Planejamento como busca no Espaço de Estados

Leliane Nunes de Barros

Motivação

- Planejamento é um problema de busca
  - *Busca em espaço de estados*
    - » Cada nó representa um estado do mundo
    - » Um plano é um caminho através do espaço de estados
  - *Busca em espaço de planos*
    - » Cada nó representa um plano parcial dado por um conjunto de operadores parcialmente instanciados e um conjunto de restrições de ordem
    - » Um plano é obtido adicionando-se cada mais e mais restrições, até obtermos um plano solução.

Planejamento como Busca

	Espaço de Estados	Espaço de Planos
Algoritmo	Planejamento Progressivo (busca para frente) (Planejamento Regressivo) (busca para trás)	POP *Partial-Order Planning*
Nós	Estados do Mundo	Planos Parciais
Arestas/ Transições	Ações Por exemplo, no mundo dos blocos proposicional: • move-A-from-B-to-C • move-B-from-A-to-Table • move-C-from-B-to-A • …	Refinamentos de Planos: • Step addition • Step reuse • Demotion • Promotion

Tópicos

- Planejamento como uma ***busca em espaço de estados***
  - » Planejamento Progressivo
  - » Planejamento Regressivo
  - » Lifting
  - » STRIPS
  - » Exemplo: O Mundo dos Blocos

Forward-search(O, s_0, g)

$s \leftarrow s_0$

$\pi \leftarrow$ the empty plan

loop

if s satisfies g then return π

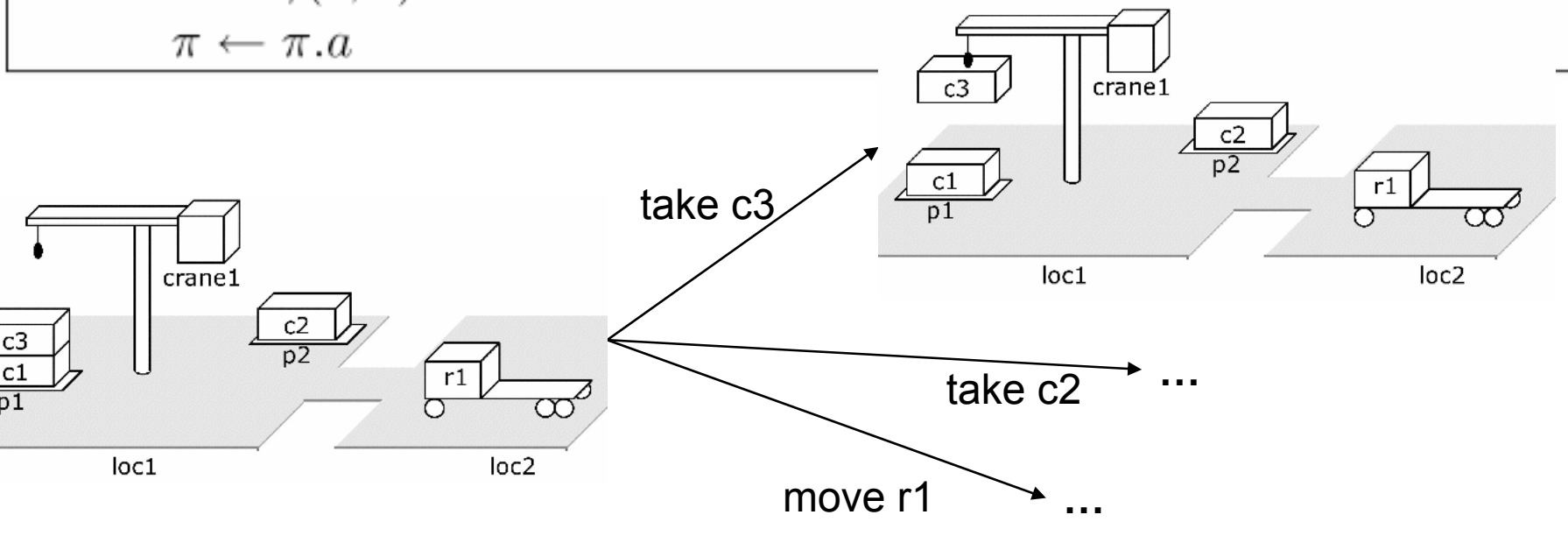
$E \leftarrow \{a | a$ is a ground instance an operator in O,
and $\mathrm{precond}(a)$ is true in $s\}$

if $E = \emptyset$ then return failure

nondeterministically choose an action $a \in E$

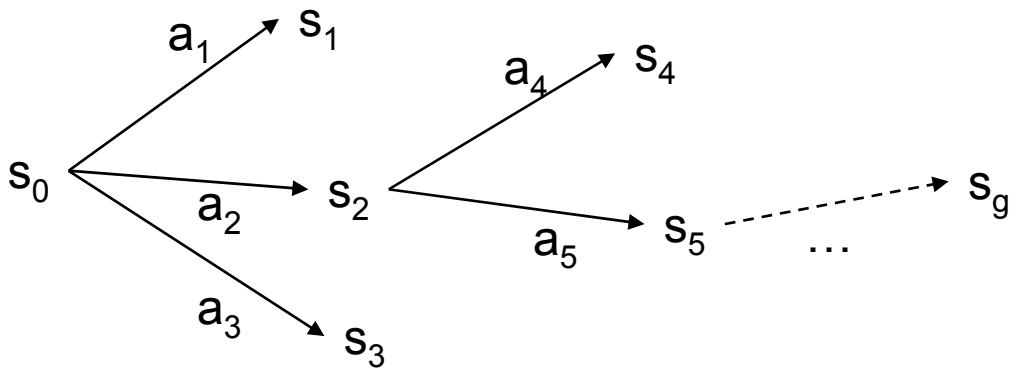
$s \leftarrow \gamma(s, a)$

$\pi \leftarrow \pi.a$



Planejamento Progressivo

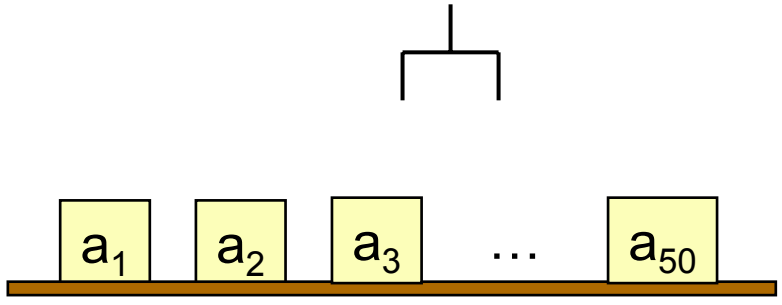
- Algumas implementações de busca para frente:
  - *breadth-first*
  - *best-first*
  - *depth-first*
  - *greedy*



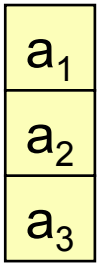
- Os algoritmos de busca *breadth-first* e *best-first* são corretos e completos
  - Porém, eles consomem muita memória: exponencial em função do tamanho da solução

- Na prática, é melhor usar uma busca *depth-first* ou *greedy*
  - Pior-caso: o uso de memória cresce linearmente em função do tamanho da solução
  - correto mas não completo
    - » como o planejamento clássico possui um número finito de estados, os caminhos não são infinitos mas podem entrar em loop ➔ é necessário evitar nós repetidos

Fator de ramificação do Planejamento Progressivo



Estado inicial



meta

- A busca para frente pode ter um fator de ramificação muito grande (veja exemplo)
- Porque isto é ruim?
  - Pode-se gastar tempo testando muitas ações irrelevantes
- É preciso construir boas funções heurísticas e/ou procedimento de poda.

Planejamento Regressivo

- No planejamento progressivo, começamos com o estado inicial e calculamos as transições de estados através da função de transição γ
  - $s' = \gamma(s,a)$
- No planejamento regressivo, começamos por um dos estados meta e calculamos a inversa da função de transição, γ^{-1}
  - Novo conjunto de sub-metas = $\gamma^{-1}(g,a)$

Transições inversas de estados

- O que significa $\gamma^{-1}(g,a)$?
- Primeiro precisamos definir *relevância* em planejamento regressivo:
  - Uma ação a é relevante para uma meta g se
    - » a torna pelo menos um dos literais de g verdadeiro
      - $g \cap \text{effects}(a) \neq \varnothing$
    - » a não torna falso nenhum dos literais de g
      - $g^{+} \cap \text{effects}^{-}(a) = \varnothing$
      - $g^{-} \cap \text{effects}^{+}(a) = \varnothing$
- Se a for relevante para g, então
  - $\gamma^{-1}(g,a) = (g - \text{effects(a)}) \cup \text{precond}(a)$

Backward-search(O, s_0, g)

$\pi \leftarrow$ the empty plan

loop

if s_0 satisfies g then return π

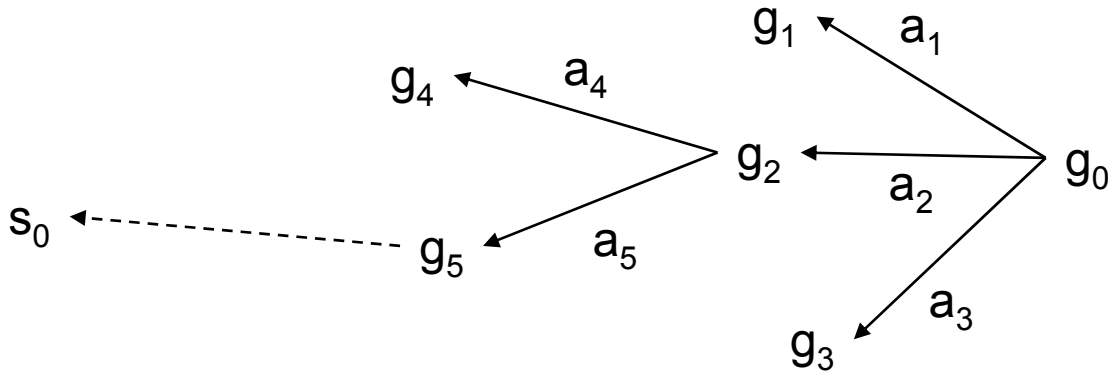
$A \leftarrow \{a | a$ is a ground instance of an operator in O
and $\gamma^{-1}(g, a)$ is defined$\}$

if $A = \emptyset$ then return failure

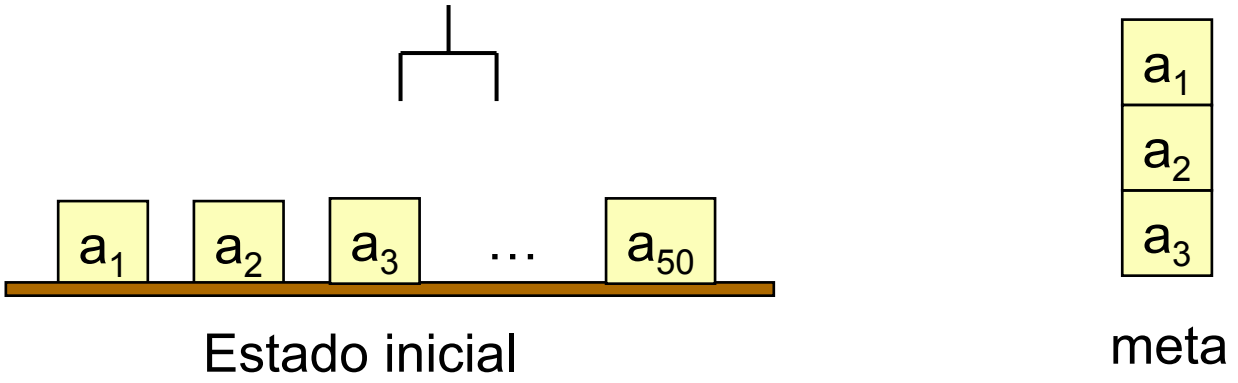
nondeterministically choose an action $a \in A$

$\pi \leftarrow a.\pi$

$g \leftarrow \gamma^{-1}(g, a)$

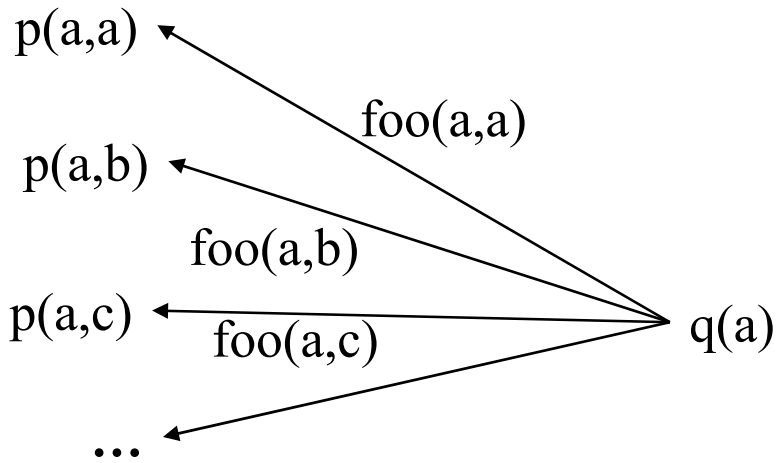


Eficiência do Planejamento Regressivo

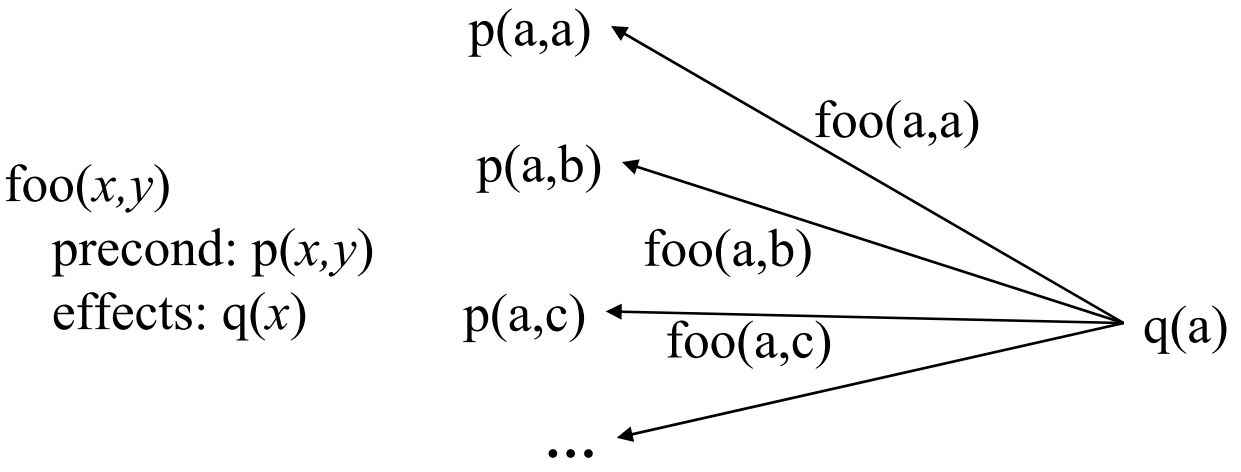


- O fator de ramificação da busca para trás é pequeno no exemplo
- Existem casos em que a ramificação pode ser muito grande
  - Muitas instâncias de operadores são avaliadas

foo(*x*,*y*)
precond: p(*x*,*y*)
effects: q(*x*)



Lifting



- Podemos reduzir o fator de ramificação se nós instanciamos *parcialmente* os operadores
  - Isto é chamado de *lifting*