O planejador FF

Gustavo Sverzut Barbieri ra008849
Henrique Dante de Almeida ra003065
Rafael Ávila de Espíndola ra003280

14 de Junho de 2004

1 Introdução

O planejador Fast Forward é resultado da evolução dos planejadores automáticos desenvolvidos ao longo da década de 90 [3]. O problema de planejamento STRIPS é intrinsecamente difícil ($\mathcal{PSPACE}-$completo para ser preciso) e uma busca exaustiva não é geralmente factível.

O FF pode ser entendido como uma tentativa de reunir heurísticas conhecidas que sejam úteis para atender o objetivo escolhido por seu autor: encontrar uma solução, qualquer que seja, muito rápido. Para isso ele utiliza uma busca local no espaço de estados. Se a busca é feita de forma inteligente, a solução tem grandes chances de ser encontrada com facilidade. Para tentar melhorar essa busca o planejador se baseia nas seguintes idéias:

- A cada passo, coletar dados que ajudem a estimar a distância à solução e a determinar melhor caminho a seguir. Estes dados devem ser simples de serem encontrados.
- Com ajuda das informações coletadas realizar uma busca local por um estado "melhor". Caminhando em seguida para este estado e esquecendo o passado.
- Se estas tentativas de busca falharem devido a *dead ends*, ou seja, se o FF se perder em caminhos que não chegam ao objetivo, parar tudo e tentar resolver o problema de novo com uma busca exaustiva utilizando *best first*.

Os dados a serem coletados foram aproveitados de conhecimento de planejadores anteriores [1, 2]. A idéia é que, em vez de tentar começar resolvendo o problema original, o planejador se preocupa em resolver alguma versão mais simples do mesmo que pode ser resolvido em tempo polinomial. A utilização do problema relaxado funciona da seguinte forma:

- O FF supõem que a solução do problema real deve ser de alguma forma parecida com a do problema relaxado. O algoritmo utiliza essa idéia para limitar a escolha do próximo passo.

- A solução do problema relaxado retorna uma distância entre o estado atual e o estado objetivo. Esta distância pode ser utilizada para estimar a distância real. O próximo passo a realizar é aquele que leva a um estado com a menor distância estimada.

O comportamento citado mostra o esforço com que o FF tenta tenta alcançar seu objetivo. Os resultados são diferentes dependendo do tipo do problema. O FF teve desempenho superior em todos os domínios de problemas do "3rd International Planning Competition" que ele participou[4]. No entanto ele falha para problemas mais difíceis como o SAT e sokoban.

O detalhamento de cada um dos passos realizados será descrito a seguir.

2 Relaxações

2.1 Ignorar as *delete lists*

Uma forma de relaxar o modelo é simplesmente ignorar as *delete lists*. O problema resultante certamente tem uma solução ótima com custo menor do que o problema original. Ou seja, é admissível. Infelizmente o problema relaxado continua sendo $\mathcal{NP}-$difícil pois o problema de cobertura por conjuntos pode ser reduzido polinomialmente para o seguinte problema de planejamento:

fatos um para cada elemento do conjunto

estado inicial vazio

objetivo todos os fatos

ações cada subconjunto do problema original determina uma ação cuja *add list* contém todos os fatos correspondentes aos elementos do subconjunto e a *delete list* é vazia. Nenhuma ação contém pré-requisitos.

O número de ações necessárias para se atingir o objetivo é igual ao número de subconjuntos utilizados na cobertura. Por este motivo é preciso encontrar uma estimativa mais fácil de ser computada.

2.2 HSP

Uma forma de se aproximar o custo da solução é estimar individualmente a dificuldade de se obter cada fato que se está procurando. O planejador HSP (no qual o FF é baseado) define os seguintes pesos

- O peso de um fato presente no estado inicial é 0
- O peso de um conjunto de fatos é a somatória dos pesos dos fatos deste conjunto

- O peso de qualquer fato que não pertença ao estado inicial é inicialmente ∞. Depois, para cada ação e para cada fato f na *add list* desta ação calcule

$$\text{peso}(f) = \min(\text{peso}(f), \text{peso}(\text{pré-condições da ação}) + 1)$$

  Repita iterativamente até que os pesos não variem mais.

Um problema desta heurística é que ela não é admissível. Considere as seguintes ações

ação 1 Sem pré-condições. Adiciona o fato f_1

ação 2 f_1 é a única pré-condição. Adiciona o fato f_2

ação 3 f_1 é a única pré-condição. Adiciona o fato f_3

Se o objetivo for um estado com f_2 e f_3 então a seqüência de ações $1, 2$ e 3 resolve o problema com custo 3. No entanto, o peso de f_2 e de f_3 é 2 e portanto a estimativa heurística é 4.

2.3 FF

A heurística utilizada no FF consiste em encontrar uma solução para para o problema relaxado (sem as*delete lists*). Espera-se que esta solução tenha um custo pequeno.

Para isso, primeiro se resolve um problema ainda mais simples: permite-se que várias ações sejam executadas simultaneamente. Continuando com o exemplo acima, o FF realizaria primeiramente a ação 1 e depois a 2 e a 3 *simultaneamente*.

Mais formalmente o algoritmo consistem em, a partir do estado inicial, aplicar simultaneamente todas as ações que tenham suas pré-condições atendidas no estado corrente e, com isto, obter o estado seguinte. A iteração continua até que o estado objetivo seja um subconjunto do estado atual.

Os estados produzidos até o momento serão utilizados como camadas na resolução do problema relaxado. Parte dos fatos de cada camada são escolhidos como objetivos da mesma:

- Os objetivos da última camada são os objetivos do problema original.
- Se um objetivo da camada i é um fato que já está presente camada $i-1$ ele é acrescido na lista de objetivos de $i-1$. Caso contrário, escolhe-se alguma ação cujos pré-requisitos estão em $i-1$ e que produza este fato. Os pré-requisitos desta ação são acrescidos à lista de objetivos de $i-1$.

Após o termino desta varredura pode-se seqüenciar as ações dentro das camadas de qualquer forma pois todos os pré-requisitos são atendidos pela camada anterior.

O tamanho deste caminho pode ser maior do que o ótimo do problema original, logo esta heurística também não é admissível. No entanto ela costuma produzir resultados menores do que a do HSP. Continuando o exemplo anterior vemos que temos três camadas. A primeira é vazia, a segunda contém f_1 a terceira contém f_1, f_2 e f_3. Os objetivos de cada camada são:

1ª nada

2ª f_1

3ª f_2 e f_3

Entre a primeira e a segunda camada a única ação que poderia ter sido escolhida é a 1. Entre a segunda e a terceira temos as ações 2 e 3. Temos então duas soluções possíveis para o problema relaxado ($1, 2, 3$ e $1, 3, 2$). Observe que intuitivamente o FF se sai melhor do que o HSP por não considerar a obtenção dos fatos f_2 e f_3 como coisas independentes.

Esta técnica foi derivada do planejador Graphplan. A principal diferença é que o Graphplan não ignora as *delete lists* e tenta conseguir uma solução para o problema original diretamente da solução relaxada (com ações simultâneas).

3 Ações Úteis

O planejador FF utiliza o conceito de "ações úteis" que, em conjunto com o *Enforced Hill Climbing*, acelera a procura. As ações úteis são definidas como aquelas que adicionam ao menos um objetivo à camada seguinte a inicial. O objetivo desta técnica é tentar seguir a mesma seqüência de passos determinada pelas camadas da solução do problema relaxado. Em uma definição formal:

$$\mathrm{H}(S) = \{o \mid \mathrm{pre}(o) \subseteq S, \mathrm{add}(o) \cap G_1(S) \neq \emptyset\}$$

No qual $\mathrm{H}(S)$ é o conjunto das ações úteis, G_1 é o conjunto dos objetivos definidos para à camada seguinte à inicial.

Estas ações, extraídas do plano relaxado, ajudam a identificar os sucessores que parecem ser mais úteis e também a detectar a ordem dos objetivos. No entanto, isto é apenas uma heurística e não um corte válido. Considere um problema com os fatos A, B e P_A e as ações

$op1$ pré-requisitos $= \emptyset$, add $= \{A\}$ e del $= \{B\}$

$op2$ pré-requisitos $= \emptyset$, add $= \{P_A\}$ e del $= \emptyset$

$op3$ pré-requisitos $= \{P_A\}$, add $= \{A\}$ e del $= \emptyset$

Se o estado inicial for $\{B\}$ e o objetivo for $\{A, B\}$ então a única solução é seqüência $op2, op3$. No entanto, ao removermos as *deltete lists* é possível encontrar uma solução de custo 1. Logo, A se torna um dos objetivos de G_1 e apenas $op1$ será considerada uma ação útil.

4 Método de busca: *Enforced Hill Climbing*

Um dos grandes problemas dos planejadores é a análise dos espaços de estado. Apesar das heurísticas serem de tempo polinomiais, as buscas ainda são muito custosas devido ao tamanho do espaço.

O HSP tenta resolver o problema usando o método *Hill Climbing* com escolha aleatória do melhor sucessor.

Já o FF utiliza uma busca em largura (*BFS — Breadth First Search*) para encontrar o sucessor. Esta busca funciona buscando-se dentre todos os vizinhos um que forneça um valor heurístico melhor que o estado atual S. Caso não se encontre, aprofunda-se no segundo nível (os vizinhos dos vizinhos) e assim sucessivamente até que se encontre um estado S' que seja melhor. O caminho de S a S' é adicionado ao plano e prossegue-se a partir de S'. A está técnica da-se o nome de *Enforced Hill Climbing*.

Sabe-se que o comportamento de um algoritmo de busca local depende da estrutura do problema a ser resolvido. O *Enforced Hill Climbing* não se perde em platôs e mínimos locais mas não é garantido que ele encontre uma solução em casos que tenham becos sem saída. Porém, Hoffmann et al. estimam que tais problemas sejam poucos nos casos presentes nos *benchmarks* sobre planejamento. Por isso está técnica é adequada para esta tarefa.

5 Comparação

Cabe, agora, uma avaliação dos resultados do FF para mostrar se todo o seu trabalho é capaz de se sair melhor que seus antecessores. Para isso, é possível mostrar a comparação com o "pai" do FF: o planejador HSP-1.0. O FF baseado neste planejador de tal forma que, se retirarmos dele o *enforced hill climbing* (e trocarmos pelo *hill climbing* simples), as ações úteis (e mantivermos todas as ações) e não utilizarmos o paralelismo no problema relaxado para calcular as distâncias estimadas, teremos o algoritmo HSP-1.0. A tabela a seguir mostra, para um grupo de domínios de problemas, uma comparação entre o FF e um simulado do HSP gerado a partir do FF desligando-se os melhoramentos citados.

A partir da tabela, pode se concluir empiricamente que o planejador FF tem melhoramento expressivo na maioria dos domínios citados. Análises isoladas das melhoras de cada uma das três características extras do FF também foram realizadas e a partir delas foram concluídas as seguintes observações:

- O uso de distância estimada do graphplan (resolução relaxada com paralelismo) melhorou os resultados sempre que a distância estimada do HSP (somatório das distâncias simples) retornava valores superestimados. Isso se deve à boa detecção de dependência entre os objetivos do graphplan.
- A busca *enforced hill-climbing* só funciona bem quando as ações úteis estão ativadas (isto é, um algoritmo com *ehc* mas sem *au* seria ruim).
- A busca *enforced hill-climbing* é mais lenta que a *hill-climbing*, pois esta, quando perdida num máximo local, escolhe saídas aleatórias. O tamanho

Problema	HSP	FF
Assembly	117.39	16.94
Blocksworld-3ops	4.06	6.11
Blocksworld-4ops	0.60	40.65
Briefcaseworld	16.35	150.00
Bulldozer	4.47	141.04
Freecell	65.73	41.44
Fridge	28.52	2.77
Grid	138.06	11.73
Gripper	2.75	0.11
Hanoi	93.76	2.70
Logistics	79.27	11.94
Miconic-ADL	150.00	59.00
Miconic-SIMPLE	2.61	0.56
Miconic-STRIPS	2.71	0.36
Movie	0.02	0.02
Mprime	73.09	26.62
Mystery	78.54	86.21
Schedule	135.50	13.77
Tireworld	135.30	85.64
Tsp	4.11	0.07

Figura 1: Comparação entre HSP-1.0 e FF [3]. Tempos em segundos. Para os tempos iguais a 150.00, o planejador foi interrompido sem uma solução

do plano, porem é melhor na primeira, por ser mais metódica nessas regiões difíceis

- As ações úteis melhoram significativamente o tempo e o tamanho da solução, mesmo quando os cortes são pequenos. O sucesso das ações úteis no planejador mostram que a probabilidade delas chegarem ao objetivo são altas.

O planejador FF parece ser o planejador ideal até que se perceba em que domínios ele funciona bem. Os domínios citados, na realidade, são domínios de teste e, na tentativa de classificá-los, percebe-se que eles são muito simples. De fato, o FF foi feito específicamente para resolver esses tipos de problemas: problemas com máximos locais pequenos e com passos reversíveis. Para confirmar isso de uma vez, os autores resolveram aplicar o uso do FF em problemas não triviais. Para o problema SAT, foram criadas diversas instâncias aleatórias e foram comparados os planejadores FF, blackbox e ipp. O resultado foi óbvio: o FF não foi capaz de resolver problemas maiores que poucas dezenas de variáveis e teve resultado inferior aos outros planejadores.

6 Conclusão

O planejador Fast Forward pode ser visto como um planejador que busca resolver os passos não determinísticos da busca com um conjunto de heurísticas relativamente simples e com um ideal bem definido: correr com muita sede ao pote. Notando que este é tanto o seu ponto forte como seu ponto fraco, dependendo do problema, percebemos seu verdadeiro objetivo: o planejador FF é um planejador especializado em vencer competições de planejamento de inteligência artificial.

Conclui-se, de um modo um tanto mais respeitoso ao vencedor da competição de planejamento de 2002, que o algoritmo é excelente para resolver problemas com bastante reversibilidade de ações (ou seja, problemas em que pode se errar um pouco no meio do caminho) e que tenham reconhecidamente máximos locais curtos. Além disso, acredita-se que o algoritmo parece ter bom potencial de desenvolvimento no caminho de vencer *dead ends*, bastando para isso utilizar buscas um pouco mais conservadoras. Por exemplo, seria útil que o estilo *hill-climbing* de busca não fosse levado tão a sério e, de tempos em tempos, o algoritmo poderia se preocupar com um pouco de *backtracking*. Trabalhos realizados pelo autor foram criados para identificar algumas classificações que os problemas podem se encaixar e algoritmos novos poderiam controlar a velocidade de subida utilizando essas informações. Enfim, o grupo acredita que o FF é um planejador de bom potencial para uso genérico e revela, pela sua velocidade, como é útil dar valor a planos que tentam, desde o começo, mostrar resultados e crescer rapidamente.

Referências

[1] Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 1636–1642, 1995.

[2] Blai Bonet, Gabor Loerincs, and Hector Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 714–719, Providence, Rhode Island, 1997. AAAI Press / MIT Press.

[3] J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search, 2001.

[4] Drew McDermott. 3rd international planning competition, 2002. http://planning.cis.strath.ac.uk/competition/.