Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
DCC014 - Inteligência Artificial

FÓRMULA PARA 24

Grupo 2:

DEOCLÉCIO PORFÍRIO FERREIRA FILHO	MAT201876043
IVANYLSON HONÓRIO GONÇALVES	MAT201776002
ÍGOR MARCHITO ZAMBONI	MAT201976020
MARCOS AQUINO ALMEIDA	MAT201276024

Orientadora: Profa.Dra. LUCIANA CONCEIÇÃO DIAS CAMPOS

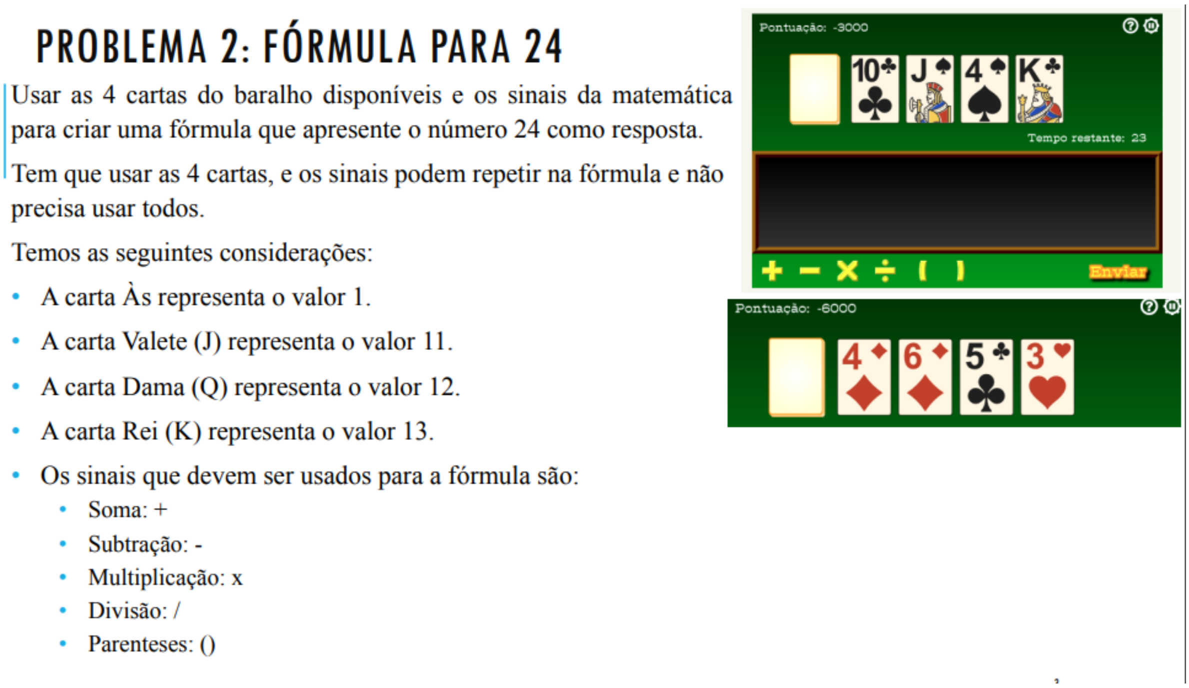
Relatório do trabalho da disciplina DCC014 - Inteligência Artificial, parte integrante da avaliação da mesma

Juiz de Fora
Janeiro de 2022

Sumário

Estudo de caso

- O Problema no computador



PROBLEMA 2: FÓRMULA PARA 24

Usar as 4 cartas do baralho disponíveis e os sinais da matemática para criar uma fórmula que apresente o número 24 como resposta.

Tem que usar as 4 cartas, e os sinais podem repetir na fórmula e não precisa usar todos.

Temos as seguintes considerações:

- A carta Às representa o valor 1.
- A carta Valete (J) representa o valor 11.
- A carta Dama (Q) representa o valor 12.
- A carta Rei (K) representa o valor 13.
- Os sinais que devem ser usados para a fórmula são:
  - Soma: +
  - Subtração: -
  - Multiplicação: x
  - Divisão: /
  - Parenteses: ()

Figura 1 - Imagem retirada [1].

- **Representação do espaço de estados**

❖ Pensando agora em uma representação que seja programável para o problema:
  ➢ Vamos considerar as posições das cartas como um vetor de inteiros com 4 posições, logo, as posições do vetor possuem índices que vão de 0 a 3, conforme a figura 2.

❖ Desta forma:
  ➢ O **estado inicial** do problema é dado por:

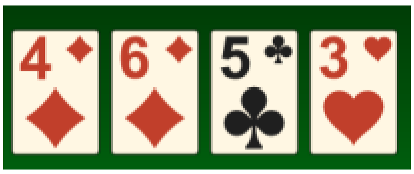


Figura 2 - Imagem retirada [1].

Vetor com 5 posições: 1ª posição contém parênteses e uma multiplicação ()*, as posições subsequentes possuem 4,6,5 e 3 respectivamente.

➢ O **estados finais (objetivos)** do problema é dado por:

01 Forma: (5 - 4 + 3) * 6 = 24
02 Forma: (5 + 3 - 4) * 6 = 24
03 Forma: 6 * (5 - 4 + 3) = 24
04 Forma: 6 * (5 + 3 - 4) = 24
05 Forma: 6 * (3 - 4 + 5) = 24

Vetor com 5 posições contendo na 1ª posição uma expressão matemática com as 4 cartas do vetor inicial e resultando no valor 24 enquanto as posições subsequentes estarão zeradas.

Regras (não de transição)

Dentro dos parênteses existirão duas expressões aritméticas simples distintas, ou seja, um + e um -. Desta forma, serão necessárias que três cartas estejam dentro dos parênteses também.

❖ Por último, precisamos definir as **regras de transição de um estado** para outro dentro do espaço de estados do problema.

Observe que atendendo às restrições:

- Aritmética básica significa apenas as 4 operações: +, -, x, /. Também parênteses (...) são permitidos para definir a ordem das operações.
- Use todos os quatro números do cartão, mas use cada número apenas uma vez.
- Você não precisa usar todas as quatro operações.

❖ Então as **regras de transição do espaço de estado** do problema são sempre baseados em regras das operações da aritmética :

➢ R1.1: O primeiro número dentro dos parênteses é o 4 (sua posição no vetor será zerada);

➢ R1.2: O primeiro número dentro dos parênteses é o 6 (sua posição no vetor será zerada);

➢ R1.3: O primeiro número dentro dos parênteses é o 5 (sua posição no vetor será zerada);

➢ R1.4: O primeiro número dentro dos parênteses é o 3 (sua posição no vetor será zerada);

- ➢ R2.1: Os sinais aritméticos simples dentro dos parênteses serão na ordem + e -;

- ➢ R2.2: Os sinais aritméticos simples dentro dos parênteses serão na ordem - e +;

- ➢ R3.1: Caso o valor do vetor na posição 1 não seja 0, o número entre os sinais dentro dos parênteses será o 4 (sua posição no vetor será zerada);

- ➢ R3.2 Caso o valor do vetor na posição 2 não seja 0, o número entre os sinais dentro dos parênteses será o 6 (sua posição no vetor será zerada);

- ➢ R3.3: Caso o valor do vetor na posição 3 não seja 0, o número entre os sinais dentro dos parênteses será o 5 (sua posição no vetor será zerada);

- ➢ R3.4: Caso o valor do vetor na posição 4 não seja 0, o número entre os sinais dentro dos parênteses será o 3 (sua posição no vetor será zerada);

- ➢ R4.1: Caso o valor do vetor na posição 1 não seja 0, o último número dentro dos parênteses será o 4 (sua posição no vetor será zerada);

- ➢ R4.2: Caso o valor do vetor na posição 2 não seja 0, o último número dentro dos parênteses será o 6 (sua posição no vetor será zerada);

- ➢ R4.3: Caso o valor do vetor na posição 3 não seja 0, o último número dentro dos parênteses será o 5 (sua posição no vetor será zerada);

- ➢ R4.4: Caso o valor do vetor na posição 4 não seja 0, o último número dentro dos parênteses será o 3 (sua posição no vetor será zerada);

- ➢ R5: A posição no vetor entre 1 e 4 que não possuir 0 como valor será zerada e o número contido nessa posição será colocado após a multiplicação na posição 0 do vetor.

SOLUÇÃO DO PROBLEMA

Exemplo:

Estado inicial: [()* | 4 | 6 | 5 | 3]
Aplicando R1.4: [(3)* | 4 | 6 | 5 | 0]
Aplicando R2.1: [(3+-)* | 4 | 6 | 5 | 0]
Aplicando R3.3: [(3+5-)* | 4 | 6 | 0 | 0]
Aplicando R4.1: [(3+5-4)* | 0 | 6 | 0 | 0]
Aplicando R5: [(3+5-4)*6 | 0 | 0 | 0 | 0]

Exemplo da primeira execução:
Estado inicial: [()* | 4 | 6 | 5 | 3]
Aplicando R1.4: [(3)* | 4 | 6 | 5 | 0]
Aplicando R2.2: [(3-+)* | 4 | 6 | 5 | 0]
Aplicando R3.3: [(3-5+)* | 4 | 6 | 0 | 0]
Aplicando R4.2: [(3-5+6)* | 4 | 0 | 0 | 0]
Aplicando R5: [(3-5+6)*4 | 0 | 0 | 0 | 0]
Como (3-5+6)*4 = 16, será um estado de impasse.

Busca Gulosa

<u>Heurística</u>: Valor mais aproximado do objetivo

Haja vista que a multiplicação é a última operação a ser feita (vide que as outras duas estão dentro dos parênteses), escolhe-se um número que seja divisível por 24 dentre as cartas (vide que as outras operações são básicas, se o número não for divisível por 24 ele precisaria ser multiplicado por um número fracionário, porém é impossível chegar a um número fracionário com duas operações básicas (+ e -) sobre 3 números inteiros não fracionários). Assim, essa divisão de 24 por uma das cartas geraria um valor objetivo a ser atingido através de uma soma e uma subtração com as cartas restantes.
Em caso de mais de uma carta ser divisível por 24 têm-se como critério de desempate usar a que gera o menor valor objetivo. Após isso, soma-se o número de cartas que ainda não foram utilizadas, conforme Figura 3. A partir do segundo nível, a carta a ser utilizada será subtraída do valor gerado pela divisão do primeiro nível e ao módulo dessa subtração será adicionado o **número de cartas restantes**. O caminho a ser tomado será, novamente, o de menor valor.

A heurística adotada para o problema é a árvore de busca, lista de abertos e fechados, da busca Gulosa (ou parte dela - o máximo que seja possível ainda avaliar os nós da árvore).

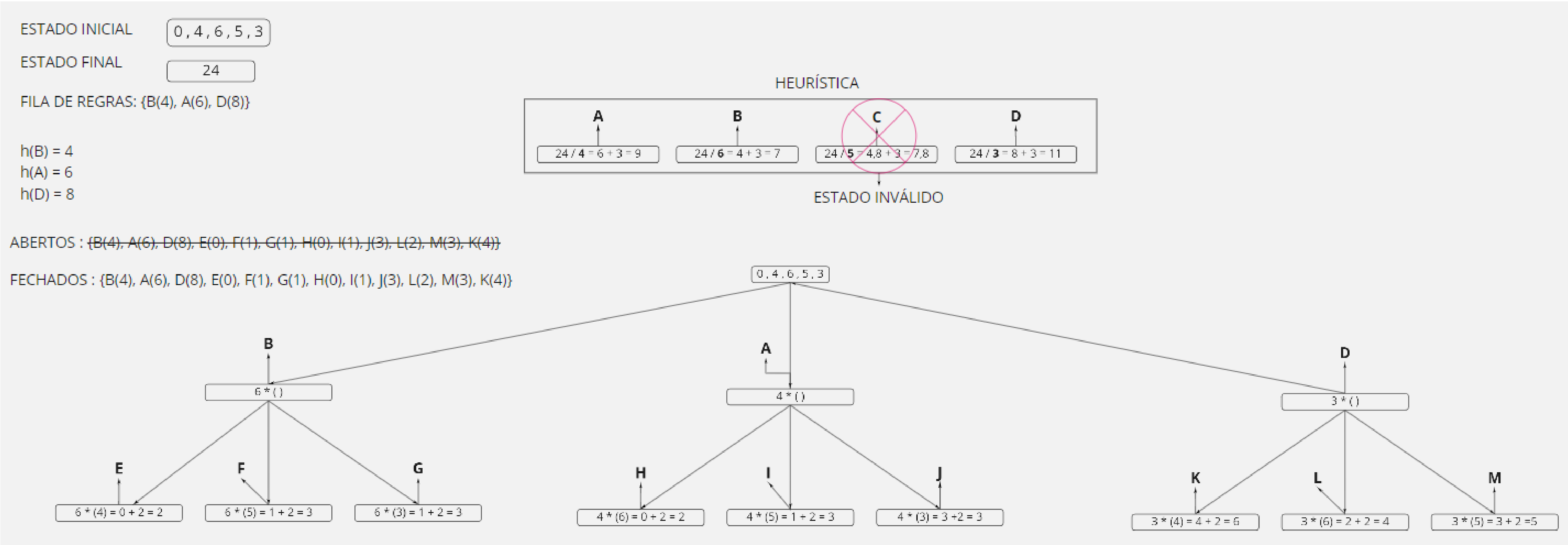


Figura 3 - Simulação da Busca Gulosa com Heurístic

Busca Ordenada

O custo real que fizeram para o problema é a árvore de busca, listas de abertos e fechados da busca Ordenada (ou parte dela para alguns problemas).

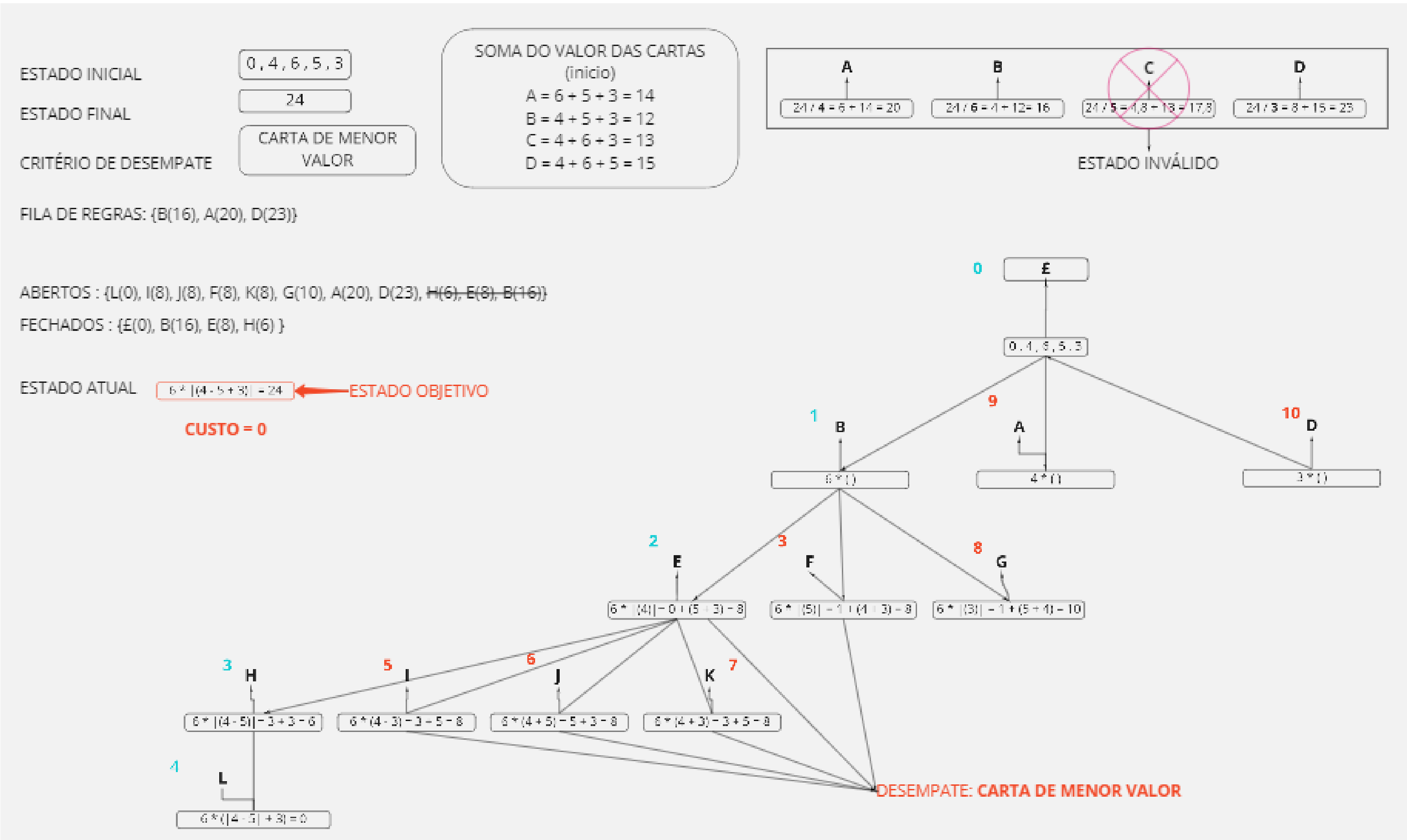


Figura 4 - Simulação da Busca Ordenada

O custo real associado com os filhos do primeiro nível é referente ao somatório das cartas restantes. Assim o filho que produz o menor valor desse somatório será o primeiro a ser visitado. A partir do segundo nível, além do somatório das cartas restantes, adiciona-se também um valor referente a um número objetivo (ver Heurística Gulosa) que é o valor da divisão de 24 pelo valor utilizado na fase anterior menos o valor que será usado no nível atual. Em caso de empate, o critério adotado é de seguir pelo caminho que utiliza a carta de menor valor. Por fim, um “atalho” foi implementado para se obter o resultado final mais eficientemente. Quando, no terceiro nível, existe uma operação de subtração que gera um valor negativo dentro dos parênteses, foi utilizado o conceito de módulos para gerar dois possíveis filhos. Um dos filhos irá conter a operação que gera o valor modular desse número negativo e o outro a operação normal. Desta forma, H teria dois filhos L e M onde M é o estado “normal” (4 - 5 + 3) e L é o estado “modular” (-4 + 5 + 3). Desempate ocorre quando a carta é de menor valor ou que vier primeiro.

Busca A*

O cálculo da função de avaliação de somar o custo real com a heurística e apresentar a busca A*, lista de abertos e fechados, e a árvore de busca (ou o máximo que conseguir fazer dessa árvore de forma que fique visível avaliar os nós).

Métodos não informados (Custo real + Custo Heurística) - Lista de abertos e fechados - (busca ordenada) - Menor função de avaliação

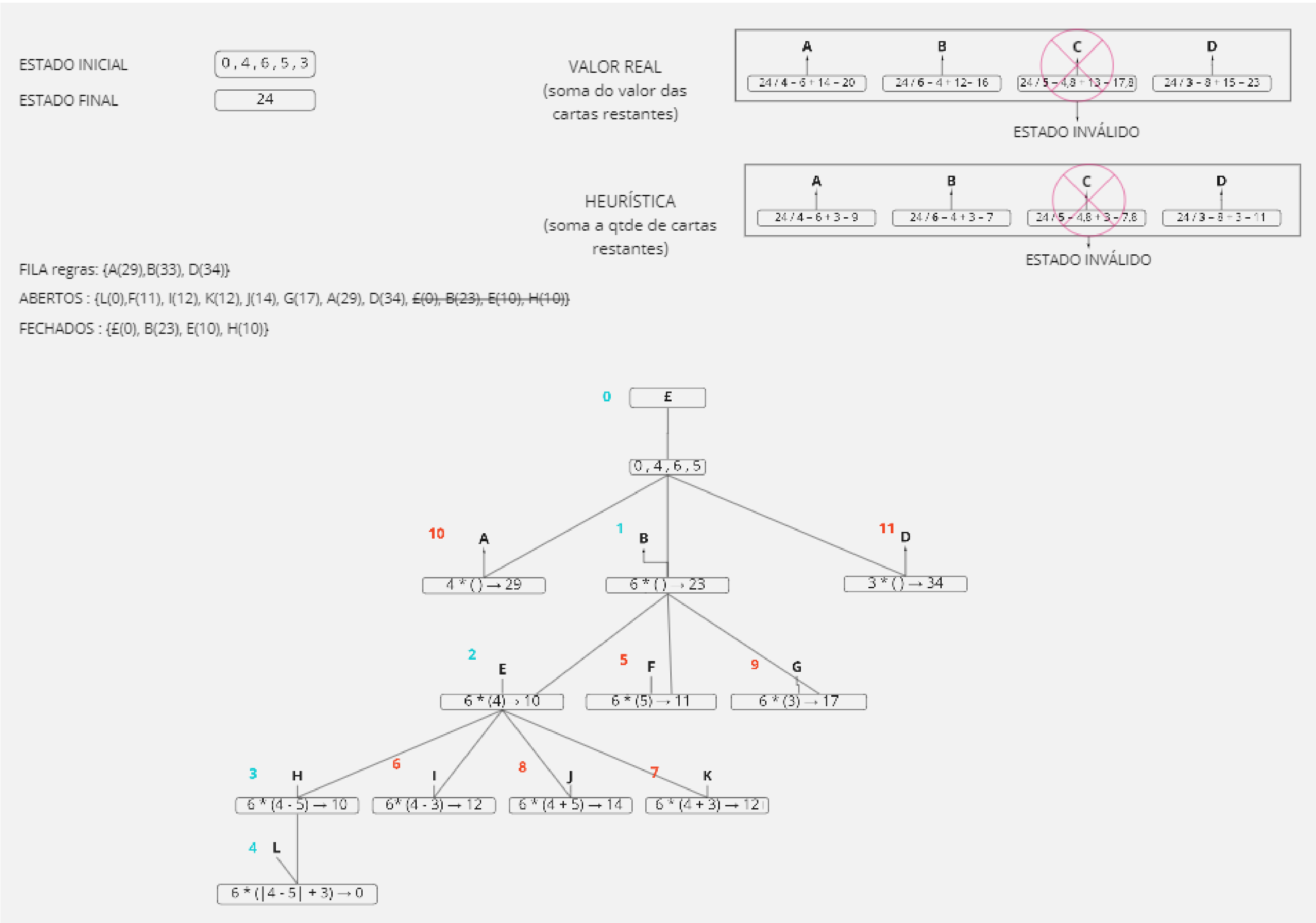


Figura 5 - Simulação da Busca A*

Pensamos da seguinte forma: É o somatório das duas formas anteriores .
A busca A* avalia os nós combinando o custo para alcançar cada nó n e o custo para ir do nó
n até o nó objetivo: f(n) = g(n) + h(n).

g(n) custo do caminho do nó inicial até o nó n.
h(n) custo estimado do caminho de custo mais baixo do nó n até o nó objetivo.
f(n) custo estimado da solução mais econômica passando por n.

ESTADO ATUAL: **£**

f(A) = g(A) + h(A) = 20 + 9 = 29

f(B) = g(B) + h(B) = 16 + 7 = 23

f(D) = g(D) + h(D) = 23 + 11 = 34

E
VALOR REAL: 6 * |(4)| = 0 + (5 + 3) = 8
HEURÍSTICA: 6 * (4) = 0 + 2 = 2

F
VALOR REAL: 4 * |(5)| = 1 + (6 + 3) = 10
HEURÍSTICA: 4 * (5) = 1 + 2 = 3

G
VALOR REAL: 4 * |(3)| = 1 + (5 + 6) = 12
HEURÍSTICA: 4 * (3) = 3 + 2 = 5

ESTADO ATUAL: **A**

f(E) = g(E) + h(E) = 8 + 2 = 10

f(F) = g(F) + h(F) = 10 + 3 = 11

f(G) = g(G) + h(G) = 12 + 5 = 17

H
VALOR REAL: 6 * |(4 - 5)| = 3 +
HEURÍSTICA: 6 * |(4 - 5)| = 3 +

I
VALOR REAL: 6 * (4 - 3) = 3 + 5
HEURÍSTICA: 6 * (4 - 3) = 3 + 1

J
VALOR REAL: 6 * (4 + 5) = 5 + 3
HEURÍSTICA: 6 * (4 + 5) = 5 + 1

K
VALOR REAL: 6 * (4 + 3) = 3 + 5
HEURÍSTICA: 6 * (4 + 3) = 3 +

ESTADO ATUAL: **E**

f(H) = g(H) + h(H) = 6 + 4 = 10

f(I) = g(I) + h(I) = 8 + 4 = 12

f(J) = g(J) + h(J) = 8 + 6 = 14

f(G) = g(G) + h(G) = 8 + 4 = 12

H
VALOR REAL: 6 * |(4 - 5 + 3)| = 0
HEURÍSTICA: 6 * |(4 - 5 + 3)| = 0

ESTADO ATUAL: **H** f(L) = g(L) + h(L) = 0 + 0 = 0

Figura 6 - Memória de cálculo da Busca A*

O algoritmo visita primeiro o nó com o menor valor de avaliação – f – a fim de encontrar a solução de custo mais baixo.

A busca A* será ótima (encontra a melhor solução) se a heurística $h(n)$ for admissível. Seja C^* o custo da solução ótima e suponha que um nó objetivo não-ótimo t apareça na lista de abertos. Então, sabemos que $h(t) = 0$ e como t não é ótimo, segue que: $f(t) = g(t) + h(t) = g(t) > C^*$.

Agora, considere um nó n na lista de abertos que pertence a um caminho de solução ótimo. Se $h(n)$ é admissível, temos que $f(n) = g(n) + h(n)\ C^*$.

Com isso, mostramos que $f(n)\ C^* < f(t)$ e assim temos que t não será visitado e A* deve retornar uma solução ótima.

Se $h(n)$ é consistente, então os valores de $f(n)$ ao longo do caminho são não-decrescentes: $f(m) = g(m) + h(m) = g(n) + C(n, m) + h(m)\ g(n) + h(n) = f(n)$, onde m é sucessor de n. Logo, só serão fechados pelo A* os vértices n tais que $f(n)\ C^*$ e os vértices na lista de fechados estão ordenados por valores crescentes de f.

Segue que a lista de abertos em A* fica ordenada por valores crescentes de $f(n)$. Assim, o primeiro nó objetivo visitado tem de ser uma solução ótima. Além disso, tem-se que A* não expande nenhum nó n cuja $f(n) > C^*$.

No problema que estamos trabalhando o nosso g(Valor Real - Busca Ordenada) e h(Resultado da Heurística - Busca Gulosa).

A Figura 6 mostra a memória de cálculo da análise em questão.
Desempate, quando necessário, é feito usando a carta de menor valor. Persistindo o empate, usa-se a ordem de entrada na lista de abertos.

Busca Minimax com Poda Alfa-Beta

Estrutura de busca Minimax para o problema é da seguinte forma:

- Nós de nível par são do tipo MAX e os nós de nível ímpar são do tipo MIN. Simular um jogo onde cada competidor (MAX, MIN) estão disputando achar o objetivo. Criar regras de utilidade para quando MAX ganhar, MIN ganhar ou der empate no caminho solução.
- A estratégia adotada pelo grupo: explicação do que foi feito para criar a árvore minimax, aplicar a poda Alfa-Beta.

A estratégia: A estratégia, além dos procedimentos padrões para de criar uma árvore minimax, está relacionada aos valores utilitários dos nós folhas. Os valores foram distribuídos de acordo com o valor objetivo já utilizado em outros métodos com uma penalidade de -24 sendo aplicada nos estados de impasse em que o multiplicador não dentro da função não é o 6.

Explicação: A ideia de se usar uma penalidade alta nos estados advindos dos nós A e D é que todas as folhas ali não serão um estado objetivo, assim garante-se que o valor que chegará até antes da raiz será menor que o de B, possibilitando assim a poda desses caminhos.

Regras: As regras seguem a mesma lógica de outras metodologias. Em caso do multiplicador gerar um número quebrado ao usado como divisor do 24 o estado passa a ser inválido. A única adição feita é a penalidade imposta sobre os estados de impasse caso o multiplicador não seja o número 6.

BUSCA MINIMAX

ESTADO INICIAL: 0 , 4 , 6 , 5 , 3

ESTADO FINAL: 24

CRITÉRIO DE DESEMPATE: CARTA DE MENOR VALOR OU A QUE ENTRAR PRIMEIRO

SOMA DO VALOR DAS CARTAS
(inicio)
A = 6 + 5 + 3 = 14
B = 4 + 5 + 3 = 12
C = 4 + 6 + 3 = 13
D = 4 + 6 + 5 = 15

v(£) = MAX{-22, 0, -22} = 0.

v(A) = MIM{-22, -20, 20} = -22.

v(B) = MIM{2, 0, 2} = 0.

v(D) = MIM{-19, -22, -21} = -22.

v(E) = MAX{2, -2, -2, 2} = 2.

v(F) = MAX{-2, 0, -2, 0} = 0.

v(G) = MAX{0, 2, 2, 0} = 2.

v(H) = MAX{-26, -22, -22, -26} = -22.

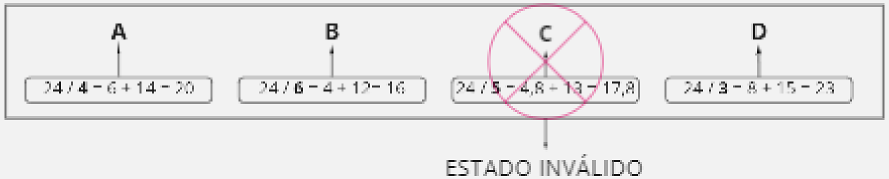
v(I1) = MAX{-26, -20, -20, -26} = -20.

v(I2) = MAX{-20, -22, -22, -20} = -20.

v(J) = MAX{-19, -21, -31, -31} = -19.

v(K) = MAX{-23, -22, -31, -31} = -22.

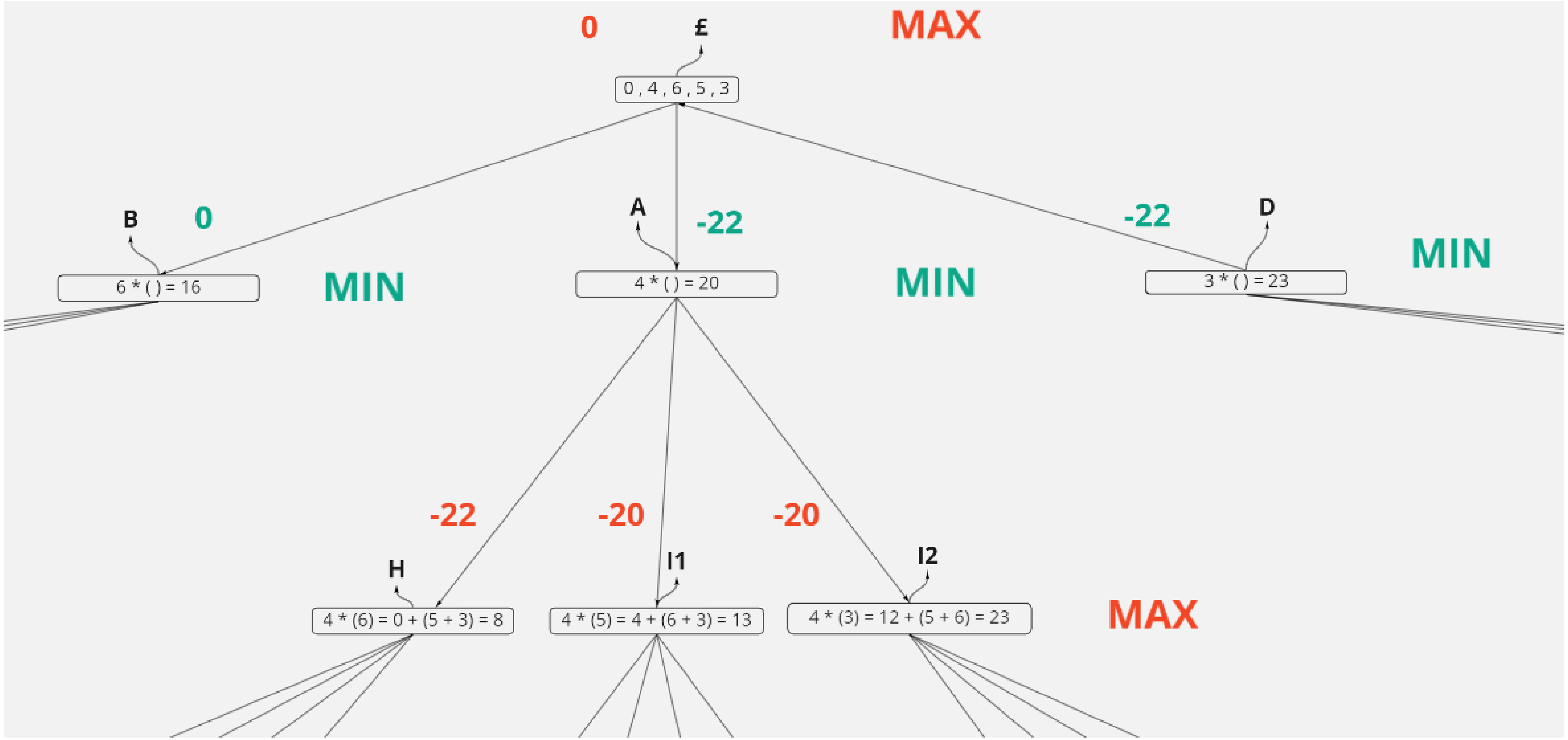
v(L) = MAX{-23, -21, -31, -31} = -21.



FILA DE GLOBAL INICIAL: {B(16), A(20), D(23)}

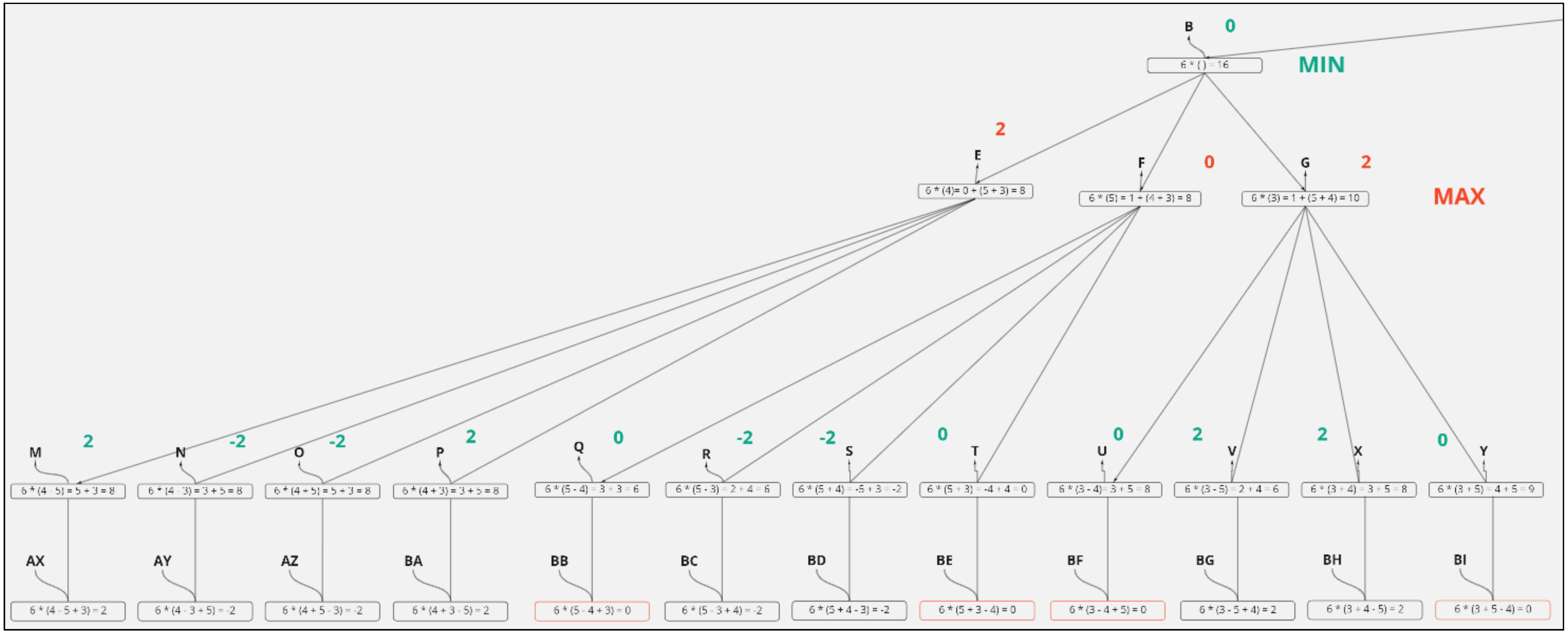
ESTADOS TERMINAS VÁLIDOS: {BB(0), BE(0), B(F), BI(0)}

JOGADA SOLUÇÃO: {£ → F → Q → BB}

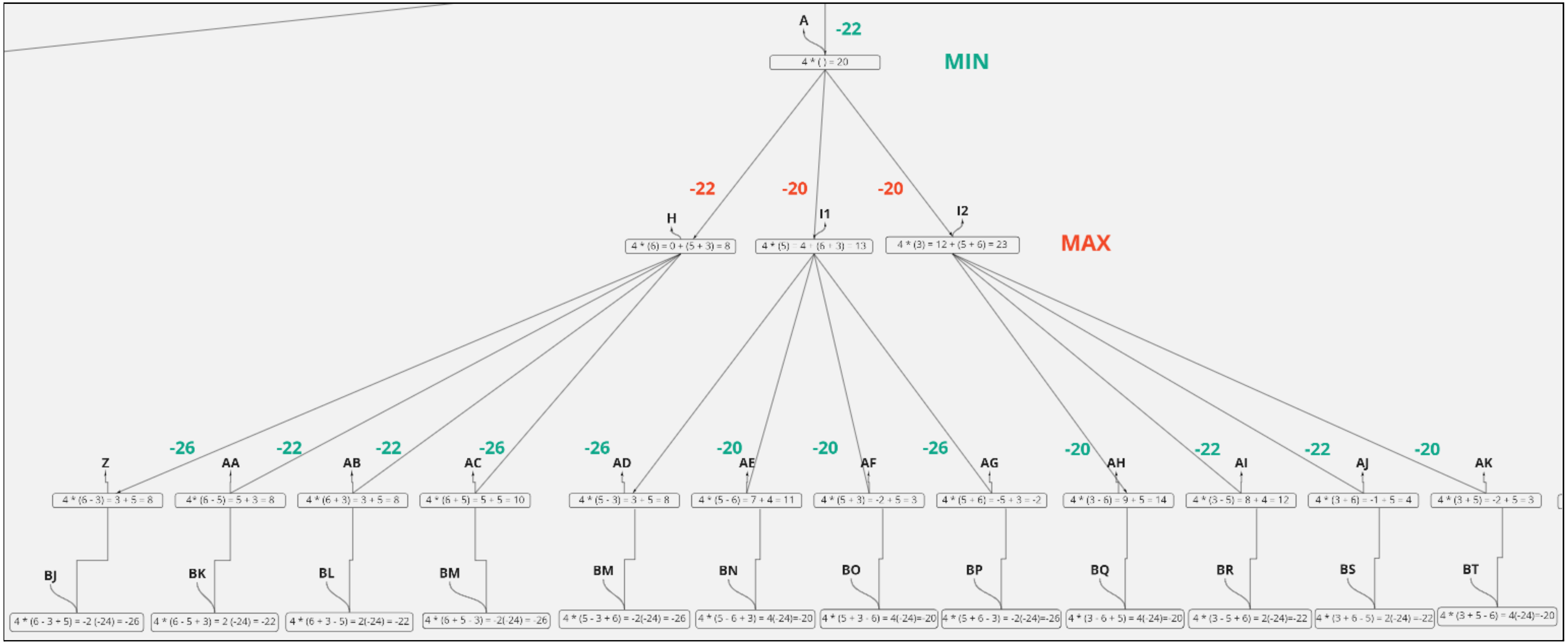


0
£
MAX
0 , 4 , 6 , 5 , 3
B
0
6 * ( ) = 16
MIN
A
-22
4 * ( ) = 20
MIN
-22
D
3 * ( ) = 23
MIN
-22
-20
-20
H
I1
I2
4 * (6) = 0 + (5 + 3) = 8
4 * (5) = 4 + (6 + 3) = 13
4 * (3) = 12 + (5 + 6) = 23
MAX

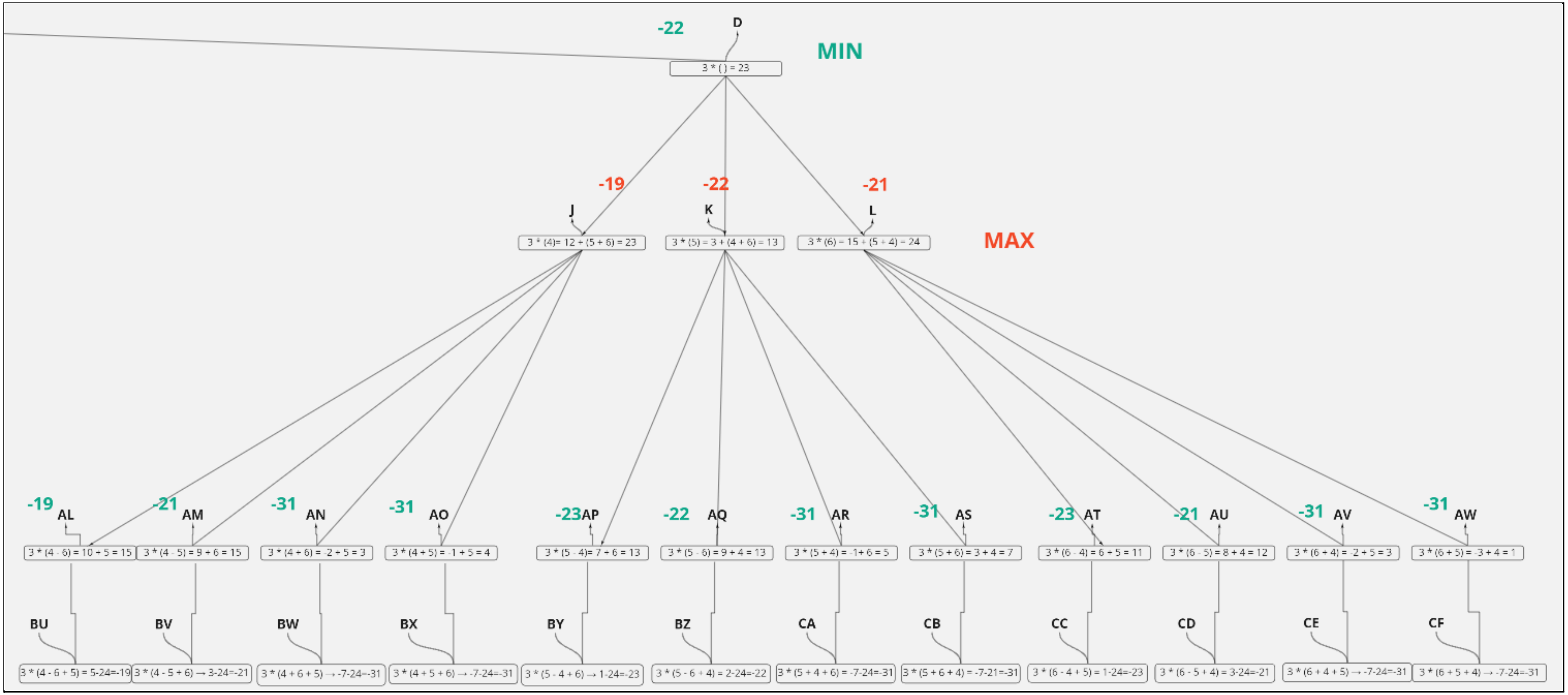
Lado esquerdo
Letra B



Centro
Letra E



Lado direito
Letra D



Referência

[1]https://ead.ufjf.br/pluginfile.php/1365799/mod_resource/content/1/Aula1%20-%20Introduc%CC%A7a%CC%83oProblemas.pdf

[2]https://ead.ufjf.br/pluginfile.php/1365800/mod_resource/content/1/Aula2%20-%20Espac%CC%A7o%20de%20Busca.pdf