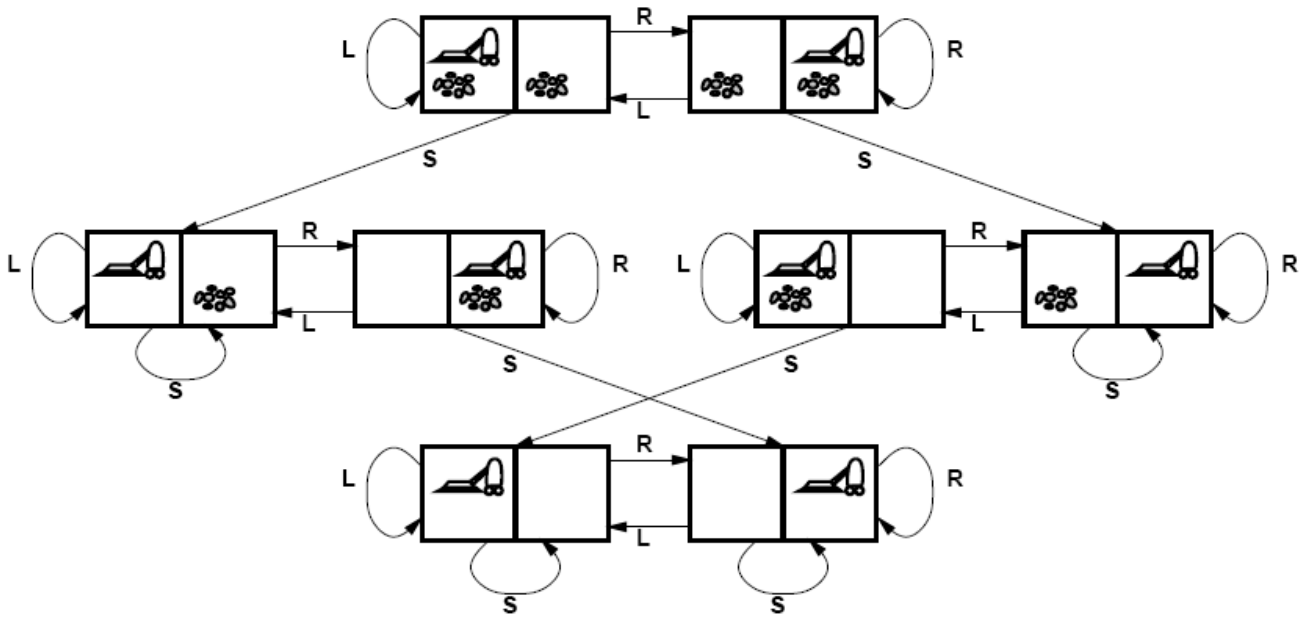
Busca em Ambientes Complexos

- O que fazer em ambientes parcialmente observáveis ou não-determinísticos?
- Os percepts passam a ter importância
  - Reduzem o possível espaço de estados
  - Permitem verificar o resultado das ações
- A solução passa a ser um **Plano de Contingência**
  - Determinam a ação a ser feita dependendo dos percepts recebidos

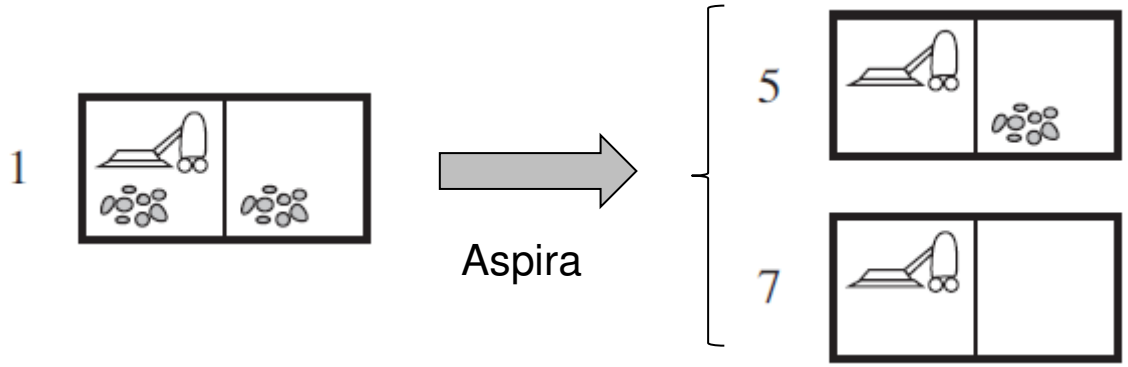
Aspirador de Pó

- Funcionando direito, o problema é resolvido por um busca simples



Aspirador de Pó Errático

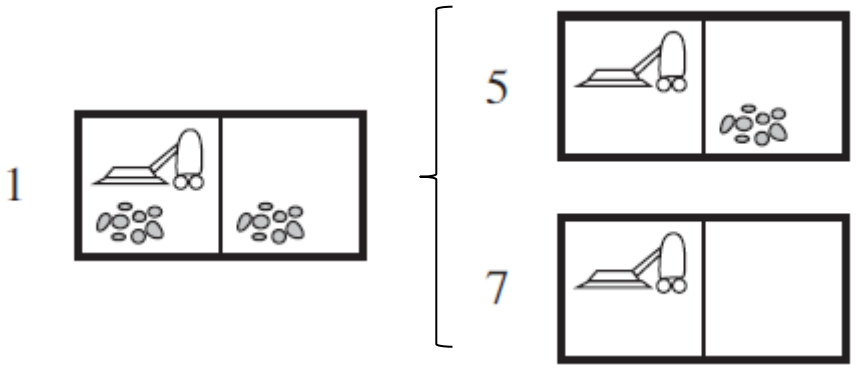
- Ação de aspirar **pode** dar errado
  - Ao ser aplicada em um quadrado sujo, pode aspirar a sujeira do quadrado o vizinho também
  - Ao ser aplicada em um quadrado limpo, pode depositar sujeira no quadrado.
- Cada ação pode resultar em um conjunto de estados ao invés de um estado único



Aspirador de Pó Errático

- A solução passa a ser um plano de contingência, no qual são usado comandos condicionais para definir as ações

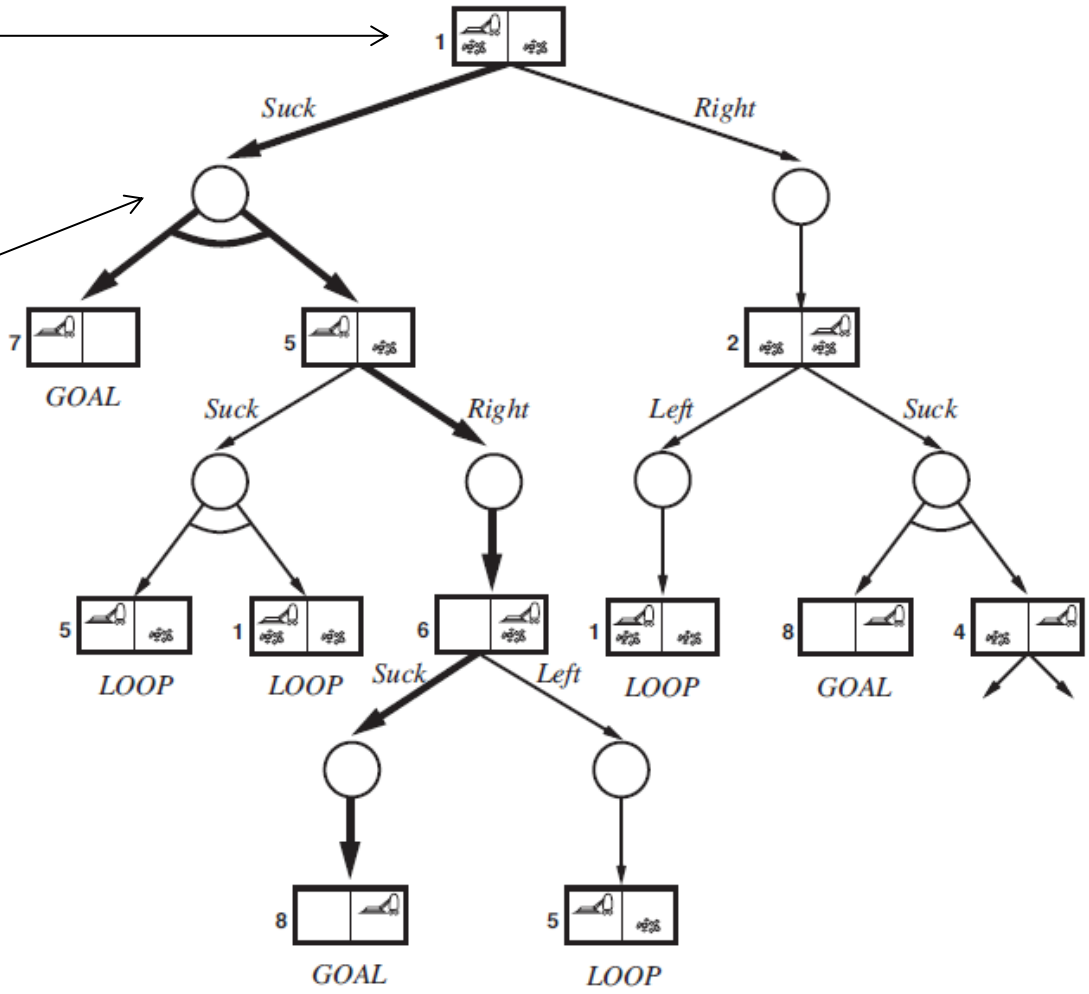
```
Aspira;
Se estado = 5 então
     direita;
     aspira;
senão
     nop;
```



- A solução é uma árvore ao invés de uma sequência de ações
- Como achar a solução?

And-Or Trees

- **Nodo Or**
  - Possíveis ações
- **Nodo And**
  - Possíveis resultados

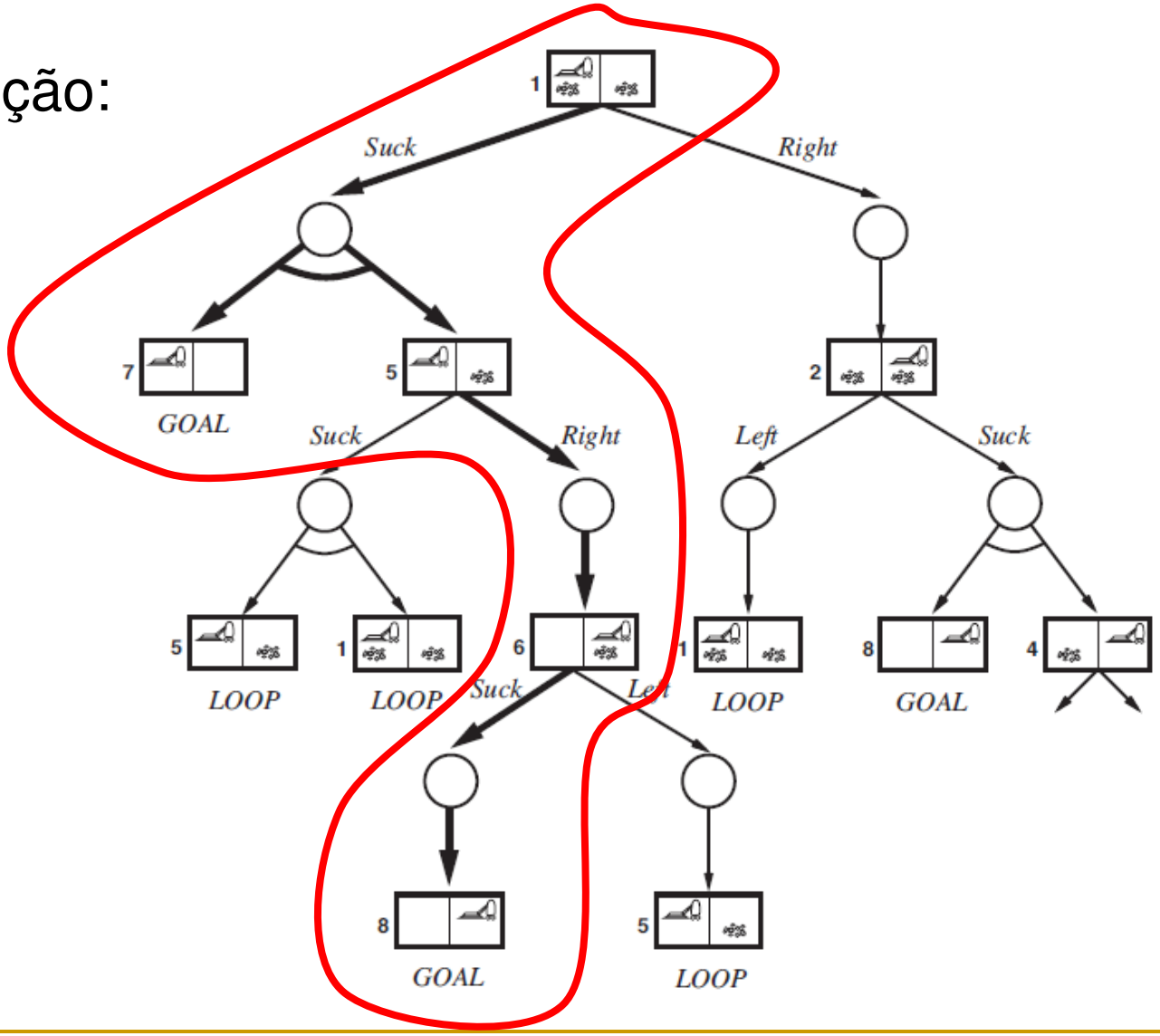


And-Or Trees

- A solução é uma subárvore que:
  1. Possui um estado gol em todas as folhas
  2. Especifica uma ação em cada nodo Or
  3. Inclui todos os *branches* dos nodos And
- Observações
  - Deve-se tomar cuidado no tratamento de loops e ações cíclicas
  - É possível intercalar planejamento e ação

And-Or Trees

Solução:



And-Or Trees

```
function AND-OR-GRAPH-SEARCH(problem) returns a conditional plan, or failure
  OR-SEARCH(problem.INITIAL-STATE, problem, [])

function OR-SEARCH(state, problem, path) returns a conditional plan, or failure
  if problem.GOAL-TEST(state) then return the empty plan
  if state is on path then return failure
  for each action in problem.ACTIONS(state) do
      plan ← AND-SEARCH(RESULTS(state, action), problem, [state | path])
      if plan ≠ failure then return [action | plan]
  return failure

function AND-SEARCH(states, problem, path) returns a conditional plan, or failure
  for each s_i in states do
      plan_i ← OR-SEARCH(s_i, problem, path)
      if plan_i = failure then return failure
  return [if s_1 then plan_1 else if s_2 then plan_2 else ... if s_{n-1} then plan_{n-1} else plan_n]
```

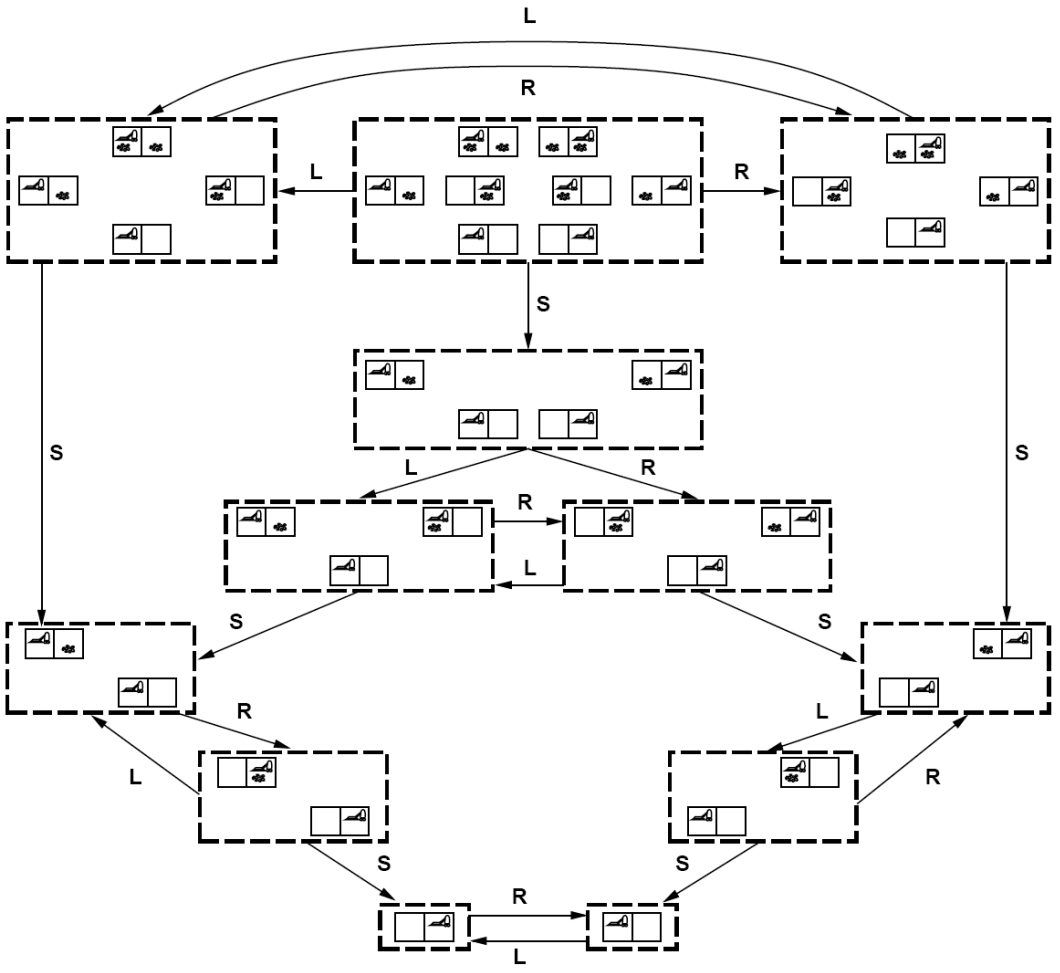
Figure 4.11 An algorithm for searching AND–OR graphs generated by nondeterministic environments. It returns a conditional plan that reaches a goal state in all circumstances. (The notation $[x \mid l]$ refers to the list formed by adding object x to the front of list l.)

Sensorless Problems

- O agente não tem sensores para avaliar o seu estado (**observação parcial**), mas sabe o resultado de suas ações
- Nesse caso, o agente trabalha com conjuntos de estados onde ele pode estar
  - *Belief State* (Estado de Crença)
- A pesquisa é feita sobre o espaço de *belief states* ao invés do espaço de estados

Sensorless Problems

- Por exemplo, no caso do aspirador de pó, ele pode estar em qualquer dos 8 estados, mas uma ação de ir para a direita restringe o conjunto para 4



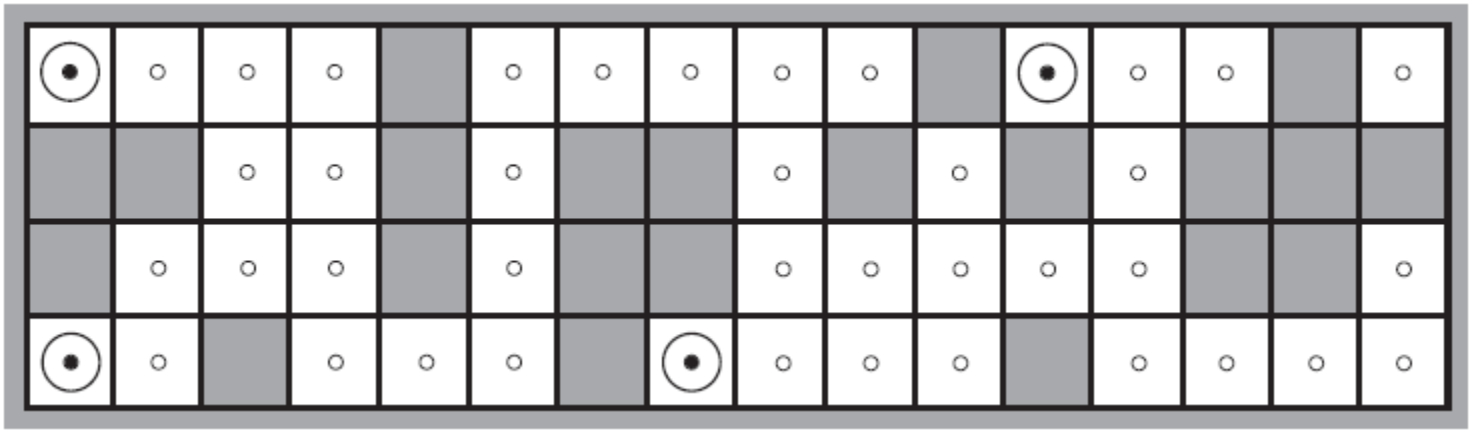
Busca em ambientes complexos

- De forma geral, quando o ambiente é parcialmente observável e/ou estocástico, o algoritmos devem **estimar** o estado
- Estimação Recursiva de Estados
- Duas etapas:
  - **Predição**
    - Qual vai ser o próximo estado dado uma ação *a*
  - **Observação** (atualização)
    - Qual estado eu estou dado uma observação *o*

$$b' = \text{Update}(\text{Predict}(b,a),o)$$

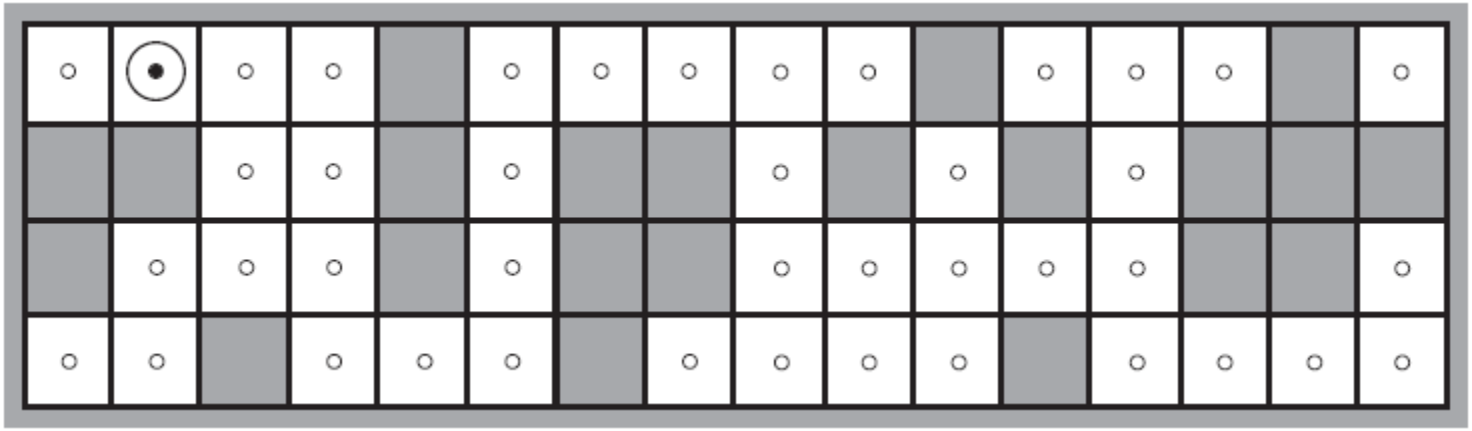
Exemplo: Localização de um robô

- Características:
  - Ambiente discreto, mapa conhecido, localização inicial desconhecida, sensores perfeitos, movimentação estocástica
- Sensores retornam *NSW*
  - b_1=Update(B_0, NSW)



Exemplo: Localização de um robô

- Robô se move
  - b_2=Predict$(b_1,move)$
  - Como o movimento é incerto, o resultado são os estados alcançáveis a partir de b_1 pela ação move
- Sensores retornam *NS*
  - b_3=Update(B_2,NS)



Exemplo: Localização de um robô

- Estimação Recursiva de Estados:

b' = Update(Predict(Update(b_0,NSW),*move*),*NS*)

- Normalmente a estimação recursiva de estados é feita sobre modelos de predição e observação probabilísticos... (Caps. 15 - 17)

Busca "On-Line"

- Devem intercalar planejamento e ação
  - Ao contrário os algoritmos de busca *off-line* que primeiro planejam depois agem
  - Normalmente necessários em ambientes dinâmicos e/ou estocásticos
  - Problemas de **exploração**

Busca "On-Line"

- Normalmente o agente tem acesso a:
  - *Ações(s)*
  - *Custo(s,a,s')* (após aplicar a e chegar a s')
  - *Test_Goal(s)*

- E, por enquanto, considera-se que os agentes:
  - Sabem onde estão
  - Reconhecem que já passaram por lá
  - Ações são determinísticas
  - Possuem uma heurística: *h(s)*

Busca "On-Line"

- Localidade
  - Tudo é feito a partir de *s*. Não se pode explorar facilmente outra região do espaço de estados
- Se as ações podem ser revertidas
  - Ambiente é *safely explorable*
- Algoritmos
  - Busca Cega – *Busca em profundidade on-line*
  - Busca Local – *Random walk*
  - Busca Heurística – *LRTA**