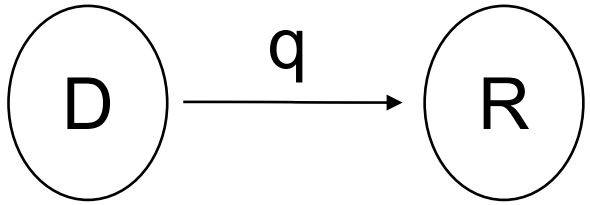
Inteligência Artificial

Busca e resolução de problemas

Teoria de Problemas

- Um problema pode ser definido como um objeto matemático P={D,R,q}, consistindo de dois conjuntos não vazios, D os dados e R os resultados possíveis, e de uma relação binária $q \subset DxR$, que consiste na condição que caracteriza uma solução satisfatória, associando a cada elemento do conjunto de dados a solução desejada.



Teoria de Problemas

- **Exemplo:** um problema de diagnóstico médico
  - O conjunto de dados disponíveis d∈d (observação da anamnese, sintomas, exames, etc.)
  - R é o conjunto de doenças possíveis
  - Solução satisfatória: encontrar o par (d,r) onde r ∈ r é o diagnóstico correto.
- A definição de um problema permite testar se um certo elemento é ou não solução, mas não guia na busca deste elemento.

Teoria de Problemas

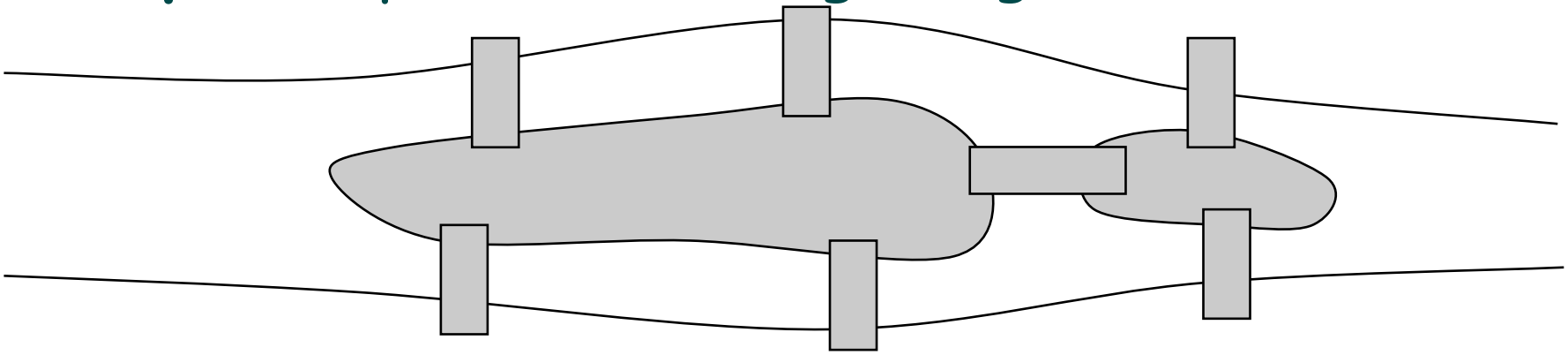
- Exemplo:
  - solução do problema da busca das raízes de um polinômio com coeficientes reais
  - consiste em associar a cada conjunto de coeficientes de um polinômio particular p(x) de grau n, n números complexos c_n de modo a satisfazer a condição de que o valor de p(x) fazendo x = c para todo n seja nulo.

Teoria de Problemas

- Historicamente a Ciência da Computação e, particularmente a Inteligência Artificial se interessa pela busca de problemas complexos que desafiam as fronteiras e os limites do conhecimento

Teoria de Problemas

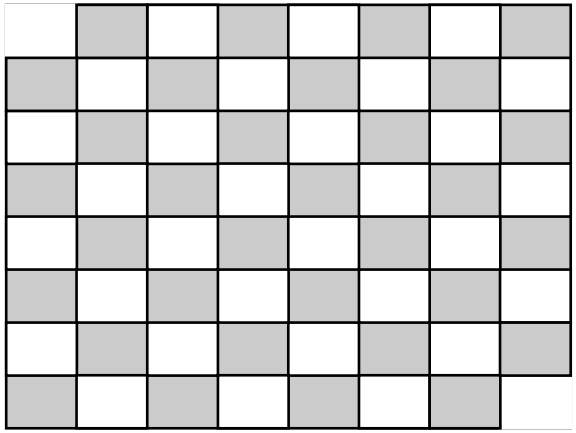
- **Exemplo:** As pontes de Königsberg

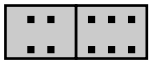


- Discutia-se nas ruas da cidade a possibilidade de atravessar todas as pontes sem repetir nenhuma. Havia-se tornado uma lenda popular a possibilidade da façanha quando Euler, em 1736, provou que não existia caminho que possibilitasse tais restrições.

Teoria de Problemas

- O tabuleiro de xadrez mutilado
- O problema consiste em cobrir totalmente o tabuleiro com pedras de jogo de dominó, considerando que uma pedra cobre exatamente duas casas contíguas do tabuleiro. A abstração que será feita nesse caso é a de não considerar a cor das casas. Sendo o estado inicial o tabuleiro vazio, tem 108 estados possíveis depois de ter colocado a primeira pedra de dominó. Pode-se verificar que por um tabuleiro de tamanho n o número de possibilidades para colocar a primeira pedra é 2n(n-1)-4.





Teoria de Problemas

Modos de definir uma FUNÇÃO PROBLEMA

1. Por ENUMERAÇÃO EXAUSTIVA
   - Fornece-se todos os conjuntos de pares (dado, resultado).
     - Ex.:Agenda de telefone.
2. DECLARATIVAMENTE
   - Definir declarativamente um problema é dar propriedades que devem ser satisfeitas pela solução do problema.
     - Ex.: O avô de alguém é aquela pessoa que é pai do pai da pessoa.
3. Por um PROGRAMA (um algoritmo)
   - Um programa de computador define a correspondência entre dados e resultados sempre que ele pára, conseguindo chegar a uma solução.
     - Ex.: Calcular uma equação
4. Por EXEMPLOS
   - O problema não completamente definido para todo valor de seus dados. Conhece-se para um subconjunto.
     - Ex.: Reconhecer um símbolo.

Teoria de Problemas

- **COMPUTABILIDADE**
  - Uma função é dita computável se é possível calcular seu valor para todos os elementos de seu domínio de definição.

- **DECIDIBILIDADE**
  - Um problema é parcialmente decidível se ele é decidível para um subconjunto próprio do seu conjunto de argumentos admissíveis

Teoria de Problemas

- **DECIDIBILIDADE**
  - A identificação dos problemas indecidíveis é uma das maiores aplicações da Teoria da Computação, pois permite demonstrar os limites teóricos dos computadores "reais"..
  - Uma família de problemas com respostas SIM/NÃO é dita indecidível se não existe algoritmo que termine com a resposta correta para todo problema da família.
    - Por exemplo, o problema: "dados dois números x e y, y é divisível por x?" é um problema de decisão. ou ainda: "Dado um número inteiro x, x é um número primo?"

Teoria de Problemas

- Se a computabilidade diz respeito à existência de solução para um problema, a complexidade se refere a quantidade de recursos necessários para resolvê-los.
- Um mesmo problema pode ter complexidade diferente, dependendo da técnica que se utiliza para resolvê-lo.

Teoria de Problemas

- Um algoritmo é dito de complexidade linear quando a quantidade de recursos para sua execução aumenta proporcionalmente à quantidade de dados envolvida no enunciado do problema.
- polinomial quando a quantidade de recursos para sua execução aumenta mais devagar do que algum polinômio função da quantidade de dados envolvida no enunciado do problema.

Teoria de Problemas

- **HEURÍSTICAS**
  - A palavra Heurística vem do grego, heurísko, significando "descubro" ou "acho". É um procedimento simplificador (embora não simplista) que, em face de questões difíceis envolve a substituição destas por outras de resolução mais fácil a fim de encontrar respostas viáveis, ainda que imperfeitas. Podendo tal procedimento ser tanto uma técnica deliberada de resolução de problemas, como uma operação de comportamento automática, intuitiva e inconsciente...
  - A capacidade heurística é uma característica humana que do lado positivo pode ser descrita como a arte de descobrir e inventar ou resolver problemas mediante a experiência (própria ou observada), somada à criatividade e ao pensamento lateral ou pensamento divergente. Como descrito acima, seja de forma deliberada ou não, heurísticas são procedimentos utilizados quando um problema a ser encarado é por demais complexo ou traz informações incompletas.

Teoria de Problemas

- **HEURÍSTICAS**
  - Conjunto de regras e métodos que conduzem à descoberta, à invenção e à resolução de problemas.
  - Uma regra, simplificação, ou aproximação que reduz ou limita a busca por soluções em domínios que são difíceis e pouco compreendidos".
  - Deste modo para um problema que não se conhece qual e' o melhor caminho em busca de uma solução, define-se uma função heurística que acredita-se levara a esta solução.
  - O papel das heurísticas - "boa solução"
  - Na IA as heurísticas são as "técnicas" que possibilitam tratar problemas NP-Completos e buscar algoritmos de ordem mínima para problemas polinomiais.

Resolução de Problemas

- Já vimos o que é um problema. Vamos agora buscar mecanismos para representá-lo e resolvê-lo, utilizando as técnicas da IA, ou seja, usando e manipulando CONHECIMENTO
- O Estudo do Conhecimento
- Resolução de Problemas por Busca
- Representação de Conhecimento

Resolução de Problemas

- O Estudo do Conhecimento
  - Uma teoria em IA consiste na especificação do conhecimento necessário a uma entidade cognitiva.
  - O que é uma ENTIDADE COGNITIVA?
    - É o "mecanismo" inteligente que permite entre outras atividades: solução de problemas, uso de linguagem, tomada de decisões, percepção, etc...
    - Na abordagem da IA Simbólica, a simulação da capacidade cognitiva requer conhecimento declarativo (definição declarativa da função) e algum tipo de raciocínio. Além disso, a evolução dos estados de conhecimento de um agente pode ser descrita em forma de linguagem (lógica ou natural).

Resolução de Problemas

- O conhecimento é central para a tarefa inteligente e na IAS, para que esta tarefa ocorra são necessários:
  - Uma BASE DE CONHECIMENTOS
  - Um MOTOR DE INFERÊNCIA
- **Base de Conhecimentos:**
  - Contém a informação específica sobre o domínio e será tão complexa quanto for o domínio e a capacidade cognitiva a ser simulada.
- **Motor de Inferência:**
  - Mecanismo que manipula a Base de Conhecimentos e gera novas conhecimentos.

Teoria dos grafos

- Enunciada no início do século XVIII por Leonhard Euler para resolver o problema das “pontes de Köninsberg”
- Conjunto de nós e arcos representando respectivamente estados e seus relacionamentos ou operadores de transição.
  - Conceitos importantes:
    - Grafos direcionados (arcos direcionais)
    - Grafos rotulados (descritores de cada nó e/ou arcos)
    - Caminhos (sequência de nos através de arcos sucessivos)
    - Grafos radicados (um único nó raiz com caminhos para todos os nós)
    - Árvores (dois nós tem um só caminho entre eles)
    - Ciclo ou circuito (caminho que começa e acaba com o mesmo vértice)

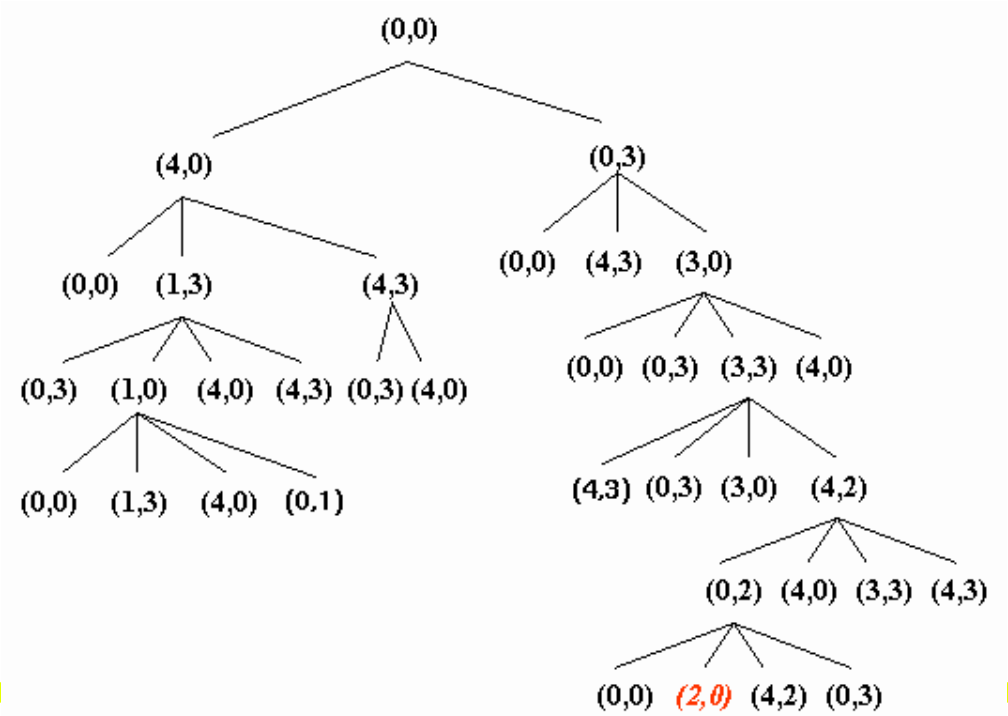
  (http://pt.wikipedia.org/wiki/Teoria_dos_grafos)

Espaço de estados

- Representando-se um problema como um grafo de espaço de estados pode se utilizar este formalismo para analisar a estrutura e a complexidade de um problema, e os procedimentos de busca necessários para resolve-lo
- Um Espaço de Estados é a representação em forma de grafo de todos os estados que podemos produzir a partir do estado inicial.
- A busca é o procedimento que consiste em percorrer esse espaço até achar o estado desejado

Espaço de estados

- Exemplo: Problema dos baldes de água: dados dois baldes, um com capacidade de 3 litros e outro com 4 litros, ambos sem marcações, encontrar um procedimento que permita que ao final o balde de 4 litros fique com 2 litros de água
- Uma representação da sequência de estados que podemos produzir a partir do estado inicial é mostrada abaixo.

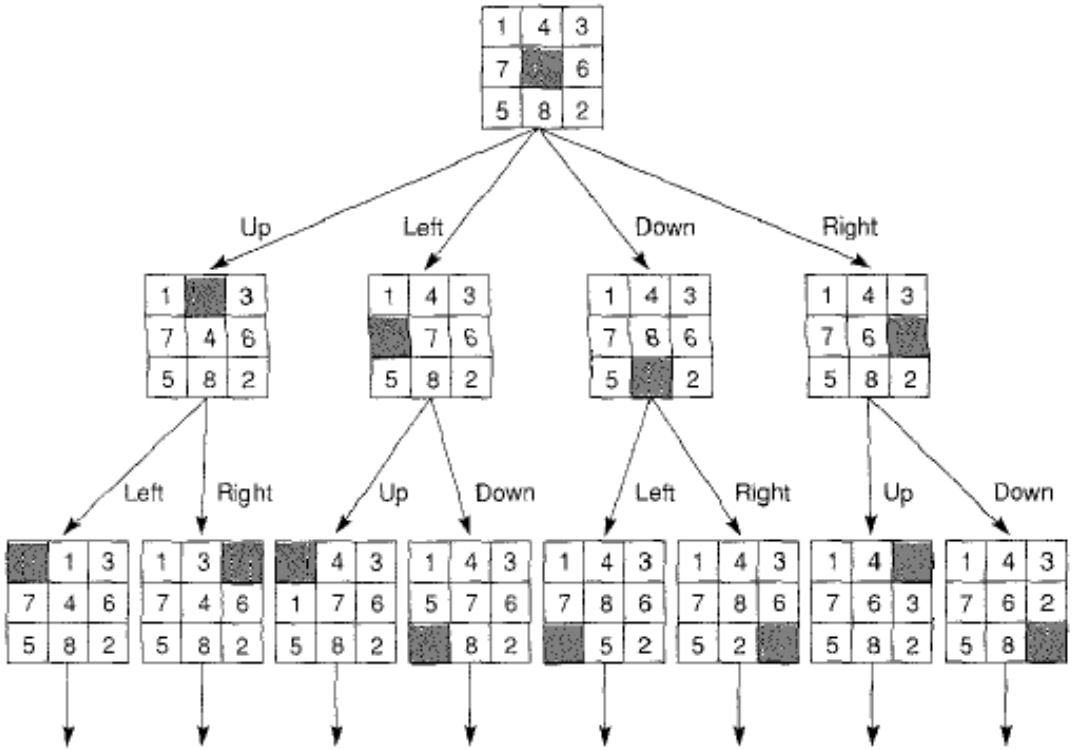


Espaço de estados

- Um espaço de estado é representado por uma quadra [N,A,S,DO] onde:
  - N é o conjunto de nós ou estados do grafo, correspondente aos estados de um processo de solução de problema
  - A é o conjunto de arcos entre os nós que correspondem aos passos de um processo de solução de problema
  - S é um subconjunto não-vazio de N que contém o(s) estado(s) inicial(ais) do problema
  - DO é um subconjunto não vazio de N que contém o(s) estado(s) objetivo(s) do problema, que podem ser descritos usando:
    - Uma propriedade mensurável dos estados encontrados na busca
    - Uma propriedade do caminho desenvolvido na busca (custos de transição)

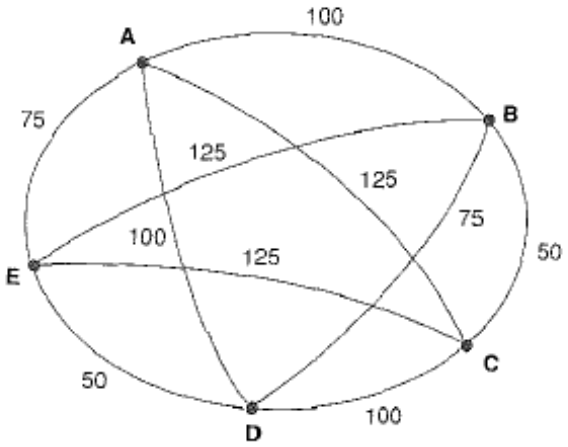
Jogo dos 8

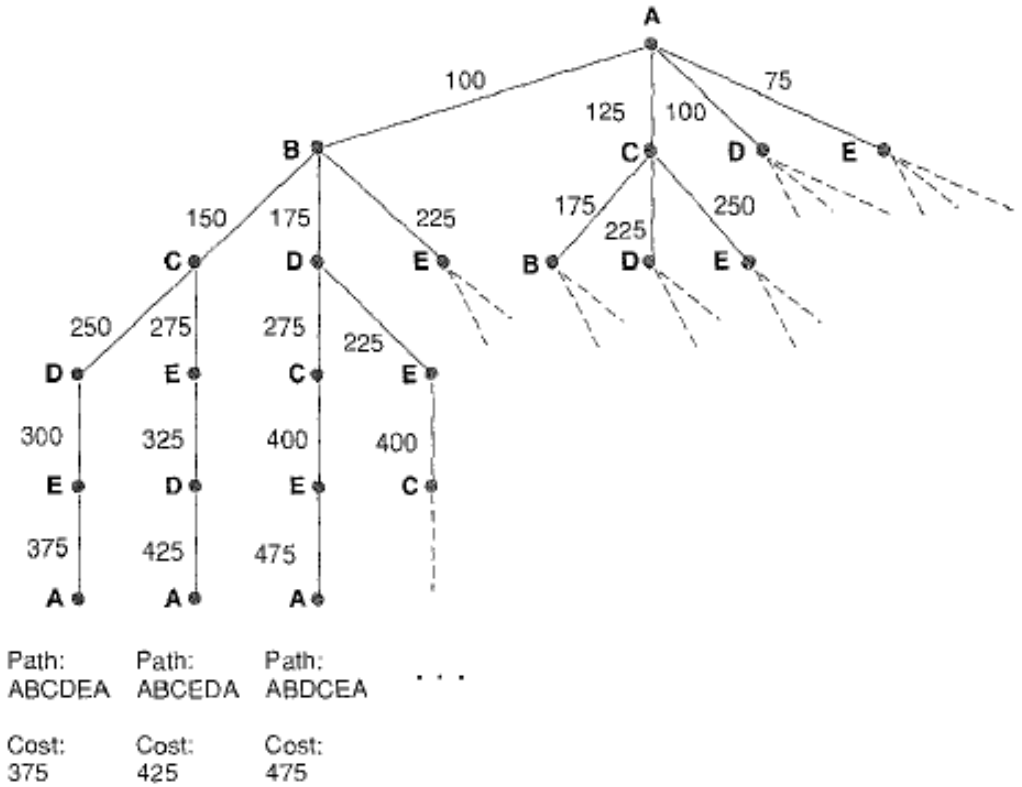
- Espaço de estados do jogo dos 8



O Caixeiro Viajante

- Espaço de estados do problema do caixeiro viajante que tenha que visitar cinco cidades e depois retornar para casa pelo caminho mais curto





- A complexidade da busca exaustiva é (N – 1) !

Exemplos

- Jogo da velha
  - Pode gerar 9! (362.880) caminhos diferentes
- Jogo de xadrez
  - 10^{120} caminhos possíveis
- Jogo de damas
  - 10^{40} caminhos
- Jogo dos 15
  - 16!
- Jogo dos 8
  - 9!
- Problema do caixeiro viajante
  - (N-1)!

Métodos de Busca

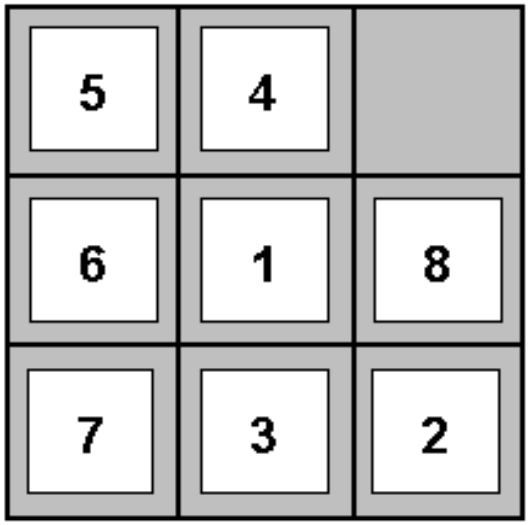
- A maioria dos problemas interessantes de IA não dispõe de soluções algorítmicas de baixa complexidade. Porém:
  - São solucionáveis por seres humanos e, neste caso, sua solução está associada à "inteligência";
  - Formam classes de complexidade variável existindo desde pequenos problemas triviais (jogo da velha) até instâncias extremamente complexas (xadrez);
  - São problemas de conhecimento total, isto é, tudo o que é necessário para solucioná-los é conhecido, o que facilita sua formalização.
  - Suas soluções têm a forma de uma seqüência de situações legais e as maneiras de passar de uma situação para outra são em número finito e conhecidas.
- Diante da falta de solução algorítmica viável, o único método de solução possível é a BUSCA.

Métodos de Busca

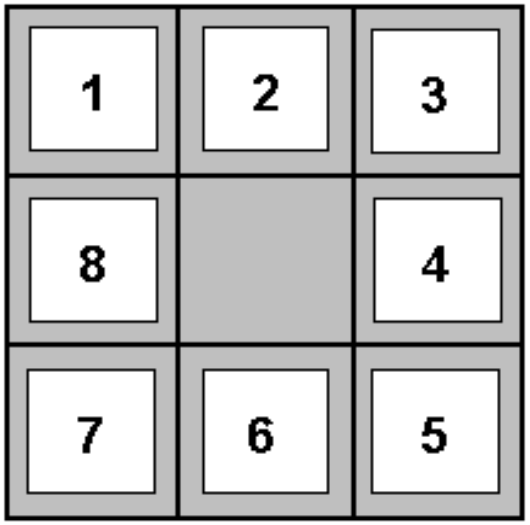
- Agentes de Resolução de Problemas
- Decidem o que fazer pela busca de ações que levem a estados desejáveis
  - estado inicial
  - operadores
  - teste de meta
  - função de custo de caminho
- Desempenho da Busca
  - Encontra uma solução?
  - É uma boa solução?
    - Custo do caminho
  - Qual o custo da busca?
    - Tempo e memória
  - Custo total=Custo da busca + custo do caminho

Métodos de Busca

- Exemplo: Jogo do Oito



Start State



Goal State

- **Estados:** local de cada uma das peças e do espaço
- **Operadores:** mover o espaço para cima, para baixo, esquerda ou direita.
- **Teste de meta:** dado na figura
- **Custo do caminho:** 1 para cada movimento

Métodos de Busca

- Estratégias de Busca
  - Critérios
    - Completude
    - Complexidade de Tempo
    - Complexidade de Espaço
    - Otimização
  - Métodos
    - Busca Cega - Não existe informação
    - Busca Heurística - Faz uso de informação

Métodos de Busca

- Método:
  - Geração dinâmica de uma árvore representando os estados alcançáveis a partir de um estado.
  - A seleção do estado a ser expandido é determinado pela estratégia de controle.
  - O processo pára quando o estado designado por um nodo folha corresponde ao objetivo.

Métodos de Busca

- Estruturas necessárias :
  - Nodo:
    - • Estado
    - • Nodo pai
    - • Operador utilizado para produzir o nodo
    - • Profundidade
    - • Custo do caminho até o nodo
  - Fila:
    - Contém os nodos produzidos que estão esperando para ser expandidos. Esses nodos constituem a
    - fronteira da busca.

Métodos de Busca

- Algoritmo geral de busca :

```
nodos <-- CRIAR-FILA(estado-inicial)
loop
    se nodos é vazio retorna falha
    nodo <-- TIRAR-PRIMEIRO(nodos)
    se TESTE-SUCESSO(nodo) tem sucesso
        retorna nodo
    novos-nodos <-- EXPANDIR(nodo)
    nodos <-- ACRESCENTAR-NA-FILA(nodos,novos-
    nodos)
fim
```

Estratégias

- Busca guiada por dados
  - encadeamento progressivo
    - parte dos fatos fornecidos e um conjunto de movimentos válidos em busca do objetivo
- Busca guiada por objetivos
  - encadeamento regressivo
    - parte do objetivo desejado, verifica que regras podem gerar este objetivo estabelecendo condições que se tornam novos subobjetivos até chegar nos fatos verificados
- Exemplo comparativo: Eu sou descendente de Papai Noel?
  - Para trás? (numero de pais)$^{\text{numero de ancestrais}}$
  - Para frente (numero de filhos)$^{\text{numero de ancestrais}}$

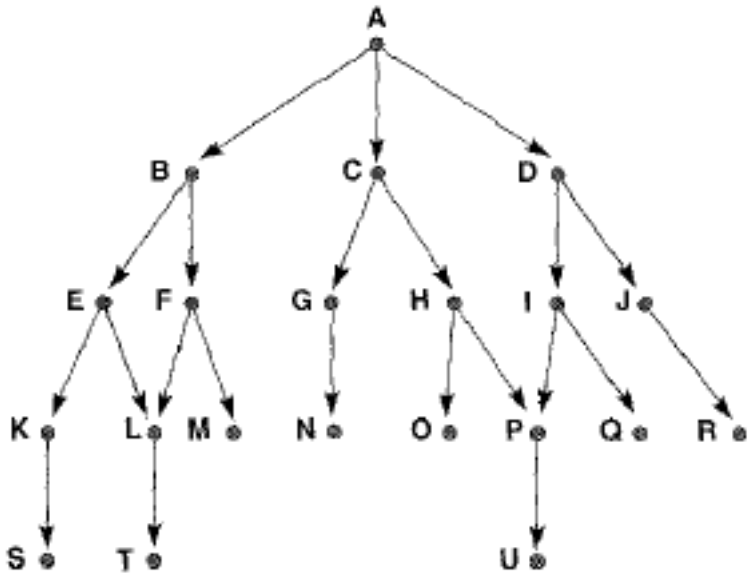
Busca Cega (Blind Search ou Uninformed Search)

- Uma estratégia de busca é dita cega se ela não leva em conta informações específicas sobre o problema a ser resolvido.

- Tipos de Busca Cega
  - Busca em largura
  - Busca pelo custo uniforme
  - Busca em profundidade
  - Busca em profundidade limitada
  - Busca por aprofundamento iterativo
  - Busca bidirecional

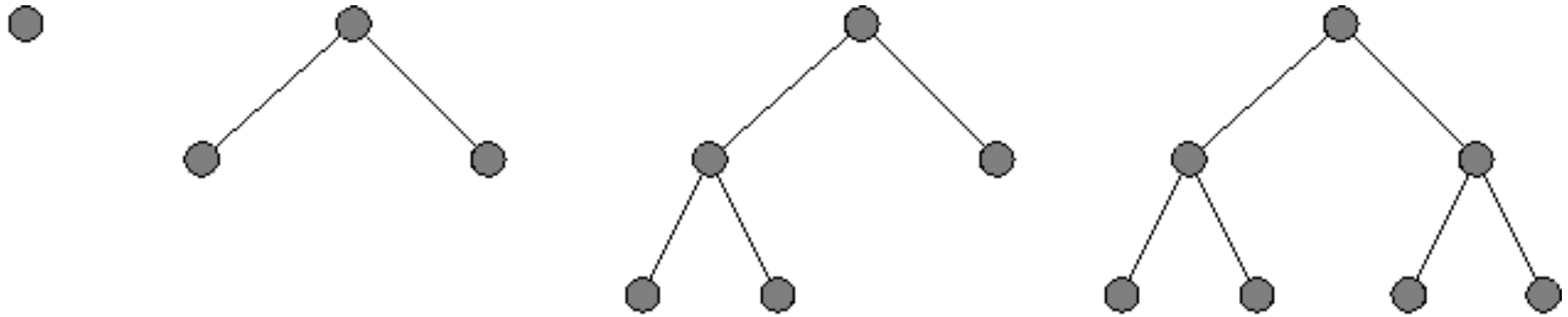
Busca em largura e em profundidade

- No grafo abaixo, a busca em profundidade examina os estados na ordem: A,B,E,K,S,L,T,F,M,C,G,N,H,O,P,U,D,I,Q,J,R
- Na busca em amplitude a ordem é: A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U



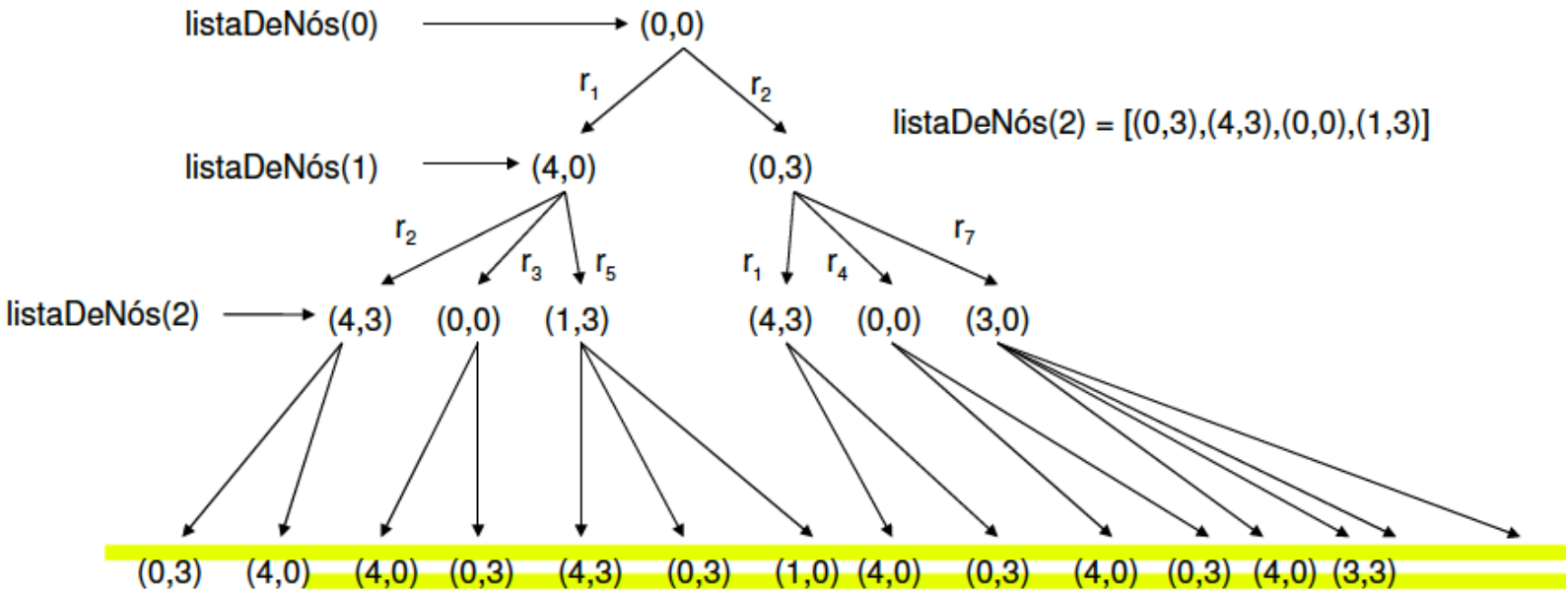
Busca em Largura (Amplitude)

- Consiste em construir uma árvore de estados a partir do estado inicial, aplicando a cada momento, todas as regras possíveis aos estados do nível mais baixo, gerando todos os estados sucessores de cada um destes estados. Assim, cada nível da árvore é completamente construído antes de qualquer nodo do próximo nível seja adicionado à árvore

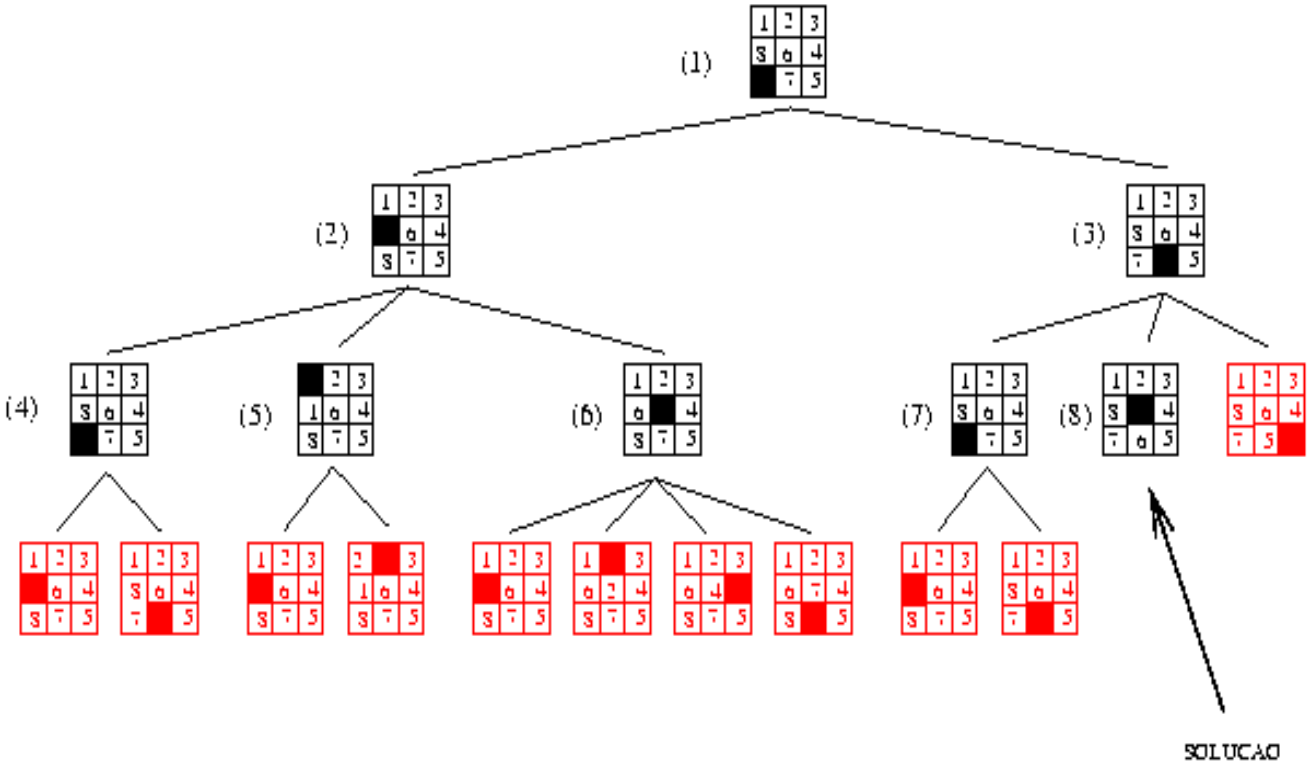


Busca em Largura (Amplitude)

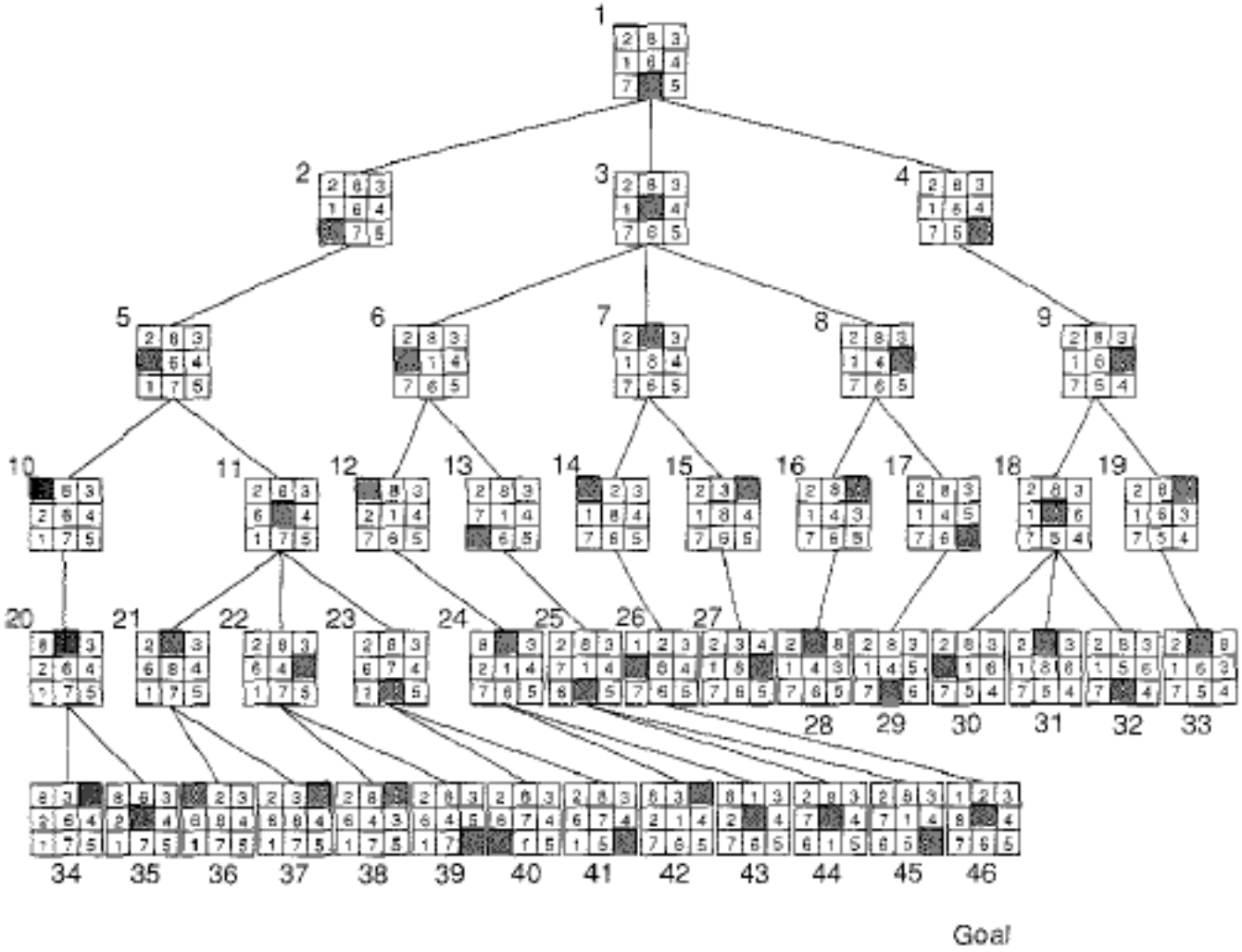
- Exemplo: Um balde de 4 litros e um balde de 3 litros. Inicialmente vazios.
  - Estado Final: um dos baldes com 2 litros de água.



Busca em Largura (Amplitude)



Busca em Largura (Amplitude)



Busca em Largura (Amplitude)

- Características: Completa e Ótima
  - Se existe solução, esta será encontrada;
  - A solução encontrada primeiro será a de menor profundidade.
- Análise de Complexidade – Tempo e Memória
  - Seja um fator de ramificação b.
  - Nível 0: 1 nó
  - Nível 1: b nós
  - Nível 2: b^2 nós
  - Nível 3: b^3 nós
  - Nível d (solução) b^d nós

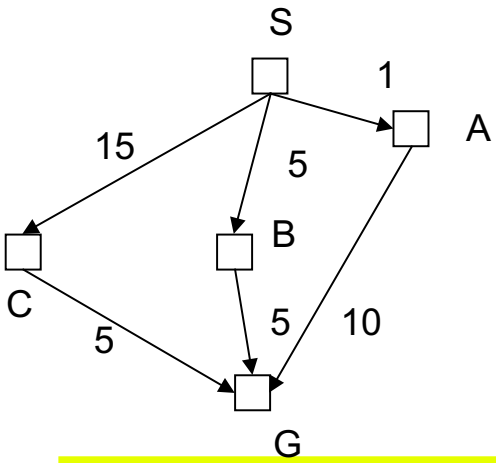
Busca em Largura (Amplitude)

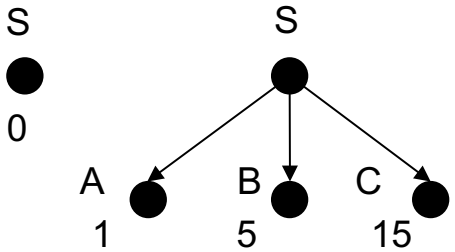
- Análise de Complexidade – Tempo e Memória

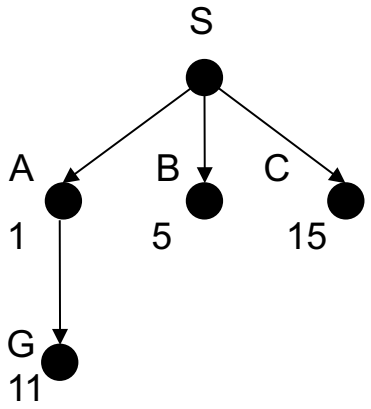
Depth	Nodes	Time	Memory
0	1	1 millisecond	100 bytes
2	111	.1 seconds	11 kilobytes
4	11,111	11 seconds	1 megabyte
6	10^6	18 minutes	111 megabytes
8	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabyte
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11,111 terabytes

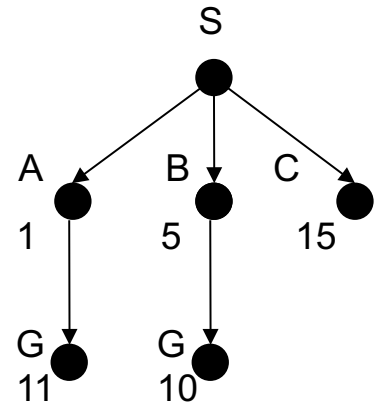
Método do Custo Uniforme

- Supondo que exista um "custo do caminho" associado a cada nó percorrido e que se deseje achar o caminho de custo mínimo.
- Neste caso, o algoritmo anterior é modificado para expandir primeiro o nó de menor custo.
- Exemplo: Problema de Rota entre S e G



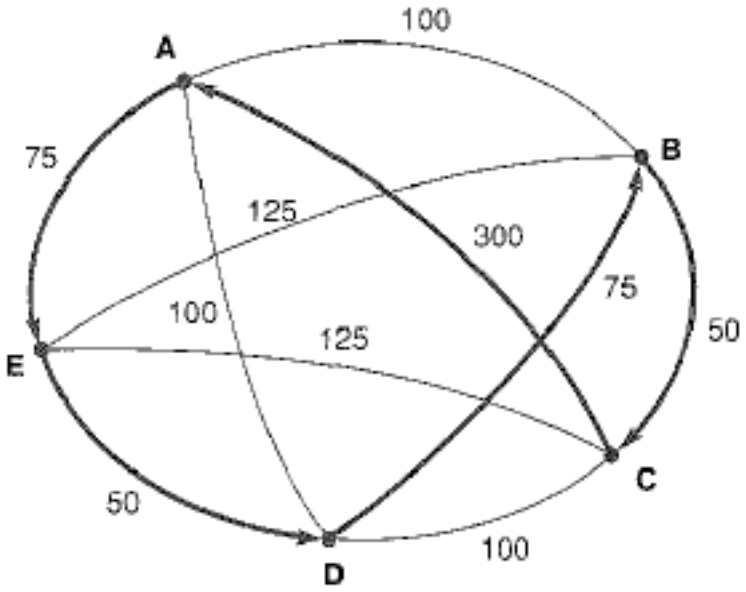






Método do Custo Uniforme

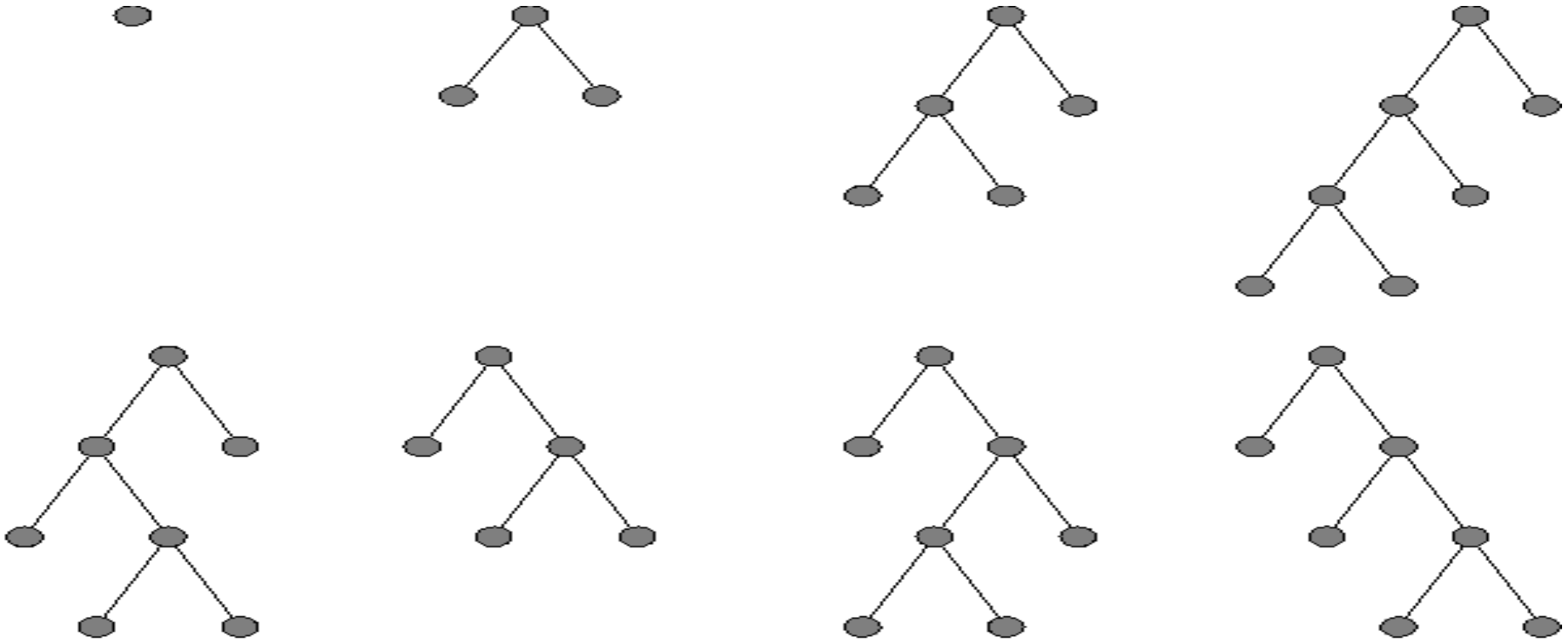
- Problema do caixeiro viajante com o caminho do vizinho mais próximo indicado em negrito (A,E,D,B,C,A) com custo total 550



G 11 G 10

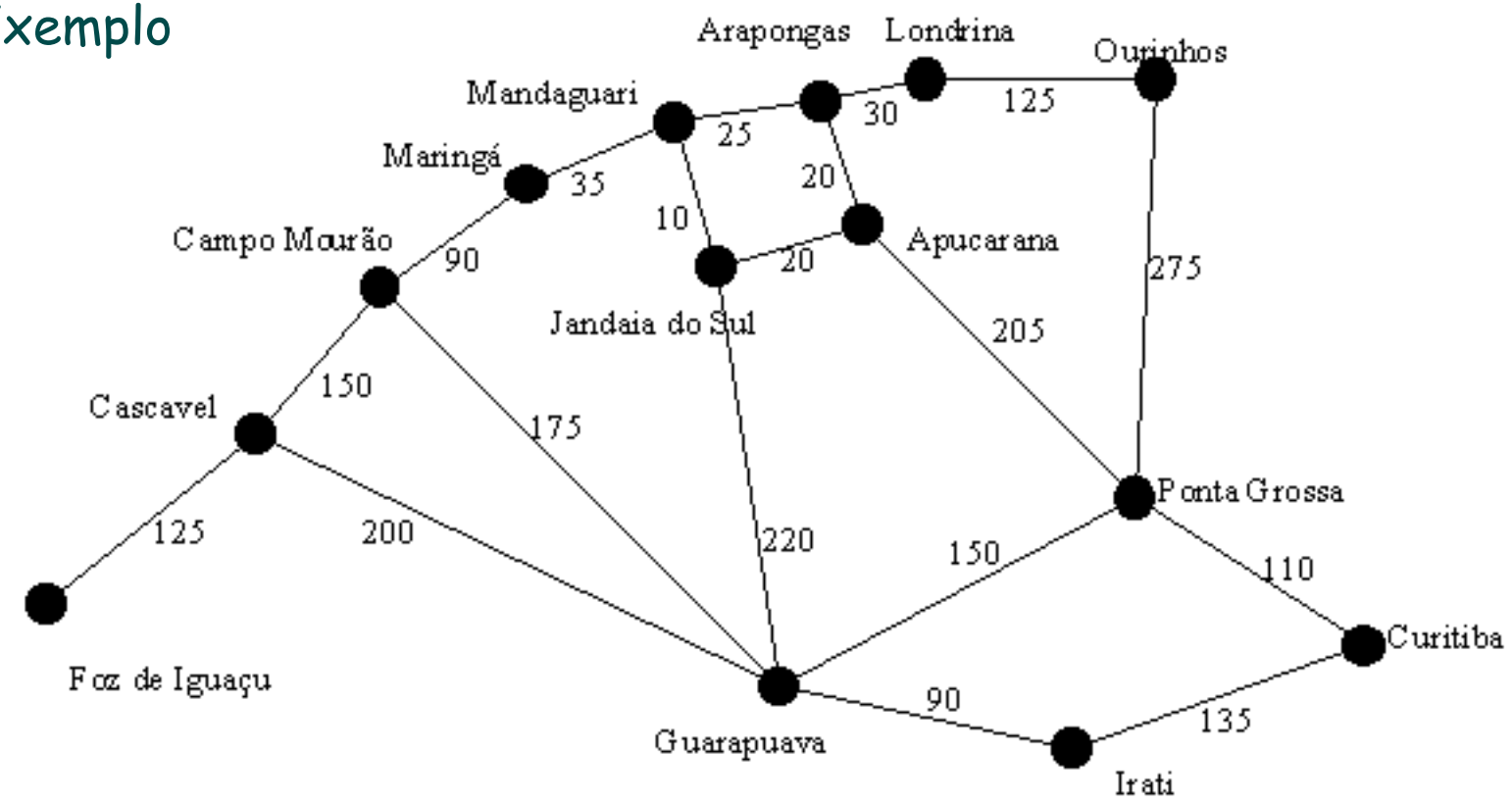
Busca em Profundidade

- Procurar explorar completamente cada ramo da árvore antes de tentar o ramo vizinho.



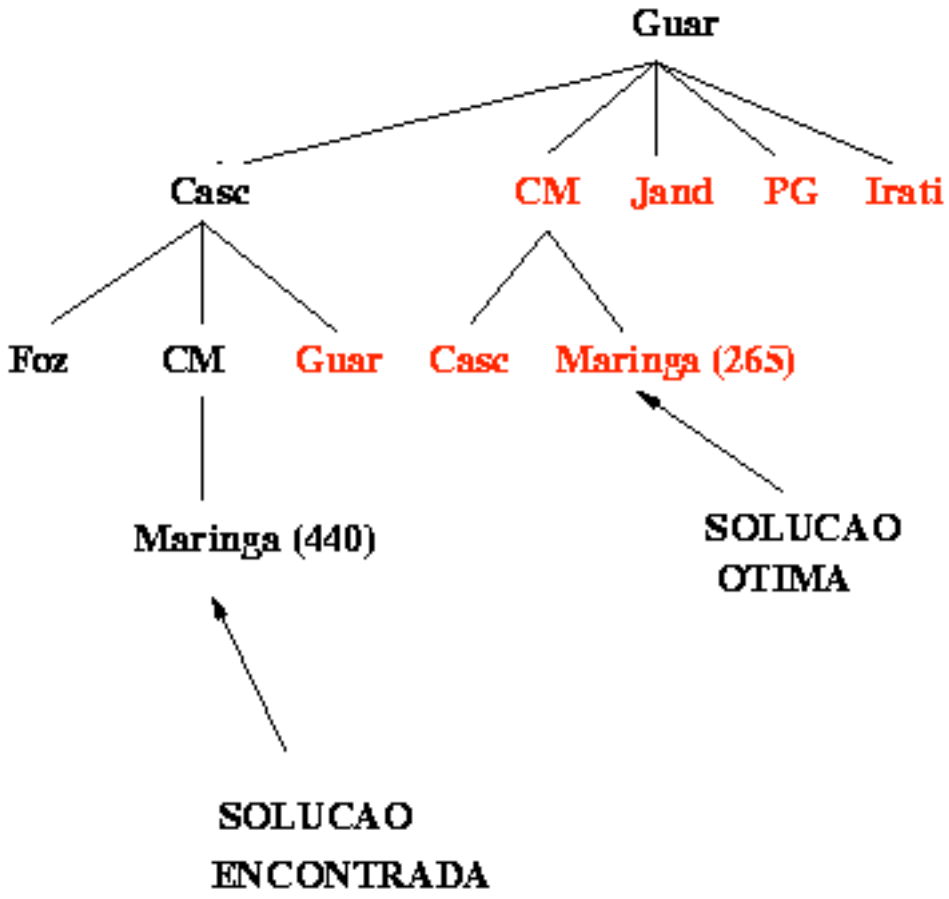
Busca em Profundidade

- Exemplo



Busca em Profundidade

- Exemplo



Busca em Profundidade

- Características: Não é Completa e Não é Ótima
  - Se admitir estados repetidos ou um nível máximo de profundidade, pode nunca encontrar a solução.
  - A solução encontrada primeiro poderá não ser a de menor profundidade.
  - O algoritmo não encontra necessariamente a solução mais próxima, mas pode ser MAIS EFICIENTE se o problema possui um grande número de soluções ou se a maioria dos caminhos pode levar a uma solução.

- Análise de Complexidade – Tempo e Memória
  - Seja m a profundidade máxima e um fator de ramificação b.
  - Tempo: b^m
  - Memória: b.m

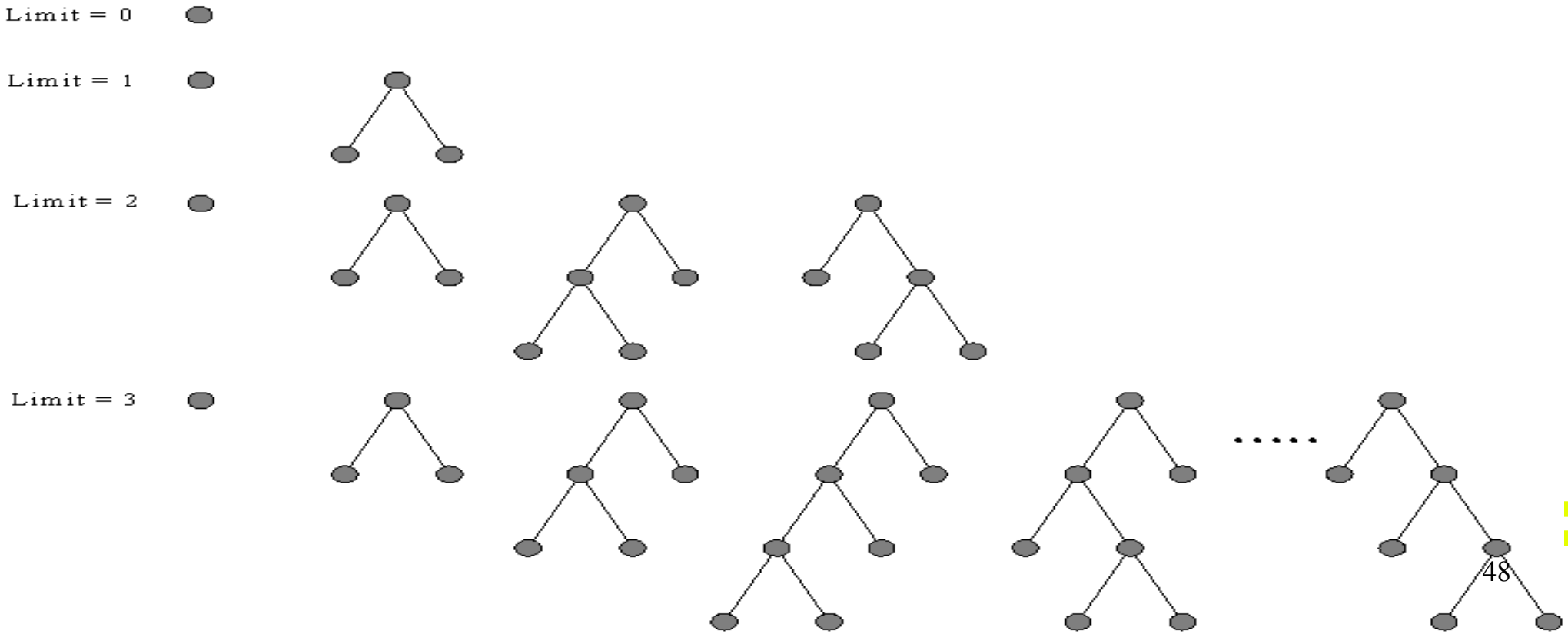
Variações na Busca em Profundidade

- Busca com profundidade limitada:
  - Escolhe-se um valor limite de profundidade que a busca não pode ultrapassar. Isso vai dar certo somente se pudermos confiar que a solução encontra-se dentro desse limite. Se não escolhermos o bom valor de limite de profundidade, pode não retornar uma solução. Esse tipo de busca resolve o problema da incompletude mas ainda é possível que seja retornada uma solução não ótima.
- Busca em profundidade iterativa:
  - Executa-se primeiro uma busca primeiro em profundidade com limite de profundidade 0, depois em largura. Se não encontramos uma solução, repetimos com limite de profundidade 1, 2, 3 e assim por diante até achar uma solução.

Busca Cega

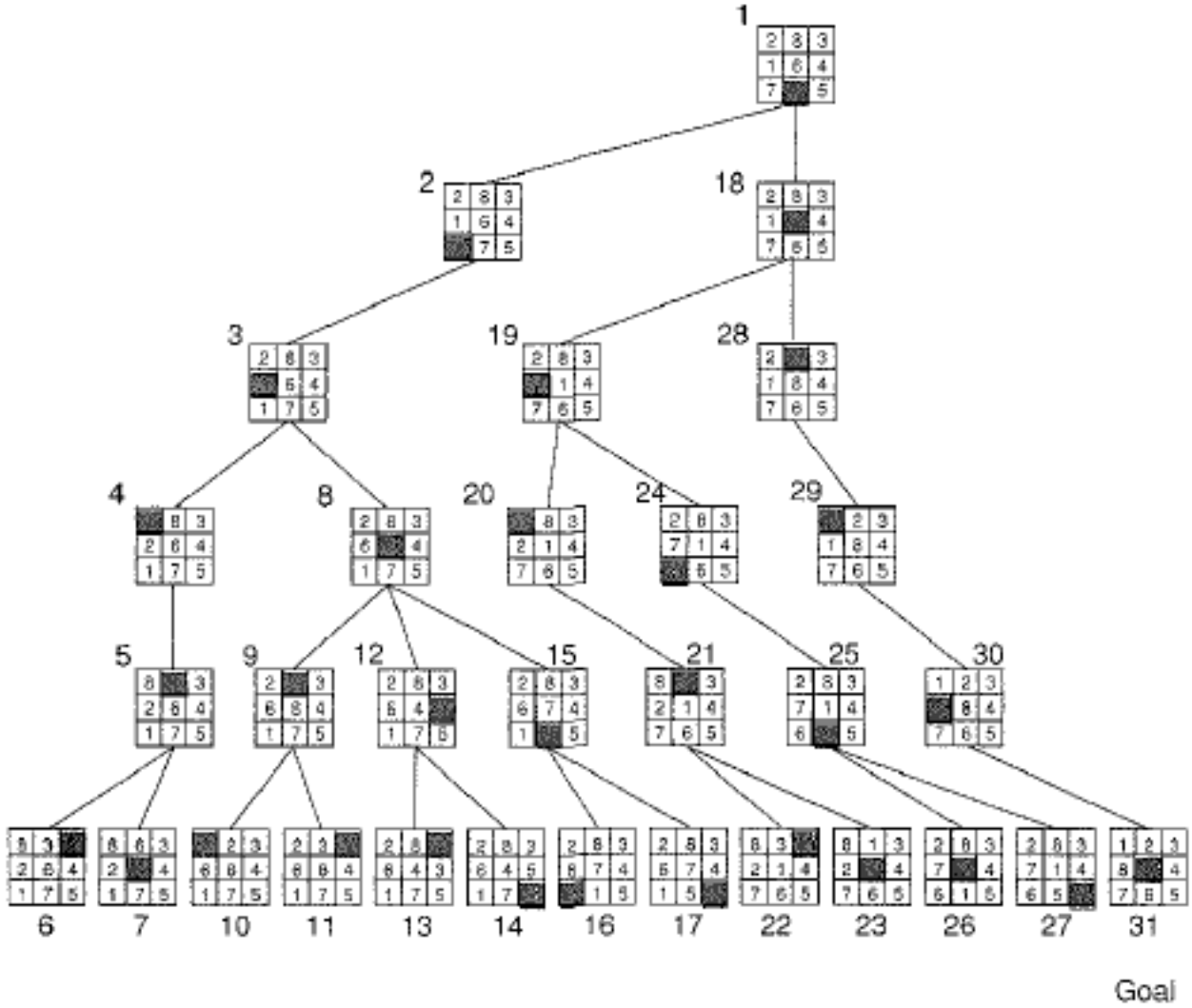
Busca por Aprofundamento Iterativo

- Teste de todos os possíveis limites com busca por profundidade limitada.
- Em geral é o melhor método quando o espaço de busca é grande e a profundidade é desconhecida.



Busca por Aprofundamento Limitado

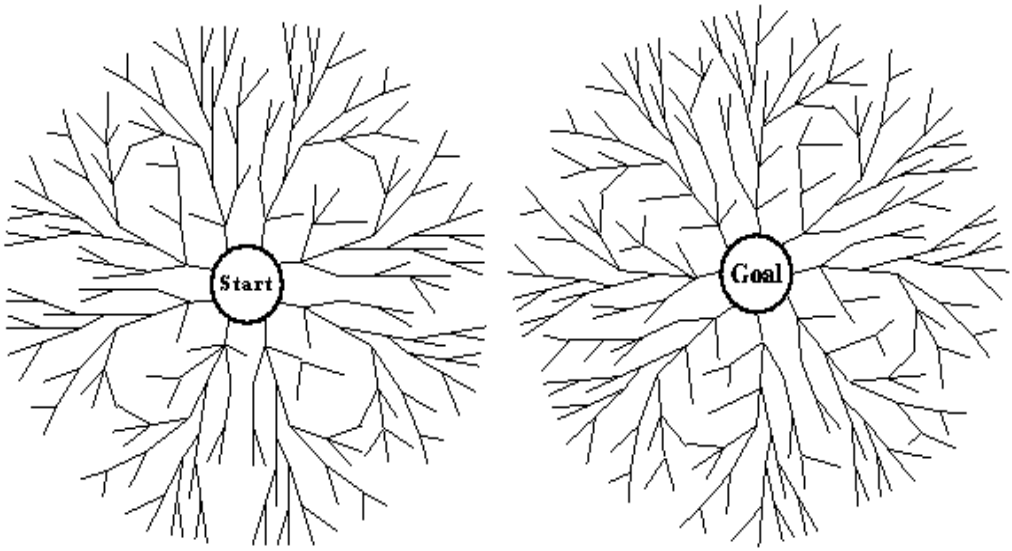
- Exemplo



Busca Bidirecional

- A idéia deste método de busca é procurar simultaneamente "para a frente" a partir do estado inicial e "para trás" a partir do estado final, e parar quando as duas buscas se encontrarem no meio.
- Nem sempre isto é possível, para alguns problemas os operadores não são reversíveis, isto é, não existe a função predecessora e portanto não é possível fazer a busca "para trás".

- Análise de Complexidade
  - Comparando com a busca em largura, o tempo e o espaço para a busca é proporcional a $2b^{d/2}$, onde d é o nível onde está a solução e b é o fator de ramificação da árvore.
  - Exemplo: Para b=10 e d=6, na busca em largura seriam gerados 1.111.111 nós, enquanto que na busca bidirecion



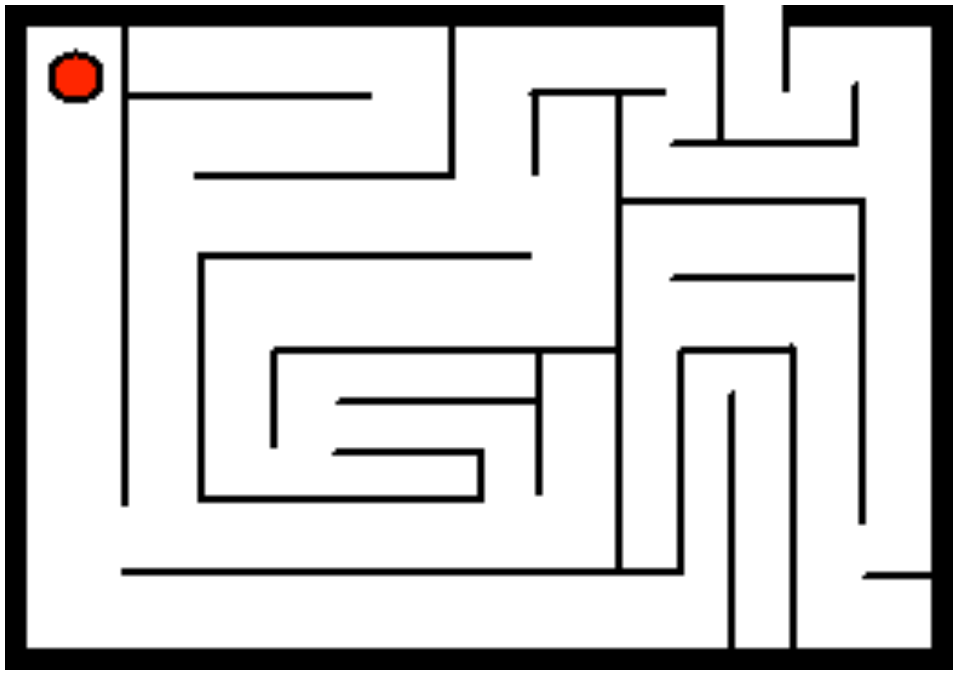
Comparação entre Métodos de Busca

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes*	Yes*	No	Yes, if $l \geq d$	Yes
Time	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	b^m	b^l	b^d
Space	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	bm	bl	bd
Optimal?	Yes*	Yes*	No	No	Yes

b: fator de ramificação
d: profundidade da solução mais rasa
m: profundidade máxima da árvore de busca
l: limite de profundidade

Exercício

- achar um caminho para sair desse labirinto



Busca Heurística (Informed Search)

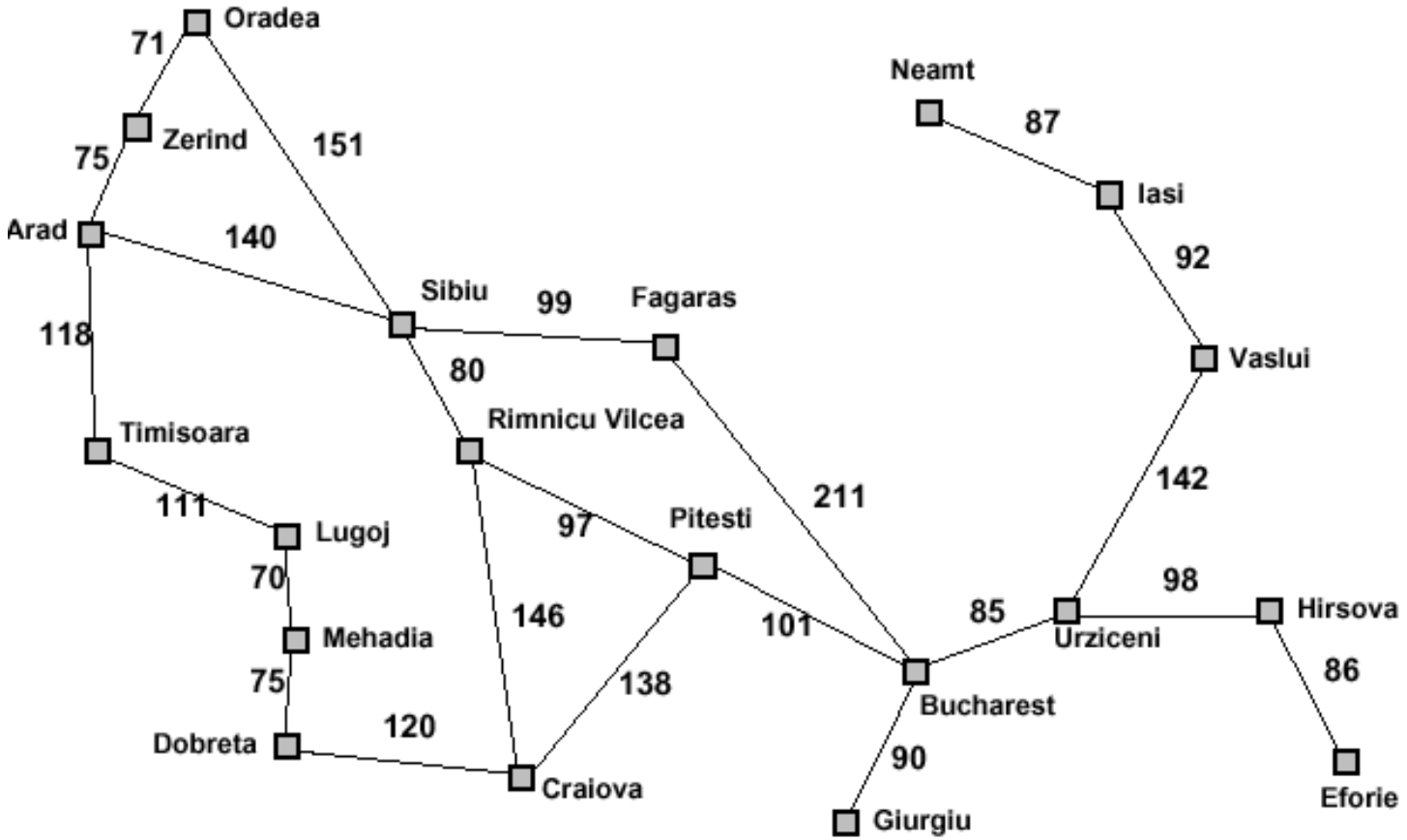
- Os métodos de busca vistos anteriormente fornecem uma solução para o problema de achar um caminho até um nó meta. Entretanto, em muitos casos, a utilização destes métodos é impraticável devido ao número muito elevado de nós a expandir antes de achar uma solução.
- Para muitos problemas, é possível estabelecer princípios ou regras práticas para ajudar a reduzir a busca.
- A técnica usada para melhorar a busca depende de informações especiais acerca do problema em questão.
- Chamamos a este tipo de informação de INFORMAÇÃO HEURISTICA e os procedimentos de busca que a utilizam de MÉTODOS DE BUSCA HEURISTICA.

Busca Heurística (Informed Search)

- A informação que pode compor uma informação heurística pode ser baseada no Custo do Caminho necessário para atingir um estado meta.
- A estimativa do CUSTO DO CAMINHO em um dado estado do problema pode ser composta pelo somatório de dois outros custos:
  - O custo do caminho desde o estado inicial até o estado atual que está sendo expandido (função g); e
  - Uma estimativa do custo do caminho do estado atual até o estado meta (função heurística h).
- A filosofia geral que move a busca heurística é: O MELHOR PRIMEIRO. Isto é, no processo de busca deve-se primeiro expandir o nó "mais desejável" segundo uma função de avaliação.

Busca Heurística
(Informed Search)

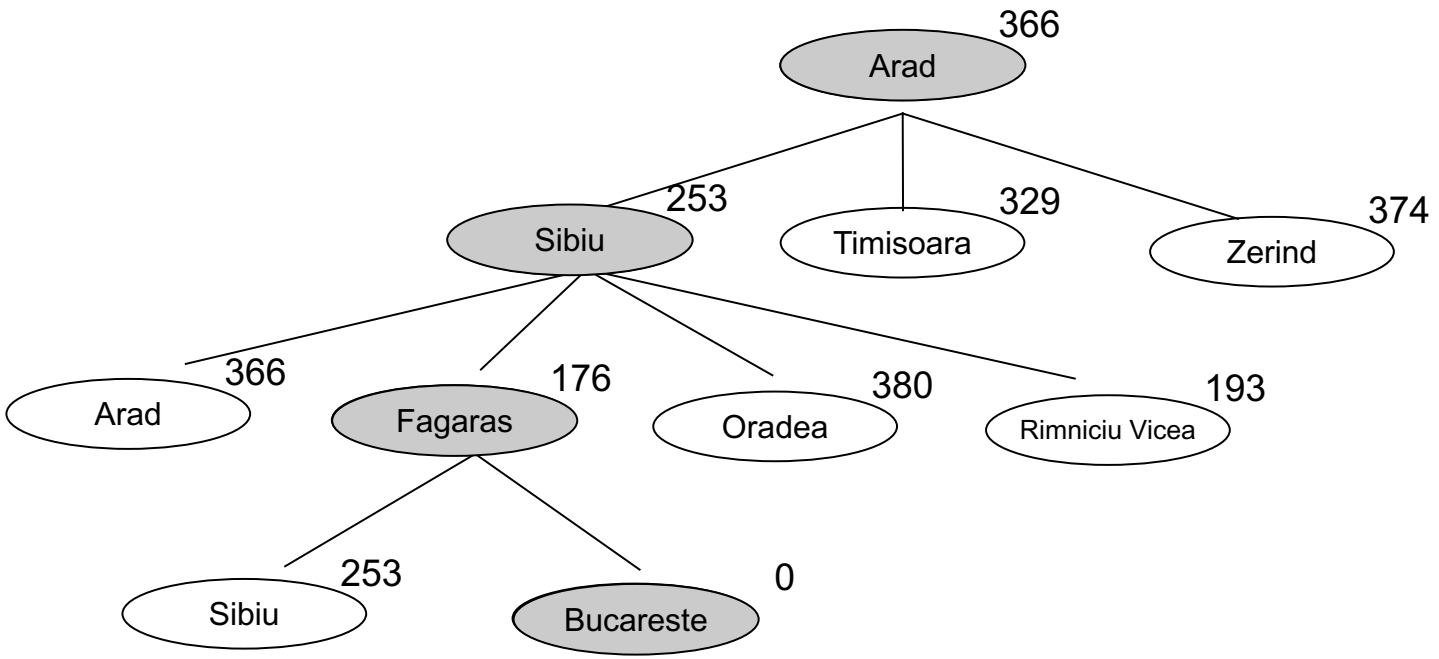
Romania with step costs in km



Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Busca Gulosa (Greedy Search)

- Semelhante à busca em profundidade com backtracking.
- Tenta expandir o nó que parece mais próximo ao nó meta com base na estimativa feita pela função heurística h(n). f(n) = h(n)
- O exemplo abaixo (Russel e Norvig) é baseado em um mapa da Romênia, o objetivo é encontrar o melhor caminho entre Arad e Bucareste
- h(n) é a distância estimada, em linha reta, de n até Bucareste.



Busca Gulosa (Greedy Search)

- Análise de Complexidade
  - É completa para busca em grafos se o espaço de estados for finito mas e se não admitir estados repetidos;
  - Tempo máximo estimado: $O(b^m)$, mas uma boa heurística pode reduzir drasticamente o tempo;
  - Espaço: $O(b^m)$, todos os nós são mantidos na memória;
  - Não garante a solução ótima.

Busca Heurística

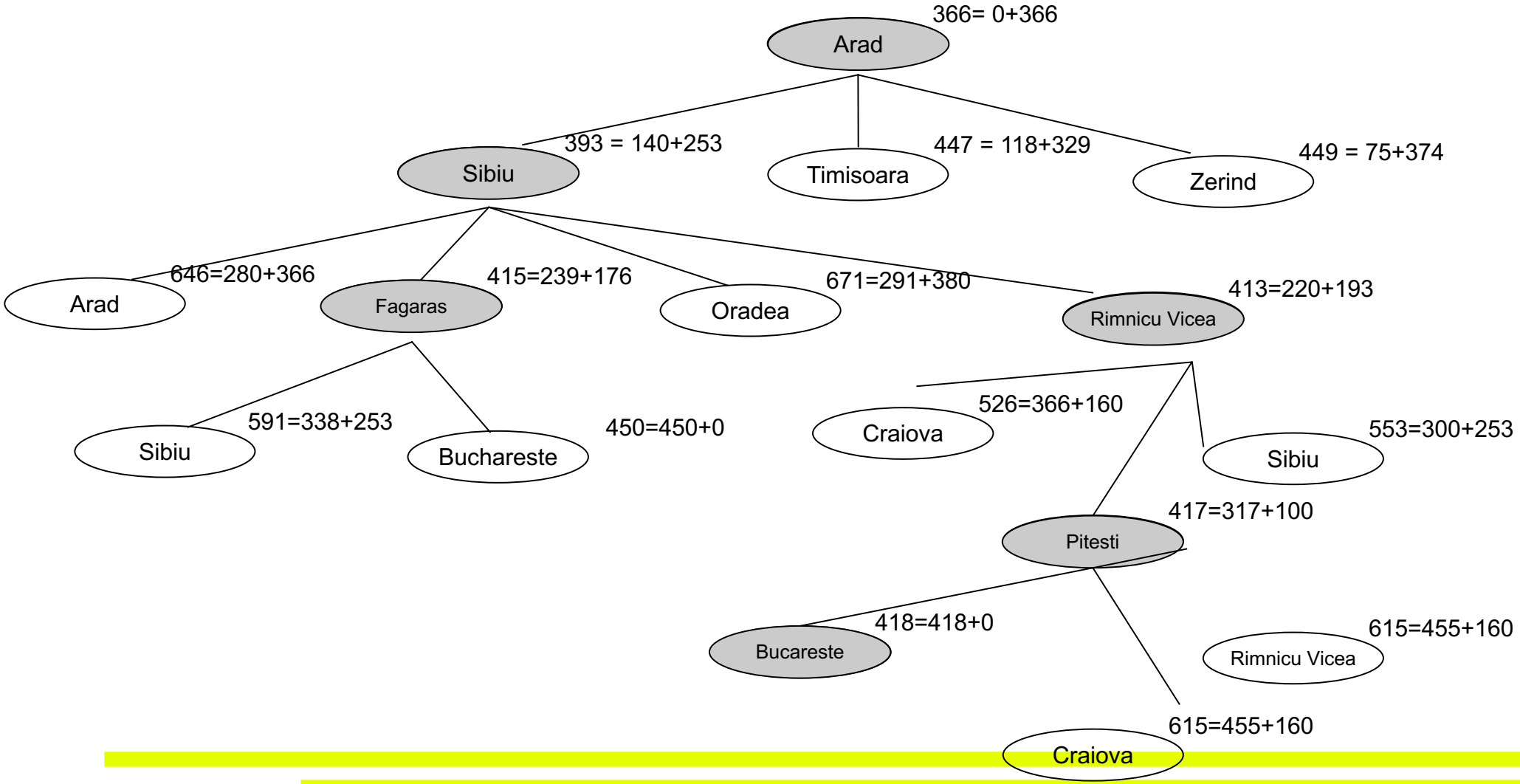
Busca A* (A estrela)

- Filosofia: procurar evitar expandir nós que já são "custosos".
- É um método de busca que procura otimizar a solução, considerando todas as informações disponíveis até aquele instante, não apenas as da última expansão.
- Todos os estados abertos até determinado instante são candidatos à expansão.
- Combina, de certa forma, as vantagens tanto da busca em largura como em profundidade
- Busca onde o nó de menor custo "aparente" na fronteira do espaço de estados é expandido primeiro.

- f(n) = g(n) + h(n) onde
  - g(n) = custo do caminho do nó inicial até o nó n.
  - h(n) = custo do caminho estimado do nó n até o nó final.
  - f(n) = custo do caminho total estimado.

Busca Heurística

Busca A* (A estrela)

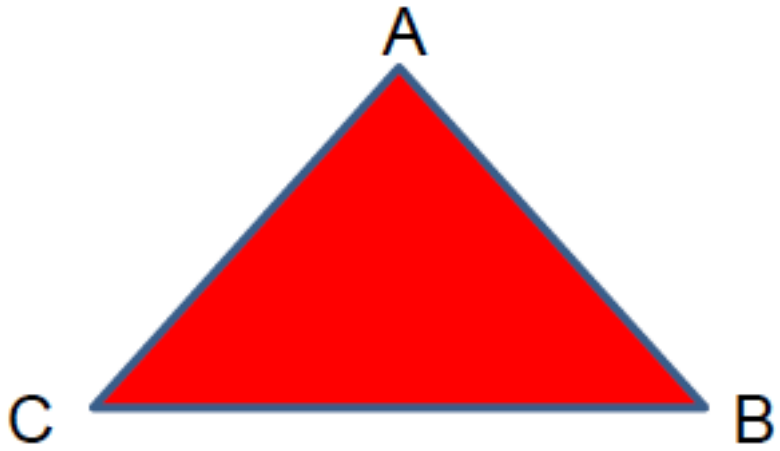


Busca A* (A estrela)

- A* expande o nó de menor valor de f a cada instante.
- A* deve usar uma heurística admissível, isto é, $h(n) \leq h^*(n)$ onde $h^*(n)$ é o **custo real** para ir de n até o nó final.
- **Admissibilidade de A***
  - Diz-se que um método de busca é ADMISSÍVEL se ele sempre encontra uma solução e se esta solução é a de menor custo.
  - A busca em largura é admissível. O mesmo não ocorre com a busca em profundidade.
- **A otimalidade de A* depende da componente h(n)**
- A busca A* é ótima, isto é, sempre encontra o caminho de menor custo até a meta se h(n) for admissível e consistente

Busca A* (A estrela)

- Para A* com BUSCA-EM-GRAFO há uma condição adicional:
- Ser CONSISTENTE ou MONOTÔNICO;
- Isto é: a f(n) deve ser não-decrescente
- Deve respeitar o teorema da DESIGUALDADE TRIANGULAR:
  - O caminho AB é sempre mais curto do que a soma dos caminhos AC e CB

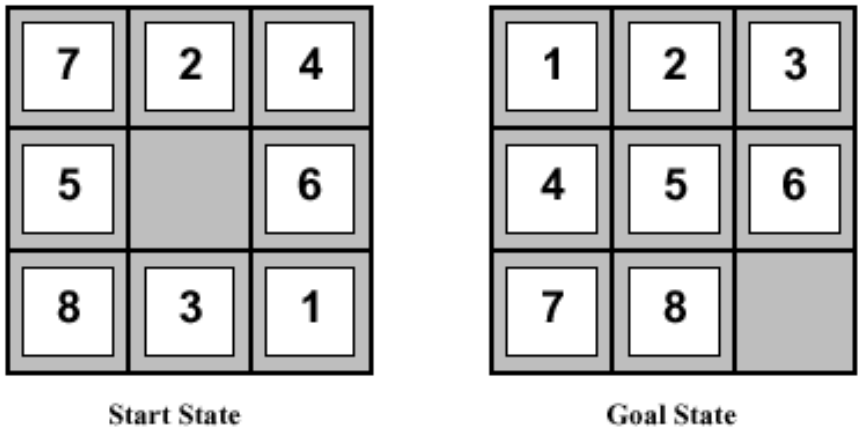


Busca A* (A estrela)

- Algoritmos que estendem o A* para otimizar o custo de memória:
  - Algoritmo A* de Aprofundamento Interativo - AIA*
  - Algoritmo de Busca Recursiva pelo Melhor – BRPM
  - A* Limitado pela Memória – LMA*
  - A* Limitado pela Memória Simplificado – LMSA*

Busca A* (A estrela)

- Quanto mais admissível a heurística, menor o custo da busca.
- Exemplos: Para o jogo do oito
  - h1(n): número de peças fora do lugar
  - h2(n): distância Manhattan (número de casas longe da posição final em cada direção)



Start State | Goal State

$h_1(S) =$?? 7

$h_2(S) =$?? 4+0+3+3+1+0+2+1 = 14

Algoritmos de Busca Local

- Em muitos problemas de otimização o caminho ao objetivo é irrelevante(a sequência de ações), interessa-nos o estado objetivo = solução
  - problemas de otimização
  - satisfação de restrições (ex. N-Rainhas – configuração final)
- •Espaço de estados = conjunto de configurações "completas" do mundo
  - Ex. um arranjo das n-rainhas no tabuleiro
- Nestes casos, podem ser usados algoritmos de busca local

Algoritmos de Busca Local

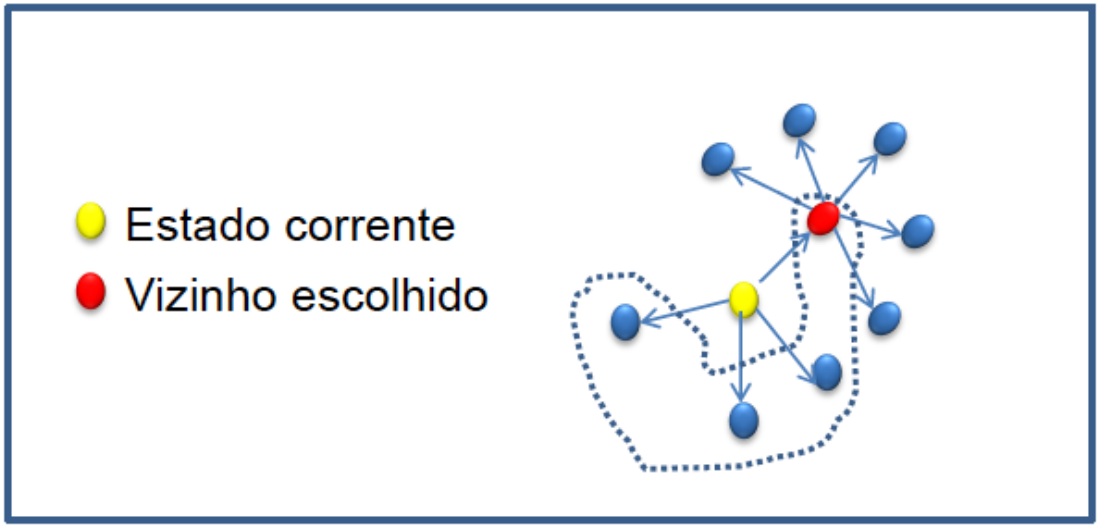
- Problemas onde o caminho é irrelevante. O que importa é a solução final:
  - Problema das oito rainhas
  - Projeto de circuitos integrados
  - Layout de organizações industriais
  - Escalonamento de jornadas de trabalho
  - Otimização de redes
  - Roteamento
  - Etc.

Algoritmos de Busca Local

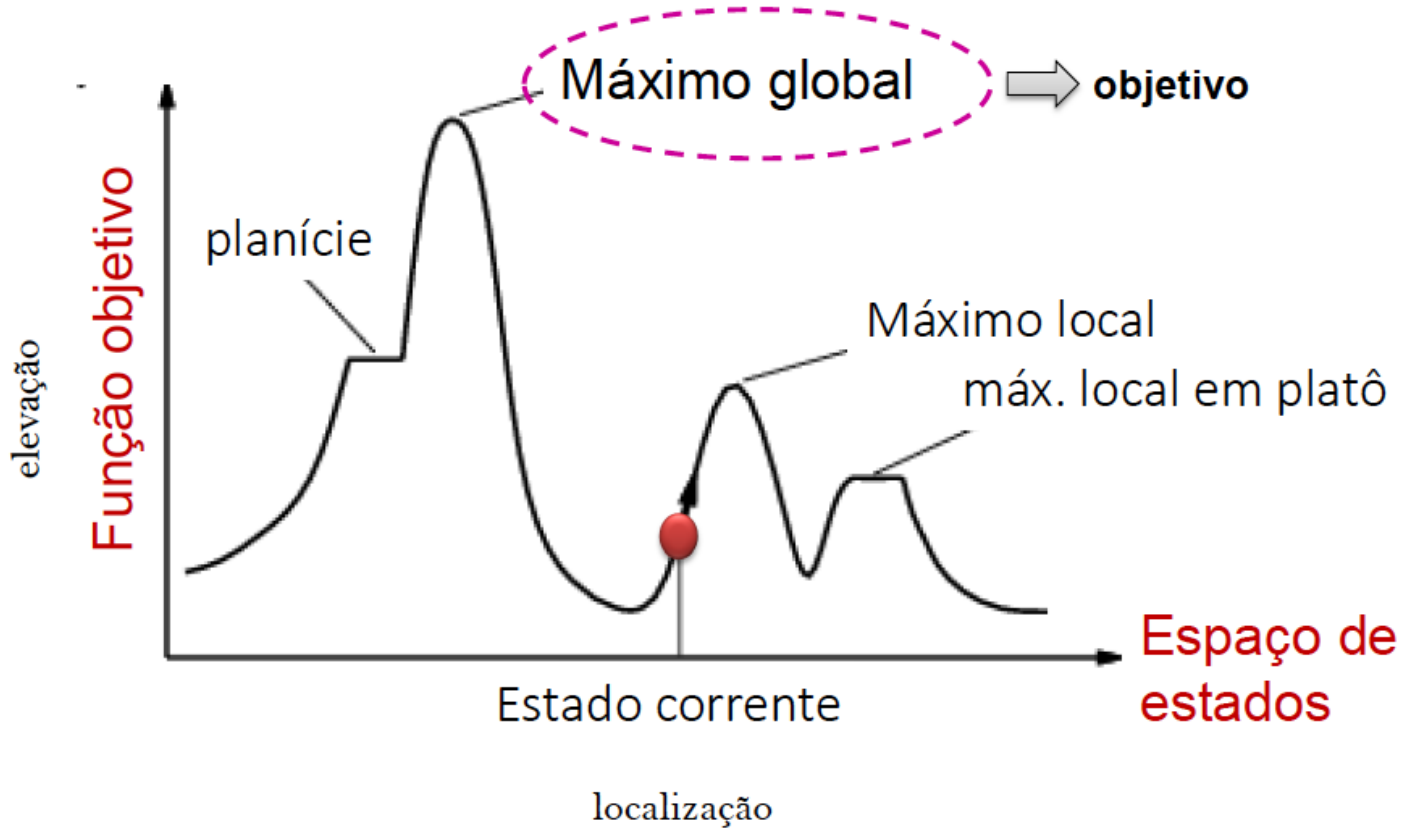
- Vantagens em relação às buscas cega e informada:
  - Usam pouca memória (normalmente, uma quantidade constante de memória)
  - Podem encontrar soluções razoáveis/factíveis em espaços de estados grandes ou infinitos (e também contínuos) para os quais algoritmos sistemáticos como os de busca cega e informada não são adequados.

Algoritmos de Busca Local

- Estratégia:
  - Manter um só estado (ou poucos) como o "atual" e tentar melhorar o mesmo movimentando-se para estados vizinhos imediatos



Algoritmos de Busca Local



Algoritmos de Busca Local

- Se o objetivo é maximizar temos que encontrar o maior pico, se for minimizar, o vale mais profundo.
- Completo: se o algoritmo consegue atingir um estado objetivo desde que ele exista.
- Ótimo: se consegue encontrar o mínimo/máximo para a função de custo.

Algoritmos de Busca Local

- É a estratégia mais simples e popular. Baseada na Busca em Profundidade.
- É um método de busca local
  - Um único estado corrente e
  - Movem-se apenas para os vizinhos desse estado
- O objetivo deve ser atingido com o menor número de passos.
- A idéia heurística que lhe dá suporte é a de que o número de passos para atingir um objetivo é inversamente proporcional ao tamanho destes passos.
- Empregando uma ordenação total ou parcial do conjunto de estados, é possível dizer se um estado sucessor leva para mais perto ou para mais longe da solução. Assim o algoritmo de busca pode preferir explorar em primeiro lugar os estados que levam para mais perto da solução.

Busca Subida da Encosta (Hill climbing)

- Laço repetitivo que se move de forma contínua no sentido do valor crescente
- Há duas variações do método:
  - **SUBIDA DE ENCOSTA SIMPLES:** Vai examinando os sucessores do estado atual e segue para o primeiro estado que for maior que o atual.
  - **SUBIDA DE ENCOSTA PELA TRILHA MAIS ÍNGREME:** Examina TODOS os sucessores do estado atual e escolhe entre estes sucessores qual é o que está mais próximo da solução.
- Este método não assegura que se atinja o ponto mais alto da montanha.
- Ele assegura somente que atingido um ponto mais alto do que seus vizinhos, então encontramos uma boa solução local.

Busca Subida da Encosta (Hill climbing)

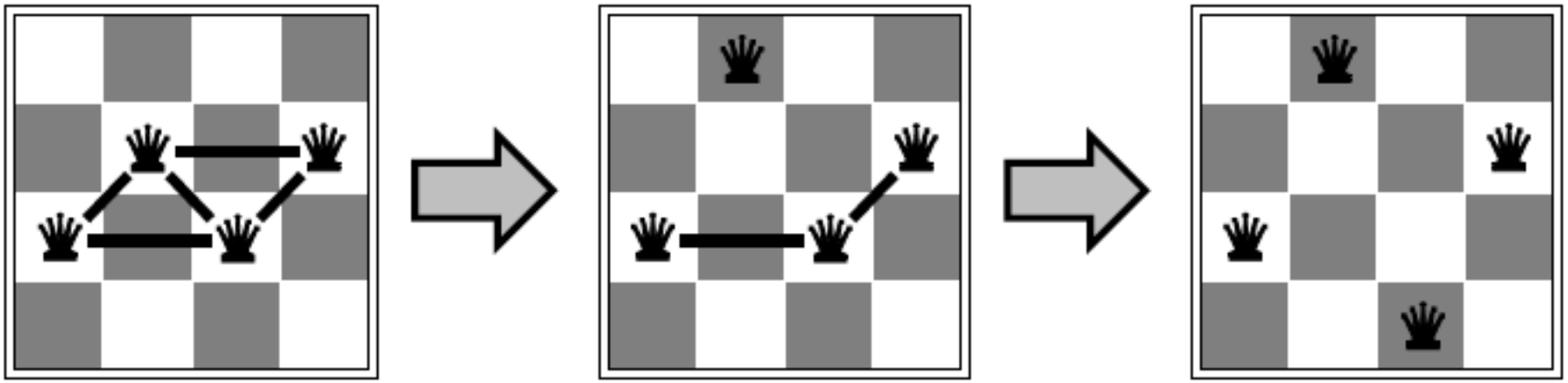
- Metáfora: escalando o Everest no meio de uma tempestade de neve com amnésia"

```
function HILL-CLIMBING( problem) returns a state that is a local maximum
    inputs: problem, a problem
    local variables: current, a node
                     neighbor, a node

    current ← MAKE-NODE(INITIAL-STATE[problem])
    loop do
        neighbor ← a highest-valued successor of current   // melhor vizinho
        if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
        current ← neighbor
```

Busca Subida da Encosta
(Hill climbing)

- Exemplo: problema das N Rainhas

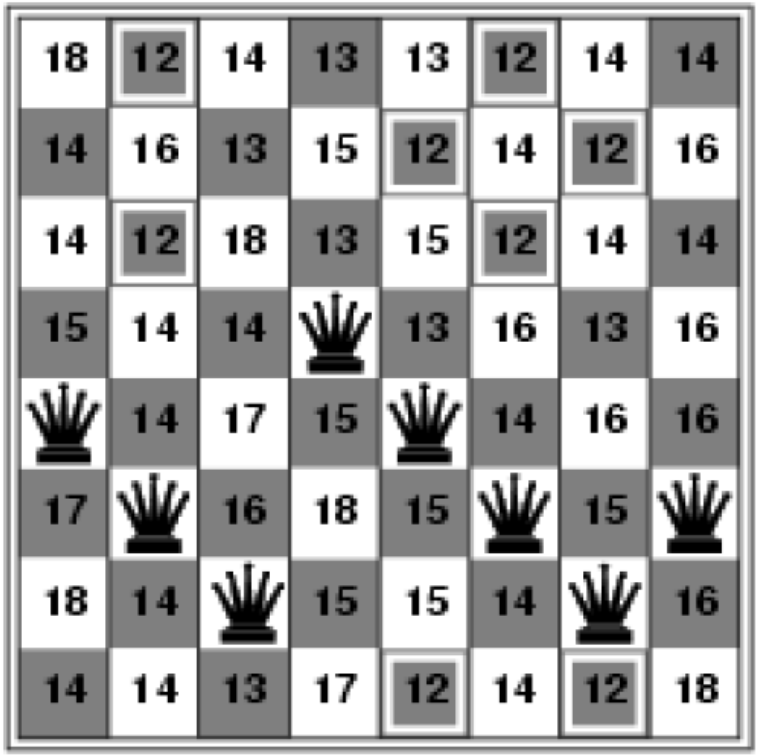


Busca Subida da Encosta (Hill climbing)

- Exemplo: problema das N Rainhas
  - Formulação: cada estado tem as 8 rainhas no tabuleiro, uma por coluna.
  - Ação: movimentar uma rainha para qualquer posição na sua coluna.
  - Função sucessora: dado um estado e uma ação, retorna o estado alcançado.
    - Para cada estado, há 56 estados sucessores: 8 rainhas x 7 posições
  - Função de custo/heurística h: número de pares de rainhas que se atacam direta ou indiretamente
  - Estado objetivo: configuração com h=0 (mínimo global)
    - somente para as soluções perfeitas

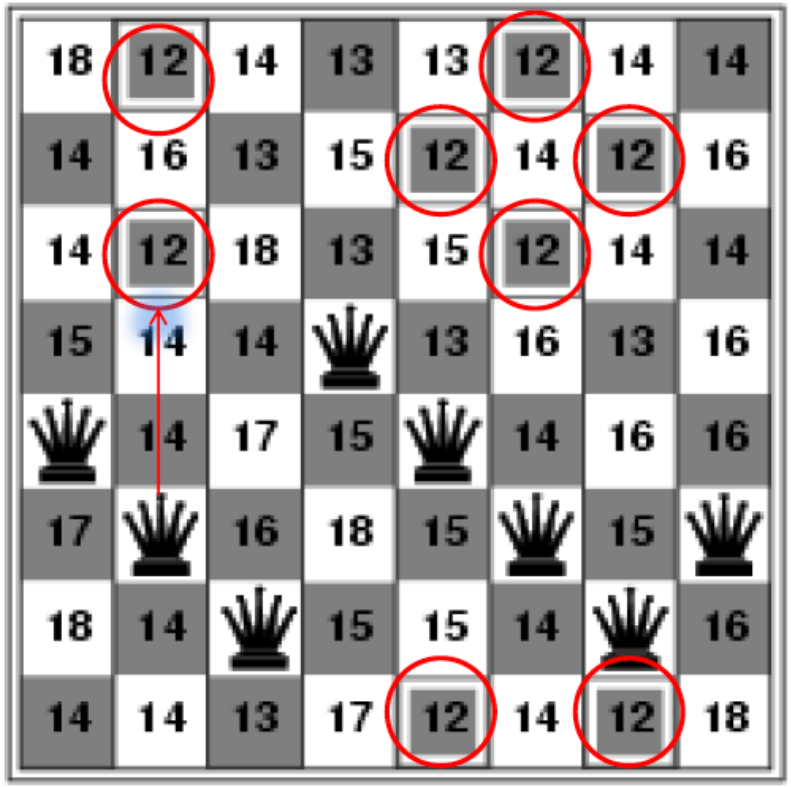
Busca Subida da Encosta
(Hill climbing)

- Neste estado, h=17 (há 17 ataques entre rainhas)
  - Qual a melhor ação (por uma escolha gulosa)?



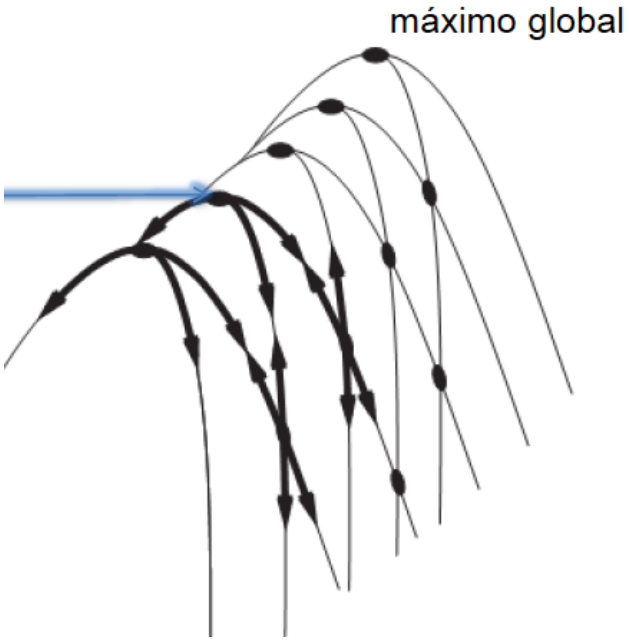
Busca Subida da Encosta
(Hill climbing)

- Resposta: qualquer movimento que leve uma das rainhas na sua coluna a uma posição marcada com 12



Busca Subida da Encosta (Hill climbing)

- Problemas:
  - Ficar parado num máximo local
  - Cumeeiras
  - Platôs
  - Os máximos locais não estão interconectados. Qualquer passo a partir deles leva a um valor mais baixo



Busca Subida da Encosta (Hill climbing)

- Variações
  - Subida pela encosta estocástica
    - Escolhe ao acaso entre os movimentos encosta acima
  - Subida pela encosta pela primeira escolha
    - Gera sucessores ao acaso até ser gerado um melhor do que o atual
  - Subida pela encosta com reinício aleatório
    - Gera estados iniciais ao acaso (continue tentando)

Busca Subida da Encosta (Hill climbing)

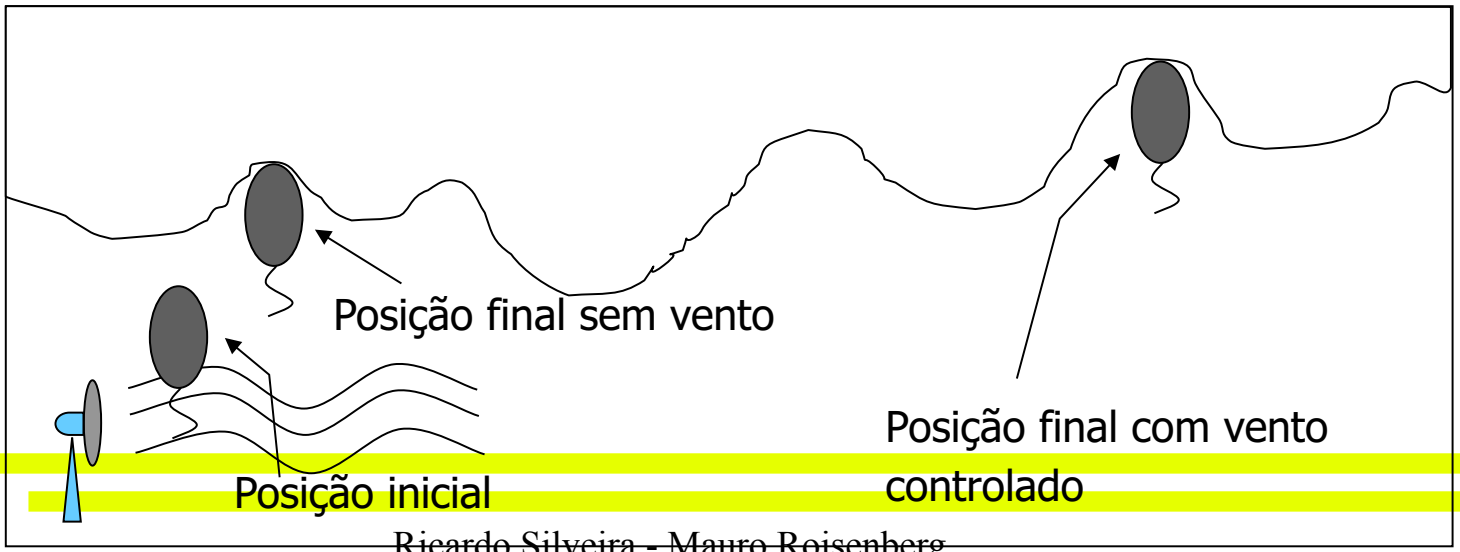
- Alternativas:
  - Subida de Encosta com Reinício Aleatório
    - O algoritmo realiza uma série de buscas a partir de estados iniciais gerados aleatoriamente.
    - Cada busca é executada
      - até que um número máximo estipulado de iterações seja atingido, ou
      - até que os resultados encontrados não apresentem melhora significativa.
    - O algoritmo escolhe o melhor resultado obtido com as diferentes buscas (diferentes reinícios).
    - Cada execução produz apenas uma solução!
  - Busca de Têmpera Simulada
  - Algoritmos genéticos

Busca por Têmpera Simulada
(Simulated Annealing)

- Combina algoritmo que nunca faz movimentos encosta abaixo com os de percurso puramente aleatórios
- Combina eficiência e completeza
- É adequado a problemas nos quais a subida de encosta encontra muitos platôs e máximos locais.
- Não utiliza backtracking e Não garante que a solução encontrada seja a melhor possível.
- Pode ser utilizado em problemas NP-completos.
- É inspirado no processo de têmpera do aço. Temperaturas são gradativamente abaixadas, até que a estrutura molecular se torne suficientemente uniforme.

Busca por Têmpera Simulada (Simulated Annealing)

- A idéia é permitir "maus movimentos" que com o tempo vão diminuindo de freqüência e intensidade para poder escapar de máximos locais.
- O que o algoritmo de têmpera simulada faz é atribuir uma certa "energia" inicial ao processo de busca, permitindo que, além de subir encostas, o algoritmo seja capaz de descer encostas e percorrer platôs se a energia for suficiente.



Busca competitiva (Jogos)

- Os jogos tem atraído a atenção da humanidade, às vezes de modo alarmante, desde a antiguidade.
- O que o torna atraente para a IA é que é uma abstração da competição (guerra), onde se idealizam mundos em que agentes agem para diminuir o ganho de outros agentes. Além disso, os estados de um jogo são facilmente representáveis (acessíveis) e a quantidade de ações dos agentes é normalmente pequena e bem definida.
- A presença de um oponente torna o problema de decisão mais complicado do que os problemas de busca, pois introduz incertezas, já que não sabemos como o oponente irá agir.
- Geralmente o oponente tentará, na medida do possível, fazer o movimento menos benéfico para o adversário.

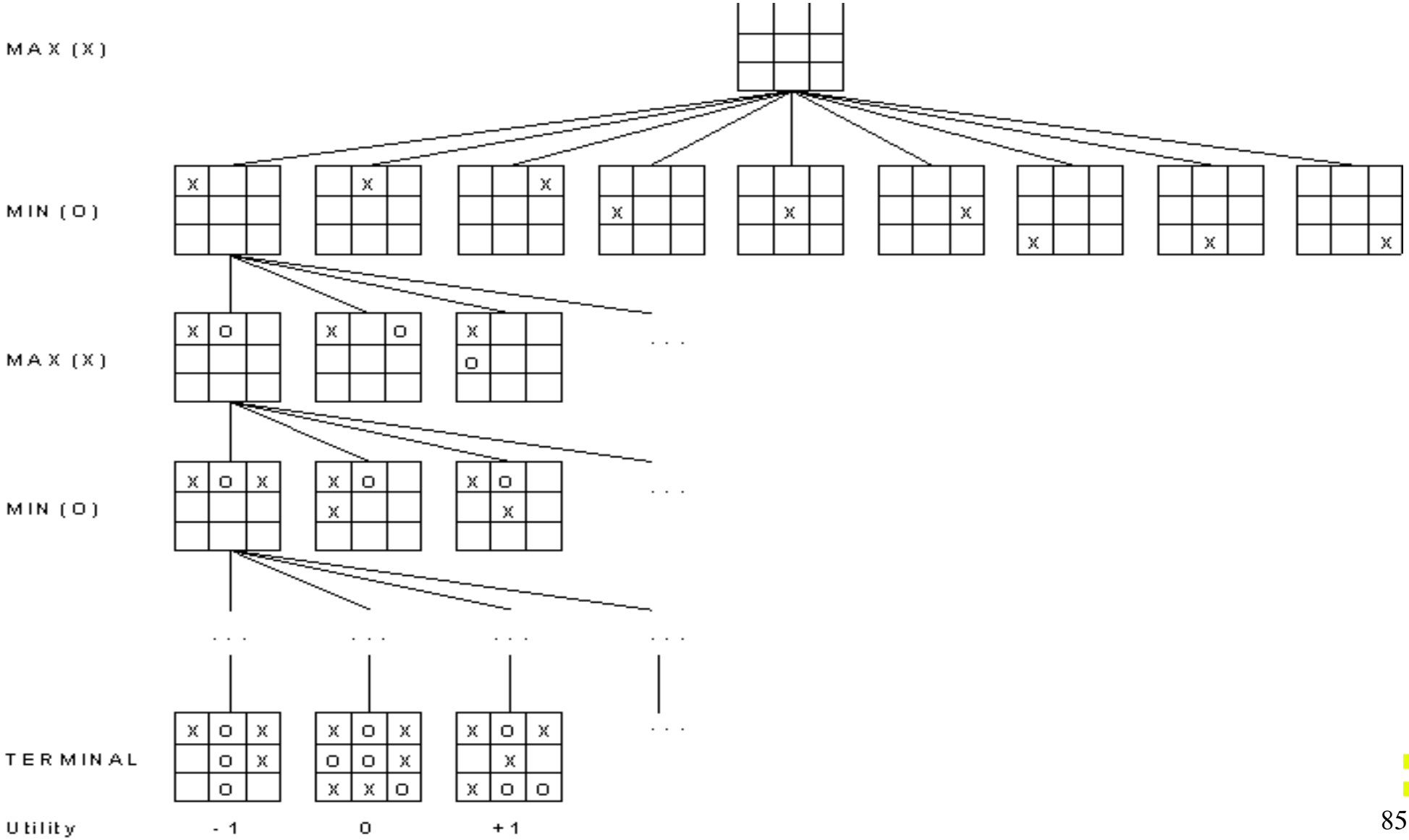
Jogos

- Jogos são, geralmente, problemas muito difíceis de resolver.
  - Xadrez difícil porque muito estados
  - Fator de ramificação 35 (número de movimentos por turno)
  - Geralmente 50 movimentos para cada jogador
  - 35^{100} estados ou nós
- Limites de tempo penalizam a ineficiência;
- Incerteza devido ao outro jogador
- Não é possível fazer a busca até o fim, de modo que devemos fazer o melhor possível baseados na experiência passada.
- Deste modo, jogos são muito mais parecidos com problemas do Mundo Real do que os problemas "Clássicos" vistos até agora.

Jogos

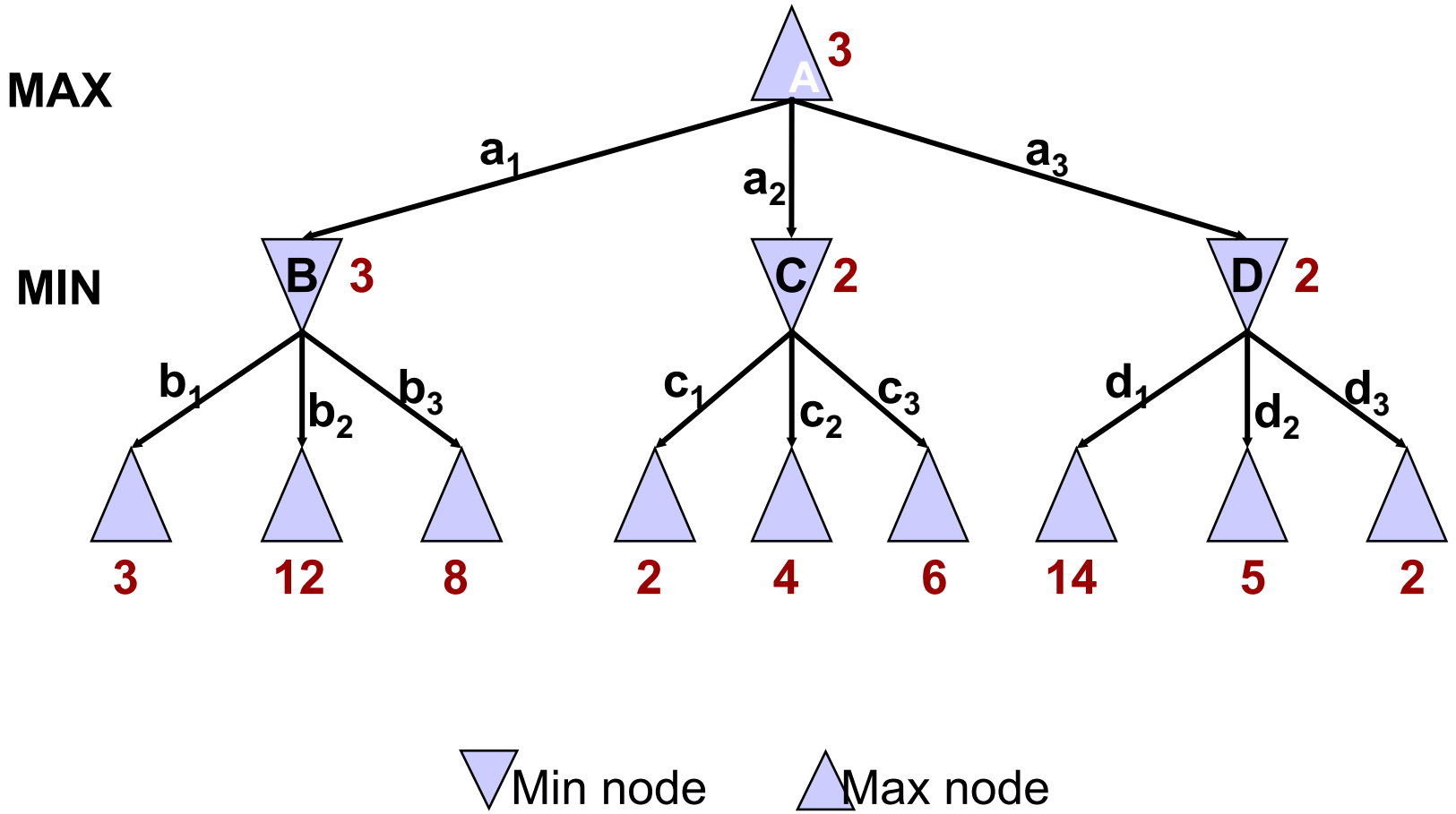
- Jogos de Duas Pessoas
  - Existem dois jogadores: MAX e MIN (MAX começa jogando).
- Jogos como um tipo de problema de busca:
  - O estado inicial;
  - Um conjunto de operadores;
  - Teste de Fim de Jogo (estados finais);
  - Função de Utilidade (payoff) – dá um resultado numérico para o resultado ou conseqüência de um jogo.
- Estratégia
  - Problemas de busca:
    - seqüência de movimentos que levam a um estado meta
  - MIN NÃO DESEJA QUE MAX ganhe;
  - MAX achar estratégia que leve a vitória independentemente dos movimentos de MIN.

Jogos



Jogos

- 1º Exemplo: Algoritmo MINMAX



Jogos

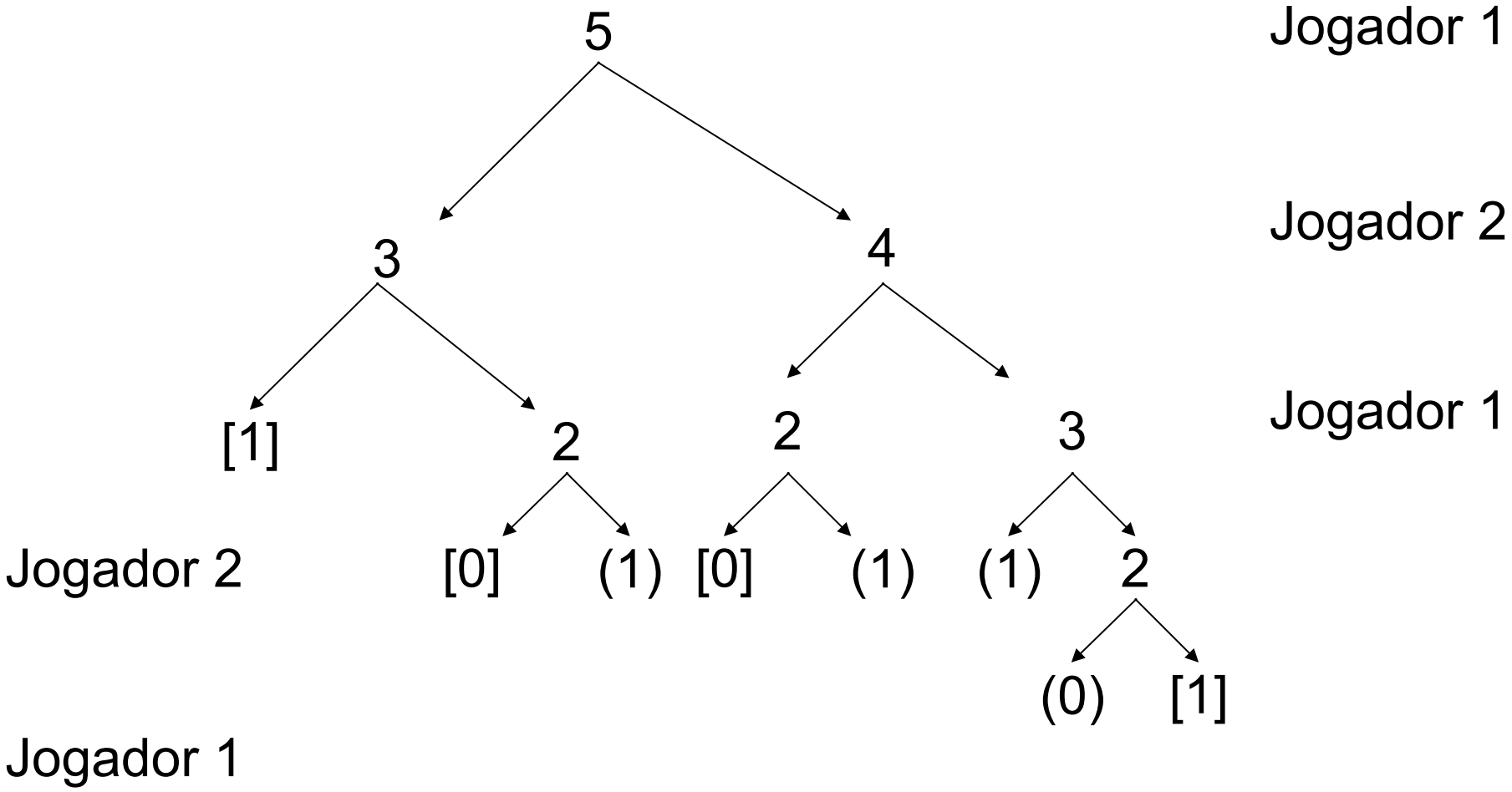
1º. Exemplo: Algoritmo MINMAX

1. Gerar toda a árvore do jogo;
2. Aplicar a função utilidade a cada nó terminal;
3. Usar a utilidade dos nós terminais para determinar a utilidade dos nós um nível acima:
   a) Quando acima é a vez de MIN fazer um movimento, escolher o que levaria para o retorno mínimo
   b) Quando acima é a vez de MAX fazer um movimento, escolher o que levaria para o retorno máximo
4. Continuar calculando os valores das folhas em direção ao nó raiz;
5. Eventualmente é alcançado o nó raiz nesse ponto MAX escolhe o movimento que leva ao maior valor.

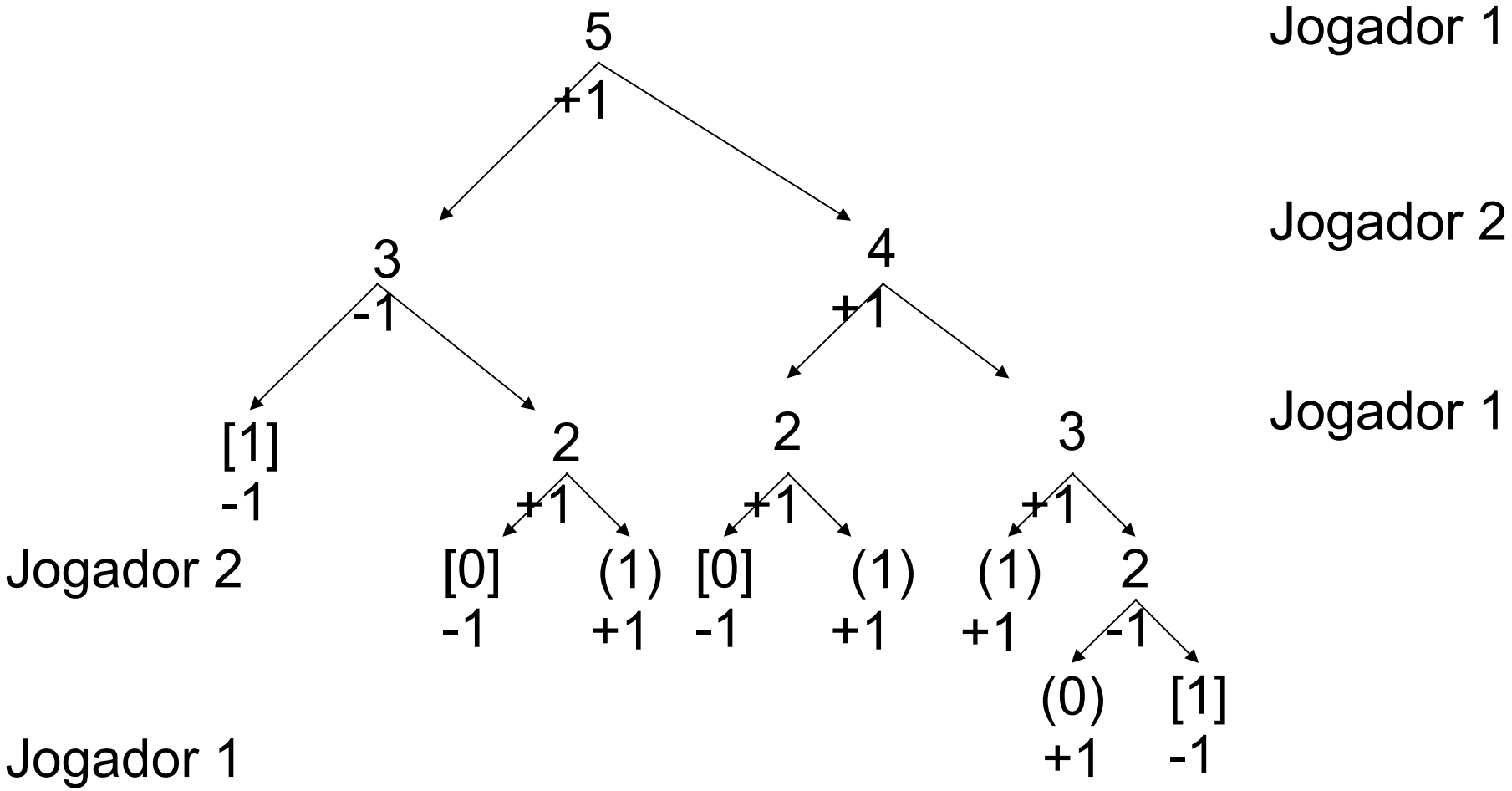
Jogo das 5 moedas

- Coloca-se uma pilha de 5 moedas
- Dois jogadores retiram alternadamente uma ou duas moedas
- O jogador que tirar aúltima moeda é derrotado

Jogo das 5 moedas



Jogo das 5 moedas



Críticas

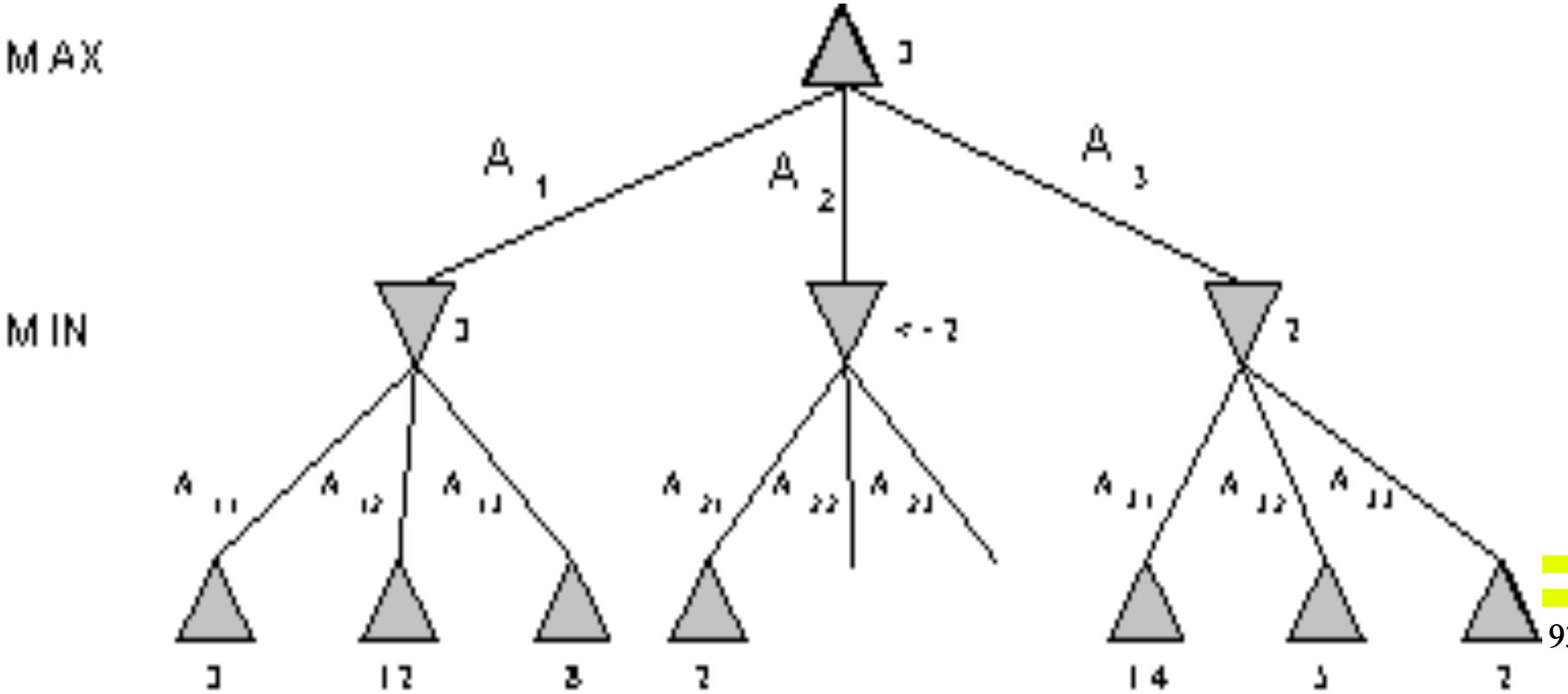
- É completa e ótima mas...
- Problemas
  - Tempo gasto é impraticável, porém o algoritmo serve como base para outros métodos mais realísticos.
  - Complexidade: $O(b^m)$.
- Para melhorar (combinar duas técnicas)
  1) Podar a arvore onde a busca seria irrelevante: **poda alfa-beta (alfa-beta pruning)**
  2) Substituir a profundidade n de min-max(n) pela estimativa de min-max(n): **função de avaliação**

Poda Alfa-Beta)

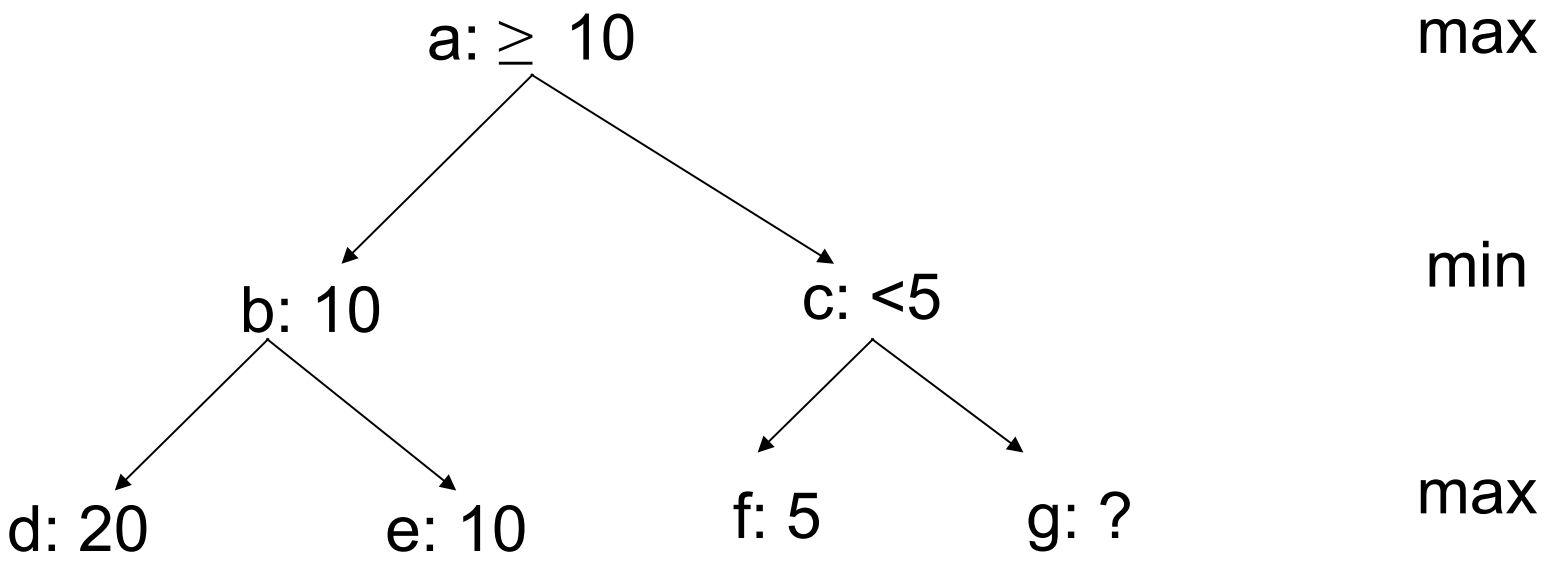
- **Função:**
  - Não expandir desnecessariamente nós durante o minimax (mas devolvendo o mesmo resultado)
- **Idéia:** não vale a pena piorar, se já achou algo melhor
- **Mantém 2 parâmetros**
  - α - melhor valor (mais alto) encontrado até então para MAX
  - β - melhor valor (mais baixo) encontrado até então para MIN
- **Teste de expansão:**
  - α não pode diminuir (não pode ser menor que um ancestral)
  - β não pode aumentar (não pode ser maior que um ancestral)

Jogos

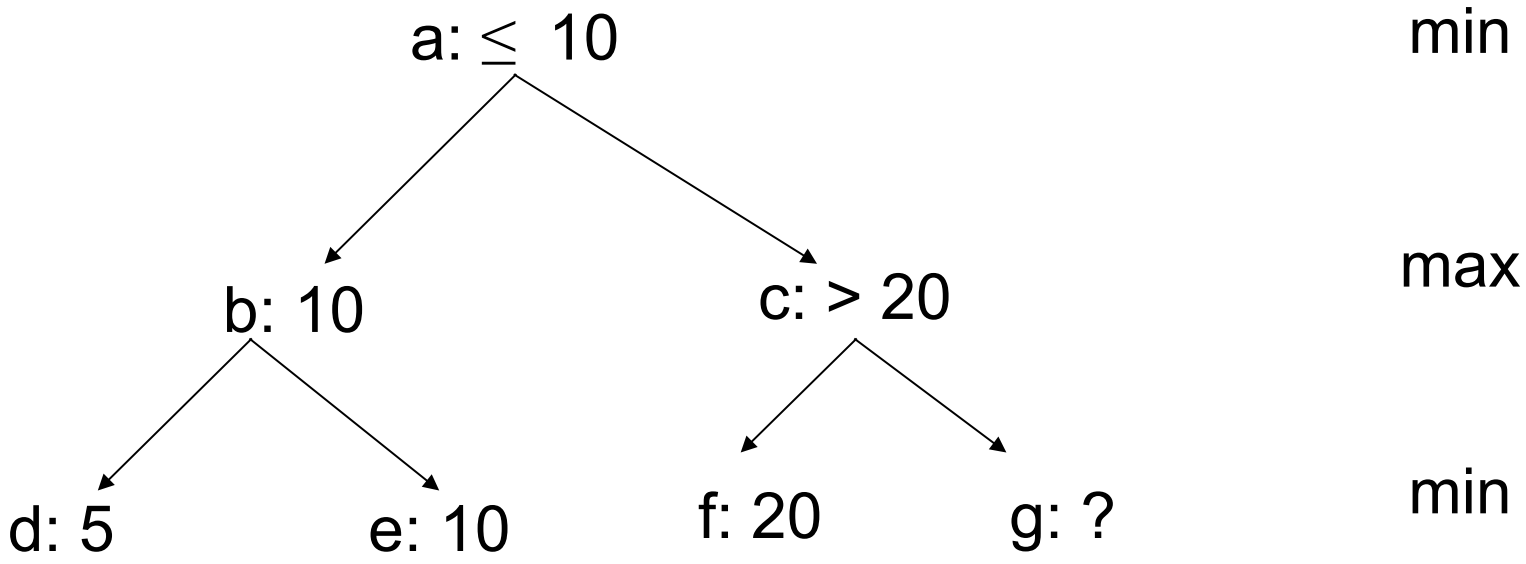
- Poda Alfa-Beta
  - Processo de eliminar ramos da árvore sem examiná-los.
  - MINIMAX é em profundidade
- Eficiência depende da ordem em que os sucessores são examinados.



Corte Alfa

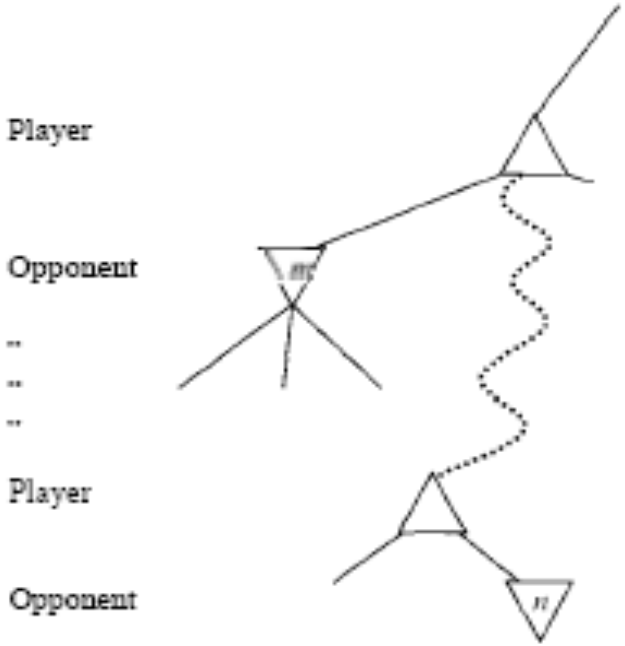


Corte Beta

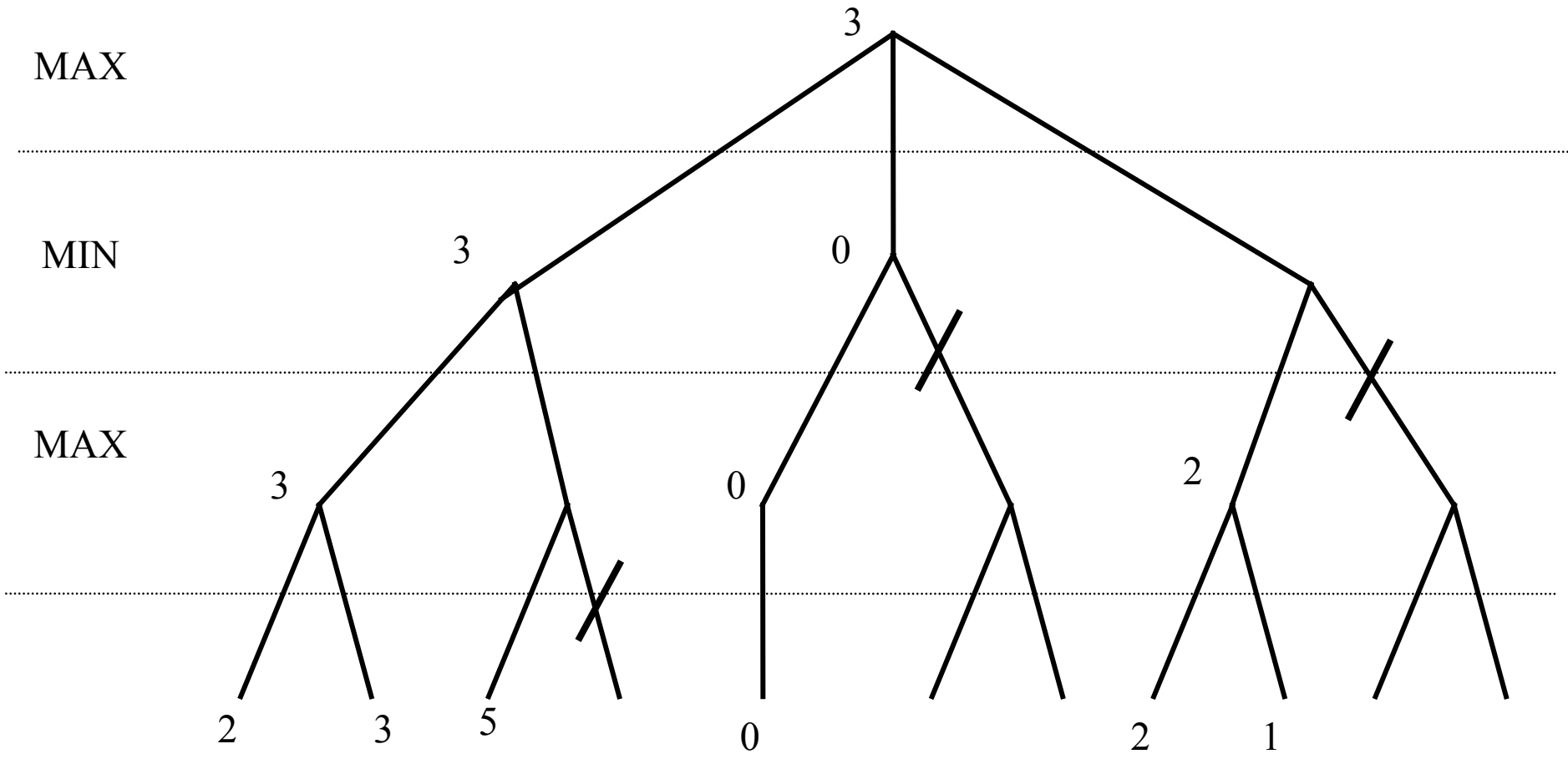


Poda alfa-beta

- α = valor da melhor escolha até o momento para MAX
- β = valor da melhor escolha até o momento para MIN



Poda Alpha-Beta: outro exemplo



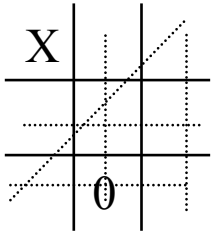
Jogos

- Decisões Imperfeitas
  - Algoritmo MINIMAX deve procurar até alcançar um nó terminal.
  - Usualmente isso não é prático (complexidade $O(b^m)$)
    - Onde b é o fator de ramificação e m a profundidade da árvore
  - Sugestão:
    - Parar a busca num tempo aceitável;
    - A função utilidade é substituída por uma função de avaliação heurística (EVAL), que retorna uma estimativa da utilidade esperada do jogo em uma dada posição;
    - Teste terminal (se nodo = folha retorna utilidade) por um teste de corte que define a priori o fim da profundidade

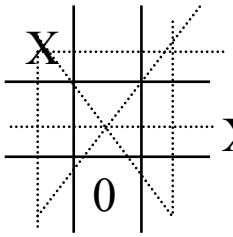
Jogos

- Função de Avaliação
  - Retorna uma estimativa da utilidade do jogo a partir de uma dada posição;
  - EX: xadrez: valor das peças, posições relativas, etc
  - Deve coincidir com a função de utilidade nos nós terminais;
  - Não deve ser difícil de calcular;
  - Deve refletir as chances reais de ganhar.

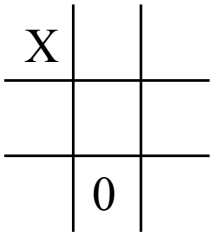
Exemplo de Função de avaliação para o jogo da velha



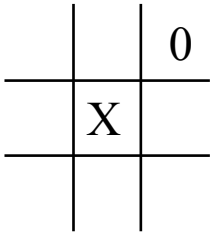
0 tem 5 possibilidades



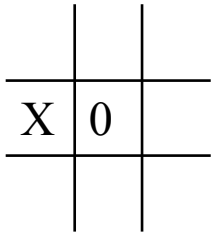
X tem 6 possibilidades



h = 6 - 5 = 1

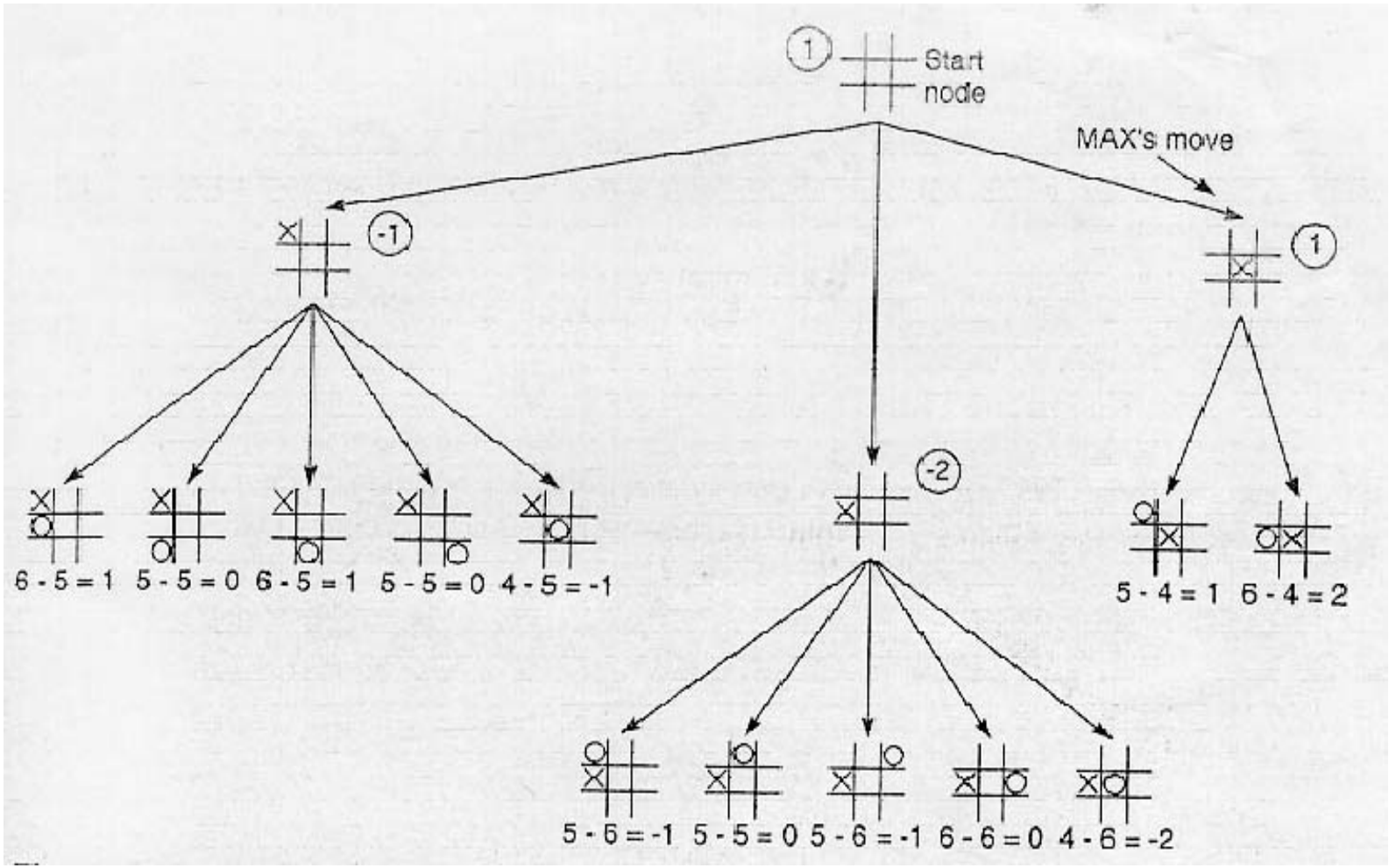


h = 5 - 4 = 1



h = 4 - 6 = = -2

Uso da Funções de Avaliação



Minimax de duas jogadas (two-ply) aplicado à abertura do jogo da velha

Quando aplicar a função de avaliação?

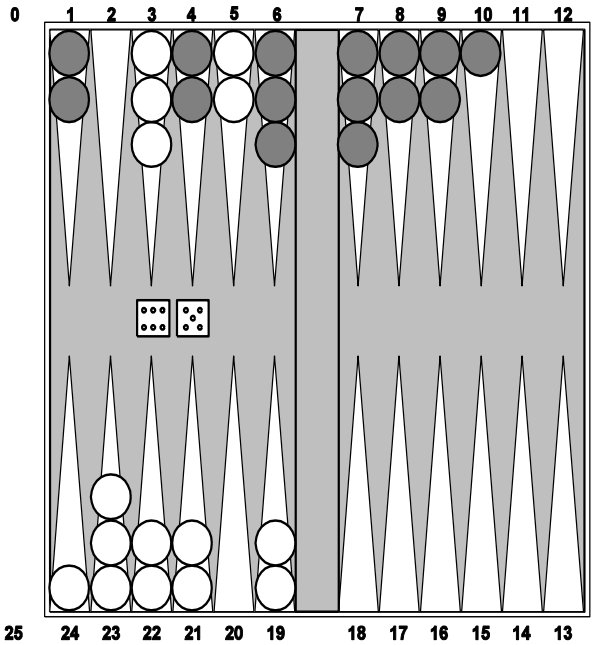
- Quando definir uma profundidade máxima ou iterativa não funciona devido à incerteza inerente ao problema
- Solução: Procura Tranqüila
  - **Idéia**: evitar avaliação em situações a partir das quais pode haver mudanças bruscas
  - No caso do jogo da velha, toda posição é tranqüila mas no xadrez não.... (ex. um peça de xadrez a ser comida)
  - **Algoritmo**: Se a situação (nó) é “tranqüila”, então aplica a função de avaliação, senão busca até encontrar uma situação “tranqüila”

Jogos com elemento de acaso

- Jogos que inclui elemento aleatório (gamão, jogos de cartas etc)
- Modela-se a árvore representando os espaços de busca incluindo camadas de nodos que representam a aleatoriedade
- Calcula-se o valor esperado com base no algorimo minimax para jogos determinísticos até um valor expectminimax que leva em conta a probabilidade estatística de um nodo sucessor ocorrer

Jogos com elementos de acaso

- Há jogos com elementos de imprevisibilidade maior do que os tratados
  - Ex. Gamão (Não sabemos quais são as jogadas legais do adversário e a A árvore de jogos (game tree) usada não vai servir!



Jogos com elementos de acaso

- Nova árvore de jogos
  - Inclui **nós de acaso**, além dos nós MAX e MIN
  - Os ramos dos nós de acaso, correspondem ao resultados do jogar dos dados do gamão, por exemplo
    - Cada ramo pode ter uma probabilidade associada
  - Não podemos mais falar de valor-minimax de um nó, mas sim de valor-minimax esperado (expectiminimax value)
- Expectiminimax (n) =

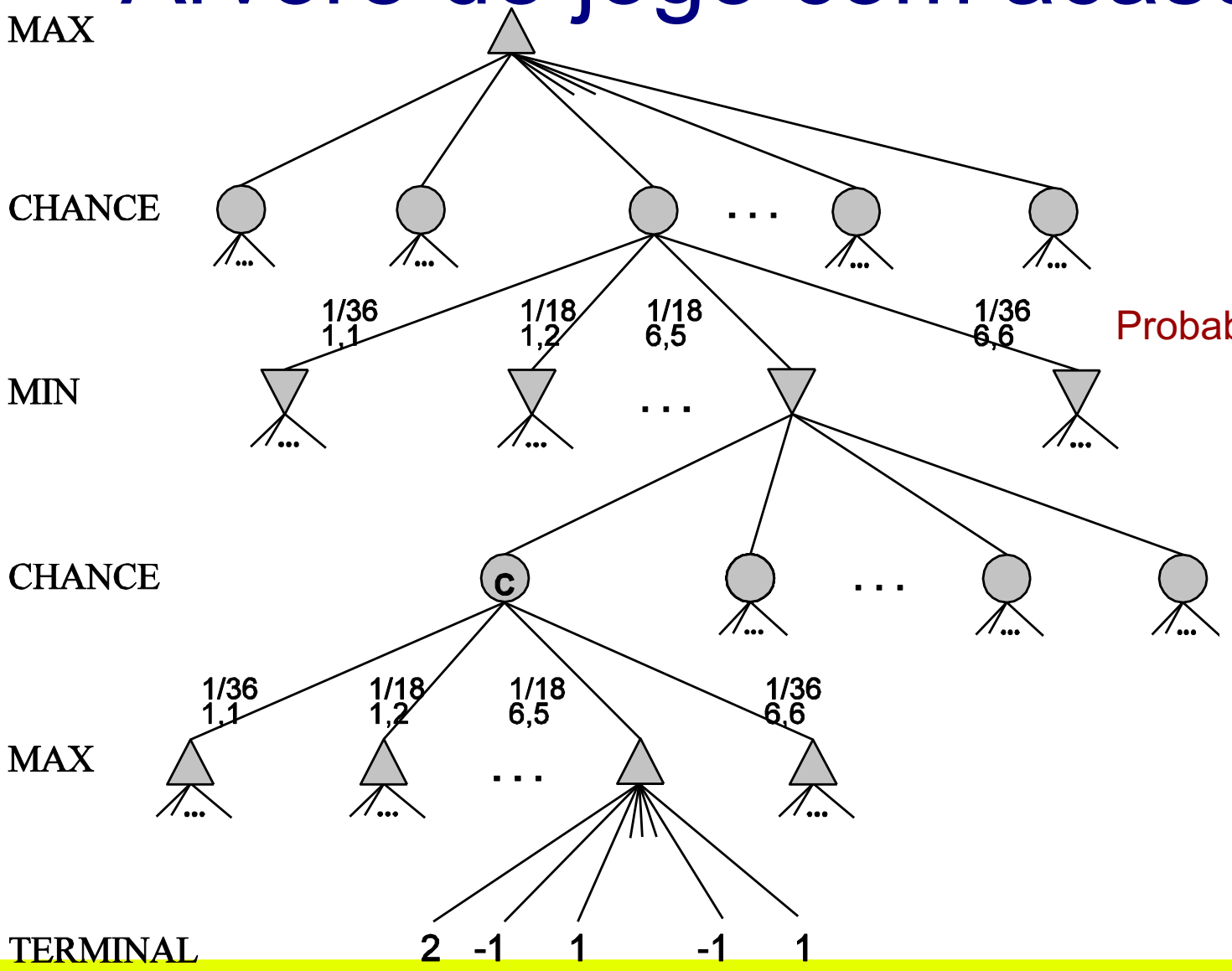
  Utility(n), se n é terminal

  $\max_{s \in Sucesssors(n)}$ Expectiminimax(s), se n é um nó Max

  $\min_{s \in Sucesssors(n)}$ Expectiminimax(s), se n é um nó Min

  $\Sigma_{s \in Sucesssors(n)}$ P(s).Expectiminimax(s), se n é um nó de acaso

Árvore de jogo com acaso



nós de acaso

Probabilidades

Jogos com vários jogadores

- Utilizam matrizes de recompensas (vetor de valores) associado a cada nó da árvore
- Para os estados terminais os vetores representam a utilidade do estado do ponto de vista de cada jogador
- Nos não terminais associados a cada jogador, a utilidade é calculada baseado na utilidade da melhor escolha para aquele jogador
- O valores de utilidades são propagados árvore acima, baseado na melhor escolha de cada jogador nos seus respectivos nós

Jogos com movimentos simultâneos

- Estudos servem para:
  - Projeto dos agentes e suas habilidades
  - Projeto de mecanismos (regras e ambientes)
- Envolvem:
  - Número de jogadores
  - Ações possíveis (operadores)
  - Matriz de recompensa
    - utilidade de cada combinação de ações que possam ser realizadas por todos os jogadores

Teoria dos Jogos

- Agentes baseados em utilidade podem atuar em ambientes incertos...
- Mas o que acontece quando a incerteza é proveniente de outros agentes e de suas decisões? E se as essas decisões são influenciadas pelas nossas?
  - A Teoria dos Jogos trata essas questões
  - É usada para tomar decisões sérias (decisões de preço, desenvolvimento de defesa nacional, etc)

TEORIA DOS JOGOS

- Se originou ao final da Segunda Guerra Mundial, como um ramo da matemática aplicada.
- Representa uma forma de modelar problemas que envolvem dois ou mais ‘tomadores de decisão’.
- Não se trata, portanto, de prescrições de como jogar um jogo e sim de mecanismos de análise de conflitos de interesse.
- A humanidade tem se ocupado com jogos ao longo de toda a sua história, embora as ferramentas de análise e a formalização dos processos envolvidos tenham sido propostas tão recentemente.

Teoria dos Jogos

- Histórico
  - John von Neumann publicou uma série de trabalhos em 1928 que culminou no livro lançado em 1944 The Theory of Games and Economic Behavior
  - Em 1950, a primeira discussão do Dilema do prisioneiro aparece, e um experimento foi conduzido neste jogo pela corporação RAND e que teve sua primeira análise no ano de 1953) popularizado pelo matemático Albert W. Tucker
  - conceito de Estratégia Evolucionariamente Estável, introduzida pelo biólogo John Maynard Smith no seu ensaio Game Theory and the Evolution of Fightin (http://pt.wikipedia.org/wiki/Estrat%C3%A9gia_evolucionariamente_est%C3%A1vel)
  - John Nash desenvolveu uma definição de uma estratégia ótima para jogos com vários jogadores onde nenhuma solução ótima ainda tinha sido definida, conhecido como equilíbrio de Nash.
  - Em 1965, Reinhard Selten introduziu seu conceito de solução do equilíbrio perfeito em sub-jogo, o qual foi depois refinado para o equilíbrio de Nash. Em 1967, John Harsanyi desenvolveu o conceito de informação completa e jogos Bayesianos. Ele juntamente com John Nash e Reinhard Selten ganharam o Prémio Nobel de Economia em 1994.

TEORIA DOS JOGOS

- Sua maior atratividade está nas aplicações, pois o conceito de jogo pode ser empregado na modelagem de situações tão diversas quanto:
  - Conflitos entre países, entre grupos sociais e entre grupos étnicos;
  - Políticas de preço, de mercado financeiro e de expansão de mercado;
  - Políticas de impostos e taxas;
  - Políticas sociais e de saúde;
  - Campanhas eleitorais e outras disputas de poder entre facções políticas;
  - Práticas esportivas;
  - Dinâmica de comportamento animal.

TEORIA DOS JOGOS

- Os objetivos também são variados e podem envolver:
  - O tipo de resultado que pode ser obtido, dadas as estratégias dos jogadores;
  - A determinação da melhor estratégia a ser tomada por um dado jogador ou por todos os jogadores, dado o cenário que se apresenta;
  - O tipo de modelo que cada jogador deve estabelecer para os demais jogadores de modo que um dado resultado ocorra para o jogo.

TEORIA DOS JOGOS

- De fato, sempre que há uma disputa de interesse entre partes que possuem algumas alternativas para tomada de decisão, a formalização destes cenários é denominada jogo.
- A teoria dos Jogos é um Conjunto de técnicas para análise desses cenários. Ela não indica ao jogador como jogar o jogo, mas aponta o que acontece quando se adota esta ou aquela estratégia de jogo.

TEORIA DOS JOGOS

- Alguns jogos são demasiadamente complexos para serem completamente modelados.
- Assim, espera-se que um modelo simplificado seja capaz de descrever os principais tipos de decisão, assim como as estratégias mais indicadas e os resultados predominantes quando todos os jogadores fazem o melhor que podem a todo momento.
- Embora a denominação de jogo induza a conceitos como recreação e passa-tempo, as aplicações pretendidas envolvem um cenário bem mais abrangente, que excursiona do mais louvável ao mais ignóbil dos jogos.

Teoria dos Jogos

- Segundo a Teoria dos Jogos, os jogos são compostos de:
  - Jogadores
  - Ações
  - Matriz de Resultado
- Cada jogador pode adotar uma Estratégia (política, diretriz)
  - Estratégia Pura:
    - Diretriz determinística: uma ação para cada situação
  - Estratégia Mista:
    - Ações selecionadas de acordo com uma distribuição probabilística:
      - Ação *a* com probablidade *p* e ação *b*, caso *c*, caso contrário

      [*p*: *a*; (1- *p*): *b*]

Par ou Impar

- Dois jogadores O e E exibem simultaneamente um ou dois dedos.
- Seja f o número de dedos. Se f é ímpar, O recebe f dólares de E e se f é par, E recebe f dólares de O
- Matriz de Recompensa:

	Odd: um	**Odd: dois**
Even: um	E = 2, O = -2	E = -3, O = 3
Even: dois	E = -3, O = 3	E = 4, O = -4

Duelo

- Dois duelistas, posicionados a uma distância expressiva entre si, estão de posse de uma pistola carregada com uma única bala e passam a caminhar um em direção ao outro, na mesma velocidade.
- A cada passo dado, cada duelista pode decidir atirar ou não, sabendo que a chance de matar o seu oponente aumenta conforme a distância entre eles diminui.

LANÇAMENTO DE NOVOS PRODUTOS NO MERCADO

- Considere que duas empresas dividem o mercado junto a uma certa linha de produtos e que elas estão em constante disputa por ampliar sua fatia de mercado e pela redução de custos de produção.
- Se uma das empresas anuncia o lançamento de um produto revolucionário naquela linha, e o investimento para viabilizar a sua produção foi elevado, o comportamento da empresa concorrente pode ser de três tipos:

LANÇAMENTO
DE NOVOS PRODUTOS NO MERCADO

- Não lançar nenhum produto novo e prestigiar ainda mais os seus produtos já lançados no mercado, esperando pelo fracasso de mercado do produto concorrente;

- Passar a investir forte no lançamento de um novo produto muito semelhante àquele já lançado pela concorrência;

- Passar a investir forte no lançamento de um novo produto, distinto daquele já lançado pela concorrência, mas que concorre pela mesma fatia de mercado.

DILEMA DO PRISIONEIRO

- Duas pessoas são presas de posse de produtos roubados e elas são interrogadas separadamente pelas autoridades judiciais.
- Ambas sabem que: se ambas se declararem inocentes (não declarando ter roubado e nem acusando a outra pessoa de roubo), não há evidências suficientes para acusá-las de roubo, havendo uma pena de um ano de prisão por posse de produtos roubados

DILEMA DO PRISIONEIRO

- Se ambas delatarem uma a outra (acusar a outra de roubo e se declarar inocente), a pena será de 3 anos de prisão para cada uma;
- Se uma pessoa delatar a outra (acusar a outra de roubo e se declarar inocente), e a outra não delatar a primeira (inocentar a primeira), Então a primeira pessoa pega uma pena de serviços à comunidade,
- sendo solta imediatamente, e aquela que não delatou mas foi delatada pega uma pena de 5 anos de prisão.

DILEMA DO PRISIONEIRO

DILEMA DO PRISIONEIRO

- Quais são as semelhanças entre o lançamento de novos produtos no mercado e o dilema do prisioneiro?
- O que caracteriza uma estratégia de jogo neste caso?

Estratégias

- Pura
  - Cada jogador especifica uma ação que será tomada em cada situação (política determinística)
- Mista
  - Cada jogador seleciona ações particulares de acordo com as circunstâncias (política aleatória)
  - Escolhe uma ação a com probabilidade p e a ação b caso contrário [p:a; (1 - p):b]
    - Uma estratégia mista para o par ou ímpar poderia ser: [0,5: um; 0,5: dois]

Estratégias

- Perfil de estratégia
  - Atribuição de uma estratégia mista a cada um dos jogadores
- Resultado
  - Resultado numérico para cada jogador, dado o perfil de estratégia
- Solução
  - Perfil de estratégia em que cada jogador adota uma estratégia racional.
    - como cada agente escolhe a estratégia mais racional conhecendo apenas uma parte do perfil (global) de estratégia que determina o resultado?
- Exemplo: goleiro X atacante batendo um pênalti

Voltando ao Dilema do Prisioneiro

- A e B cometeram um crime
- Se confessarem pegam 5 anos de prisão
- Senão pegam 1 ano se for não for provado a culpa e 10 se for
- A polícia oferece um acordo separadamente a cada um deles:
  - Se delatar o parceiro, ficará livre

Dilema do prisioneiro

	Alice: testemunhar	Alice: recusar
Bob: testemunhar	A = -5, B =-5	A = -10, B = 0
Bob: recusar	A = 0, B= -10	A= -1, B = -1

Se nenhum dos dois suspeitos confessar eles serão acusados de receptação de roubo
Com pena de um ano de prisão mas se confessarem o roubo a pena é de 5 anos
Mas se delatar o companheiro é o líder ele pegará 10 anos de pena e o delator é perdoado

Dilema do prisioneiro

- Eles devem testemunhar ou se recusarem a testemunhar?
- Ou seja, qual estratégia adotar?

- Estratégia Dominante:
  - Uma estratégia qualquer *s* para o jogador *p* **domina fortemente** a estratégia *s'* se o resultado de *s* é melhor para *p* considerando-se (apesar de) **TODAS** as escolhas de estratégias pelos demais jogadores
  - Uma estratégia *s* para o jogador *p* **domina fracamente** a estratégia *s'* se o resultado de s é **melhor** que s' em **pelo menos um** perfil de estratégia e não é pior que em qualquer outro
  - **Estratégia Dominante** é uma estratégia que domina todas as outras
  - É irracional não usar uma estratégia dominante, caso exista

Dilema do prisioneiro

- Testemunhar é a estratégia dominante
  - Considera o caso racional do adversário e é a melhor para todos
- Resultado **Ótimo de Pareto**
  - Se não há nenhum outro resultado preferido por todos os jogadores
  - Um resultado é Dominado de Pareto por outro resultado se todos os jogadores preferem o outro resultado
- Equilíbrio de Nash
  - Se nenhum jogador pode se beneficiar com a troca (sozinho) de estratégia
- Equilíbrio de Estratégia Dominante
  - Quando cada jogador tem uma estratégia dominante, a combinação delas é um Equilíbrio de Estratégia Dominante

Dilema do prisioneiro

- Qual será a decisão de Alice se ela for racional? E de Bob?
  - Testemunhar (estratégia dominante)
- Equilíbrio de Estratégia Dominante:
  - Situação onde cada jogador possui uma estratégia dominante
- Então, eis que surge o dilema (porque é um dilema?):
  - Resultado para o ponto de equilíbrio é **Dominada de Pareto** pelo resultado: {recusar, recusar} !
  - Um resultado é dito **Dominada de Pareto** por outro se todos jogadores preferirem esse resultado
- Há alguma maneira de Alice e Bob chegarem ao resultado (-1, -1)?
  - Opção permitida mais pouco provável
  - Poder atrativo do ponto de equilíbrio !

Equilíbrio de Nash

- Equilíbrio de Nash:
  - *Agentes não possuem intenção de mudar de estratégia*
  - Condição necessária para uma solução
  - John Nash provou que todo jogo possui um equilíbrio assim definido
- Equilíbrio de Estratégia Dominante é um Equilíbrio de Nash
- Mas esse conceito afirma mais:
  - Existem estratégias que se equilibram mesmo que não existam estratégias dominantes

O EQUILÍBRIO DE NASH

- O dilema do prisioneiro é somente um exemplo de jogo.
- Em muitos jogos estratégicos não há a estrutura perversa que leva as estratégias dominantes a serem inferiores a outro resultado.
- Em um jogo, é necessário identificar o "**Equilíbrio de Nash**".

EQUILÍBRIO DE NASH

- **Equilíbrio de Nash** representa uma situação em que nenhum jogador **pode melhorar a sua situação** dada a **estratégia seguida pelo jogador adversário**.

- Um **EQUILÍBRIO DE NASH** é uma situação em que os agentes econômicos que estão interagindo entre si **escolhem**, para cada um deles, **a melhor estratégia para si** com base nas estratégias escolhidas pelos demais.

EQUILÍBRIO DE NASH

- Em um Equilíbrio de Nash, **nenhum** dos **jogadores tem incentivo** para alterar sua estratégia, desde que nenhum outro jogador possa escolher uma estratégia melhor, dadas as escolhas dos outros jogadores, ou seja, estamos diante de um **jogo não-cooperativo.**

EQUILÍBRIO DE NASH

- No equilíbrio de Nash, nenhum jogador se **arrepende de sua estratégia**, dadas as posições de todos os outros. Ou seja, um jogador não está necessariamente feliz com as estratégias dos outros jogadores, **apenas está feliz com a estratégia que escolheu** em face das escolhas dos outros.

Exemplo

- Exemplo:
  - Uma companhia de fabricante de hardware (Best) e outra de discos (ACME)
- Dois equilibrios de Nash: {dvd, dvd} e {cd, cd}
- Um equilíbrio Pareto Dominated: {dvd, dvd}
- Nenhum equilíbrio de estratégia dominante
- **Neste caso a melhor escolha deve ser a Pareto Dominated**

	DVD	**CD**
DVD	A = 9; B = 9	A = -1; B = -5
CD	A = -5; B = -1	A = 5; B = 5