Inatel

Estratégias de Busca

Disciplina: Inteligência Computacional (C210A/B)

Curso: Engenharia de Computação e Software

Profª. Victoria Dala Pegorara Souto

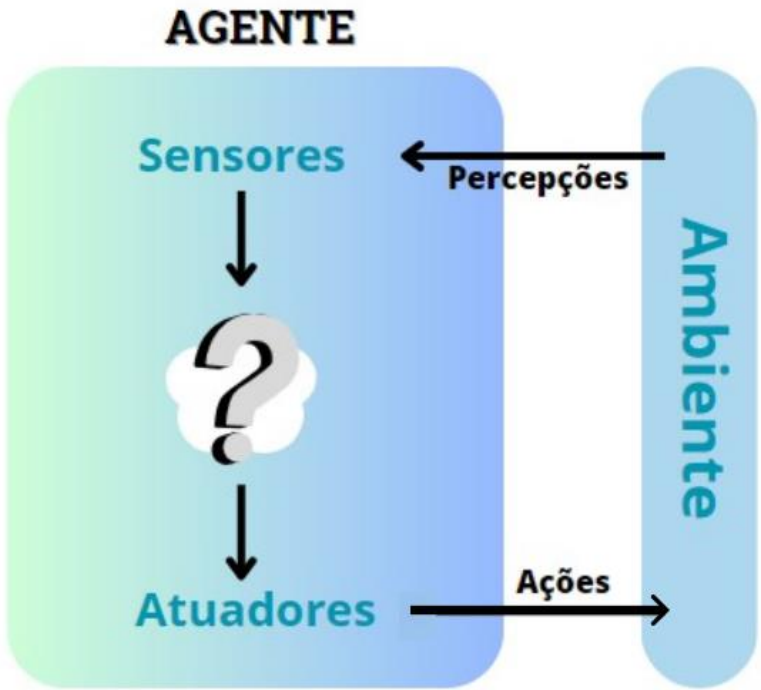
Sumário

- Agentes Inteligentes
- Teoria de Problemas
- Resolução de Problemas
- Estratégias de Busca
  - Busca Cega
  - Busca Gulosa
  - Busca A*

Agentes Inteligentes

Agentes Inteligentes

- **Definição:**
  - Um **Agente** é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de **Sensores** e de agir sobre esse ambiente por intermédio de **Atuadores**.



Exemplos de Agentes:

Um agente humano:

Sensores: olhos, ouvidos, ...

Atuadores: mãos, pernas, boca, ...

Um agente robótico:

Sensores: câmeras, detectores da faixa de infravermelho, ...

Atuadores: motores, braço robótico, ...

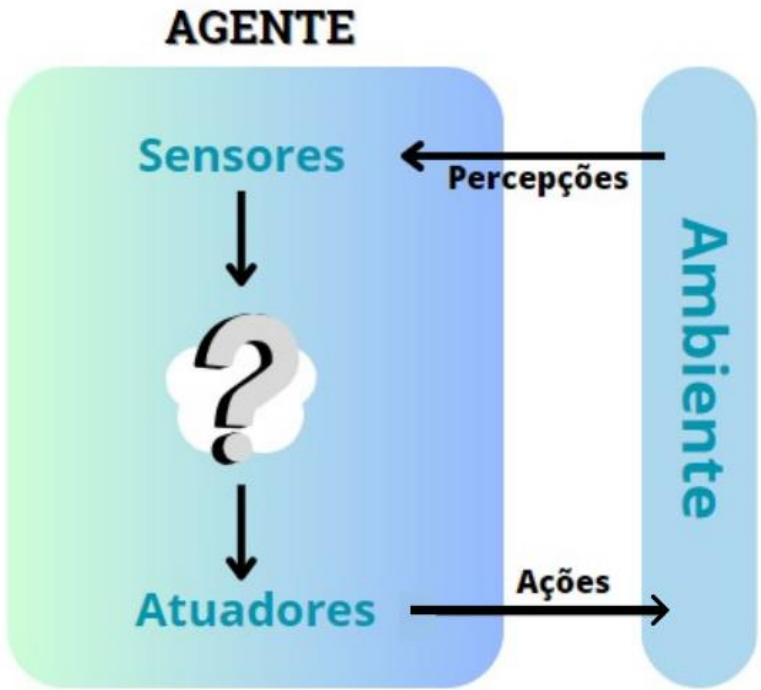
Um agente de software:

Sensores: teclas digitadas, conteúdo de arquivos, pacotes de redes, ...

Atuadores: exibição de algo na tela, gravação de arquivos, envio de pacotes de rede, ...

Agentes Inteligentes

- **Definição:**
  - Um **Agente** é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de **Sensores** e de agir sobre esse ambiente por intermédio de **Atuadores**.



Percepções: Entradas perceptivas do agente em um dado instante.

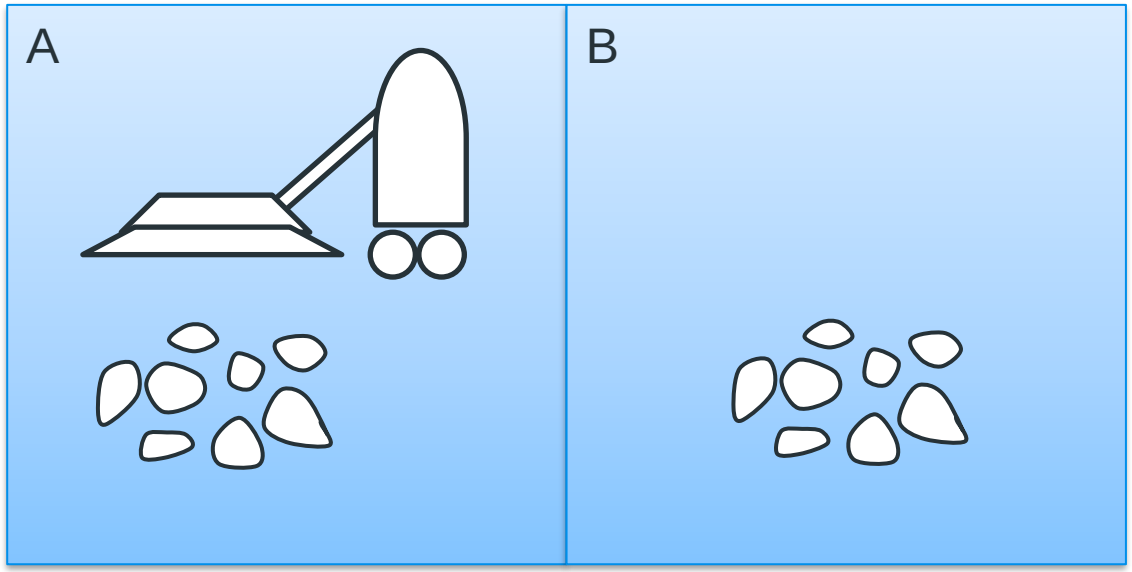
Sequência de Percepções do Agente: História completa de tudo o que o agente já percebeu.

Ações:

A escolha de ação de um agente em qualquer instante dado pode depender da sequência inteira de percepções recebidas até o momento, mas não de percepções não recebidas.

Agentes Inteligentes

- **O mundo do aspirador de pó:**

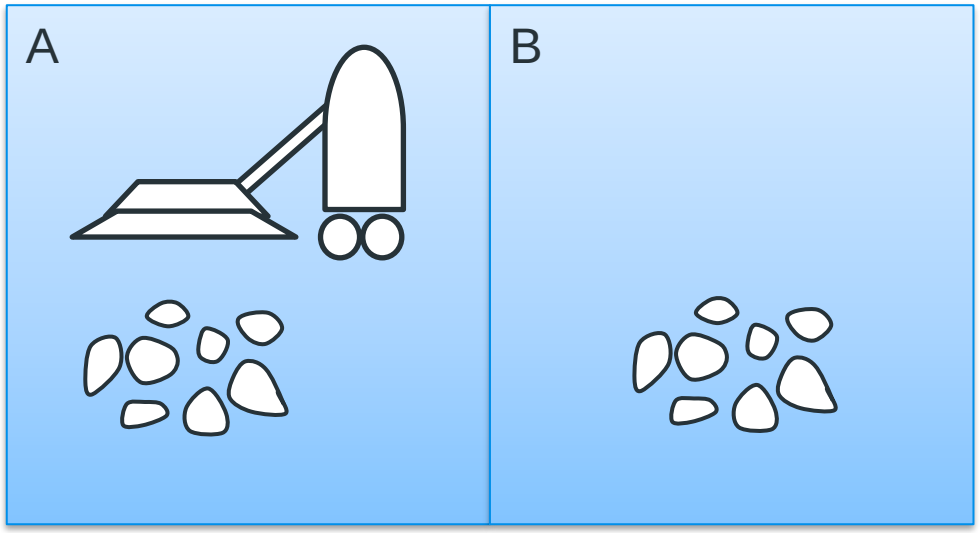


- O comportamento do agente é descrito pela **função do agente** que mapeia qualquer sequência de percepções específica para uma ação.

Ex.: Se o quadrado atual estiver sujo, então aspirar, caso contrário mover-se para o outro quadrado.

Agentes Inteligentes

- **O mundo do aspirador de pó:**



Tabulação da função do agente:

Sequencia de percepções	Ação
[A, Limpo]	Direita
[A, Sujo]	Aspirar
[B, Limpo]	Esquerda
[B, Sujo]	Aspirar
[A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Sujo]	Aspirar
...	...
[A, Limpo], [A, Limpo], [A, Limpo]	Direita
[A, Limpo], [A, Limpo], [A, Sujo]	Aspirar
...	...

Conjunto infinito de funções do agente

Qual a maneira correta de preencher a tabela?

O que torna um agente bom ou ruim, inteligente ou estúpido?

Agentes Inteligentes

Conceito de Racionalidade:

- **Agente Racional** → Aquele que faz tudo certo – em termos conceituais, toda entrada da tabela correspondente à função do agente é preenchida de forma correta.

O que significa fazer tudo correto?

- Quando um agente é colocado em um ambiente, gera uma sequência de ações de acordo com as percepções que recebe. Essa sequência de ações faz com que o ambiente passe por uma sequência de estados. Se a sequência for desejável, o agente teve um desempenho bom.

Noção de desejável definida por uma **Medida de desempenho** que avalia qualquer sequência dada dos estados do ambiente.

Medidas de desempenho devem ser projetadas de acordo com o resultado desejado no ambiente e não de acordo com o comportamento esperado do agente.

Agentes Inteligentes

Conceito de Racionalidade:

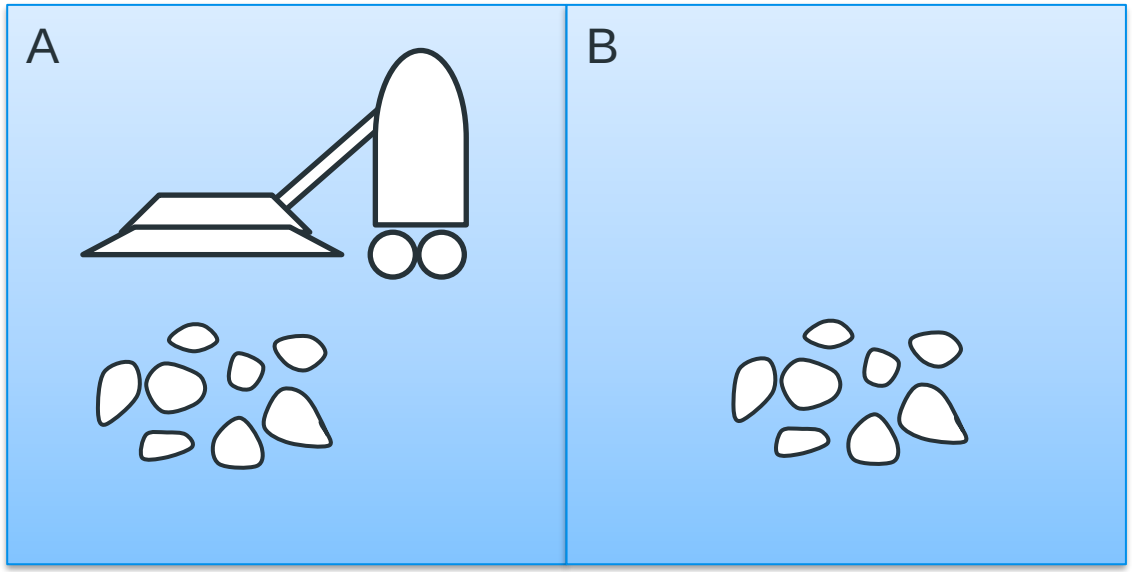
- A definição do que é racional em qualquer instante dado depende de quatro fatores:
  - A medida de desempenho que define o critério de sucesso.
  - O conhecimento prévio que o agente tem do ambiente.
  - As ações que o agente pode executar.
  - A sequência de percepções do agente até o momento.

- Isso conduz a uma definição de um **Agente Racional**:

> *"Para cada sequência de percepção possíve, um agente racional deve selecionar uma ação que venha maximizar sua medida de desempenho, dada a evidência fornecida pela sequência de percepções e por qualquer conhecimento interno do agente."*

Agentes Inteligentes

- **O mundo do aspirador de pó:**



- Se o quadrado atual estiver sujo, então aspirar, caso contrário mover-se para o outro quadrado.

O Agente aspirador é racional?

Agentes Inteligentes

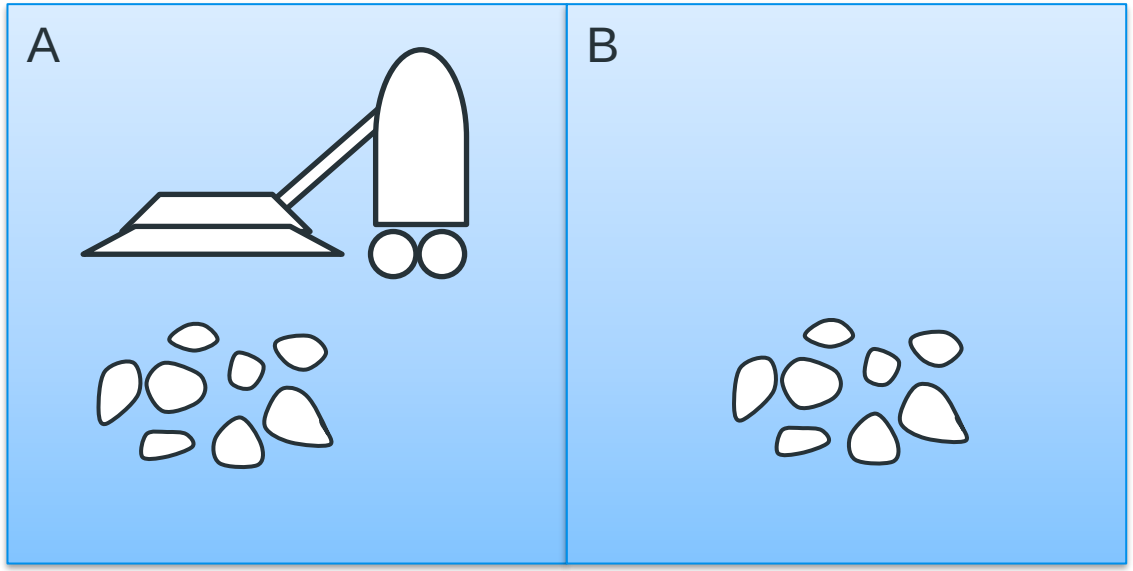
Conceito de Racionalidade:

- Um agente racional deve ser **autônomo** – ele deve aprender o que puder para compensar um conhecimento prévio parcial ou incorreto.
- Raramente temos autônomia completa desde o início: quando o agente tem pouca ou nenhuma experiência.
- O projetista deve fornecer ao agente de IA algum conhecimento inicial, bem como habilidade para aprender.
- Depois de adquirir experiência suficiente sobre seu ambiente, o comportamento de um agente racional pode se tornar efetivamente independente do seu conhecimento anterior.
- **→ Um único agente racional pode ter sucesso em diferentes ambientes!!!**

Agentes Inteligentes

Ambiente de Tarefa:

- "Problemas" para os quais os agentes racionais são as "soluções".



O mundo do aspirador de pó.

Definimos:

- Medida de desempenho;
- Ambiente;
- Atuadores;
- Sensores.

Ambiente de Tarefa

PEAS *(Performance, Environment, Actuators, Sensors)*

- Ao projetar um agente, a **primeira etapa** deve ser sempre especificar o ambiente de tarefa de forma tão completa quanto possível.

Agentes Inteligentes

Ambiente de Tarefa (PEAS):

Exemplo: Motorista de táxi automatizado.

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Motorista de taxi	Viagem segura, rápida, dentro da lei, confortável, maximizar lucro	Estradas, outros tipos de tráfego, pedestres, clientes	Direção, Acelerador, freio, sinal, buzina, visor	Câmeras, sonar, velocímetro, GPS, odômetro, acelerômetro, sensores do motor, teclado

- **Que Medida de Desempenho gostaríamos que o motorista automatizado tivesse como objetivo?**

  - Chegar ao destino correto;
  - Minimizar o consumo de combustível e desgaste;
  - Minimizar o tempo e/ou o custo de viagem;
  - Minimizar as violações às leis de trânsito e as perturbações a outros motoristas;
  - Maximizar a segurança e o conforto dos passageiros;
  - Maximizar os lucros;
  - ...

Podem ser conflitantes!
Fazer uma escolha!!

Agentes Inteligentes

Ambiente de Tarefa (PEAS):

Exemplo: Motorista de táxi automatizado.

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Motorista de taxi	Viagem segura, rápida, dentro da lei, confortável, maximizar lucro	Estradas, outros tipos de tráfego, pedestres, clientes	Direção, Acelerador, freio, sinal, buzina, visor	Câmeras, sonar, velocímetro, GPS, odômetro, acelerômetro, sensores do motor, teclado

- **Qual é o ambiente de direção que o táxi enfrentará?**

  - Diferentes tipos de estradas;
  - Estradas com outros tipos de tráfego: pedestres, animais, policiamento, buracos ...
  - Motorista deverá interagir com o passageiro;
  - Lado da direção (direita e/ou esquerda);
  - ...

Quanto mais restrito o ambiente mais fácil o projeto!

Agentes Inteligentes

Ambiente de Tarefa (PEAS):

Exemplo: Motorista de táxi automatizado.

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Motorista de taxi	Viagem segura, rápida, dentro da lei, confortável, maximizar lucro	Estradas, outros tipos de tráfego, pedestres, clientes	Direção, Acelerador, freio, sinal, buzina, visor	Câmeras, sonar, velocímetro, GPS, odômetro, acelerômetro, sensores do motor, teclado

- **Qual os atuadores para um táxi automatizado?**

  - Controle do motor através do acelerador;
  - Controle sobre a direção e frenagem;
  - Tela de exibição;
  - Sintetizador de voz para se comunicar com os passageiros.
  - ...

Agentes Inteligentes

Ambiente de Tarefa (PEAS):

Exemplo: Motorista de táxi automatizado.

Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Motorista de taxi	Viagem segura, rápida, dentro da lei, confortável, maximizar lucro	Estradas, outros tipos de tráfego, pedestres, clientes	Direção, Acelerador, freio, sinal, buzina, visor	Câmeras, sonar, velocímetro, GPS, odômetro, acelerômetro, sensores do motor, teclado

- **Qual os sensores para um táxi automatizado?**

  - Câmeras que podem possui infravermelho ou sensor sonar para detectar distâncias de outros carros e obstáculos;
  - Velocímetro;
  - Conjunto de sensores do motor, combustível e sistema elétrico (tradicional);
  - GPS;
  - Teclado e/ou microfone para o passageiro solicitar um destino.
  - ...

Agentes Inteligentes

Ambiente de Tarefa (PEAS):

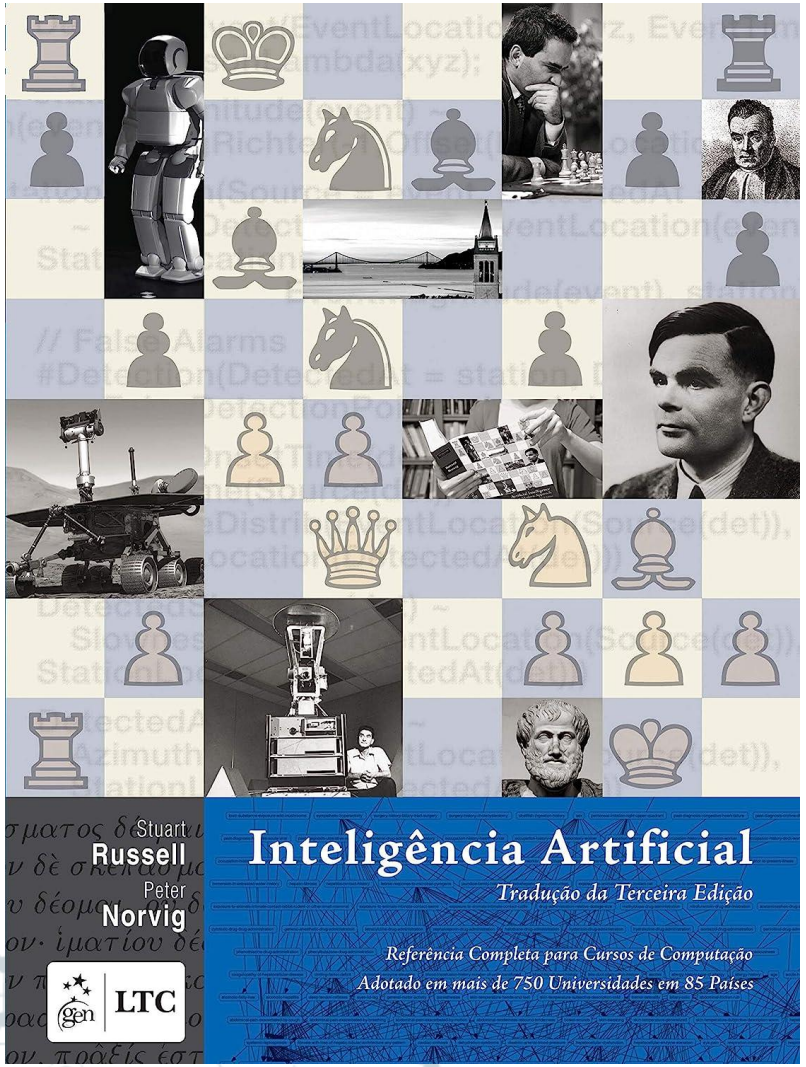
Tipo de agente	Medida de desempenho	Ambiente	Atuadores	Sensores
Motorista de taxi	Viagem segura, rápida, dentro da lei, confortável, maximizar lucro	Estradas, outros tipos de tráfego, pedestres, clientes	Direção, Acelerador, freio, sinal, buzina, visor	Câmeras, sonar, velocímetro, GPS, odômetro, acelerômetro, sensores do motor, teclado
Sistema de diagnóstico médico	Paciente saudável, minimizar custos , processos judiciais	Paciente, hospital, equipe	Exibir perguntas, testes, diagnósticos, tratamento, indicações	Entrada pelo teclado para sintomas, descobertas, respostas do paciente
Análise de imagens de satélite	Definição correta da categoria da imagem	Links de transmissão de satélite em órbita	Exibir a categorização da cena	Arrays de pixels em cores
Robô de seleção de peças	Porcentagem de peças em bandejas corretas	Correia transportadora com peças; bandejas	Braço e mão articulados	Câmera, sensores angulares articulados
Controlador de refinaria	Maximizar pureza, rendimento, segurança	Refinaria, operadores	Válvulas, bombas, aquecedores, mostradores	Sensores de temperatura, pressão, produtos químicos
Instrutor de inglês interativo	Maximizar nota de aluno em teste	Conjunto de alunos, testes de agência	Exibir exercícios, sugestões, correções	Entrada pelo teclado

Agentes Inteligentes

- **Estrutura de Agentes:**
  - Até agora descrevemos o comportamento do agente → A ação executada após qualquer sequência de percepções específica.
  - **Funcionamento interno do agente →** Programa do agente que implementa a função do agente – que mapeia percepções em ações.
  - Supomos que esse programa será executado em agum tipo de dispositivo de computação com sensores e atuadores físicos – chamamos esse conjunto de arquitetura.
  - **O trabalho da IA é projetar o programa do agente.**

Agente = Arquitetura + Programa

Referências



- RUSSELL, Stuart; NORVIG, Peter (Peter Norvig); SOUZA, Vandenberg Dantas De, Inteligência artificial. Rio de Janeiro, RJ: Editora Campus, 2004 - 2013, ISBN 978-85-352-1177-1 / 978-85-352-3701-6.

→ Ler Capítulo 2

Teoria de Problemas

Teoria de Problemas

- A IA se ocupa da resolução de problemas, para tal é necessário o conhecimento sobre o problema e sobre as técnicas para manipular esse conhecimento e obter a solução.

- Resolver um problema é diferente de ter um método para resolvê-lo.

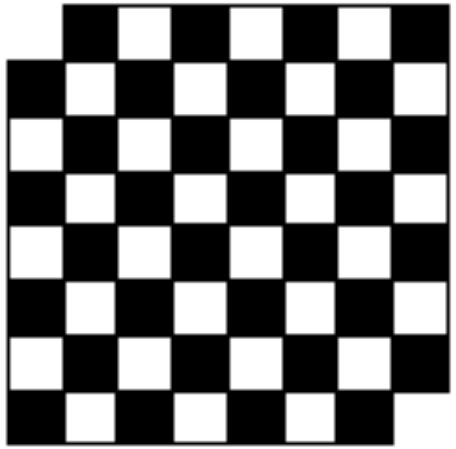
- Antes de tentar buscar a solução de um problema, deve-se responder as seguintes perguntas:
  - Quais são os dados?
  - Quais são as soluções possíveis?
  - O que caracteriza uma solução satisfatória?

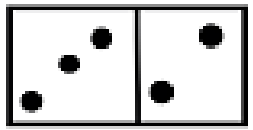
O que é um Problema?

Exemplos de Problemas

Tabuleiro de Xadrez Mutilado

- Nesse problema, temos um tabuleiro de jogo de xadrez onde dois quadrados em cantos opostos foram cortados, de modo que restam apenas 62 quadrados.





- Agora, pegamos 31 dominós feitos de modo que cada dominó cubra exatamente dois quadrados.
- *É possível dispor os 31 dominós de modo que eles cubram todos os 62 quadrados do tabuleiro?*

Teoria de Problemas

- **Definição:** Um problema é um objeto matemático $P = \{D, R, q\}$.
  - D – Conjunto não vazio de **dados**;
  - R – Conjunto não vazio com os possíveis **resultados.**

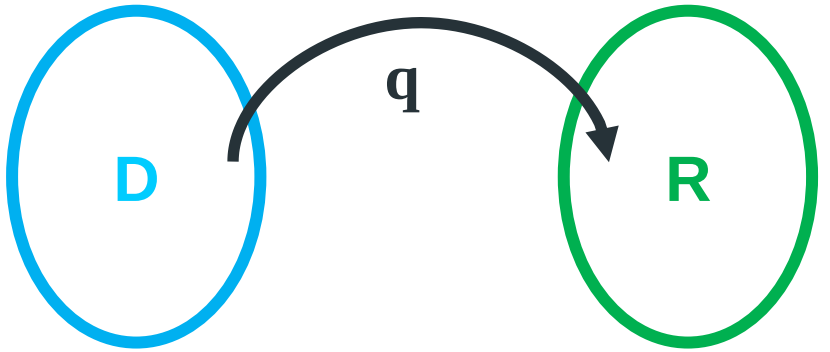
$$q \subset D \times R$$

✓ Condição que caracteriza uma **solução satisfatória**, associando cada elemento do conjunto de dados a um elemento do conjunto de resultados.

Exemplos de Problemas

- **Diagnóstico Médico**
  - O conjunto de dados disponíveis $d \in D$ (*observação dos sintomas, exames, etc*);
  - R é o conjunto de doenças possíveis;
  - **Solução Satisfatória:** encontrar o par (d, r) onde $r \in R$ é o diagnóstico correto.

- A definição de um problema permite testar se um certo elemento é ou não solução, mas não guia na busca deste elemento.



Teoria de Problemas

- **Estratégias Básicas para Resolver Problemas:** As estratégias constituem os modos básicos de raciocínio para resolver problemas.

1. **Pela definição do problema** - o qual se apresenta como uma função, estes modos de raciocínio devem se adaptar ao modo que a função foi definida.

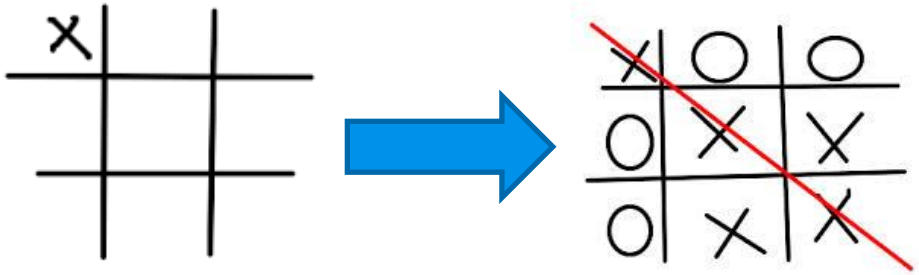
2. **Por enumeração exaustiva -** o conhecimento necessário para resolver o problema está na enumeração (busca exaustiva ou força bruta).

3. **Declarativamente -** leva frequentemente a problemas de busca. "Utilizar um método de busca em que, por passos sucessivos se aproxima da solução, usando algumas vezes técnicas sem grande justificativa teórica". **ABORDAGEM DA IA SIMBÓLICA!**

4. **Por exemplos -** Se o problema foi definido por exemplos, deverá se usar um método para aproximar a função. **ABORDAGEM DA IA CONEXIONISTA!**

Exemplos de Problemas

- **Jogo da Velha**



- **Missionários e Canibais**

- **As Torres de Hanóis**

Resolução de Problemas

Dedica-se ao estudo e elaboração de algoritmos, capazes de resolver, por exemplo, problemas considerados intratáveis do ponto de vista da computação convencional.

Resolução de Problemas

- Um dos **objetivos da IA** é resolver problemas que o homem não sabe resolver facilmente ou num tempo razoável, desde que sejam completamente formalizados.

Teoria de Problemas

- Um **problema** pode ser definido formalmente por **cinco** componentes:
  - **Estado Incial** em que o agente começa;
  - **Ações:** Uma descrição das ações possíveis que estão disponíveis para o agente;
    - Formulação mais comum: uso de uma **função sucessor**.
    - **Estado inicial e função sucessor**: definem o espaço de estados do problema.
    - **Caminho** no espaço de estados – sequência de estados conectados por uma sequência de ações.

Teoria de Problemas

- Um **problema** pode ser definido formalmente por **cinco** componentes:
  - **Modelo de transição:** Descrição do que cada ação faz (devolve o estado que resulta de executar uma ação *a* em estado *s*)

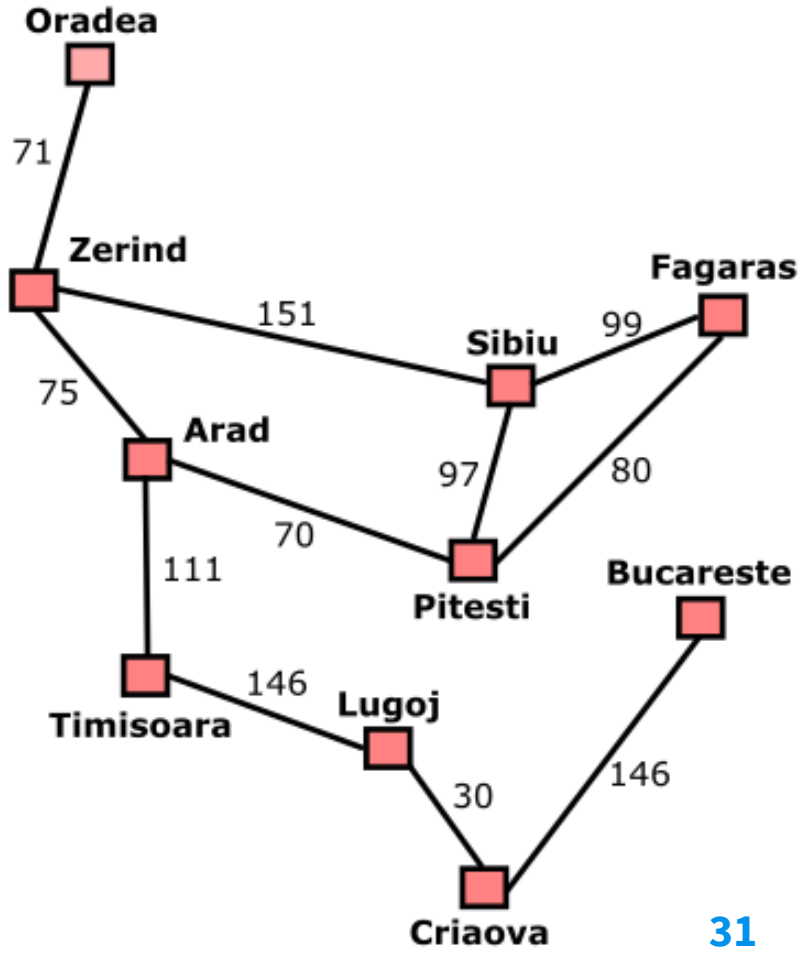
Estado Inicial + Ações + Modelo de Transição = Espaço de Estados do problema

Espaço de Estados → Conjunto de todos os estados acessíveis a partir do estado inicial, por qual sequência de ações – Forma um **grafo** em que os nós são estados e os arcos entre os nós são ações.

Teoria de Problemas

- Um **problema** pode ser definido formalmente por **cinco** componentes:
  - **Modelo de transição:** Descrição do que cada ação faz (devolve o estado que resulta de executar uma ação *a* em estado *s*)

Espaço de Estados → Conjunto de todos os estados acessíveis a partir do estado inicial, por qual sequência de ações – Forma um **grafo** em que os nós são estados e os arcos entre os nós são ações.



Teoria de Problemas

- Um **problema** pode ser definido formalmente por **cinco** componentes:
  - **Teste de objetivo** que determina se um estado é um estado objetivo;
  - **Custo de Caminho** função que atribui um custo numérico a cada caminho.

Teoria de Problemas

- Um **problema** pode ser definido formalmente por **cinco** componentes:
  - **Estado Incial** em que o agente começa;
  - **Ações:** Uma descrição das ações possíveis que estão disponíveis para o agente;
  - **Modelo de transição:** Descrição do que cada ação faz (devolve o estado que resulta de executar uma ação *a* em estado *s*) .

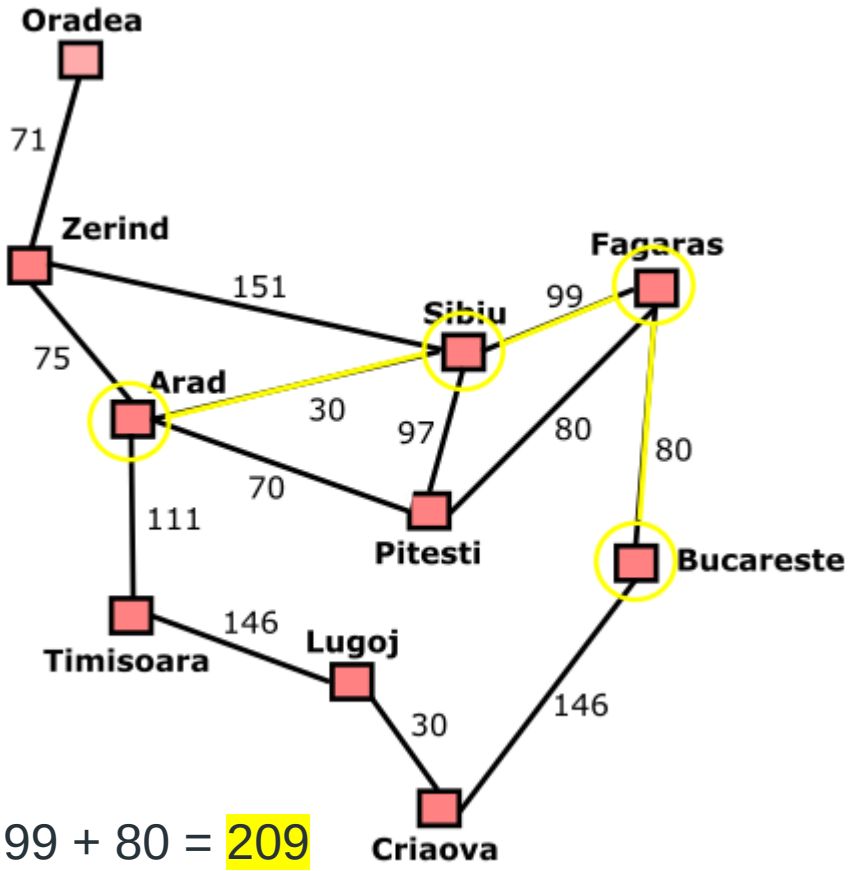
    Espaço de Estados → Conjunto de todos os estados acessíveis a partir do estado inicial, por qual sequência de ações – Forma um **grafo** em que os nós são estados e os arcos entre os nós são ações.

  - **Teste de objetivo** que determina se um estado é um estado objetivo;
  - **Custo de Caminho** função que atribui um custo numérico a cada caminho.

Teoria de Problemas

- **Exemplo:** Problema de roteamento saindo de Arad com o objetivo de chegar em Bucareste.

1. **Espaço de estados:** Mapa da Romênia
2. **Estado inicial:** Arad
3. **Estado Objetivo:** Bucharest
4. **Ações Possíveis:** Ir de uma cidade para outra
5. **Custo:** Distância entre as cidades



Custo: 30 + 99 + 80 = 209

Exemplos de Problemas

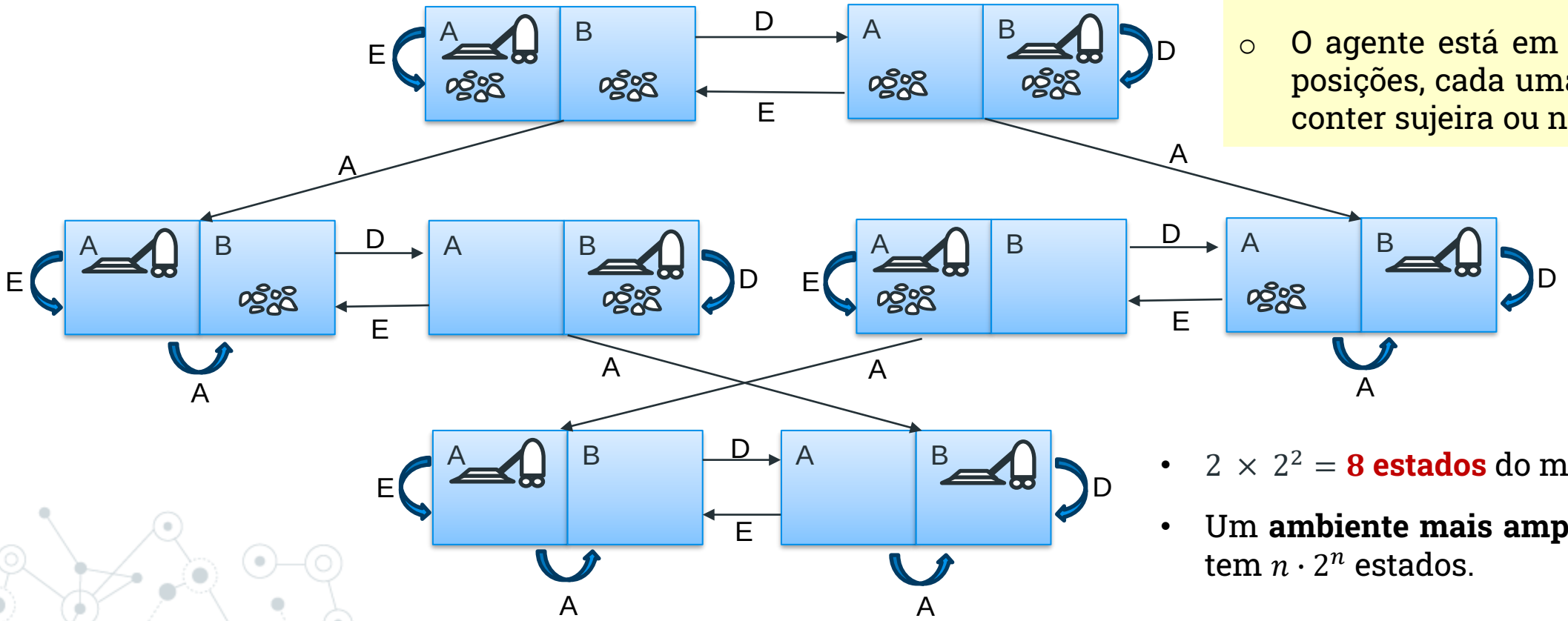
- **Problemas de Mundo Simplificado:**
  - Mundo do Aspirador de Pó
  - Quebra-cabeça de 8 peças
  - Quebra-cabeça das 8 rainhas

- **Problemas do mundo real:**
  - Problema de Exames médicos
  - Problema de Diagnósticos médicos
  - Problema de Roteamento
  - Problema de Layout de placas
  - Problema de Navegação de Robôs
  - Problema de Pesquisas na Internet

Exemplo de Problemas de Mundo Simplificado

Exemplos de Problemas

- **Mundo do Aspirador de Pó**



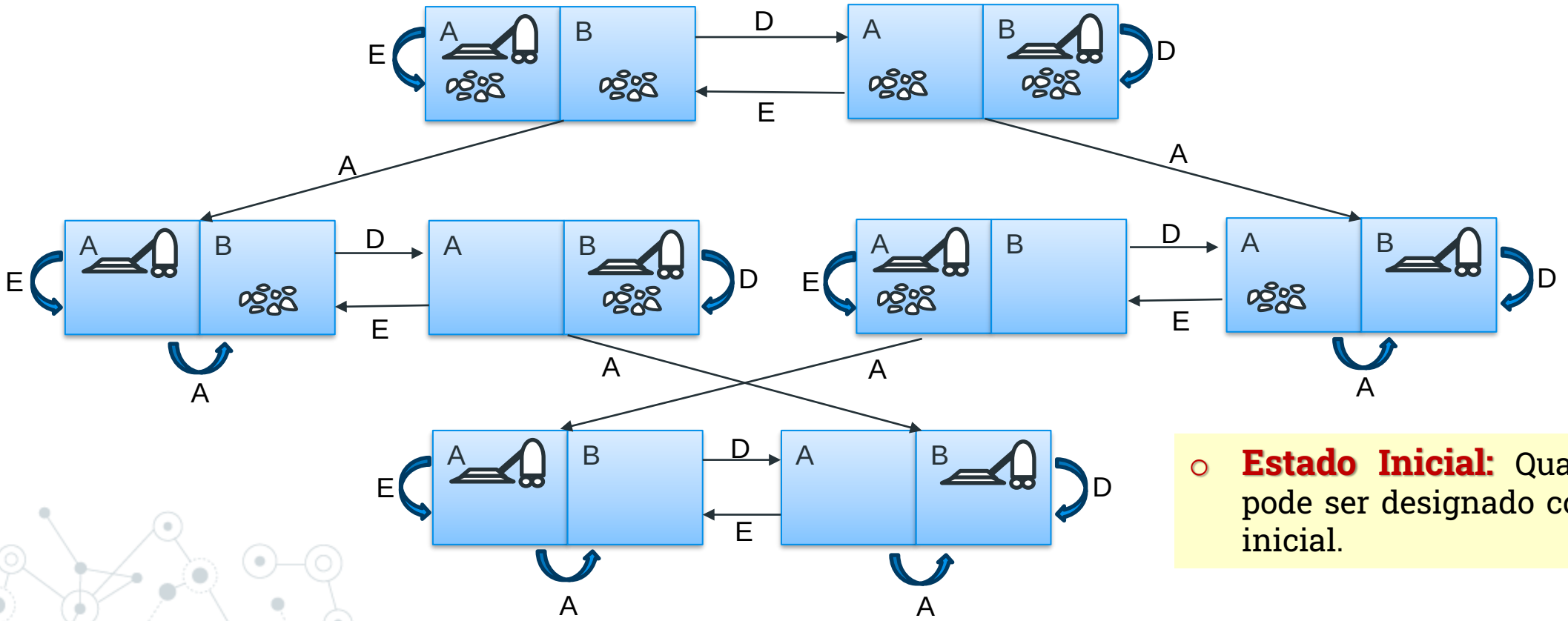
- **Estados:** O estado é determinado tanto pela **posição** do agente como da **sujeira**.
- O agente está em uma entre duas posições, cada uma das quais pode conter sujeira ou não.

- $2 \times 2^2 = 8$ **estados** do mundo possíveis.
- Um **ambiente mais amplo** com n posições tem $n \cdot 2^n$ estados.

Espaço de Estados → 8 Estados

Exemplos de Problemas

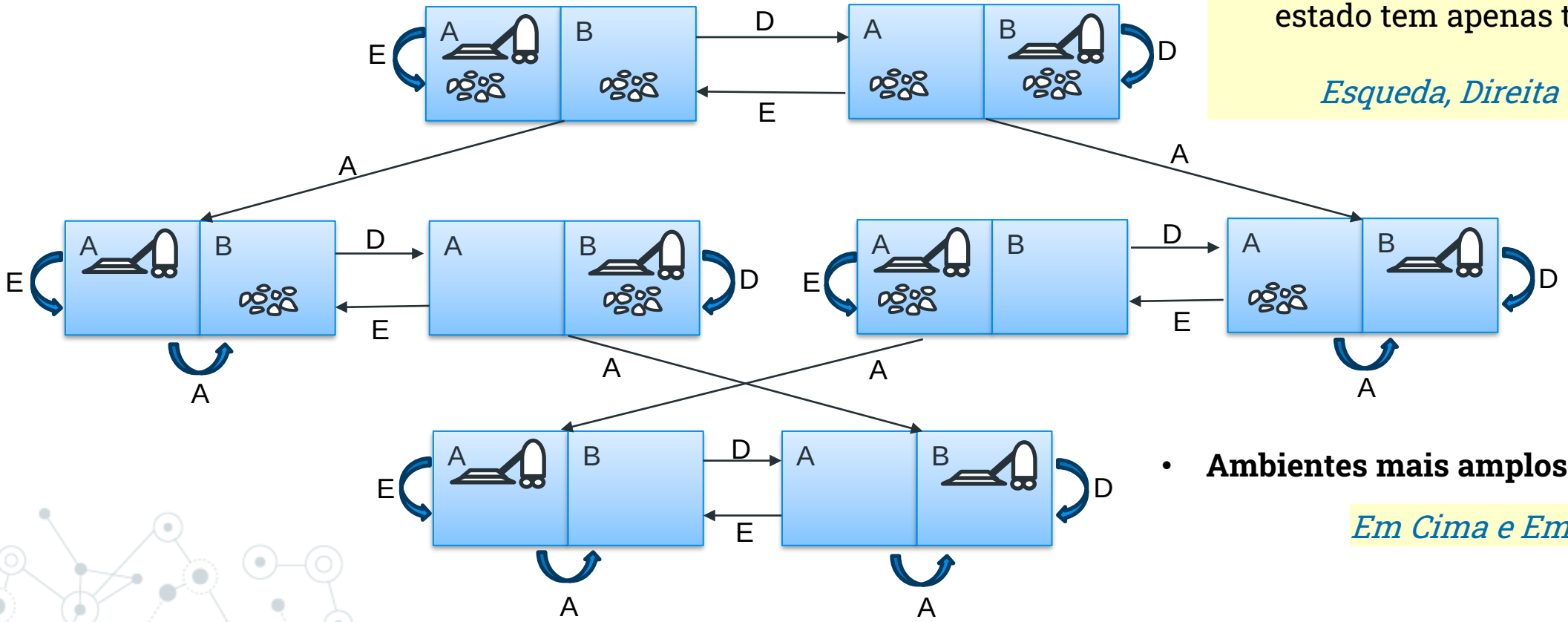
- **Mundo do Aspirador de Pó**



- **Estado Inicial:** Qualquer estado pode ser designado como o estado inicial.

Exemplos de Problemas

- **Mundo do Aspirador de Pó**



- **Ações:** Nesse ambiente, cada estado tem apenas três ações:

  Esqueda, Direita e Aspirar

- **Ambientes mais amplos** podem incuir:

  Em Cima e Embaixo

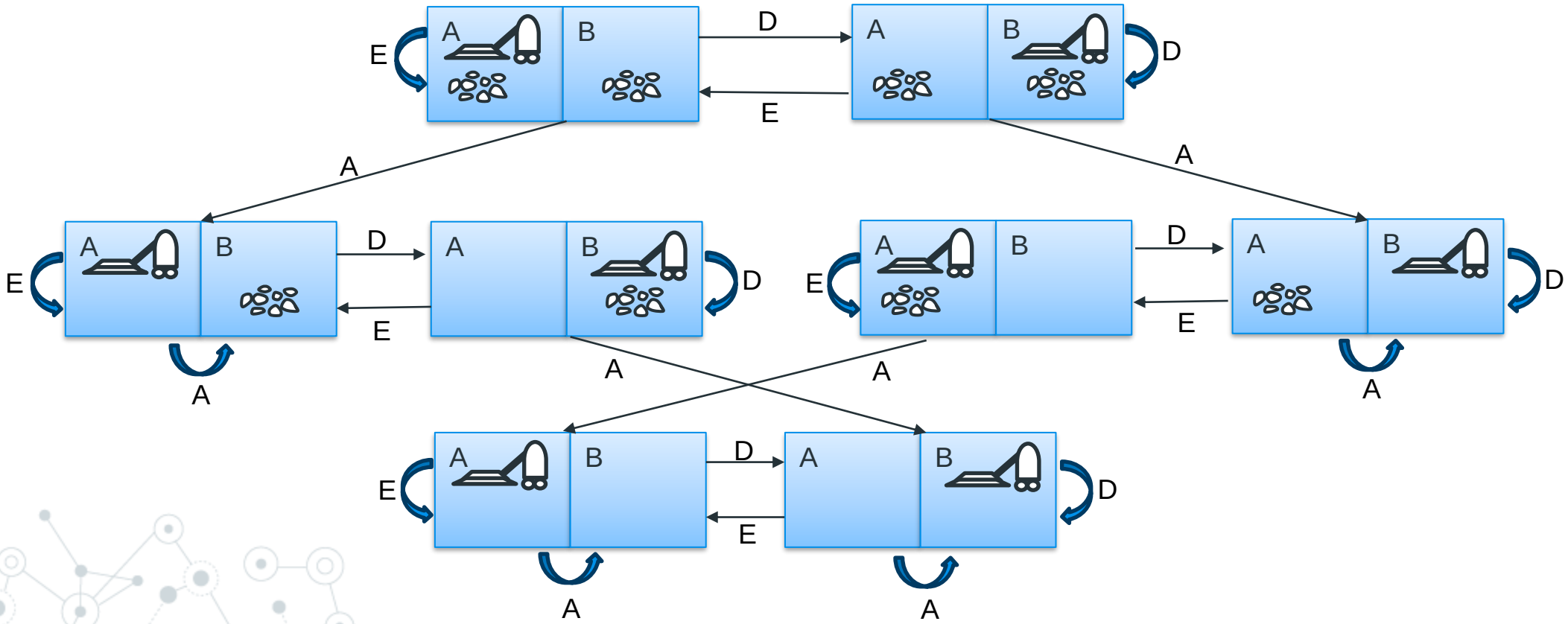
Exemplos de Problemas

Mundo do Aspirador de Pó

- **Modelo de Transição:** As ações têm seus efeitos esperados, a não ser as ações:

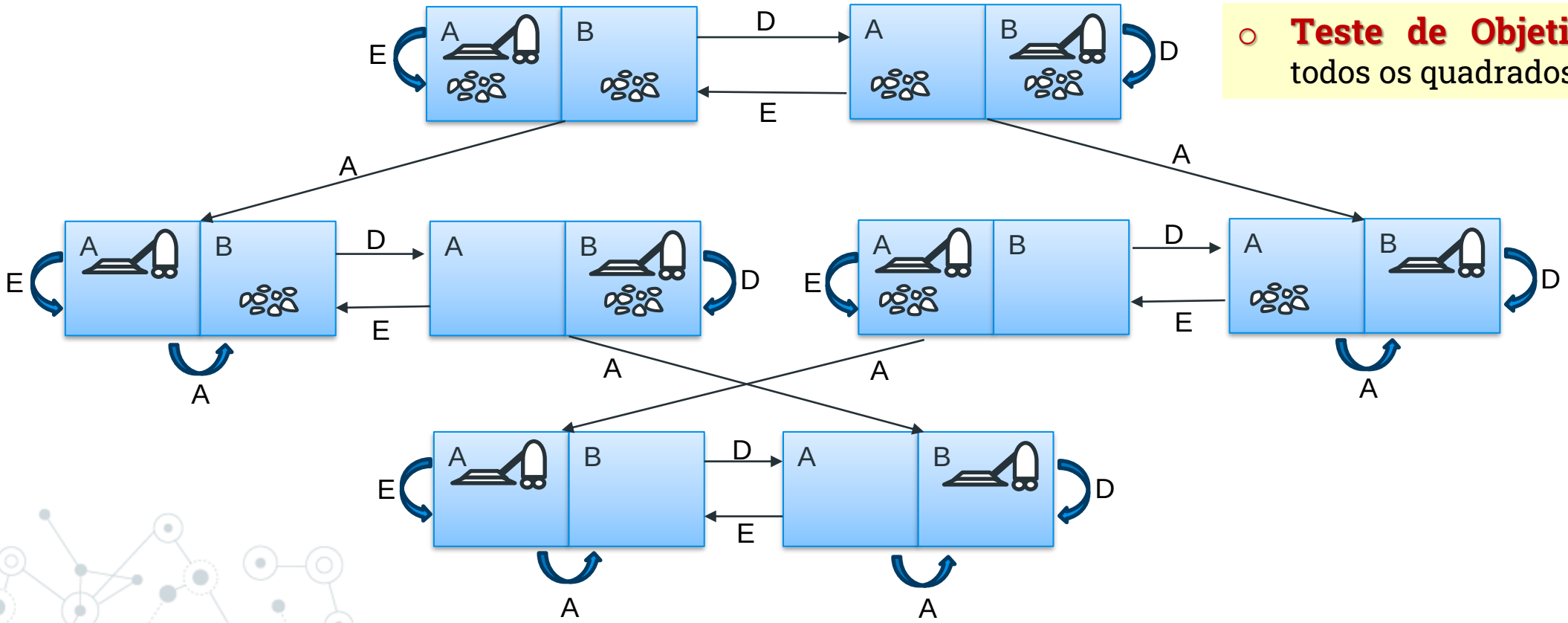
Mover para a Esquerda no quadrado mais à Esquerda.
Mover para a Direita no quadrado mais à Direita
Aspiras, no quadrado limpo.

- **Essas ações não tem nenhum efeito.**



Exemplos de Problemas

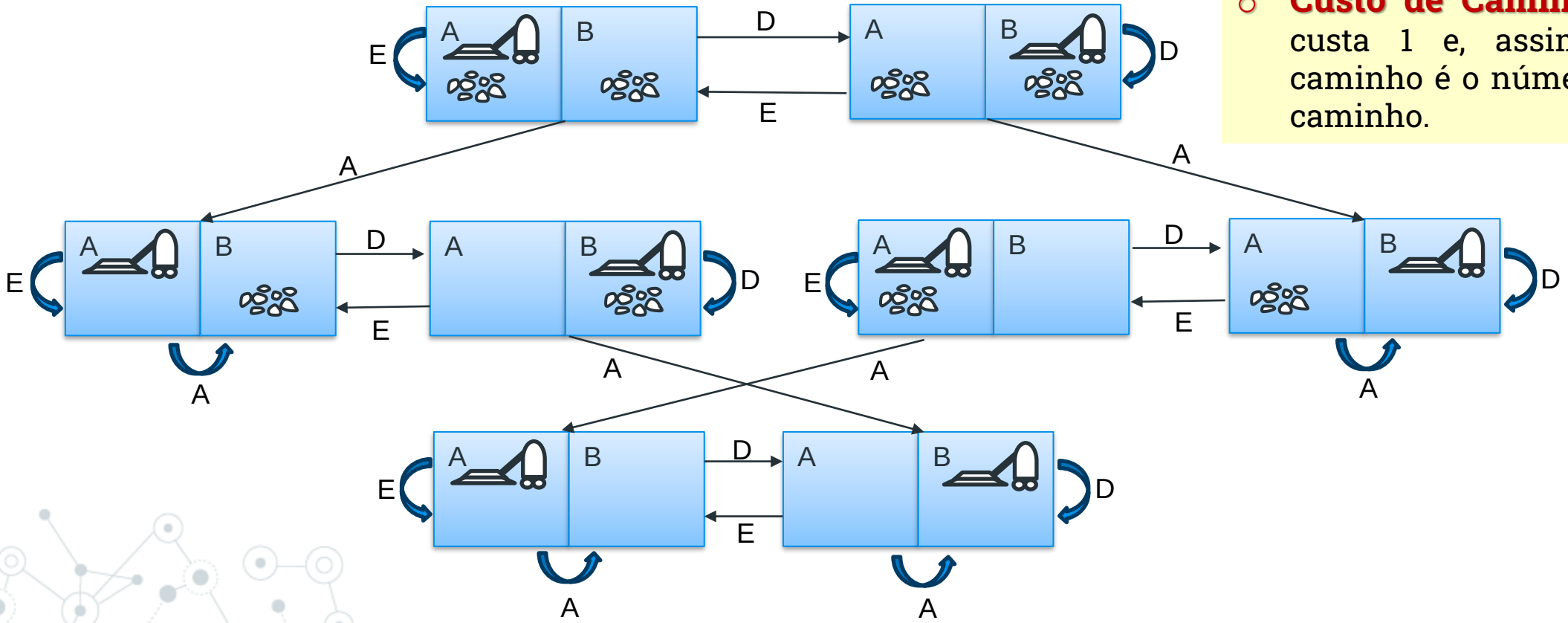
- **Mundo do Aspirador de Pó**



- **Teste de Objetivo:** Verifica se todos os quadrados estão limpos.

Exemplos de Problemas

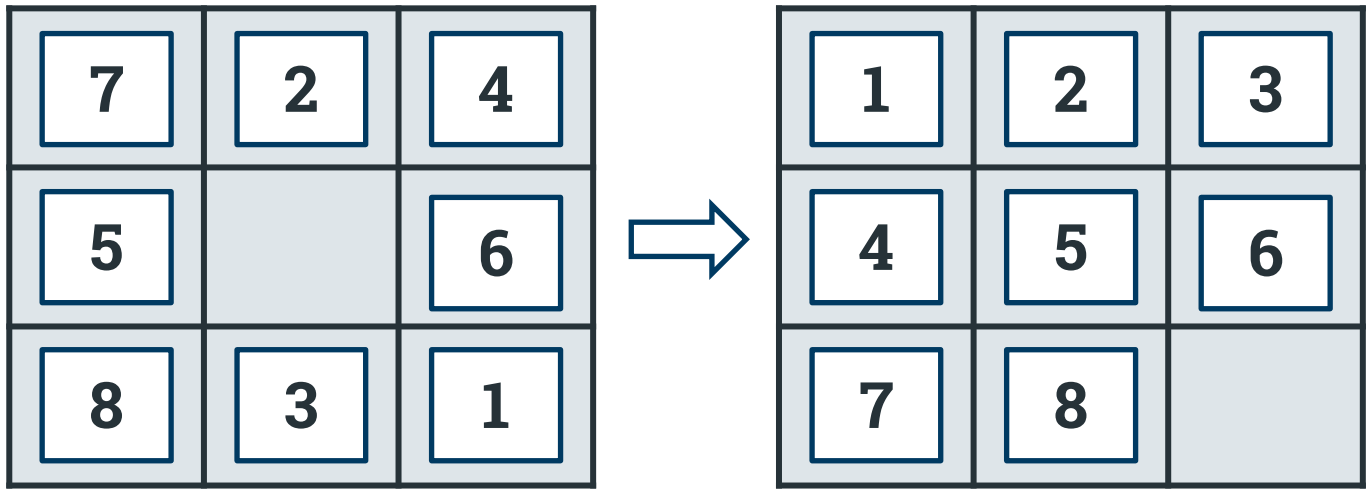
- **Mundo do Aspirador de Pó**



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

Exemplos de Problemas

- **Quebra-Cabeça de Oito Peças**



Estado Inicial

Estado Objetivo

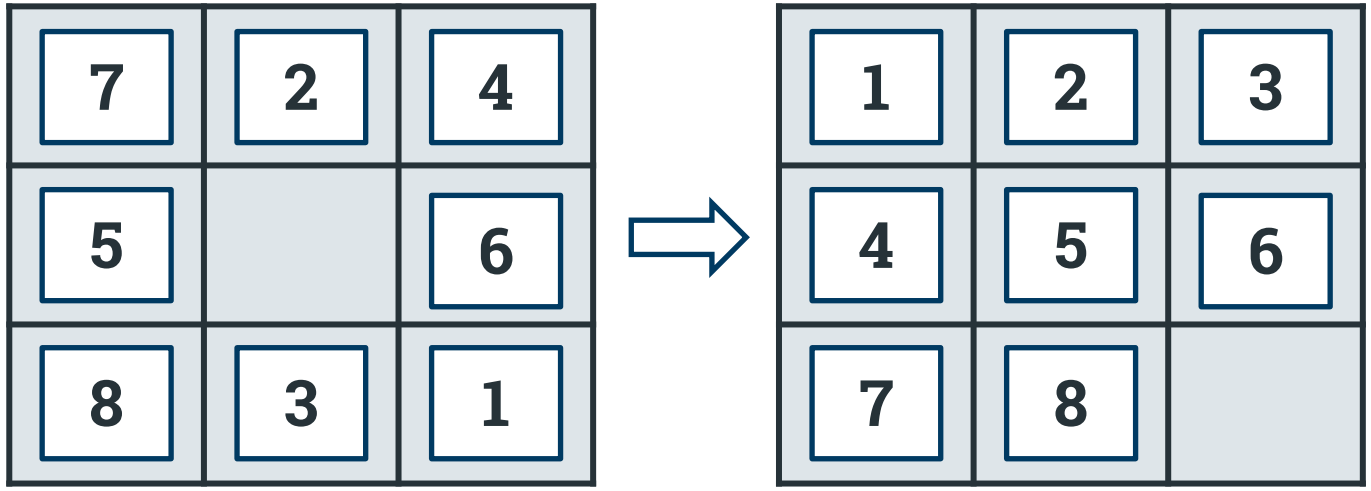
- **Estados:** Uma descrição de estado especifica a **posição de cada uma das oito peças e do quadrado vazio** em um dos nove quadrados.

- **Estado Inicial:** Qualquer estado pode ser designado como o estado inicial.

- Qualquer objetivo específico pode ser alcançado a partir de exatamente metade dos estados iniciais possíveis.

Exemplos de Problemas

- **Quebra-Cabeça de Oito Peças**



Estado Inicial

Estado Objetivo

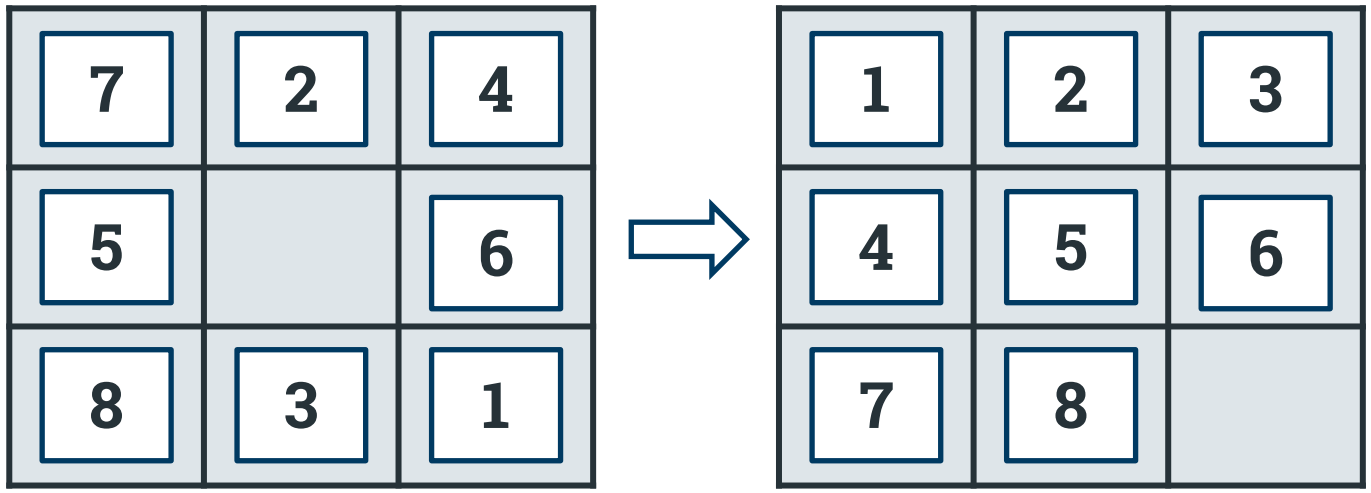
- **Ações:** A formulação mais simples define as ações como:

  Movimentos do quadrado vazio para Esquerda, Direita, Para Cima ou Para Baixo.

- Pode haver subconjuntos diferentes desses, dependendo de onde estiver o quadrado vazio.

Exemplos de Problemas

Quebra-Cabeça de Oito Peças



Estado Inicial

Estado Objetivo

- **Modelo de Transição:** Dado um estado e ação, ele devolve o estado resultante.
- **Teste de Objetivo:** Verifica se o estado corresponde à configuração de estado objetivo.
- **Custo de Caminho:** Cada passo custa 1 e, assim, o custo do caminho é o número de passos do caminho.

- O quebra-cabeça de oito peças tem **9!/2 = 181.440 estados** acessíveis e é resolvido com facilidade.

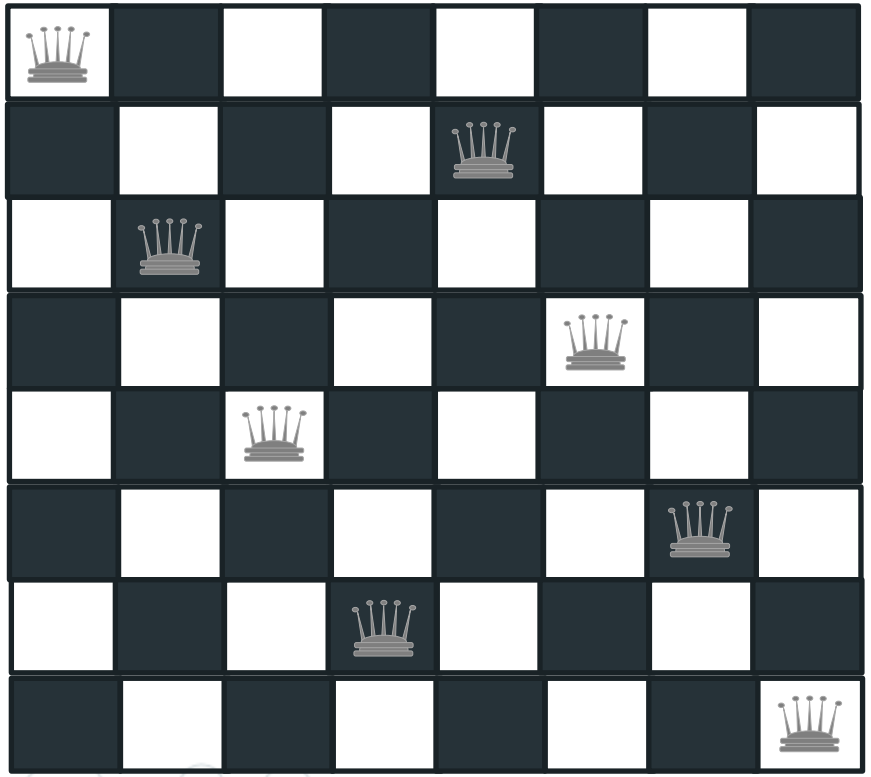
15 peças → **1.3 trilhão de estados.**
24 peças → 10^{25} **estados.**

Exemplos de Problemas

- **Quebra-Cabeça de Oito Peças**
  - **Abstração incluídas**
    - As ações são reduzidas a seus **estados iniciais e finais**, ignorando-se as posições intermediárias por onde o bloco está deslizando.
    - Foram abstraídas ações como sacudir o tabuleiro quando as peças ficam presas ou extrair as peças com uma faca e colocá-las de volta no tabuleiro.
  - Pertence à família de **quebra-cabeças de blocos deslizantes** – usados com frequência como problemas de teste para novos algoritmos de busca em IA.

Exemplos de Problemas

- **Problema das 8 Rainhas**

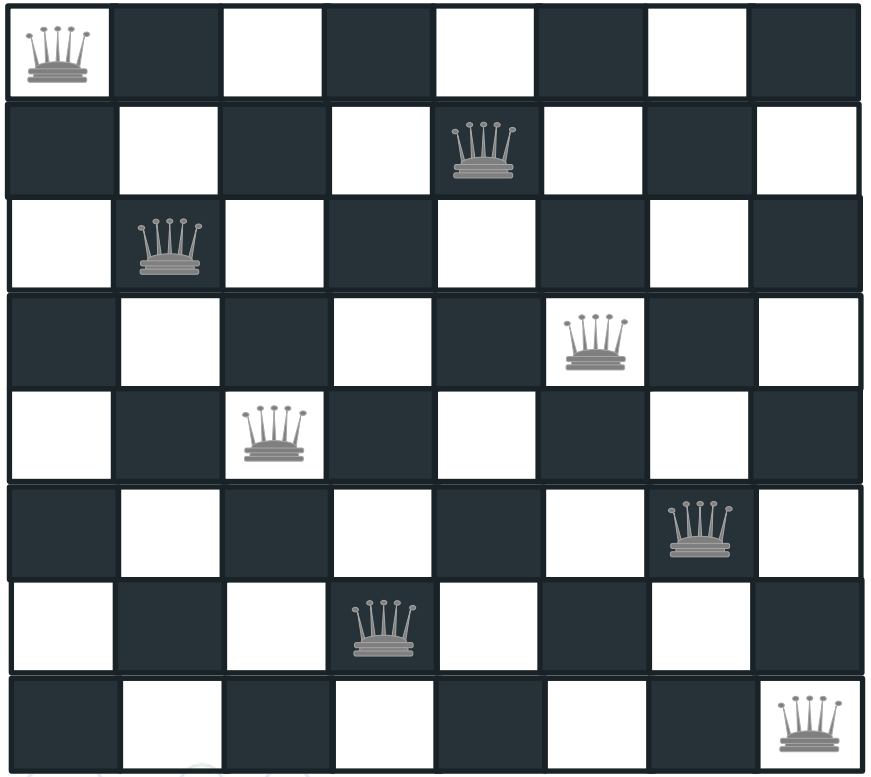


- O objetivo é **posicionar as 8 rainhas** no tabuleiro de tal forma que nenhuma rainha possa atacar qualquer uma das outras. *Uma rainha ataca qualquer peça situada na mesma linha, coluna ou diagonal.*
  - **Estados:** Qualquer disposição de 0-8 rainhas no tabuleiro é um estado.
  - **Estado Inicial:** Nenhuma rainha no tabuleiro.
  - **Ações:** Colocar uma rainha em qualquer quadrado vazio.
  - **Modelo de Transição:** Devolver uma rainha adicionada em qualquer quadrado específico no tabuleiro.
  - **Teste de Objetivo:** Oito rainhas estão no tabuleiro e nenhuma é atacada.

$64 \cdot 63 \cdot \cdots \cdot 57 \approx 1{,}8 \times 10^{14}$ sequências possíveis.

Exemplos de Problemas

- **Problema das 8 Rainhas**



- O objetivo é **posicionar as 8 rainhas** no tabuleiro de tal forma que nenhuma rainha possa atacar qualquer uma das outras. *Uma rainha ataca qualquer peça situada na mesma linha, coluna ou diagonal.*

✓ **SIMPLIFICAÇÃO DO PROBLEMA:**

Proíbe a colocação de uma rainha em qualquer quadrado que já estiver sob ataque.

- **Estados:** Todas as possíveis disposições de n rainhas ($0 \leq n \leq 8$), uma por coluna nas n colunas mais à esquerda, sem que nenhuma rainha ataque outra.

- **Ações:** Adicione uma rainha a qualquer quadrado na coluna vazia mais à esquerda, de tal modo que ela não seja atacada por qualquer outra rainha.

2057 sequências possíveis.

Exemplo de Problemas do Mundo Real

- São aqueles cujas soluções de fato preocupam as pessoas.
- Eles tendem a não representar uma única descrição consensual, mas tentaremos dar uma ideia geral de suas formulações.
- Sua descrição é normalmente **complexa.**

Exemplos de Problemas

- **Problema de Roteamento**
  - Rotas de pacotes na internet, **rotas de aviões em aeroportos** e passagem de trilhas em placas de circuitos.

✓ **VERSÃO SIMPLIFICADA DO PROBLEMA:**

- **Estados:** Cada um é representado por uma posição (por exemplo das aeronaves) e pela hora atual.
- **Estado Inicial:** Situação das aeronaves na hora 00:00am.
- **Ações:** Definida conforme movimento previsto para as aeronaves ao longo do dia.
- **Teste de Objetivo:** Os voos devem chegar e partir no horário correto.
- **Custo de Caminho:** Consumo de recursos (combustível, tempo de pessoal, etc.).

Exemplos de Problemas

Diagnóstico Médico

- Em um sistema de diagnóstico médico, podemos partir de um conjunto de **perguntas que serão respondidas** por exames clínicos e laboratoriais, por exemplo:
  - *O Cliente tem febre?*
  - *Está com algum sintoma na pele?*
  - *Reclama de alguma dor?*
  - *Está com catarro no pulmão?*
  - *Quais os valores máximo e mínimos da pressão arterial?*
  - *Qual o valor do colesterol HDL?*
- Também podemos observar que determinadas perguntas possuem relação entre si, formando uma **árvore**.

Exemplos de Problemas

Diagnóstico Médico

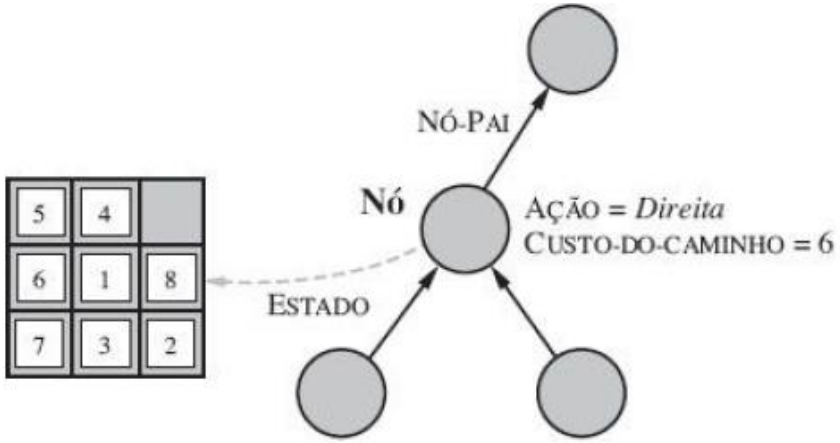
- **Estados:** Cada pergunta que é feita ocupa um estado.
- **Estado Inicial:** Desconhecimento total da condição do paciente.
- **Ações:** Dependendo da pergunta pode ser sim ou não (ex: o paciente tem febre), também pode ser limites de um exame (ex: qual o valor da pressão arterial máxima).
- **Teste de Objetivo:** Encontrar o diagnostico correto para a doença do paciente.
- **Custo de Caminho:** Novamente depende da pergunta, pode ser minutos do médico ou valores cobrados pelo laboratório.

Montagem de Árvore

- Depois de formular alguns problemas, precisamos **resolvê-los**.
- Uma solução é uma sequência de ações, de modo que os algoritmos de busca consideram várias sequências de ações possíveis.
- As sequências de ações possíveis que começam a partir do estado inicial formam uma **árvore de busca** com o estado inicial na raiz;
  - Os **ramos** são as ações;
  - Os **nós** correspondem aos estados no espaço de estados do problema.

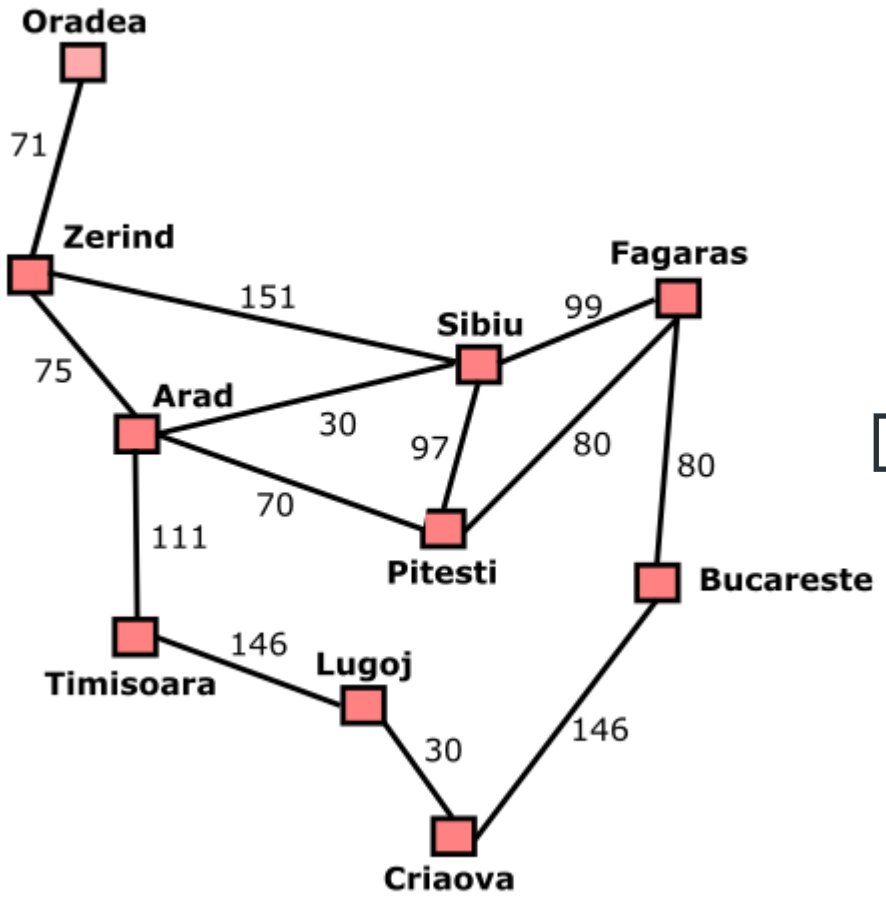
Montagem de Árvore

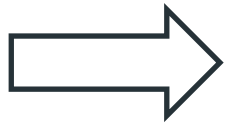
- **Representação de um Nó:**
  - **ESTADO:** o estado no espaço de estado a que o nó corresponde;
  - **PAI:** o nó na árvore de busca que gerou esse nó;
  - **AÇÃO:** a ação que foi aplicada ao pai para gerar o nó;
  - **CUSTO-DO-CAMINHO:** o custo, tradicionalmente denotado por $g(n)$, do caminho do estado inicial até o nó, indicado pelos ponteiros para os pais.

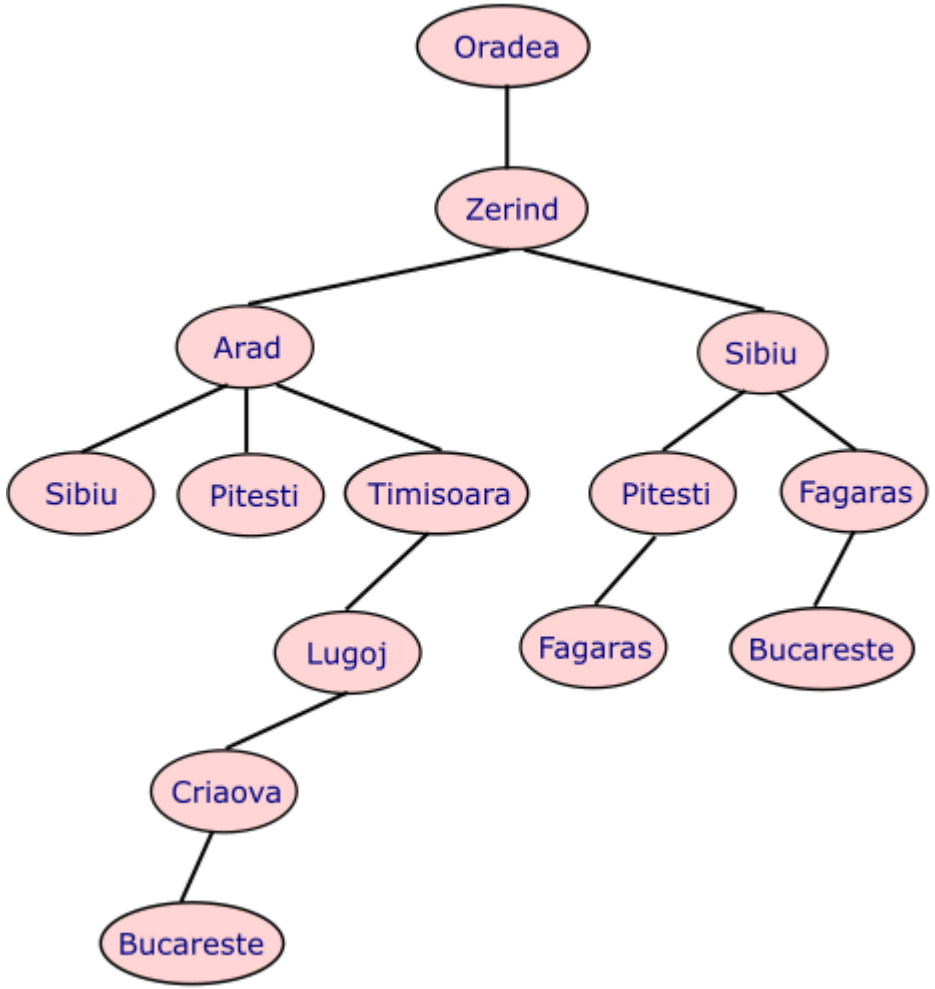


Montagem de Árvore

- **Problema de Roteamento**



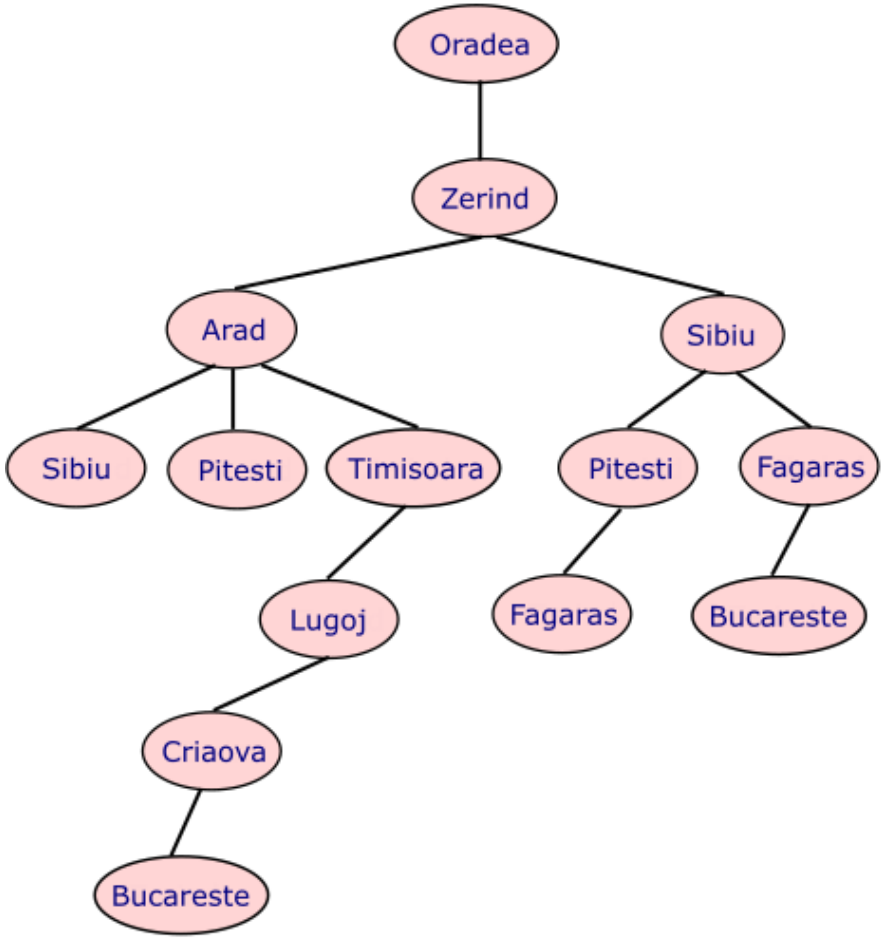




Montagem de Árvore

Problema de Roteamento

- Um **Nó** é uma anotação da estrutura de dados usada para representar a árvore de busca.
- Um **Estado** corresponde a uma configuração do mundo. Assim, os **nós** estão em caminhos particulares, tal como definido pelos ponteiros PAI, enquanto os estados não estão.
- Além disso, *dois nós diferentes podem conter o mesmo estado do mundo se esse estado for gerado através de dois caminhos de busca diferentes.*



Estratégias de Busca

Estratégias de Busca

Por Que Busca?

> *"Um **agente** com várias opções imediatas de valor desconhecido pode decidir o que fazer examinando, em primeiro lugar, diferentes possíveis sequências de ações que levam a estados de valores conhecidos e, em seguida, escolher a melhor sequência de ações "*

- A busca simula ações no mundo com certo nível de abstração para entender como proceder ou resolver um problema.

- **Agentes de Resolução de Problemas**
  - Um tipo de agente baseado em objetivo

Estratégias de Busca

- Um **agente** com várias opções imediatas pode decidir o que fazer comparando diferentes sequências de ações possíveis.
- Esse processo de procurar pela melhor sequencia é chamado de **busca**.

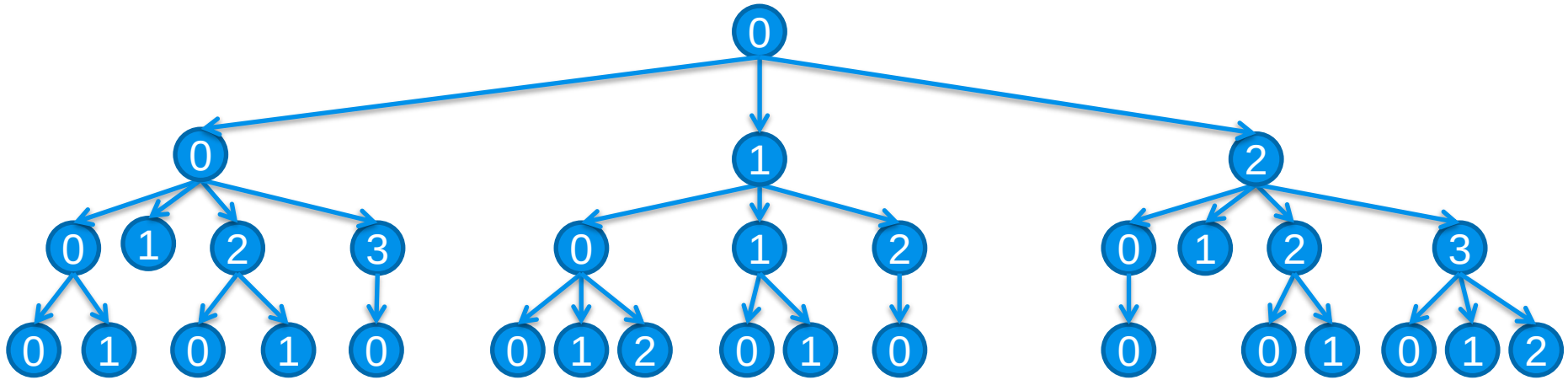
Formular Objetivo → Buscar → Executar

- **Abordagens de Busca:**
  - **Busca Cega** (*Sem informação/Não informada*): Não tem informação sobre qual sucessor é mais promissor para atingir a meta.
  - **Busca Heurística** (*Busca Com Informação/Informada*): Possui informação (estimativa) de qual sucessor é mais promissor para atingir a meta.

    É uma busca cega com algum guia ou orientação.

Todas as estratégias de busca se distinguem pela ordem em que os nós são expandidos.

Busca Cega



Estratégias de Busca

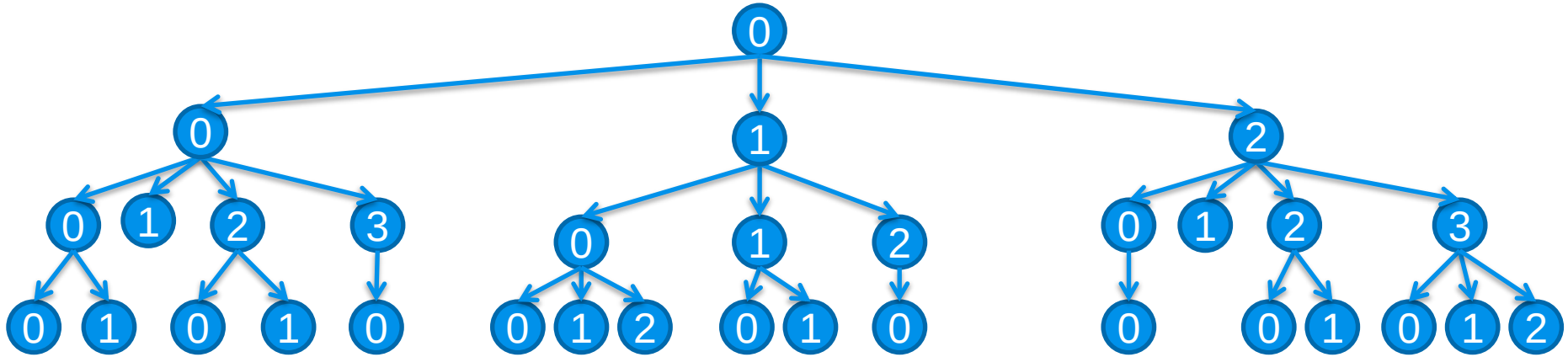
- **Busca Cega** *(Blind Search ou Uninformed Search).*
  - Uma estratégia é dita Busca Cega se ela *não têm nenhuma informação* adicional sobre estados, além daquelas fornecidas na definição do problema. Tudo o que elas podem fazer é gerar sucessores e distinguir um estado objetivo de um estado não objetivo.

- Todas as estratégias de busca se distinguem pela **ordem** em que os nós são expandidos.

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

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*
  - Estratégia simples em que o nó raiz é expandido primeiro, em seguida todos os sucessores do nó raiz são expandidos, depois os sucessores desses nós, e assim por diante. Em geral, todos os nós em dada profundidade na árvore de busca são expandidos, antes que todos os nós no nível seguinte sejam expandidos.

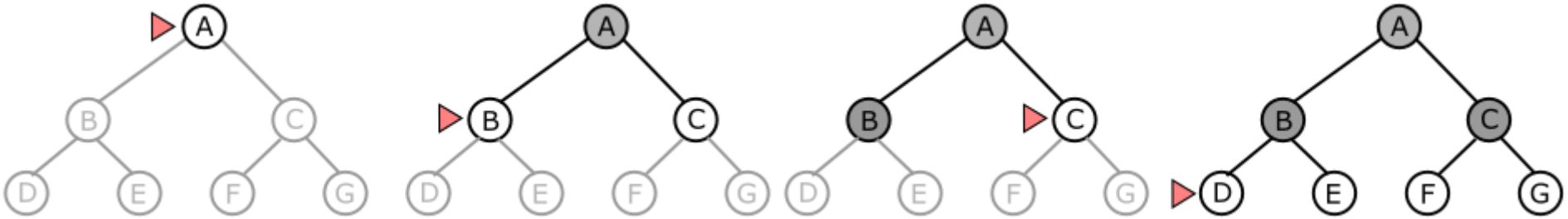


Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Ordem de expansão dos nós:

1. Nó raiz
2. Todos os nós de profundidade 1
3. Todos os nós de profundidade 2, etc ...



Busca em largura em uma árvore binária simples. Em cada fase, o próximo nó a ser expandido é indicado por um marcador.

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Características: Completa e Ótima

- Se existe solução, esta será encontrada;
- A solução encontrada primeiro será a de menor profundidade.

Vantagens:

- Completo
- Ótimo, sob certas condições (por exemplo, é ótimo se os operadores sempre têm o mesmo custo).

Desvantagens:

- Requer muita memória e tempo (complexidade exponencial). Todos os caminhos selecionados possuem a mesma importância.

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*
  - **Suponha a busca em uma árvore uniforme onde cada estado tenha b sucessores.**
    - A raiz da árvore de busca gera b nós no primeiro nível;
    - Cada um dos nós gera b outros nós, totalizando b^2 no segundo nível.
    - Cada um desses outros nós gera b outros nós, totalizando b^3 no terceiro nível e assim por diante.
    - Suponha que a solução esteja na profundidade d. No pior caso, é o último nó gerado naquele nível. Então o **número total de nós gerados** é:

$$b + b^2 + b^3 + \cdots b^d = O(b^d)$$

Assim, para $d = 10$, temos:

Tempo de processamento de um milhão de nós/segundo;

Consumo de memória de 1000 bytes/nó.

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

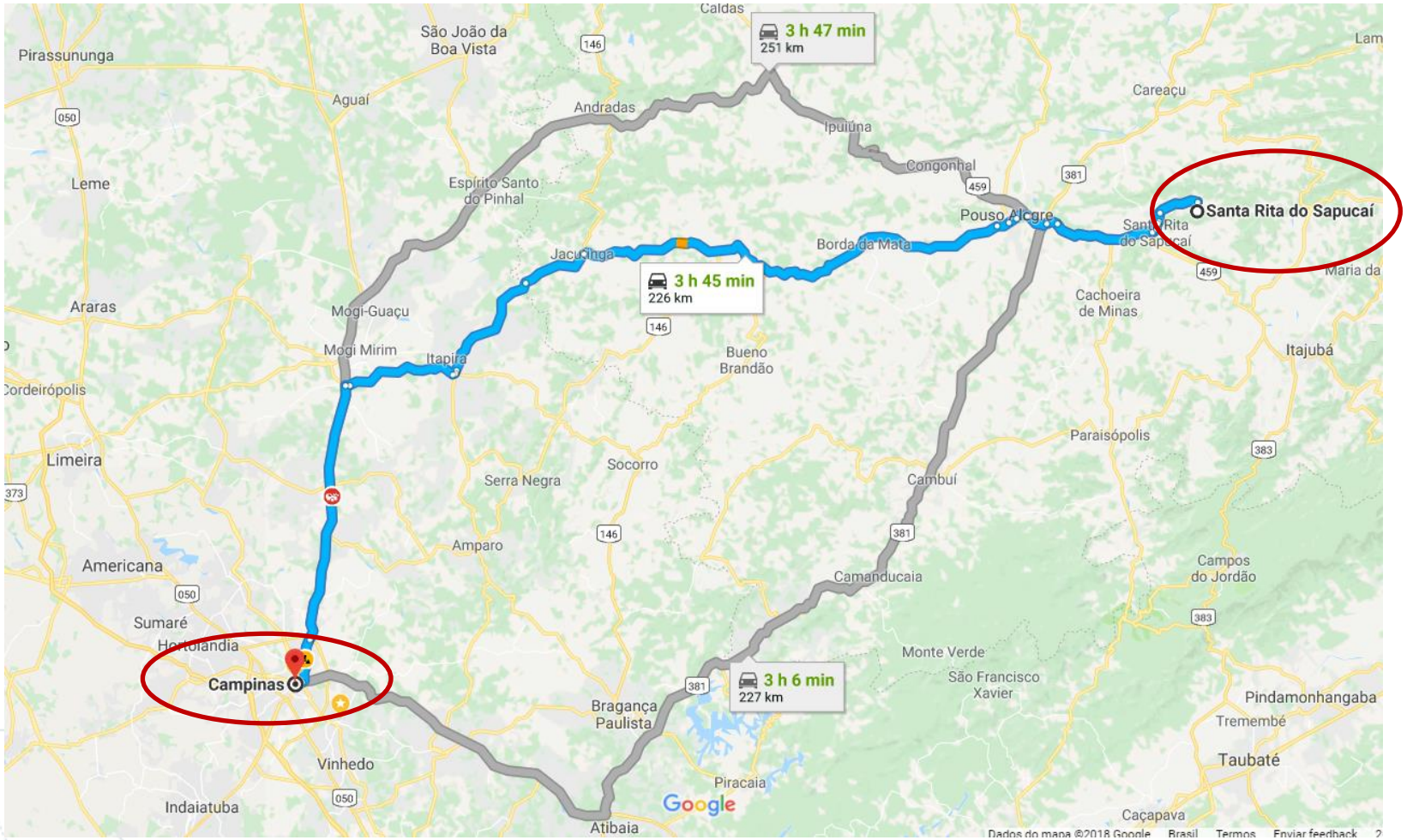
Profundidade	Nós	Tempo	Memória
2	110	0,11 milissegundos	107 KB
4	11.110	11 milissegundos	10,6 MB
6	10^6	1,1 segundo	1 GB
8	10^8	2 minutos	103 GB
10	10^{10}	3 horas	10 TB
12	10^{12}	13 dias	1 PB
14	10^{14}	3.5 anos	99 PB
16	10^{16}	350 anos	10 EB

Observações:

- *Os requisitos de memória são um problema maior para a busca em largura do que o tempo de execução.*
- O tempo ainda é um fator importante.
- Em geral, *os problemas de busca de complexidade exponencial não podem ser resolvidos por métodos sem informação, para qualquer instância, exceto as menores.*

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

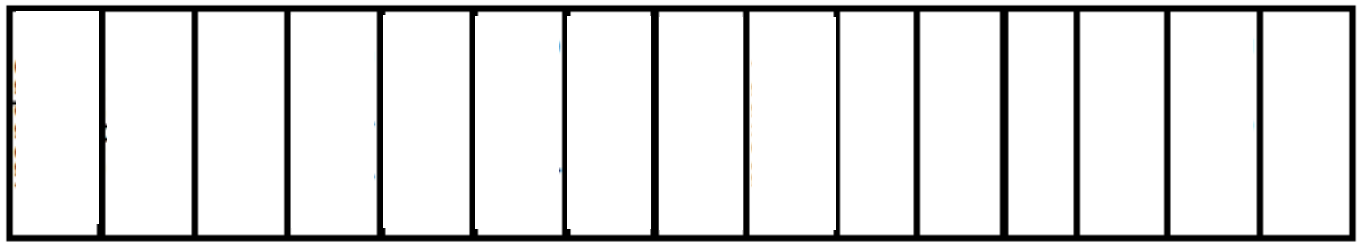


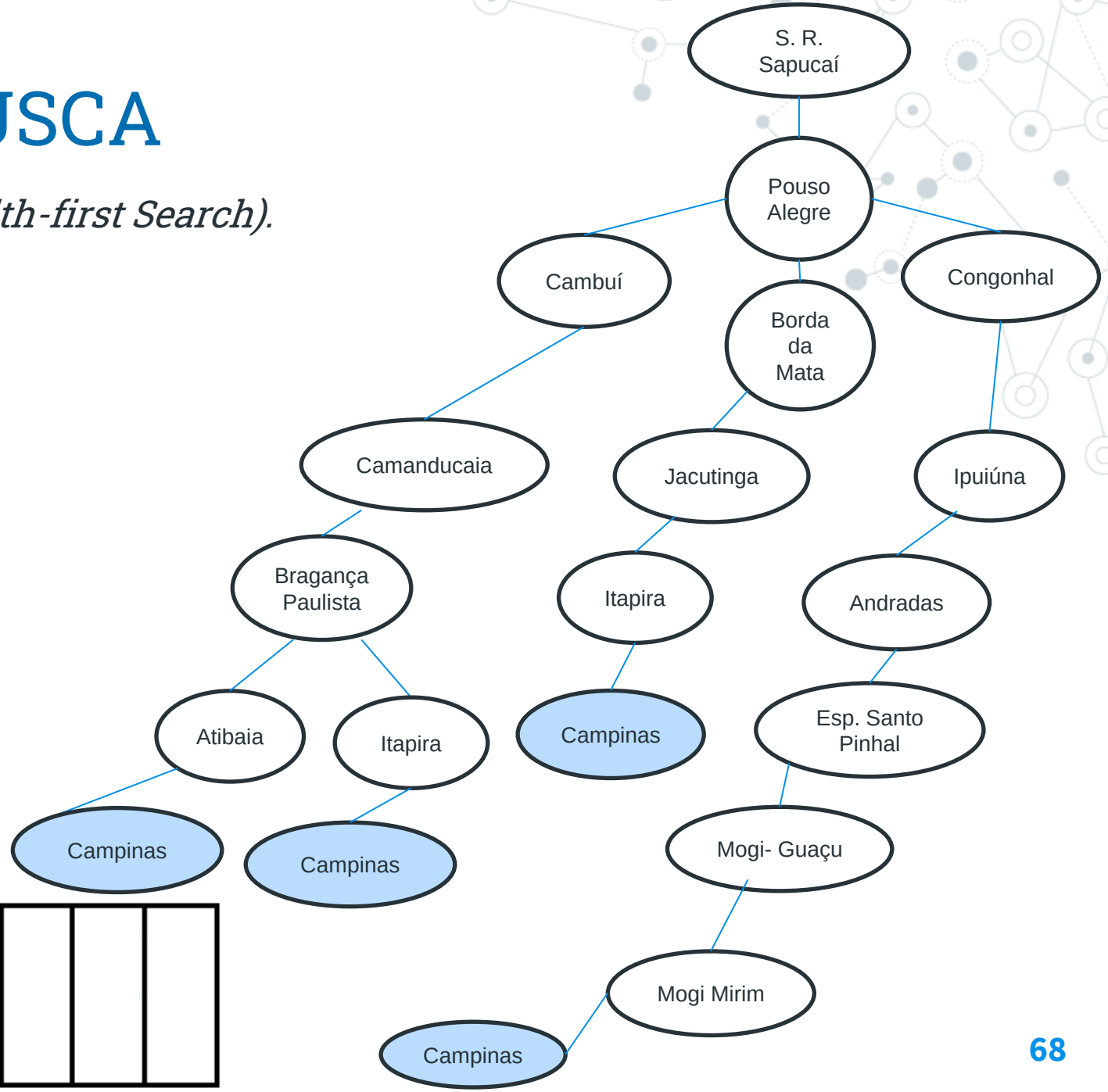
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



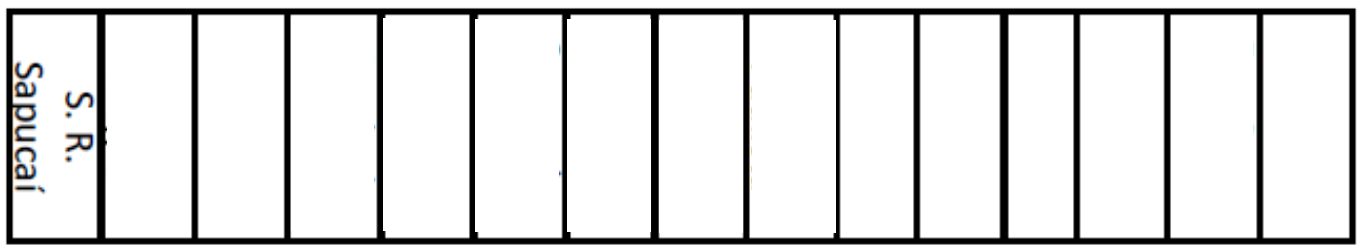


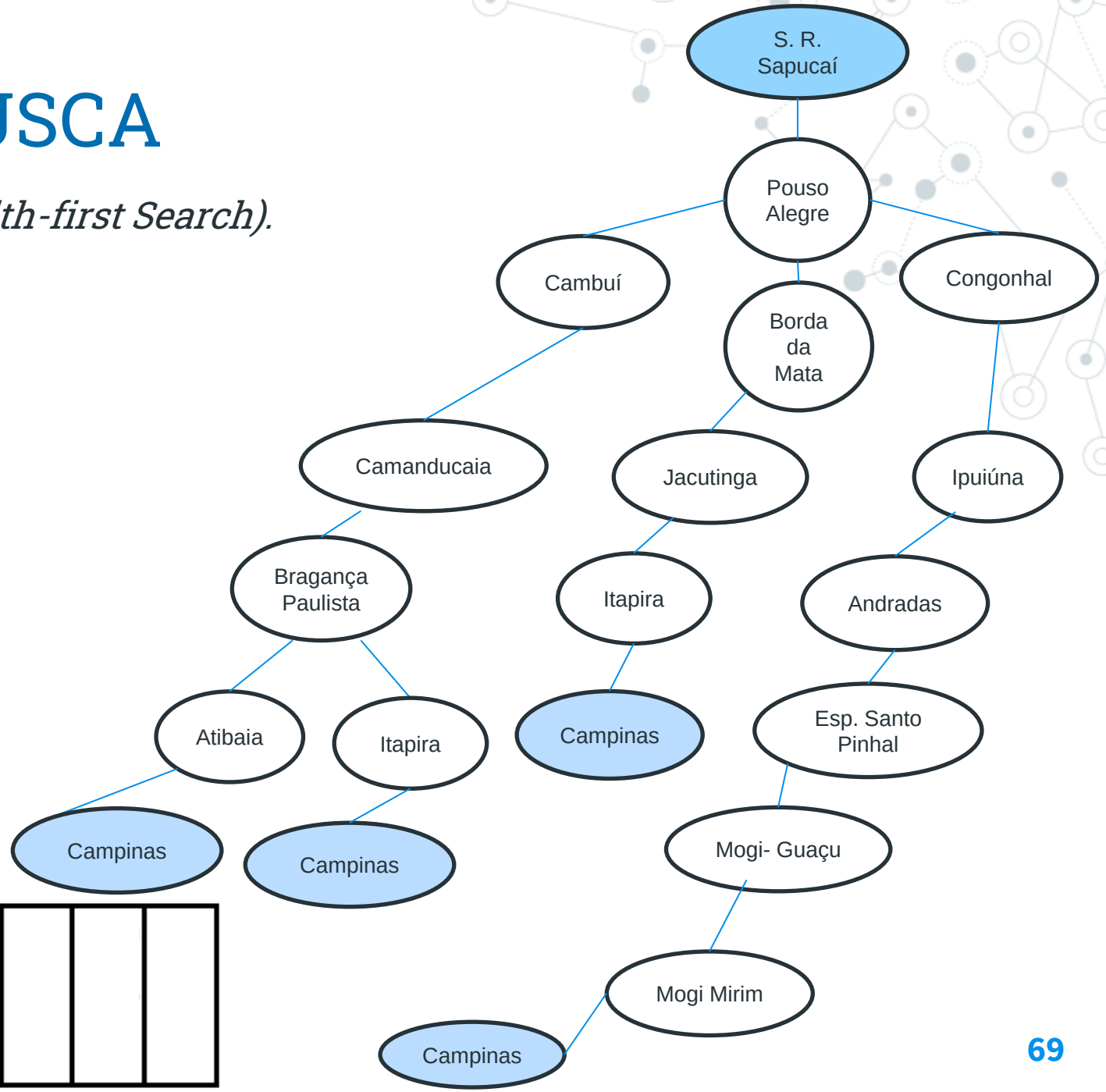
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



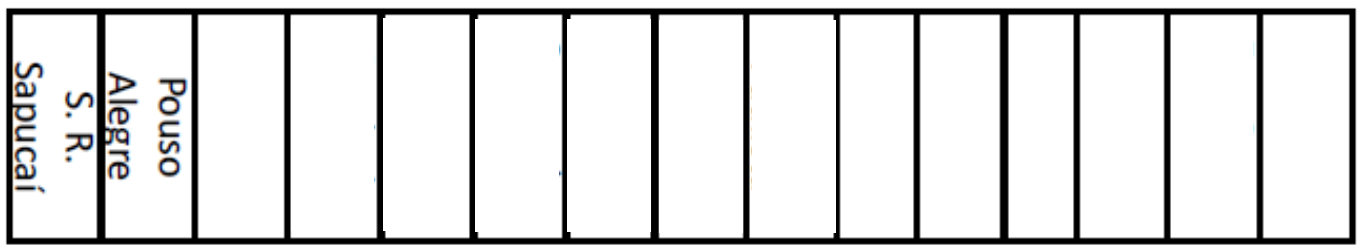


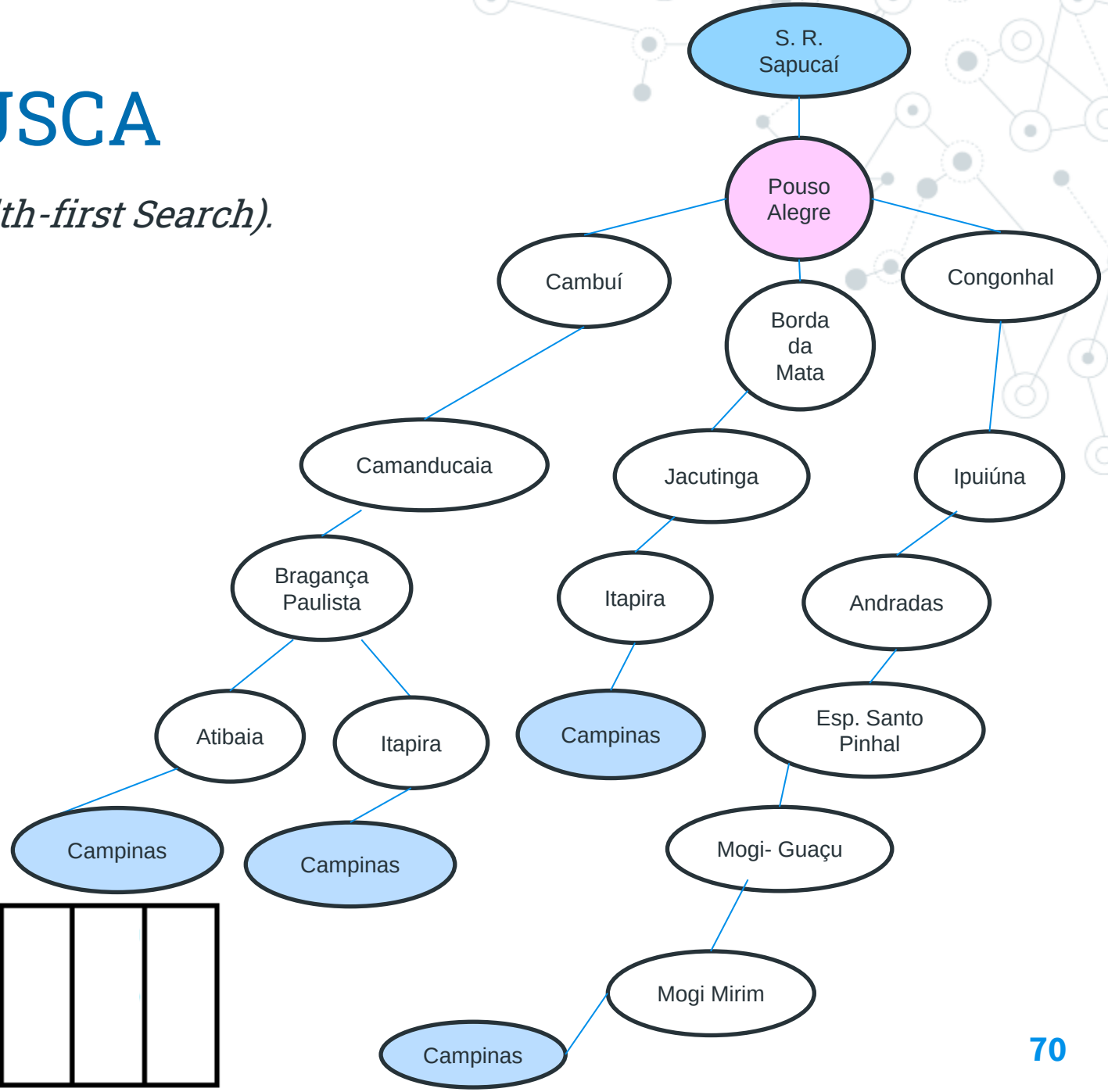
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



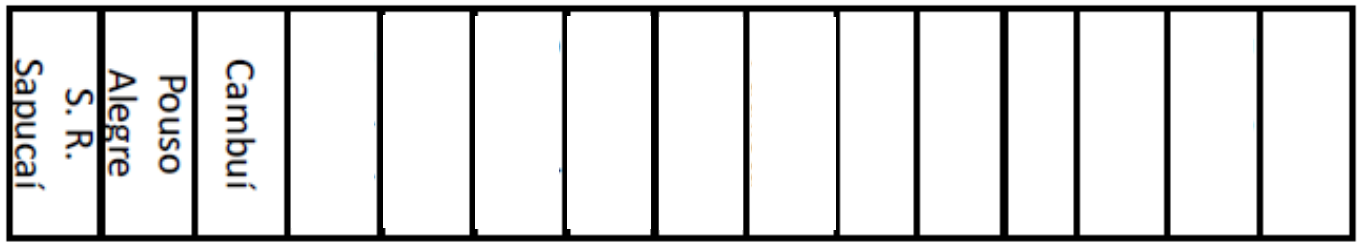


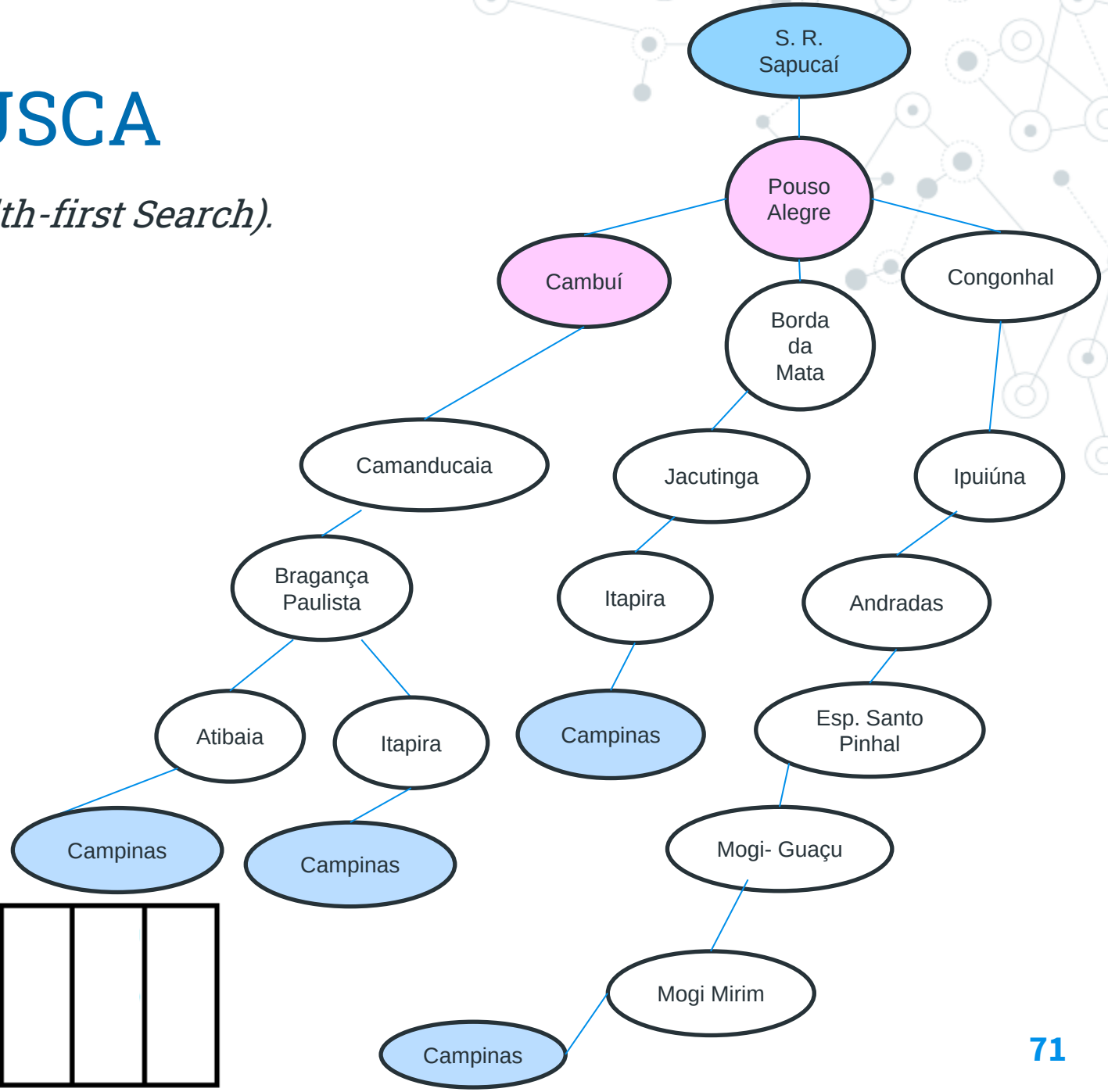
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



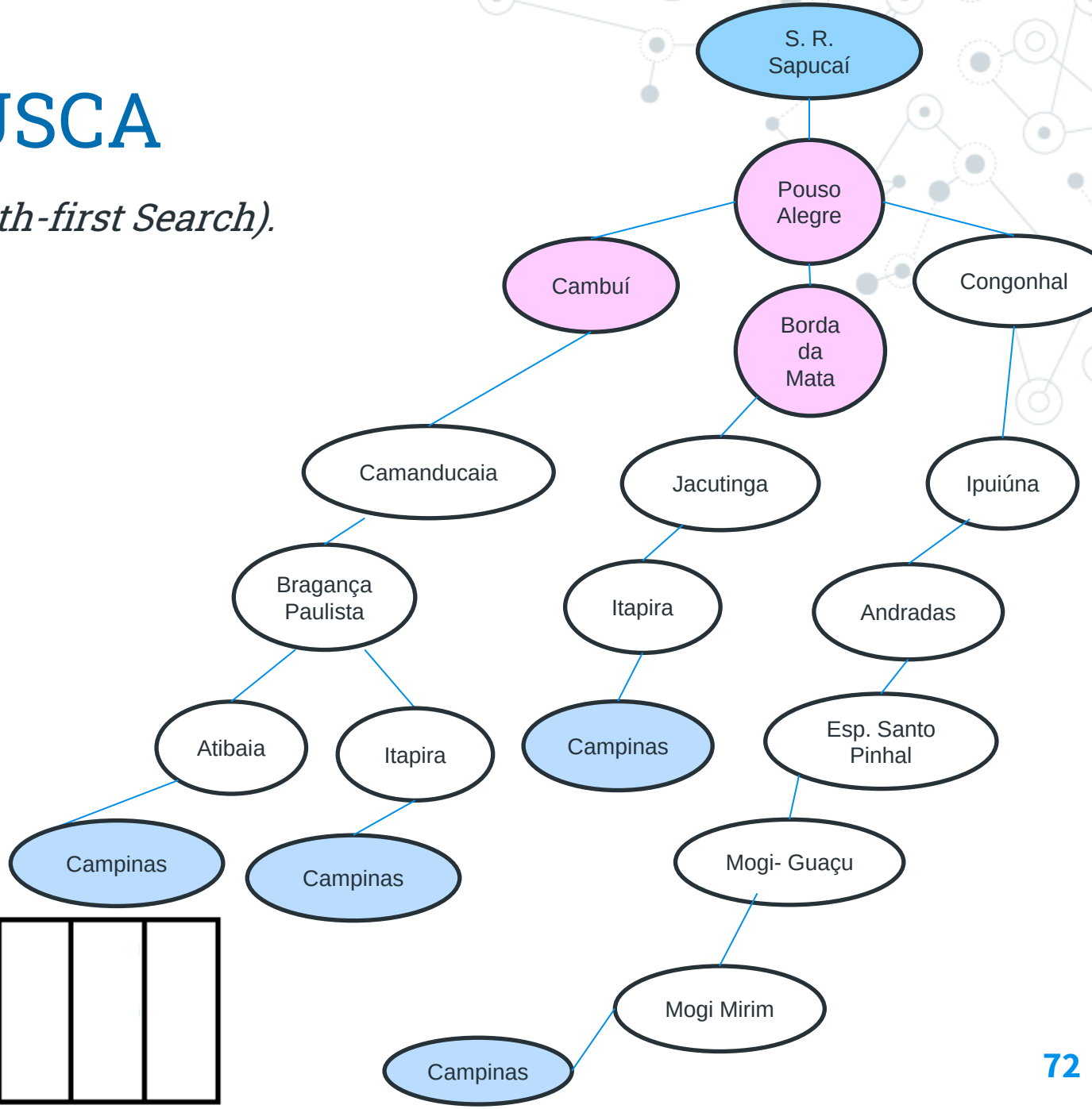


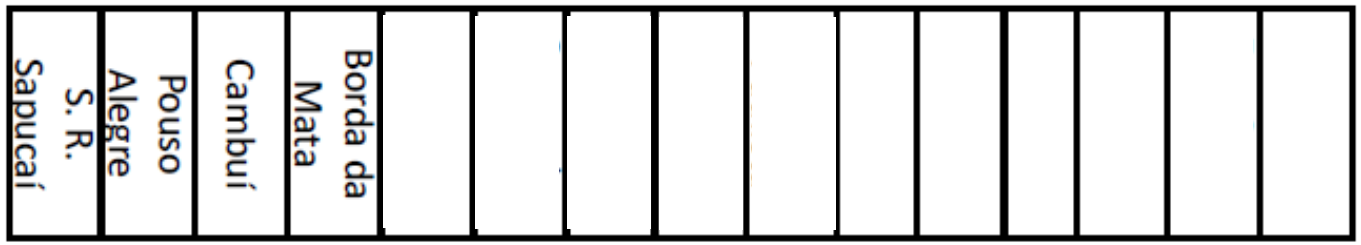
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



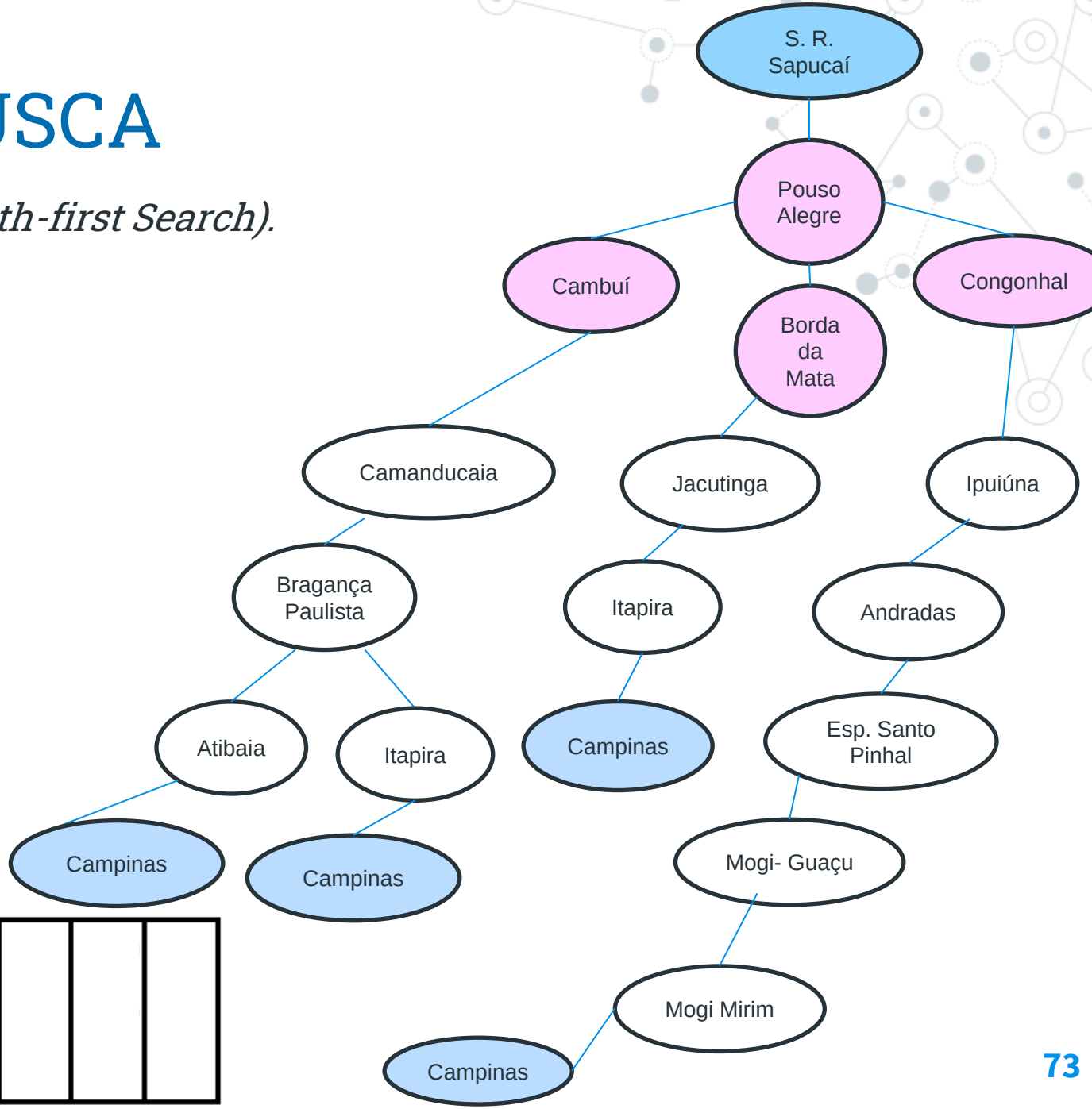


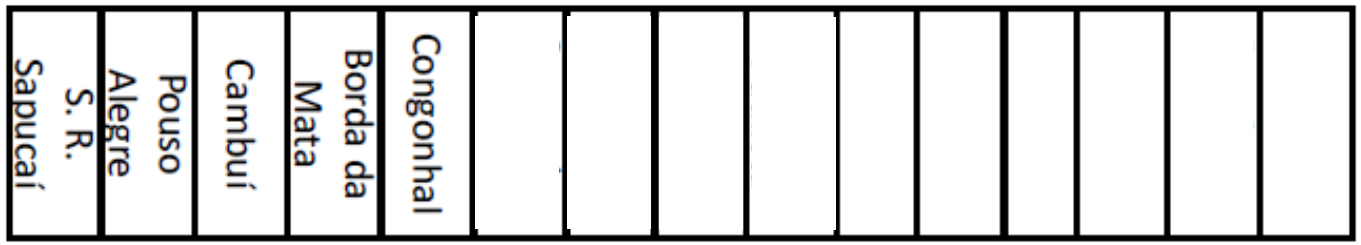
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



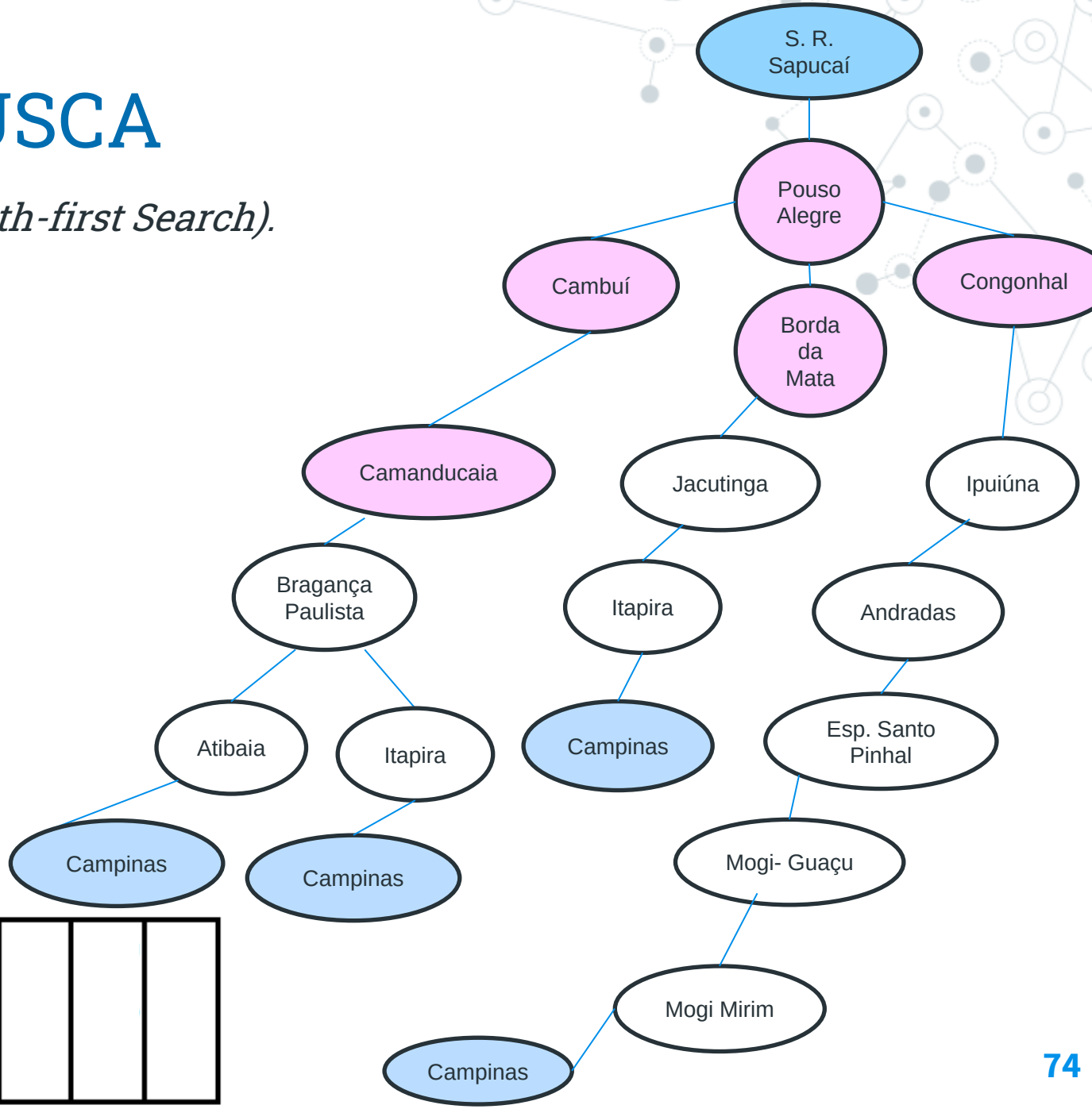


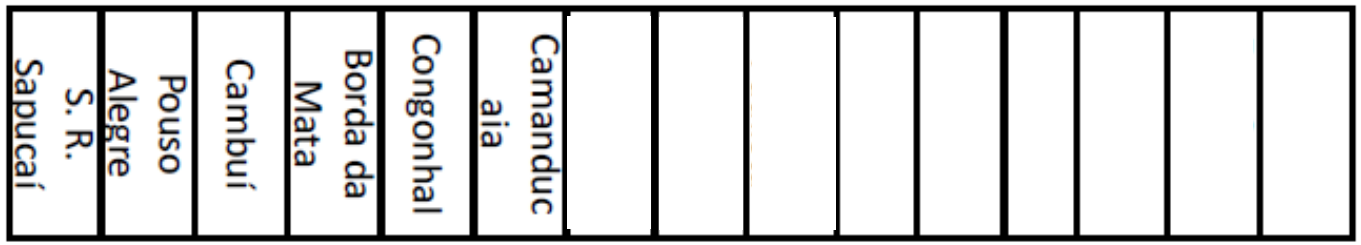
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



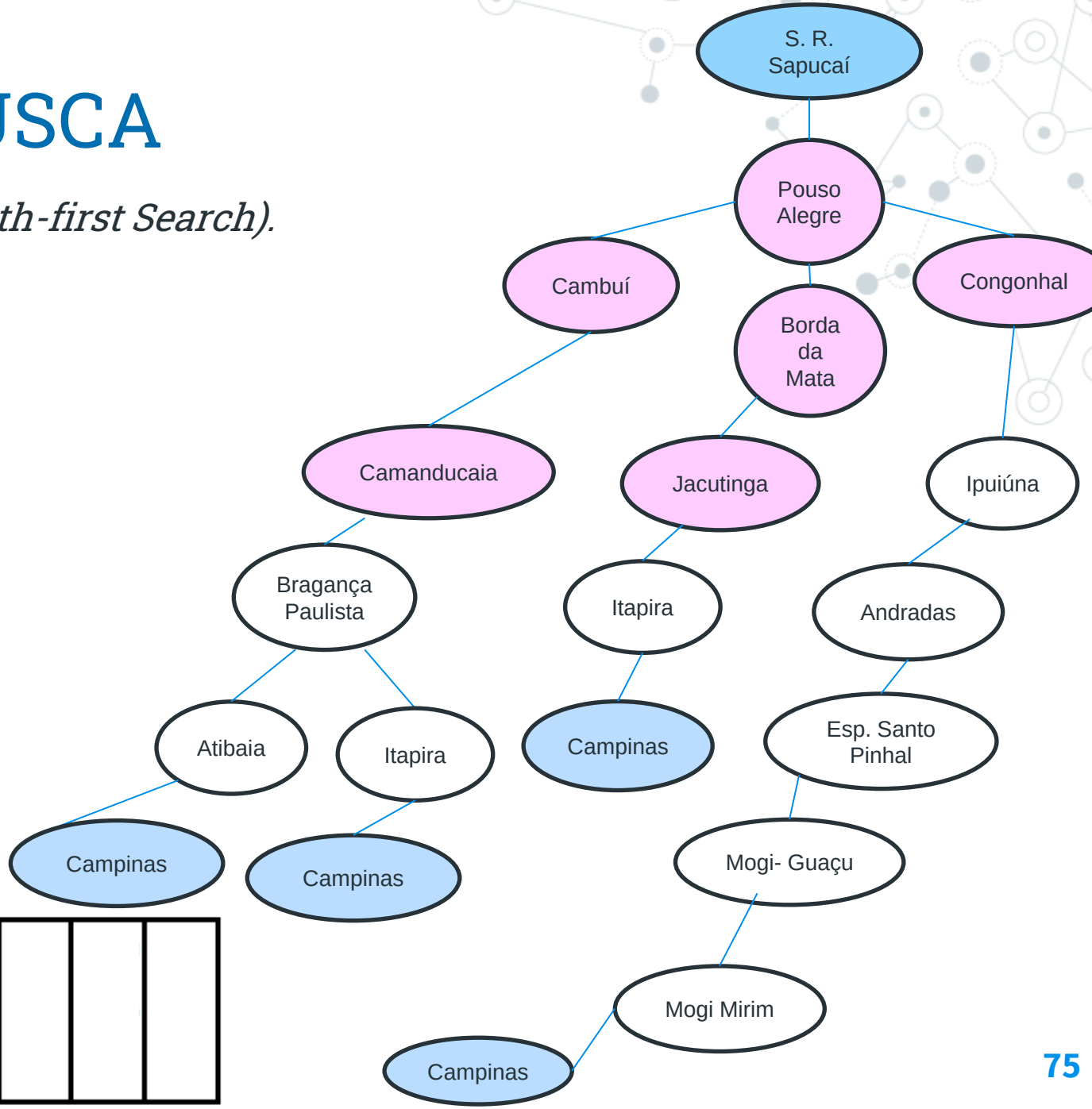


Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



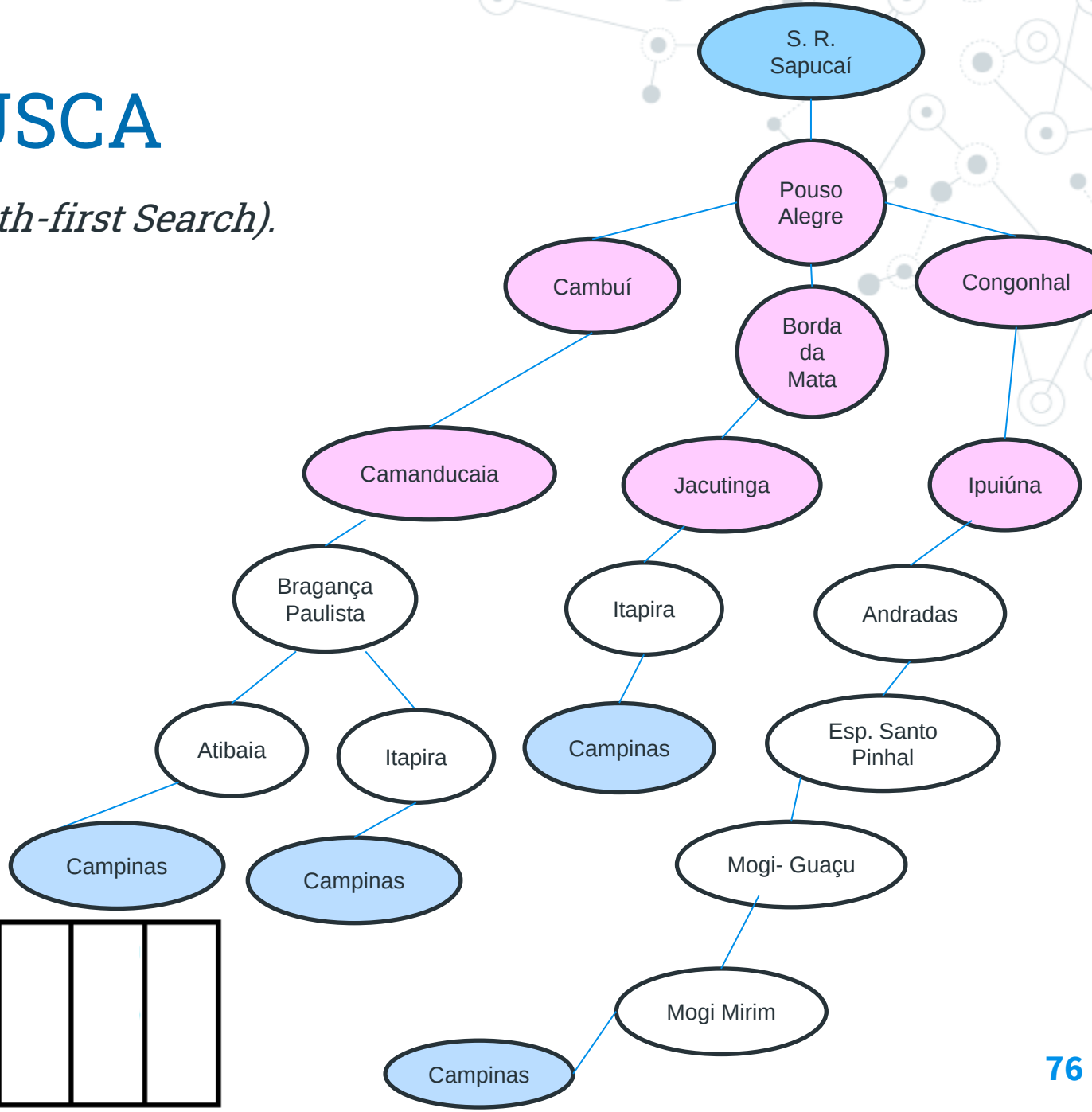
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga								

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



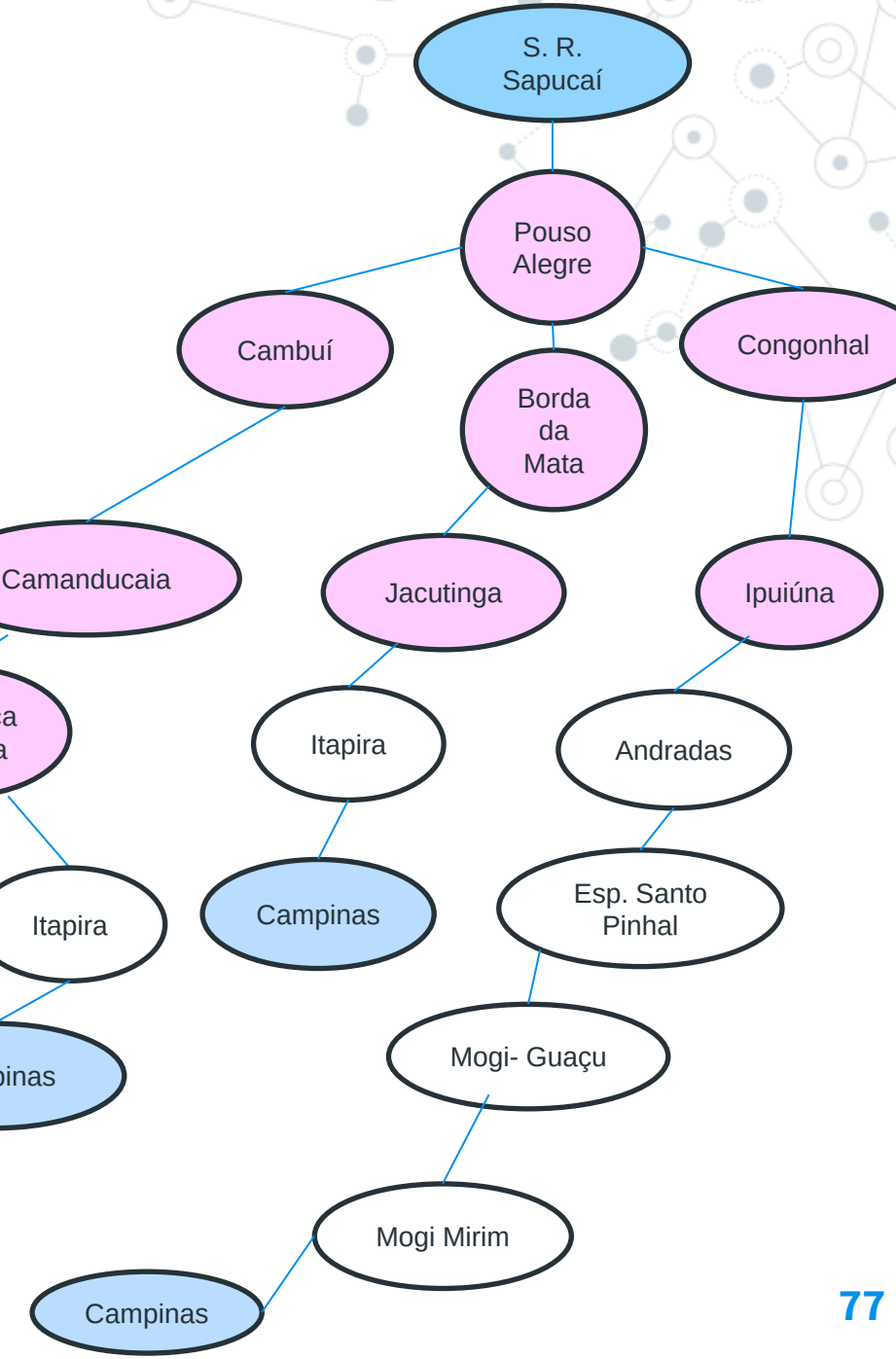
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna							

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



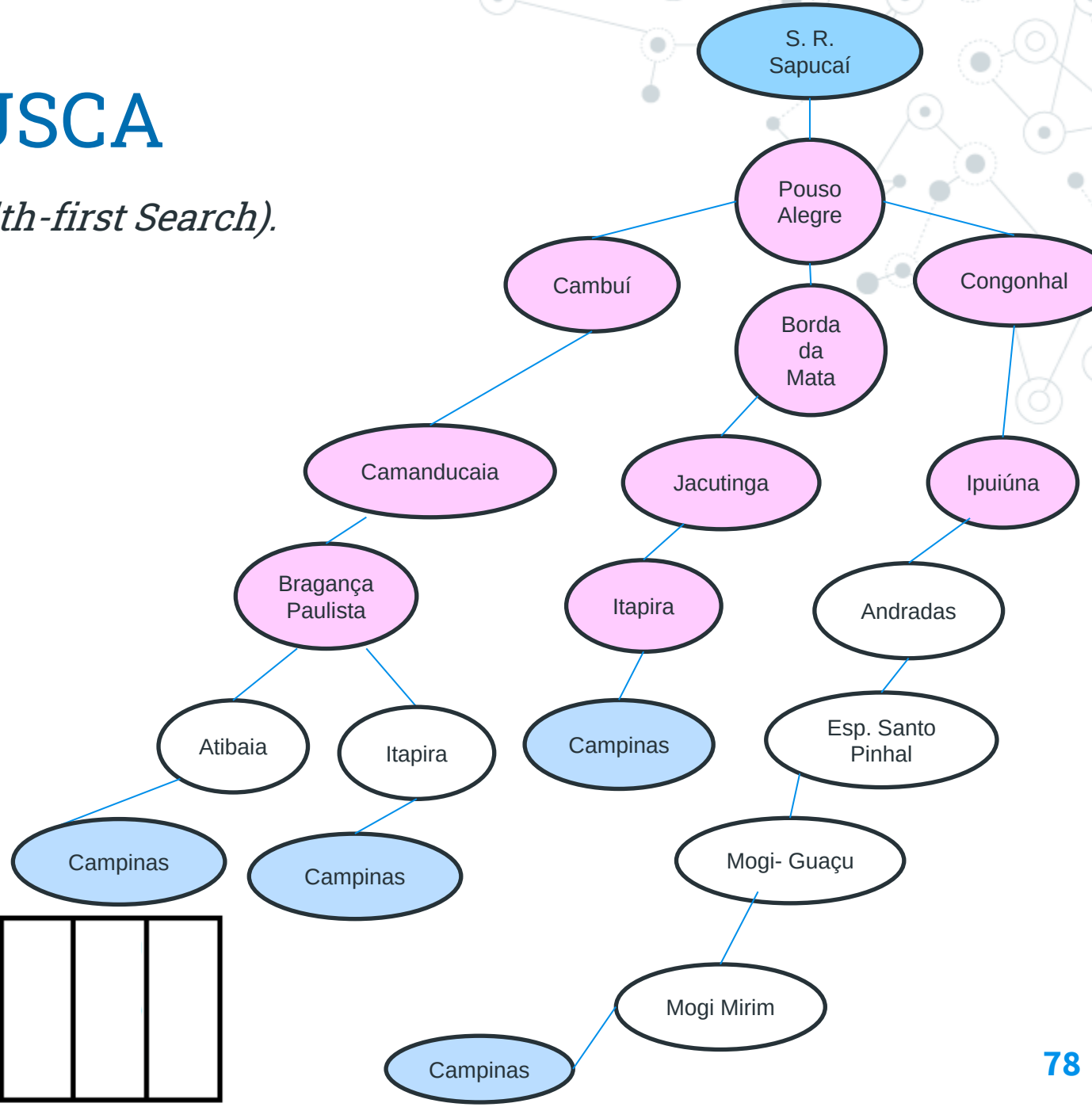
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna	Bragança Paulista						

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



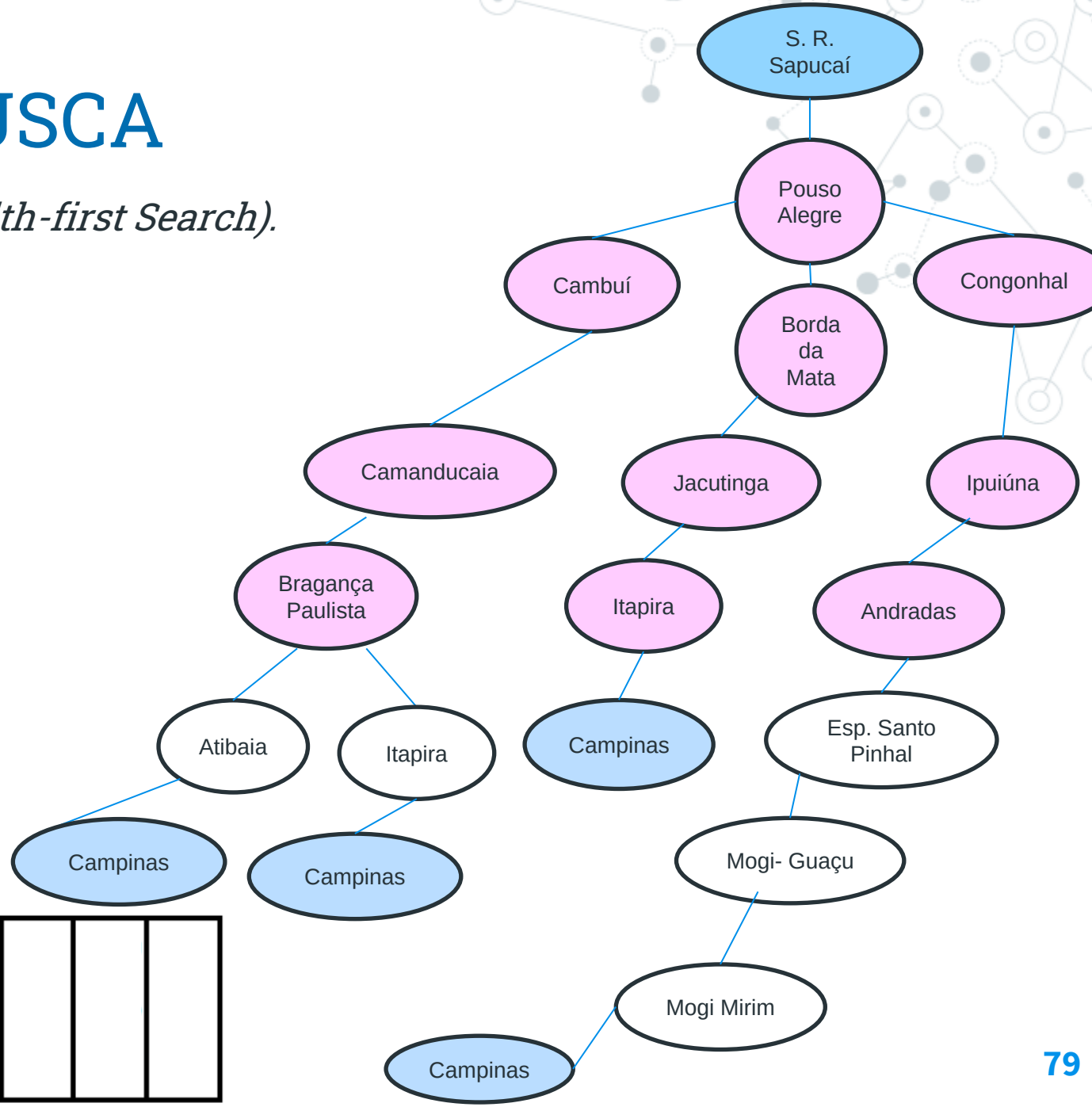
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna	Bragança Paulista	Itapira					

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



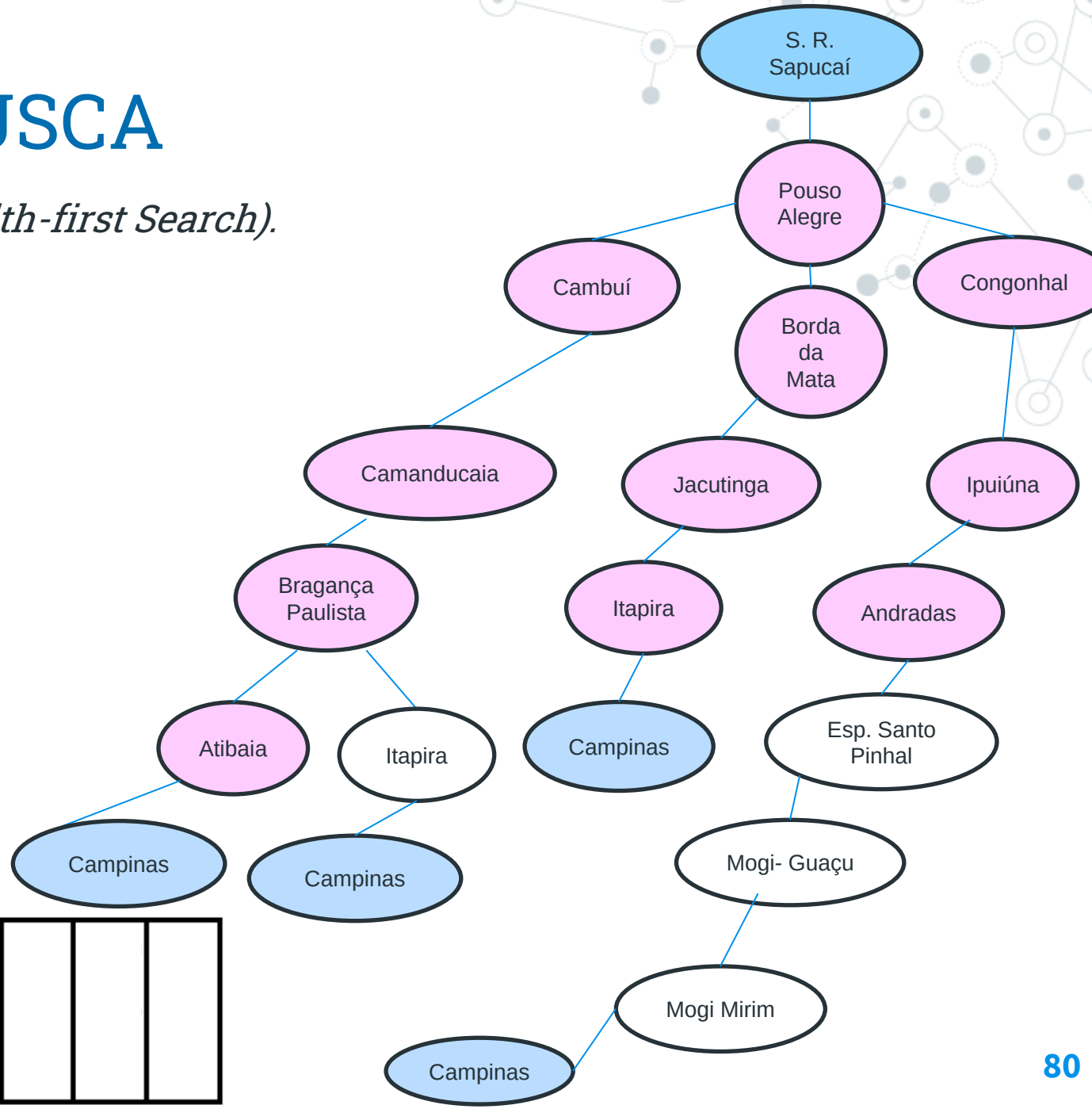
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna	Bragança Paulista	Itapira	Andradas				

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



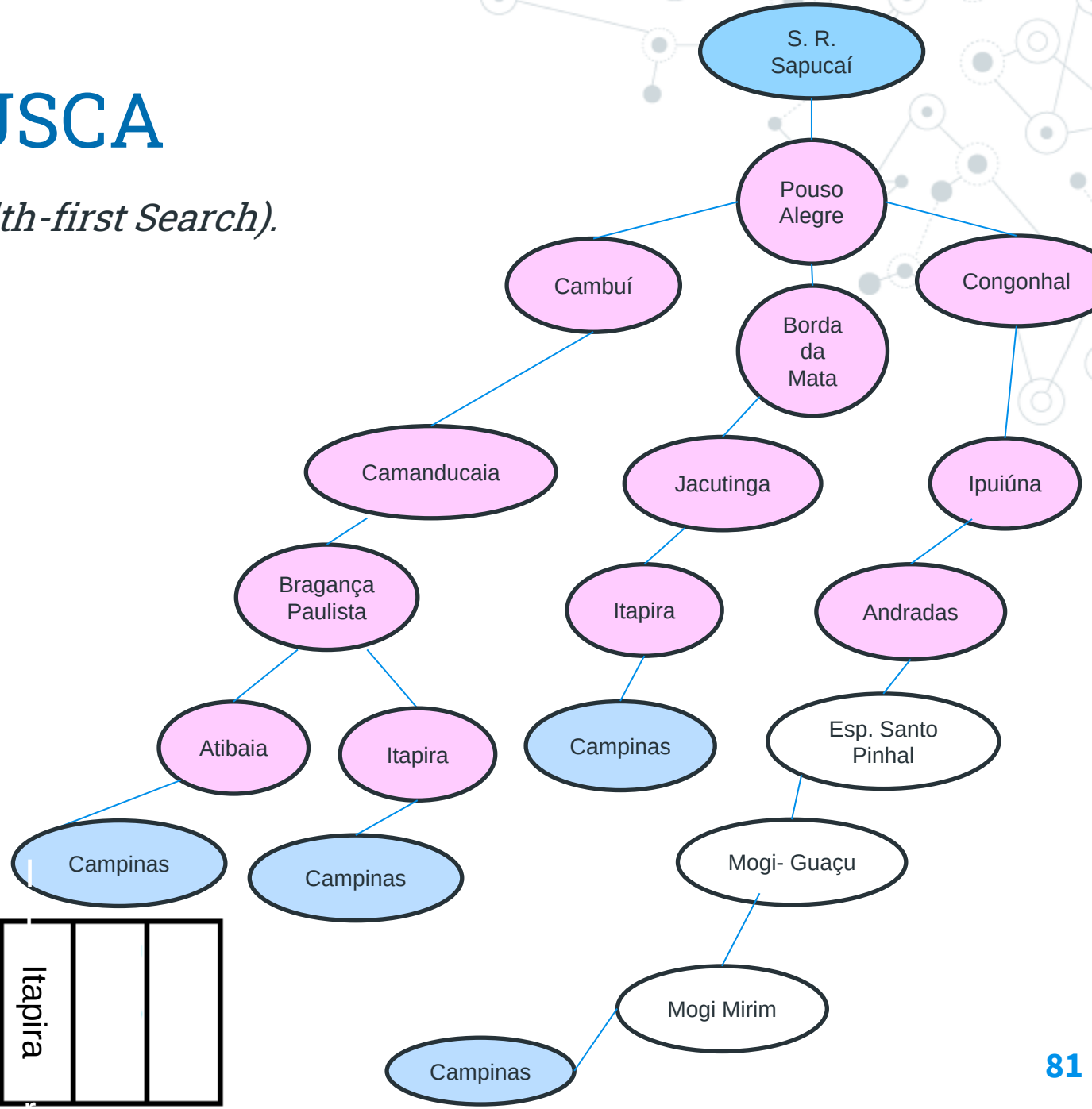
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna	Bragança Paulista	Itapira	Andradas	Atibaia			

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



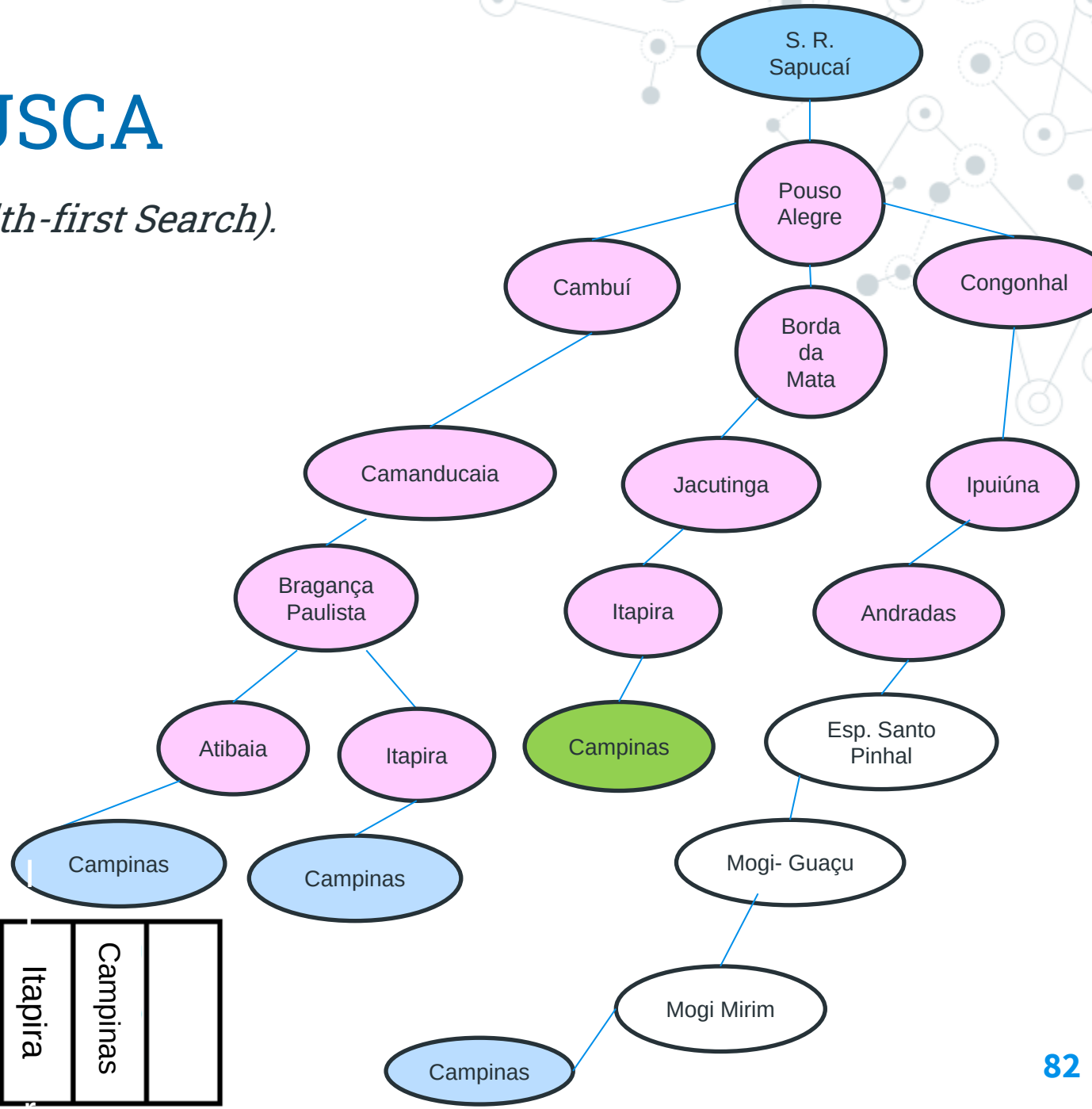
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna	Bragança Paulista	Itapira	Andradas	Atibaia	Itapira		

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



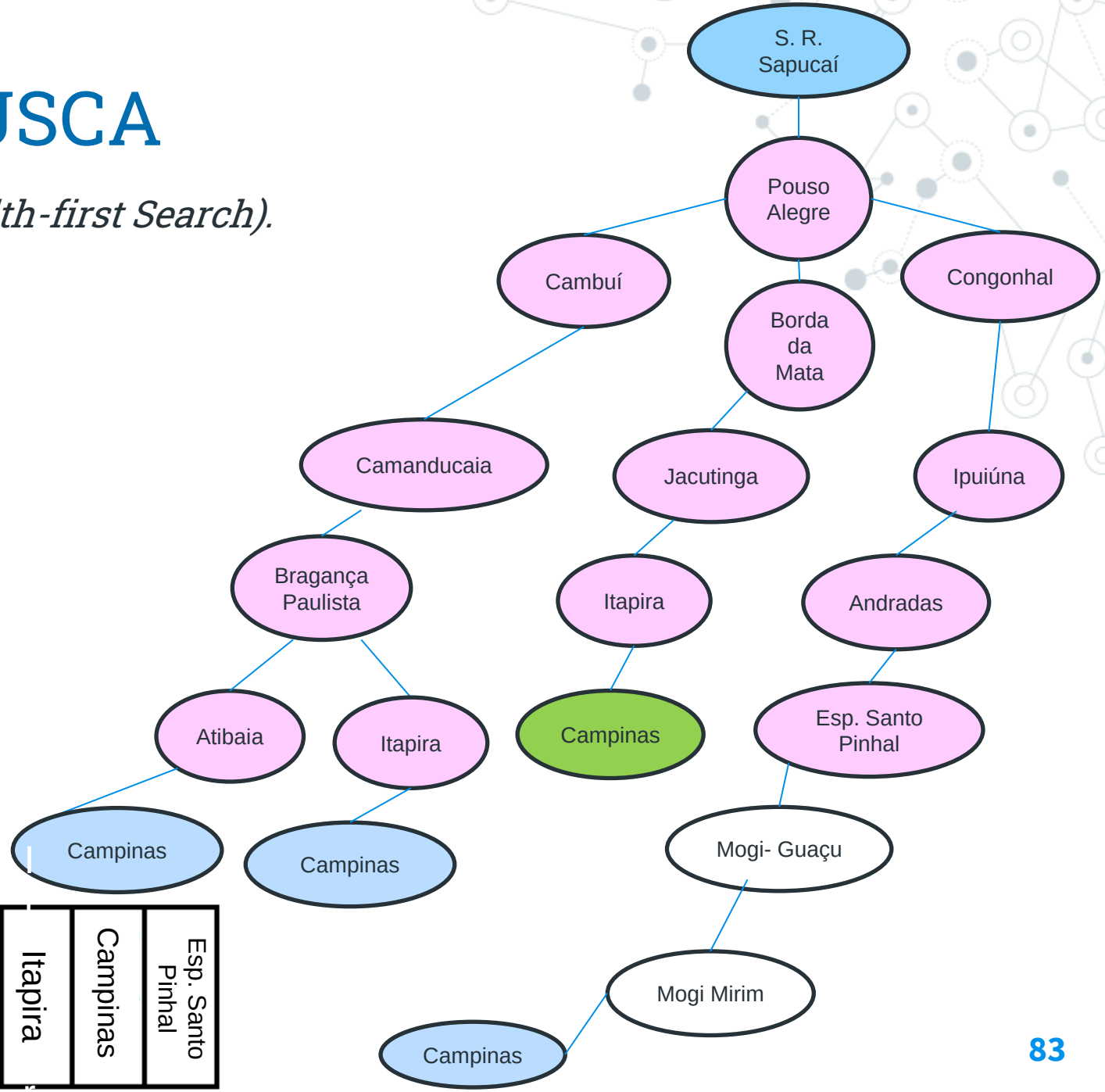
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna	Bragança Paulista	Itapira	Andradas	Atibaia	Itapira	Campinas	

Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas



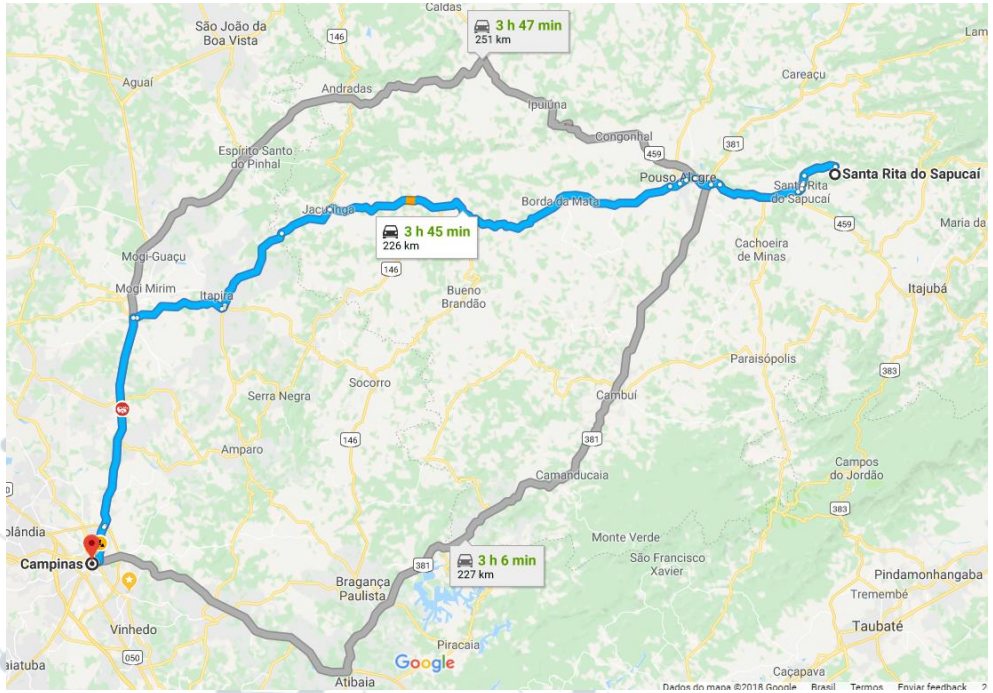
S. R. Sapucaí	Pouso Alegre	Cambuí	Borda da Mata	Congonhal	Camanducaia	Jacutinga	Ipuiúna	Bragança Paulista	Itapira	Andradas	Atibaia	Itapira	Campinas	Esp. Santo Pinhal

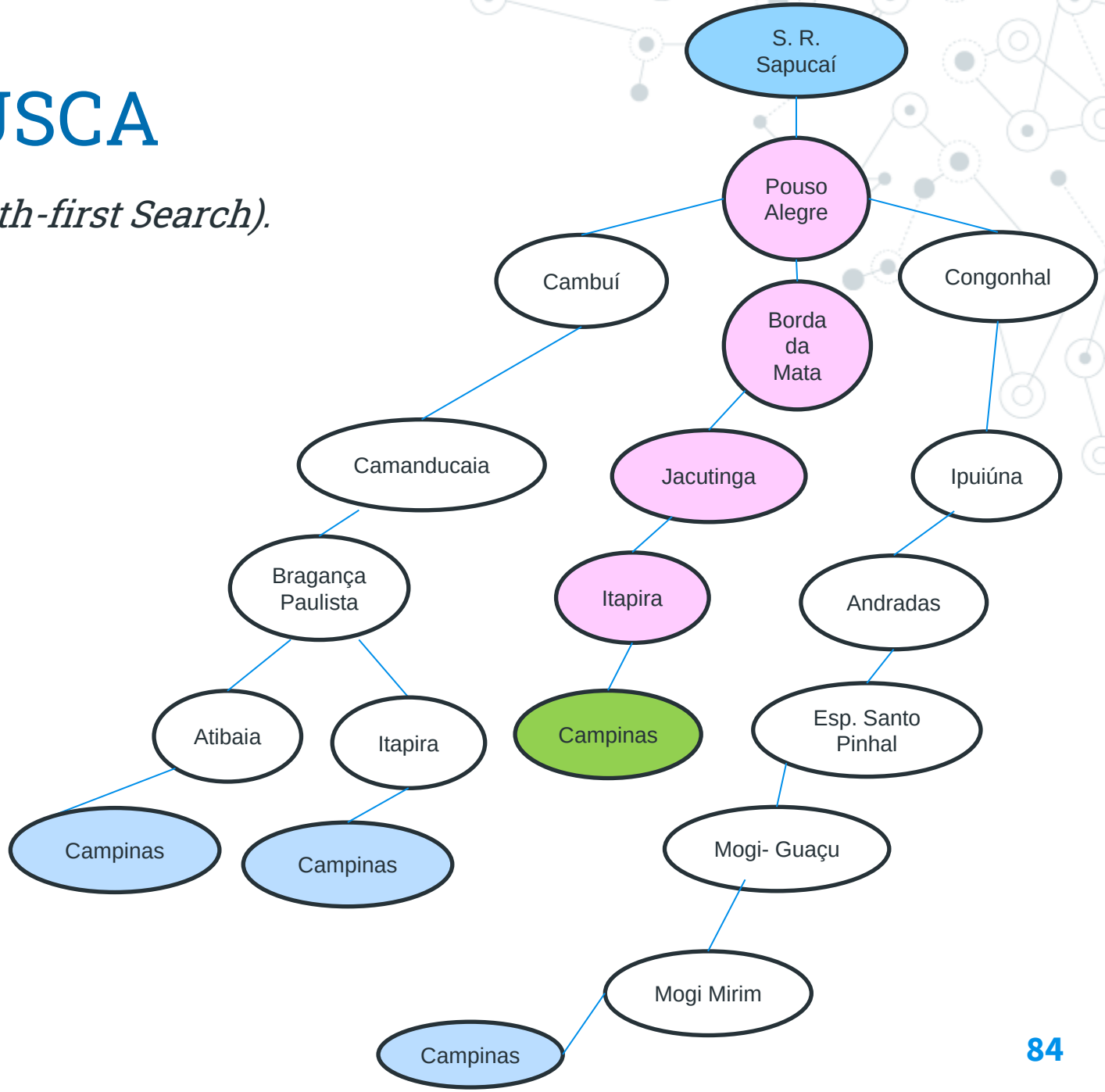
Estratégias de Busca

- **Busca em Largura** *(BrFS – Breadth-first Search).*

Problema de Roteamento:

- **Origem:** Santa Rita do Sapucaí
- **Destino:** Campinas





Estratégias de Busca

- **Busca em Profundidade** *(DFS – Depth-first Search)*
  - A busca em profundidade sempre expande o nó mais profundo na borda atual da árvore de busca. A busca prossegue imediatamente até o nível mais profundo da árvore de busca, onde os nós não têm sucessores. À medida que esses nós são expandidos, eles são retirados da borda e, então, a busca "retorna" ao nó seguinte mais profundo que ainda tem sucessores inexplorados.
  - Procurar explorar completamente cada ramo da árvore antes de tentar o ramo vizinho.
  - *O que acontece quando nenhuma regra pode ser aplicada, ou a árvore atinge uma profundidade muito grande sem que tenha encontrado uma solução?*
    - Neste caso ocorre o **BACKTRACKING**, ou seja, o algoritmo "volta atrás" e tenta outro caminho.

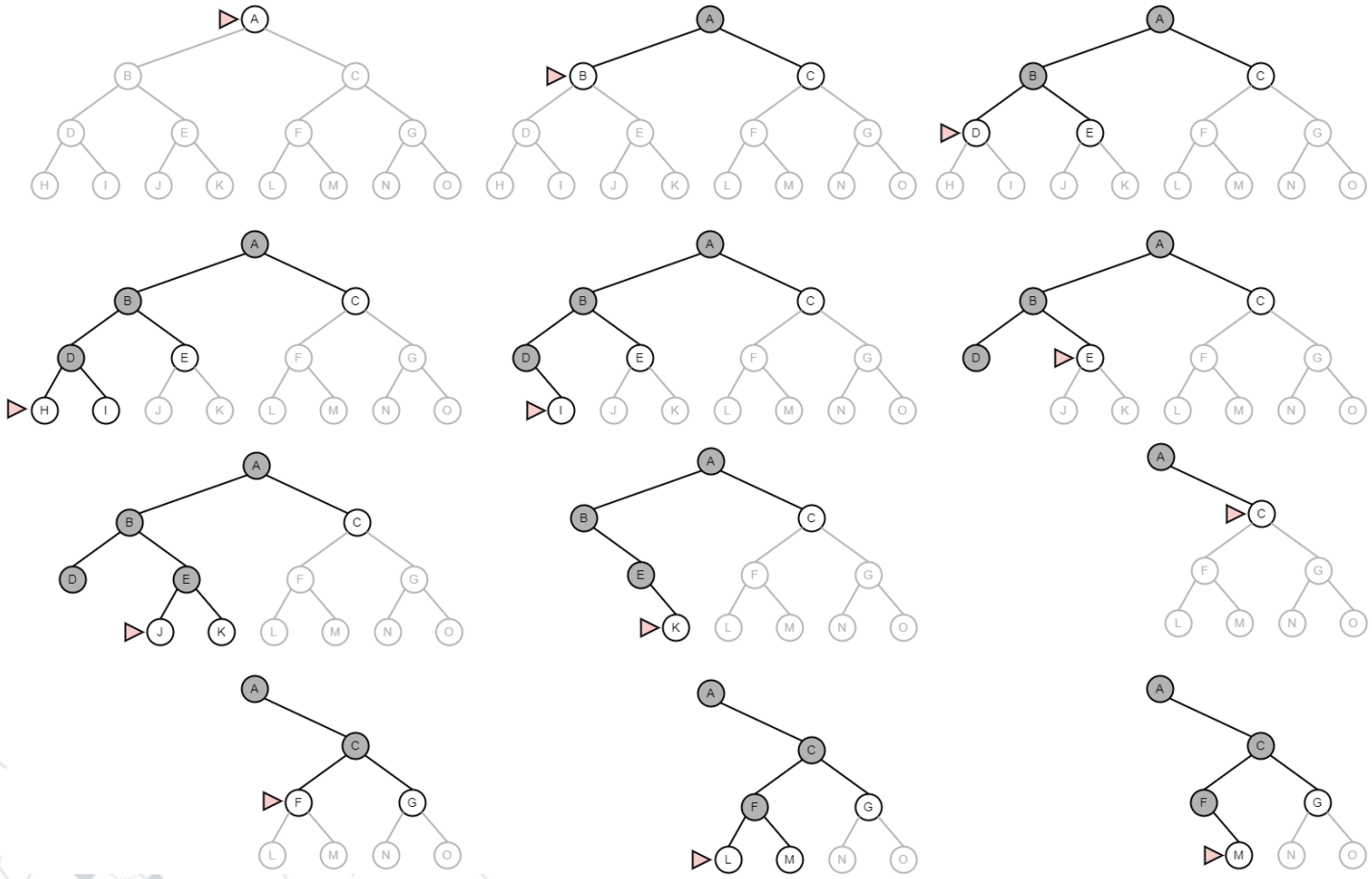
Estratégias de Busca

- **Busca em Profundidade**

  - A busca em profundidade sempre expande o nó mais profundo na borda atual da árvore de busca.
  - A busca prossegue imediatamente até o nível mais profundo da árvore de busca, onde os nós não têm sucessores.
  - À medida que esses nós são expandidos, eles são retirados da borda e, então, a busca "retorna" ao nó seguinte mais profundo que ainda tem sucessores inexplorados.

Estratégias de Busca

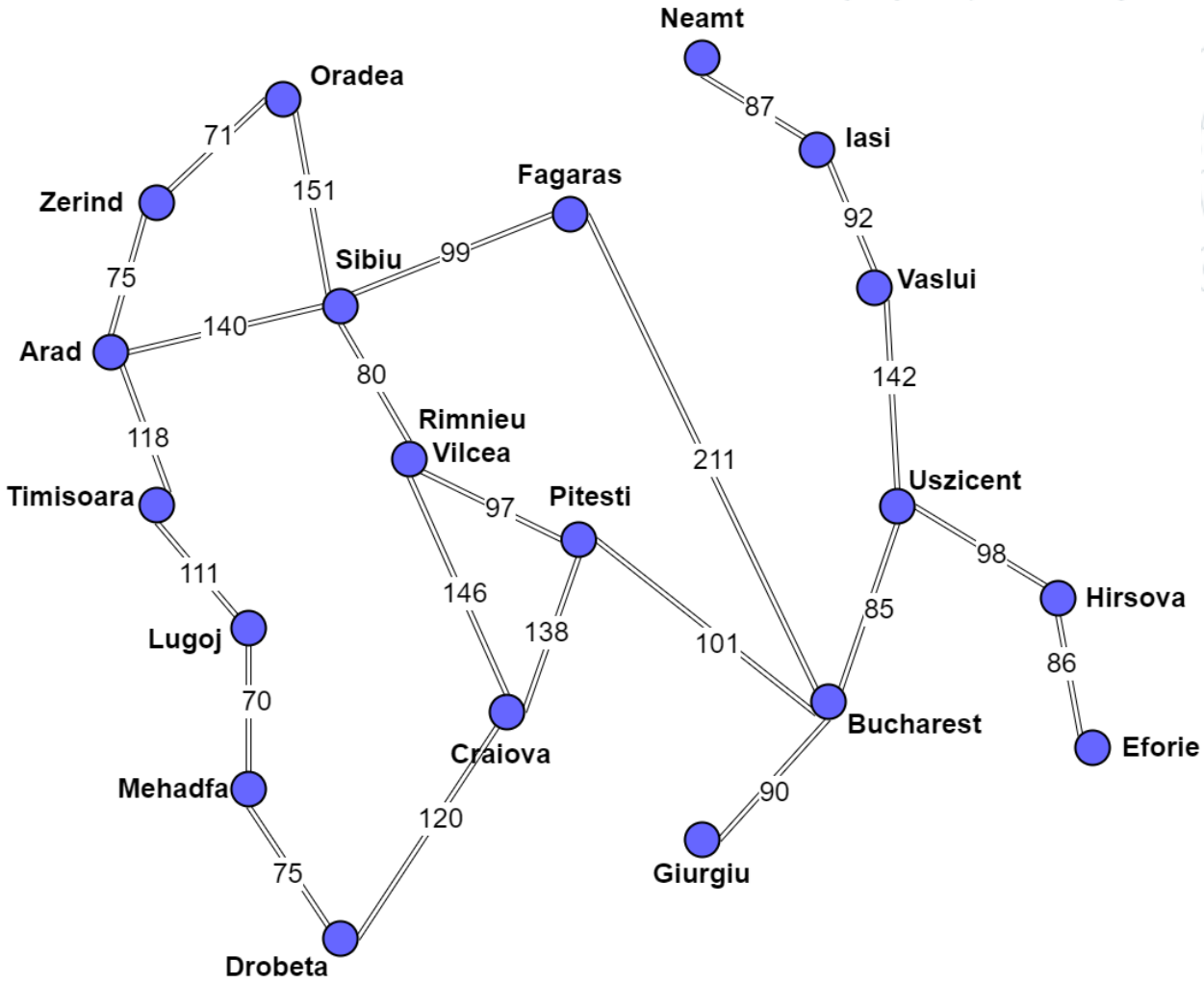
- **Busca em Profundidade em uma Árvore Binária**



- A região inexplorada é mostrada em cinza claro.
- Os nós explorados sem descendentes na borda são removidos da memória.
- Os nós na profundidade 3 não têm sucessores e M é o único nó objetivo.

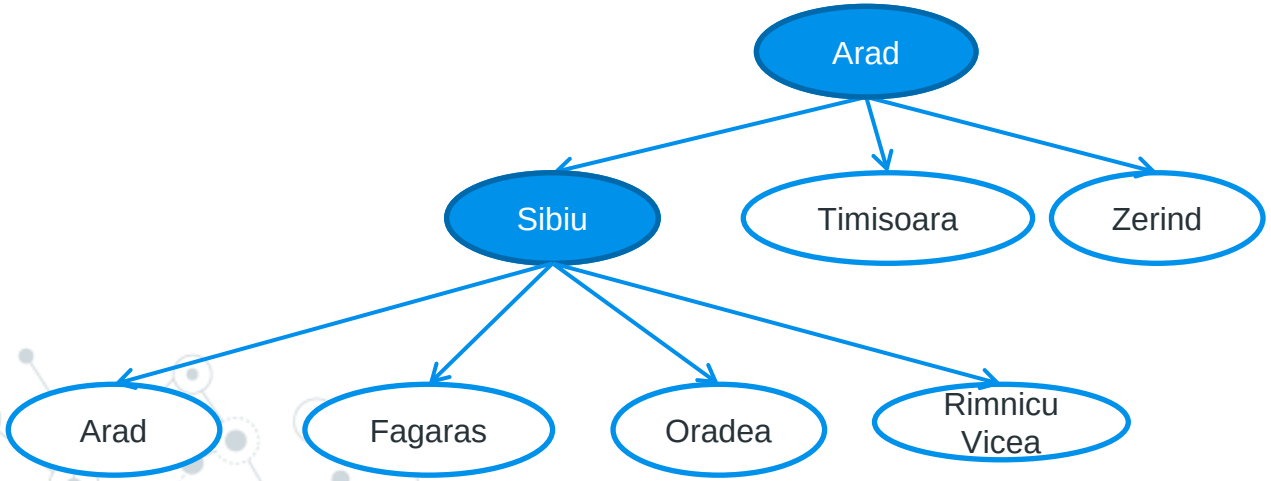
Estratégias de Busca

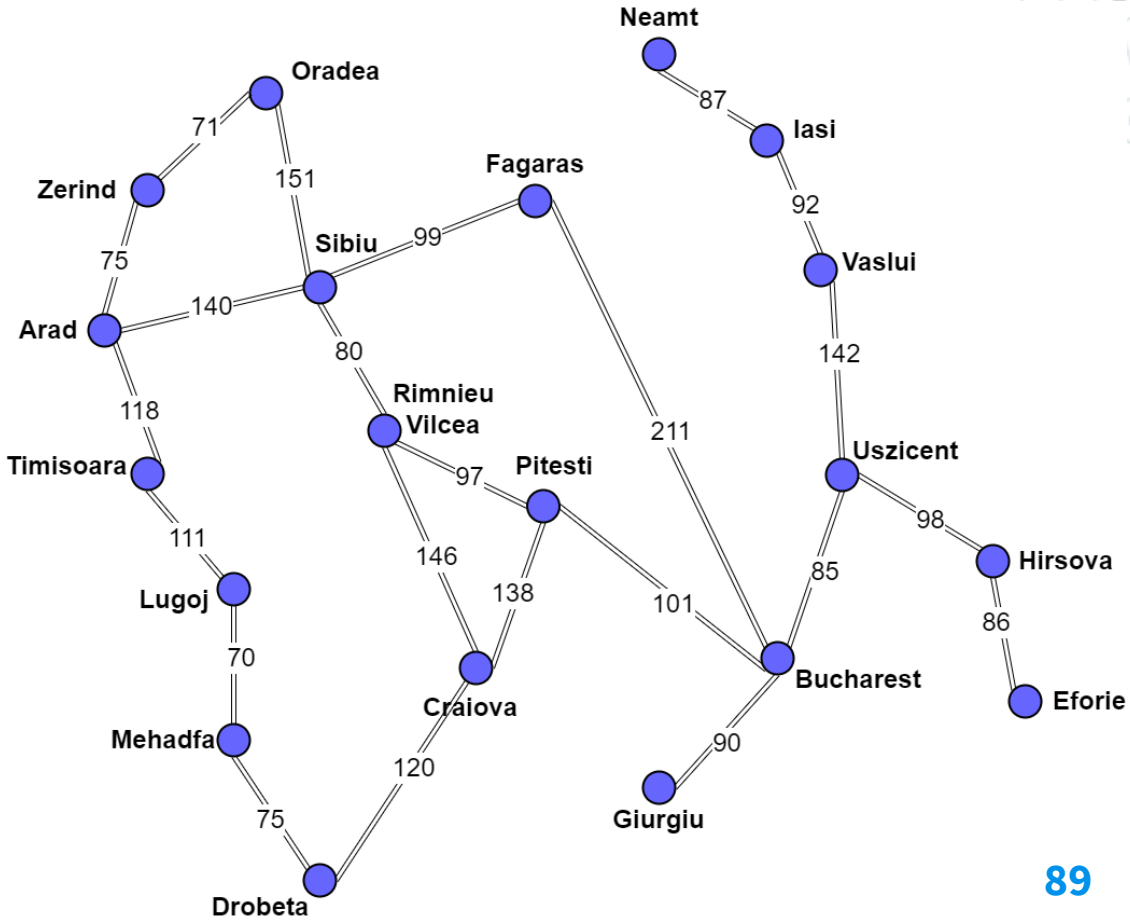
- **Busca em Profundidade**
  - **Exemplo:** Mapa rodoviário de parte da Romênia.



Estratégias de Busca

- **Busca em Profundidade**
  - **Exemplo:** Mapa rodoviário de parte da Romênia.





Estratégias de Busca

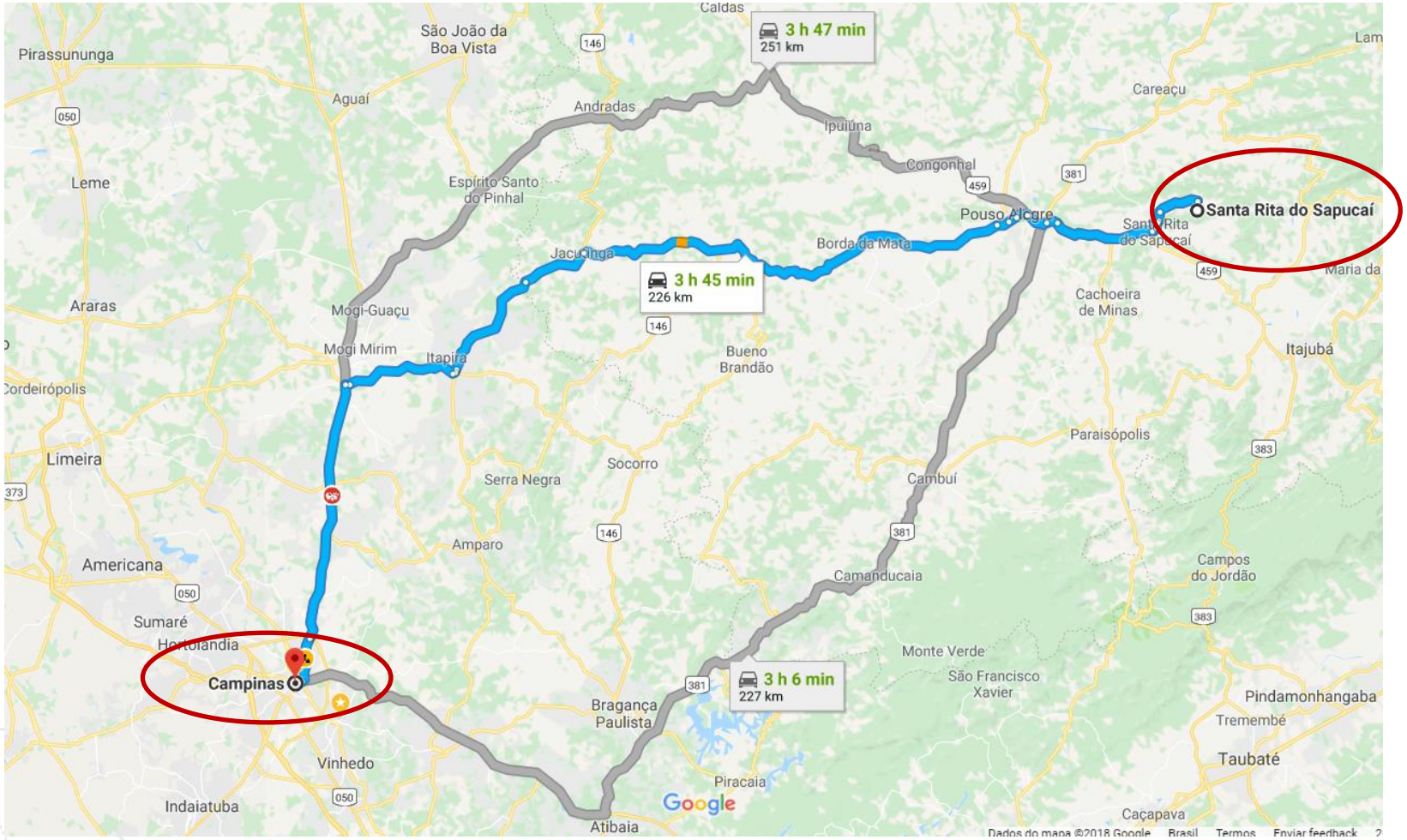
- **Busca em Profundidade**

Características: Não é Completa* e Não é Ótima

- Realizando uma busca em árvore onde um nó já visitado pode ser visitado novamente, poderia gerar um laço infinito, como no caso Arad-Sibiu-Arad;
- Se admitir estados repetidos ou um nível máximo de profundidade, pode **nunca encontrar a solução.**
- O algoritmo **não encontra necessariamente a solução ótima**, mas pode ser **MAIS EFICIENTE** se o problema possui um grande número de soluções ou se a maioria dos caminhos pode levar a uma solução.

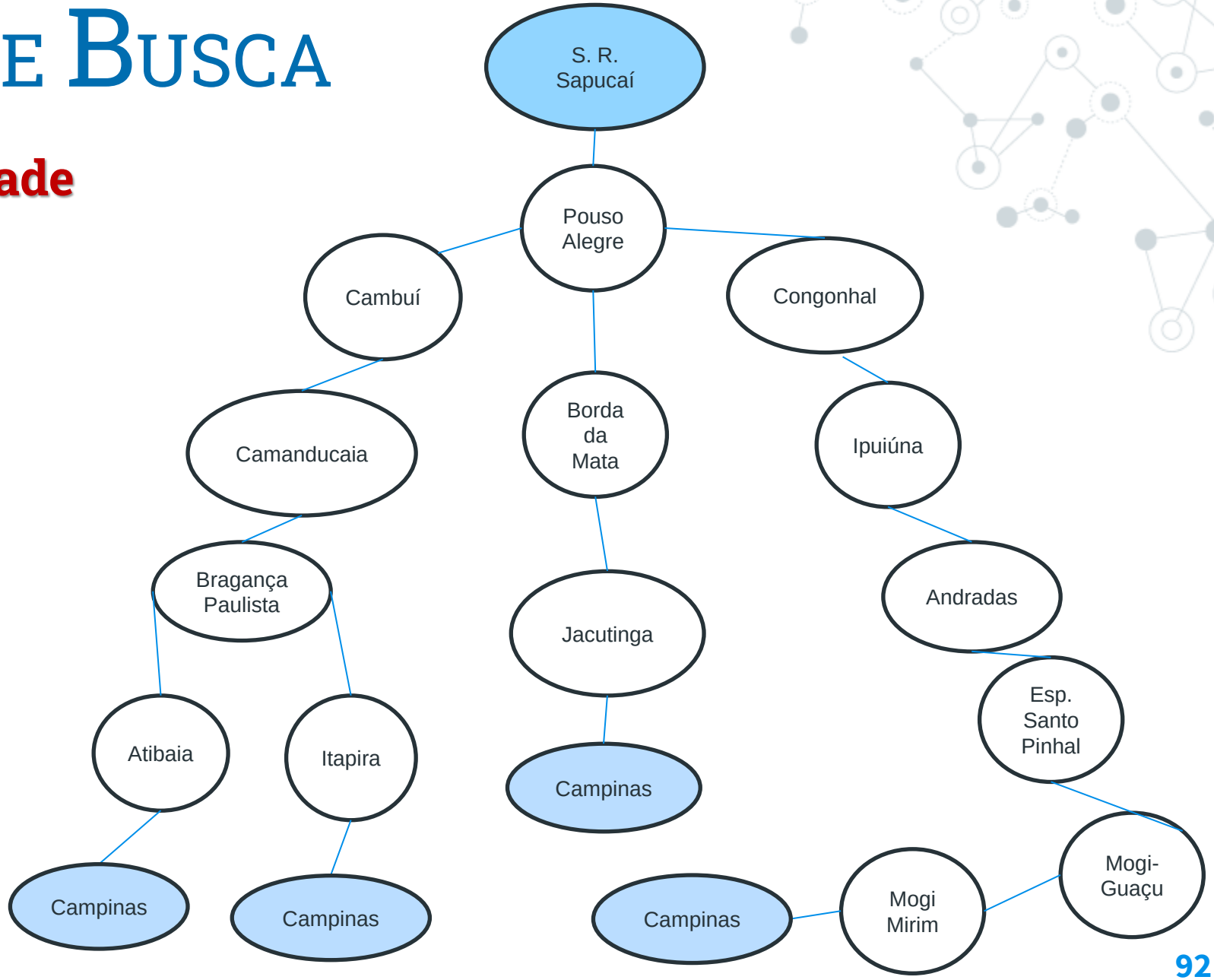
Estratégias de Busca

- **Busca em Profundidade**



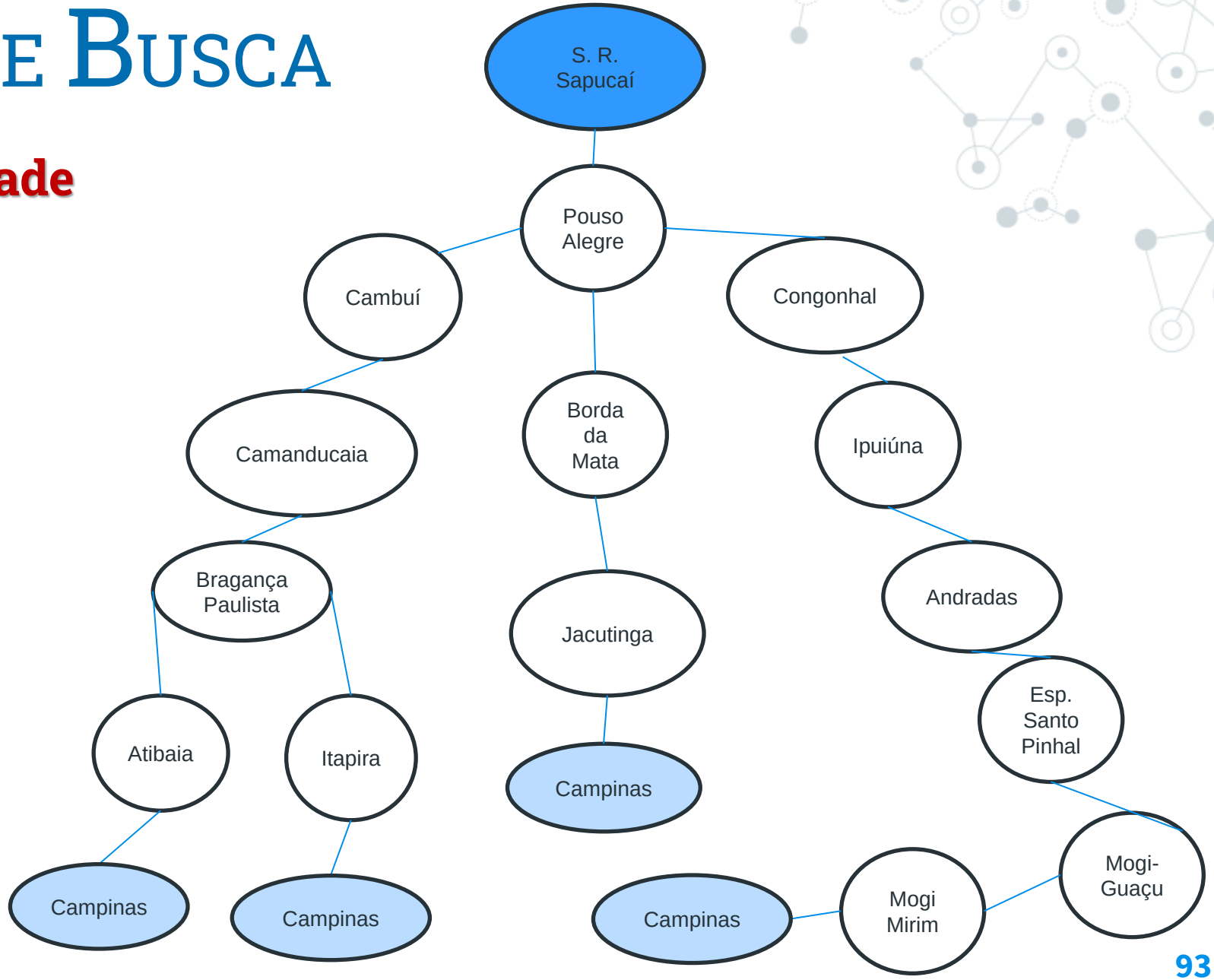
Estratégias de Busca

- **Busca em Profundidade**



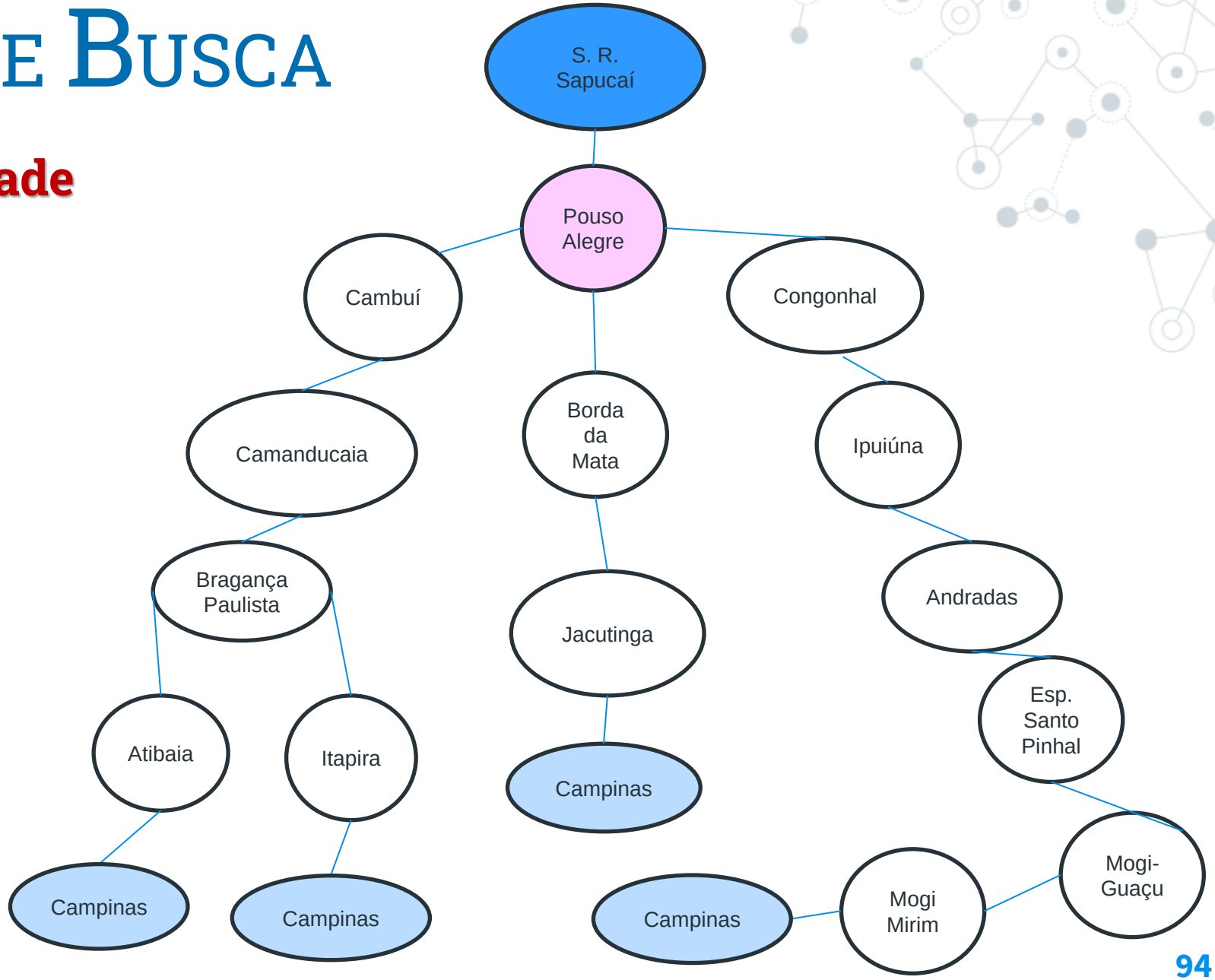
Estratégias de Busca

- **Busca em Profundidade**



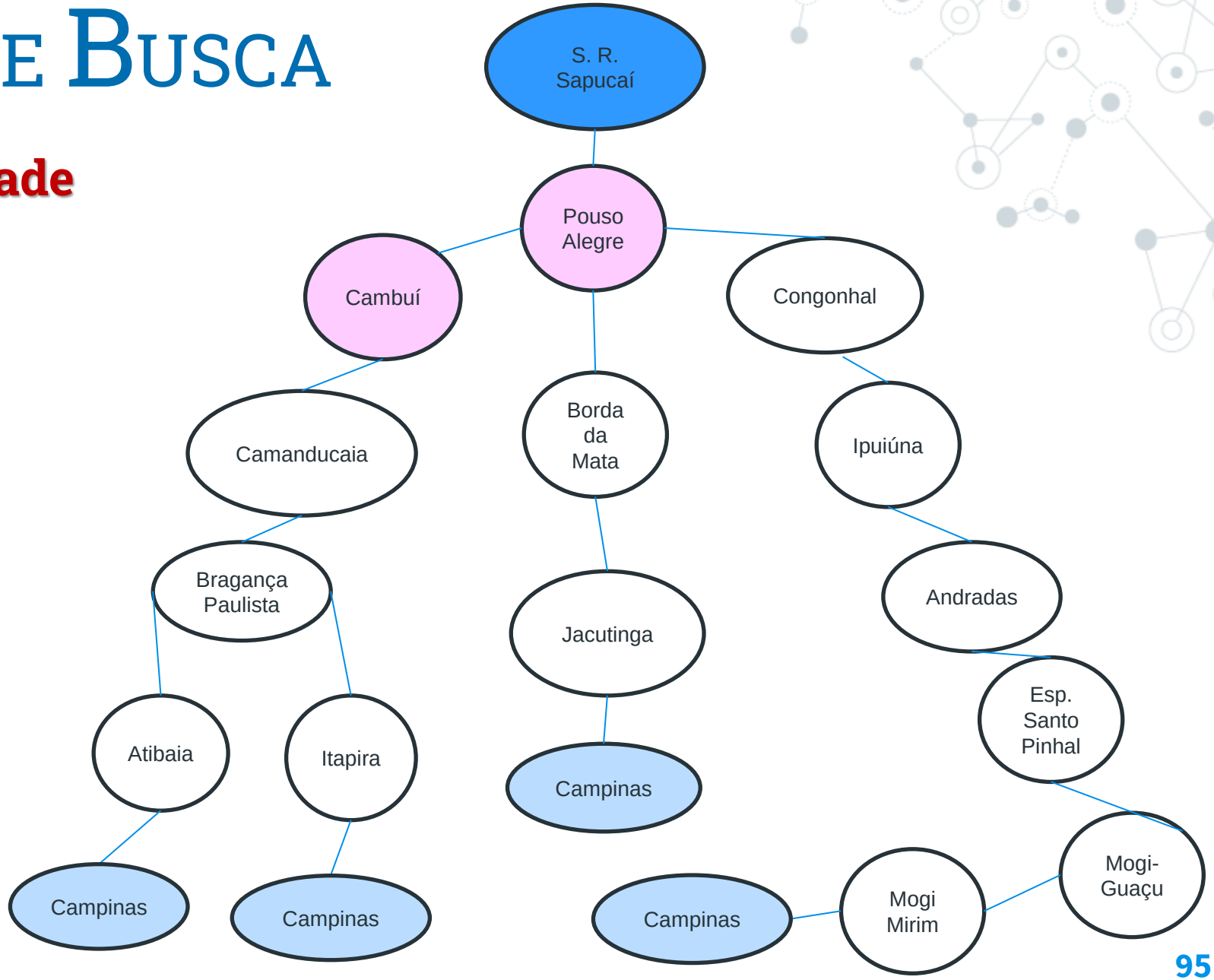
Estratégias de Busca

- **Busca em Profundidade**



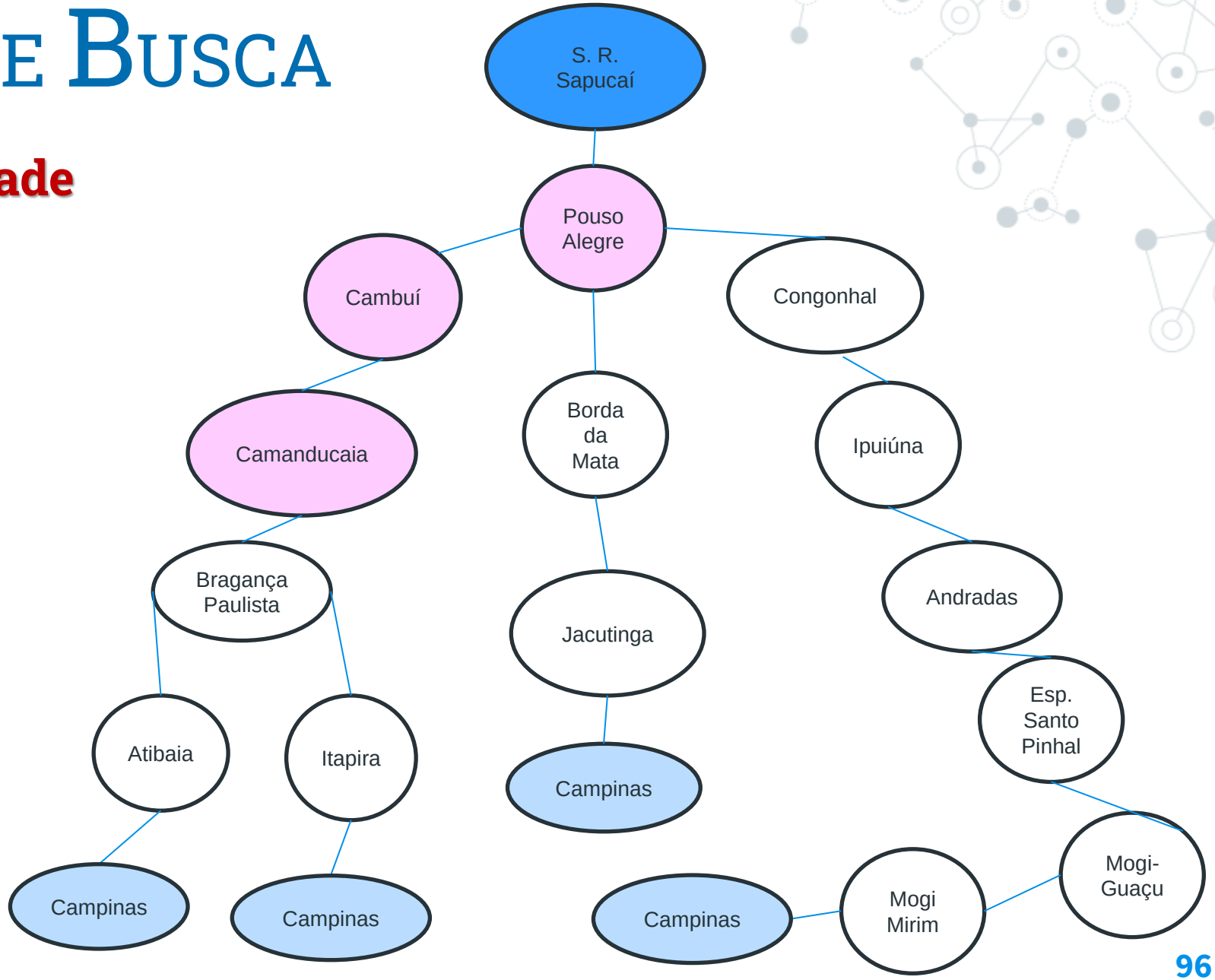
Estratégias de Busca

- **Busca em Profundidade**



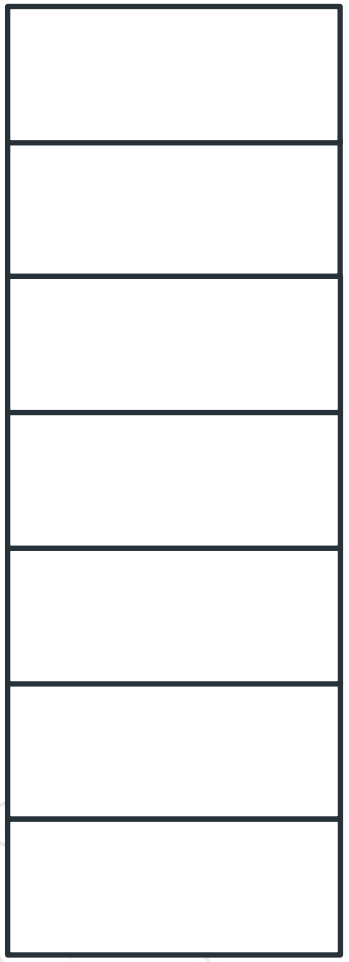
Estratégias de Busca

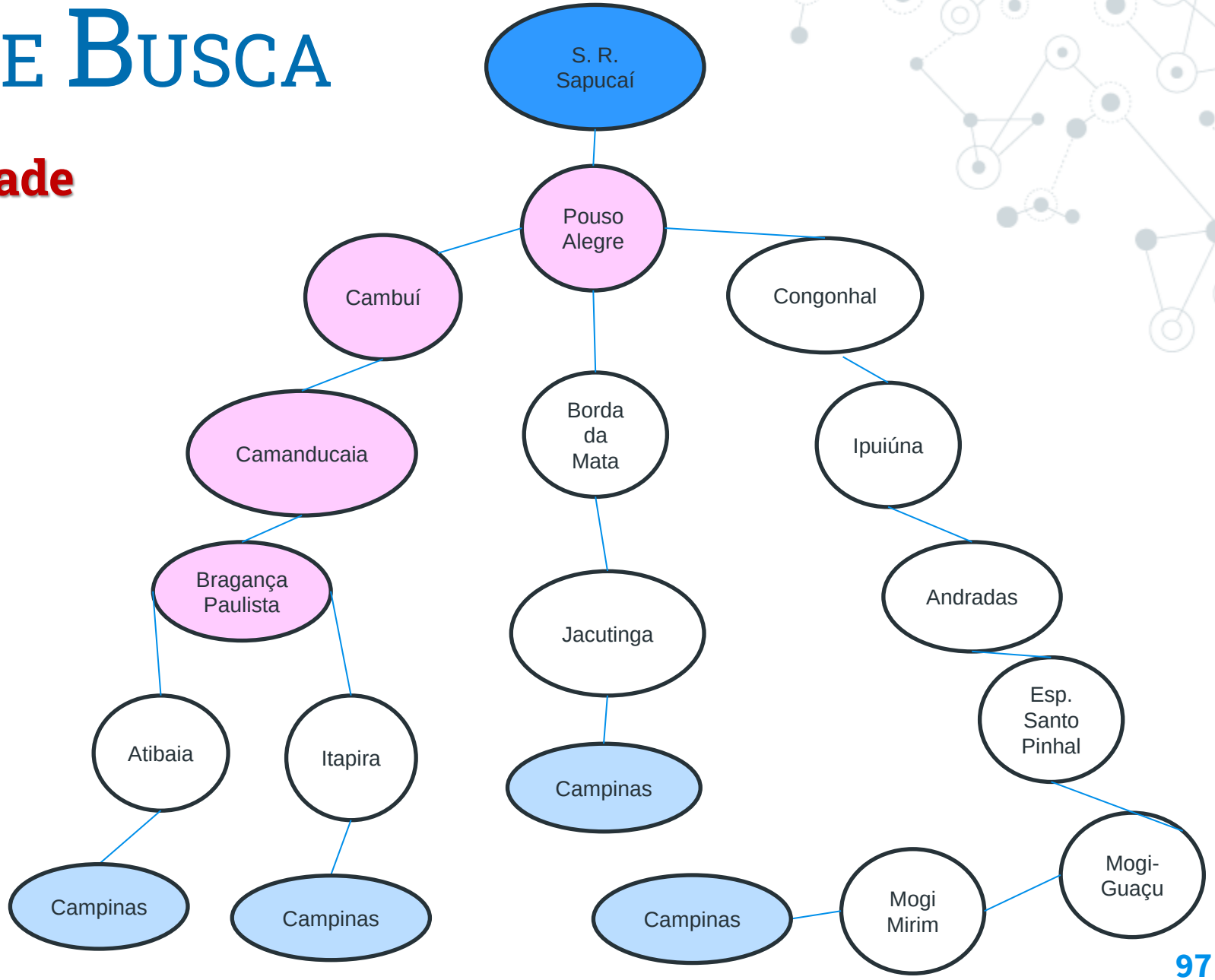
- **Busca em Profundidade**



Estratégias de Busca

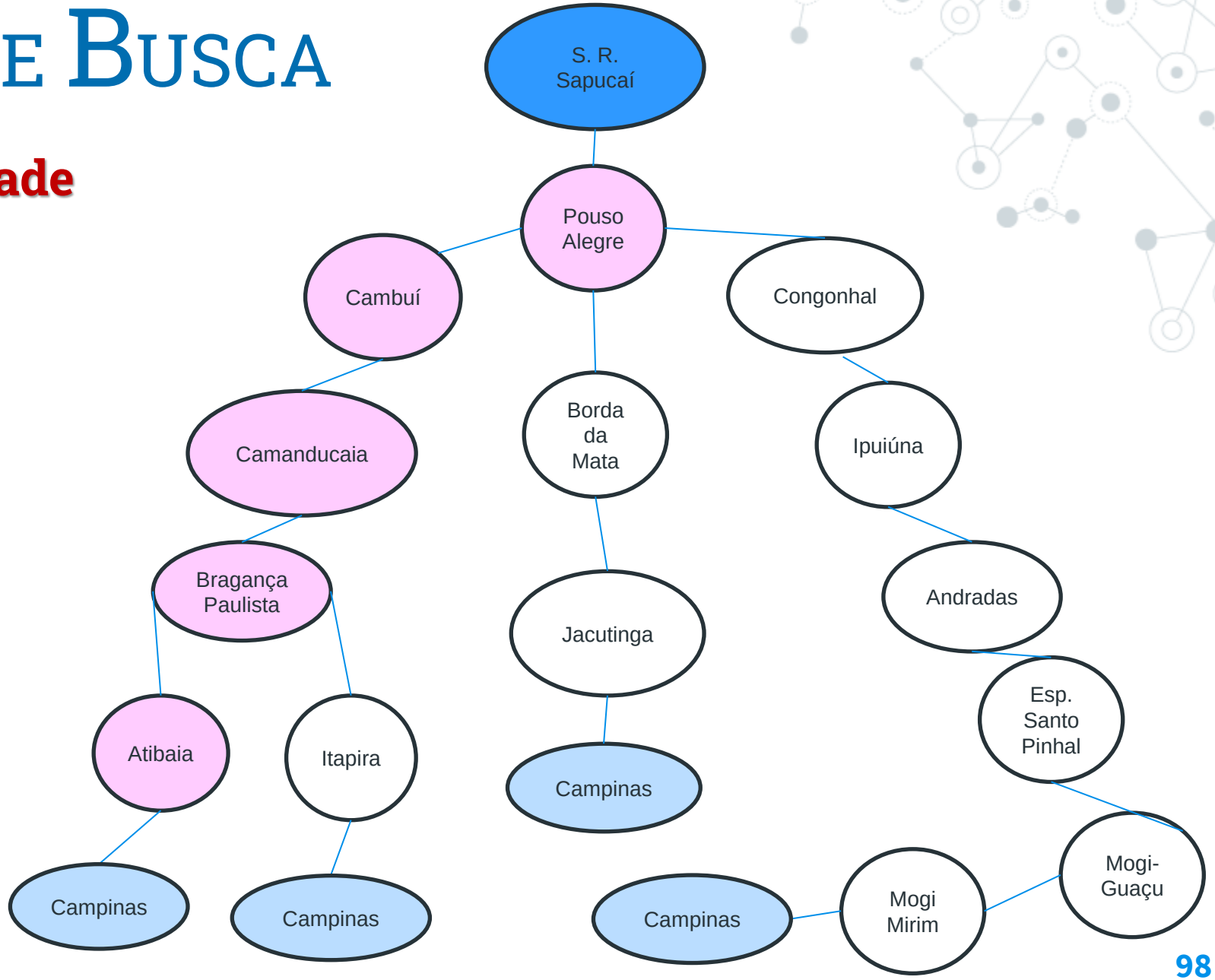
- **Busca em Profundidade**





Estratégias de Busca

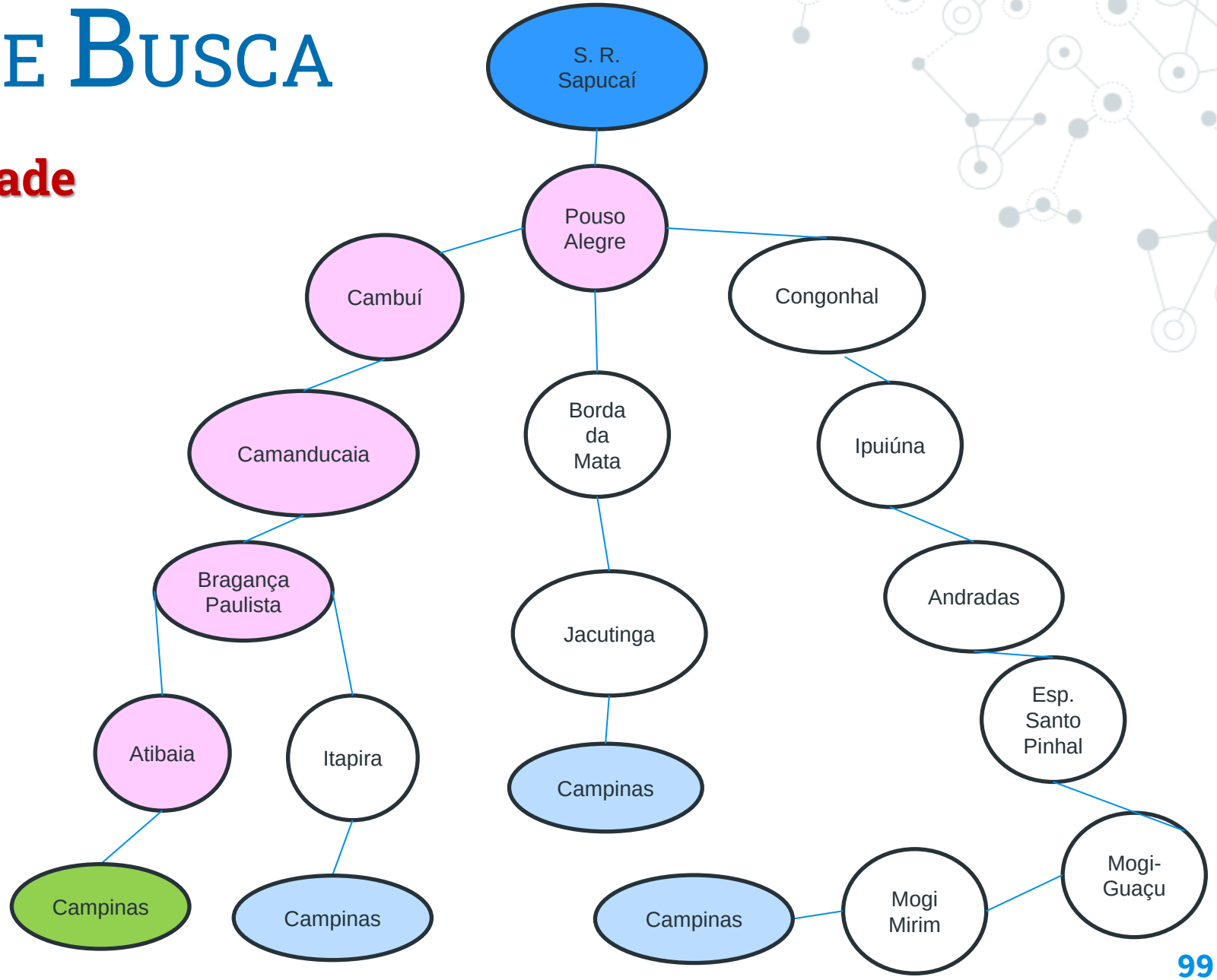
- **Busca em Profundidade**



Estratégias de Busca

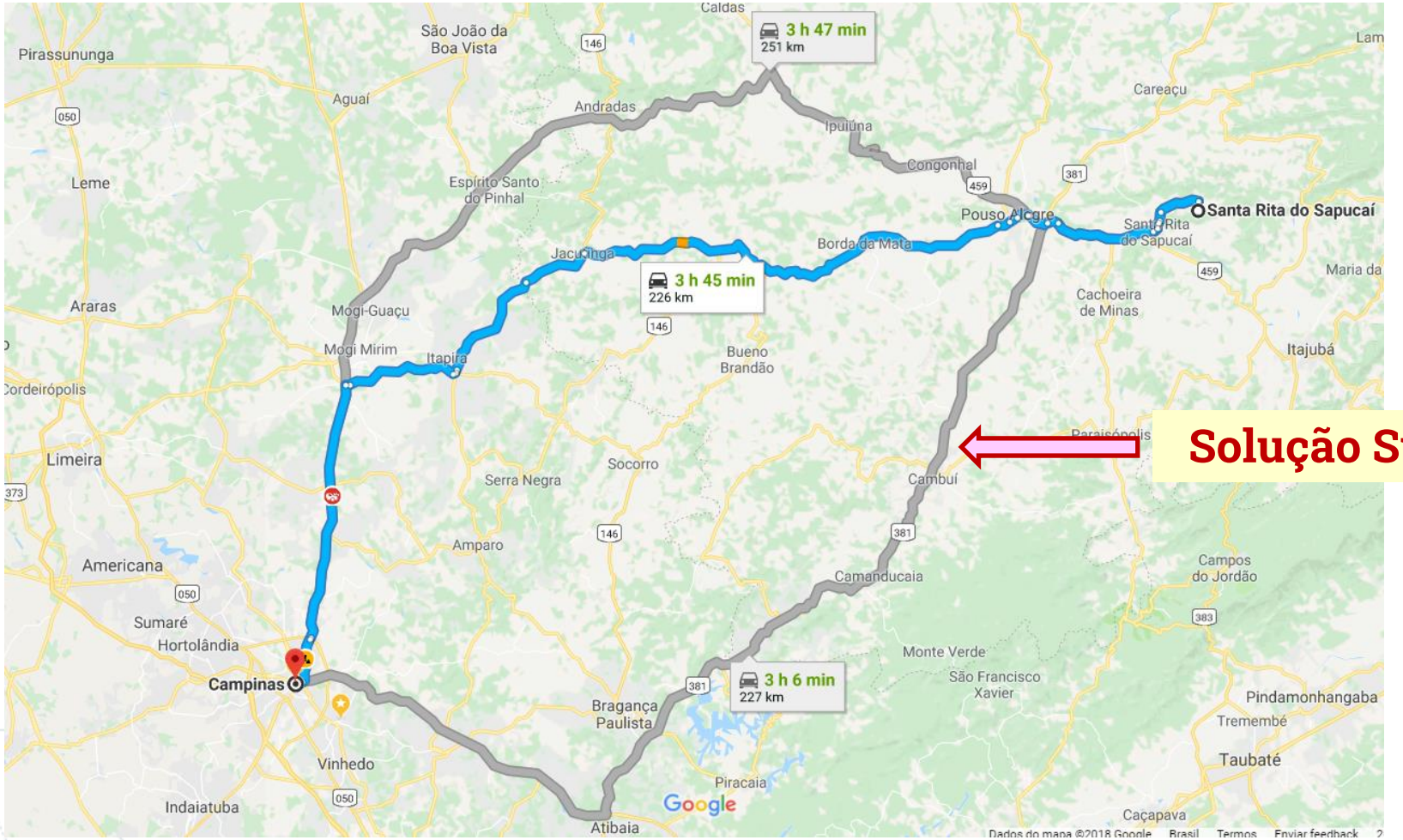
- **Busca em Profundidade**

Campinas
Atibaia
Bragança Paulista
Camanducaia
Cambuí
Pouso Alegre
S. R. Sapucaí



Estratégias de Busca

- **Busca em Profundidade**



Solução Subótima

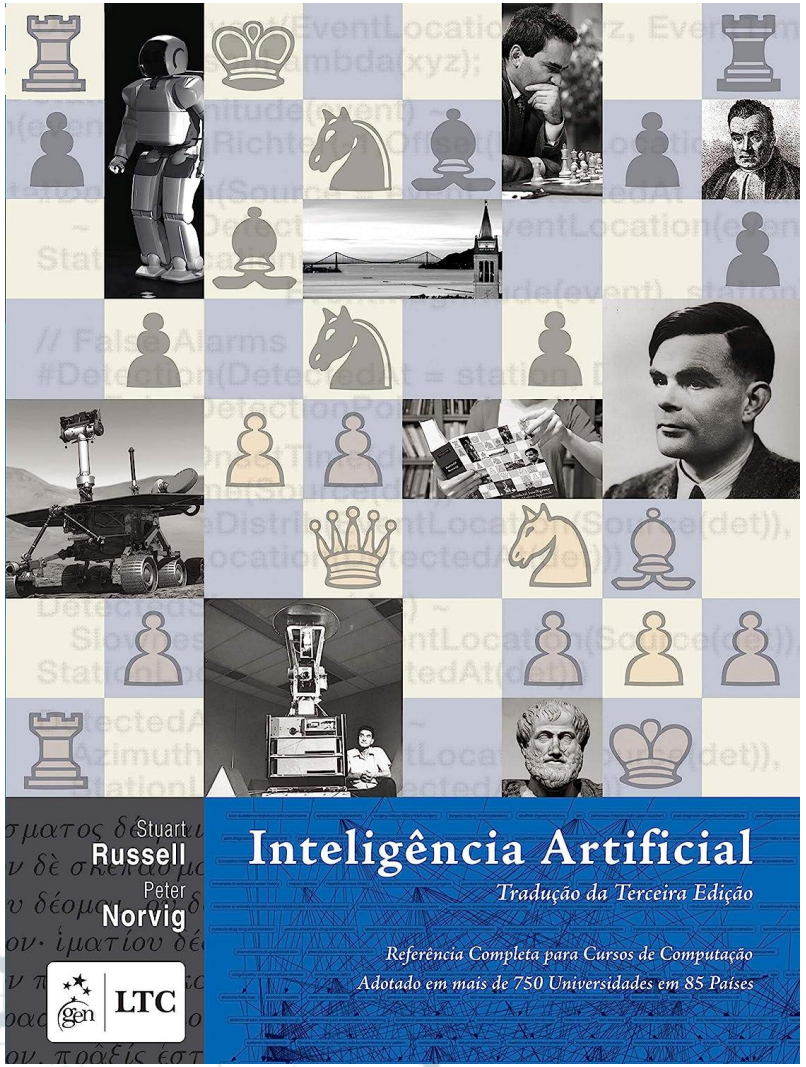
Estratégias de Busca

- **Busca em Profundidade**

Observações:

- Busca em profundidade utiliza menos memória porque armazena apenas um único caminho do nó raiz até o nó folha;
- Quanto ao tempo, a busca em profundidade é geralmente mais rápida.
- As buscas em largura e profundidade não fazem uso de nenhum conhecimento para encontrar sua solução, fazendo uma busca exaustiva dentro do seu espaço. Para contornar este problema, pode-se usar os **métodos heurísticos**.

Referências



- RUSSELL, Stuart; NORVIG, Peter (Peter Norvig); SOUZA, Vandenberg Dantas De, Inteligência artificial. Rio de Janeiro, RJ: Editora Campus, 2004 - 2013, ISBN 978-85-352-1177-1 / 978-85-352-3701-6.

→ Ler Capítulo 3

Obrigada pela Atenção!

Dúvidas?

victoria.souto@Inatel.br