Planejamento

Sistemas Inteligentes

Profa. Josiane

setembro/2006

De problemas de busca a planejamento

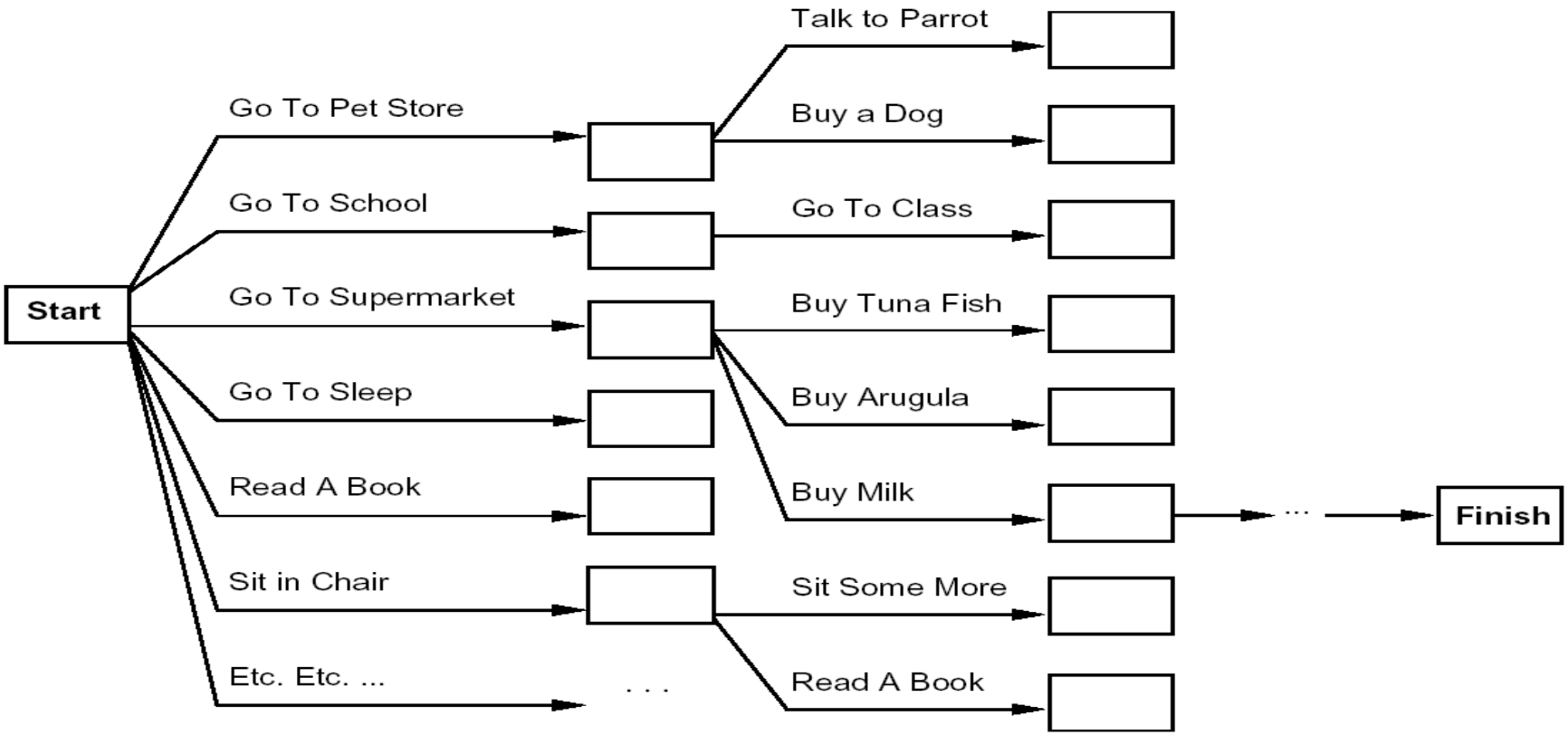
- Agentes baseado em planejamento diferem dos agentes de busca quanto à representação dos objetivos, estados e ações, e na representação e a construção de seqüências de ações.

- Elementos básicos de um agente resolvedor de problemas baseado em busca:
  - Representação das ações (geram a descrição dos estados sucessores).
  - Representação dos estados (as descrições de todos os estados são completas).
  - Representação dos objetivos (a única informação que o agente tem sobre o objetivo é o teste do objetivo e a função heurística).
  - Representação dos planos. (somente uma seqüência de ações a partir do estado inicial).

De problemas de busca para planejamento

- Considere um agente com o seguinte problema "*pegue um litro de leite, um cacho de banana e uma liquidificador*".
  - Estado inicial: agente em casa, sem nenhum dos objetos
  - Conjunto de operadores: todas as coisas que o agente pode fazer
  - Função heurística: talvez o número de coisas ainda não adquiridas
- Algoritmos de busca
  - Fator de ramificação -> centenas ou milhões
  - Muitos estados e muitas ações a considerar
  - Função heurística -> pode decidir qual estado está mais perto do objetivo, mas não pode eliminar ações a considerar

De problemas de busca para planejamento



Bases do planejamento

- O planejamento está fundamentado sobre algumas idéias chaves:
  - Abrir a representação de estados, objetivos e ações.
    - Possibilita o uso de descrições em alguma linguagem formal.
  - Relaxar o requisito de construção seqüencial de soluções.
    - O planejador é livre para adicionar ações no plano onde elas forem necessárias.
    - Fazendo decisões "obvias ou importantes primeiro" o planejador reduz o fator de largura.
  - Uso de divisão-e-conquista através de sub-planos.
    - A maioria das partes do mundo são independentes da maioria das outras partes.

A linguagem de problemas de planejamento

- A representação deve tornar possível a criação de algoritmos de planejamento
  - Para tirar proveito da estrutura lógica do problema
- A linguagem deve ser:
  - Suficientemente expressiva para descrever uma ampla variedade de problemas
  - Mas restritiva o bastante para permitir que algoritmos eficiente operem sobre ela
- Linguagem básica de representação de planejadores clássicos:
  - STRIPS: *Stanford Research Institute Problem Solver*

Representação de Estados e objetivos

- Planejadores decompõe o mundo em condições lógicas
- Estado = conjunção de literais positivos
  - Literais proposicionais: *Pobre* ^ *Desconhecido*
  - Literais primeira ordem devem ser básicos e livres de funções:
    - *Em(Avião1, Viracopos)* ^ *Em(Avião2, StDumont)*
    - Não são permitidos, por exemplo: *Em(Pai(João), StDumont)*
- Objetivo = Conjunção de literais básicos positivos
  - Estado parcialmente especificado
  - Exemplo: *Rico* ^ *Famoso, Em(P2, Taiti)*
  - Um estado proposicional *s* satisfaz a um objetivo *g* se *s* contém todos os átomos em *g* (e possivelmente outros)
    - Ex: Rico ^ Famoso ^ Miserável satisfaz o objetivo Rico ^ Famoso

Representação de ações

- Uma ação é especificada em termos de:
  - Precondições que devem ser válidas, antes de ser possível executá-la
  - Efeito que resulta quando ela é executada

- Ex: Uma ação de um avião voar de uma lugar para outro

```
Ação (Voar(p, de, para)
PRECOND: Em(p, de) ^ Avião(p) ^ Aeroporto(de) ^
  Aeroporto(para)
EFFECT: ¬Em(p, de) ^ Em(p, para))
```

- Denominado **esquema de ação** -> representa várias ações diferentes, derivadas pela instanciação das variáveis

Semântica para problemas de planejamento

- Uma ação é **aplicável** a qualquer estado que satisfaz a precondição
  - Caso contrário, a ação não tem nenhum efeito
- Exemplo
  - Estado atual: *Em (P1, JFK) ^ Em (P2, SFO) ^ Avião (P1) ^ Avião (P2) ^ Aeroporto (JFK) ^ Aeroporto (SFO)*
  - Satisfaz a precondição: *Em(p, de) ^ Avião(p) ^ Aeroporto(de) ^ Aeroporto(para)*
  - Fazendo as unificações, temos que a ação concreta abaixo é aplicável
    - *Voar(P1, JFK, SFO)*
  - Depois da aplicação desta ação o estadual atual se torna:
    - *Em (P1, SFO) ^ Em (P2, SFO) ^ Avião (P1) ^ Avião (P2) ^ Aeroporto (JFK) ^ Aeroporto (SFO)*

A aplicação da ação

- O resultado da aplicação de uma ação *a* a um estado *s é* igual a *s*, exceto pelo fato de que:
  - Qualquer literal positivo *P* no efeito de *a* ser adicionado a *s*
  - Qualquer literal negativo ¬*P* ser removido de *s*
- Se um efeito positivo já estiver em *s*, ele não aparece duas vezes
- Se um efeito negativo não estiver em *s*, esta parte do efeito será ignorada
- **Hipótese de STRIPS**: todo literal não mencionado no efeito permanece inalterado

A Solução em problemas de planejamento

- Forma mais simples:
  - Seqüencia de ações que, quando executada no estado inicial, resulta em um estado que satisfaz o objetivo

Expresssividade da STRIPS

- Restrições na representação = algoritmos mais eficientes
  - Ex: Literais livres de funções
- STRIPS não é suficientemente expressiva para alguns domínios reais
- Variantes da linguagem STRIPS foram desenvolvidas:
  - Importante: ADL – *Action Description Language*
  - A ação voar em ADL:

```
Ação(Voar(p:Avião, de: Aeroporto, para:Aeroporto),
  PRECOND: Em(p, de) ^ (de ≠ para)
  EFFECT: ¬Em(p, de) ^ Em(p, para))
```

  - Sintaxe-padrão: Linguagem de Definição de Domínio de Planejamento (PDDL)

- **STRIPS**
- Só literais positivos nos estados
- Assume mundo fechado
- Efeito P^¬Q significa inserir P e remover Q
- Não há variáveis nos objetivos
- Objetivos são conjunções
- Os Efeitos são conjunções
- Não suporta igualdade
- Não suporta tipos
- Símbolos de funções não são permitidos

- **ADL**
- Literais positivos e negativos nos estados
- Assume mundo aberto
- Efeito P^¬Q significa inserir P e ¬Q e apagar ¬P e Q
- Variáveis quantificadas nos objetivos
- Objetivos com conjunções e disjunções
- Efeitos condicionais são permitidos: **when** P:E significa que P é efeito só se E for verdadeiro
- Suporta igualdade
- Variáveis podem ter tipos

Problema do transporte aéreo de cargas

Início(*Em*(*C1*, *SFO*) ^ *Em*(*C2*,*JFK*) ^ *Em*(*A1*,*SFO*) ^ *Em*(*A2*,*JFK*) ^ *Carga*(*C1*) ^ *Carga*(*C2*) ^ *Avião*(*A1*) ^ *Avião*(*A2*) ^ *Aeroporto*(*JFK*) ^ *Aeroporto*(*SFO*))

Objetivo(*Em*(*C1*,*JFK*) ^ *Em*(*C2*,*SFO*))

Ação(*Carregar*(*c*,a,*l*)

PRECOND: *Em*(*c*,*l*) ^ *Em*(*a*,*l*) ^ *Carga*(*c*) ^ *Avião*(*a*) ^ *Aeroporto*(*l*)

EFFECT: ¬*Em*(*c*,*l*) ^ *Dentro*(*c*,*a*))

Ação(*Descarregar*(*c*,*a*,*l*)

PRECOND: *Dentro*(*c*,*a*) ^ *Em*(*a*,*l*) ^ *Carga*(*c*) ^ *Avião*(*a*) ^ *Aeroporto*(*l*)

EFFECT: *Em*(*c*,*l*) ^ ¬*Dentro*(*c*,*a*))

Ação(*Voar*(*a*,*de*,*para*)

PRECOND: *Em*(*a*,*de*) ^ *Avião*(*a*) ^ *Aeroporto*(*de*) ^ *Aeroporto*(*para*)

EFFECT: ¬*Em*(*a*,*de*) ^ *Em*(*a*,*para*))

[*Carregar*(*C1*,*A1*,*SFO*), *Voar*(*A1*,*SFO*,*JFK*), *Descarregar*(*C1*,*A1*,*JFK*), *Carregar*(*C2*,*A2*,*JFK*), *Voar*(*A2*,*JFK*,*SFO*), *Descarregar*(*C2*,*A2*,*SFO*)]

O problema do pneu sobressalente

Início(Em(Furado, Eixo) ^ Em(Sobresselente,Bagageira))

Objetivo(Em(Sobresselente,Eixo))

Ação(Remove(Sobresselente,Bagageiro)

PRECOND: Em(Sobresselente,Bagageiro)

EFFECT: ¬Em(Sobresselente,Bagageiro) ^ Em(Sobresselente,Chão))

Ação(Remove(Furado,Eixo)

PRECOND: Em(Furado,Eixo)

EFFECT: ¬Em(Furado,Eixo) ^ Em(Furado,Chão))

Ação(Colocar(Sobresselente,Eixo)

PRECOND: Em(Sobresselente,Chão) ^ ¬Em(Furado,Eixo)

EFFECT: Em(Sobresselente,Eixo) ^ ¬Em(Sobresselente,Chão))

Ação(DeixarDuranteANoite

PRECOND:

EFFECT: ¬ Em(Sobresselente,Chão) ^ ¬ Em(Sobresselente,Eixo) ^ ¬Em(Sobresselente,Bagageira) ^ ¬Em(Furado,Chão) ^ ¬Em(Furado,Eixo))

O mundo de blocos

- **O que estamos falando:** O domínio consiste de um conjunto de blocos sobre uma mesa. Os blocos podem ser empilhados. O robô só pode pegar um bloco por vez. O objetivo é fazer uma ou mais pilhas de blocos.

- **Vocabulário:** Nós utilizamos *Sobre(b,x)* para dizer que o bloco *b* está sobre *x*, onde *x* é outro bloco ou a mesa. Para mover o bloco utilizamos *Mover(b,x,y)*, movendo o bloco *b* da posição *x* para a *y*. Utilizamos *Livre(x)* para dizer que não há nada sobre *x*

- **Operadores:** A descrição para *Mover* seria:

  Ação(Mover(b,x,y),
  PRECOND: Sobre(b,x) $\wedge$ Livre(b) $\wedge$ Livre(y)
  EFFECT: Sobre(b,y) $\wedge$ Livre(x) $\wedge \neg$Sobre(b,x) $\wedge \neg$Livre(y))

- Infelizmente a propriedade *Livre* não é válida quando *x ou y* é a mesa. Introduzimos então o operador *MoverParaMesa*:

  Ação(MoverParaMesa(b,x),
  PRECOND: Sobre(b,x) $\wedge$ Livre(b),
  EFFECT: Sobre(b,Mesa) $\wedge$ Livre(x) $\wedge \neg$Sobre(b,x))

O mundo de blocos

- Manter *MovererParaMesa* desta forma não impede que o planejador utilize *Mover*(b, x, *Mesa*) ao invés de utilizar *MoverParaMesa*(b,x)
  - Isto não produzirá planos incorretos mas aumentará o espaço de busca
  - Podemos introduzir o predicado *Bloco*(b) e *Bloco*(y)
  - Podemos acrescentamos também o predicado (b≠y) para evitar ações do tipo Mover(B, C, C)

Ação(Mover(b,x,y),
PRECOND: Sobre(b,x) ∧ Livre(b) ∧ Livre(y) ∧ Bloco(b) ∧ Bloco(y) ∧ (b≠x) ∧ (b≠y) ∧ (x≠y),
EFFECT: Sobre(b,y) ∧ Livre(x) ∧ ¬Sobre(b,x) ∧ ¬Livre(y))

Ação(MoverParaMesa(b,x),
PRECOND: Sobre(b,x) ∧ Livre(b) ∧ Bloco(b) ∧ Bloco(y) ∧ (b≠x),
EFFECT: Sobre(b,Mesa) ∧ Livre(x) ∧ ¬Sobre(b,x))

Iniciar(Sobre(A, Mesa) ∧ Sobre(B, Mesa) ∧ Sobre(C, Mesa) ∧ Bloco(A) ∧ Bloco(B) ∧ Bloco(C) ∧ Livre(A) ∧ Livre(B) ∧ Livre(C))

Objetivo(Sobre(A, B) ∧ Sobre(B, C))

Algoritmos de planejamento

- Busca no espaço de estados
  - Para frente: a partir do estado inicial (por progressão)
  - Para trás: a partir do objetivo (por regressão)
- Estado inicial da busca
  - Estado inicial do problema de planejamento
- Ações aplicáveis a um estado
  - Todas aquelas cujas precondições são satisfeitas
  - Função sucessora única e simples
- Teste de objetivo
  - Verifica se o estado satifaz o objetivo do problema de planejamento
- Custo do passo
  - Geralmente igual a 1

Busca para a frente

- Ineficiente
  - Não ataca o problema de ações irrelevante
  - Gera um fator de ramificação muito grande
  - Problema das cargas
    - 10 aeroportos, 5 aviões, 20 itens de carga -> Mover toda a carga do aeroporto A para B
    - Em média 1000 ações possíveis
    - Média de 1000^{41} nós na árvore de busca
  - Precisa de uma heurística muito precisa

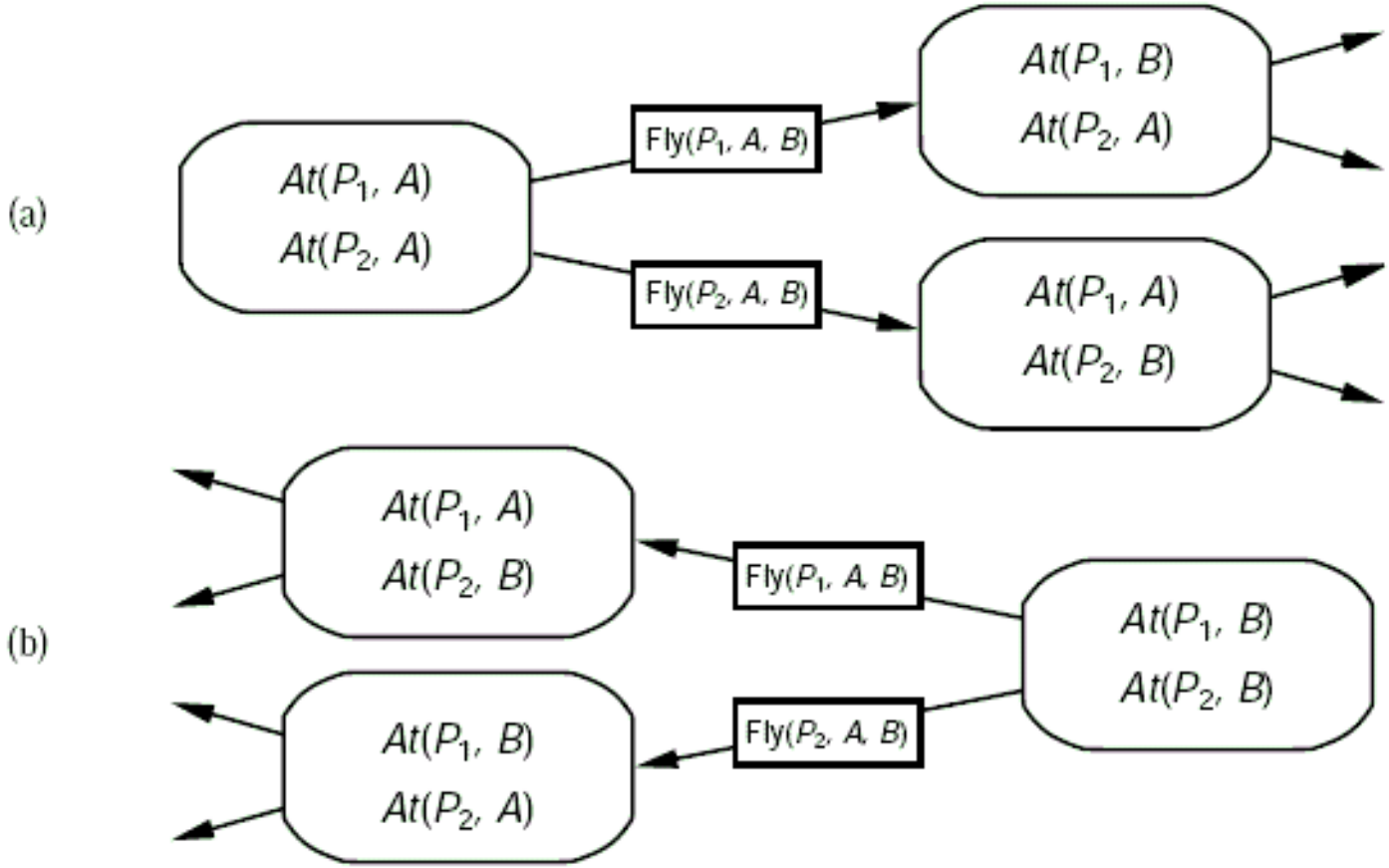
Busca para trás

- Vantagem: Permite considerar apenas ações **relevantes**
- Menor fator de ramificação
  - Ex: problema de cargas
    - 1000 ações para a frente e 20 ações para trás
- Uma ação é relevante para um objetivo conjuntivo
  - Se ela alcança um dos elementos da conjunção do objetivo
  - Ex: objetivo problema de cargas 10 aeroportos e 20 itens em B
    - $Em(C_1, B) \wedge Em(C_2, B) \wedge Em(C_3, B) \wedge \ldots \wedge Em(C_{20}, B)$
    - Podemos buscar por ações que tem cada elemento da conjunção como efeito
    - Por exemplo: $Descarregar(C_1, p, B)$ tem como efeito $Em(C_1, B)$
- Quais são os estados a partir dos quais a aplicação de uma dada ação leva ao objetivo?

Busca para trás

- Buscar por ações que chamadas **consistentes**
  - Alcancem algum literal desejado
  - Não desfaçam quaisquer literais desejados
- Seja G a descrição de um objetivo e A uma ação consistente e relevante
  - O estado predecessor correspondente é descrito como:
    - Quaisquer efeitos positivos de A que aparecem em G são eliminados
    - Cada literal de precondição de A é adicionado, a menos que já apareça
  - O algoritmo pára quando é gerada uma descrição de predecessor que é satisfeita pelo estado inicial do problema de planejamento

Busca para (a) Frente (b) trás



Heurística para busca no espaço de estados

- Nem a busca para trás e nem a busca para frente são eficientes sem uma heurística
- A idéia básica
  - Observar os efeitos das ações
  - Observar os objetivos que devem ser alcançados
  - Avaliar quantas ações serão necessárias para alcançar todos os objetivos
- Duas abordagens
  - Derivar um problema relaxado
    - Remover todas as precondições das ações
    - Remover efeitos negativos
  - Fingir um algoritmo puro de dividir-e-conquistar irá funcionar
    - Independência de subobjetivo