Algoritmos de busca com e sem informação

Airton Bordin Junior

[airtonbjunior@gmail.com]

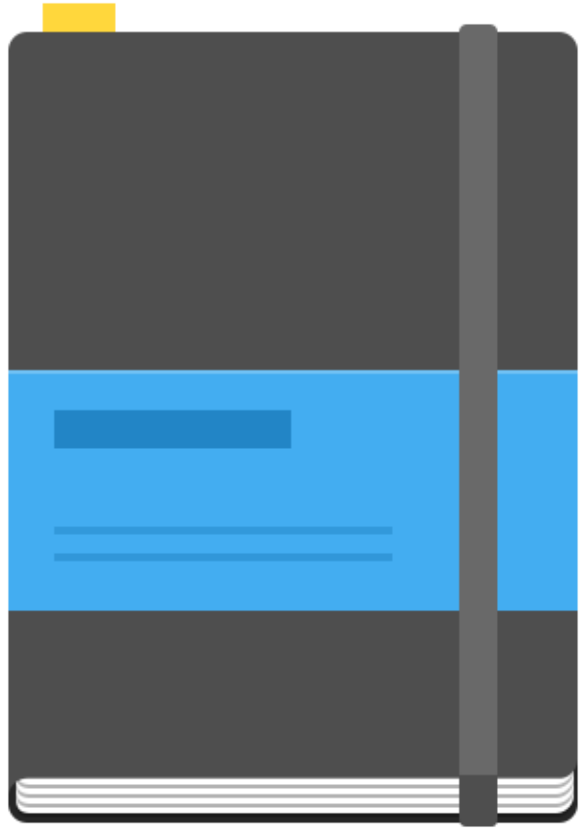
Metaheurísticas - Prof. Dr. Celso Gonçalves Camilo Junior

Mestrado em Ciência da Computação 2017/2

Universidade Federal de Goiás (UFG) - Instituto de Informática – Agosto/2017

Programação

- Introdução
- Busca informada
- Busca não informada
- Busca local e otimização
- Referências



Introdução

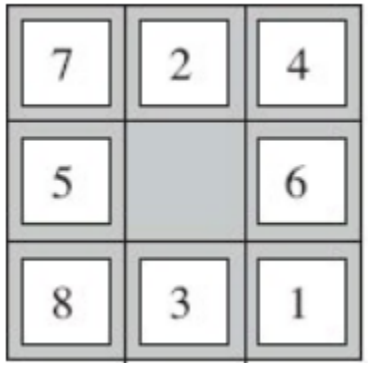
- **Formulação de problemas:** processo de decidir que ações e estados devem ser considerados, dado um objetivo;
- **Objetivo:** conjunto de estados do mundo que deseja alcançar;
- **Busca:** processo de procurar a sequência de ações que alcançam o objetivo.

Problema bem definido

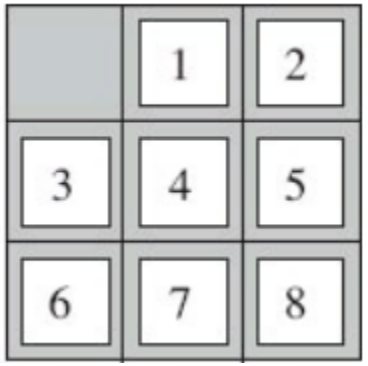
- Componentes
  1. Estado inicial;
  2. Ações;
  3. Modelo de transição;
  4. Teste de objetivo
  5. Custo de caminho.
- Ambiente do problema: representado pelo espaço de estados
  - Solução: um caminho do estado inicial ao estado objetivo.

Problema bem definido

Quebra-cabeça de oito peças



Estado inicial



Estado objetivo

Estado inicial: Qualquer estado;

Ações: A formulação mais simples define as ações como movimentos do quadrado vazio Esquerda, Direita, Para Cima ou Para Baixo. Pode haver subconjuntos diferentes desses, dependendo de onde estiver o quadrado vazio;

Modelo de transição: Dado um estado e ação, ele devolve o estado resultante; por exemplo, se aplicarmos Esquerda para o estado inicial na figura, o estado resultante terá comutado o 5 e o vazio;

Teste de objetivo: Verifica se o estado corresponde à configuração de estado objetivo mostrada na Figura (são possíveis outras configurações de objetivos);

Custo de caminho: Cada passo custa 1 e, assim, o custo do caminho é o número de passos do caminho.

Algoritmos de busca

- Analisados em termos de
  - Completeza;
  - Otimização;
  - Complexidade de tempo;
  - Complexidade de espaço.
- Complexidade
  - *b:* fator de ramificação no espaço de estados;
  - *d:* profundidade da solução mais rasa.

Algoritmos de busca

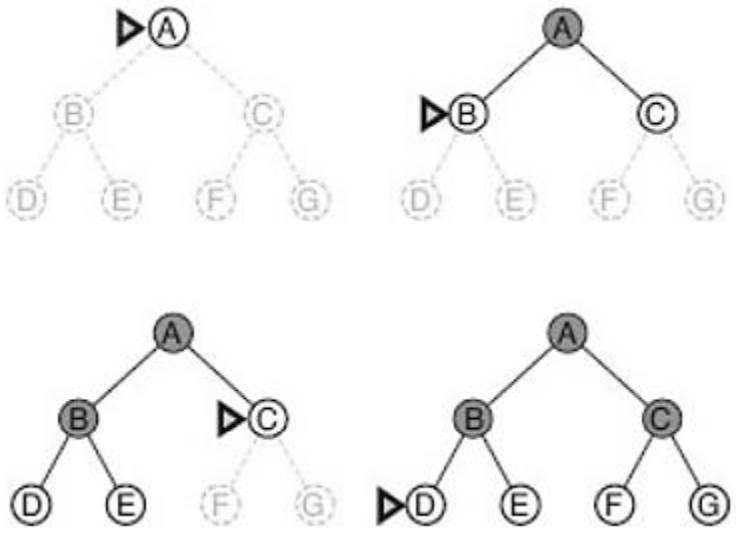
- Analisados em termos de
  - Completeza
    - O algoritmo oferece a garantia de encontrar uma solução quando ela existir?
  - Otimização
    - A estratégia encontra a solução ótima?
  - Complexidade de tempo
    - Tempo necessário para encontrar a solução.
  - Complexidade de espaço.
    - Memória necessária para encontrar a solução.

Busca não informada

- Possuem acesso apenas à definição do problema;
- Algoritmos básicos
  - Busca em largura;
  - Busca de custo uniforme;
  - Busca em profundidade;
  - Busca de aprofundamento iterativo;
  - Busca bidimensional.

Busca não informada

- Busca em largura



Completeza: Sim (b finito)
Otimização: Sim (se custo for igual para todos os passos)
Complexidade tempo: 1+b+b2+b3+... +bd = O(b^d)
Complexidade espaço: O(b^d) (mantém todos os nós na memória)

Busca não informada

- Busca de custo uniforme
  - Extensão da Busca em Largura para encontrar a solução ótima para qualquer valor de passo;
  - Expande o nó n que apresenta o menor custo de caminho g(n);
  - Teste de objetivo é realizado quando um nó é selecionado para expansão (ao invés de realizar quando o nó é gerado).

Busca não informada

- Busca de custo uniforme

Completeza: garantido caso cada passo tiver um custo maior que ε (com ε > 0 e pequeno)

Otimização: garantido caso expandir os nós respeitando g(n)

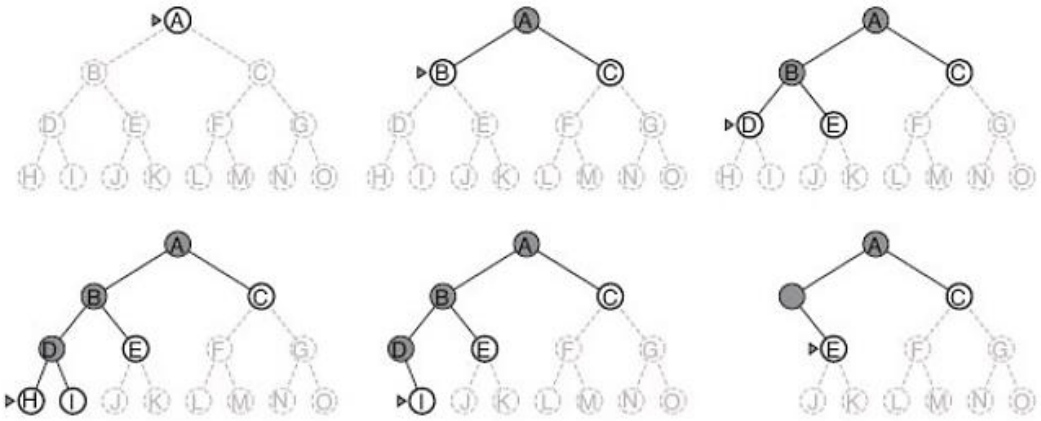
Complexidade tempo: $O(b^{b1+[C*/\varepsilon]})$

Complexidade espaço: $O(b^{b1+[C*/\varepsilon]})$

C*: custo da solução ótima

Busca não informada

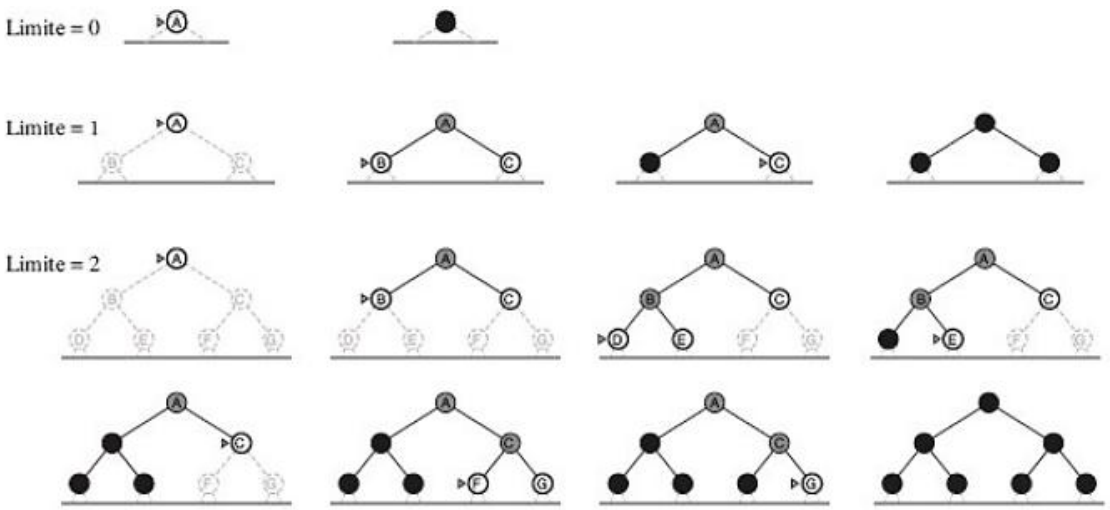
- Busca em profundidade



Completeza: Não
Otimização: Não (para na primeira solução encontrada)
Complexidade tempo: O(b^m)
Complexidade espaço: O(b*m)

Busca não informada

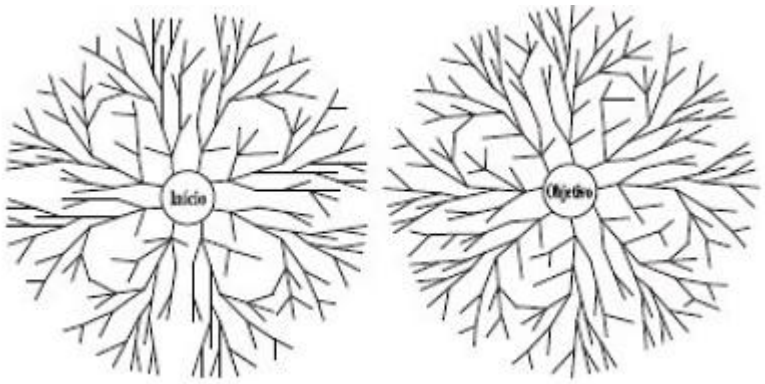
- Busca de aprofundamento iterativo



Completeza: Sim (b finito)
Otimização: Sim (se custo for igual para todos os passos)
Complexidade tempo: O(b^d)
Complexidade espaço: O(b*d)

Busca não informada

- Busca bidimensional



Completeza: Sim (b finito e ambos usam busca em largura)
Otimização: Sim (se custo for igual para todos os passos e amos usam busca em largura)
Complexidade tempo: $O(b^{d/2})$
Complexidade espaço: $O(b^{d/2})$

Busca informada

- Podem ter acesso a uma função heurística *h(n)* que estima o custo da solução a partir de *n*;
- Uma função heurística *h(n)* deve ser capaz de estimar o custo de uma solução começando pelo estado do nó n.
  - Como construir tal função?
    - Conceber **problemas relaxados** para os quais uma solução ótima pode ser facilmente encontrada;
    - Aprender com a **experiência**.

Busca informada

- Podem ter acesso a uma função heurística *h(n)* que estima o custo da solução a partir de *n*;
- Algoritmos básicos
  - Busca gulosa de melhor escolha;
  - Busca A*;
  - Busca recursiva de melhor escolha (RBFS) e busca A* de memória limitada (SMA*).

Busca informada

- Busca de melhor escolha
  - Instância do algoritmo geral da Busca em Árvore;
  - Nó é selecionado para expansão com base em uma função de avaliação *f(n);*
  - *f(n)* é analisada como uma estimativa de custo
    - Nó com a menor avaliação será expandido primeiro.
  - A maior parte dos algoritmos de melhor escolha inclui como componente de *f* uma função heurística *h(n)*
    - Custo estimado do caminho de menor custo do estado do nó *n* para um estado objetivo. (*n* for objetivo, *h(n)* = 0)

Busca informada

- Busca gulosa de melhor escolha
  - Tenta expandir o nó que está mais próximo do objetivo
    - Isso pode conduzir a uma solução rapidamente.
  - Avalia os nós usando apenas a função heurística (*f(n) = h(n)*);
  - Desempenho depende da qualidade de *h(n)*;

Busca informada

- Busca A*
  - *f(n) = g(n) + h(n)*
    - *g(n): custo para alcançar o nó*
    - *h(n): custo para ir do nó até o objetivo*
  - Condições para ser ótima: **admissibilidade** e **consistência**

Busca informada

- Busca A*
  - **Heurística admissível**: nunca superestima o custo de atingir o objetivo;
  - **Heurística consistente**: para cada nó *n* e para todo sucessor *n'* de *n* gerado por uma ação *a*, o custo estimado de alcançar o objetivo de *n* não é maior do que o custo do passo de chegar a *n'* + o custo estimado de alcançar o objetivo de *n'*.
    - $h(n) \leq c(n, a, n') + h(n')$

Busca informada

- Busca recursiva de melhor escolha
  - Tenta imitar a operação de busca padrão pela melhor escolha usando apenas um espaço linear de memória;
  - Semelhante a busca em profundidade recursiva
    - Em vez de continuar indefinidamente seguindo o caminho atual, usa a variável *f_limite* para acompanhar o *f-valor* do melhor caminho alternativo disponível de qualquer ancestral do nó atual.

Busca informada

- Busca A* simplificada de memória limitada
  - Procede exatamente como o A*, expandindo a melhor folha até que a memória esteja cheia. Nesse ponto, não poderá adicionar um novo nó à árvore de busca sem suprimir um antigo;
  - Sempre suprime o pior nó folha (maior *f_valor*).
- Faz o *backup* do valor do nó esquecido em seu pai
  - Ancestral de uma subárvore esquecida conhece a qualidade do melhor caminho daquela subárvore.
- Se todos os descendentes de um nó *n* forem esquecidos, não saberemos para onde ir a partir de *n*, mas ainda teremos uma ideia de como vale a pena ir a algum lugar de *n*.

Busca informada

- Geração de heurísticas admissíveis
  - **Problema relaxado: p**roblema com poucas restrições sobre as ações;
  - Qualquer solução ótima do problema original será uma solução do problema relaxado;
    - Problema relaxado pode ter melhores soluções
  - Custo de uma solução ótima para um problema relaxado é uma **heurística admissível** para o problema original.

Busca informada

- Bancos de dados de padrões
  - Armazenar os custos exatos de solução para todas as instâncias possíveis do subproblema;
  - Heurística admissível *hBD* para cada estado completo encontrado durante uma busca
    - Exame da configuração correspondente do subproblema no banco de dados.
  - O próprio banco de dados é construído através de busca reversa do objetivo e do registro do custo de cada novo padrão encontrado..

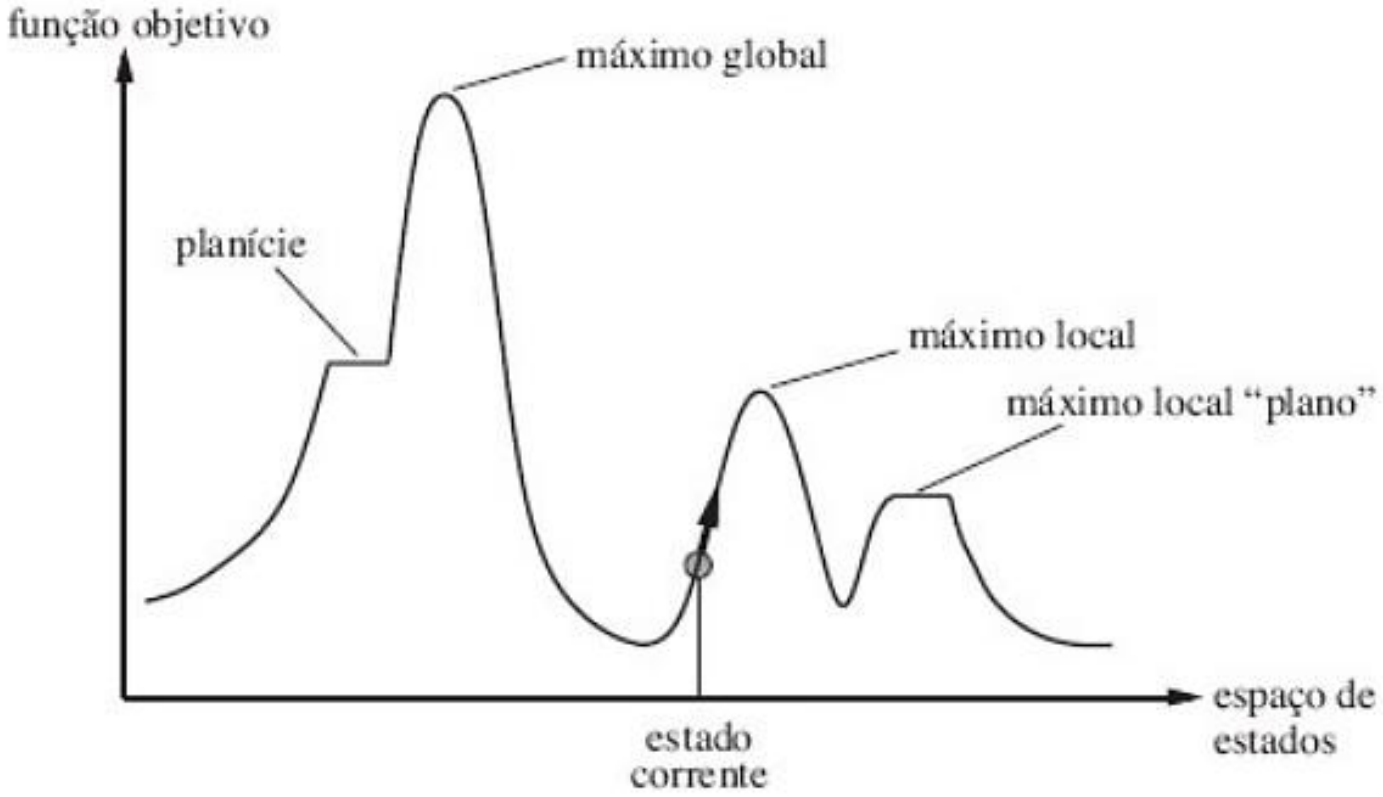
Busca local e otimização

- Quando o caminho até o objetivo não importa, podemos considerar uma classe diferente de algoritmos
  - Algoritmos de busca local operam usando um único estado atual e, em geral, se movem apenas para os vizinhos desse estado.
- Úteis para **problemas de otimização**
  - Encontrar o melhor estado de acordo com uma função objetivo.

Busca local e otimização

- Vantagens
  - Usam pouquíssima memória (normalmente um valor constante);
  - Frequentemente podem encontrar soluções razoáveis em grandes ou infinitos (contínuos) espaços de estados para os quais os algoritmos sistemáticos são inadequados.

Busca local e otimização



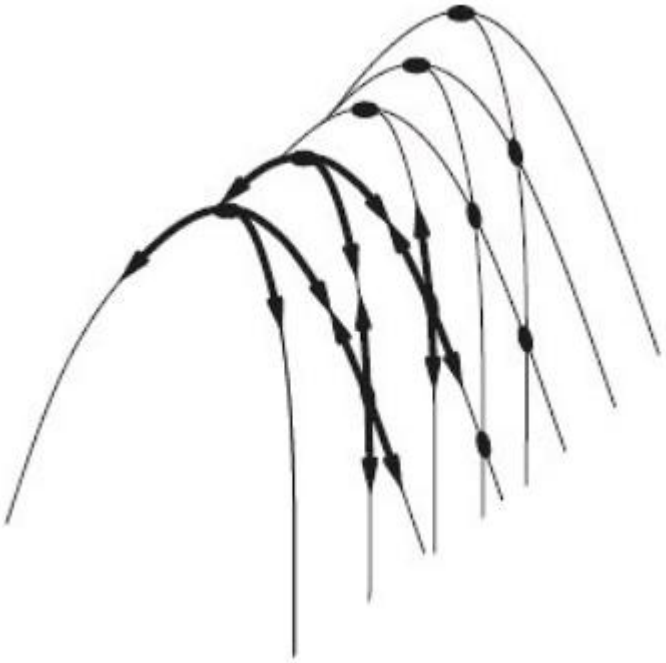
Topologia de espaço de estados

- **Algoritmo completo** sempre encontra um objetivo (caso exista)
- **Algoritmo ótimo** sempre acha um mínimo/máximo global

Busca local e otimização

- Busca de subida de encosta
  - Se move de forma contínua no sentido do valor crescente;
  - Termina quando alcança um “pico” em que nenhum vizinho tem valor mais alto;
  - Com frequência fica paralisada
    - Máximos locais;
    - Cordilheiras;
    - Platôs.

Busca local e otimização



Cordilheiras

- Sequência de máximos locais que torna muito difícil a navegação para algoritmos gulosos;
- A partir de cada máximo local, todas as ações disponíveis apontam encosta abaixo

Busca local e otimização

- Têmpera simulada
  - Escolhe um movimento aleatório;
    - Se o movimento melhorar a situação sempre será aceito;
  - Caso contrário, o algoritmo aceitará o movimento com alguma probabilidade menor que 1;
    - Probabilidade decresce exponencialmente com a “má qualidade” do movimento — o valor ΔE segundo o qual a avaliação piora.

Busca local e otimização

- Têmpera simulada
  - Probabilidade também decresce à medida que a "temperatura" T se reduz;
  - Movimentos "ruins" têm maior probabilidade de serem permitidos no início, quando T estiver alto. Tornam-se mais improváveis conforme T diminui;
  - Se o escalonamento diminuir T com lentidão suficiente, o algoritmo encontrará um valor ótimo global com probabilidade próxima de 1.

Busca local e otimização

- Busca em feixe local
  - Mantém o controle de *k* estados (em vez de um);
  - Começa com *k* estados gerados aleatoriamente;
  - A cada passo, são gerados todos os sucessores de todos os *k* estados
    - Se um deles for um objetivo, o algoritmo para.
  - Se não for, seleciona *k* melhores sucessores a partir da lista completa para repetir o procedimento.

Busca local e otimização

- Algoritmos genéticos
  - Variante de busca em feixe estocástica na qual os estados sucessores são gerados pela combinação de dois estados pais (em vez de único estado);
  - Reprodução sexuada;

Referências

- RUSSELL, Stuart; NORVIG, Peter; INTELLIGENCE, Artificial. A modern approach. **Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs**, v. 25, p. 27, 1995.