Agentes de Procura:

Em contraste com agentes reativos, os Agentes de procura são mais sofisticados, possuindo maior memória e capacidade de simulação/antecipação.

- **Memória:** Possuem mais memória e Compreendem relações entre estados, permitindo uma visão global do problema.
- **Controlador:** Mais sofisticado que permite simular/antecipar que permite ao agente prever as consequências de ações.
  Enquanto não resolvido e tiver alternativas: analisa estado, determina regras aplicáveis, aplica de modo simulado todas as regras, escolhe um estado resultante
- **Restrições:** Operadores de mudança de estado que ajudam a determinar estados a evitar ou limitar.

Resolução de Problemas:

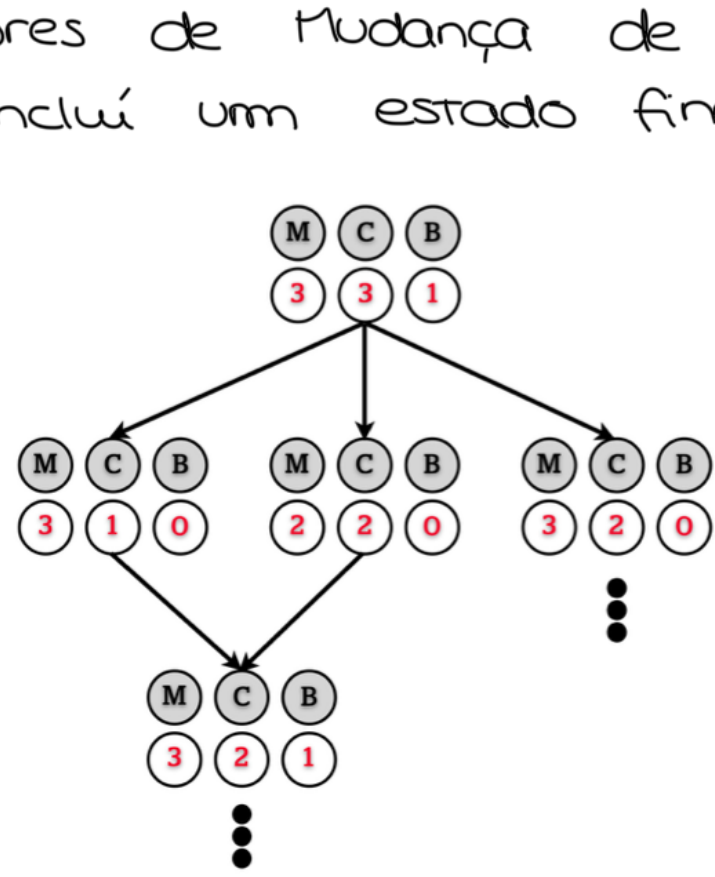
- **Estado:** Deve identificar de forma completa cada situação.

No problema dos missionários e Canibais, o estado seria algo tipo: M-1 (Missionários à direita).
C-1 (Canibais à direita).
B-0 (Posição do Barco).

- **Operadores de Mudança de Estado:** Determinam todas as ações possíveis a partir de cada Estado.

No problema seria algo tipo: (1) Transportar para a outra margem (1 missionário, 2 canibais, 1 missionário e 1 canibal ...). Não pode violar as restrições do sistema!

- **Árvore/Espaço de Procura:** Resulta da aplicação recursiva dos operadores de Mudança de Estado ao Estado inicial (Raíz). Incluí um estado final (objetivo).



Algoritmo Geral de Procura:

O processo é formalizado por um algoritmo geral:

1. inicializa a árvou de procura com o estado inicial do problema.
2. Loop
   - 2.1 se não há candidatos para expandir, devolve falha.
   - 2.2. Escolhe um estado na fronteira para expansão, de acordo com a estratégia.
   - 2.3. Se o estado contém o objetivo, devolve a solução
   - 2.4. Acrescenta à árvou de procura a expansão, com os seus sucessores, de acordo com a estratégia.
3. Terminar função.

Procura Cega: Detalhes das Estratégias

1. Largura Primeiro (Breath-First search)

- Funcionamento: Explora todos os nós a um determinado nível de profundidade antes de passar para o próximo nível. Utiliza uma fila (FIFO) para gerir os nós a expandir.

1. nós (fazer a fila, já com estado inicial)
2. Loop
   - 2.1. se nós está vazio, devolve falha.
   - 2.2. se Teste(nós[0]), devolve nós[0].
   - 2.3. Expande nós[0] e insere na fila.
3. Fim da função.

2. Profundidade Primeiro (Depth-First Search):

- Funcionamento: Explora o caminho mais profundo possível antes de retroceder e explorar outros ramos. Utiliza uma pilha (LIFO) para gerir os nós a explodir.

1. nós (fazer a pilha, já com o estado inicial)
2. Loop
   - 2.1. Se nós está vazio, devolve falha.
   - 2.2. se Teste(nós[0]), devolve nó. (topo da pilha)
   - 2.3. Expande nós[0] e coloca no topo da pilha.
3. Fim da função.

	Largura	Profundidade
Desempenho (completo):	Sim.	Não. (2)
Discriminador:	Não. (1)	Não. (1)
Tempo:	$O(r^n)$.	$O(r^n)$.
Espaço:	$O(r^n)$.	$O(r \times n)$.

(1) Garante a solução mais curta mas não garante ser a melhor em termos de custo.
(2) Pode entrar em loops infinitos ou a caminhos longos que não levam à solução.

3. Custo Uniforme (Uniform Cost Search):

- Funcionamento: Semelhante à Breath-search. Prioritiza a expansão de nós com o menor custo acumulado desde o estado inicial.

Nota: $g(k)$ → custo desde o nó inicial até ao k.

Desempenho completo: sim, desde que $g(\text{sucessor}(k)) \geq g(k)$.
Discriminador: Sim (solução de menor custo).
Económico: Não? (Devido à necessidade de ordenar por custo)

4. Profundidade Limitada (Depth-Limited search):

- Funcionamento: Objetivo de evitar ciclos infinitos. variação da Depth-First, mas com limite de profundidade

Nota: Usar quando se conhece o nível máximo a que pode chegar.

Desempenho completo: Sim, se o limite for $\geq$ ao da solução
complexidade temporal: $O(r^{\ell})$
Complexidade Espacial: $O(r \times \ell)$, ℓ = limite de profundidade.

5. Aprofundamento Progressivo (Iterative Deepening Depth-First search)

- Funcionamento: Quando não conhecemos o ℓ, então vamos iterando o seu valor (no algoritmo 4.) até que encontre a solução.

1. From i in $[0, +\infty[$:
   1.1. Procura Profundidade Limitada (limite i)
      1.1.1. Se encontrar solução retorna.
2. Fim da função.

Desempenho completo: Sim.
complexidade temporal: $O(r^{\ell})$
Complexidade Espacial: $O(r \times e)$, e é o limite de depth.