Inteligência Artificial

Professor: José Eurípedes Ferreira de Jesus Filho
jeferreirajf@gmail.com

Universidade Federal de Jataí – UFJ

Aula anterior

- Problemas **bem definidos**.
  - Estado inicial;
  - Conjunto de ações;
  - Espaço de estados;
  - Teste de objetivo;
  - Custo do caminho.

- Buscas:
  - Busca em **largura**;
  - Busca em **profundidade**;
  - Busca em **profundidade limitada**.

- Repetição de estados.

Agenda

- Introdução.
- Busca com informação.
- Heurísticas.
- Problemas de otimização.
- Busca local.
- Busca on-line.

Introdução

- Agentes da aula anterior.
    - Buscavam uma solução varrendo sistematicamente o espaço de busca.
        - Partiam de um estado inicial;
        - Geravam sistematicamente novos estados;
        - Avaliavam os novos estados de acordo com o objetivo;
        - Paravam uma vez que o objetivo fosse satisfeito/alcançado.

- A varredura sistemática na maioria dos casos é ineficiente!
    - Espaço de busca é gigantesco;
    - Impossibilidade de gerar sistematicamente novos estados;
    - Problemas de recurso computacional;
    - Problemas NP-Difíceis;

Busca com informação

- Busca com informação.
    - Não se prende somente a definição do problema;
    - Utiliza algum conhecimento específico do problema no processo de busca;

Heurísticas

- *Best-First*.
  - Especialização generalizada da busca em árvore;
  - Baseada em uma **função de avaliação** $f(n)$ que consegue avaliar o quão bom é um dado nó n.
  - Dado a fila de estados do processo de busca em árvore, seleciona-se para expansão sempre o melhor, de acordo com a função $f(n)$.
  - Dessa forma, a ideia é que se você sempre expande o melhor estado possível, mais rapidamente você consegue chegar na solução!

Heurísticas

- Heurísticas.
  - A função $f(n)$ é utopia na maior parte dos casos, pois se soubéssemos uma função que sempre calcula o melhor estado possível, faríamos um caminho direto até a solução ótima.
  - Podemos compor (ou até substituir) $f(n)$ utilizamos uma função $h(n)$, que faz uma **estimativa** do quão bom é o nó (estado) n.
  - $h(n)$ é chamado de **função heurística**.
  - No geral, **definimos $\boldsymbol{h(n)}$** como custo estimado do caminho mais econômico (ou de maior recompensa) do nó n até um nó objetivo.
  - Algoritmos heurísticos fazem uso da função $h(n)$.
  - Existe uma família inteira de algoritmos que fazem uso de diferentes funções heurísticas.

Heurísticas

- Busca gulosa.
    - Expande a árvore de busca sempre para o nó (estado) mais próximo do objetivo (meta).
    - A ideia é que esse tipo de escolha obterá uma solução mais rapidamente.
    - Para este tipo de heurística, $f(n) = h(n)$ puramente.
    - Nem sempre a minimização de $h(n)$ é ótimo!
    - Portanto, a busca gulosa não é ótima.
    - Também não é completa.
    - Heurísticas gulosas, no entanto, são soluções ótimas de vários problemas clássicos!

Heurísticas

- Busca A^*.
  - Considera o custo de chegar ao próximo estado e o custo do próximo estado até o objetivo.
  - $f(n) = g(n) + h(n)$.
  - Na função, $g(n)$ fornece o custo do nó inicial até o nó atual n e $h(n)$ é o custo estimado de n até o objetivo.
  - $g(n)$ portanto sempre é um valor conhecido, exato.
  - Se $h(n)$ for uma **heurística admissível**, A^* será ótima para busca em grafo. É possível torná-la completa também.
  - Heurísticas admissíveis são aquelas que nunca superestimam o custo real para alcançar o objetivo. Ou seja, $g(n) + h(n) \leq g(n) + h^*(n) = f^*(n)$.
  - Dizemos então que $h(n)$ para A^* sempre será uma heurística otimista, quer dizer, sempre imagina que o custo será menor do que realmente é.

Heurísticas

- Busca A^*.
  - Quando trabalhamos com grafos, queremos assegurar que o caminho ótimo sempre será o primeiro a ser seguido.
  - Isso evita que a heurística descarte por conta de caminhos repetidos no grafo o caminho ótimo.
  - Essa propriedade é válida se impusermos o **requisito de consistência** ou também chamado de **requisito de monotonicidade**.
  - Uma heurística $h(n)$ é consistente se dado um estado atual n, um próximo estado n' atingido a partir de uma ação a, temos:

    $h(n) \leq c(n, a, n') + h(n')$,
  - Quer dizer, o custo estimado de alcançar o objetivo a partir de n sempre será menor ou igual ao custo do passo para chegar em n' somado ao custo de chegar ao objetivo a partir de n'.

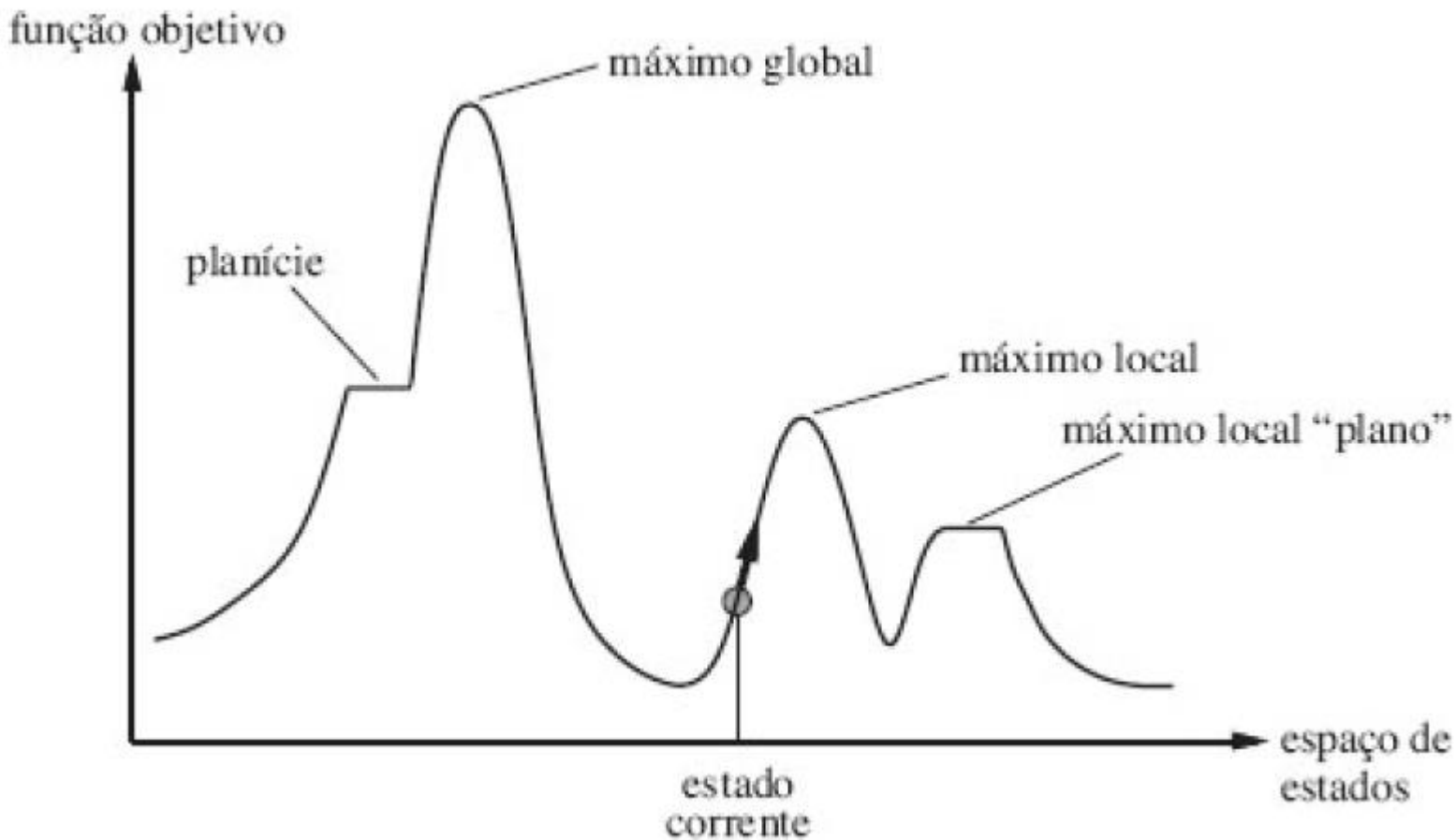
Heurísticas

- Busca A^*.
  - A^* é uma heurística ótima para busca em grafos se $h(n)$ é consistente.
  - Se $h(n)$ é consistente então os valores de $f(n)$ ao longo de qualquer caminho são não-decrescentes.
  - Ainda temos que A^* é uma heurística **otimamente eficiente**, quer dizer, nenhum outro algoritmo tem a garantia de expandir um número menor de nós até encontrar o objetivo.
  - Infelizmente A^* não é praticável para a maior parte dos casos.
  - Em geral, o espaço de busca de A^* ainda é exponencial.

Heurísticas

- Problema relaxado.
    - Dado um problema original, a versão relaxada do problema é uma versão na qual admite-se menos restrições do que a versão original.
    - As restrições descartadas são chamadas de restrições relaxadas.
    - Pode-se ainda modificar algumas restrições para admitir soluções que não seriam viáveis ao problema original.
    - Relaxação de problemas é uma técnica heurística de resolução.

Problemas de otimização



Problemas de otimização

- Problemas irrestritos.
    - Problemas cujas soluções não precisam satisfazer restrições.
- Problemas restritos.
    - As soluções devem satisfazer restrições rígidas para os valores de uma ou mais variáveis.
    - São comuns em PO.
- Problemas lineares.
    - São problemas restritos nos quais as restrições são desigualdades lineares.
    - As restrições formam uma região convexa.
    - A função objetivo também deve ser linear.

Problemas de otimização

- Problemas irrestritos.
    - Possuem métodos exatos e rápidos como solução.
- Problemas lineares.
    - Possuem métodos exatos como solução.
    - Podem ser resolvidos em tempo polinomial no número de variáveis.
- Problemas não-lineares.
    - Problemas cuja função objetivo ou uma ou mais restrições não são lineares.
    - Possuem métodos exatos como solução.
    - Dependendo do problema, os métodos exatos ficam impraticáveis.

Busca local

- Caminho até o objetivo é irrelevante.
    - Problema das oito rainhas;
    - Problema de escalonamento de tarefas;
    - Problema de dimensionamento de lotes;
    - Problema de roteamento de veículos...

- Considera apenas utilizando o **estado atual**.
    - Não fazem uma busca sistemática pelo espaço de busca;
    - Não varre vários caminhos;
    - Geralmente se “**movem**” somente para os “**vizinhos**” do estado atual.

Busca local

- Busca de subida de encosta (*hill climbing*)

```
Função SUBIDA-DE-ENCOSTA(problema) retorna um estado que é um máximo local
    Entradas: problema, descrição de um problema
    Variáveis locais: atual, um nó; vizinho, um nó

    Atual ← CRIAR-NÓ( ESTADO-INICIAL[problema] )

    Repita
        vizinho ← um sucessor de atual com valor mais alto

        Se VALOR[vizinho] ≤ VALOR[atual] então retorne ESTADO[atual]

        atual ← vizinho
```

Busca local

- Subida de encosta estocástica.
    - Escolhe ao acaso os movimentos encosta acima.
    - A probabilidade de escolha geralmente é proporcional em relação a declividade do movimento.
- Subida de encosta pela primeira escolha.
    - Gera aleatoriamente diversos vizinhos escolhendo diversos movimentos ao acaso.
    - O primeiro vizinho melhor que a solução atual então assume o lugar da solução atual.
- É possível reiniciar as subidas a partir de **diferentes estados iniciais**.

Busca local

- Mais algoritmos heurísticos ou meta-heurísticos baseados em busca local:
    - Busca Tabu;
    - *Simulated Annealing*;
    - GRASP;
    - Algoritmos Genéticos;
    - Sistema Imune;
    - Etc...

Busca on-line

- Busca off-line.
    - A solução completa é inicialmente calculada.
    - Somente depois de se ter uma solução, o agente “entra” no mundo real para executar a solução.
    - Não utiliza a percepção.

- Busca on-line.
    - Intercala computação e ação.
    - Necessidade de percepção do ambiente para calcular a próxima ação.
    - Boa abordagem para sistemas dinâmicos ou semi-dinâmicos.
    - Necessária em problemas de exploração.

Busca on-line

- Problemas de busca-online.
    - Necessário um agente que também executa ações e não SOMENTE computação.
    - Só acessa um sucessor caso de fato execute uma ação.
    - É possível construir uma função $h(s)$ que avalia a distância do estado atual até o estado objetivo.
    - É normal comparar $h(s)$ com o valor real do custo do caminho que o agente de fato seguiria se conhecesse o espaço de busca com antecedência. Damos o nome de **razão competitiva** a essa comparação.
    - No geral, deseja-se minimizar essa razão.

Exercício

- O que são heurísticas?

Exercício

- Qual a filosofia da heurística *best-first*?

Exercício

- Defina busca gulosa e busca A^*.

Exercício

- O que são heurísticas admissíveis e o que é o requisito de consistência? Qual a relação entre os dois?

Exercício

- Por que agentes baseados em busca off-line não são bons em sistemas dinâmicos?