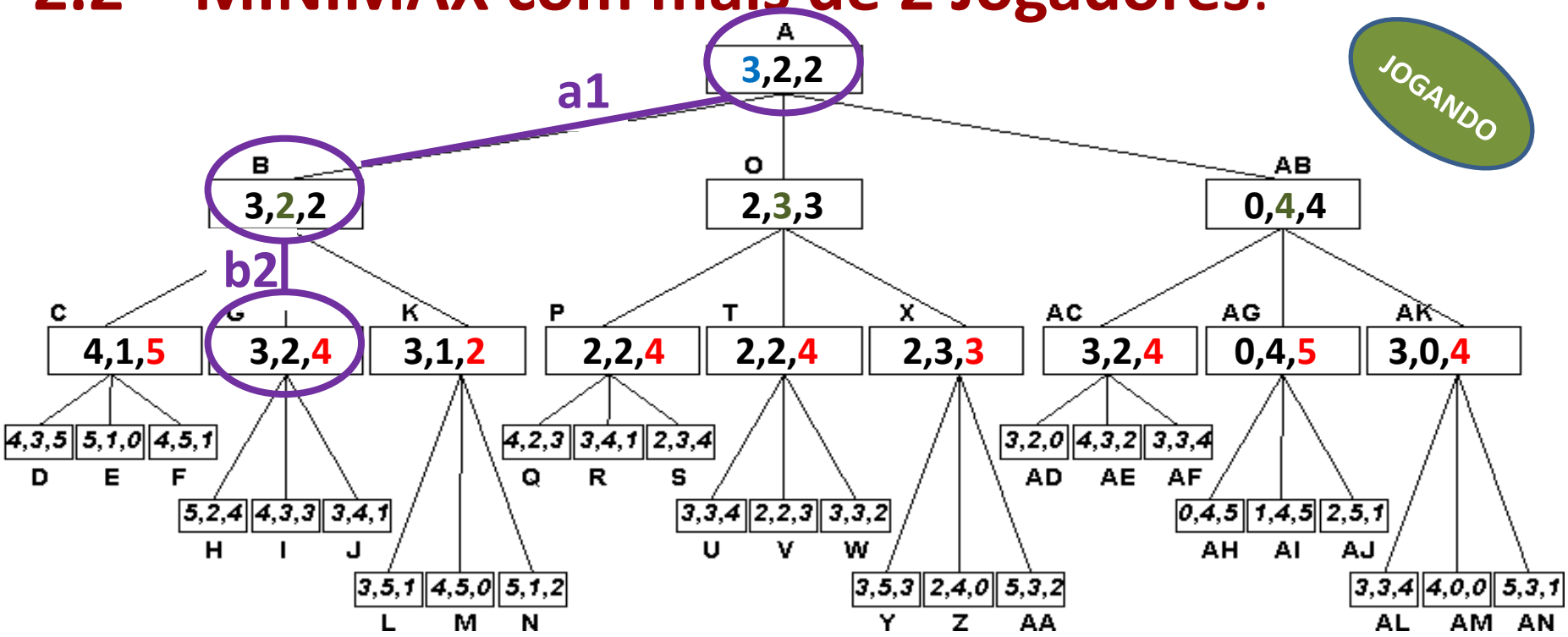
2.2 – MINIMAX com mais de 2 Jogadores:



- Analisa-se o valor MINIMAX para o Jogador 2 através do valor propagado no vetor B (que foi a escolha do Jogador 1).

2 – Busca MINIMAX

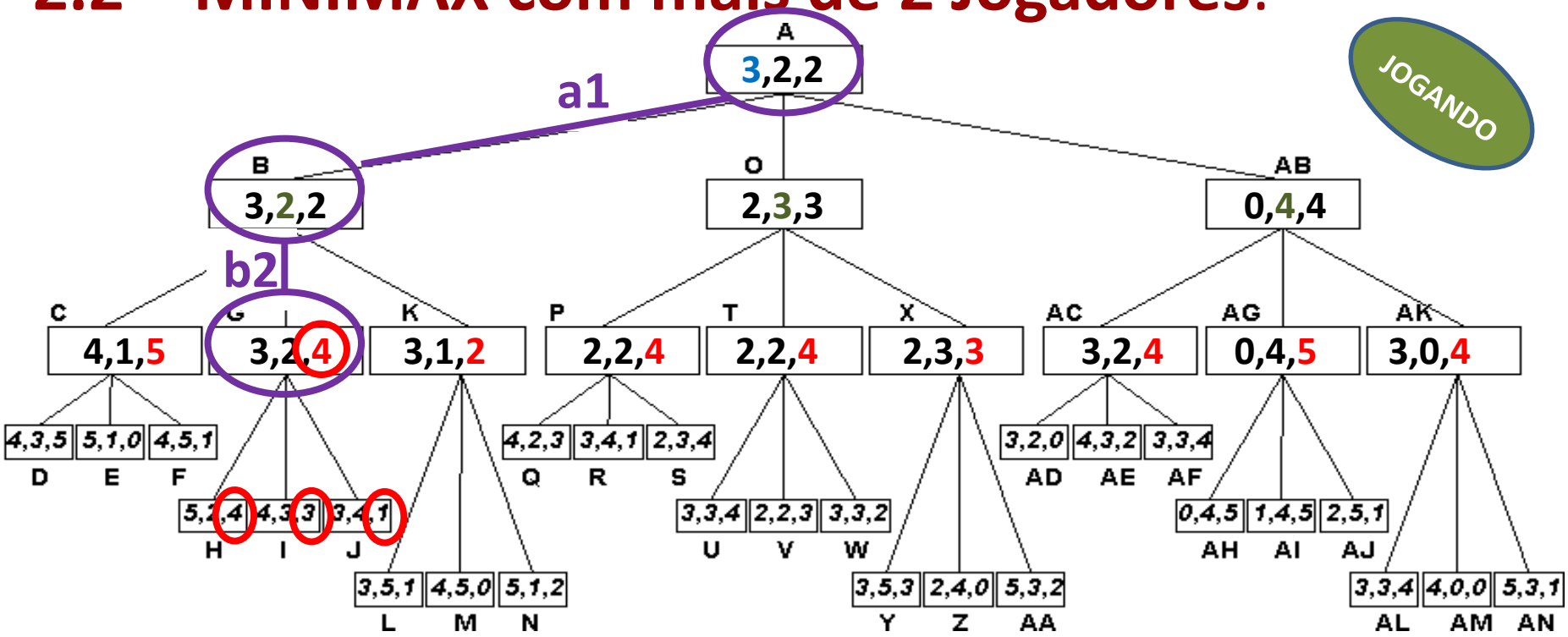
2.2 – MINIMAX com mais de 2 Jogadores:



- Identifica-se a decisão MINIMAX, sendo a ação **b2** a escolha ótima para o Jogador 2, porque leva ao sucessor com o mais alto valor MINIMAX para o Jogador 2.

2 – Busca MINIMAX

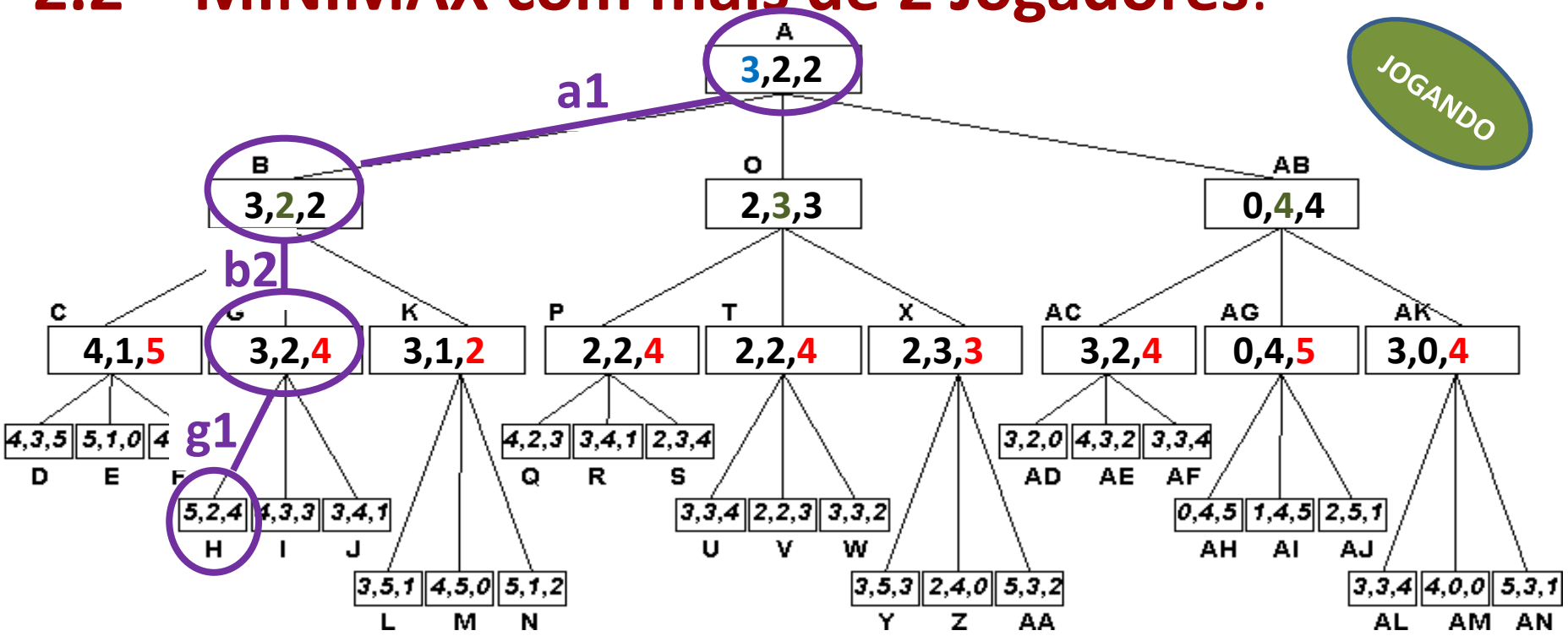
2.2 – MINIMAX com mais de 2 Jogadores:



- Analisa-se o valor MINIMAX para o Jogador 3 através do valor propagado no vetor G (que foi a escolha do Jogador 2).

2 – Busca MINIMAX

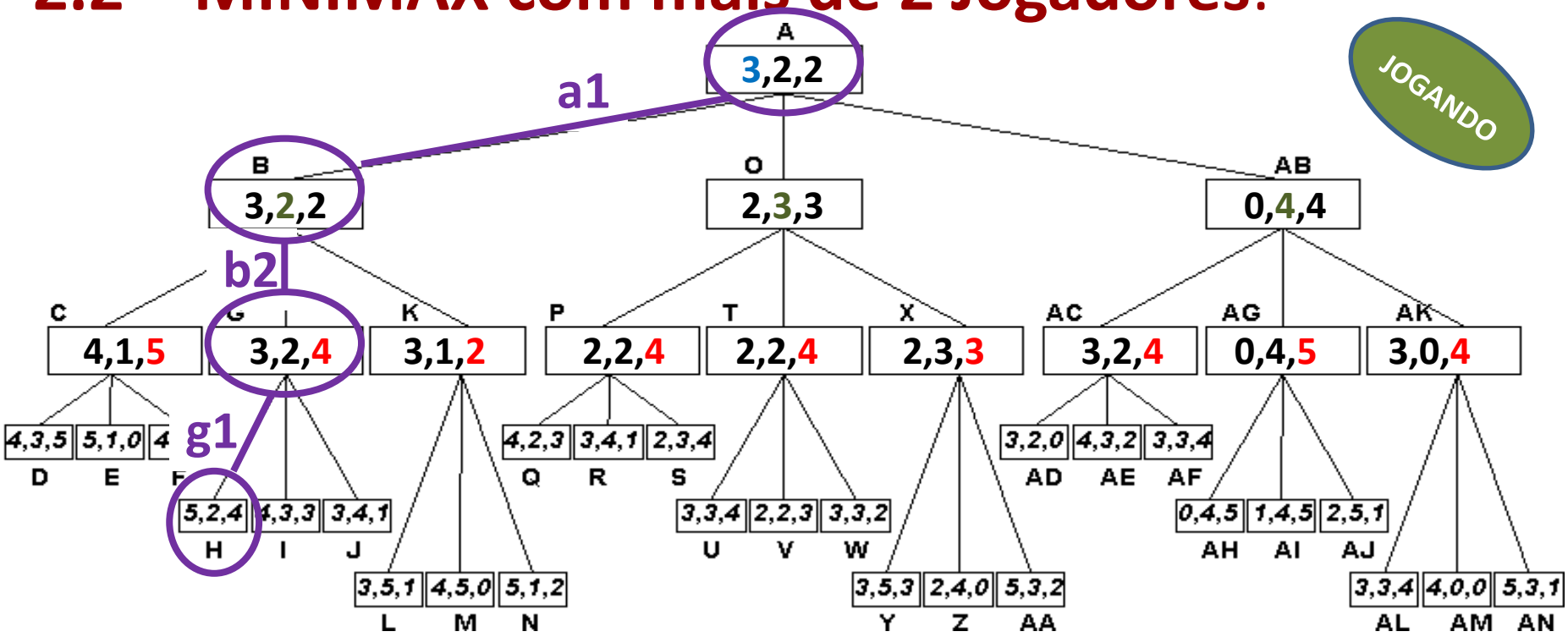
2.2 – MINIMAX com mais de 2 Jogadores:



- Identifica-se a decisão MINIMAX, sendo a ação **g1** a escolha ótima para o Jogador 3, porque leva ao sucessor com o mais alto valor MINIMAX para o Jogador 3.

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores:



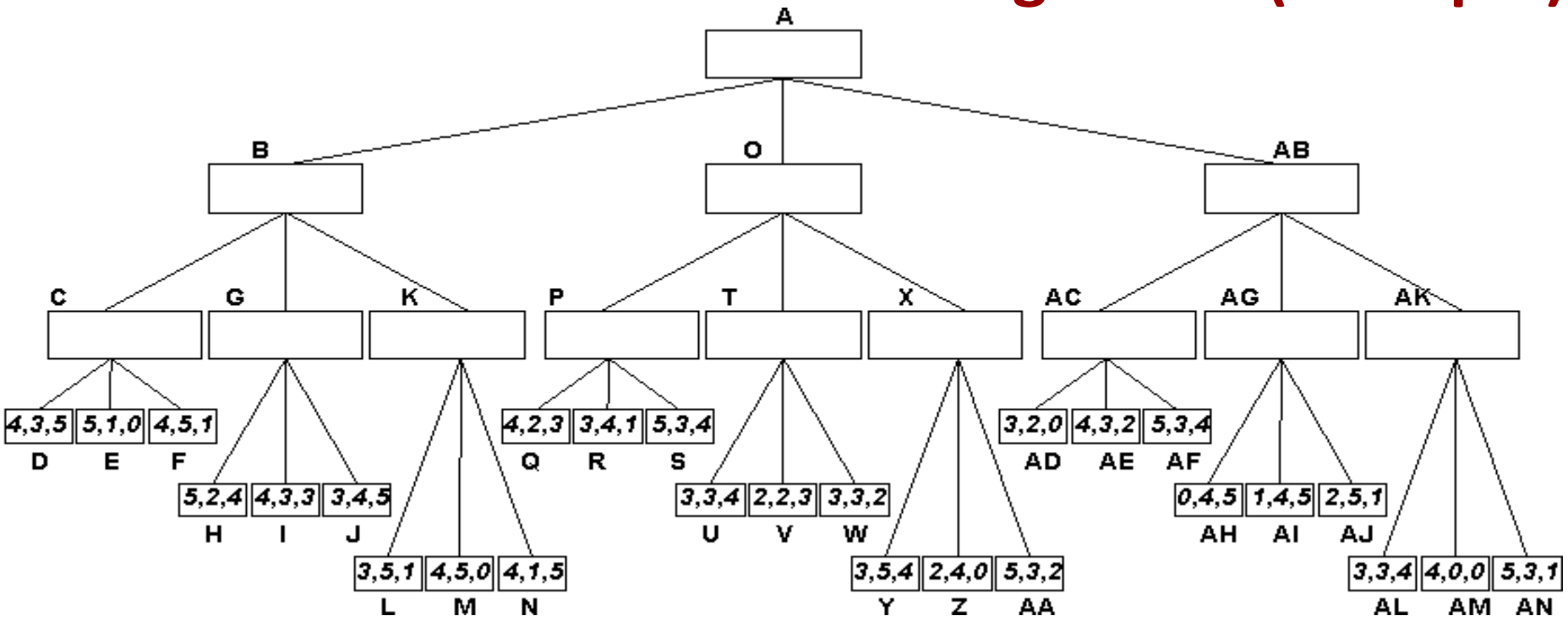
- Assim, tem-se a árvore do jogo como sendo:

$$A \rightarrow B \rightarrow G \rightarrow H$$

E o **vencedor** deste jogo é aquele que possuir o maior valor no vetor terminal, neste caso no vetor H(5,2,4) que é o **Jogador 1**.

2 – Busca MINIMAX

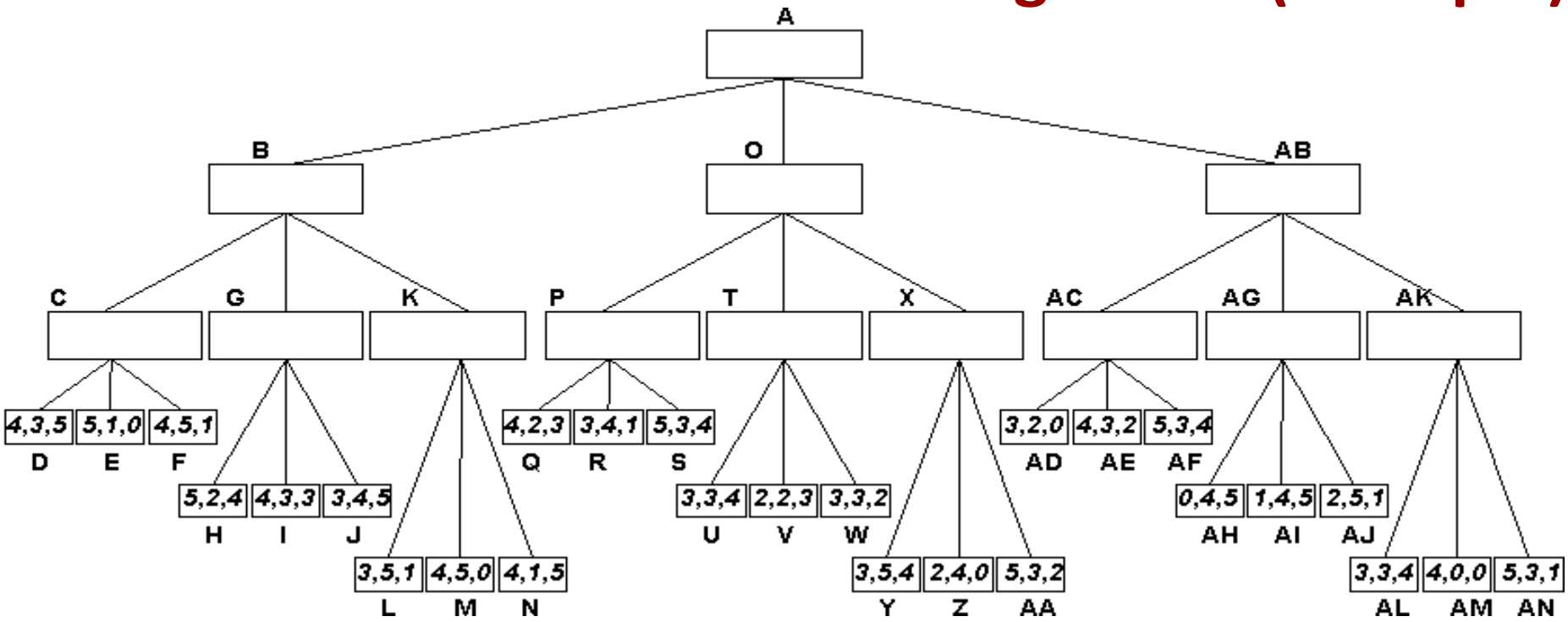
2.2 – MINIMAX com mais de 2 Jogadores (exemplo):



- Considere um jogo com 3 jogadores, com 3 possibilidades de escolha por jogada e 1 rodada por jogador. A utilidade de cada jogada é [0,1,...,5].

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores (exemplo):

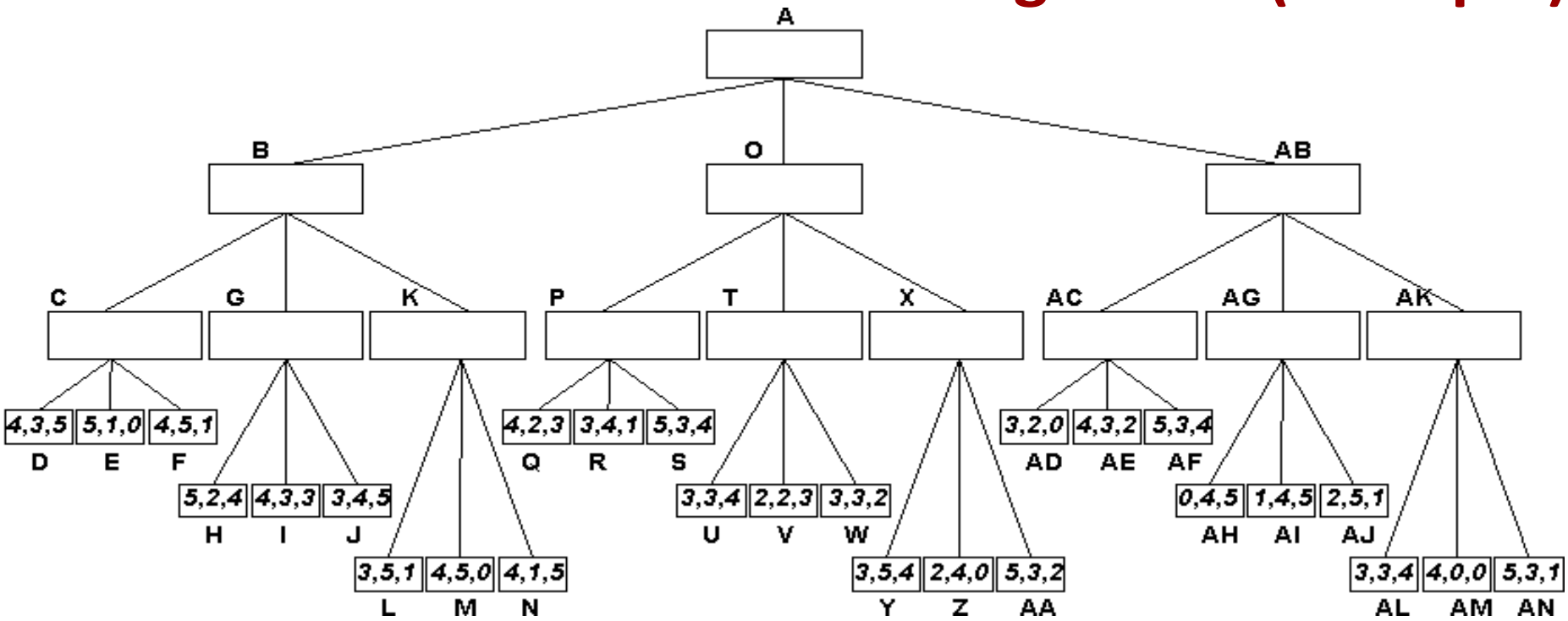


a) Se todos os jogadores seguirem o algoritmo MINIMAX, quem vence?

b) Se todos os jogadores seguirem o algoritmo MINIMAX como será a árvore do jogo?

2 – Busca MINIMAX

2.2 – MINIMAX com mais de 2 Jogadores (exemplo):



c) Se apenas o JOGADOR 1 seguir o MINIMAX, como ele poderia vencer? Monte a árvore deste jogo.

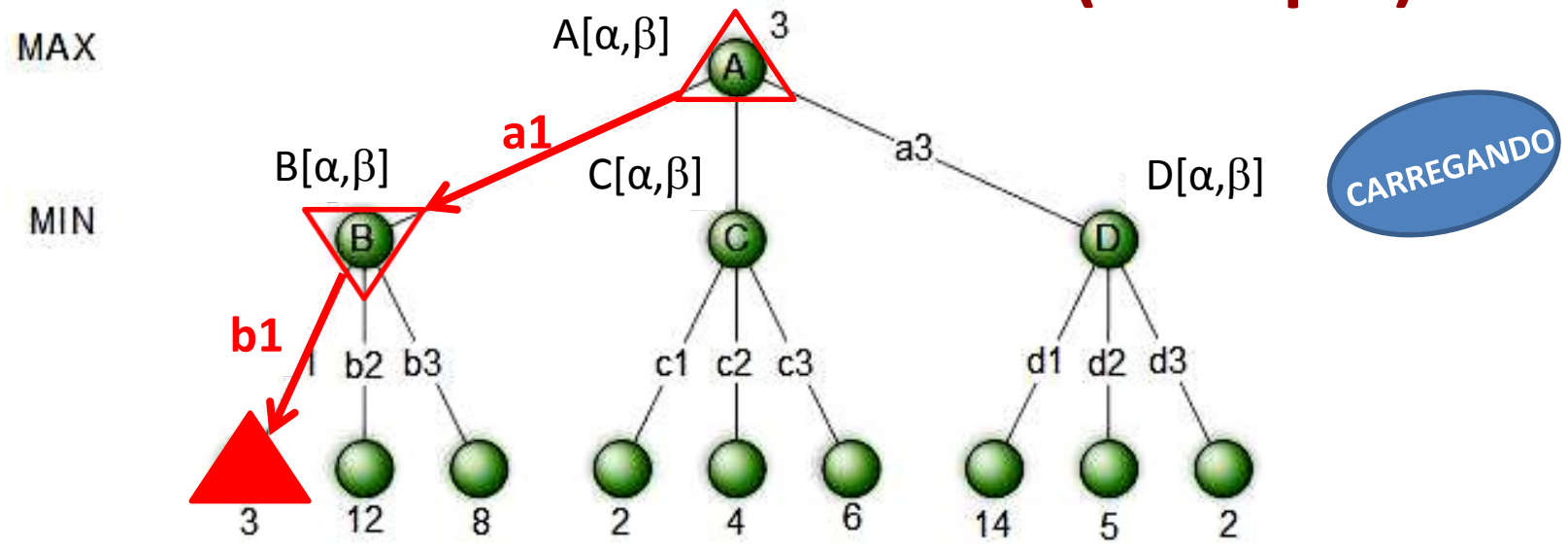
d) Se apenas o JOGADOR 3 seguir o MINIMAX, como ele poderia vencer? Monte a árvore deste jogo.

2.3 – MINIMAX com PODA alfa-beta

- O problema da busca MINIMAX é que **o número de estados de jogo** que ela tem de examinar **é exponencial** em relação ao número de movimentos!
- A ideia, **para diminuir o esforço computacional**, é a possibilidade de calcular a decisão MINIMAX correta **sem examinar todos os nós** na árvore de jogo.
- Quando aplicada a uma árvore MINIMAX, a técnica de **PODA retorna o mesmo movimento que o MINIMAX** retornaria, mas poda as ramificações que não terão influência sobre a decisão final.

2 – Busca MINIMAX

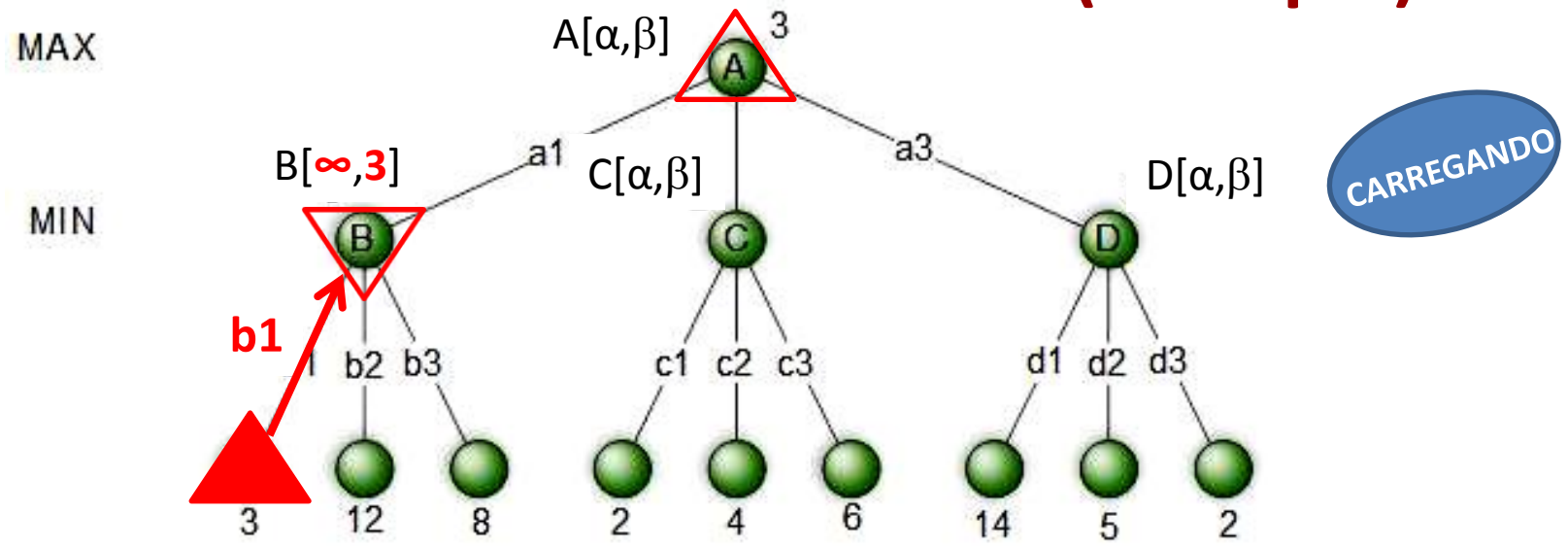
2.3 – MINIMAX com PODA alfa-beta (exemplo):



- Faz-se a busca em profundidade, a partir da raiz.
- O primeiro nó de **MIN**, identificado por **B**, possui uma folha **b1** com valor 3.

2 – Busca MINIMAX

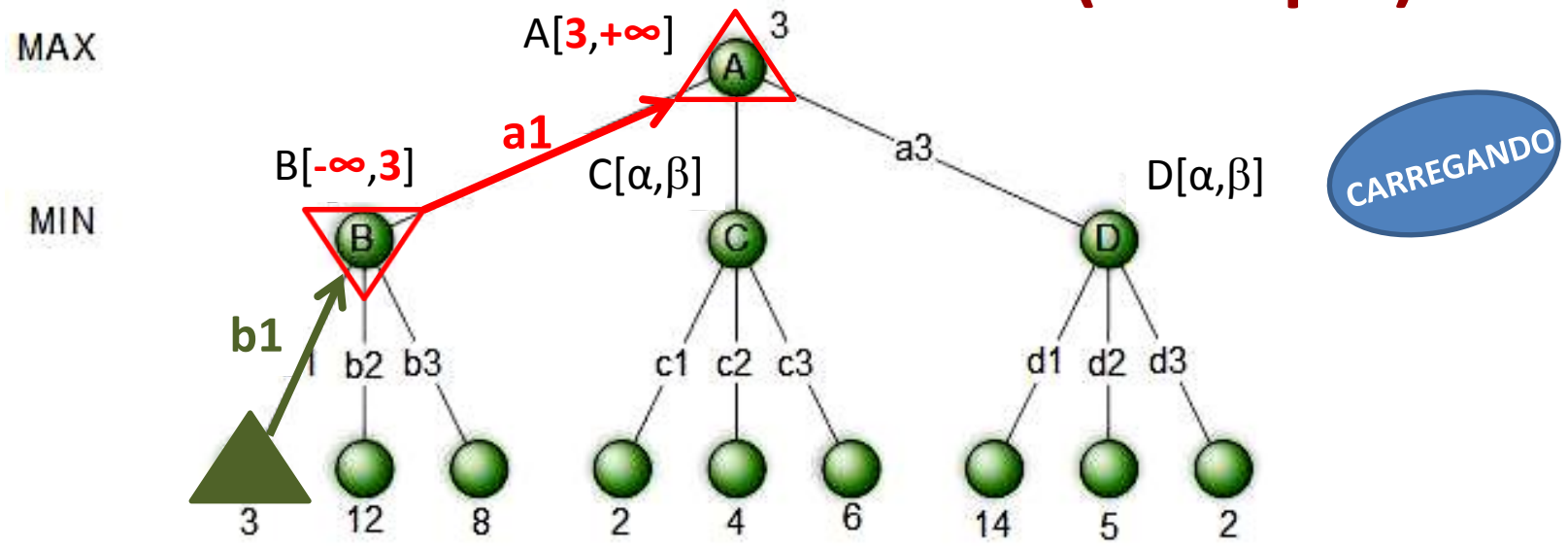
2.3 – MINIMAX com PODA alfa-beta (exemplo):



- Pode-se concluir que, por seu um nó de **MIN**, **B** terá no máximo o valor **3**, e no mínimo um valor infinito qualquer.

2 – Busca MINIMAX

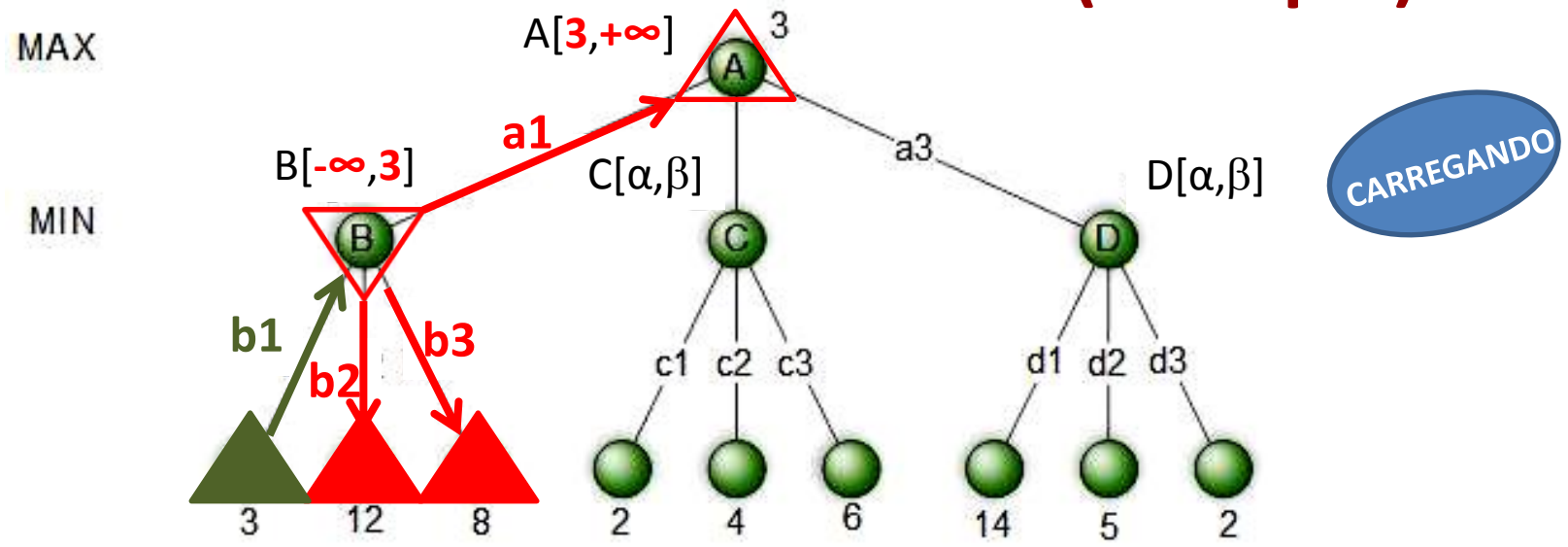
2.3 – MINIMAX com PODA alfa-beta (exemplo):



- Da mesma forma, pode-se concluir que, por seu um nó de **MAX**, **A** terá no mínimo o valor **3**, e no máximo um valor (-) infinito qualquer.

2 – Busca MINIMAX

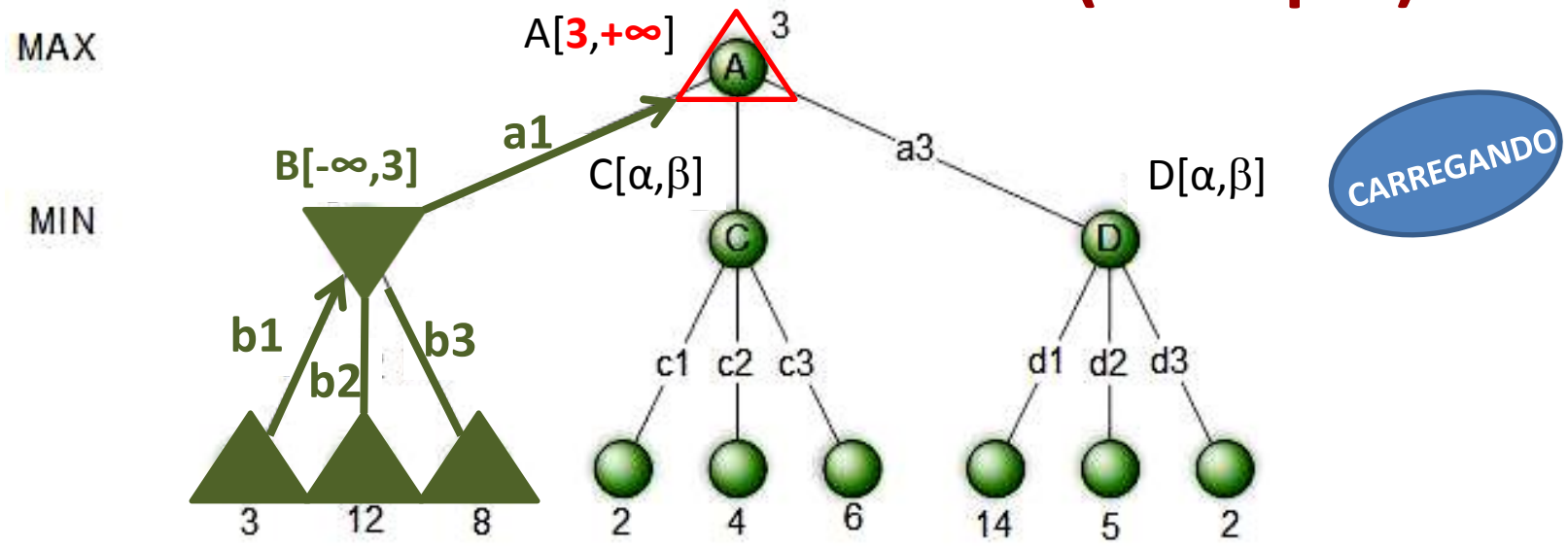
2.3 – MINIMAX com PODA alfa-beta (exemplo):



- Analisando a folha **b2** e **b3**, encontra-se valores maiores que **3**.
- O algoritmo evitaria estes 2 movimentos, apesar de tê-los visitado.

2 – Busca MINIMAX

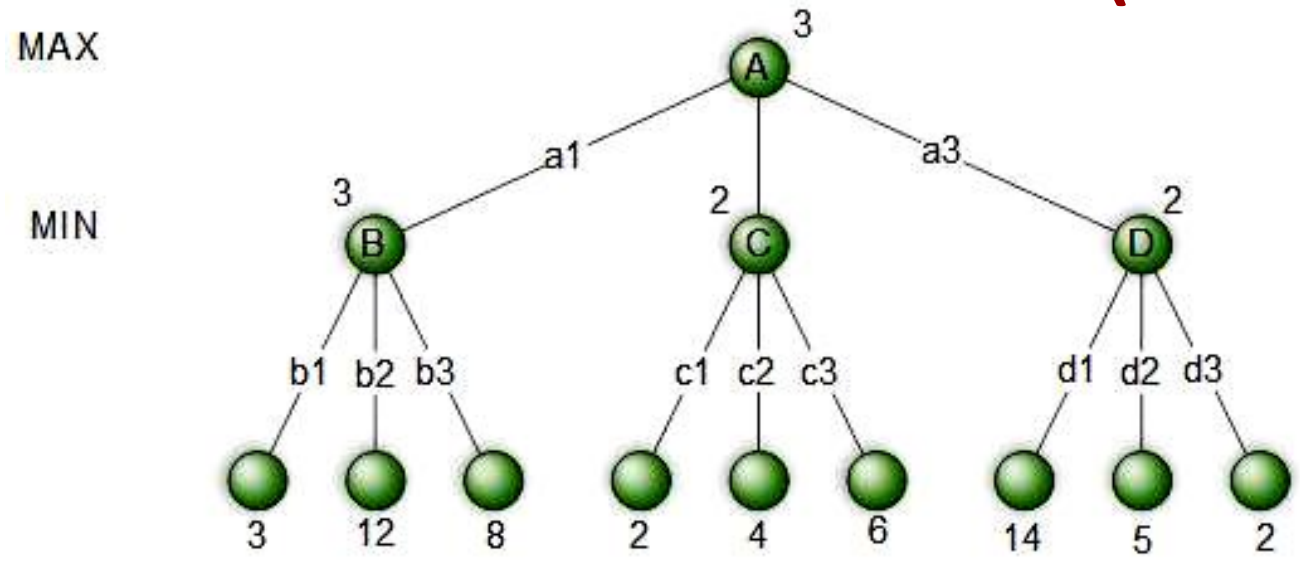
2.3 – MINIMAX com PODA alfa-beta (exemplo):



- Analisando a folha **b2** e **b3**, encontra-se valores maiores que **3**.
- O algoritmo evitaria estes 2 movimentos, apesar de tê-los visitado.

2 – Busca MINIMAX

2.3 – MINIMAX com PODA alfa-beta (exemplo):



CARREGANDO

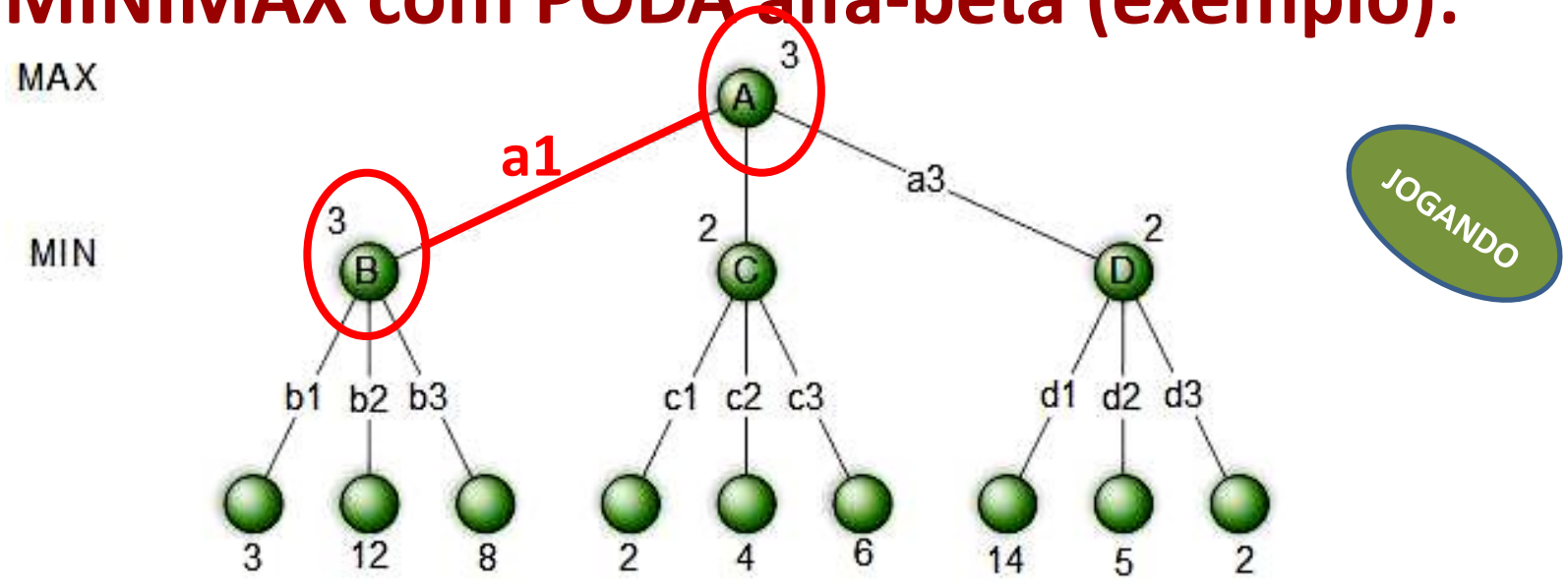
- Conhecendo todos os valores MINIMAX, pode-se calcular o valor MINIMAX, a partir da raiz:

$$\mathbf{MINIMAX(A) = max\{min(3,12,8), min(2,4,6), min(14,5,2)\}}$$

- Conhecendo todos os valores MINIMAX, pode-se montar a **Árvore de Jogo**, a partir da raiz, como temos a seguir...

2 – Busca MINIMAX

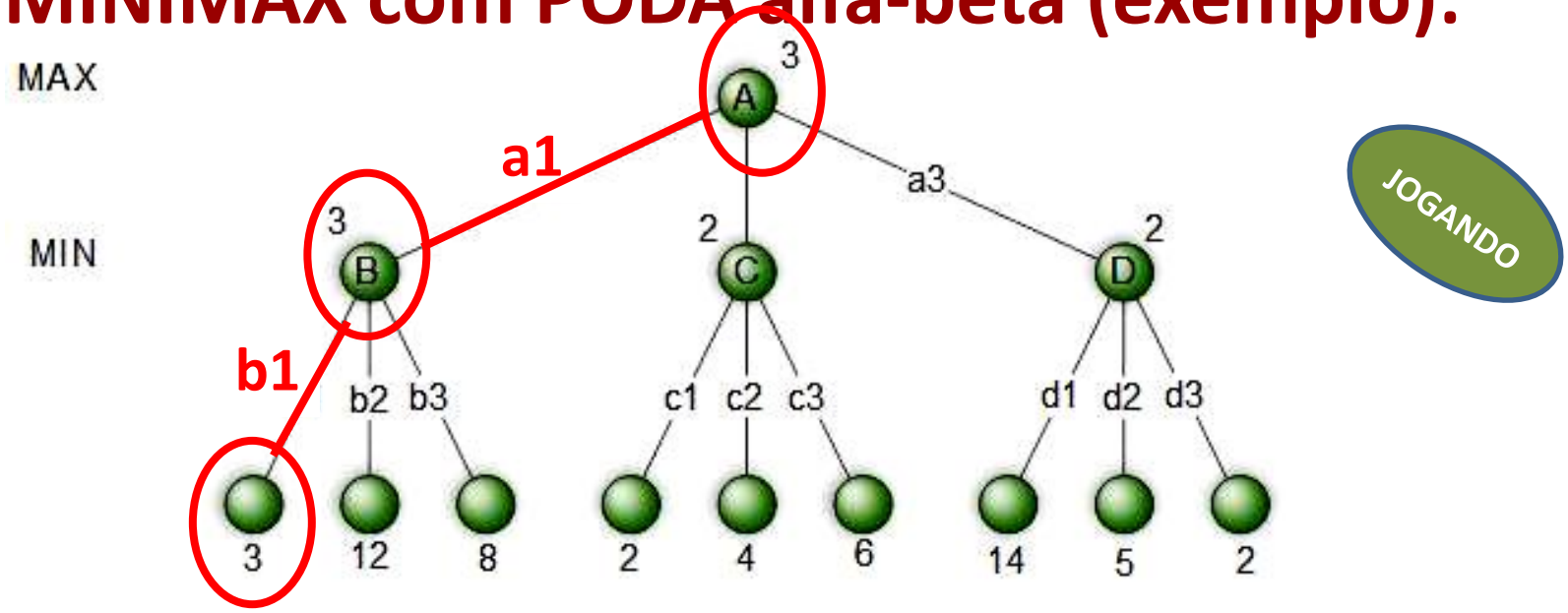
2.3 – MINIMAX com PODA alfa-beta (exemplo):



- Identifica-se a decisão MINIMAX na raiz, sendo a ação **a1** a escolha ótima para MAX, porque leva ao sucessor com o mais alto valor MINIMAX.

2 – Busca MINIMAX

2.3 – MINIMAX com PODA alfa-beta (exemplo):



- Assim pode-se identificar a decisão MINIMAX na raiz, a ação **a1** é a escolha ótima para MAX, porque leva ao sucessor com o mais alto valor MINIMAX.
- E a decisão MINIMAX em B, a ação **b1** é a escolha ótima para MIN, porque leva ao sucessor com o mais baixo valor MINIMAX.

Bibliografias

Obrigatórias:

1. RUSSELL, Stuart J; NORVIG, Peter. **Inteligência Artificial.** 2ª ed. Rio de Janeiro: Campus, 2004, Capítulo 6.

Bibliografias

Recomendadas:

1. http://www.tippychess.com/conteudo.asp?titulo=teoria_dos_jogos
2. http://www.tippychess.com/conteudo.asp?titulo=minimax
3. http://www.tippychess.com/conteudo.asp?titulo=poda_alfa_beta
4. http://centria.di.fct.unl.pt/~jja/iia/trabalhos/jogos/jogos.htm