

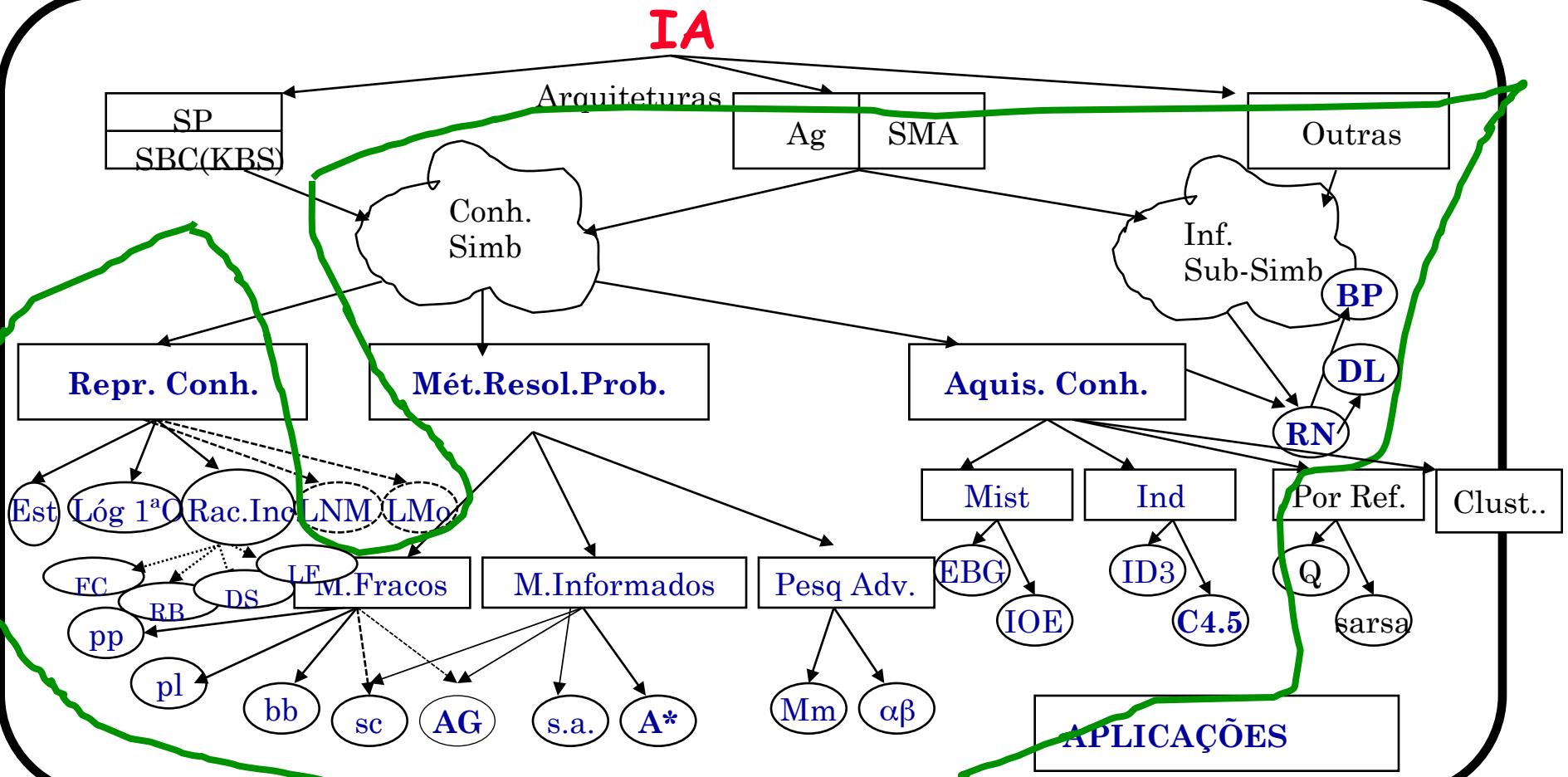
INTELIGÊNCIA ARTIFICIAL

II- MÉTODOS DE RESOLUÇÃO DE PROBLEMAS

- a) Motivação
- b) Características da pesquisa
- c) Representação dos Estados
- d) Algoritmos de pesquisa de soluções

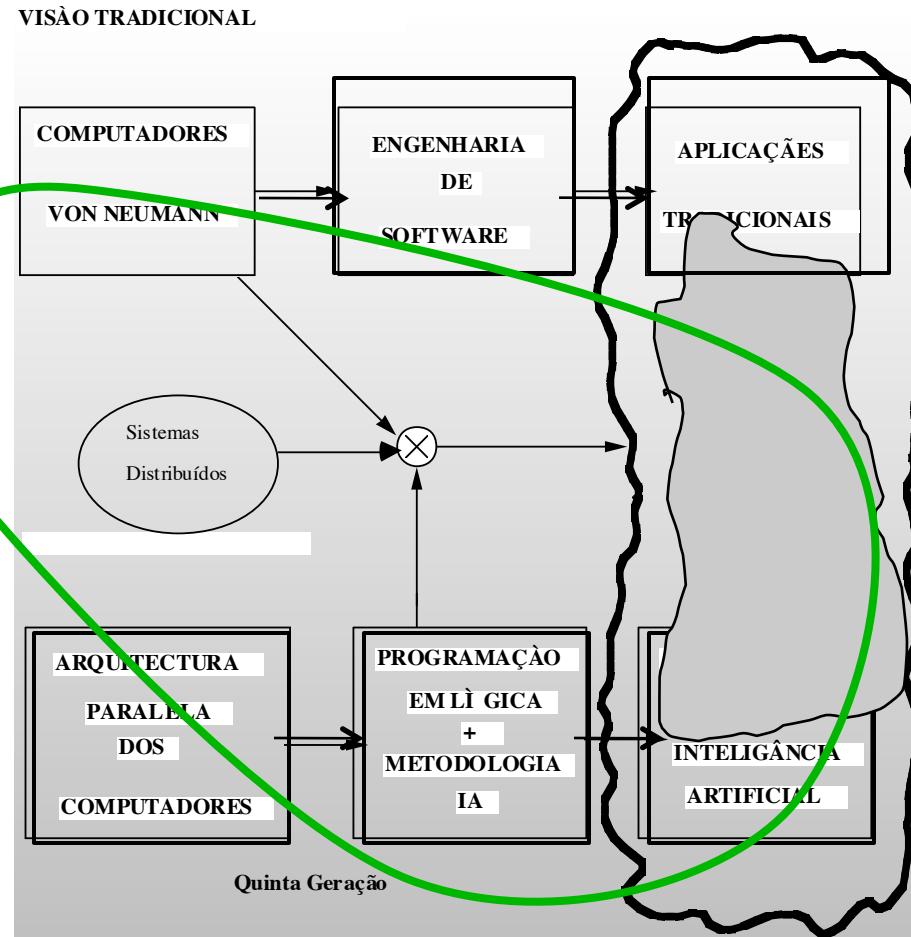
INTELIGÊNCIA ARTIFICIAL

a) Motivação



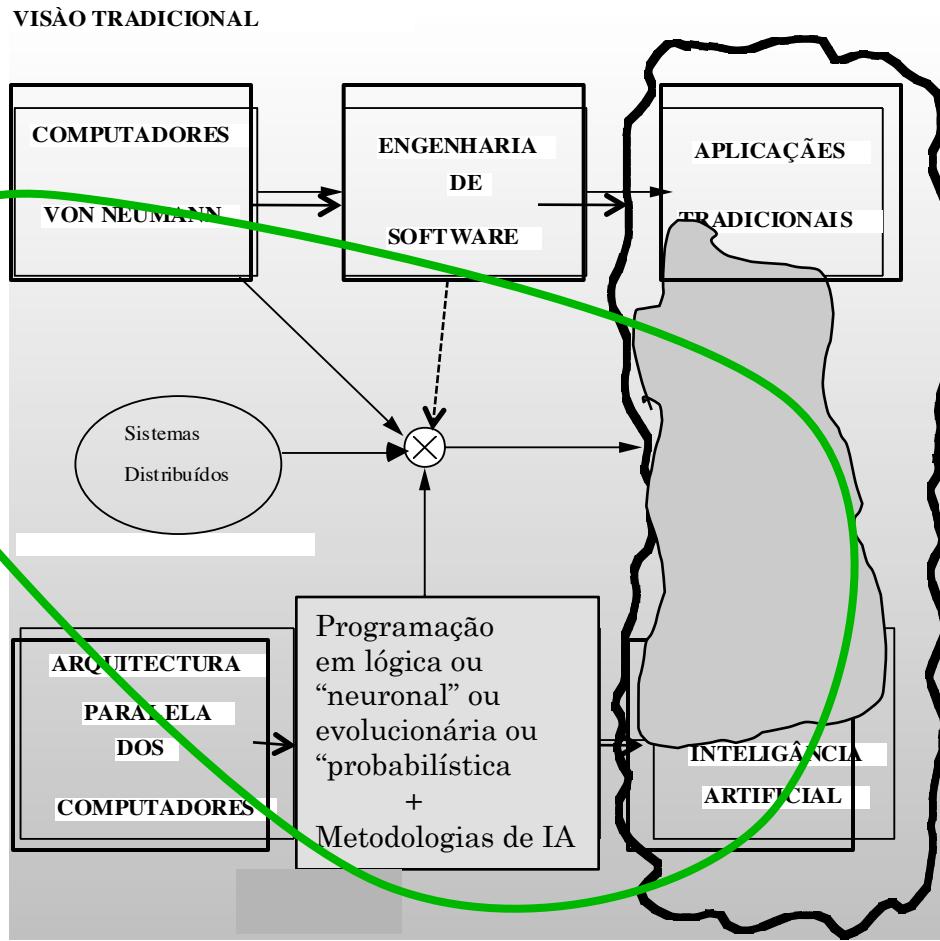
INTELIGÊNCIA ARTIFICIAL

a) Motivação



INTELIGÊNCIA ARTIFICIAL

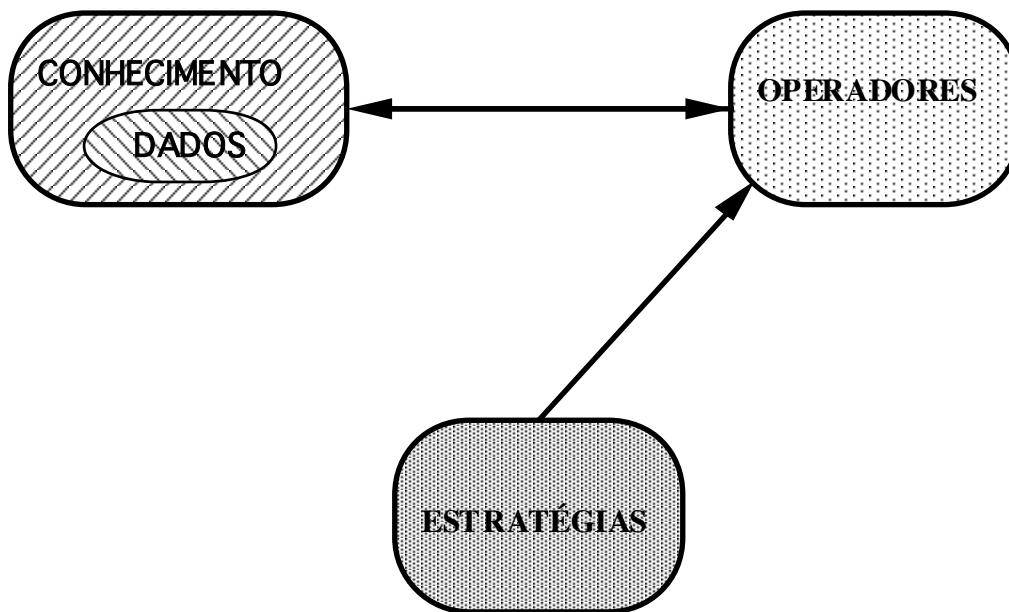
a) Motivação



INTELIGÊNCIA ARTIFICIAL

PARADIGMA BÁSICO DOS
SISTEMAS DE PRODUÇÕES

: “SEPARABILIDADE”



INTELIGÊNCIA ARTIFICIAL

MÉTODOS DE RESOLUÇÃO DE PROBLEMAS / pesquisa de solução

ARQUITETURA básica:

SISTEMAS DE PRODUÇÕES: a primeira Arquitetura básica dos Sistemas de IA

DESCRIÇÃO do problema:

Através de Estados em um Espaço de Estados (também funções objetivo)

REPRESENTAÇÃO do conhecimento:

Ex.: usando Lógica (de predicados, difusa, não monótona, modal) e de estruturas de informação

RESOLUÇÃO do problema: movimentação no espaço de estados da configuração Inicial à configuração Objetivo

ARQUITETURA do sistema computacional básico: uso do paradigma dos Sistemas de Produções como arquitetura básica para Sistemas Baseados em Conhecimento separando Factos, Operadores e Algoritmo de Controlo (estratégia)

INTELIGÊNCIA ARTIFICIAL

FORMULAÇÃO DOS PROBLEMAS

ESTADOS

ESTADO INICIAL

AÇÕES

MODELO de Transição (Efeitos)

TESTE do OBJETIVO

CUSTO do Caminho

Depende do paradigma: Heurístico, Metaheurístico, Evolucionário, RNA...

INTELIGÊNCIA ARTIFICIAL

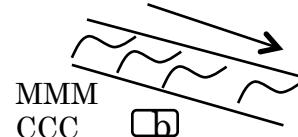
Problemas como Espaço de estados:

Exemplo dos Missionários e Canibais:

Representação do estado: *misscan(NMissMDir, NCanMDir, B).*

Estado inicial: *misscan(3, 3, d).*

Estado objetivo: *misscan(0, 0, e).*



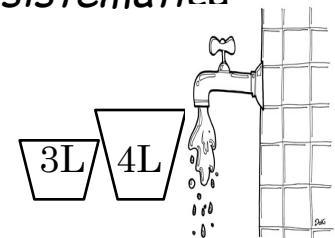
*Não existindo heurísticas , a pesquisa é sistemática
(com restrições para o estado solução).*

Exemplo dos Baldes:

Representação do estado: *baldes(NLb4, NLb3).*

Estado inicial: *baldes(0,0).*

Estado objetivo: *baldes(2,_).*



Regras (Operadores): *Encher, Esvaziar, Verter de um para outro até encher ou esvaziar,..*

INTELIGÊNCIA ARTIFICIAL

Ex: Puzzle de 8

Estados: Descrição da localização de cada algarismo e do branco

Operadores: Movimentação do branco para E,D,C,B

Teste: Estado igual ao Objetivo ?

Custo: Cada ação o mesmo custo de 1. Custo do Passo = N° acções

Estratégia: dependente de heurística

4	2	8
6	7	
1	3	4

Ex: 8 Rainhas (8-Queens) (ou N Rainhas)

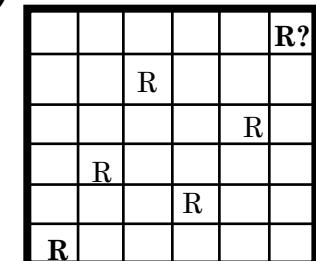
Estados: Descrição de qualquer arranjo de rainhas de 1 a 8 no Tabuleiro

Operadores: Colocar 1 rainha (na coluna à direita vazia não atacável)

Teste: 8 rainhas não atacáveis?

Custo: O passo não tem custo. Só interessa o estado final

Estratégia: tentativas ordenadas e com retrocesso sistemático



INTELIGÊNCIA ARTIFICIAL

b) Características da pesquisa de soluções e dos problemas

CARACTERÍSTICAS dos Sistemas de Produções **Puros**:

BD (F) --> *acessibilidade total*

OP (R) --> *independencia total*

EC --> *sistemática ou heurística*

Vantagens : *MODULARIDADE* facilita a descrição da pesquisa
mais fácil incrementar
Modelização das "acções guiadas pelos dados"

IA, como disciplina da **Ciência da Computação**, estuda as técnicas que resolvem problemas difíceis (por ex. com grau de complexidade **Exponencial**) em tempo **polinomial** explorando conhecimento do domínio do problema → **Controlo baseado em heurísticas**

INTELIGÊNCIA ARTIFICIAL

A IA como disciplina da **Ciência da Computação**, estuda as técnicas que resolvem problemas complexos (por ex. com grau de complexidade **Exponencial**) em tempo **polinomial**, explorando conhecimento do **domínio** do problema → Controlo da pesquisa das soluções **baseado em Heurísticas**

INTELIGÊNCIA ARTIFICIAL

ex: Caixeiro Viajante (Traveling Salesman Problem)

Pesquisa sistemática $\rightarrow (N-1)!$ Complexidade de ordem $O(N!)$

heurística:

- Selecionar (e fixar) arbitrariamente o primeiro nodo
- Comparar distâncias às próximas cidades não visitadas
- Repetir o passo anterior até visitar todas as cidades visitadas

$$(N-1)+(N-2)+ \dots + (N-(N-1)) = N-1 + N-2 + \dots + N-1 = 1/2N*(N-1)$$

Ex: $N=11, 10!=3\ 628\ 800$

$$10^2 = 100$$

ex: puzzle de 8

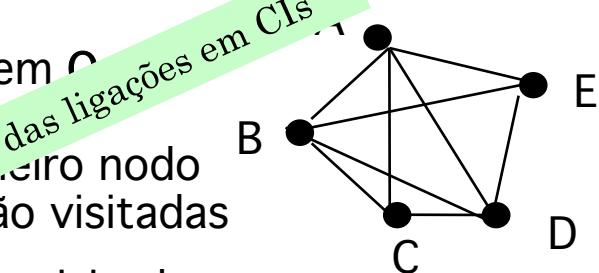
estados: configurações possíveis

regras: mover \leftarrow

Complexidade:

espaço de estados $9! = 362.880 \dots$ é muito!...

É necessário uma heurística



2	8	6
1		7
3	4	5

INTELIGÊNCIA ARTIFICIAL

Características do problema:

- a) decomponível?
- b) passos podem ser desfeitos? Ignorados?
- c) Universo previsível?
- d) Base dos Conhecimentos Coerente?
- e) Quer apenas a solução? (ou interacção)?

Exemplos:

- demonstração de teoremas (b) "passos ignoráveis")
- puzzle de 8 b)
- bridge não c)
- Acção de um Robô a)
- xadrez não b)

Planeamento ("Planning") pode simular a acção em problemas "irrecuperáveis"
Não b)

INTELIGÊNCIA ARTIFICIAL

Características dos Sistemas de Produções :

Monótonos: Aplicando $R_1 \in CRA$ isso não impede de mais tarde aplicar $R_i, \forall R_i \in CRA$ ($CRA = \text{conj}^o$ de Regras Aplicáveis no momento t_1)

Parcialmente comutativos: Se uma sequência de R_i 's leva dos estados A a B então qualquer permutação de R_i 's ainda aplicáveis, também leva de A a B

COMUTATIVO: MONÓTONO + PARCIALMENTE COMUTATIVO

Teoricamente: todos os S. Produções são aplicáveis a todos os problemas

Praticamente: Sistemas comutativos: melhores em problemas "ignoráveis". (não se fazem mudanças irrevogáveis. Criam-se outras). Ex. Demonstração de Teoremas.

S. Produções não monótonos e parcialmente comutativo: quando as coisas mudam mas podem ser "desfeitas" (puzzle 8)

INTELIGÊNCIA ARTIFICIAL

Agente simples de Resolução de problemas

Função *Ag_r_p_s(perceção)* retorna (*ação*)

Entrada: *perceção*

Estática: *seq* /* sequência de ações. Inicialmente vazia
estado
objetivo
problema /* uma formulação do problema

estado ← Atualiza_Estado(*estado*, *perceção*)

SE *seq* vazia ENTÃO

objetivo ← Formular_Objetivo(*estado_final*)
 problema ← Formular_Problema(*estado*, *objetivo*)
 seq ← Pesquisa(*problema*)

ação ← Cabeça(*seq*)
 seq ← Resto(*seq*)
 retorna (*ação*)

Executar

Pesquisar

Formular

INTELIGÊNCIA ARTIFICIAL

Existem **Métodos Fracos** e **Métodos Informados** para **pesquisa de soluções**.

- **Métodos Fracos:**

- Usam técnicas genéricas de pesquisa da solução independentes do problema.
- São sensíveis à explosão combinatória dos estados a pesquisar.

Outros fatores influenciando a escolha do método de pesquisa:

- Direcção da pesquisa, topologia da árvore,
- representação do conhecimento,

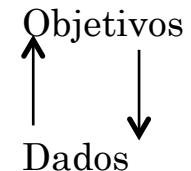
INTELIGÊNCIA ARTIFICIAL

-Direcção da pesquisa da(s) Solução (ões):

- Encadeamento Direto (“**forward** chaining”)

Vs

- Encadeamento inverso (“**backward** chaining”)



- Da situação inicial para o Objetivo ou vice-versa.

• Qual alternativa escolher:

Se o conjunto de objetivos possíveis é pequeno → Encadeamento inverso

Outras considerações:

- Fator de ramificação
- Similaridade com o processo real de raciocínio

INTELIGÊNCIA ARTIFICIAL

-Exemplos:

-**Demonstrador de Teoremas:** Encadeamento Inverso.

Deve partir-se do teorema (a partir dos axiomas muitos outros teoremas e corolários seriam deriváveis).

-**Sistema Pericial:** Naturalidade da inferência é importante.

-**Integração simbólica:** Encadeamento Direto.

Não faria sentido começar com um qualquer integral e verificar se seria o resultado da integração da expressão dada.

INTELIGÊNCIA ARTIFICIAL

-Árvores versus Grafos:

- O Espaço de Estados é implícito.
 - Só se vai explicitando os Estados à medida que se vai progredindo na pesquisa.
-
- Um Grafo pode transformar-se em Árvore:
 - aumenta o número de nodos, simplifica a identificação dos Passos.
 - Uma Árvore pode transformar-se em Grafo:
 - diminui número de nodos, complica os testes a fazer nos nodos.

INTELIGÊNCIA ARTIFICIAL

c) Representação do Conhecimento (Estados):

-Problema do Enquadramento ("Frame Problem")

Como representar de forma eficiente (uma sequência de) estados?

- Todo o estado de cada vez
- Manter a descrição do estado **inicial** e registrar as alterações
- Manter o estado **atual** por atualização do anterior e guardando o que deve ser desfeito/atualizado quando do retrocesso.

-Como registar o que não muda e o que se altera por efeito co-lateral?:

- Axiomas de Enquadramento

-Exemplos:

-não muda a cor quando se muda um objeto de posição...

-Tendo um objeto em cima de uma mesa e mudando a localização da mesa também muda a do que se mantém em cima.

INTELIGÊNCIA ARTIFICIAL

Funções de mérito/custo:

Associar aos possíveis estados seguintes um mérito/custo próprio calculado por uma função dependente do problema.

Ex:

Xadrez (medir vantagem de peças);

Puzzle-de-8 (nº de blocos no lugar certo);

Tic-Tac-Toe (linha ganhável já com 1 peça, com 2 peças).

NOTA: Balancear o custo do cálculo da função heurística e os ganhos na pesquisa devido ao seu uso.

INTELIGÊNCIA ARTIFICIAL

Sumariando esta primeira abordagem, diremos que nas Arquiteturas de Sistemas Computacionais Baseados em Conhecimento e em métodos da Inteligência Artificial:

- é fundamental manter em módulos *separados* as **estratégias de controlo** para a pesquisa de soluções e as estruturas de Representação do **Conhecimento**
- a escolha da **Estratégia de Pesquisa** está ligada à caraterização do problema

INTELIGÊNCIA ARTIFICIAL

- REVISÃO:

- Sistemas de Produções (ex. de SBC)
- Pesquisa de Soluções em Espaço de Estados
- Representação de Estados
- "Frame Problem"
- Características dos Sistemas de Produções e dos Problemas

- Estratégias sistemáticas
 - Direcção da pesquisa
 - Heurísticas.
 - Árvores Vs Grafos

INTELIGÊNCIA ARTIFICIAL

d) Algoritmos e MÉTODOS DE PESQUISA

MÉTODOS FRACOS (pouco ou nada informados):

"Gerar e testar"

"Primeiro em Profundidade"

"Primeiro em Largura"

"Ramifica e limita" ("Branch and Bound")

"Subir a colina" ("hill climbing")

Arrefecimento Simulado" ("simulated annealing")

Algoritmos Genéticos

Decomposição

Satisfação de restrições

"Análise meios Fins" ("Means Ends Analysis")

INTELIGÊNCIA ARTIFICIAL

d) Algoritmos e MÉTODOS DE PESQUISA

GERAR E TESTAR: Gera a possível solução e testa-a comparando com o objetivo
Geração sistemática: Primeiro em profundidade com "backtracking"
Combina-se com outras estratégias.
Ex: DENDRAL (gerar descrição de moléculas e testar com satisfação de restrições)

DENDRAL + METADENDRAL: “the formation of hypotheses of **organic molecular structure** from mass spectral data.” (Ed. Feigenbaum)

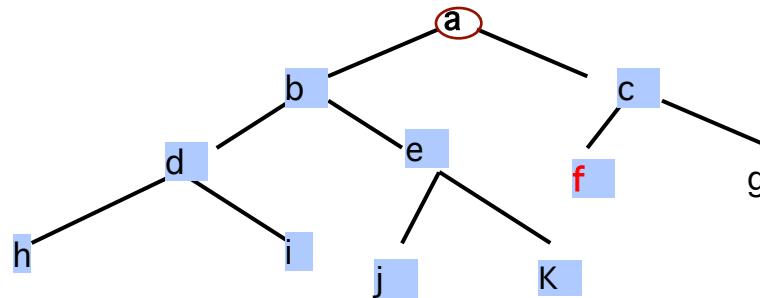
INTELIGÊNCIA ARTIFICIAL

- Direcções de pesquisa já mencionadas:
 - Encadeamento Inverso
 - Encadeamento Direto
 - Pesquisa Bidirecional
- Selecção dos filhos de cada nó (estado):
 - Primeiro em profundidade ("depth first")
 - Primeiro em Largura ("breadth first")
 - Retrocesso Sistemático ("Backtracking"): Quando, depois de se aplicar um operador a um estado, se desfaz essa operação e se tenta aplicar a um novo estado, esquecendo a anterior.
 - Retrocesso quando:
 - a) se gera estado já anteriormente visitado
 - b) se ultrapassa a aplicação de um número estipulado de operadores
 - c) não existem mais operadores aplicáveis ao nó em causa

INTELIGÊNCIA ARTIFICIAL

Algoritmos de Pesquisa de soluções

- Primeiro em profundidade ("depth first")
 - Expande sempre o nó a maior profundidade
 - Usa pouca memória:
nós folhas sem descendentes podem ser removidos



Outras variantes com o Aprofundamento Progressivo serão vistas mais à frente

INTELIGÊNCIA ARTIFICIAL

Variante do Primeiro-em- profundidade

ALGORITMO DE RETROCESSO SISTEMÁTICO (**BACKTRACKING**)

Função Backtrack (Lista_Dados)

{Lista_Dados = caminho; Dados= estado a analizar}

1 Dados <- cabeça(Lista_Dados)
2 SE membro(Dados, corpo(Lista_Dados)) ENTÃO falha
3 SE Dados= Objetivo ENTÃO fim (14)
4 SE Dados incoerentes c/solução ENTÃO falha
5 SE comprimento (Lista_Dados) > Limite ENTÃO falha
6 Operadores <- op_aplicáveis(Dados)
7 LOOP: SE Operadores = vazio ENTÃO falha
8 Op<- cabeça(Operadores)
9 Operadores <- corpo(Operadores)
10 Dados_op <- Op(Dados)
11 Lista_Dados_Op <- [Dados_Op, Lista_Dados]
12 Passo <- Backtrack(Lista_dados_Op)
13 SE Passo = falha ENTÃO LOOP
14 RETORNA [Op, Passo]

Dados= Estado

INTELIGÊNCIA ARTIFICIAL

Algoritmo "PRIMEIRO EM LARGURA"

- **Primeiro em largura:**

- todos os nós em cada nível da árvore de pesquisa são explorados primeiro

- **Vantagens:**

- Encontra sempre a **solução** se existir.

- Encontra a **melhor** em comprimento do passo.

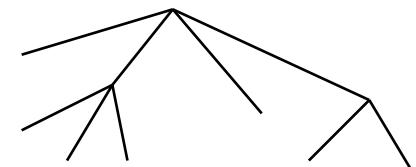
- **Desvantagem:** - tem de guardar todos os nós do nível.

- Implica mais **Memória**, trabalhoso.

- Havendo muitos passos que levam a soluções é preferível a estratégia "primeiro em profundidade"

- **"Pesquisa em feixe" ("beam search")**

- Método semelhante ao "primeiro em largura" mas, em cada nível não se expandem todos os nós mas somente alguns seleccionados (os melhores)



INTELIGÊNCIA ARTIFICIAL

Algoritmo Primeiro em Largura (breadth-first)

SE à cabeca da lista do 1º caminho aparece o objetivo
ENTAO esse caminho é a solução

SENÃO encontra o caminho mais curto
Remover esse 1º caminho da Lista
de caminhos candidatos e **gerar**
todos os caminhos resultantes
da expansão do seu nó cabeça
com novos sucessores imediatos
e junte esta nova lista de caminhos
candidatos na cauda da Lista anterior.

Execute de novo a estratégia
primeiro_em_largura.

Sucessão da Lista dos Caminhos:

$[[a]] \rightarrow [[b,a],[c,a]] \rightarrow [[c,a],[d,b,a],[e,b,a]] \rightarrow$
 $[[d,b,a],[e,b,a],[f,c,a],[g,c,a]] \rightarrow \dots$

INTELIGÊNCIA ARTIFICIAL

Análise do MÉTODO "Primeiro-em-Largura"

r factor de ramificação médio

p profundidade máxima (ou nível da solução)

r^p número de nós no último nível

Nº de nós da árvore de profundidade p, (p+1 níveis de 0 a p)

$$1+r+r^2+r^3+\dots+r^p = \sum_{i=0 \dots p} r^i = (r^{p+1}-1)/ (r-1)$$

Complexidade do algoritmo de pesquisa da solução primeiro_em_largura:
aproximadamente $O(r^p)$

Se expandir o nível seguinte $p+1$ até encontrar a solução em p: $O(r^{p+1})$

INTELIGÊNCIA ARTIFICIAL

Método do "**BRANCH AND BOUND**" ("ramifica-e-limita")

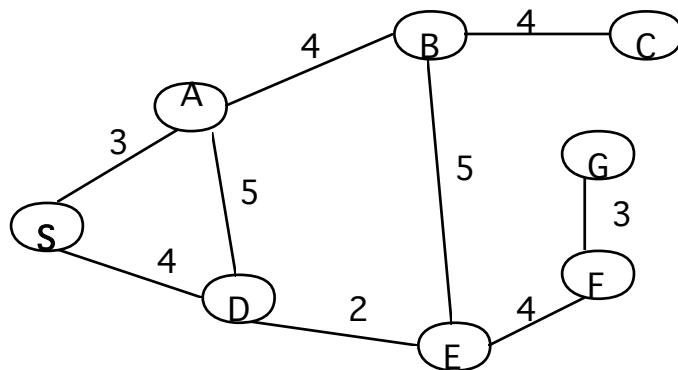
Pesquisa de custo uniforme

- Um processo de encontrar caminhos ótimos para a solução é a técnica de pesquisa chamada "Branch and Bound" (**Ramifica e Limita**)
- Em cada nível da árvore de pesquisa é continuado sempre o caminho até aí de **menor custo**.
- Após cada expansão comparam-se de novo os custos de todos os caminhos e volta-se a estender o de menor custo até ao momento.
- Repete-se o processo até o Objetivo ser encontrado.

INTELIGÊNCIA ARTIFICIAL

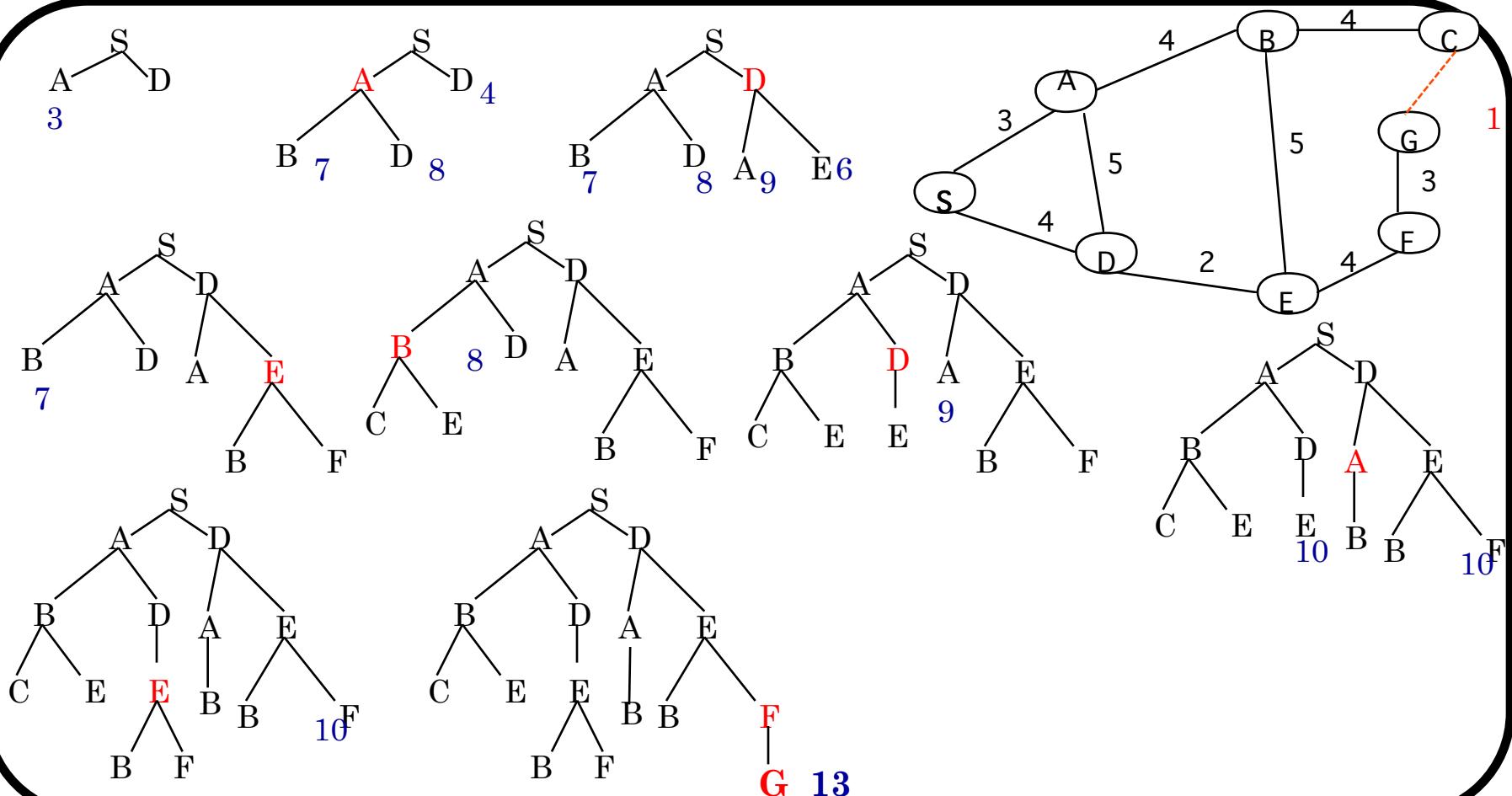
Método do **"BRANCH AND BOUND"** ("ramifica-e-limita")

- A primeira solução encontrada **poderá ser** a de menor custo.
Mas tem de se verificar isso.



- O Exemplo representa um mapa de estradas e respetivas distâncias.
O problema é encontrar o caminho mais curto entre S e G

INTELIGÊNCIA ARTIFICIAL



INTELIGÊNCIA ARTIFICIAL

ALGORITMO BÁSICO DO "BRANCH AND BOUND" ("ramifica-e-limita")

- 1 Formar uma Lista de caminhos parciais (caminhos de pesquisa da solução)
A Lista inicial tem comprimento 1 (estado inicial).
- 2 ATÉ Lista vazia OU Objetivo atingido
determinar se o 1º caminho da Lista atinge Objetivo
 - 2a SE sim ENTÃO ir para 3
 - 2b SE não
 - 2b1 retirar 1º caminho da Lista
 - 2b2 formar novos caminhos a partir do removido
aplicando os **operadores possíveis**
 - 2b3 acrescente os novos caminhos na Lista
 - 2b4 ordene **crescentemente** pelos **custos** acumulados
- 3 SE estado Objetivo alcançado ENTÃO sucesso /* pode ter de analizar caminhos
4 SENÃO falha /* com menor custo até então

INTELIGÊNCIA ARTIFICIAL

"Progressive /Iterative Deepening" **Aprofundamento Progressivo** (combinado comp_e_p)

Quando há **constrangimentos de tempo**, é necessário ter sempre uma avaliação presente para dar a melhor hipótese até ao momento.

Avalia-se qual a melhor das hipóteses para cada uma das profundidades limite da árvore de pesquisa gradualmente aumentada.

Chama-se **repetidamente** um algoritmo de pesquisa sistemática (**p_e_p**) para profundidade limitada mas com o **limite variando iterativamente** de 1 a p

(profundidade máxima que o tempo vai permitir ou nível da solução)

Não é demasiado custoso em tempo fazer estas avaliações!

INTELIGÊNCIA ARTIFICIAL

"Progressive /Iterative Deepening" **Aprofundamento Progressivo** (combinado comp_e_p)

r fator de ramificação média

P profundidade máxima

n numero total de nós requerendo avaliação

r^p número de nós no último nível (profundidade p, p+1 níveis de 0 a p)

Análise Temporal
 $O(r^p)$

$$1+r+r^2+r^3+\dots+r^p = (r^{p+1}-1)/(r-1)$$

nº de nós avaliados quando avaliamos uma vez até p (não expande p+1 como P_e_L)

INTELIGÊNCIA ARTIFICIAL

Aprofundamento Progressivo (Iterativo) P_em_P

outros nós requerendo avaliação:

a raiz p+1 vezes, (nível 1) r nodos p vezes, r^2 nodos p-1 vezes... r^p nodos 1 vez. (não expande)

$$N_{ap} = \sum_{i=0 \dots p} ((r^{i+1}-1)/ (r-1))$$

para valores elevados de r este somatório é aproximadamente $\sim (r^{p+2}-1)/ (r-1)^2$

A razão do nº de nós analisados em aprofundamento progressivo pelo nº de nós em p fixa: $((r^{p+2}-1)/ (r-1)^2) / ((r^{p+1}-1)/ (r-1)) \sim r/r-1$

Por ex: Se $r=10$, e $p=5$, o nº de nós até ao último nível seria 111.111 (prof. fixa)

Em aprofundamento progressivo teríamos 123.450

mas 11% de nós a calcular só!! Mesmo que repetissemos iterativamente a pesquisa.

ID (AP) é o método de pesquisa não informado preferível quando o espaço de procura é grande, a profundidade da solução desconhecida e o tempo limitado

INTELIGÊNCIA ARTIFICIAL

Aprofundamento Progressivo (Iterativo) P_em_P

outros nós requerendo avaliação adicional:

a raíz $p+1$ vezes, r nodos p vezes, r^2 nodos $p-1$ vezes... r^p nodos 1 vez.
(não expande)

$$N_{ap} = \sum_{i=0 \dots p} ((r^{i+1}-1)/ (r-1))$$

para valores elevados de r este somatório é *aproximadamente* $\sim (r^{p+2}-1)/ (r-1)^2$

A **razão** do nº de nós analisados em aprofundamento progressivo pelo nº de nós em p fixa: $((r^{p+2}-1)/ (r-1)^2) / ((r^{p+1}-1)/ (r-1)) \sim r/r-1$

INTELIGÊNCIA ARTIFICIAL

Comparação entre métodos **sistemáticos** p-e-l, p-e-p, c-uniforme (BB) e ap

	p-e-l	c-un	p-e-p	ap
tempo	r^{p+1}	$r^{1+(C^*/\varepsilon)}$	r^m	r^p
memória	r^{p+1}	$r^{1+(C^*/\varepsilon)}$	r^*m	r^*p
óptimo	s^*	s	N	s^*
Completo	s^{**}	s^{**}	N	s^{**}

r fator de ramificação média

p profundidade da solução

m profundidade máxima

C^* custo da solução óptima

ε limite inferior do custo de ação

$r^{1+(C^*/\varepsilon)} > r^p$ na maioria das vezes
(se custos iguais, então r^{p+1})

* se custos semelhantes ** se r finito

O algoritmo de custo uniforme não faz sentido ser analizado em termos de “p”
pois não se trata só da profundidade mas sim do custo dos passos! (C)

INTELIGÊNCIA ARTIFICIAL

Algoritmo “**SUBIR A COLINA**” (Hill Climbing)

% Método irrevogável. Gera hipótese e testa. Mas recebe “feedback” indicando qual a melhor direcção aconselhável para o próximo movimento. Não regide.
% Usa conhecimento local.

1-Considerar o Estado Inicial

2-SE solução ENTÃO RETORNAR
SENÃO

Aplicar a esse Estado os operadores possíveis

3- Para cada Novo estado gerado,

Testar proximidade com a solução

SE Solução RETORNAR
SENÃO

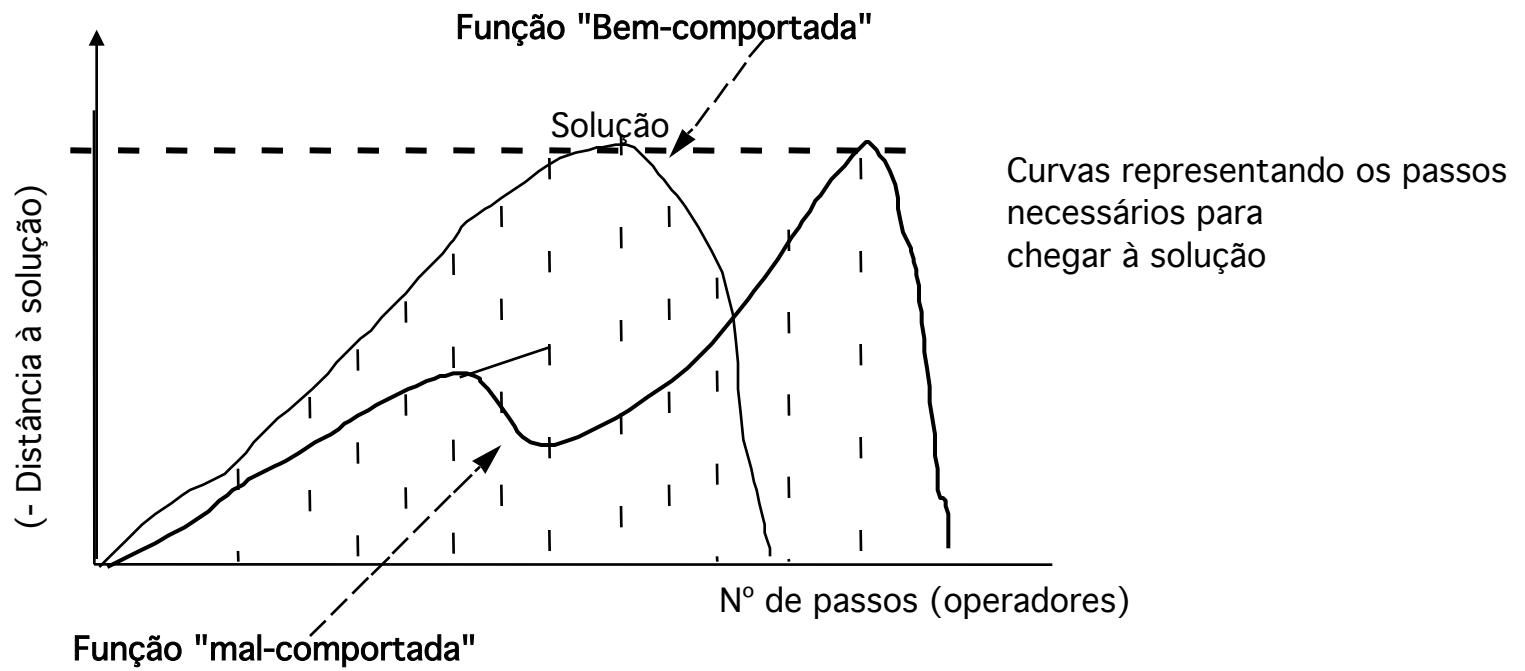
Seleccionar o Estado seguinte mais **próximo** da solução como sendo o Novo Estado

VOLTAR a 2

Problemas com este Algoritmo:

Funções (de custo) não “bem Comportadas” com “máximos locais” ou “planaltos”

INTELIGÊNCIA ARTIFICIAL



“Subir a Colina” nunca evolui na pesquisa descendo. Resulta ser INCOMPLETO!

INTELIGÊNCIA ARTIFICIAL

Dois tipos de “hill climbing” (subir-a-colina):

- 1) “Hill climbing” básico: Gera **um a um** os sucessores do estado atual
Encontrado um mais próximo da solução que o Estado Atual, selecciona-o e aplica-o

- 2) “Steepest ascent”: Gera **todos** os sucessores possíveis (“ascenção íngreme”) selecciona o mais próximo da solução

INTELIGÊNCIA ARTIFICIAL

a) scolina(Ef,_):- objetivo(Ef).
 scolina(Ea,[Ea | R]):- suc(Ea,ESucs),
 (ESucs==[], write('insucesso')
 ; select(ESucs,Eseg),
 scolina(Eseg, R)).

suc(Ea, Esuc):-
 findall(F, sucessor(Ea,F), Esucs).

select([E1],E1).
select([E1,E2 | OEs],Eseg):-
 estimativa(E1,C1),% estimativa de custos
 estimativa(E2,C2),
 (C1<C2, select([E1 | OEs],Eseg)
 ; select([E2 | OEs],Eseg)
).

INTELIGÊNCIA ARTIFICIAL

inicio(a).

objetivo(x).

sucessor(a,b).

sucessor(a,d).

sucessor(a,c).

sucessor(b,e).

sucessor(b,f).

sucessor(b,g).

sucessor(c,h).

sucessor(d,i).

sucessor(e,j).

...

estimativa(b,4).

estimativa(c,5).

estimativa(d,6).

estimativa(e,3).

...

Cálculo de sucessores e de estimativas pode ser implícito
e não descritos explicitamente como neste caso

INTELIGÊNCIA ARTIFICIAL

"ARREFECIMENTO SIMULADO" ("Simulated Annealing") ("Recristalização Simulada")
% "Annealing" é o processo de arrefecer um metal liquefeito até solidificar.
% Inputs: descrição do problema, variação do parâmetro "temperatura").
% A temperatura T controla a probabilidade de seleccionar estados que se afastem
% da solução

Função **SIMULATED_ANNEALING**(*problema, temperatura*) RETORNA Solução
Estado_corrente <-- estado_inicial(*problema*)
PARA $t=1$ ATÉ ∞ FAZER {*t, variável tempo*}
 T<-- variação[*t*] { Temperatura varia com o tempo}
 SE $T=0$ ENTÃO RETORNA Estado_Corrente
 Estado_próximo<-- qualquer Estado **Sucessor** do Estado_Corrente
 ΔE <-- valor_h[Estado_Próximo] - valor_h[Estado_Corrente]
 SE $\Delta E > 0$ ENTÃO Estado_Corrente <-- Estado_Próximo
 SENÃO Estado_Corrente <-- Estado_próximo com probabilidade $e^{-\Delta E / T}$

% SE variação baixar T suficientemente devagar, +provável encontrar um ótimo global
% Algoritmo não completo nem ótimo (como o subir-a-colina)

INTELIGÊNCIA ARTIFICIAL

"ARREFECIMENTO SIMULADO" ("Simulated Annealing") ("Recristalização Simulada")
% "Annealing" é o processo de arrefecer um metal liquefeito até solidificar.
% Inputs: descrição do problema, variação do parâmetro "temperatura").
% A temperatura T controla a probabilidade de seleccionar estados que se afastem
% da solução

Função **SIMULATED_ANNEALING**(problema, temperatura) RETORNA Solução
Estado_corrente <-- estado_inicial(problema)
PARA t=1 ATÉ ∞ FAZER
 T<-- variação[t]
 SE T=0 ENTÃO RETORNA Estado_Corrente
 Estado_próximo<-- qualquer Estado Sucessor do Corrente
 $\Delta E \leftarrow \text{valor_h}[\text{Estado_Próximo}] - \text{valor_h}[\text{Estado_Corrente}]$
 SE $\Delta E > 0$ ENTÃO Estado_Corrente <-- Estado_Próximo
 SENÃO Estado_Corrente <-- Estado_próximo com probabilidade $e^{-\Delta E / T}$

% SE variação baixa T suficientemente devagar, +provável encontrar um óptimo global
% **Algoritmo não completo nem óptimo** (como o subir-a-colina)

INTELIGÊNCIA ARTIFICIAL

- O Algoritmo “**arrefecimento simulado**” é uma variação do “**subir a colina**” em que, no **início** do processo de pesquisa da solução, alguns movimentos (hipóteses) mais afastadas da solução podem ser geradas (combina com o “random walk”).

Há variantes do algoritmo básico

- Torna assim a pesquisa mais independente dos pontos iniciais.
- A função heurística chama-se **função objetivo** se se pretender minimizar.
- Dever-se-á então dizer que se “**desce um vale**” em vez de “subir a colina”.
- É similar a uma fusão de metais que depois se arrefece baixando o nível de energia.
- Muito usada em problemas de escalonamento

INTELIGÊNCIA ARTIFICIAL

A aceitação de um novo estado que aparentemente se afaste da solução é feita com uma probabilidade $e^{\Delta E/T}$
(ou $e^{-\Delta E/T}$ dependendo da definição de ΔE sendo neste caso positivo)

- O algoritmo gera um número aleatório e verifica se é ou não menor que $e^{-\Delta E/T}$
(ou $e^{\Delta E/T}$)

Se sim aceita-se o estado mais afastado da solução.

Quanto maior a diferença (mais negativa, i.e. pior o estado, menor a prob. de ser escolhido)

$$e^{-|\Delta E|/T} = e^{1/|\Delta E|/T} = e^{T/|\Delta E|}$$

INTELIGÊNCIA ARTIFICIAL

REVISÃO

- Métodos de Resolução de Problemas

Métodos Sistemáticos de Pesquisa da solução:

- Direcção da pesquisa da solução

- Encadeamento Direto
- Encadeamento Inverso
- Bidirecional

- Seleção dos sucessores

- Métodos fracos:

- Gerar-e-testar (Ex:Dendral)
- primeiro -em-profundidade
- (retrocesso sistemático)
- primeiro-em-largura
- “Branch and Bound”
- Subir-a-colina. 2 versões
- Arrefecimento simulado

INTELIGÊNCIA ARTIFICIAL

REVISÃO

Métodos Sistemáticos de Pesquisa da solução:

"Primeiro-em-Largura"

Complexidade $O(r^{p+1})$

óptimo (*) e completo

variante: "pesquisa-em-feixe"

Aprofundamento progressivo

"Branch and Bound"

Comparação das complexidades: p-e-l, p-e-p, bb, ap

INTELIGÊNCIA ARTIFICIAL

Algoritmos para a Evolução

(Evolutionary Programming)

("Machine Learning - The Evolutionaries")

INTELIGÊNCIA ARTIFICIAL

- “*O que é bom para a Natureza é bom para os Sistemas Artificiais*”
- “The Origin of Species on the basis of Natural Selection”
- Charles Darwin → Marco histórico do Conhecimento
- Organismos adaptados ao meio reproduzem-se
- Sucessores tem um grau de similitude com os antepassados
- Mutações aleatórias podem ter ou não sucesso
- **Evolução e Mutação** permitem adaptação em ambientes (moderadamente) dinâmicos

INTELIGÊNCIA ARTIFICIAL

Computação baseada na Evolução:

- Porquê?

A Natureza permite a Evolução bem sucedida de organismos através de **selecção e reprodução** com alguma **mutação**

Reprodução de estados mais adaptados

Computação Evolucionária (Friedberg, 1958 ; JHolland, 1975):

- Quando?

Pesquisa paralela em várias árvores

- Espaço de Pesquisa grande e complexo
- Facilidade de pesquisa paralela
- Não necessidade de solução ótima (mas boa)

- Exemplo:

otimizar uma função $f(x_1, \dots, x_{100})$ sendo f muito complexa.
se $x_i=0$ ou $x_i=1$, $x_i \in \{0,1\}$ Espaço de pesquisa $2^{100} \approx 10^{30}$
Pesquisa exaustiva fora de questão

INTELIGÊNCIA ARTIFICIAL

Representação funcional do Algoritmo para Evolução

Função A_E(população, Função_Adaptação)

REPETIR

 pais \leftarrow SELEÇÃO(população, Função_Adaptação)

 população \leftarrow REPRODUÇÃO(pais)

 ATÉ encontrar algum Indivíduo suficientemente apto

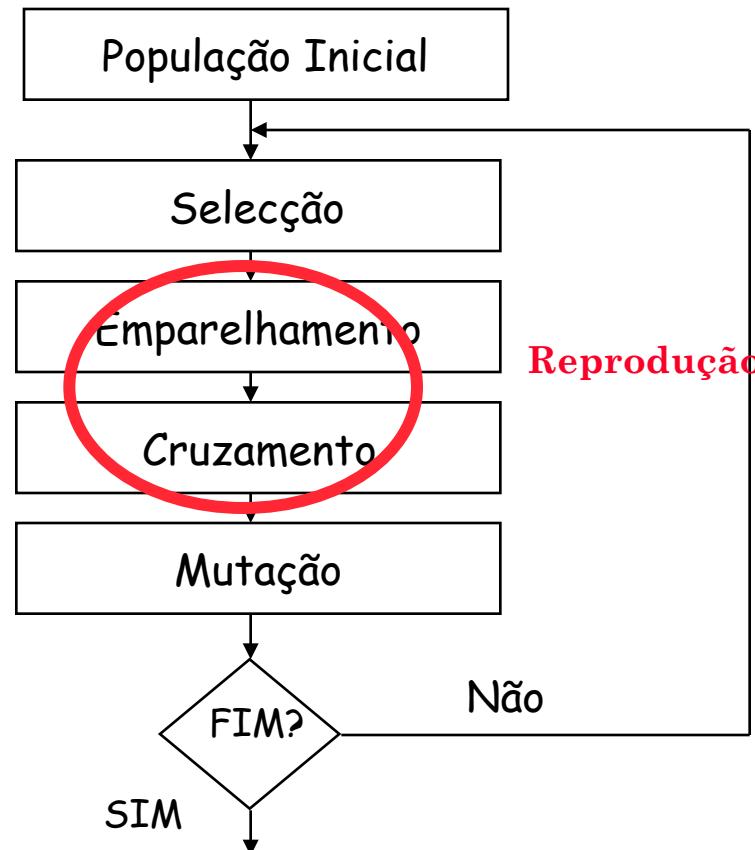
RETORNAR o indivíduo mais apto segundo a função de adaptação

Elemento da população=Estado

- **Algoritmos para a Evolução são um método de pesquisa pouco informada, “paralela” e estocástica**

INTELIGÊNCIA ARTIFICIAL

Diagrama do Algoritmo para Evolução



INTELIGÊNCIA ARTIFICIAL

Questões:

- Qual a Função de Adaptação (*Fitness Function*)?
- Como representar os Indivíduos?
- Como Selecionar os Indivíduos?
- Como se reproduzem os Indivíduos?

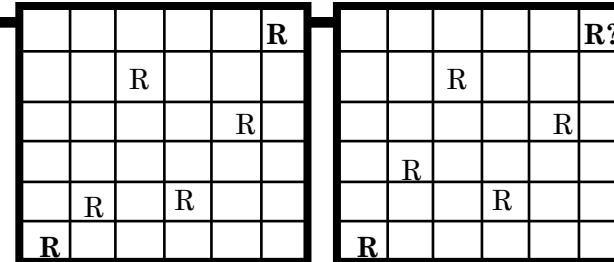
INTELIGÊNCIA ARTIFICIAL

Qual a Função de Adaptação?

- Depende do problema
- Recebe um Indivíduo e dá como resultado um Real
(Melhores Estados (Indivíduos) → maiores valores)
- *Por ex:* A função pode representar o número de exemplos, na população, com os quais um indivíduo é consistente (**homogeneização**) ou o contrário (**heterogeneização**);
- *nº de exemplos classificados corretamente (problema de divisão em Classes)*

INTELIGÊNCIA ARTIFICIAL

Qual a Função de Adaptação?



Por ex: Problema das N-raínhas não atacáveis em um Tabuleiro de Xadrez (aqui N=6).

Cromossoma com N Genes cada um representando a posição de uma Raínya no tabuleiro (8 X 8)

(Vector com N=8 dígitos de 1 a 8
ou

com $8 \cdot \log_2 8$ bits)

1 3 6 8 2 7 4 5

000 010 101 111 001 110 011 100

Função de Adaptação: inversamente proporcional ao somatório dos ataques sofridos por cada Raínya

INTELIGÊNCIA ARTIFICIAL

- Qual a Função de Adaptação?
 - Depende do problema
 - Recebe um Indivíduo e dá como resultado um Real
- *ex:* Treinar o disparo de arma controlada por um AE sobre um alvo móvel.
 - » Depende de variáveis como vento, velocidade do alvo, distância, tipo de arma ... (Estado= Cromossoma com as vars)
 - » Função de Adaptação nos **testes experimentais**: dadas as condições da decisão do tiro, qual a distância a que o projétil ficou do alvo
 - » É um exemplo de problema em que os AE/AGs deram bons resultados

INTELIGÊNCIA ARTIFICIAL

Como representar os Indivíduos?

- Indivíduos representados como cadeias de caracteres (strings) sobre um alfabeto.
- Podem usar-se matrizes, vectores -
- Indivíduos são os Genomas, cromossomas.
- Cada instância do indivíduo é um genótipo codificando a origem do fenótipo de um indivíduo.
 - No ADN (DNA) o Alfabeto é AGTC (Adenina, Guanina, Timina, Citosina)
 - Cada elemento do indivíduo (do cromossoma) é um Gene.
 - N° de Genes depende do N° de Atributos a representar
 - Se a Representação dos indivíduos é binária temos Algoritmos Genéticos $A = \{0, 1\}$ 0 e 1 são os “alelos”

Genótipo: conjunto de Genes recebidos do pai e da mãe. São esses genes, mais as influências do meio, que determinarão o fenótipo de um ser.

ADN - ácido desoxirribonucleico

INTELIGÊNCIA ARTIFICIAL

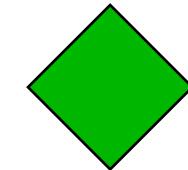
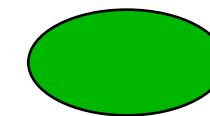
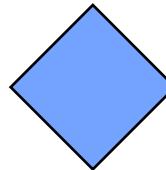
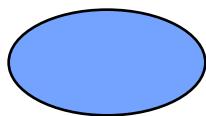
Como representar os Indivíduos?

Podem usar-se vetores

Indivíduos são Cromossomas.

Algoritmos Genéticos $A=\{0,1\}$

Nº de genes depende do Nº de Atributos a representar



2 atributos: cor(2) e formato (3)

cor: 1 bit e formato 2bits

0	1	0
---	---	---

Modelos estatísticos $f(x_1, \dots, x_n)$: Estimativa de parâmetros.

- Escalar valor das variáveis para inteiros multiplicando por 10^n sendo n o número de decimais (precisão desejada): Nova Variável = inteiro(val anterior da variável * 10^n)

Transforme a nova variável para a forma binária

INTELIGÊNCIA ARTIFICIAL

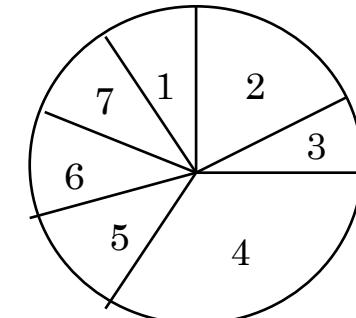
- **Selecção:**

- Estratégia incluindo **aleatoriedade**.
- Probabilidade da selecção **proporcional** à adaptabilidade observada.
- Se indivíduo x é 2 vezes melhor que y , terá o dobro das **probabilidades** de ser seleccionado para reprodução
- $fa(C_i)$ dá a adaptação do Cromossoma C_i e $\sum_{i=1}^{aN} fa(C_i)$ a soma das adaptações de toda a população

Probabilidade de C_i ser seleccionado é $fa(C_i) / \sum_{i=1}^{aN} fa(C_i)$

Existem outros métodos de selecção (ex: selecção **elitista**)

Elitista: o(s) melhor(es) da geração n passa(m) sempre para a geração $n+1$

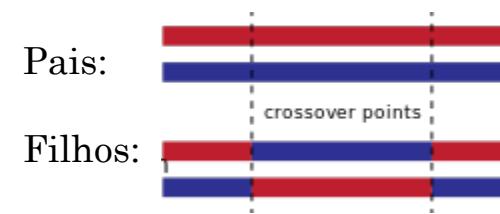
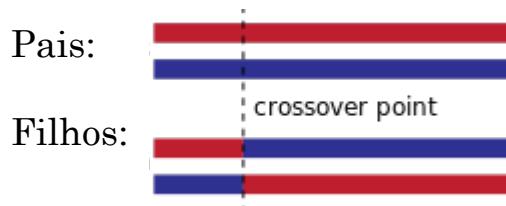


Ex. de Roleta (gira N vezes)

INTELIGÊNCIA ARTIFICIAL

Como se reproduzem os Indivíduos/Cromossomas?

- por Cruzamento e Mutação.
- Indivíduos seleccionados com probabilidade P_s são emparelhados aleatoriamente.
- Para cada par, com uma dada probabilidade P_{cr} , (pelo menos) um ponto de corte ("crossover") é escolhido, de 1 a N sendo N o comprimento do Cromossoma representando o indivíduo. O material genético é trocado tendo como referência o ponto de corte. Outra estratégia é usar $C > 1$ pontos de corte.



INTELIGÊNCIA ARTIFICIAL

Como se reproduzem os Indivíduos/Cromossomas?

Ainda outras estratégias de cruzamento:

“Crossover” Uniforme (UX) e “Crossover” Meio Uniforme (HUX)

UX permite que os pais contribuam para os descendentes ao nível do Gene e não do segmento.

Se a razão da mistura for 0.5, os descendentes tem cerca de metade dos genes de cada um dos pais. Mas essa razão pode ser diversa

Pais:



Filhos:

UX avalia cada bit nos cromossomas dos pais, sendo diferentes troca-os com a probabilidade de 0,5.

UX “parece” ser melhor que a troca de segmentos. Permite uma melhor exploração do espaço de estados possíveis. Não está provado teoricamente. HUX troca metade dos bits diferentes.

INTELIGÊNCIA ARTIFICIAL

Como se reproduzem os Indivíduos/Cromossomas?

estratégias de cruzamento “Crossover” Uniforme UX

UX permite que os pais contribuam para os descendentes ao nível do Gene e não do segmento.

Pode gerar-se uma “máscara” e trocar ou não em conformidade:

Progenitor 1 1 0 1 0 1 0 1 1 1 1 0 1 0 1

Progenitor 2 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1

Máscara 1 1 1 0 0 1 1 0 0 0 1 1 0 1 0

FILHO 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1

FILHO 2 1 0 1 1 0 0 1 0 0 1 1 0 1 0 1

UX “parece” ser melhor que a troca de segmentos. Permite uma melhor exploração do espaço de estados possíveis. Não está provado teoricamente.

INTELIGÊNCIA ARTIFICIAL

Como se reproduzem os Indivíduos/Cromossomas?

Na estratégia (“esquema”) de cruzamento Meio-Uniforme” (HUX), exatamente metade dos bits **diferentes** são trocados.

Portanto, primeiro calcula-se D, a distância de “Hamming” (número de bits diferentes). O número de bits trocados daqueles que não são iguais (nos pais) é exatamente D/2.

- Um GENE também pode sofrer **Mutação** para um valor diferente, mas com uma pequena probabilidade P_m

INTELIGÊNCIA ARTIFICIAL

Como se reproduzem os Indivíduos/Cromossomas?

- Cruzamento com 1 ponto de corte, gerado aleatoriamente:

• Geração N (progenitores) Geração N+1 (descendentes)

1010 | 101111

1010 00000

0011 | 000000

0011 101111

- Cruzamento com 2 pontos de corte, também gerados aleatoriamente:

• Geração N (progenitores) Geração N+1 (descendentes)

1010 | 101111 | 10101

1010 000001 10101

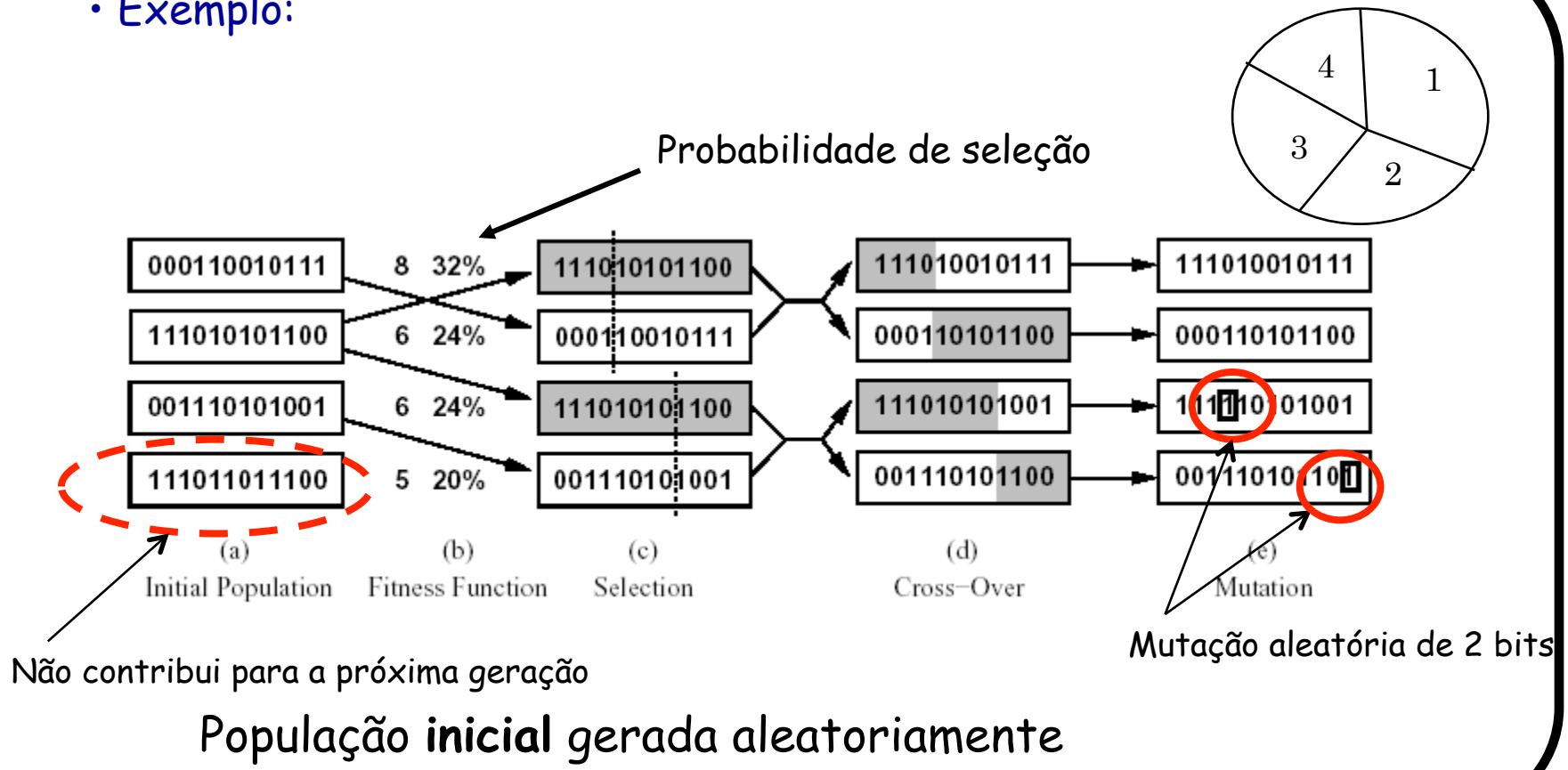
0011 | 000001 | 11111

0011 101111 11111

- Um GENE também pode sofrer **Mutação** para um valor diferente, mas com uma pequena probabilidade P_m

INTELIGÊNCIA ARTIFICIAL

- Exemplo:



INTELIGÊNCIA ARTIFICIAL

Funções Multimodais tem mais que um máximo e podem ter só um ou vários máximos globais que são difíceis (**Hard**) de calcular.

Como operam os Algoritmos Genéticos?

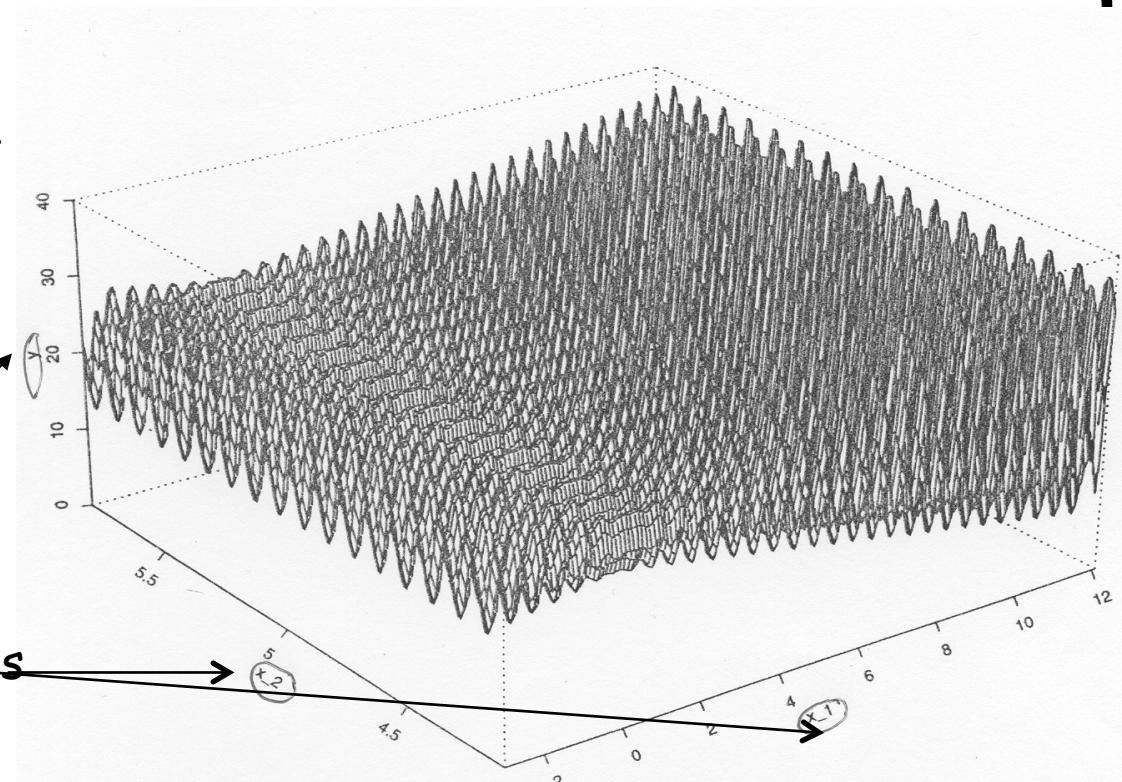
Exemplo de problema:

$$\text{Max } f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$$

Onde: $-3.0 \leq x_1 \leq 12.1$ e
 $4.1 \leq x_2 \leq 5.8$.

Valor da função
Valores das Variáveis

Exemplo de Z.Michalewicz
U.North Carolina, USA



INTELIGÊNCIA ARTIFICIAL

Assumir que a precisão é de 4 casas decimais para cada variável.

- $x_1 \in [-3.0, 12.1]$.

Este intervalo é dividido em 15.1×10000 intervalos de tamanhos iguais.

Dado que $2^{17} < 151000 \leq 2^{18}$,

18 bits são necessários para representar a variável x_1 .

- $x_2 \in [4.1, 5.8]$.

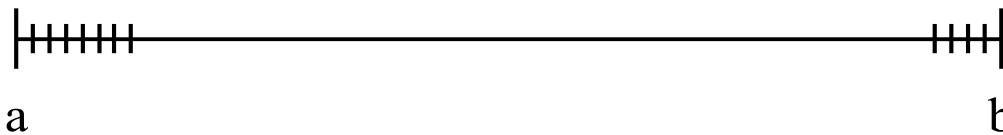
Este intervalo é dividido em 1.7×10000 intervalos de tamanhos iguais.

Dado que $2^{14} < 17000 \leq 2^{15}$,

15 bits são necessários para representar a variável x_2 .

INTELIGÊNCIA ARTIFICIAL

Representação dos indivíduos



- String (000...0) corresponde ao **a**
- String (111...1) corresponde ao **b**
- String (i_1, \dots, i_k) corresponde a (K bits):

$$a + \text{decimal } (i_1, \dots, i_k) * (b - a) / 2^k - 1$$

(aproximado pois 2^k dá mais posições que as necessárias)

Ex: 64 pontos a partir da posição 8 (até 72)
Usam-se 6 bits (2^6)
O indivíduo representado por 111111
corresponde a:
 $8 + (63 * (72-8) / 2^6 - 1) = 72$

INTELIGÊNCIA ARTIFICIAL

Vamos considerar a string de $18 + 15 = 33$ bits:

ex: (010001001011010000111110010100010).

Os primeiros 18 bits, 010001001011010000,

Representam $x_1 = -3.0 + \text{decimal}(010001001011010000_2) \times 12.1 - (-3.0) / 2^{18} - 1$

$$= -3.0 + 70352 \times 15.1 / 262143 = -3.0 + 4.052426 = 1.052426$$

INTELIGÊNCIA ARTIFICIAL

Os próximos 15 bits,

111110010100010, representam

$$x_2 = 4.1 + \text{decimal}(111110010100010_2) \times 5.8 - 4.1 / 2^{15} - 1$$

$$= 4.1 + 31906 \times 1.7 / 32767 = 4.1 + 1.655330 = 5.755330$$

INTELIGÊNCIA ARTIFICIAL

Portanto o cromossoma:

(01000100101101000011110010100010)

Corresponde a:

$$\langle x_1, x_2 \rangle = \langle 1.052426, 5.755330 \rangle.$$

$$f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$$

O valor de adaptação para este cromossoma é:

$$f(1.052426, 5.755330) = 20.252640$$

Para optimizar a função f usando um algoritmo genético, criamos uma **população de dimensão = 20 cromossomas**.

Todos os 33 bits em todos os 20 cromossomas são inicializados **aleatoriamente**.

INTELIGÊNCIA ARTIFICIAL

População
de 20 cromossomas (v_i)
inicializados aleatoriamente

$$\begin{aligned}\vec{v}_1 &= (100110100000011111101001101111) \\ \vec{v}_2 &= (11100010010011011001010100011010) \\ \vec{v}_3 &= (00001000011001000001010111011101) \\ \vec{v}_4 &= (100011000101101001111000001110010) \\ \vec{v}_5 &= (00011101100101001101011111000101) \\ \vec{v}_6 &= (0001010001001010100101011111011) \\ \vec{v}_7 &= (00100010000110101111011011111011) \\ \vec{v}_8 &= (100001100001110100010110101100111) \\ \vec{v}_9 &= (01000000010110001011000001111100) \\ \vec{v}_{10} &= (000001111000110000011010000111011) \\ \vec{v}_{11} &= (011001111110110101100001101111000) \\ \vec{v}_{12} &= (110100010111101101000101010000000) \\ \vec{v}_{13} &= (111011111010001000110000001000110) \\ \vec{v}_{14} &= (010010011000001010100111100101001) \\ \vec{v}_{15} &= (11101110110110000100011111011110) \\ \vec{v}_{16} &= (110011110000011111100001101001011) \\ \vec{v}_{17} &= (01101011111001111010001101111101) \\ \vec{v}_{18} &= (01110100000001110100111110101101) \\ \vec{v}_{19} &= (0001010100111111110000110001100) \\ \vec{v}_{20} &= (10111001011001111001100010111110)\end{aligned}$$

Resultados da avaliação
(sobre a adaptação) de
cada um dos cromossomos
iniciais

$$\begin{aligned} -3.0 \leq x_1 \leq 12.1 & \quad \text{e} \\ 4.1 \leq x_2 \leq 5.8. & \end{aligned}$$

- $eval(\vec{v}_1) = f(6.084492, 5.652242) = 26.019600$
- $eval(\vec{v}_2) = f(10.348434, 4.380264) = 7.580015$
- $eval(\vec{v}_3) = f(-2.516603, 4.390381) = 19.526329$
- $eval(\vec{v}_4) = f(5.278638, 5.593460) = 17.406725$
- $eval(\vec{v}_5) = f(-1.255173, 4.734458) = 25.341160$
- $eval(\vec{v}_6) = f(-1.811725, 4.391937) = 18.100417$
- $eval(\vec{v}_7) = f(-0.991471, 5.680258) = 16.020812$
- $eval(\vec{v}_8) = f(4.910618, 4.703018) = 17.959701$
- $eval(\vec{v}_9) = f(0.795406, 5.381472) = 16.127799$
- $eval(\vec{v}_{10}) = f(-2.554851, 4.793707) = 21.278435$
- $eval(\vec{v}_{11}) = f(3.130078, 4.996097) = 23.410669$
- $eval(\vec{v}_{12}) = f(9.356179, 4.239457) = 15.011619$
- $eval(\vec{v}_{13}) = f(11.134646, 5.378671) = 27.316702$
- $eval(\vec{v}_{14}) = f(1.335944, 5.151378) = 19.876294$
- $eval(\vec{v}_{15}) = f(11.089025, 5.054515) = 30.060205$
- $eval(\vec{v}_{16}) = f(9.211598, 4.993762) = 23.867227$
- $eval(\vec{v}_{17}) = f(3.367514, 4.571343) = 13.696165$
- $eval(\vec{v}_{18}) = f(3.843020, 5.158226) = 15.414128$
- $eval(\vec{v}_{19}) = f(-1.746635, 5.395584) = 20.095903$
- $eval(\vec{v}_{20}) = f(7.935998, 4.757338) = 13.666916$

INTELIGÊNCIA ARTIFICIAL

Se usassemos uma estratégia de Seleção “**Elitista**” poderíamos **consevar** sempre na próxima Geração alguns (n_1) dos mais bem adaptados.

Consideram-se com maior **probabilidade** para o cruzamento os n_2 seguintes mais bem adaptados

$$n_1 + n_2 = N_{\text{total}}$$

INTELIGÊNCIA ARTIFICIAL

Probabilidade dos indivíduos serem seleccionados

$$F = \sum_{i=1}^{20} eval(\vec{v}_i) = 387.776822.$$

Considera-se uma Função de Escala F

$$p_1 = eval(\vec{v}_1)/F = 0.067099 \leftarrow$$

$$p_2 = eval(\vec{v}_2)/F = 0.019547$$

$$p_3 = eval(\vec{v}_3)/F = 0.050355$$

$$p_4 = eval(\vec{v}_4)/F = 0.044889$$

$$p_5 = eval(\vec{v}_5)/F = 0.065350$$

$$p_6 = eval(\vec{v}_6)/F = 0.046677$$

$$p_7 = eval(\vec{v}_7)/F = 0.041315$$

$$p_8 = eval(\vec{v}_8)/F = 0.046315$$

$$p_9 = eval(\vec{v}_9)/F = 0.041590$$

$$p_{10} = eval(\vec{v}_{10})/F = 0.054873$$

INTELIGÊNCIA ARTIFICIAL

$$p_{11} = eval(\vec{v}_{11})/F = 0.060372$$

$$p_{12} = eval(\vec{v}_{12})/F = 0.038712$$

$$p_{13} = eval(\vec{v}_{13})/F = 0.070444$$

$$p_{14} = eval(\vec{v}_{14})/F = 0.051257$$

$$p_{15} = eval(\vec{v}_{15})/F = \boxed{0.077519} \leftarrow \text{Valor máximo}$$

$$p_{16} = eval(\vec{v}_{16})/F = 0.061549$$

$$p_{17} = eval(\vec{v}_{17})/F = 0.035320$$

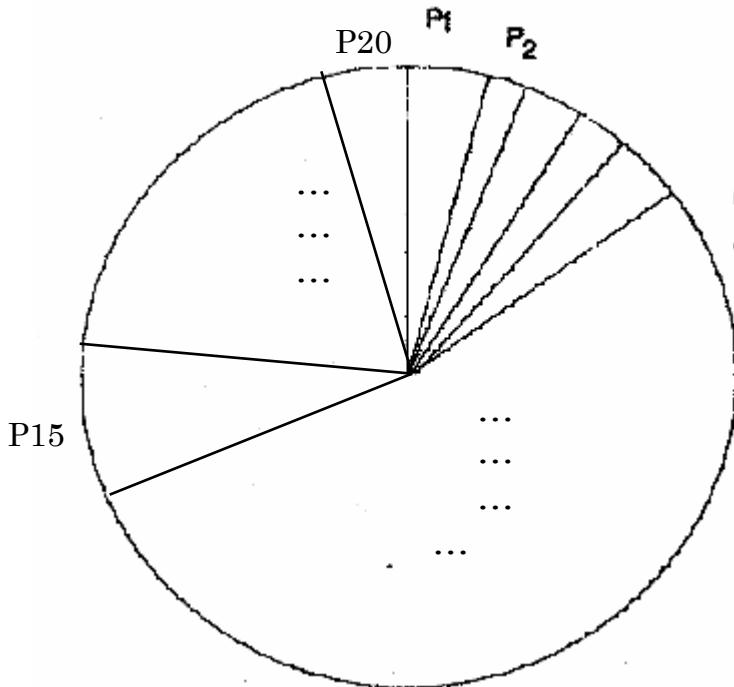
$$p_{18} = eval(\vec{v}_{18})/F = 0.039750$$

$$p_{19} = eval(\vec{v}_{19})/F = 0.051823$$

$$p_{20} = eval(\vec{v}_{20})/F = 0.035244$$

INTELIGÊNCIA ARTIFICIAL

“Roleta” para escolher aleatórios



Somatório das probabilidades =1
P20 termina o sector circular
onde inicia p1

INTELIGÊNCIA ARTIFICIAL

Repartição dos intervalos na “roleta”
para a seleção dos cromossomas (entre 0 e 1)

$$q_1 = 0.067099 \quad q_2 = 0.086647 \quad q_3 = 0.137001$$

$$q_4 = 0.181890 \quad q_5 = 0.247240 \quad q_6 = 0.293917$$

$$q_7 = 0.335232 \quad q_8 = 0.381546 \quad q_9 = 0.423137$$

$$q_{10} = 0.478009 \quad q_{11} = 0.538381 \quad q_{12} = 0.577093$$

$$q_{13} = 0.647537 \quad q_{14} = 0.698794 \quad q_{15} = 0.776314$$

$$q_{16} = 0.837863 \quad q_{17} = 0.873182 \quad q_{18} = 0.912932$$

$$q_{19} = 0.964756 \quad q_{20} = 1.000000$$

(Pequenos erros na última casa decimal)

INTELIGÊNCIA ARTIFICIAL

Resultados da geração de números aleatórios:

0.513870 0.175741 0.308652 0.534534 0.947628
0.171736 0.702231 0.226431 0.494773 0.424720
0.703899 0.389647 0.277226 0.368071 0.983437
0.005398 0.765682 0.646473 0.767139 0.780237

$q_1 = 0.067099 \quad q_2 = 0.086647 \quad q_3 = 0.137001$
 $q_4 = 0.181890 \quad q_5 = 0.247240 \quad q_6 = 0.293917$
 $q_7 = 0.335232 \quad q_8 = 0.381546 \quad q_9 = 0.423137$
 $q_{10} = 0.478009 \quad q_{11} = 0.538381 \quad q_{12} = 0.577093$
 $q_{13} = 0.647537 \quad q_{14} = 0.698794 \quad q_{15} = 0.776314$
 $q_{16} = 0.837863 \quad q_{17} = 0.873182 \quad q_{18} = 0.912932$
 $q_{19} = 0.964756 \quad q_{20} = 1.000000$

Por exemplo:

os 1º e 4º números caem na secção da roleta correspondente a V_{11}
o 2º a v_4 e o 3º na correspondente a v_7 ... V_{15} aparece 4 vezes.

Se a política fosse **elitista**, conservar-se-iam os n1 melhores cromossomas
e só se geravam 20- n1 Números (por Ex: n1=2, geravam-se 18)

INTELIGÊNCIA ARTIFICIAL

População seleccionada v'

De notar que alguns cromossomas da população inicial **não são seleccionados** para reprodução(V_2) e outros são-nos várias vezes (por ex. V_{11} e V_{15})

Desaparece na próxima geração

- $\vec{v}'_1 = (0110011111011010110000110111000) \ (\vec{v}_{11})$
- $\vec{v}'_2 = (10001100010110100111000001110010) \ (\vec{v}_4)$
- $\vec{v}'_3 = (0010001000001101011101101111011) \ (\vec{v}_7)$
- $\vec{v}'_4 = (0110011111011010110000110111000) \ (\vec{v}_{11})$
- $\vec{v}'_5 = (0001010100111111110000110001100) \ (\vec{v}_{19})$
- $\vec{v}'_6 = (10001100010110100111000001110010) \ (\vec{v}_4)$
- $\vec{v}'_7 = (11101110110111000010001111011110) \ (\vec{v}_{15})$
- $\vec{v}'_8 = (00011101100101001101011111000101) \ (\vec{v}_5)$
- $\vec{v}'_9 = (01100111110110101100001101111000) \ (\vec{v}_{11})$
- $\vec{v}'_{10} = (000010000011001000001010111011101) \ (\vec{v}_3)$
- $\vec{v}'_{11} = (11101110110111000010001111011110) \ (\vec{v}_{15})$
- $\vec{v}'_{12} = (010000000101100010110000001111100) \ (\vec{v}_9)$
- $\vec{v}'_{13} = (00010100001001010100101011111011) \ (\vec{v}_6)$
- $\vec{v}'_{14} = (100001100001110100010110101100111) \ (\vec{v}_8)$
- $\vec{v}'_{15} = (101110010110011110011000101111110) \ (\vec{v}_{20})$
- $\vec{v}'_{16} = (1001101000000011111110100110111111) \ (\vec{v}_1)$
- $\vec{v}'_{17} = (000001111000110000011010000111011) \ (\vec{v}_{10})$
- $\vec{v}'_{18} = (111011111010001000110000001000110) \ (\vec{v}_{13})$
- $\vec{v}'_{19} = (111011101101110000100011111011110) \ (\vec{v}_{15})$
- $\vec{v}'_{20} = (110011110000011111100001101001011) \ (\vec{v}_{16})$

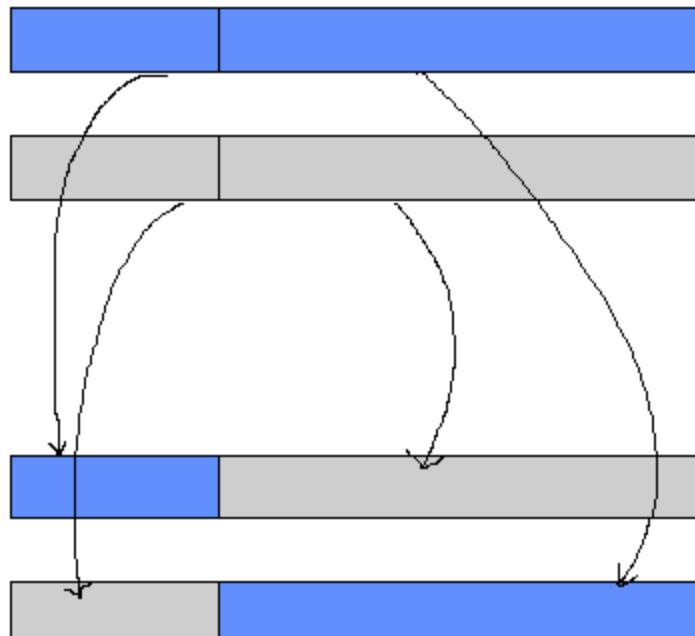
INTELIGÊNCIA ARTIFICIAL

Antes

Reprodução por Cruzamento

1 ponto de "Crossover"

Depois



INTELIGÊNCIA ARTIFICIAL

Há outras versões

Assumindo probabilidade de Cruzamento

$P_c = 0.25$ (poderia ser bastante maior, p.ex: 0.75 ou até 1)

Números aleatórios gerados:

0.822951 0.151932 0.625477 0.314685 0.346901

0.917204 0.519760 0.401154 0.606758 0.785402

0.031523 0.869921 0.166525 0.674520 0.758400

0.581893 0.389248 0.200232 0.355635 0.826927

Assim as gerações
não são
completamente
diferentes.

Há alguma
continuidade na
evolução

Números aleatórios inferiores a P_c são o 2°, 11°, 13°, 18°
Serão estes os realmente emparelháveis

INTELIGÊNCIA ARTIFICIAL

Reprodução por Cruzamento de v'_2 com v'_{11} e de v'_{13} com v'_{18} :
Ponto de cruzamento entre o 9º e o 10º bit, aleatório)

pais	$\vec{v}'_2 = (100011000 101101001111000001110010)$ $\vec{v}'_{11} = (111011101 10111000010001111011110)$
filhos	$\vec{v}''_2 = (100011000 10111000010001111011110)$ $\vec{v}''_{11} = (111011101 101101001111000001110010)$.

Pontos de cruzamento variam.

pais	$\vec{v}'_{13} = (00010100001001010100 101011111011)$ $\vec{v}'_{18} = (11101111101000100011 0000001000110)$
filhos	$\vec{v}''_{13} = (00010100001001010100 0000001000110)$ $\vec{v}''_{18} = (11101111101000100011 101011111011)$.

Condições diversas do cruzamento
Ponto de cruzamento entre o 20º e o 21º bit, aleatório)

Novos cromossomas gerados : v''_2 , v''_{11} , v''_{13} e v''_{18} :

INTELIGÊNCIA ARTIFICIAL

Nova população de cromossomas →

$$\vec{v}_1 = (01100111110110101100001101111000)$$

$$\vec{v}_2' = (10001100010111000010001111011110)$$

$$\vec{v}_3' = (00100010000011010111101101111011)$$

$$\vec{v}_4 = (01100111110110101100001101111000)$$

$$\vec{v}_5 = (0001010100111111110000110001100)$$

$$\vec{v}_6 = (100011000101101001111000001110010)$$

$$\vec{v}_7 = (11101110110111000010001111011110)$$

$$\vec{v}_8 = (00011101100101001101011111000101)$$

$$\vec{v}_9 = (01100111110110101100001101111000)$$

$$\vec{v}_{10} = (000010000011001000001010111011101)$$

$$\rightarrow \vec{v}_{11}' = (1110111011011001111000001110010)$$

$$\vec{v}_{12} = (010000000101100010110000001111100)$$

$$\rightarrow \vec{v}_{13}' = (000101000010010101000000001000110)$$

$$\vec{v}_{14} = (100001100001110100010110101100111)$$

$$\vec{v}_{15} = (10111001011001111001100010111110)$$

$$\vec{v}_{16} = (100110100000001111111010011011111)$$

$$\vec{v}_{17} = (000001111000110000011010000111011)$$

$$\rightarrow \vec{v}_{18}' = (111011111010001000111010111111011)$$

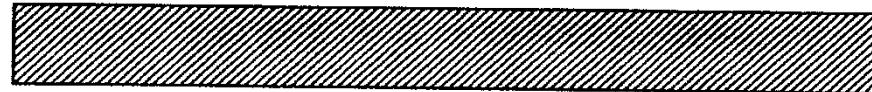
$$\vec{v}_{19} = (1110111011011100001000111110111110)$$

$$\vec{v}_{20} = (110011110000011111100001101001011)$$

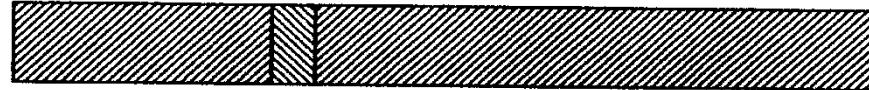
INTELIGÊNCIA ARTIFICIAL

Mutação

Antes



Depois



INTELIGÊNCIA ARTIFICIAL

Assumir, $p_m = 0.01$ como a probabilidade de mutação.

Dos números gerados aleatoriamente, os seguintes são < p_m

(isto é : do 1º ao 111º foram gerados nºs superiores a p_m , etc...)

Bit position	Random number
112	0.000213
349	0.009945
418	0.008809
429	0.005425
602	0.002836

	Posi	Chromo	Bit pos
112	4		13
349	11		19
418	13		22
429	13		33
602	19		8

O bit na posição 112 da População é o 13º bit de V'_4

O bit na posição 349 da População é o 19º bit de V''_{11}

Mutações

- $\vec{v}_1 = (01100111110110101100001101111000)$
- $\vec{v}_2 = (100011000101110000100011111011110)$
- $\vec{v}_3 = (00100010000011010111101101111011)$
- $\vec{v}_4 = (0110011111010101100001101111000)$
- $\vec{v}_5 = (0001010100111111110000110001100)$
- $\vec{v}_6 = (100011000101101001111000001110010)$
- $\vec{v}_7 = (111011101101110000100011111011110)$
- $\vec{v}_8 = (00011101100101001101011111000101)$
- $\vec{v}_9 = (01100111110110101100001101111000)$
- $\vec{v}_{10} = (000010000011001000001010111011101)$
- $\vec{v}_{11} = (111011101101101001011000001110010)$
- $\vec{v}_{12} = (010000000101100010110000001111100)$
- $\vec{v}_{13} = (000101000010010101000100001000111)$
- $\vec{v}_{14} = (100001100001110100010110101100111)$
- $\vec{v}_{15} = (101110010110011110011000101111110)$
- $\vec{v}_{16} = (1001101000000011111110100110111111)$
- $\vec{v}_{17} = (000001111000110000011010000111011)$
- $\vec{v}_{18} = (111011111010001000111010111111011)$
- $\vec{v}_{19} = (1110111001011100001000111110111110)$
- $\vec{v}_{20} = (110011110000011111100001101001011)$

Resultados da avaliação da nova população

O novo melhor cromossoma é \vec{v}_{11}

\vec{v}_{11} tem melhor avaliação (33.351874)

do que o anterior melhor,

\vec{v}_{15} (30.060205)

$eval(\vec{v}_1) = f(3.130078, 4.996097) = 23.410669$
$eval(\vec{v}_2) = f(5.279042, 5.054515) = 18.201083$
$eval(\vec{v}_3) = f(-0.991471, 5.680258) = 16.020812$
$eval(\vec{v}_4) = f(3.128235, 4.996097) = 23.412613$
$eval(\vec{v}_5) = f(-1.746635, 5.395584) = 20.095903$
$eval(\vec{v}_6) = f(5.278638, 5.593460) = 17.406725$
$eval(\vec{v}_7) = f(11.089025, 5.054515) = 30.060205$
$eval(\vec{v}_8) = f(-1.255173, 4.734458) = 25.341160$
$eval(\vec{v}_9) = f(3.130078, 4.996097) = 23.410669$
$eval(\vec{v}_{10}) = f(-2.516603, 4.390381) = 19.526329$
$eval(\vec{v}_{11}) = f(11.088621, 4.743434) = 33.351874$
$eval(\vec{v}_{12}) = f(0.795406, 5.381472) = 16.127799$
$eval(\vec{v}_{13}) = f(-1.811725, 4.209937) = 22.692462$
$eval(\vec{v}_{14}) = f(4.910618, 4.703018) = 17.959701$
$eval(\vec{v}_{15}) = f(7.935998, 4.757338) = 13.666916$
$eval(\vec{v}_{16}) = f(6.084492, 5.652242) = 26.019600$
$eval(\vec{v}_{17}) = f(-2.554851, 4.793707) = 21.278435$
$eval(\vec{v}_{18}) = f(11.134646, 5.666976) = 27.591064$
$eval(\vec{v}_{19}) = f(11.059532, 5.054515) = 27.608441$
$eval(\vec{v}_{20}) = f(9.211598, 4.993762) = 23.867227$

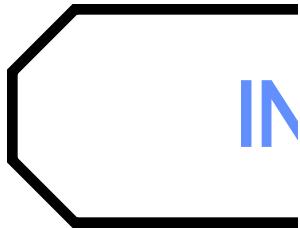
INTELIGÊNCIA ARTIFICIAL

Note que a adaptação total da nova população:

F (logo após uma nova geração) é **447.049688**, muito maior do que a adaptação total da população anterior, **387.776822**.

Também, o melhor cromossoma agora (v_{11}) tem uma melhor avaliação (**33.351874**) do que o melhor cromossoma (v_{15}) da população anterior (**30.060205**).

Depois de 1000 gerações...



$\vec{v}_1 = (11101111011001101110010101010111011)$
 $\vec{v}_2 = (111001100110000100010101010111000)$
 $\vec{v}_3 = (11101111011101101110010101010111011)$
 $\vec{v}_4 = (111001100010000110000101010111001)$
 $\vec{v}_5 = (11101111011101101110010101010111011)$
 $\vec{v}_6 = (111001100110000100000100010100001)$
 $\vec{v}_7 = (110101100010010010001100010110000)$
 $\vec{v}_8 = (111101100010001010001101010010001)$
 $\vec{v}_9 = (111001100010010010001100010110001)$
 $\vec{v}_{10} = (11101111011101101110010101010111011)$
 $\vec{v}_{11} = (110101100000010010001100010110000)$
 $\vec{v}_{12} = (110101100010010010001100010110001)$
 $\vec{v}_{13} = (11101111011101101110010101010111011)$
 $\vec{v}_{14} = (111001100110000100000101010111011)$
 $\vec{v}_{15} = (111001101010111001010100110110001)$
 $\vec{v}_{16} = (111001100110000101000100010100001)$
 $\vec{v}_{17} = (111001100110000100000101010111011)$
 $\vec{v}_{18} = (111001100110000100000101010111001)$
 $\vec{v}_{19} = (111101100010001010001110000010001)$
 $\vec{v}_{20} = (11100110011000010000010101010111001)$



$eval(\vec{v}_1) = f(11.120940, 5.092514) = 30.298543$
 $eval(\vec{v}_2) = f(10.588756, 4.667358) = 26.869724$
 $eval(\vec{v}_3) = f(11.124627, 5.092514) = 30.316575$
 $eval(\vec{v}_4) = f(10.574125, 4.242410) = 31.933120$
 $eval(\vec{v}_5) = f(11.124627, 5.092514) = 30.316575$
 $eval(\vec{v}_6) = f(10.588756, 4.214603) = 34.356125$
 $eval(\vec{v}_7) = f(9.631066, 4.427881) = 35.458636$
 $eval(\vec{v}_8) = f(11.518106, 4.452835) = 23.309078$
 $eval(\vec{v}_9) = f(10.574816, 4.427933) = 34.393820$
 $eval(\vec{v}_{10}) = f(11.124627, 5.092514) = 30.316575$
 $eval(\vec{v}_{11}) = f(9.623693, 4.427881) = \underline{\underline{35.477938}}$
 $eval(\vec{v}_{12}) = f(9.631066, 4.427933) = \overline{35.456066}$
 $eval(\vec{v}_{13}) = f(11.124627, 5.092514) = 30.316575$
 $eval(\vec{v}_{14}) = f(10.588756, 4.242514) = 32.932098$
 $eval(\vec{v}_{15}) = f(10.606555, 4.653714) = 30.746768$
 $eval(\vec{v}_{16}) = f(10.588814, 4.214603) = 34.359545$
 $eval(\vec{v}_{17}) = f(10.588756, 4.242514) = 32.932098$
 $eval(\vec{v}_{18}) = f(10.588756, 4.242410) = 32.956664$
 $eval(\vec{v}_{19}) = f(11.518106, 4.472757) = 19.669670$
 $eval(\vec{v}_{20}) = f(10.588756, 4.242410) = 32.956664$

$f(x_1, x_2)$

Valor máximo encontrado

INTELIGÊNCIA ARTIFICIAL

Questões:

Qual a cardinalidade da população inicial? (diversidade Vs complexidade)

Representação binária ou não?

Quais os melhores valores para as probabilidades de Cruzamento e Mutação?

Quais as estratégias de Cruzamento a usar?
Elitismo?

INTELIGÊNCIA ARTIFICIAL

Questões:

Há diferença no exemplo entre geração aleatória e o AG?

No fim os cromossomas tendem a ser mais similares que no início.
Resultado do cruzamento é mais diverso no início.
Tal como no “simulated annealing”, maior variabilidade do passo seguinte no início que no fim.

do cromossoma:

(0100010010110100 0011110010100010)

e do cromossoma:

(1001101000000011 1111101001101111)

resultam:

(010001001011010011 11101001101111)

e

(100110100000001100 111110010100010)

Portanto, os indivíduos gerados **não são assim tão diferentes** como se a geração fosse aleatória (mas depende dos pontos de “crossover”)

INTELIGÊNCIA ARTIFICIAL

- **Problema do contentor:** Encher com o maior valor possível sem exceder o peso máximo (50).
- **EX: 10 objetos com os seguintes valores e pesos:**

$$V[i] = \{10, 50, 20, 10, 40, 10, 30, 5, 5, 10\}$$

$$P[i] = \{20, 10, 10, 5, 10, 15, 5, 5, 5, 5\}$$

Função de adaptação $\sum_{i=1}^{10} (V[i] * X_j[i])$ - Penalização para evitar indivíduos que são falsas hipóteses

Cromossomas com 10 bits (0 não incluir objeto i, 1 incluir)

População inicial aleatória:

X1 1111000111

X2 0000111111

X3 1100100001



Peso da carga ≤ 50

INTELIGÊNCIA ARTIFICIAL

Problemas "NP-difíceis" e onde podem ser gerados cromossomos inválidos

- **Problema do contentor/mochila:** Encher com o maior **valor** possível sem exceder o **peso** máximo (50).
- EX: 10 objetos com os seguintes valores e pesos:

$$V[i] = \{10, 50, 20, 10, 40, 10, 30, 5, 5, 10\}$$

$$P[i] = \{20, 10, 10, 5, 10, 15, 5, 5, 5, 5\}$$

Knapsack problem



Função de adaptação $\sum_{i=1}^{10} (V[i]*X_j[i])$ - Penalização

Peso da carga ≤ 50

$$\text{Penalização}(X_j) = R * ((\sum_{i=1}^{10} X_j[i]*P[i]) - C_{max})^2$$

$$C_{max}=50 \quad R=\max_{i=1}^{10} (V[i]/P[i]) \text{ tal que } X_j[i]=0$$

Cromossomas com 10 bits (0 não incluir objeto i, 1 incluir)

População inicial aleatória:

X1 1111000111

Conteúdo do Contentor

X2 0000111111

X3 1100100001

INTELIGÊNCIA ARTIFICIAL

Função de adaptação $\sum_{i=1 \text{ a } 10} (V[i]*X[i])$ - Penalização

$$\text{Penalização}(X_j) = R * (\sum_{i=1 \text{ a } N} X_j [i] * P[i]) - C_{\max})^2$$

$$C_{\max} = 50$$

$$R = \max_{i=1 \text{ a } N} (V[i]/P[i]) \text{ tal que } X_j [i] = 0$$

Penalização pode ser linear, quadrática ou logarítmica

O fundamental é desclassificar cromossomas inviáveis ou maus

INTELIGÊNCIA ARTIFICIAL

Teorema do Esquema (John Holland)

Assume-se: O AG usa cromossomos binários, seleção proporcional à qualidade e um único ponto de corte para o cruzamento

Ex: $H=1??01$

Um **Esquema** é uma palavra **H** sobre um alfabeto alargado $\{1,0,?\}$ (ou $*$)

Ordem de um Esquema $o(H)$ é o seu número de posições fixas

Comprimento definidor $\delta(H)$ de um Esquema é a distância entre a 1ª e última posições definidas no esquema

Esquemas correspondentes a instâncias de maior qualidade tendem a aumentar a sua frequência com o tempo (nas gerações seguintes)

ex: esquema no problema das 8 rainhas:
236?????

Observação (cruzamento) pode destruir o Esquema.

Seja o cromossoma $011 | 1000$ com o ponto de corte indicado:

Seja $H1 =?1? | ???0$ e $H2=??.? | 10??$

$O(H1)= 2; O(H2)=2; \delta(H1)=5; \delta(H2)=1$

H1 tem maior probabilidade de ser destruído que H2

INTELIGÊNCIA ARTIFICIAL

Teorema do Esquema

Um Esquema é uma palavra H sobre um alfabeto alargado $\{1,0,?\}$
Ordem de um Esquema $\sigma(H)$ é o seu número de posições fixas
Comprimento definidor $\delta(H)$ de um Esquema é a distância entre
a 1^a e última posições definidas no esquema

Teorema do Esquema: O número de instâncias de H cresce exponencialmente com o tempo (gerações) se o Esquema obedecer a três propriedades:

- Correspondar a instâncias com **qualidade** acima da média na população considerada;
- Baixa **ordem**
- Baixo **comprimento definidor**

- O que é bom para melhorar a qualidade da população

INTELIGÊNCIA ARTIFICIAL

REVISÃO

Algoritmos para a Evolução

Princípio: Seleção Natural: Adaptação → Reprodução

Algoritmos para Evolução Quando?

Espaço de pesquisa complexo ("hard")

Soluções podem ser subótimas

Função de Adaptação depende do problema

Representação dos Indivíduos: Cromossomas, Genes

Algoritmos Genéticos: {0,1}

Escalar e traduzir para binário

Seleção: probabilidade $fa(C_i) / \sum_{i=1}^{aN} fa(C_i)$

Cruzamento: Quantos se cruzam e quais, e qual/quais o(s) ponto(s) de "crossover"

Mutação com baixa probabilidade

Exemplos

Teorema do Esquema

INTELIGÊNCIA ARTIFICIAL

MÉTODOS INFORMADOS

Método "O Melhor Primeiro" (bf):

- Seleciona o estado **mais promissor** de um conjunto deles usando uma Função Heurística para a avaliação.
- Realiza uma pesquisa em **grafo**, mas guarda uma árvore de prova.
- Combina as estratégias "primeiro em profundidade" e "primeiro em largura".
- É um Método informado pois aplica conhecimento **global** do problema.
- Deve comparar-se a diminuição do **custo** do passo **versus** o aumento do **custo** da pesquisa desse passo.
- O poder heurístico é usado para ordenar os estados (nós) a pesquisar.

Escolhas NÃO são irrevogáveis

INTELIGÊNCIA ARTIFICIAL

MÉTODOS INFORMADOS

- Exemplo de função heurística “simplista” na resolução do “puzzle de 8”:

$$f(n) = p(n) + e(n)$$

f-função de custos

1	2	6
8		4
7	3	5

onde $p(n)$ profundidade do nó n e a heurística $e(n)$ número de erros

Notar que se na função $f(n)$ a parcela $e(n)=0$ então temos o algoritmo “primeiro em largura” (ou BB)
(BB =algoritmo de Custo-Uniforme)

- A função heurística usa uma métrica de algum modo ligado à distância à solução

INTELIGÊNCIA ARTIFICIAL

Métodos “o melhor-primeiro” (“Best first”):

- algoritmos “gananciosos” (“greedy”) usam $h^*(n)$ – estimativas do futuro
- algoritmo do custo uniforme (BB) usam $g(n)$ - custos até ao presente
- algoritmo **A*** usa $f^*(n) = g(n) + h^*(n)$ usam uma combinação dos dois.

INTELIGÊNCIA ARTIFICIAL

Algoritmo A*

Escolhe (o sucessor) "o melhor primeiro" fazendo uma pesquisa em Grafo.

Cada **Nó** de pesquisa representa *Estado+Valor da F.Heurística + Apontador para Nó pai + Apontadores para Nós Sucessores.*

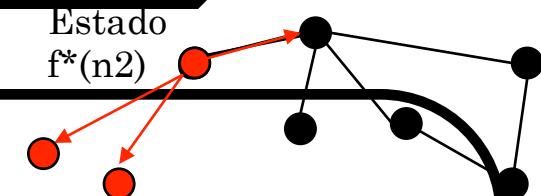
São mantidas duas Listas de Nós:

LABERTA: Nós gerados com respetivas Funções de Avaliação e ainda **não expandidos**;

LFECHADA: Nós já examinados (**expandidos**);

Função Heurística: $f^*(n) = g(n) + h^*(n)$

h^* representa a estimativa do custo do passo desde o nó corrente até à solução



INTELIGÊNCIA ARTIFICIAL

Algoritmo A*

LABERTA <- Estado_Inicial; LFECHADA <- NIL;

ATÉ encontrar solução FAZER

SE LABERTA=NIL ENTÃO RETORNAR (Falso)
SENÃO

Retirar de LABERTA o Nó N mais promissor (melhor f*)
Colocar N na LFECHADA

SE N=solução RETORNAR (N ou Passo)
SENÃO Gerar Sucessores de N

(continua no próximo quadro)

INTELIGÊNCIA ARTIFICIAL

PARA CADA Sucessor de N **FAZER**

Sucessor aponta para N

Computar $g(\text{Sucessor}(N)) = g(N) + \text{custo do arco de } N \text{ a Sucessor}(N)$

SE $\text{Sucessor}(N) \in \text{LABERTA}$ comparar valores de g respetivos

SE $g(\text{Sucessor}(N)) < g(\text{Nó}_\text{antigo})$ **ENTÃO**

Refazer apontador do Nó_antigo para N {novo pai}

Registrar novo custo $g(\text{Nó})$ e novo $f^*(\text{Nó})$

{**SENÃO** nada}

INTELIGÊNCIA ARTIFICIAL

SENÃO {Nó \notin LABERTA}

SE Sucessor(N) \in LFECHADA **ENTÃO**
Comparar $g(\text{Sucessor}(N))$ com $g(\text{Nó}_\text{antigo})$

SE $g(\text{Sucessor}(N)) < g(\text{Nó}_\text{antigo})$ **ENTÃO**
Refazer Apontador do Nó_antigo para N
Registar novo custo $g(\text{Nó})$ e $f^*(\text{Nó})$
Propagar a melhoria para Sucessores de Nó antigo
até que Nó esteja em LABERTA ou não haja Sucessores

{**SENÃO** nada}

SENÃO {Sucessor(N) **não pertence** a qualquer das Listas}

Junte Sucessor(N) a LABERTA
Computar $f^*(\text{Sucessor}(N)) = g(\text{Sucessor}(N)) + h^*(\text{Sucessor}(N))$

FIM

INTELIGÊNCIA ARTIFICIAL

Como f^* também depende de g , em cada nível não se escolhe só "o mais promissor" (melhor estimativa para futuro) mas a melhor combinação da promessa com o melhor passo até então (menor custo até ao momento).

No algoritmo A*, h^* deve sub-estimar h (o custo) para encontrar o caminho de custo ótimo (ser otimista).

Se $h^*=0$, depende só do g (BB); e se $g=\text{constante}$, então seríamos "largura"

ADMISSIBILIDADE da heurística

COROLÁRIO: Decréscimo Progressivo da Admissibilidade

Se h^* raramente sobre-estima h mais do que d , então A* raramente encontrará solução de custo superior a $d+\text{custo óptimo}$ do passo para a solução

INTELIGÊNCIA ARTIFICIAL

1	2	6
8		4
7	3	5

$$f(n) = p(n) + e(n)$$

A parcela $e(N)$ na f. heurística do "Puzzle de 8" é uma **sub**-estimativa pois é um limite **inferior** do número de movimentos necessários para alcançar o objetivo.

$h^*(N)=e(N)$ é mais informado que $h^*(N)=0$ ("p-em_l")

Seleccionar uma boa f.heurística é crucial (diferente do "B&B")

No problema do "puzzle de 8" é preferível $h(N)=D(N)$ onde $D(N)$ é a soma das distâncias a que cada elemento se encontra da sua posição correta.

Outra f. heurística :

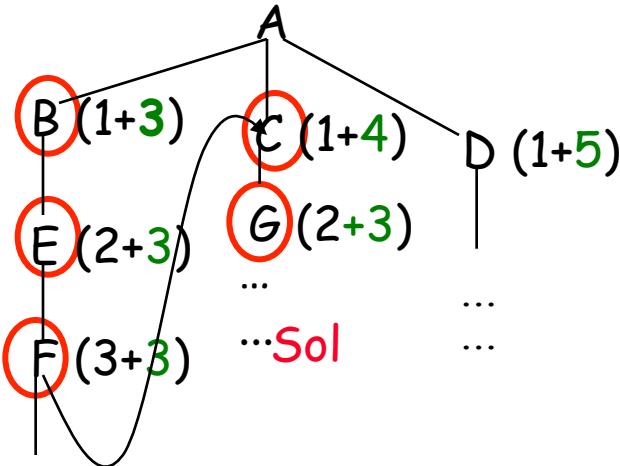
$h(N)=D(N)+3*S(N)$ onde $S(N)$ testa cada número em posição não central.

$S(n)=0$ se n forá do sítio é **seguido** do seu vizinho; $S(n)=1$ na casa central;
 $S(n)=2$ nos outros casos.

$h^*(n)$ é aqui uma função de custo

INTELIGÊNCIA ARTIFICIAL

Exemplo de sub-estimativa de h^*



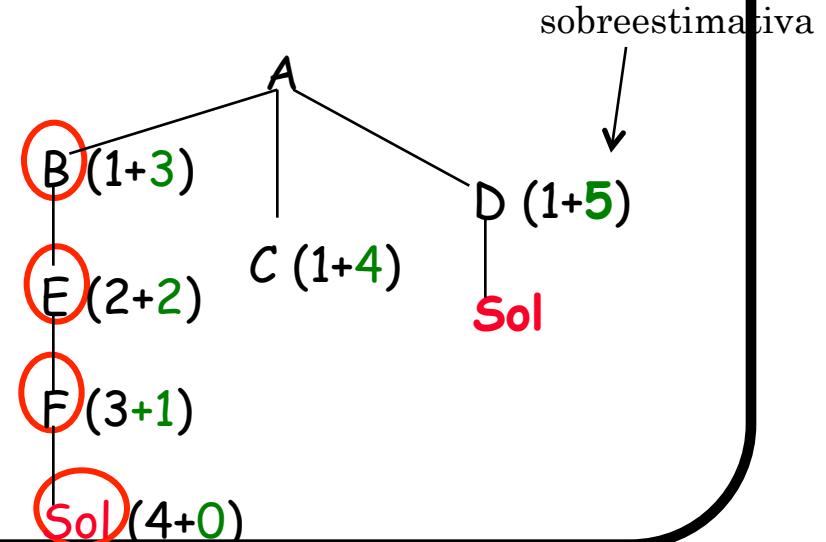
Sub-estimámos $h(B)$.

Perdeu-se tempo mas nada impede de encontrar o melhor passo através de C

Exemplo de sobre-estimativa de h^*

Sobre-estimámos valor de $h(D)$

Não se encontraria o melhor passo
(comp.2) devido a ter sobre-estimado
 $h(D)$



INTELIGÊNCIA ARTIFICIAL

- **A*** não só gasta bastante **tempo** como usa muita **memória** ao guardar as listas de nós.

Por isso não se aplica a problemas de muito grande escala.

- Usa-se **IDA*** **A*** com aprofundamento iterativo para reduzir a memória necessária (pesquisa tipo "p-e-p" de cada vez; usa-se o valor da função de custo (f^*) em vez da profundidade (p) para parar e recomeçar de novo a expansão). Outros: **RBFS**

- Outra condição da heurística para aplicar **A*** :
 A^* é ótimo quando a heurística $h^*(n)$ é **consistente**:

$h^*(n) \leq c(n, a, n') + h(n')$ sendo n' um sucessor de n

(se fosse $>$, não estavamos a usar um passo menos custoso partindo de n e passando por n' . Estávamos a sobre-estimar h^*)

INTELIGÊNCIA ARTIFICIAL

Comparação do número de nós gerados (em média para 1200 casos aleatórios) para encontrar o caminho para a solução no problema do "Puzzle-15" AIMA, S.Russel, P.Norvig

Nós gerados			
p	AP (ID)	A* ($h^*=e(N)$)	A* ($h^*(N)=D(N)$)
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	3644035	227	73
....

INTELIGÊNCIA ARTIFICIAL

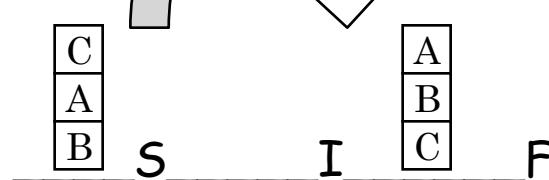
Exemplo em **Prolog** de estratégias básicas de resolução de problemas

Conceitos:

- Representação dos **Estados** do Problema
- Ações** legais de mudança de estado (operadores)
- Estados **Inicial** e **Objetivo**

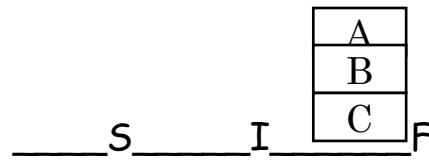
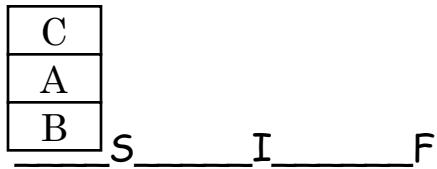
Cálculo de estados sucessores: $s(X, Y, C)$ Y é o sucessor de X a um custo C .

Exemplo de problema para um robô manipulador no “mundo dos blocos”:



Espaço na mesa para 3 pilhas de objetos

INTELIGÊNCIA ARTIFICIAL



Representação dos estados: por listas de 3 elementos (posições).
Cada elemento é uma lista com os blocos (uma pilha)

Estado inicial: [[C,A,B], [], []]

Estado Objetivo: [[A,B,C], [], []] ou [[], [A,B,C], []] ou [[], [], [A,B,C]]

Regra para cálculo de sucessor:

Estado2 é sucessor de Estado1 se existem pelo menos 2 pilhas (uma pode até ser vazia) no Estado1 tal que o bloco do topo da pilha1 é movido para o topo da pilha2, formando assim o Estado2.

INTELIGÊNCIA ARTIFICIAL

$s(\text{PilhasA}, [\text{P1}, [\text{Topo1} | \text{P2}] | \text{Outras_P}]) :- \text{del}([\text{Topo1} | \text{P1}], \text{PilhasA},$
Estado 1 *Estado 2* $\text{PilhasB}), /*\text{retira } 1^{\text{a}} \text{ pilha}*/$
Lista de 3 Pilhas $\text{del}(\text{P2}, \text{PilhasB}, \text{Outras_P}). /*\text{retira } 2^{\text{a}} \text{ pilha} */$

$\text{del}(X, [X | Y], Y).$

Incompleto! Faltam predicados s/2

$\text{del}(X, [Y | L], [Y | L1]) :- \text{del}(X, L, L1).$

OU $\text{objetivo}([_, _, [A, B, C]]).$

$\text{objetivo}(\text{Estado}) :- \text{member}(['A', 'B', 'C'], \text{Estado}).$

$\text{resolver}([[], [C, A, B], [], []], \text{Solução}).$

Estado (Nó na árvore de pesquisa)

Passo entre inicio e qualquer nó objetivo

--> Pesquisa primeiro em profundidade.

Partindo do estado N encontra-se o passo Sol:

a) Se $N = \text{objetivo}$ --> $Sol = [N | Passo]$

b) Se N_1 é sucessor de N e Sol_1 é o passo de N_1 a objetivo
--> $Sol = [N | Sol_1]$

(aproveita-se a estratégia embutida no Prolog):

$\text{resolver}(N, [N | Sol]) :- \text{objetivo}(N).$

$\text{resolver}(N, [N | Sol_1]) :- s(N, N_1), \text{resolver}(N_1, Sol_1).$

INTELIGÊNCIA ARTIFICIAL

```
s(PilhasA, [P1, [Topo1| P2]| Outras_P]):- det([Topo1], PilhasA,  
      Estado 1           Estado 2          PilhasB), /*retira 1ª pilha*/  
      del (P2, PilhasB, Outras_P). /*retira 2ª pilha */
```

```
del (X, [X|Y], Y).  
del (X, [Y|L], [Y|L1]):-del (X, L, L1).
```

Incompleto! Faltam predicados s/2

```
objetivo (Estado):- member (['A', 'B', 'C'], Estado).  
resolver ([[C', 'A', 'B'], [ ], [ ]], Solução).
```

Passo entre Início e qualquer nó objetivo

--> **Pesquisa primeiro em profundidade:**

Partindo do estado N encontra-se o passo Sol:

a) Se $N = Sol$ --> $Sol = [N]$

b) Se N_1 é sucessor de N e Sol_1 é o passo de N_1 a objetivo
--> $Sol \leftarrow [N_1 | Sol_1]$

(aproveita-se a estratégia embutida no Prolog):

resolver(N, [NSol]):- objetivo(N).

resolver(N, [NSol1]):- s(N,N1), resolver(N1,[NSol1]).

INTELIGÊNCIA ARTIFICIAL

Variações:

- a) Evitando ciclos quando \exists no grafo e
- b) limitando a profundidade da pesquisa

resolver(N, S):-p_prof([], N, S) *Caminho até ao momento, Nó a analisar*

a) **p_prof (Passo, N,[N|Passo]):-objetivo (N).**

p_prof(Passo, N, S) :-
s(N,N1), not (membro (N1,Passo)),
p_prof ([N|Passo], N1, S).

- OU -

b) **p_prof2 (N, [N],_):-objetivo (N).**

p_prof2 (N, [N|Sol], Maxprof): -
Maxprof >0, s(N, N1),
Max1 is Maxprof-1,
p_prof2 (N1, Sol, Max1).

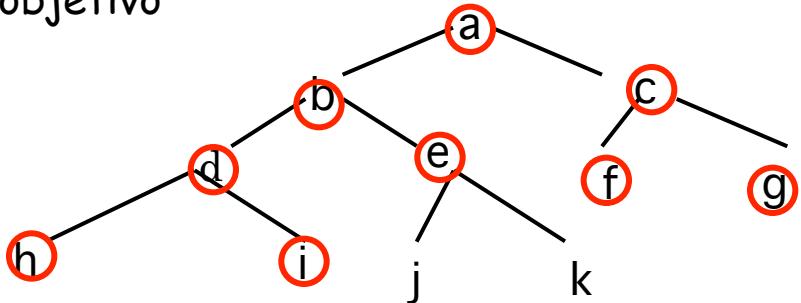
Para a **combinação das variantes** necessitamos de um predicado
p_prof com 4 argumentos

INTELIGÊNCIA ARTIFICIAL

Algoritmo Primeiro em Largura (breadth-first)

SE à cabeça do 1º passo aparece o objetivo
ENTÃO esse passo é a solução
SENÃO encontra o passo **mais curto**; **remover** esse 1º passo do conjº de passos candidatos e **gerar** todos os passos resultantes da extensão do seu nó cabeça com novos sucessores imediatos e **junte** este novo conjº de passos na cauda do conjunto anterior.

Execute de novo a estratégia `primeiro_em_largura`.



Pode ser implementado em qualquer Linguagem de programação

Sucessão da Lista dos Passos:
[[a]] \rightarrow [[b,a],[c,a]] \rightarrow [[c,a],[d,b,a],[e,b,a]] \rightarrow
[[d,b,a],[e,b,a],[f,c,a],[g,c,a]] \rightarrow ...

INTELIGÊNCIA ARTIFICIAL

Primeiro-em-Largura em Prolog

```
resolver(Inicio,Sol):- p_largura([ [Inicio] ],Sol).  
  
p_largura([ [N|RPasso] | _ ], [N|RPasso]):- objetivo(N).  
p_largura([ [N|RPasso] | Passos ],Sol):-  
    (findall( [M,N|RPasso],  
              (s(N,M), not(member(M,[N|RPasso]))))  
     NovosPassos),  
    conc(Passos,NovosPassos, NLP), !, %Junta à lista as  
    p_largura(NLP,Sol)) % expansões do Nó.  
    : %alternativa quando falha sucessor de N  
    p_largura(Passos,Sol).  
  
% Falta definir predicados conc/3 e s/2
```

INTELIGÊNCIA ARTIFICIAL

Algoritmo "o Melhor Primeiro" aplicado ao puzzle de 8:

Predicados específicos do problema:

- s (Nodo, Nodo1, Custo)
- objetivo (Nodo) ~~Custo~~
- h^* (Nodo, H) H é o custo heurístico, estimado, do passo mais barato entre Nó e o objetivo

Nó no espaço de estados é a lista de coord. dos: espaço vago, da peça 1, .. , da peça 8 → uma Lista de 9 posições

- $\text{objetivo} ([2:2, 1:3, 2:3, 3:3, 3:2, 3:1, 2:1, 1:1, 1:2]).$
branco, 1, ... , 8

Posição Final		
3	1	2
2	8	4
1	7	6
1	2	3

INTELIGÊNCIA ARTIFICIAL

- relação auxiliar:

$d(Q1, Q2, D)$. A distância entre 2 posições no tabuleiro para integrar a f^*

D é a distância "Manhattan" entre $Q1$ e $Q2$ medida na horizontal + distância na vertical

inicio([2:2, 1:3, 3:2, 2:3, 3:3, 3:1, 2:1, 1:1, 1:2]).

Custos de cada arco no espaço de estados = 1

/*Cálculo de Sucessores*/

$s([Livre|L], [T|L1], 1) :- troca(Livre, T, L, L1).$ /* troca Livre
/* L' com T em L' dá $L1$ */

$troca(Livre, T, [T|L], [Livre|L]) :- d(Livre, T, 1).$

/* T é o Livre do sucessor se estiver à dist. 1 de Livre */

$troca(Livre, T, [T1|L], [T1|L1]) :- troca(Livre, T, L, L1).$

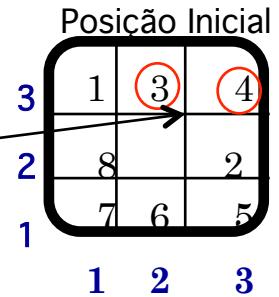
/*distância Manhattan*/

$d(X:Y, X1:Y1, D) :- dif(X, X1, Dx), dif(Y, Y1, Dy),$
D is $Dx+Dy$.

$dif(A, B, D) :- D is A-B, D >= 0, !;$ D is $B-A.$

Posição Inicial		
3	1	3
2	8	2
1	7	6
2	5	3

INTELIGÊNCIA ARTIFICIAL



$S(n)=0$ se n forá do sítio é seguido do seu vizinho na posição final;
 $S(n)=1$ na casa central.

$S(n)=2$ nos outros casos.

$$h^* = d(n) + 3s(n)$$

$$h^* (\text{inicio}) = 4 + 3 * 4$$

$D=4$ e $s=4$ ("2" e "4" fora de sequência)

/*função heuristica h^* é a soma das distâncias de cada bloco à sua posição correcta+3 vezes o "score" relativo à sequência em que aparecem*/

$h([\text{Livre}|L], H)$: -

objetivo ([Livre1|Ob]), distot(L, Ob, D),
 $\text{seq}(L, S), H \text{ is } D + 3 * S.$

Estado a calcular a sua H

$\text{distot} ([], [], 0).$

$\text{distot} ([T|L], [T1|L1], D)$: -

$d(T, T1, D1), \text{distot}(L, L1, D2), D \text{ is } D1 + D2.$

$\text{seq} ([\text{Prim}|L], S)$: - $\text{seq} ([\text{Prim}|L], \text{Prim}, S).$

$\text{seq} ([T1, T2|L], \text{Prim}, S)$: - $\text{score}(T1, T2, S1), \text{seq} ([T2|L], \text{Prim}, S2), S \text{ is } S1 + S2.$

$\text{seq} ([\text{Ult}], \text{Prim}, S)$: - $\text{score}(\text{Ult}, \text{Prim}, S).$

INTELIGÊNCIA ARTIFICIAL

```
score(2:2,-,1):-!. /*bloco no centro vale 1*/                      resolver(Inicio, Solucao)

score(1:3,2:3,0):-!.      score(2:3,3:3,0):-!.      score(3:3,3:2,0):-!.
score(3:2,3:1,0):-!.      score(3:1,2:1,0):-!.      score(2:1,1:1,0):-!.
score(1:1,1:2,0):-!.      score(1:2,1:3,0):-!. /*sucessor correcto vale 0*/
score(_,_,2)      /*Blocos fora da sequência*/

objectivo ([2:2,1:3,2:3,3:3,3:2,3:1,2:1,1:1,1:2]).
inicio1   ([2:2,1:3,3:2,2:3,3:3,3:1,2:1,1:1,1:2]).
.....
/*display do passo solução como lista de posições de tabuleiro*/

mostre ([J]).
mostre ([P | L]):-mostre(L),nl,write(' _ _'), mostrepos(P).
| *display de um tabuleiro*
mostrepos([B0,B1,B2,B3,B4,B5,B6,B7,B8]):-  
    member(Y,[3,2,1]), member (X,[1,2,3]),
    member (Bloco-X:Y,[` '-B0,1-B1,2-B2,3-B3,4-B4,5-B5,6-B6,7-B7,8-B8]),
    write (Bloco)
```

haveria que aplicar estes predicados específicos ao algoritmo genérico de A*

INTELIGÊNCIA ARTIFICIAL

REVISÃO

Métodos Informados:

"Melhor-Primeiro"

- Função heurística $f^* = g + h^*$

gananciosos

custo uniforme

- Algoritmo A^*

LF e LA

Corolário do decremento progressivo da Admissibilidade

- Programas Prolog para pesquisa:

- Primeiro-em-Profundidade

- Variantes

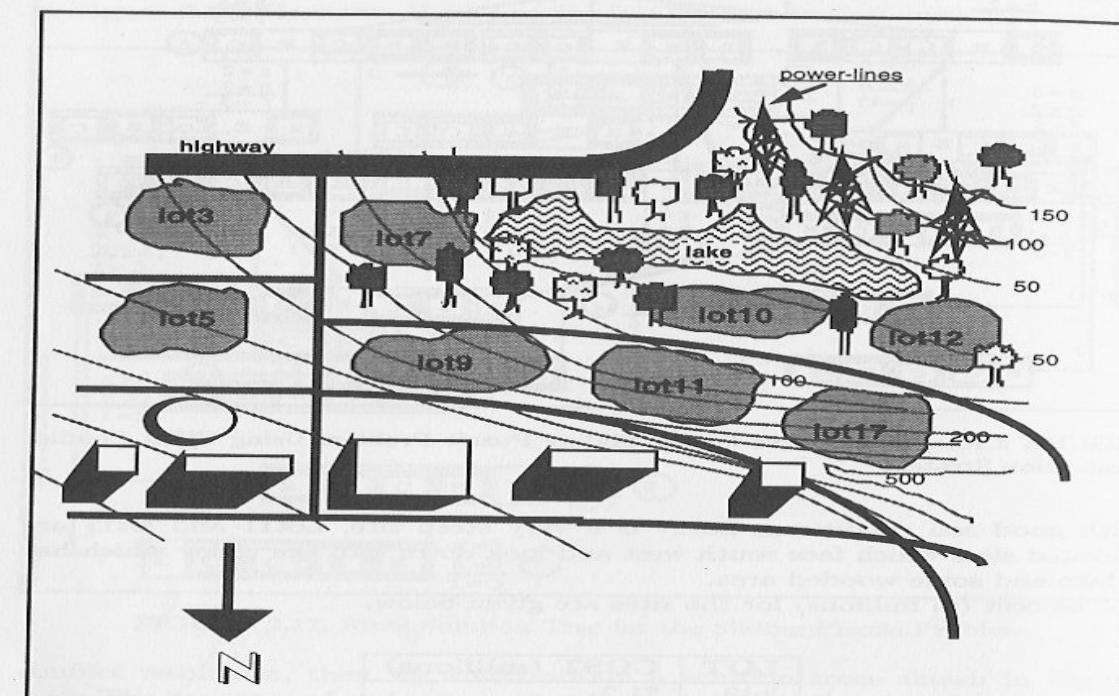
- Primeiro-em-Largura

- Definição em Prolog do problema "Puzzle de 8" para a aplicação do "Melhor-primeiro"

INTELIGÊNCIA ARTIFICIAL

- **Aplicação do Algoritmo A***
- (Exemplo prático de aplicação combinada de métodos)
- **Dados** 8 Lotes de terreno com 5 possíveis **Usos**:
- *Recreio, Apartamentos, Habitação Social, Cemitério, Lixeira*
- **Critérios (Restrições) à atribuição:**
 - *C1 Recreio* próximo do lago
 - *C2* Evitar terrenos inclinados para *Construção, Cemitério e Lixeira*
 - *C3 Solos instáveis maus para Construção*
 - *C4 Autoestrada longe de Habitações e Recreio*
- **OBJETIVO:** Atribuir Lotes a Utilizações minimizando custos e respeitando critérios

INTELIGÊNCIA ARTIFICIAL



Só 3 perto do lago
e só 2 longe da A.Estrada

R. Sriram
“Intelligent Systems for Engineering”,
Springer

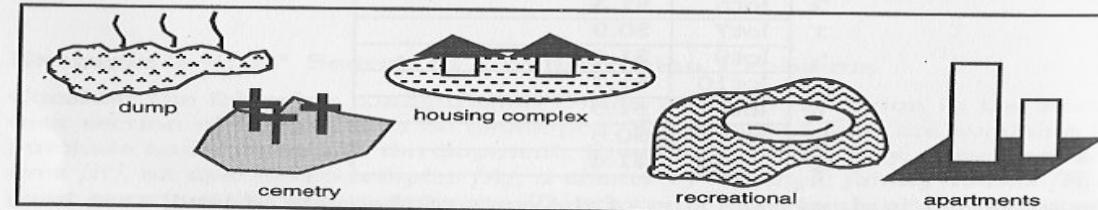


FIGURE 2.19. The Site for Allocating Landuses

INTELIGÊNCIA ARTIFICIAL

	I3	I5	I7	I9	I10	I11	I12	I17
solo	b	b	b	b	m	b	m	m
incl.	b	b	b	b	r	m	r	mm
estr.	m	b	m	b	b	b	b	b
Lago	m	m	b	m	b	m	b	m
custo	1,2	1,3	0,9	1,6	1,7	1	1,4	0,8

Representação: Sol_i Lista das atribuições de Usos a Lotes no estado i

LU_i Lista dos Usos ainda não considerados

LI_i Lista dos Lotes ainda não atribuídos

Estado inicial: $Sol_0 = [], LU_0 = [r,a,h,c,l], LI_0 = [I3, I5, I7, I9, I10, I11, I12, I17]$

Objetivo: $Sol_i = [r-Lr, a-La, h-Lh, c-Lc, l-Ll] \quad LU_i = [] \quad LI_i = [L1, L2, L3]$

Função heurística: $f^*(N) = g(N) + h^*(N)$

$h^*(N)$ soma dos custos dos **p mais baratos lotes** ainda não atribuídos;

p é o numero de Usos ainda não contemplados

$g(N)$ custo da solução parcial até ao momento

	I3	I5	I7	I9	I10	I11	I12	I17
solo	b	b	b	b	m	b	m	m
Inc	b	b	b	b	r	m	r	mm
estr.	m	b	m	b	b	b	b	b
Iago	n	n	b	n	b	n	b	n
cust 1,2	1,3	0,9	1,6		1,7	1	1,4	0,8

Recreio : o mais restrito (próximo do Lago, longe da A.E.)

Combinação do A* com Satisfação de Restrições:

- O fator de ramificação no início seria de $8 * 5$

Aplicando as restrições e o princípio do Mais restrito primeiro, (R), reduz-se a Árvore de Pesquisa.

Estamos aqui também a usar
Heurística

Existe IDA*- aumento progressivo da profundidade e aplicação de f*

ARTIFICIAL

Mais baratos no início : (17) 0,8 -(7) 0,9 -(11) 1 -(3) 1,2- (5) 1,3

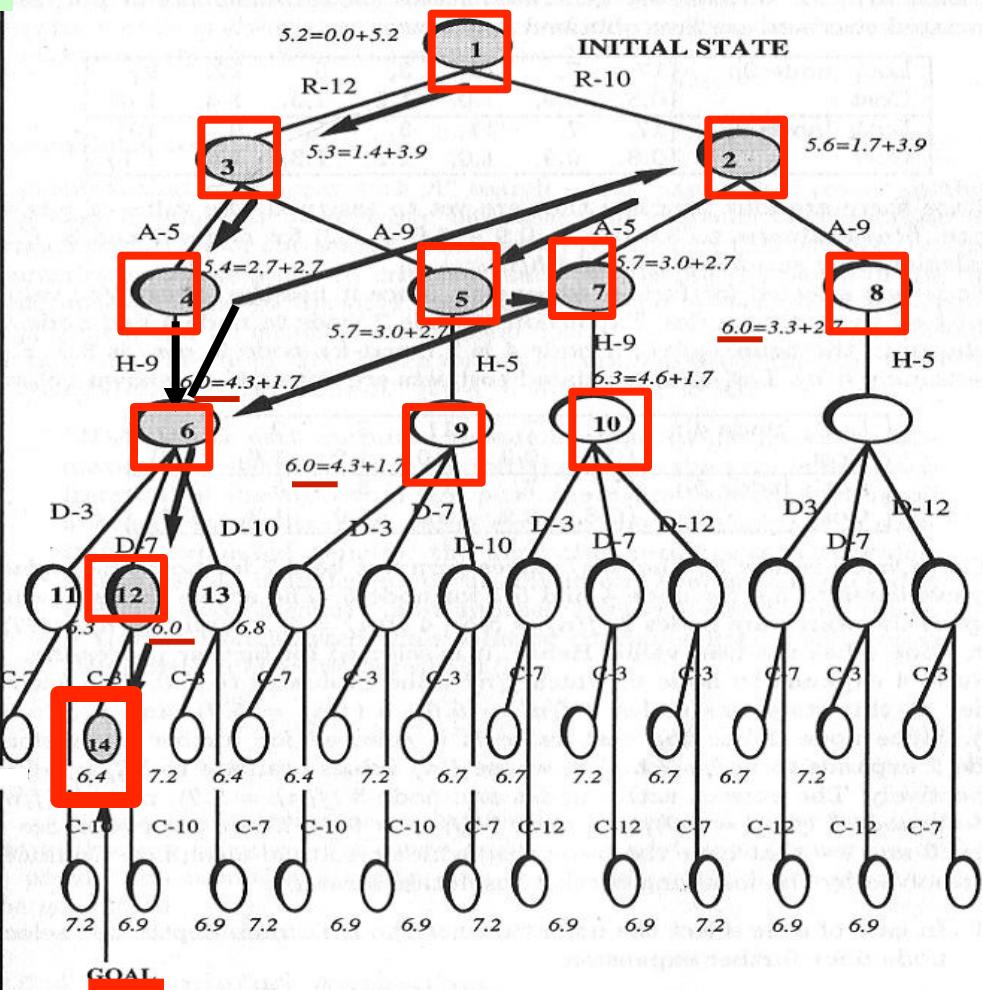


FIGURE 2.20. Solution Trace Using A* Search

INTELIGÊNCIA ARTIFICIAL

MÉTODO DA DECOMPOSIÇÃO

- Decomposição quando um problema complexo pode ser decomposto em vários problemas mais simples tratáveis em paralelo.

• Procedimento separa
DADOS <-- BD inicial

{Di},<-- Decomposição de DADOS

A TE Todos os Di satisfazerem a condição final FAZER
Seleccionar D entre {Di} que não satisfaz condição final
Tirar D de {Di}

Seleccionar Regra R aplicável a D

DD <-- resultado da aplicação de R a D

{di} <-- decomposição de DD

juntar {di} a {Di}

- Estruturas úteis para representar a atividade de S. de Produções são grafos E/OU
- ex:BD inicial CBZ
Operadores:
Objetivo BD só com Ms

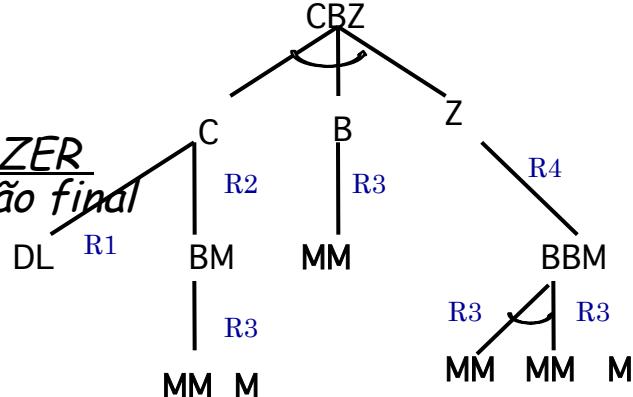
Operadores:

R1:C-->(D,L)

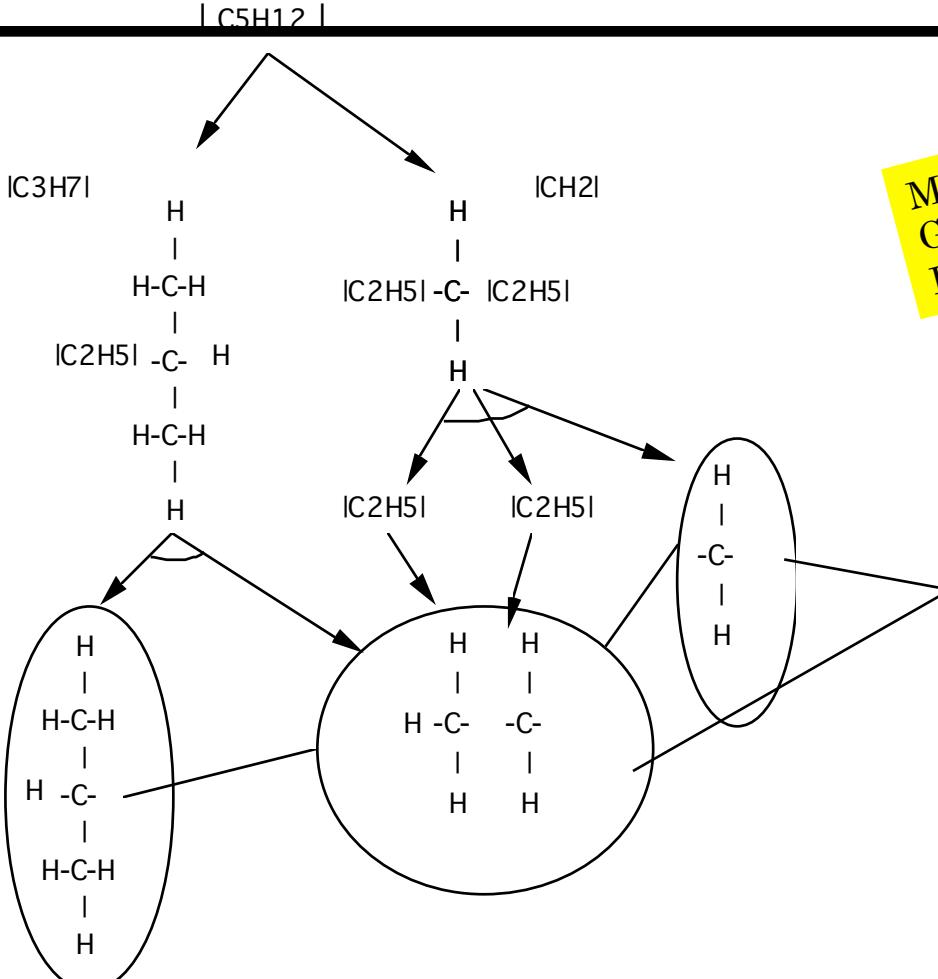
R2:C-->(B,M)

R3:B-->(M,M)

R4:Z-->(B B M)



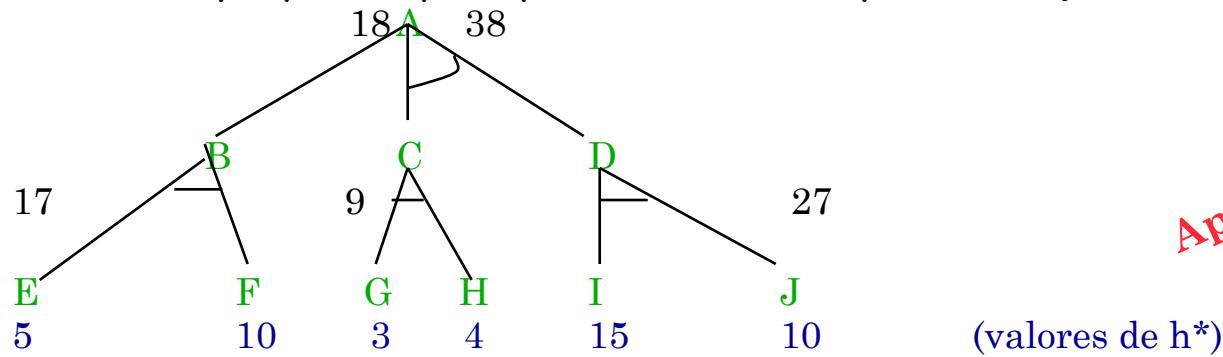
INTELIGÊNCIA ARTIFICIAL



MOLGEN:
Geração de Estruturas Moleculares
Partindo de fórmulas atómicas

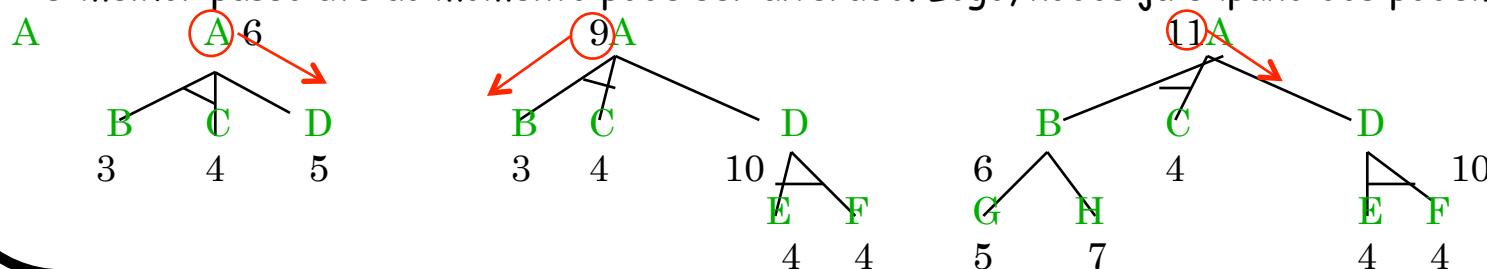
INTELIGÊNCIA ARTIFICIAL

A não é apropriado para problemas decompostos. Seja o exemplo em que cada arco custa 1:*



Procedimento genérico:

- Começar no nó inicial, expandi-lo, escolher o melhor passo. Guardar nós ainda não expandidos. Considerar um destes (N) e expandi-lo. Computar h^* dos sucessores. Calcular $f^*(N)$ tendo em conta os h^* dos sucessores.
- **Propagar o novo f^* para cima** (para trás).
- O melhor passo até ao momento pode ser alterado. Logo, nodos já expandidos podem ser revisitados.



INTELIGÊNCIA ARTIFICIAL

SATISFAÇÃO DE RESTRIÇÕES: Combina-se com os métodos anteriores.

Usa-se quando existem restrições conhecidas a priori e presentes durante a pesquisa.

A descrição do **Estado completa-se com a das Restrições** e efectuam-se **pesquisas** nos dois espaços.

PROCEDIMENTO GENÉRICO

REPETIR ATÉ Solução completa encontrada

 Seleccionar **nó não expandido** no Grafo de Pesquisa

 Aplicar-lhe as **Regras de Constrangimento** e Gerar novas Restrições

 SE conjunto das Restrições incoerente ENTÃO RETORNAR falso

 SE conjunto das Restrições descreve a Solução RETORNAR verdadeiro

 SENÃO Aplicar as **Regras de expansão** ao Espaço de Estados

 E Gerar novas Soluções parciais consistentes com as Restrições

 Inserir esses **novos Estados** no Grafo de pesquisa

EXEMPLO de aplicação do método da Satisfação de Restrições:

- a) Numéricas;
- b) Simbólicas;

INTELIGÊNCIA ARTIFICIAL

Problemas por SATISFAÇÃO DE RESTRIÇÕES: Formalmente temos um conjº de Variáveis X_i , um conjº de Restrições R_j . Cada X_i tem um domínio não vazio D_i .

Um **Estado** do problema é definido por uma **atribuição** de valores a pelo menos algumas das variáveis $\{V_1=v_1, \dots, V_n=v_n\}$.

A atribuição é **consistente** se não viola qualquer restrição.

Uma **solução** tem uma atribuição consistente para todas as variáveis.

Variáveis discretas podem ter domínios finitos ou infinitos.

Ex: *Problem das 8 Rainhas* tem 8 variáveis no domínio $\{1, 2, 3, 4, 5, 6, 7, 8\}$ (domínio finito)

Se o número máximo de elementos do domínio das n variáveis for d , o número máximo de atribuições é $O(d^n)$

Variáveis podem ter restrições **unárias** (ex: $V_1 > 0$), **binárias** (envolvendo pares; ex: $V_1 < V_2 + d$) ou de **ordem superior** ($V_1 = V_2 \neq V_3$).

Ex: cripto aritmética usa restrições de ordem superior.

Ex: $TWO + TWO = FOUR$

Como todas as letras deverão ter valores diferentes: $AllDiff(F, U, T, W, R, O)$ ou então exprimir todas as restrições binárias: $F \neq U; F \neq T, \dots$

INTELIGÊNCIA ARTIFICIAL

a) Problema da medição do perfil da montanha

Dados: medidas iniciais de altitudes com instrumentos de precisão diferente

Objetivo: determinar o Perfil da montanha aplicando propagação de restrições

Aplicar a fórmula da Relaxação (também usada em análise de imagens)

Fórmula da Relaxação: $A_i^n = C_i * B_i^n + [(1 - C_i) * (A_{i+1}^{n-1} + A_{i-1}^{n-1})/2]$

A: Array das Altitudes a calcular

B: Array das medidas efectuadas

C: Array das confianças nas medidas

n: Número de Iterações

Algoritmo simplificado (ex. de Satisfação de Restrições Numéricas)

Considerar Array corrente (iteracção n)

FAZER ATÉ valores alterarem "pouco"

Para cada elemento A_i^n do Array A

Calcular novo valor pela fórmula da Relaxação

Inserir em novo array A^{n+1}

Novo Array torna-se Array corrente

INTELIGÊNCIA ARTIFICIAL

In "Artificial Intelligence"
Patrick Winston

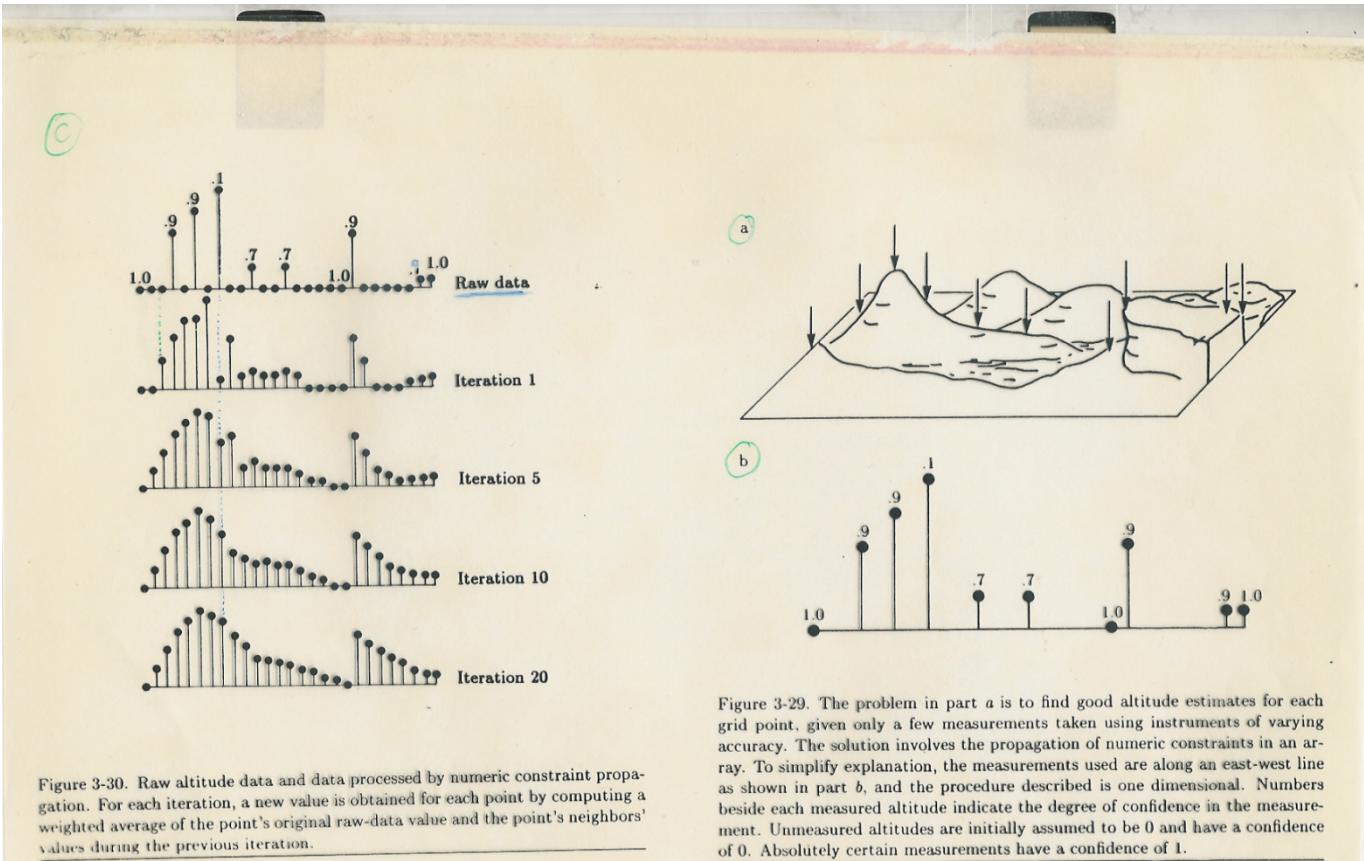


Figure 3-30. Raw altitude data and data processed by numeric constraint propagation. For each iteration, a new value is obtained for each point by computing a weighted average of the point's original raw-data value and the point's neighbors' values during the previous iteration.

INTELIGÊNCIA ARTIFICIAL

Exemplo de Propagação de Restrições simbólicas - Ex. da Criptografia:
(código "send more money")

$$\begin{array}{r} t_3 \ t_2 \ t_1 \ t_0 \\ s \ e \ n \ d \\ m \ o \ r \ e \\ \hline m \ o \ n \ e \ y \end{array}$$

$$\begin{array}{r} SEND \\ +MORE \\ \hline MONEY \end{array}$$

Domínio das variáveis, [0..9]

Estados: a corrente atribuição de valores numéricos às variáveis;

Operadores: atribuição de valores que permitam verificar as **regras da soma**;

Restrições: Valores numéricos anteriormente atribuídos impõe restrições a valores ainda a atribuir (devem ser **diferentes** para letras diferentes) (*alldiff*)

predicados a utilizar para a descrição do problema:

valor(X,VX)
mais(T,P1,P2,S,NT)

INTELIGÊNCIA ARTIFICIAL

t3 t2 t1 t0
s e n d
m o r e
m o n e y

Exemplo de Propagação de Restrições simbólicas em prolog simples
- Ex. da Criptoaritmética (*usando restrições binárias*):

resolva :- valor(t0, T0), valor(m,M), valor(d,D), D ≠ M,
valor(e,E), E ≠ M, E ≠ D,
mais(T0,D,E,Y,T1), Y ≠ M, Y ≠ E, Y ≠ D,

valor(n,N), N ≠ M, N ≠ D, N ≠ Y, N ≠ E,
valor(r,R), R ≠ M, R ≠ Y, R ≠ D, R ≠ E, R ≠ N,
mais(T1,N,R,E,T2),

valor(o,O), O ≠ M, O ≠ R, O ≠ E, O ≠ D, O ≠ Y, O ≠ N,
mais(T2,E,O,N,T3),
valor(s,S), S ≠ M, S ≠ O, S ≠ N, S ≠ R, S ≠ Y, S ≠ E, S ≠ D,
mais(T3,S,M,O,M),

escreva([(s,S), (e,E), (n,N), (d,D), (m,M), (o,O), (r,R), (y,Y)]).

Cláusula “resolva/0”

INTELIGÊNCIA ARTIFICIAL

```
valor(t0,0).      Valor(m,1).
```

```
valor(X,VX) :- membro(VX, [0,1,2,3,4,5,6,7,8,9]). % usa backtracking
```

```
membro(X,[X|_]).
```

```
membro(X,[_|Y]):- membro(X,Y).
```

```
mais(T,A,B,S,NT):- S is (A+B+T) mod 10, NT is (A+B+T)//10, !.
```

```
escreva([]).
```

```
escreva([(X,Y)|L]):- write(X), write(Y), nl, escreva(L).
```

$$\begin{array}{r}
 t2 \ t1 \ t0 \\
 \quad \quad \quad t \ w \ o \\
 + \quad \quad \quad t \ w \ o \\
 \hline
 f \ o \ u \ r
 \end{array}$$

INTELIGÊNCIA ARTIFICIAL

Exemplo de Propagação de Restrições simbólicas em prolog simples
 - Ex. da Criptoaritmética (*usando restrições binárias*):

resolva :- valor(t0, T0), valor(f,F), valor(o,O), O ≠ F,
 mais(T0,O,O,R,T1), R ≠ F, R ≠ O,

valor(w,W), W ≠ F, W ≠ O, W ≠ R,
 mais(T1,W,W,U,T2), U ≠ F, U ≠ O, U ≠ R, U ≠ W,

valor(t,T), T ≠ F, T ≠ O, T ≠ R, T ≠ W, T ≠ W,
 mais(T2,T,T,O,F),

escreva([(f,F), (o,O), (u,U), (r,R), (t,T), (w,W), (r,R)]).

INTELIGÊNCIA ARTIFICIAL

REVISÃO

Métodos de pesquisa de Soluções:

"O Melhor primeiro":

Admissibilidade da Heurística (optimista, coerente)

Ex. de A combinado com restrições
princípio de "o mais restrito primeiro"*

Método da decomposição

AO*

Métodos de Satisfação de Restrições

Ex. de restrições Numéricas

Ex. de restrições Simbólicas

INTELIGÊNCIA ARTIFICIAL

Pesquisa em “Jogos” ou
Estratégias de Pesquisa considerando adversários

- **Jogos**: domínio privilegiado para aplicação e estudo destas técnicas de pesquisa porque:
 - a **função de avaliação** é sempre possível de computar
 - as **regras** a aplicar para gerar novos estados são (normalmente) em número não exagerado e bem definidas
 - o **objetivo** é inequívoco e bem caracterizado

INTELIGÊNCIA ARTIFICIAL

- Problemas que exigem **planeamento** no sentido de antecipação ("look ahead")
 - Jogos de soma-nula, informação perfeita e determinísticos
- Exemplo de função de avaliação (por ex. no jogo de Xadrez):
 - Considerar uma soma pesada de vários fatores como:
 - Vantagem do número de peças
 - Posição da rainha
 - Controlo do centro do Tabuleiro

INTELIGÊNCIA ARTIFICIAL

Minimax é um procedimento tipo expansão “primeiro em profundidade”. A profundidade máxima é limitada a priori.

Procedimento básico do **Algoritmo MiniMax**:

Determinar SE profundidade limite atingida
 OU Nível é Minimizador
 OU Nível é Maximizador

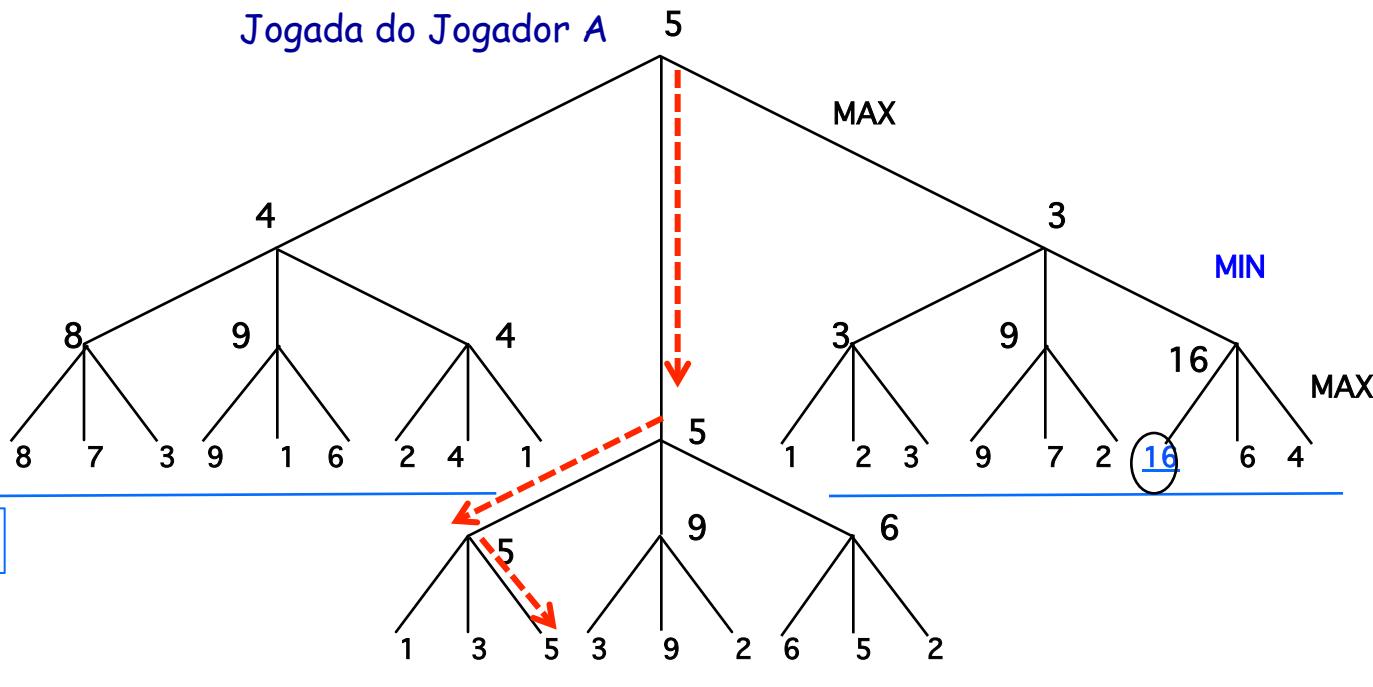
ENTÃO

 SE profundidade limite
 Calcular valor do estado corrente
 Retornar resultado

 SE nível Minimizador
 Aplicar MiniMax aos Sucessores e Retornar Mínimo
 SE nível Maximizador
 Aplicar MiniMax aos Sucessores e Retornar Máximo

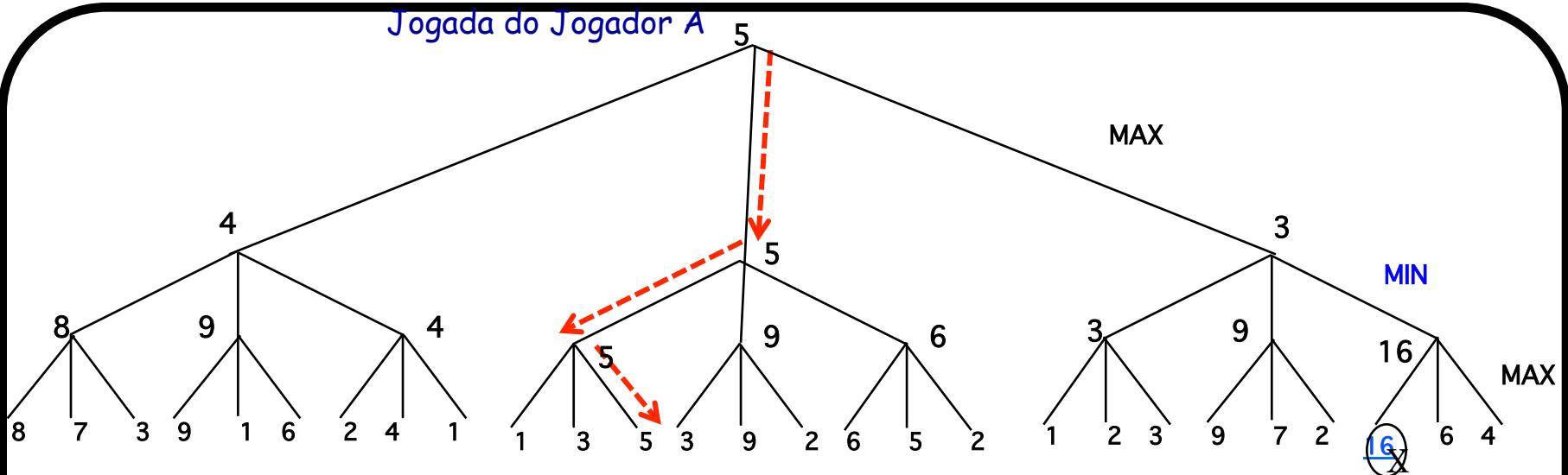
Recursividade

INTELIGÊNCIA ARTIFICIAL



- Processo de pesquisa com MINI-MAX
- exemplo de árvore de pesquisa de profundidade 3 e factor de ramificação 3
- os valores da função de mérito são relativas ao jogador "A"

INTELIGÊNCIA ARTIFICIAL



Nível 3

- Processo de pesquisa com MINI-MAX
- exemplo de árvore de pesquisa de profundidade 3 e factor de ramificação 3
- os valores da função de mérito são relativas ao jogador "A"

INTELIGÊNCIA ARTIFICIAL

Problemas com o Minimax ...

INTELIGÊNCIA ARTIFICIAL

Seja o Jogo do Xadrez:

fator médio de ramificação ~35

tempo entre 2 jogadas: 2,5 min

computador: por ex. analiza 10 000 estados/segundo

Estados analizáveis: $150 \text{ s} * 10^4 = 1\ 500\ 000$

Nº de estados a profundidade p : $r^p = 35^p$

$$35^4 = 1\ 500\ 625$$

Bom jogador humano consegue prevêr de 6 a 8 jogadas à frente

Conclusão: Minimax é demasiado custoso em tempo

INTELIGÊNCIA ARTIFICIAL

Características do DEEPBLUE:

interesse da IBM no poder do Hardware ...

Xadrez no DEEPBLUE usa alfa-beta em **30 potentes** processadores da IBM.

Pesquisa em média 30.000 milhões de posições (nós) em cada movimento +

Função de avaliação representando mais de 8000 características +

Base de dados com 700.000 jogos de xadrez +

Base de dados de finais de Jogos

Minimax continuaria a ser demasiado custoso em tempo



INTELIGÊNCIA ARTIFICIAL

Cortes ALFA-BETA ($\alpha \beta$ pruning)

- Permite, em relação ao procedimento Minimax, diminuir o número de nós visitados e de funções calculadas nos nós avaliados
- É um **método** que usa "primeiro-em-profundidade" mas que o pode abandonar para um "*Branch and Bound*"
- Inclui-se um **Límite** para cada jogador: Límite **inferior** para valor a minimizar (**β valor mais baixo que jogador min já assegurou no passo**) e um Límite **superior** para o valor a Maximizar (**α valor mais alto do jogador Max no respetivo passo**)
- A pesquisa dos sucessores de um nó termina quando deixar de se verificar $\alpha < \beta$

Ganho de eficiência retornando os **mesmos** valores que o MINIMAX

INTELIGÊNCIA ARTIFICIAL

Procedimento básico do Minimax com cortes $\alpha\beta$

Determinar se o nível é : topo OU profundidade limite OU maximizador OU minimizador

SE nível=topo ENTÃO $\alpha=-\text{MAXINT}$ e $\beta=\text{MAXINT}$

SE nível=profundidade máxima ENTÃO computar função de avaliação dos nós

RETORNA resultado V

SE nível=maximizador ENTÃO

ENQUANTO (todos os sucessores não forem examinados com Minimax $\alpha\beta$ E $\alpha < \beta$) FAZER
 α é o maior dos valores seguintes:

% Atualiza α

valor herdado do nível anterior e resultado v de Minimax $\alpha\beta$ aplicado aos sucessores

Aplique ao próximo sucessor já os novos valores de α e β

SE $\alpha < \beta$ RETORNE v= α SENÃO RETORNE v= β

% Corte β

RECUSIVIDADE

SE nível=minimizador ENTÃO

ENQUANTO (todos os sucessores não forem examinados E $\alpha < \beta$) FAZER

β é o menor dos valores seguintes:

% Atualiza β

valor herdado do nível anterior e resultado v de Minimax $\alpha\beta$ dos sucessores

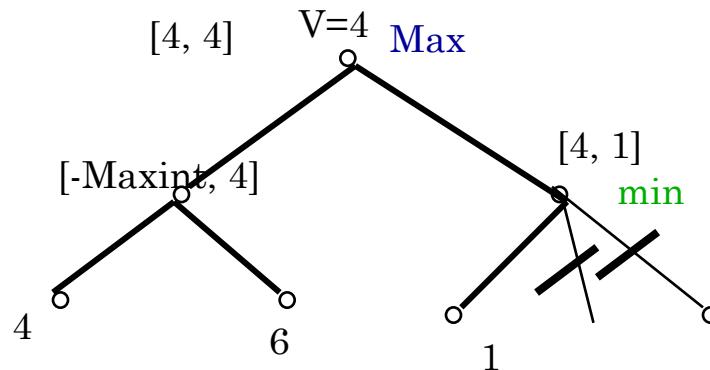
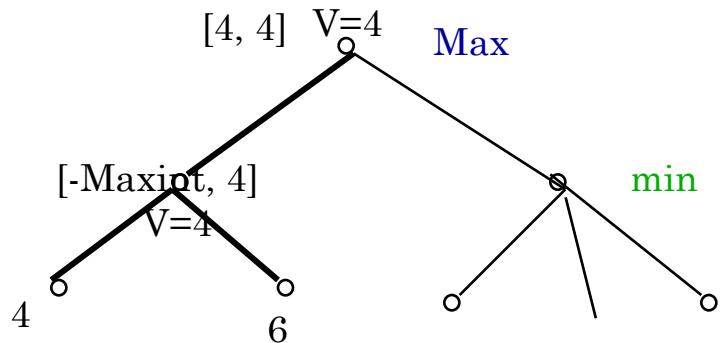
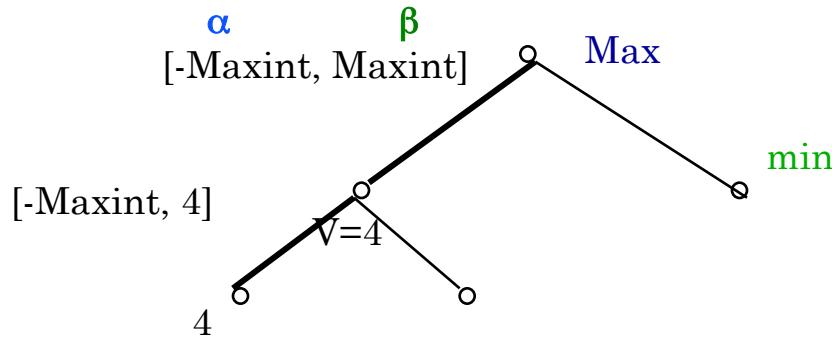
Aplique ao próximo sucessor já os novos valores de α e β

SE $\alpha < \beta$ RETORNE v= β SENÃO RETORNE v= α

% Corte α

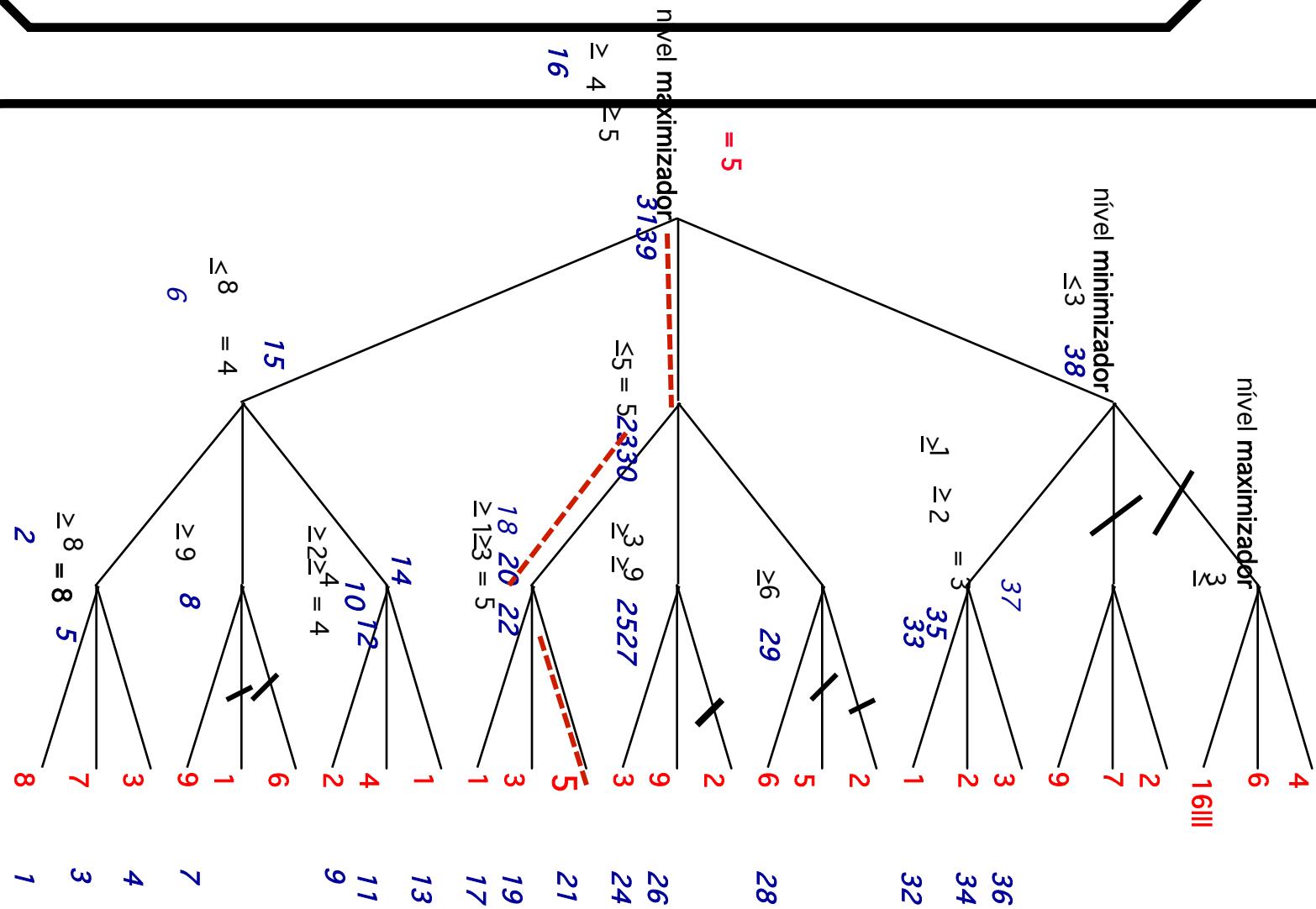
INTELIGÊNCIA ARTIFICIAL

Procedimento básico do Minimax com cortes $\alpha\beta$



V na raiz = $\max(\min(4, 6), \min(1, X)) = \max(4, Y)$ em que $Y \leq 1 \rightarrow V=4$

INTELIGÊNCIA ARTIFICIAL



INTELIGÊNCIA ARTIFICIAL

Algoritmo Negamax

Este algoritmo é semelhante ao Minmax mas tira partido das heurísticas poderem ser as mesmas para o jogador e seu adversário (como é o caso do Xadrez)

Enquanto que o Minmax maximiza a heurística do jogador e **minimiza** a do adversário, o Negamax nega (**multiplica por -1**) o valor da heurística correspondente ao nível onde teria de **minimizar**. Assim não necessita de saber qual é o nível em que se encontra e **maximiza sempre**.

```
function negamax(node, depth,  $\alpha$  ,  $\beta$  , color)
    if node is a terminal node or depth = 0
        return color * the heuristic value of node
    else
        foreach child of node
             $\alpha$  := max(  $\alpha$  , -negamax(child, depth-1, -  $\beta$  , -  $\alpha$  , -color))
            {the following if statement constitutes alpha-beta pruning}
            if  $\alpha \geq \beta$       break
    return  $\alpha$ 
```

$$\text{Max}(a,b) = - \text{Min}(-a,-b)$$
$$\text{Min}(a,b) = -\text{Max}(-a,-b)$$

INTELIGÊNCIA ARTIFICIAL

explicação em O'Reilly:
"Algorithm in a Nut Shell, Chpt7"

Algoritmo Negamax

Este algoritmo é semelhante ao Minmax mas tira partido das heurísticas poderem ser as mesmas para o jogador e seu adversário (como é o caso do Xadrez)

Enquanto que o Minmax maximiza a heurística do jogador e **minimiza** a do adversário, o Negamax nega (**multiplica por -1**) o valor da heurística correspondente ao nível onde teria de **minimizar**. Assim não necessita de saber qual é o nível em que se encontra e **maximiza sempre**.

```
função negamax(nó, prof, α , β , cor)
    if nó é terminal ou prof= 0
        return cor * valor heurístico do nó
    else
        gerar sucessores (nó)
        melhor_Valor := - maxint
        foreach sucessor
            V:= -negamax(sucesor, prof-1, -β , -α , -cor)
            melhorValor:= max(melhorValor,V)
            α := max(α ,V)
            if α ≥ β      falha {cortes alpha-beta}
        return melhorValor
```

Max(a,b)=- Min(-a,-b)
Min(a,b)=Max(-a,-b)
inicialização:
cor= 1
α :=- Maxint
β := Maxint

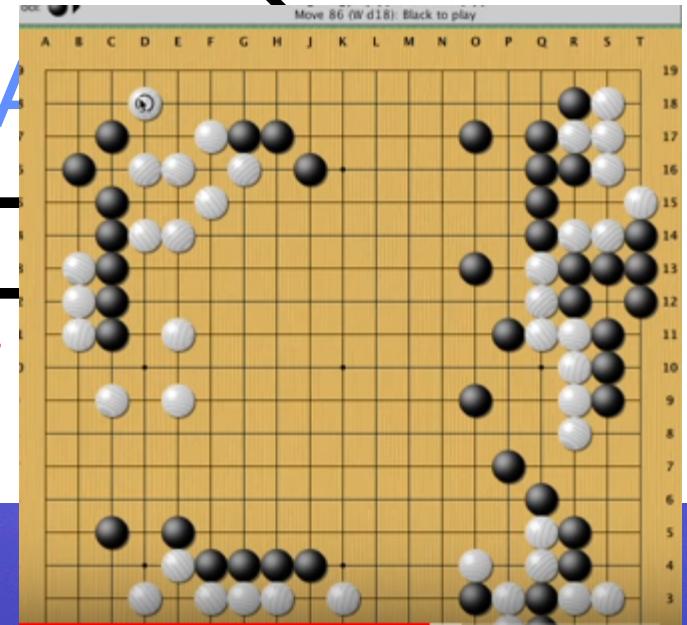
INTELIGÊNCIA ARTIFICIAL

Algoritmo **MCTS**- “Monte Carlo Tree Search”

Programas para jogos complexos sem super poder computacional.

Tabuleiro do GO é 19 X 19 e o fator de ramificação inicial de 361 !!

Difícil especificar função de avaliação porque o domínio do território não é evidente nas várias fases do jogo



The Future of Go



INTELIGÊNCIA ARTIFICIAL

At the 2017 Future of Go Summit, AlphaGo beat Ke Jie, the world No.1 ranked player at the time, in a three-game match.

Algoritmo MCTS- "Monte Carlo Tree Search"

- Algoritmo "Monte Carlo Tree Search" tem sido usado em jogos não determinísticos como poker e teve grande sucesso vencendo no jogo GO
- ALPHAGO (de Google DeepMind) venceu o Campeão mundial de GO usa MCTS e DL para aprender os melhores valores nos nós alternativos
- ALPHAGO combina o algoritmo de pesquisa em árvore com "Deep Neural Networks"
- Aprendeu com milhões de jogadas anteriores de jogos vencedores
- Aprendeu depois com jogos entre várias versões de si próprio através de "Reinforcement Learning"

MCTS expande na árvore de pesquisa os **nós** considerados **mais promissores**.

Baseia-se em muitas **jogos** feitos anteriormente com **movimentos aleatórios** e cujo resultado servirá para pesar o valor dos nós (estados) por onde passou. Os **melhores nós pertencem a movimentos de jogos ganhadores**

INTELIGÊNCIA ARTIFICIAL

Método baseado em Diferença Temporal para calcular o Valor estimado de um estado $V(s)$

Algoritmo "Reinforcement Learning"

$$Q(s, a) = Q(s, a) + \alpha [r + \gamma \max_a Q(s', a') - Q(s, a)]$$

onde $0 < \alpha < 1$ é a **taxa de aprendizagem**, representando um impacto da atualização no valor corrente

r é o **reforço** obtido pela execução de a em s

$0 < \gamma < 1$ é o **fator de desconto** representando a importância dos valores Q nos estados futuros para o valor a atualizar ($\gamma = 0$ só conta o reforço imediato)

Escolha das Ações

Selecionar a ação com melhor valor de Qualidade

Em ambientes dinâmicos não determinísticos essa qualidade pode alterar-se. Deve experimentar novas hipóteses.

Compromisso entre "potenciar" o conhecido ("exploitation") e "explorar" o novo ("exploration")

INTELIGÊNCIA ARTIFICIAL

Algoritmo "Reinforcement Learning"

A probabilidade $P(a_t, s)$ da escolha da ação a no instante t é, pela fórmula Softmax:

$$P(s, a_t) = \frac{e^{Q_t(s,a)/\tau}}{\sum_{b \in A} e^{Q_t(s,b)/\tau}}$$

INTELIGÊNCIA ARTIFICIAL

Algorítmo MCTS

"Pure Monte Carlo Game Search": O mesmo número de jogos até ao fim são realizados **antes** de cada movimento

4 passos em cada ronda do MCTS:

Seleção: inicia na raiz R e **seleciona nós** sucessores até às folhas;

Expansão: ← Excetuando para folhas L (Estados terminais do jogo com Vitória/Derrota crie os respetivos sucessores e escolha um deles C ;

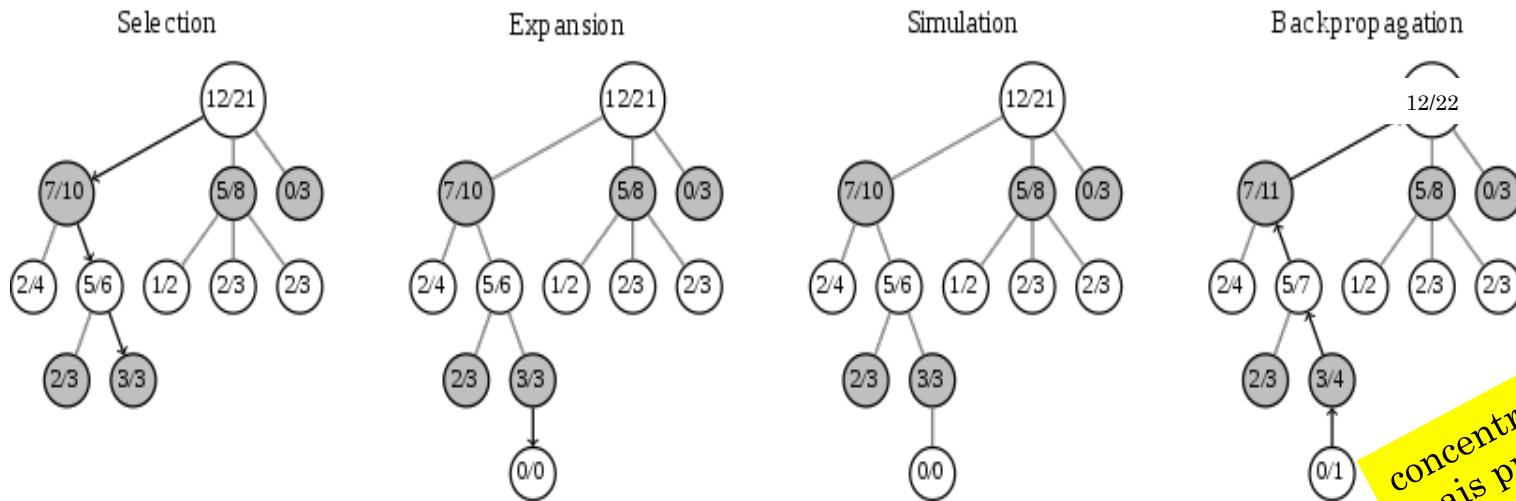
Simulação ("playout"): **Jogue o jogo até ao fim** a partir de C

Retropropagação("backpropagation"): Use os resultados desse jogo para **atualizar o valor dos nós** no caminho de C a R

INTELIGÊNCIA ARTIFICIAL

Algoritmo MCTS

Passos em uma ronda. Nodos contém Vitórias/Jogos



concentra-se nos nós
mais promissores

Antes de cada movimento realizam-se tantas rondas quantas o tempo o permitir.

A seleção dos nós deve ter em conta o **número de simulações realizadas e de vitórias**.

Tende para o ótimo quando o nº de rondas tende para infinito

INTELIGÊNCIA ARTIFICIAL

“Exploitation” Versus “Exploration”

Por vezes deve selecionar-se nós com poucas simulações (“Exploration”)

Uma fórmula para balancear “Exploitation” e “Exploration” é a da UCT
(Upper Confidence Bound 1 aplicada a árvores) *Introduzida por:*
Levente Kocsis e Csaba Szepesvári.

- Seleciona-se o nó que resultar no maior valor da fórmula:

$$w_i/n_i * c \sqrt{(\log(N_t)/n_i)} \quad (\text{log. natural})$$

- w_i vitórias depois do movimento i -ésimo;
- n_i simulações no nó, depois do movimento i -ésimo;
- c parâmetro de exploração — teoricamente igual a $\sqrt{2}$; na prática escolhido empiricamente;
- N_t número de simulações até ao momento. Igual a soma de todos os n_i .
- A primeira componente da fórmula corresponde à “**exploitation**” (maior para nós com maior razão Vit/Simu). Segunda corresponde à “**exploration**”. Maior para nós com menos simulações

INTELIGÊNCIA ARTIFICIAL

Vantagens Vs Desvantagens

MCTS converge para o Minimax mas muito devagar.

Vantagem relativamente ao Minimax alfa-beta:

- Não necessita de explicitar a **função de avaliação**.
- Basta implementar a **mecânica do jogo**: expandir para os sucessores possíveis e reconhecer o fim do jogo.
- Aplica-se a **qualquer** jogo com adversários
- Vai **reduzindo muito o fator de ramificação** pois concentra-se nos nós mais promissores.
- **Interrompido** em qualquer momento consegue dar o movimento mais promissor encontrado até aí pois resulta de jogos completos anteriores.