

Algoritmos de Busca I

Prof. Dr. Rafael Teixeira Sousa

UFMT

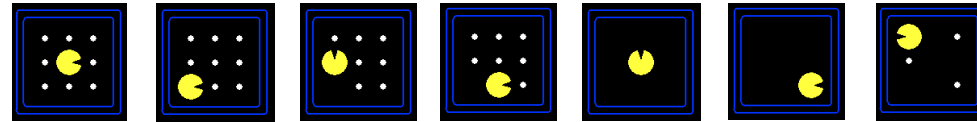
Outline

- Problema de Busca
- Busca sem informação:
 - Busca em profundidade
 - Busca em largura
- Busca com informação:
 - Busca Gulosa
 - Busca A*

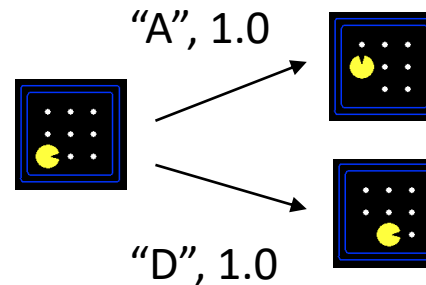
Problemas de busca

- Um problema de busca consiste em:

- Um **espaço de estados**



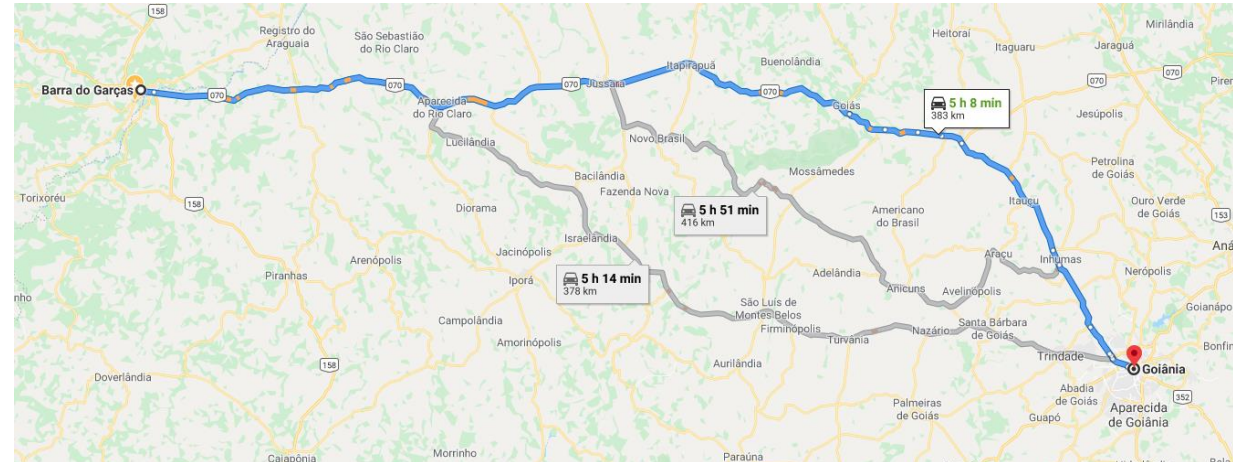
- Uma **função** de ações
(Com ações e custos)



- Um **estado inicial** e um **estado objetivo**
- **Uma solução** é uma sequência de ações (uma plano) que transforma o estado inicial no estado objetivo

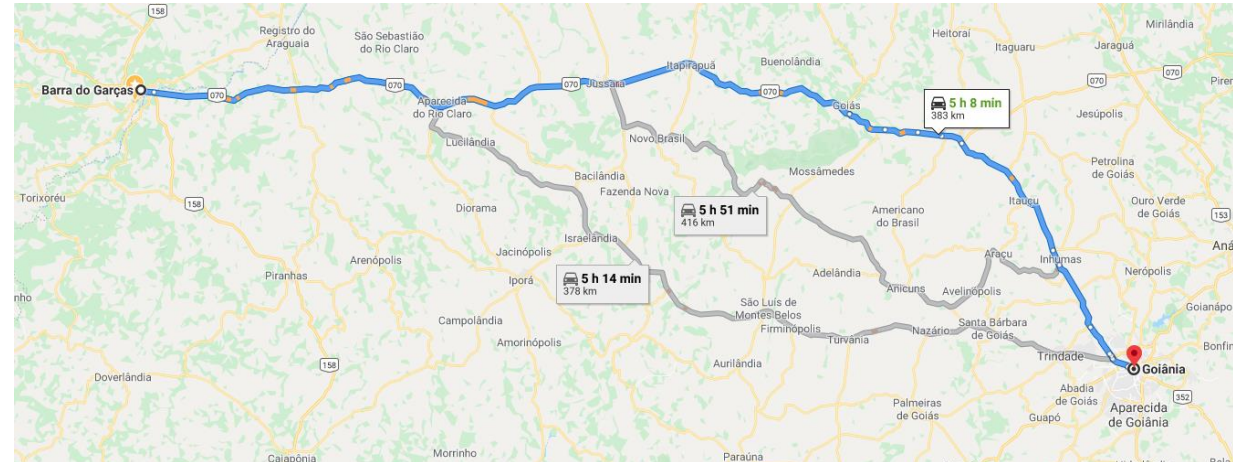
Exemplo

- Estado inicial:
 - Barra do Garças – MT
- Estado objetivo:
 - Goiânia – GO
- Espaço de estados:
 - Cidades
- Ações:
 - Dirigir em direção a cidades vizinhas



Exemplo

- Estado inicial:
 - Barra do Garças – MT
- Estado objetivo:
 - Goiânia – GO
- Espaço de estados:
 - Cidades
- Ações:
 - Dirigir em direção a cidades vizinhas
- Métrica de desempenho:
 - Distância
 - Tempo
 - Segurança



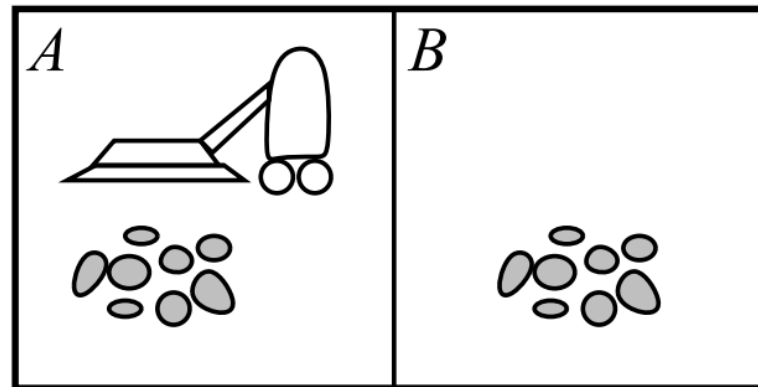
Exemplo

- Solução:
 - Procurar por sequência de ações que alcançam o objetivo



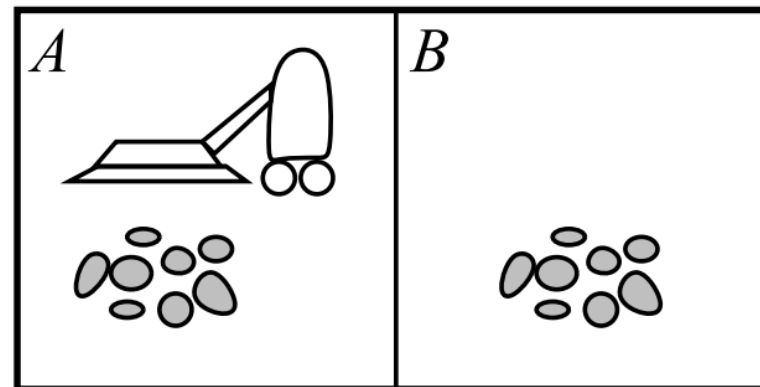
Aspirador de pó

- Percorre quadrados procurando por sujeira a limpar

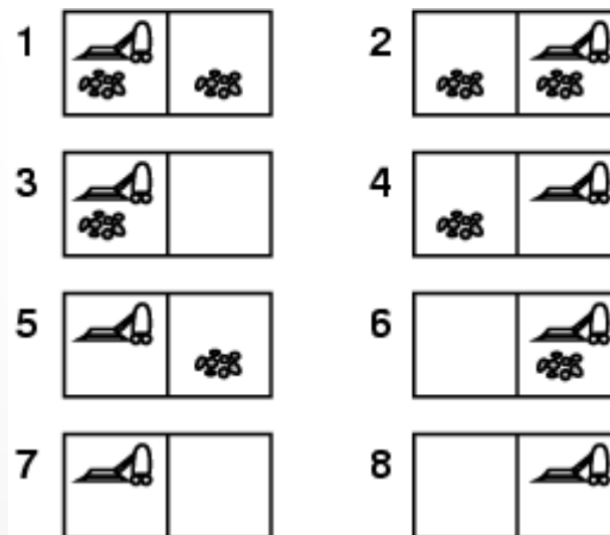


Aspirador de pó

- Percorre quadrados procurando por sujeira a limpar

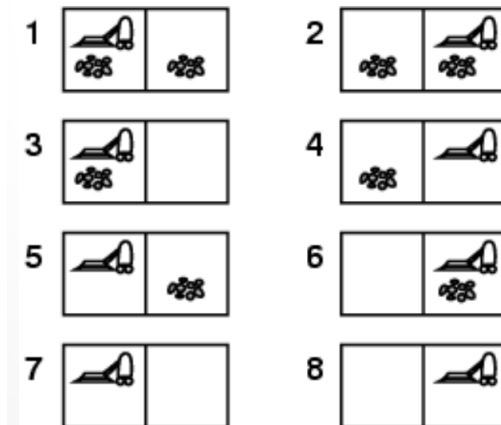
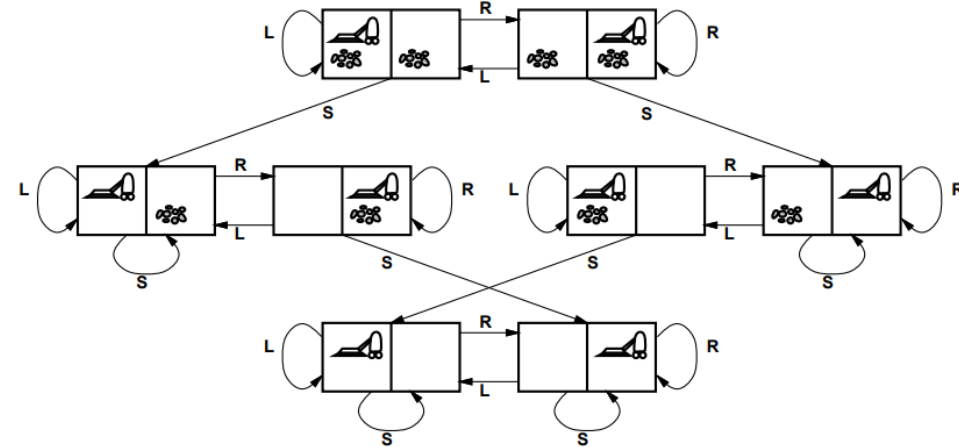


- Espaço de estados
 - 2 quadrados, contendo ou não sujeira
 - 8 estados possíveis



Aspirador de pó – grafo de estados

- Estados
- Ações
 - *Esquerda, Direita, Aspira, FazNada*
- Objetivo:
 - Não ter sujeira (estado 7 ou 8)
- Custo:
 - 1 por ação (exceto o *FazNada*)



Quebra-cabeça de 8 peças

- Estados:
 - Todas as possíveis posições dos números
- Ações:
 - Mover quadrado para esquerda, direita, acima e abaixo
- Objetivo:
 - Goal State
- Custo:
 - 1 por movimento

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

Quebra-cabeça de 8 peças

- Estados:
 - Todas as possíveis posições dos números
- Ações:
 - Mover quadrado para esquerda, direita, acima e abaixo
- Objetivo:
 - Goal State
- Custo:
 - 1 por movimento

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

NP-hard

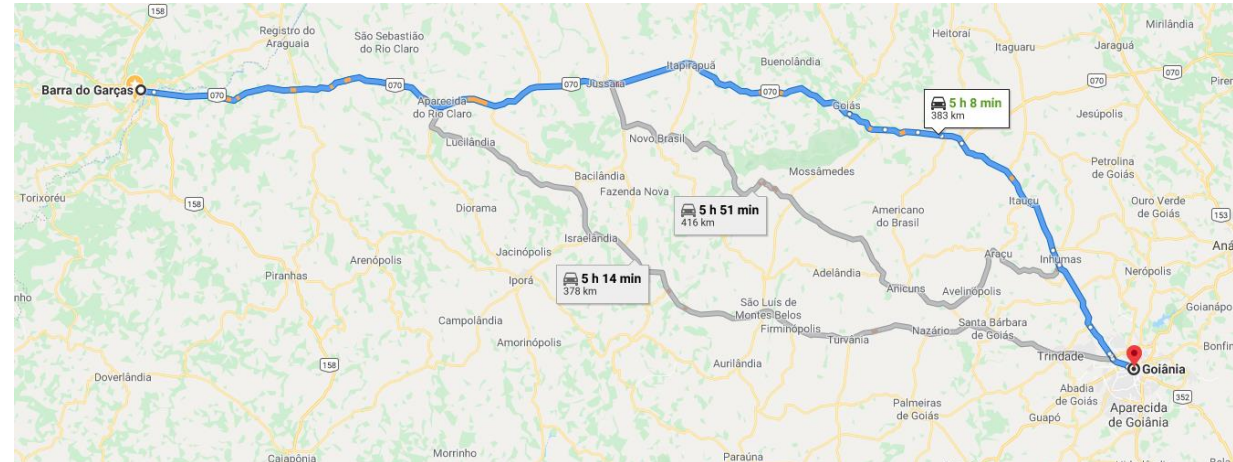
8 peças: 181 mil estados

15 peças: 1,3 trilhão de estados

24 peças: 10^{25} estados

Exemplo

- Estado inicial:
 - Barra do Garças – MT
- Estado objetivo:
 - Goiânia – GO
- Espaço de estados:
 - Cidades
- Ações:
 - Dirigir em direção a cidades vizinhas



Exemplo

- Estado inicial



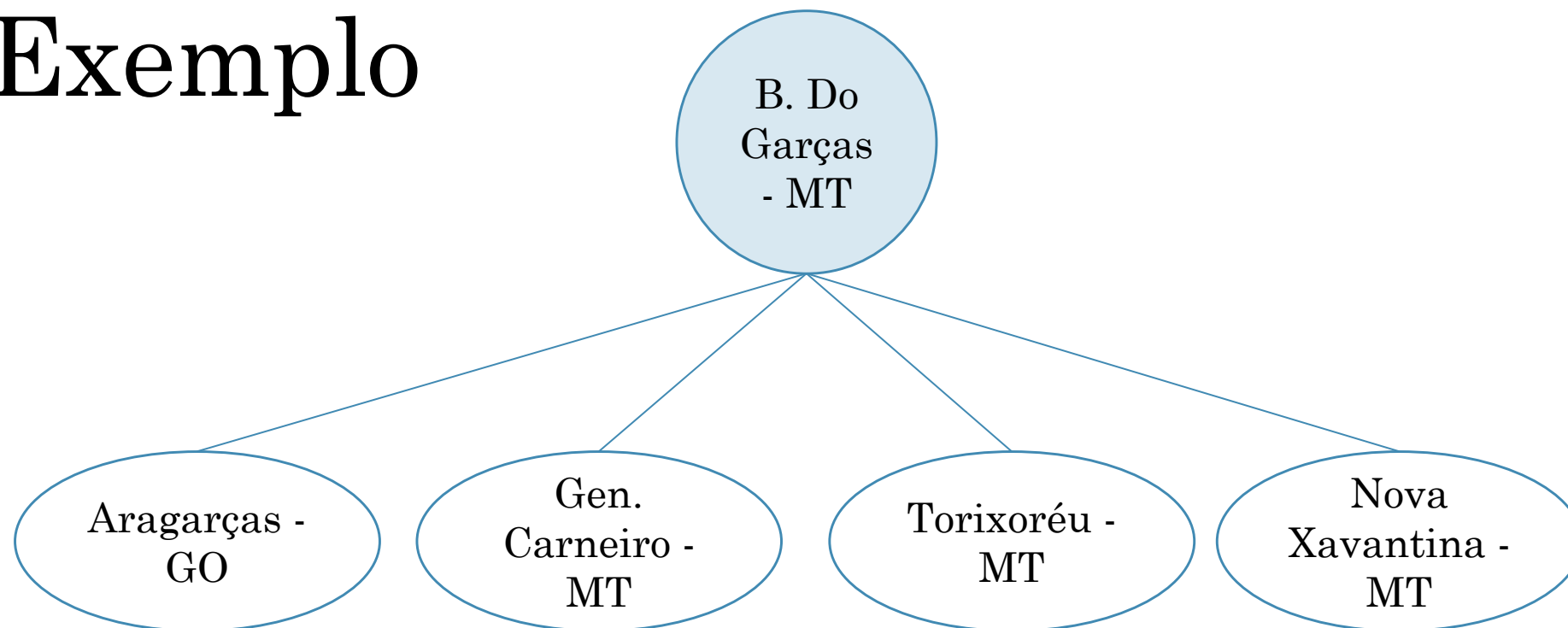
Exemplo

- Estado inicial



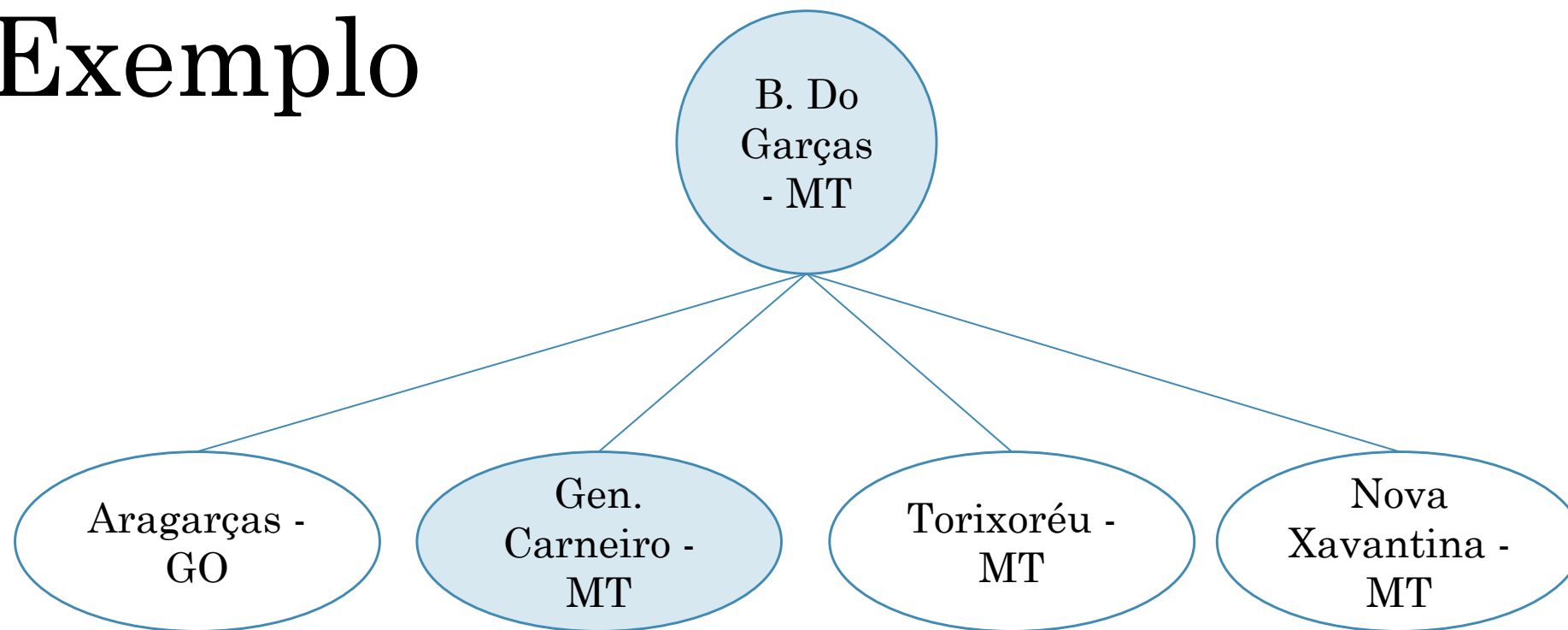
É objetivo? → não

Exemplo



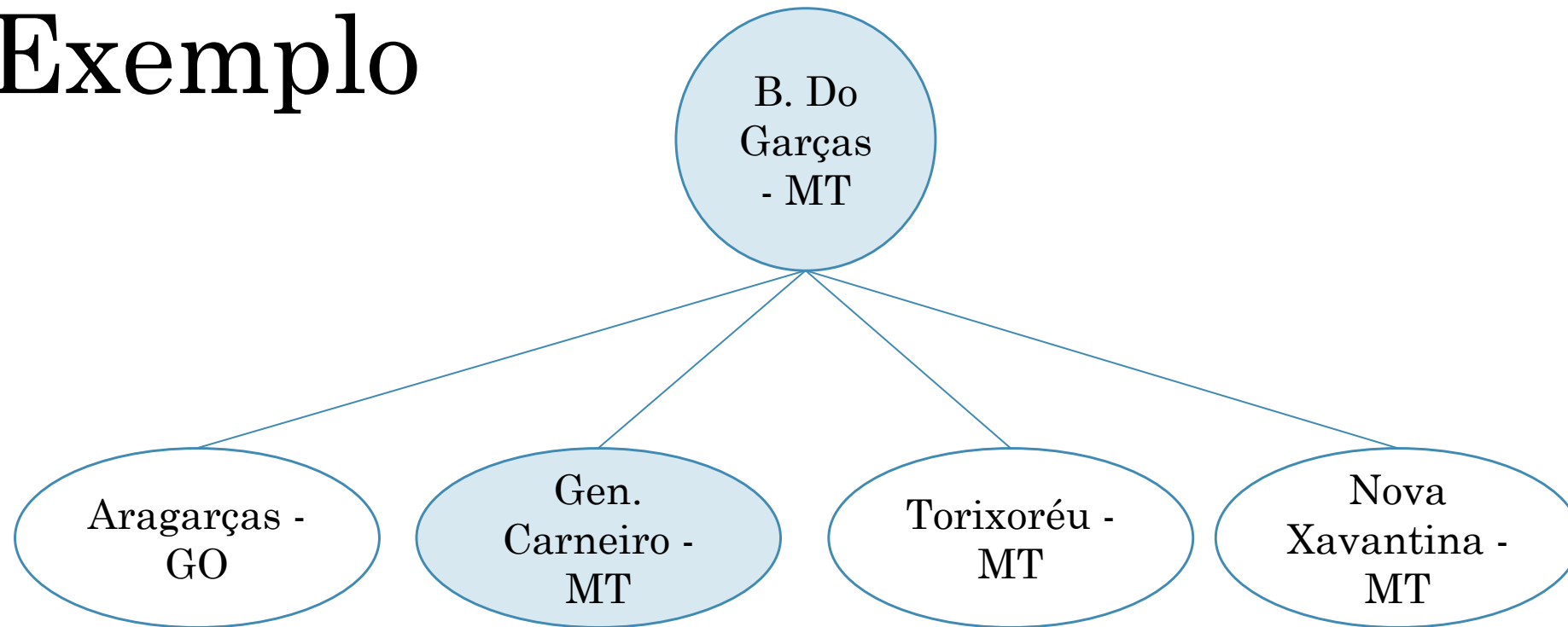
Método de busca: Avaliar uma opção e deixar outras para mais tarde

Exemplo



É objetivo? → não

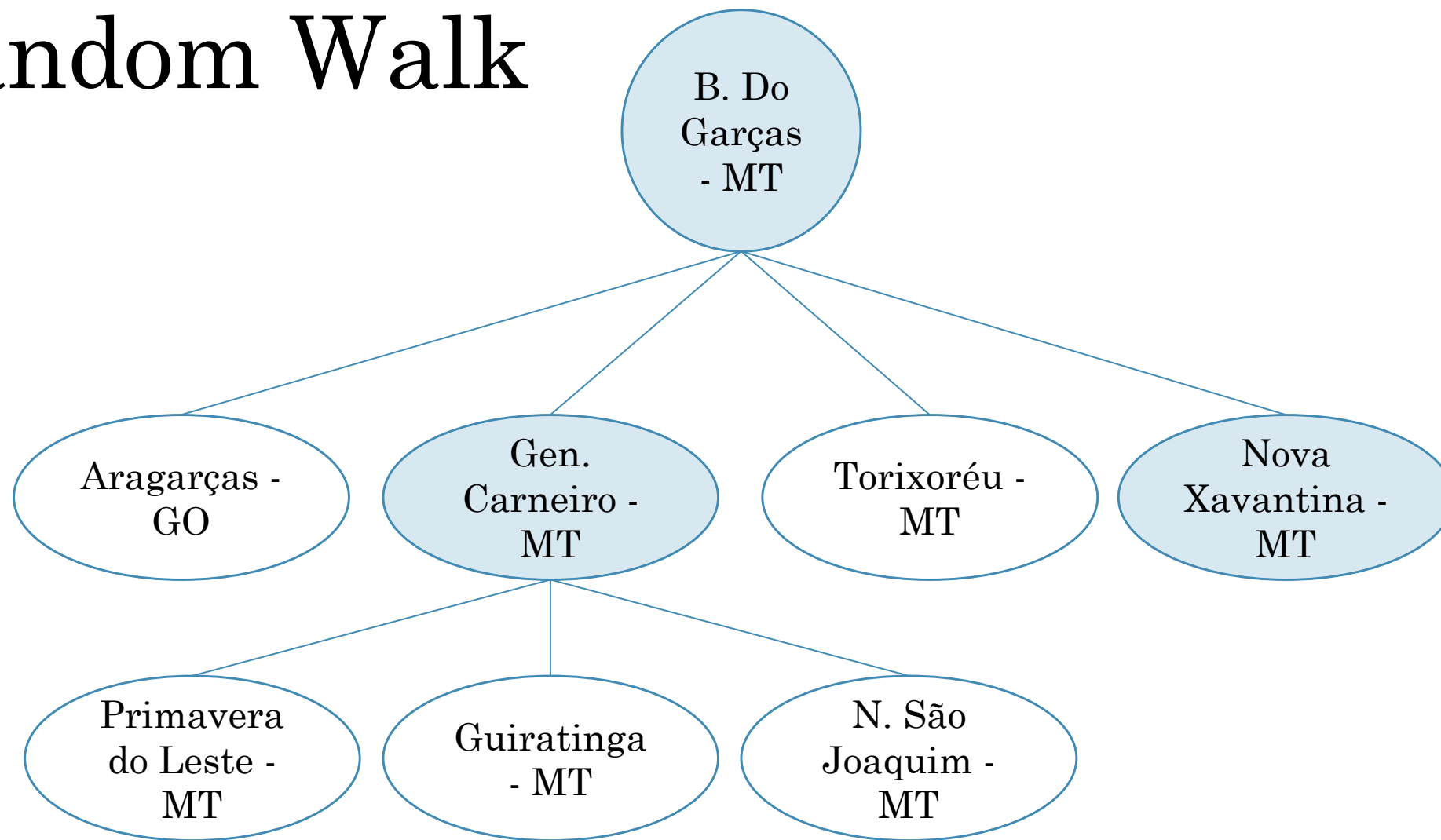
Exemplo



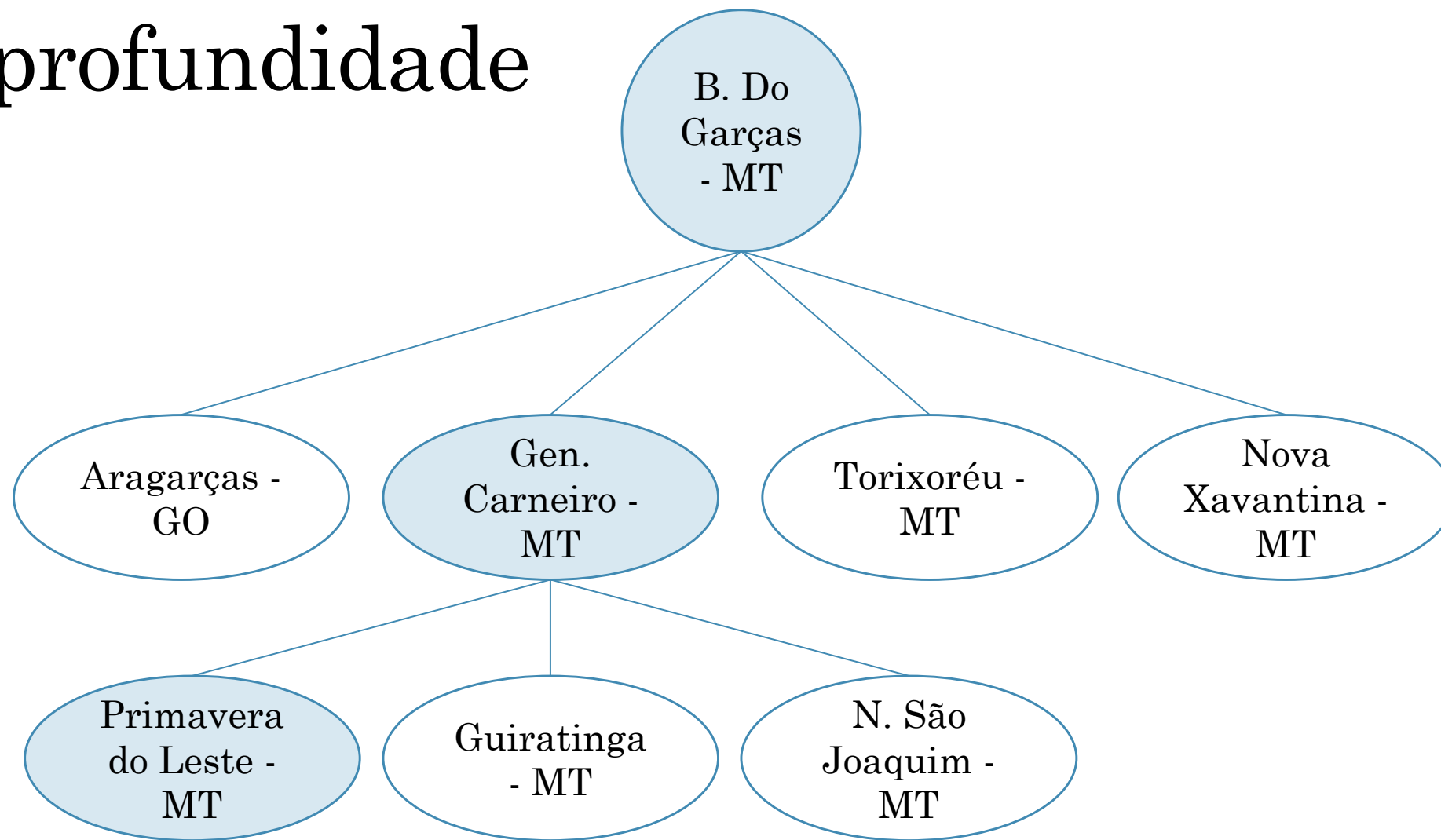
É objetivo? → não

A escolha do próximo estado a ser avaliado é definido pela **estratégia de busca**

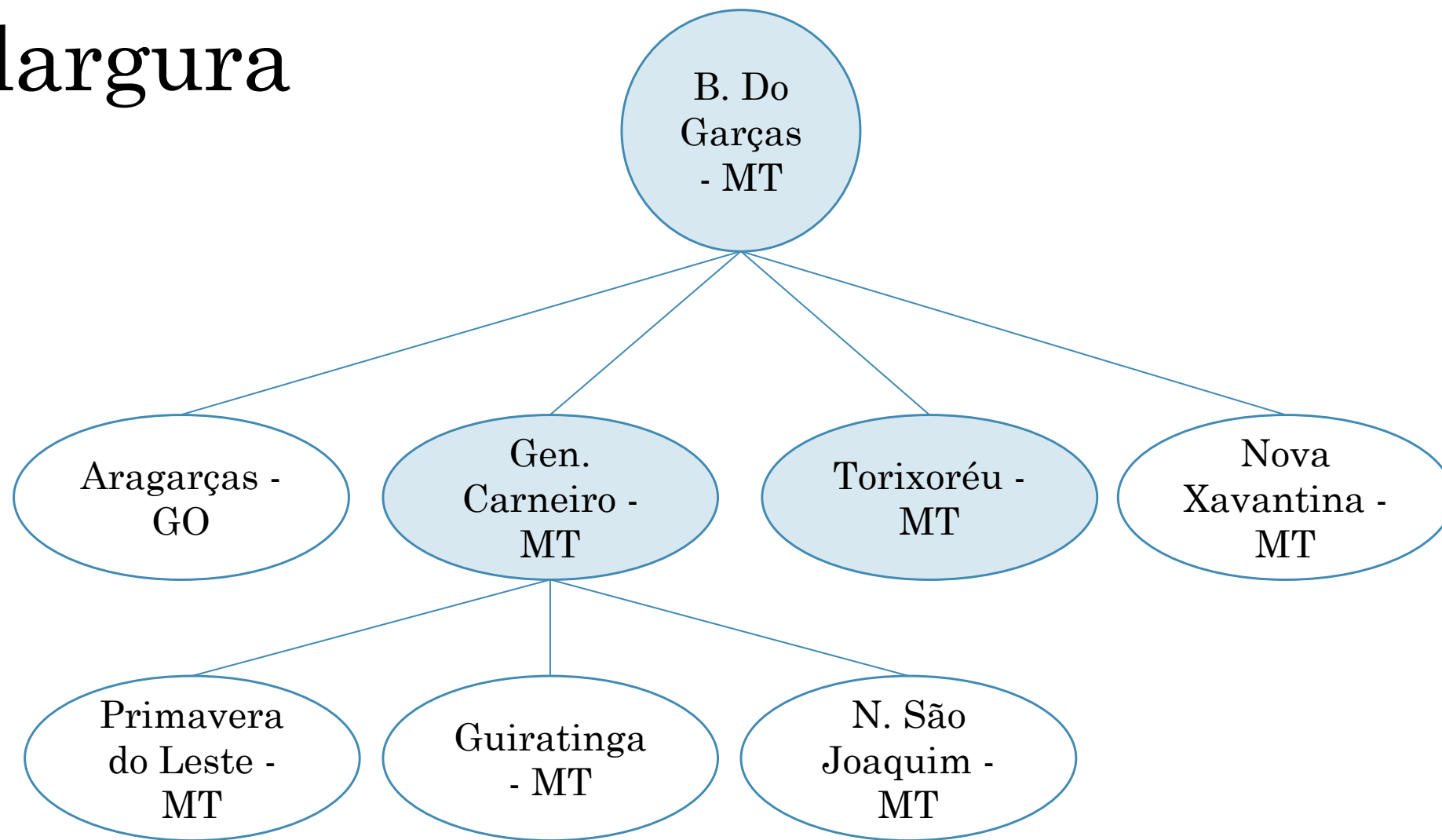
Random Walk



Busca em profundidade



Busca em largura



Métodos de busca

- Uma **modelagem** de problemas de busca em uma **árvore de estados**
- Algoritmos de busca em grafo são usados para encontrar a **solução**

Recapitulando...

- Problemas de busca são compostos por **espaço de estados**, **ações**, estado **inicial** e estado **objetivo**
- Uma **solução** leva do estado inicial ao final por meio de ações
- Algoritmos de **busca em grafos** podem ser usadas para encontrar soluções válidas

Busca sem informação

- Encontram soluções:
 - Gerando sistematicamente novos estados e
 - Comparando-os com o objetivo

- São ineficientes na maioria dos casos

- **Buscas com informação**

- Usam conhecimento específico do problema
 - Podem ser mais eficientes

- Exemplo: `quebra_cabeca_8.py`

Busca com informação

- Busca Heurística
 - Utilizada conhecimento específico do problema e
 - Definição do próprio problema
- Tentativa de expandir os caminhos mais **promissores** primeiro
- Heurística é a função que **estima** a distância ao objetivo (qualidade do novo estado)

Heurística

- Heurística provém do grego antigo εὕρισκω, transl. heurísko, ‘**eu encontro**’, ‘**eu acho**’
- **Heurísticas** são processos cognitivos empregados em decisões não racionais, sendo definidas como estratégias que **ignoram parte da informação** com o objetivo de tornar a escolha mais fácil e rápida
- **Heurística do reconhecimento:** é uma das heurísticas mais simples pois tem por base a recaptção de memórias e o reconhecimento de alternativas. Assim que uma alternativa é reconhecida, a procura por alternativas para a decisão é tomada pela alternativa que se reconhece
 - Vou pedir o sanduíche na lanchonete que já conheço

Busca com informação

- Abordagem geral:
 - Expande nós com base em **função de avaliação** $f(n)$
 - Mede distância até o objetivo, considerando heurística
 - Nó com menor custo é selecionado
 - Vários algoritmos
 - Funções de avaliações diferentes
- Implementação: Gera uma fila de nós de acordo com $f(n)$

Busca gulosa

- Tenta expandir **nó mais próximo à meta**
 - Supõe que levará a uma solução rápida
 - Avalia nós usando apenas a **função heurística**

$$f(n) = h(n)$$

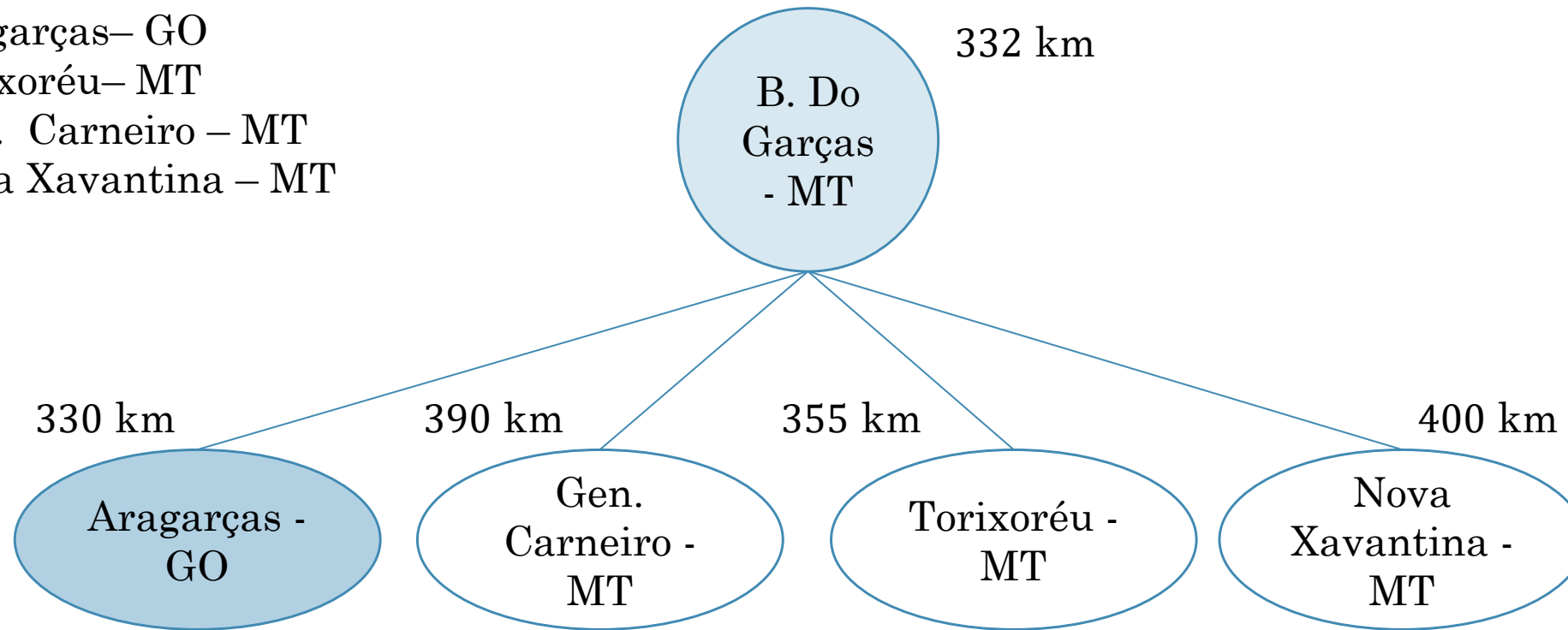


Exemplo

Função heurística $h(n)$: Distância em linha reta até o objetivo

Fila:

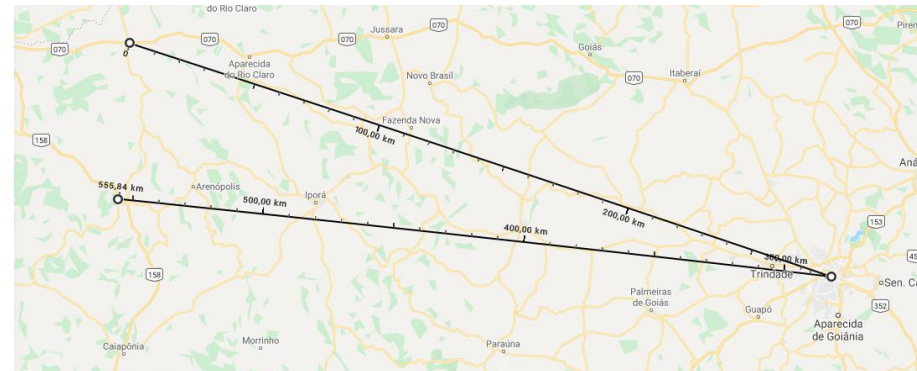
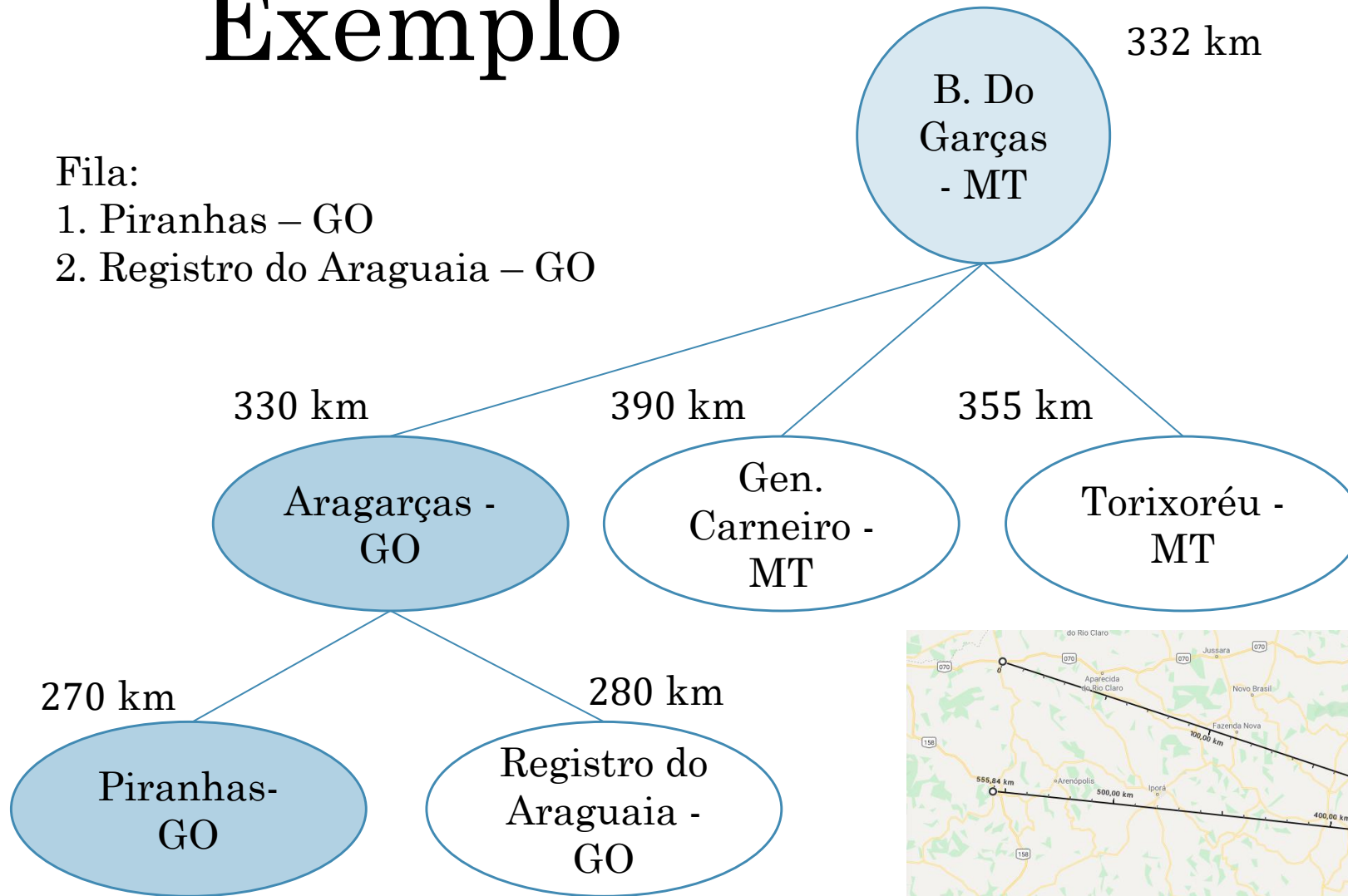
1. Aragarças– GO
2. Torixoréu– MT
3. Gen. Carneiro – MT
4. Nova Xavantina – MT



Exemplo

Fila:

1. Piranhas – GO
2. Registro do Araguaia – GO



Busca gulosa

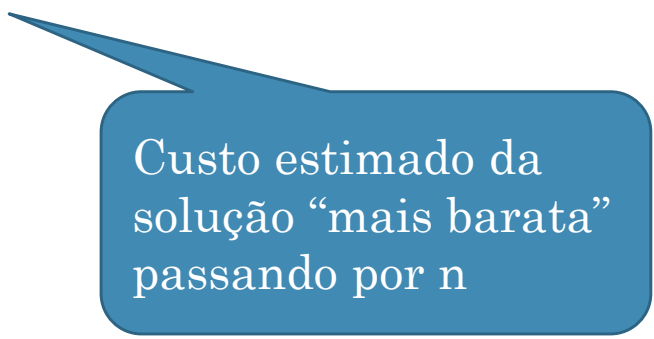
- Como uma **busca em profundidade** orientada por heurística
- Não garante solução **ótima**
 - Avalia apenas o melhor nó seguinte e não toda a solução
- Pode entrar em loop infinito

Busca A*

- Minimiza custo total estimado da solução
- Avalia nós combinando:
 - $g(n)$: custo real do caminho para alcançar cada nó
 - Custo do nó inicial até o nó n
 - $h(n)$: custo estimado do nó atual até o objetivo

$$f(n) = g(n) + h(n)$$

Ideia: Evitar expandir caminhos caros



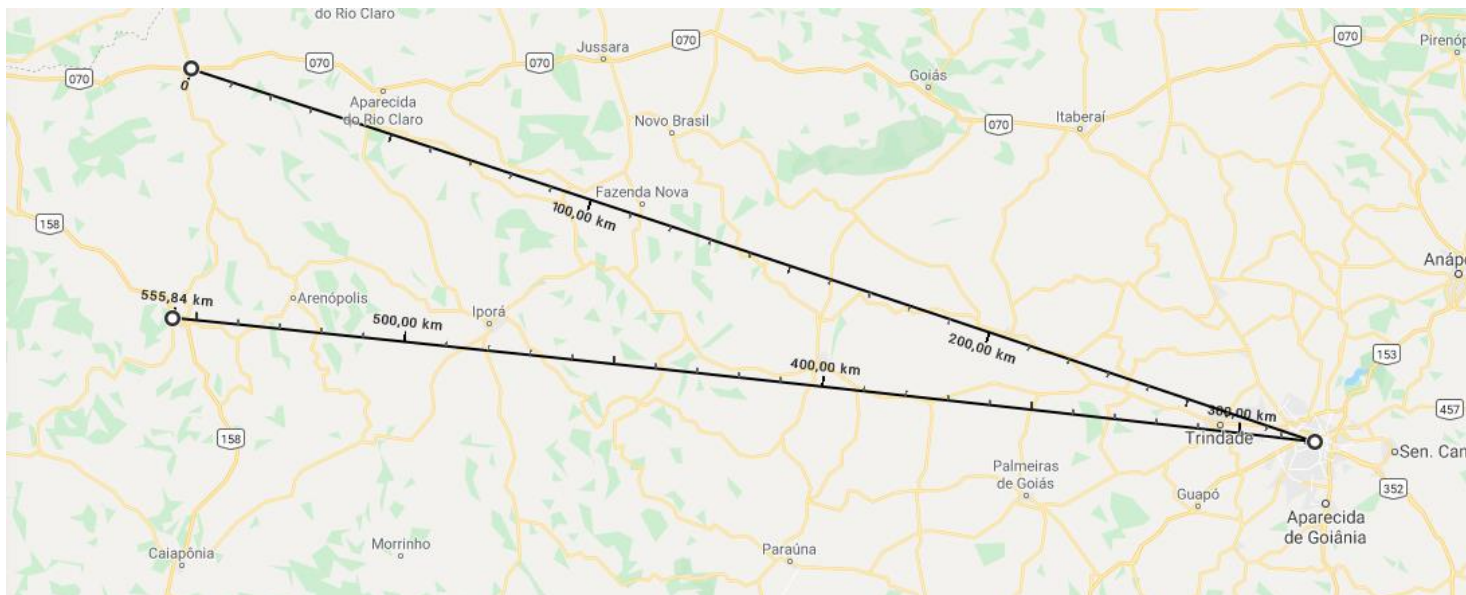
Custo estimado da
solução “mais barata”
passando por n

Busca A^*

- Se a função heurística $f(n)$ satisfaz algumas condições A^* é **completa e ótima**
 - Em busca de árvore, é ótima se $h(n)$ for **heurística admissível**
 - **Heurística admissível:** Nunca superestima o custo de alcançar o objetivo.
 - Supõe que o custo da resolução do problema é menor do que é na realidade
 - Assim, $f(n)$ nunca irá superestimar o custo verdadeiro de uma solução.
- **Completo:** sempre encontra uma solução
- **Ótimo:** sempre encontra a melhor solução

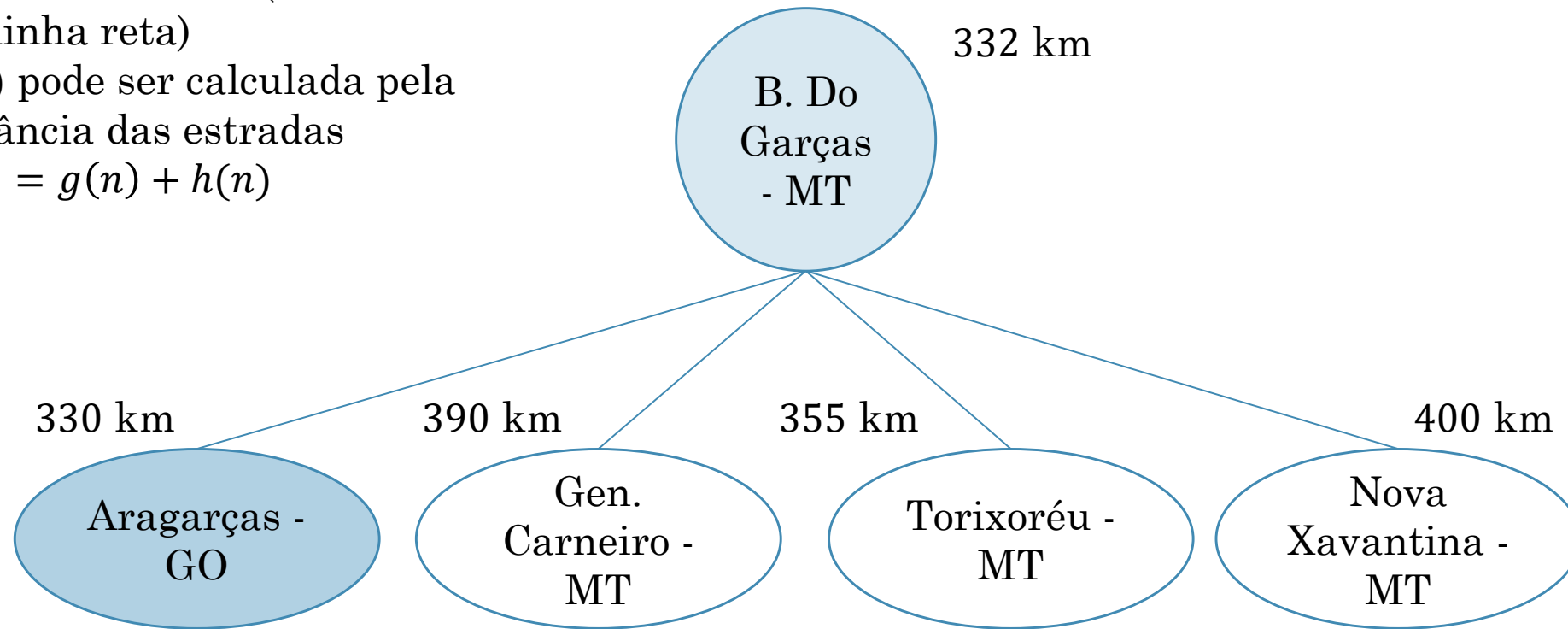
Heurística da linha reta

- $h(n)$ é admissível (distância em linha reta)
- Resultado $h(n)$ sempre vai ser igual ou menor que o custo real
 - Menor estrada entre duas cidades é uma linha reta



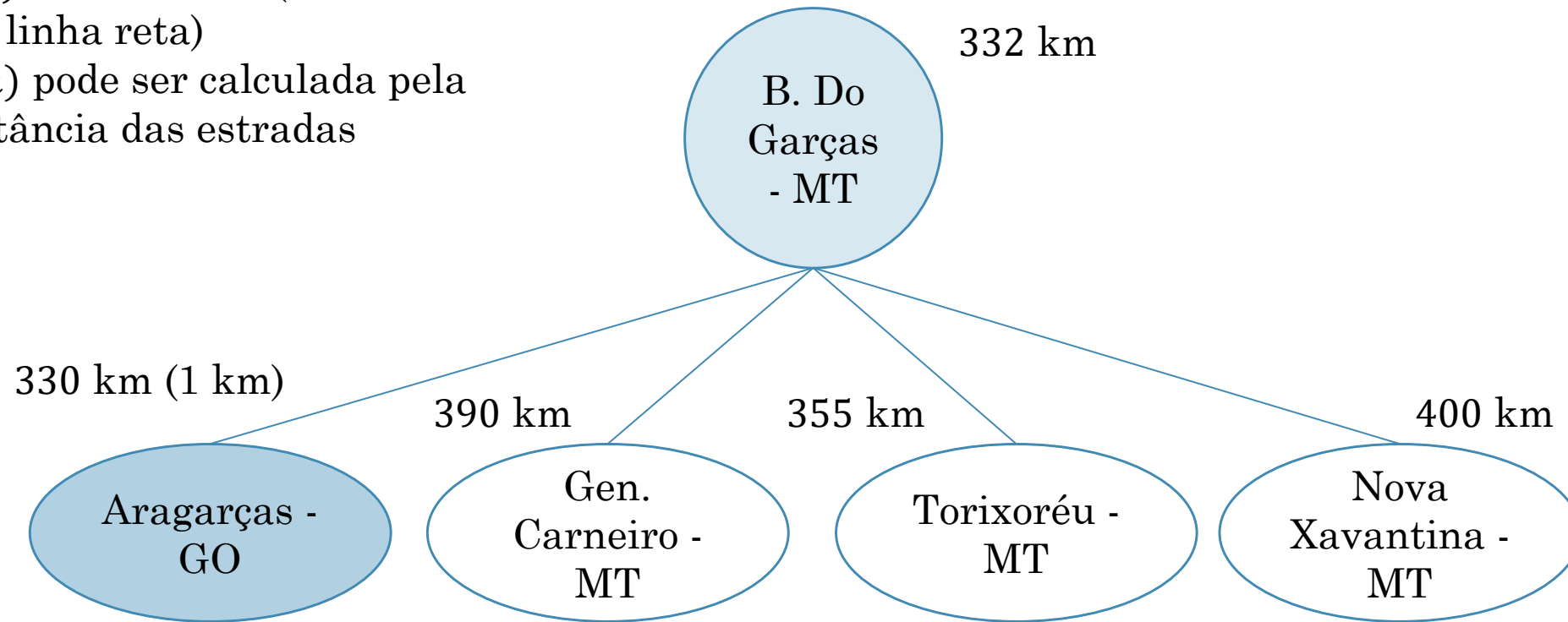
Exemplo

- $h(n)$ é admissível (distância em linha reta)
- $g(n)$ pode ser calculada pela distância das estradas
- $f(n) = g(n) + h(n)$



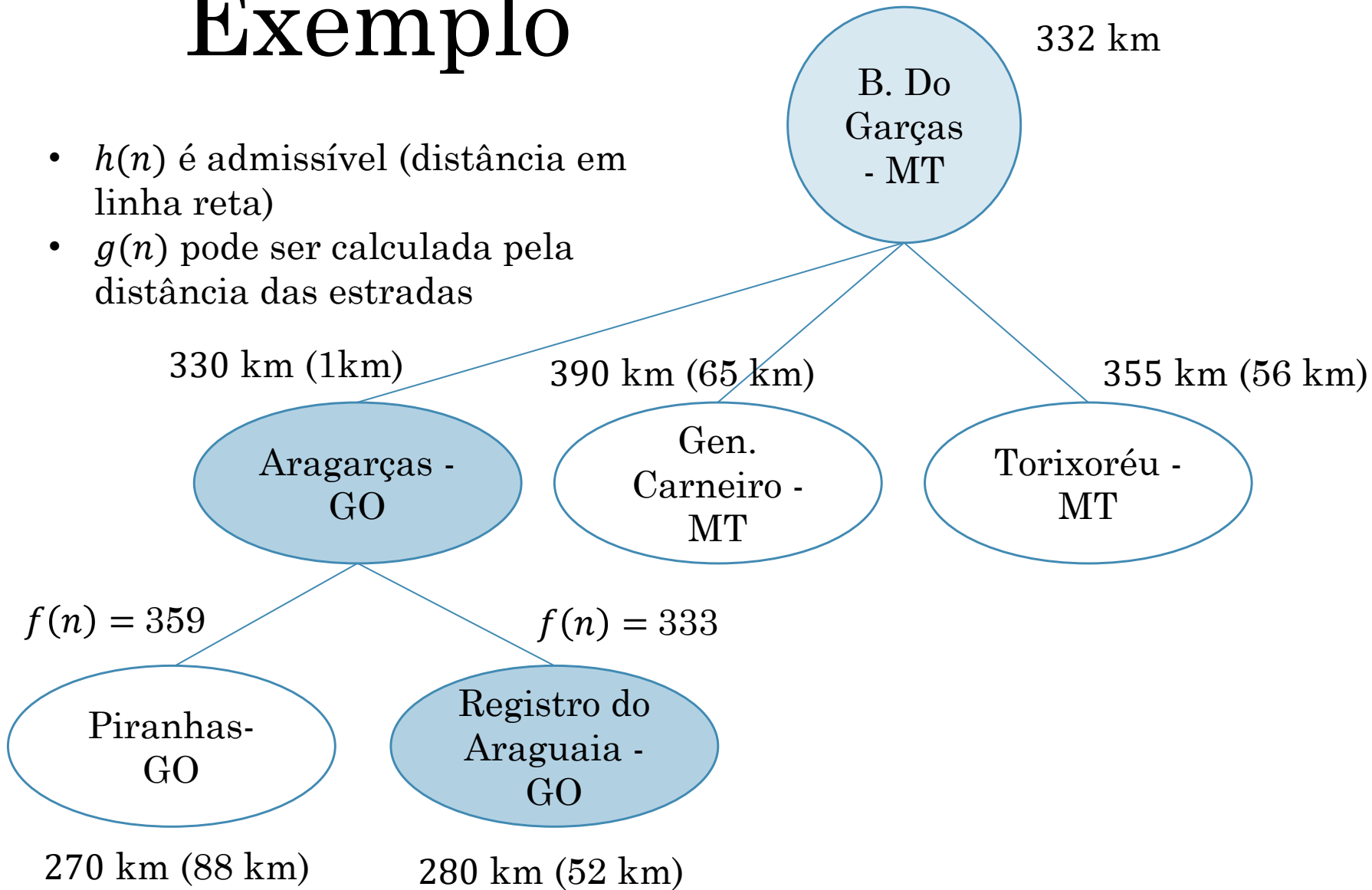
Exemplo

- $h(n)$ é admissível (distância em linha reta)
- $g(n)$ pode ser calculada pela distância das estradas

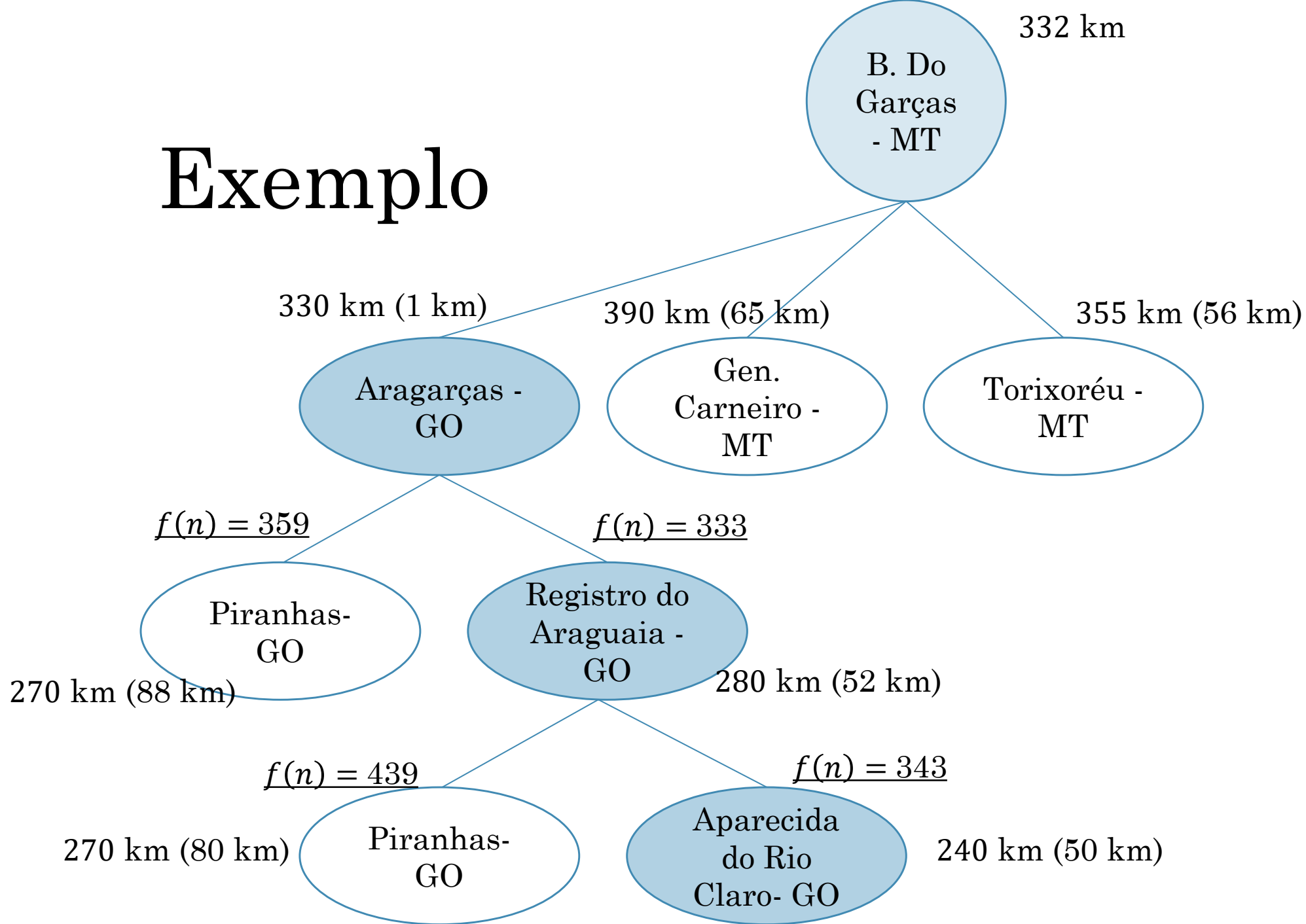


Exemplo

- $h(n)$ é admissível (distância em linha reta)
- $g(n)$ pode ser calculada pela distância das estradas



Exemplo



Busca A^*

- Complexidade de tempo ainda é exponencial na maioria dos casos
- Complexidade de espaço é exponencial
 - Mantém todos os nós na memória
 - Esgota espaço antes de tempo

Busca A* iterativo

- Determinar um limiar máximo de custo para os nós serem salvos
- A cada iteração o limiar sobe

Exemplos

- Quebra-cabeça de 8 peças
 - $h1$ = número de blocos em posições erradas
 - Admissível, pois cada bloco deve ser movido ao menos uma vez
 - $h2$ = distância dos blocos de suas posições objetivo
 - Admissível, ao menos terá que deslocar $h2$ vezes

$h1 = 8$
 $h2 = 18$

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Custo da busca

- Número de nós expandidos para encontrar solução

Nº de blocos	Profundidade	$A^*(h1)$	$A^*(h2)$
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	3644035	227	73

Recapitulando...

- Busca com informação utiliza conhecimento específico do problema
- Heurística é a função que **estima** a distância ao objetivo
- Busca gulosa.
 - Pode entrar em loop
- Busca A*.
 - Completa
 - Ótima
 - Alternativa “inteligente” ao Dijkstra
- Exemplo: `quebra_cabeca_8_A_star.py`