Planejamento em IA - Problema do Mundo dos Blocos

relatório

Equipe:
Crislânio Macêdo

Sumário

1 - Introdução

Na seção 1.1 é feita uma introdução rápida do problema tratado, na seção 1.2 é descrito como se pretende resolver o problema, e os elementos envolvidos para a resolução do mesmo. Na seção 2.2 é descrito a estratégia para resolução do problema, e mostrado o algoritmo para resolver o problema. Na seção 3.1 é analisado os resultados de tempo de execução do problema para cada instancia do problema e por último é feito algumas considerações a respeito da estratégia usadas.

1.1 - O que é o Problema.

O Mundo dos Blocos é um dos domínios de planejamento mais famosos em IA. Imagine um conjunto de blocos colocados em uma mesa. O objetivo é construir um ou mais pilhas verticais de Blocos. Pode-se pegar apenas um bloco por vez, e colocá-lo na mesa ou em cima de um outro bloco. Só pode ser retirado o bloco no topo da pilha.

A realização das ações é feita através de um 'braço' robótico, com as seguintes ações.

Represesentações:

Ações do Agente

pick-up (?x - block)	o agente pega um bloco de cima da mesa e o segura
put-down (?x - block)	o agente coloca o bloco que está segurando em cima da mesa
stack (?x - block ?y - block)	o agente coloca o bloco que está segurando em cima de outro bloco
unstack (?x - block ?y - block)	o agente pega um bloco de cima de outro bloco e o segura

1.3 - Planejamento da Resolução.

Foi feito análises dos arquivos que continham o plano de ações para se fazer a procura por uma solução para se atingir um estado final a partir de um estado inicial.

exemplo:

blocks-4-0.strips

...

unstack_b_d

on_b_d;clear_b;handempty

holding_b;clear_d;~clear_b;~handempty;~on_b_d

unstack_d_b

on_d_b;clear_d;handempty

holding_d;clear_b;~clear_d;~handempty;~on_d_b

clear_c;clear_a;clear_b;clear_d;ontable_c;ontable_a;ontable_b;ontable_d;handempty

on_d_c;on_c_b;on_b_a

Foi criado um parser para a leitura de arquivos.strips com o objetivo de ler as ações (nome da ação, precondição,efeito) e verificar em cada estado(formado com preposições) se satisfaz o estado final (no exemplo acima: on_d_c;on_c_b;on_b_a)

2 - Estratégia

A estratégia abordada para se chegar a satisfazer o estado final, foi utilizar uma lista de ações(nome da ação, precondição, efeito), uma fila com os estados e outra com os índices dos estados, e uma árvore no qual cada nó pode ter qualquer quantidade de filhos e é possível fazer o caminho de retorno de um nó filho até a raiz, recuperando o conjunto de ações da raiz até esse filho.

2.1 - Algoritmo

Foi criado dois estados para pegar as ações iniciais e finais. E uma lista de ações, além de uma fila para guardar os estados e outra os índices. Logo em seguida era necessário verificar as ações (nome da ação, precondição, efeito, ou seja ler a cada três linhas). Era necessário agora verificar se a cada estado satisfazia o estado final (de aceitação), assim que enquanto o estado atual (primeiro da fila) não conter o estado final como subconjunto, ou seja, enquanto não satisfazemos o estado final, coletamos todas as ações possíveis a partir do estado atual e se as precondições da ação atual estão todas satisfeitas pelo estado atual passa a ser uma é uma ação possível.
Para cada ação possível criamos um estado idêntico ao estado atual, e logo em seguida modificamos o novo estado buscando satisfazer as pós condições (efeito) desta ação. É mantida uma fila com os estados e os ids, a solução é obtida através de um retorno de caminho na árvore, chamamos de returnpath (que retorna o caminho retorno de um nó filho até a raiz, recuperando o conjunto de ações)

3 - Resultados

Analisamos os arquivos .strips com as respectivas instâncias. blocks-4-0, blocks-4-1, blocks-5-0, blocks-5-1, blocks-6-0. Os resultados dos tempos recorrentes são descritos logo abaixo.

3.1 - Tempo Gasto Para Geração do Plano

Foi executado as instancias uma vez e logo em seguida novamente. Os resultados são um pouco curioso dado a divergência em alguns casos de melhora em tempo significante.
Foi usado um computador com as seguintes configurações:
Marca: Acer Aspire E 14
processador: Intel core i3 (1.7 GHz, 3MB L3 cache).
memória: 4GB DDR3 L, 500 GB HDD.

blocks-4-0: 0.85 e 0.79 segundos.
blocks-4-1: 0.36 e 0.37 segundos.
blocks-5-0: 15, 34 e 16,13 minutos.
blocks-5-1: 7,47 7,84 minutos.
blocks-6-0: 5,96 horas.
blocks-6-1: 5,96 horas.

Foi executado o arquivo blocks-10.0.strips por mais de 3 horas (e por motivos técnicos preferiu-se não executá-lo novamente) foi feito uma análise ‘heurística’ e notou-se que o tempo decorrido para 10-0 instâncias seria inviável.
As ações geradas pelos testes foram.
blocks-4-0:

['pick-up_b']
['stack_b_a']
['pick-up_c']
['stack_c_b']
['pick-up_d']
['stack_d_c']

blocks-4-1:

['unstack_b_c']
['put-down_b']
['unstack_c_a']
['put-down_c']
['unstack_a_d']
['stack_a_b']
['pick-up_c']
['stack_c_a']
['pick-up_d']
['stack_d_c']

blocks-5-0:

['unstack_c_e']
['put-down_c']
['pick-up_d']
['stack_d_c']
['unstack_e_b']
['put-down_e']
['unstack_b_a']
['stack_b_d']
['pick-up_e']
['stack_e_b']
['pick-up_a']
['stack_a_e']

blocks-5-1: 7,47 7,84 minutos.

['unstack_b_a']
['put-down_b']
['unstack_a_d']
['stack_a_e']
['pick-up_b']
['stack_b_a']
['pick-up_c']
['stack_c_b']
['pick-up_d']
['stack_d_c']

blocks-6-0:

['unstack_d_a']
['put-down_d']
['unstack_f_e']
['stack_f_d']
['unstack_e_b']
['stack_e_f']
['unstack_a_c']
['stack_a_e']
['pick-up_b']
['stack_b_a']
['pick-up_c']
['stack_c_b']

blocks-6-1: 5,96 horas.

3.2 - Considerações

Com as realizações dos testes foi observado que o algoritmo para pequenas instâncias executou bem, mas para instâncias maior do que 6 é inviável. Uma curiosidade vista foi a 'discrepância' relativa entre os tempos de execução para um mesmo arquivo, dado que o mesmo foi executado imediatamente depois. O maior problema encontrado foi que dado um espaço de busca muito grande o algoritmo utilizado não funcionará bem, sendo preciso fazer uma 'busca' nas soluções com mais eficiência por exemplo, uma busca aleatória dentre as possíveis ações possíveis.
Foi observado a dificuldade envolvida em se fazer um planejamento geral, principalmente se para a resolução do problema for usado busca pois tem um fator de ramificação grande (para instâncias maiores pode ficar inviável), tem uma dificuldade maior em se descartar ações a priori, considera ações a partir de um estado atual uma após outra e o objetivo é testado para cada estado; para cada novo estado, um novo teste idêntico precisa ser feito. Assim que é necessário alguma outra heurística ou forma de resolver problemas de planejamento de maneira mais eficiente.

para mais informações acesse:
http://www.crislaniomacedo.zz.vc/2016/03/planejamento-em-inteligencia-artificial-mundo-dos-blocos/