Planejadores Heurísticos no Espaço de Estados

Leliane Nunes de Barros

Busca heurística no espaço de estados

- Volta à busca no espaço de estados!
- Após alguns anos de pesquisa na área de planejamento no espaço de planos (POP, UCPOP, SNLP, etc), os pesquisadores acumularam uma grande experiência na construção de algoritmos dedicados
- Essa experiência foi usada na construção de boas heurísticas

Heurísticas que exploram a estrutura do espaço de busca do domínio de planejamento

Exemplo de planejadores heurísticos

- UNPOP [Drew McDermott, 96] *Greedy Regression-Match Graphs*
- HSP [Geffner, 97] *Heuristic Search Planner*
- FF [Hoffmann, 2000] *Fast Forward*

Planejadores heurísticos

- Heurística h(s) é computada resolvendo um problema relaxado. Exemplo:
  - distância de Manhattan
  - comprimento do plano solução de um problema relaxado em que são removidas todas as listas de eliminação das ações
- Heurística informativa e admissível … mas ainda recai num problema intratável

HSP – Heuristic State Planner

- Forward planning
- algoritmos de busca: Hill-Climbing ou A* com uso de uma função heurística derivada de uma *descrição de alto-nível de ações*

HSP – Heuristic State Planner

- A heurística $\mathrm{h(s)}$ é uma estimativa do número de passos necessários para resolver o problema relaxado, ou seja, num domínio em que as listas de eliminação das ações são desprezadas.
- s: estado, p: proposição
- $g(\{\mathrm{p}\};s)$: custo estimado para atingir p apartir de s
- $g(Prec(op;s))$: custo estimado para atingir as precondições da ação op de s

Heurística do planejador HSP

*Função heurística **aditiva***

Custo estimado para atingir a proposição p, a partir de s

$$g(\{p\};s) = \begin{cases} 0 & se\ p \in s \\ \\ min_{op \in O(p)}[1 + g(\underbrace{Prec(op)}_{\text{conjunto de átomos C}};\ s)] & caso\ contrário \end{cases}$$

Para conjuntos de átomos C:

$$g(C;s) = \sum_{r \in C} g(\{r\};s)$$

Heurística resultante para um estado s qualquer:

$$h_{add}(s) = \sum_{p \in G} g(\{p\};s)$$

Heurística para o HSP

*Função heurística **max***

- Custo estimado para atingir a proposição p apartir de s

$$g(\{p\};s) = \begin{cases} 0 & se\ p \in s \\ min_{op \in O(p)}[1 + g(\underbrace{Prec(op)}_{\text{conjunto de átomos } C};\ \mathrm{s})] & caso\ contrário \end{cases}$$

Para conjuntos de átomos C :

$$g(C;s) = \max_{r \in C} g(\{r\};s)$$

Heurística resultante para um estado s qualquer:

$$h_{max}(s) = \max_{p \in G} g(\{p\};s)$$

Heurísticas para o HSP

heurística aditiva:

– informativa mas não-admissível

– faz a suposição de metas independentes

heurística max:

– admissível mas não informativa

– supõe metas dependentes

– faz a suposição que o plano de comprimento máximo pode atingir um conjunto de sub-metas de uma vez, por exemplo, a lista de precondições de uma ação

Algoritmo para calcular h(G;s)

- polinomial no número de átomos e ações
- versão do algoritmo Bellman-Ford para encontrar o caminho mais curto em grafos

Cálculo de h(G;s)

Uma maneira possível de se calcular as heurística do HSP é dada pelos seguintes passos:

inicializar custos $g(p;s)$

$g(p;s)=0$ se $p \in s$ e $g(p;s)= \propto$ caso contrário

Usa um sistema de ***planejamento progressivo*** para a meta simples p. Quando p for adicionado pelo operador op atualize $g(p;s)$

$$g(\{p\};s) = min[g(\{p\};s), (1 + g(\overbrace{Prec(op)}^{\text{conjunto de átomos C}}; s)]$$

condição de parada: quando $g(p;s)$ não mudar mais de valor

duas opções para o cálculo do custo de um conjunto de átomos: custo aditivo ou custo max

The FastForward Planner (FF)

- FF faz busca heurística no espaço de estados.
- O cálculo de h(s) é baseado na solução de um problema relaxado usando o grafo de planejamento
- Aprimoriza a heurística do HSP, avaliando o tamanho de uma solução em quantidade de ações.

Extração de heurísticas a partir do grafo de planejamento

- O grafo de planejamento, além de fornecer uma estimativa da distância a partir de s_0 para atingir cada proposição alcançável *p* , também fornece informações de *mutex* (μP_i)
  - O procedimento *Solution extraction* seleciona um conjunto de proposições *g* em uma camada somente se nenhum par de elementos em *g* for um *mutex*.
- Para um problema relaxado não existem mutexes!!!
-

Extração de heurísticas a partir do grafo de planejamento

- Lembremos como GraphPlan trabalha:

  loop

  Graph expansion: Leva tempo polinomial

  extend a "planning graph" forward from the initial state until we have achieved a necessary (but insufficient) condition for plan existence

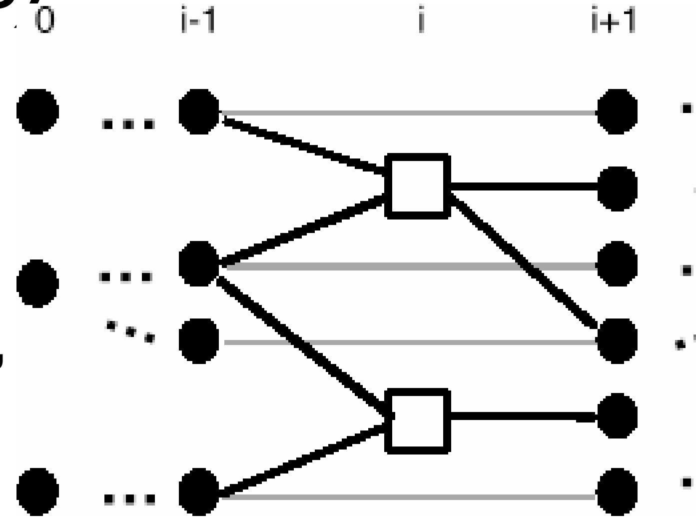
  Solution extraction: Leva tempo exponencial

  search backward from the goal, looking for a correct plan

  if we find one, then return it

  repeat

Usando o Grafo de Planejamento para calcular *h(s)*



- h(s) é calculado como uma processo simplicado de extração de plano
- Construa um grafo de planejamento, começando em *s* até que os átomos de G sejam alcançados
- Seja R o nível da primeira camada (da esquerda para direita) que "possivelmente atinge" p
- Escolha a ação com o menor número de precondições e que, se possível, já foi escolhida para as demais proposições
- h(s) = número de ações escolhidas

FF na competição de planejamento AIPS-2000

- FastForward foi um dos melhores
- Nessa competição todos os problemas de planejamento eram clássicos
- FF ganhou um prêmio de “outstanding performance”
- FF encontrou planos num tempo muito curto quando comparado com outros planejadores clássicos

FF na competição de planejamento AIPS-2002

- FF ficou na média entre todos os planejadores
- LPG (graphplan + local search) se saiu muito melhor
- Uma das coisas que causaram dificuldades para o FastForward
  - Os problemas na competição de AIPS-2002 foram além de planejamento clássico e envolveram:
    - Variáveis numéricas, otimização, durações de tempo

Exemplo: domínio D^{m}S^{2*}

Problema:

- Estado Inicial: {I1, I2, I*},
- Estado Goal: {G2, G1, G*}
- domínio DmS2*, definido pelas ações da tabela abaixo

Ação: A_i^1 Precondição: $\{I_i\}$ Adição: $\{P_i\}$ Eliminação: $\{P_j \mid j < i\}$	Ação: A_i^2 Precondição: $\{P_i\}$ Adição: $\{G_i\}$ Eliminação: $\{P_j \mid j < i\}$
Ação: $A^1{}_*$ Precondição: $\{I_*\}$ Adição: $\{G_*\}$ Eliminação: $\{I_i \mid \forall i\} \cup \{G_i \mid \forall i\}$	Ação: $A^2{}_*$ Precondição: $\{G_i \mid \forall i\}$ Adição: $\{G_*\}$ Eliminação: $\{I_i \mid \forall i\}$

Exemplo: domínio D^mS^{2*}

- Para i = 2

Ação: A_1^1 Precondição: $\{I_1\}$ Adição: $\{P_1\}$ Eliminação: $\{\}$	Ação: A_1^2 Precondição: $\{P_1\}$ Adição: $\{G_1\}$ Eliminação: $\{\}$	Ação: A_1^* Precondição: $\{I^*\}$ Adição: $\{G_*\}$ Eliminação: $\{I_1, I_2, G_1, G_2\}$
Ação: A_2^1 Precondição: $\{I_2\}$ Adição: $\{P_2\}$ Eliminação: $\{P_1\}$	Ação: A_2^2 Precondição: $\{P_2\}$ Adição: $\{G_2\}$ Eliminação: $\{P_1\}$	Ação: A_2* Precondição: $\{G_1, G_2\}$ Adição: $\{G^*\}$ Eliminação: $\{I_1, I_2\}$

Resolver o problema para D^mS^{2*} com os seguintes algoritmos

- FF
- HSP
- POP
- Para o POP usar ainda as ações

Ação: Start Precondição: {} Adição: {I_1, I_2, I*} Eliminação: {}	Ação: Finish Precondição:{G2, G1, G*} Adição: {} Eliminação: {}