Planejamento Automático: SALVEM AS GIRAFAS

Bruna Prauchner Vargas[1] **João A. L. de Moraes Júnior**[1]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

{bruna.prauchner,joao.adalino}@acad.pucrs.br

1. Introdução

O segundo trabalho consiste em escrever um novo domínio de planejamento em PDDL. Depois disso, deve-se criar diversos arquivos de problemas para o domínio elaborado, de dificuldade crescente, para avaliar a escalabilidade do planejador utilizado. Os planejadores utilizados neste trabalho são

- PDDL4J
- Fast Downward
- Web Planner
- Planning Domains

2. Descrição do problema

O objetivo do trabalho é salvar girafas fugitivas do zoológico de Madagascar. O domínio é composto por locais, agentes e girafas. As girafas apenas são salvas com a ajuda dos agentes e elas passam por diferentes situações de perigo. Os riscos que elas correm são

- Afogamento
- Perder-se na mata
- Incêndio

Os agentes podem exercer certas funções, são elas

- Salva-vidas
- Resgatador
- Bombeiro
- Zelador

Dessa forma, definiu-se que cada instância de agente possui apenas uma função. Uma girafa que está se afogando só pode ser salva por um salva-vidas, uma que está perdida na mata só será salva por um resgatador, e a que estiver em um incêndio é preciso de um bombeiro para salvá-la. Somente um zelador pode levar as girafas para o zoológico.

3. PDDL

3.1. Predicados

No domínio, os predicados têm o agente e todas as funções que ele pode exercer; todos os locais; a girafa e as situações que ela pode estar; tem-se também um predicado para dizer onde o agente ou girafa está. E o predicado para dizer que a girafa está salva é objetivo de todos os problemas.

3.2. Ações

As principais ações

- Mover
  Com esta ação é possível mover o agente(seja qual for sua instância) para qualquer um dos locais.
- Salvar a girafa
  É a ação do zelador levar a girafa de volta pro zoológico e ela tem que estar completamente em segurança.
- Mudar para
  Nesta ação, o agente pode se transformar nas funções que ele exerce.
- Salvar do(a)
  O agente executa a função que está exercendo, como por exemplo, salvar uma girafa do incêndio.
- Levar pro hospital
  Quando uma girafa estiver com dor ou machucada, o enfermeiro(uma instância de agente) leva ela ao hospital.
- Cirurgia
  Quando uma girafa está machucada, é preciso fazer uma cirurgia, para isso um médico e um enfermeiro são necessários.

3.3. Problemas

O primeiro problema é extremamente simples em que tem-se um agente e duas girafas fora do zoológico, a solução esperada é que o agente se transforme em um zelador e leve as girafas de volta. Ao longo dos problemas, as dificuldades vão crescendo. Já no problema 4, tem-se somente 1 agente e existem 6 girafas, dentre elas, 2 estão em um incêndio, 3 estão se afogando e 1 está perdida na floresta. O objetivo de todos os problemas é que as girafas estejam sãs e salvas no zoológico. Como todas as ações tinham custo 1, por padrão dos planejadores, o agente não tinha conhecimento de que uma girafa que está se afogando deveria ser resgatada primeiro que uma perdida na floresta. Ao pesquisar mais sobre PDDL viu-se que era possível colocar duração nas ações, ou alterar seu custo, entre outras alternativas.

3.4. Planjeadores usados

Qual é a sequência de tarefas que cada agente deve realizar para cumprir a missão de salvar todas as girafas? Existem muitas alternativas para resolver cada um dos problemas e é por isso que ferramentas computacionais são essenciais. Um dos jeitos de responder a essa questão é utilizando um planejador PDDL.

3.5. Hardware Utilizado

Dados sobre o computador utilizado para realizar os experimentos.

Tabela 1. Dados sobre o computador usado

	Computador
Processador	Intel core i7
Placa de vídeo	NVIDIA® GeForce®
Memória RAM	8GB
Sistema operacional	14.04.1-Ubuntu

3.6. Fast Downward

É um planejador de progressão, ou seja começa no estado inicial e segue determinando as ações aplicáveis, na direção pra frente. Esse planejador não usa diretamente a representação PDDL proposicional de uma tarefa de planejamento. A entrada é traduzida para uma representação alternativa que é conhecida como "tarefas de planejamento de valores múltiplos", o que torna explícita muitas das restrições implícitas de uma tarefa de planejamento proposicional [1].O Fast Downward venceu a faixa "clássica"(isto é, proposicional, não otimizadora) da 4ª Competição Internacional de Planejamento no ICAPS 2004. A instalação não é simples e os problemas demoravam bastante para serem executados.

3.7. PDDL4J

Fácil instalação e os problemas foram resolvidos em pouco tempo. O objetivo deste planejador é facilitar o desenvolvimento de ferramentas JAVA para planejamento automático usando a linguagem PDDL.

3.8. Web Planner e Planning Domains

Ambos eram sites na internet em que podia fazer o *upload* do arquivo de domínio e do problema. Simples e sem complicações. Executaram todos os problemas muito rápido, porém, a solução frequentemente tinha mais ações que os outros planejadores.

4. Resultados Obtidos

Usando busca A*(lmcut) no planejador Fast Downward.

Tabela 2. Fast-Downward A*(lmcut)

Problema	Custo do Plano	Estados Expandidos	Número de Estados Registrados	Tempo Total	Memória Usada
1	5	6	13	0.00109497s	3720 KB
2	9	24	42	0.00132497s	3720 KB
3	15	126	205	0.00317908s	3720 KB
4	31	6726	11296	0.224201s	4276 KB
5	23	21925	50768	1.1997s	6232 KB
6	24	12798406	39725258	32 minutos	2185572 KB
7	11	30	43	0.00113013s	3720 KB
8	14	820	1099	0.0151422s	3852 KB
9	35	15294579	23256305	22 minutos	1207888 KB
10					

Usando outras duas heurísticas, tentou-se executar pelo menos dois problemas com ela, porém, ao tentar com o problema 6, que é considerado um dos mais complexos, ela não acha a solução por falta de memória, como mostra a tabela 3.

Tabela 3. Fast Downward usando A*(ipdb)

Problema	Custo do Plano	Estados Expandidos	Número de Estados Registrados	Tempo Total	Memória Usada
3	15	16	47	0.127054s	0,003988 GB
6					3,6 GB

Como é mostrado, a partir do problema 4, não consegue-se mais gerar uma solução usando essa heurística por falta de memória. O mesmo acontece quando é usada a heurística *blind* em que procura sem saber nenhuma informação, gerando todos os casos.

Tabela 4. Fast Downward usando A*(blind)

Problema	Custo do Plano	Estados Expandidos	Número de Estados Registrados	Tempo Total	Memória Usada
3	15	285	308	0.00362001s	0,00372 GB
6					3,6 GB

Depois de realizar os testes usando esse planejador, todos os problemas foram rodados novamente usando PDDL4J na mesma máquina.

Tabela 5. Usando PDDL4J

Problema	Custo do Plano	Tempo Total	Memória Usada
1	5	0,13s	0,03 MBytes
2	9	0,13s	0,03 MBytes
3	15	0,14s	0,05 MBytes
4	31	1,10s	1,10 MBytes
5	23	1,51s	2,49 MBytes
6	24	103,4s	321,70 MBytes
7	11	0,16s	0,03 MBytes
8	14	0,26s	0,17 MBytes
9	35	237,47s	602,39 MBytes
10			

Comparando a tabela 2 com a 5, fica claro que o planejador Fast Downward usou mais memória e demorou muito mais tempo para entregar a solução dos problemas, isso acontece devido as heurísticas escolhidas.

Devido a complexidade do problema 10 ser alta, não foram encontradas soluções a tempo de entregar o trabalho usando Fast Downward ou PDDL4J, por isso, o plano de solução do problema 10 foi resolvido no *Planning Domains* conforme mostra a tabela 6.

Tabela 6. Planning Domains - Problema 10

	Nodos Gerados Durante a Busca	Nodos Expandidos Durante a Busca	Tempo Total
Problema 10	14204	548	0.432s

5. Heurísticas Usadas

As heurísticas de planejamento são usadas para evitar espaço de busca exponencial e não dispersar recursos com caminhos irrelevantes otimização de busca.

Utilizando busca local A*, sabe-se que a eficiência é ótima e só expande nós com

$$f(n) \leq f* \tag{1}$$

Onde f* é o custo do caminho ótimo.
Nenhum outro algoritmo ótimo garante expandir menos nós.

6. Animação Gráfica

Para uma melhor visualização do trabalho, foi desenvolvida uma interface gráfica em Unity com o intuito de animar a solução dos problemas, visto que apenas a solução que os planejadores retornam podem se tornar massantes de compreender.

Nesta interface é possível visualizar o domínio do trabalho, assim como os problemas criados.

A solução dos problemas é encontrada automáticamente em tempo real através do planejador PDDL4J. Todas as soluções são apresentadas na interface.

Com a solução pronta, é possível executar uma simulação da mesma, onde é disponibilizada uma animação do passo a passo resultante da solução, tornando o trabalho muito mais interativo.

7. Conclusão

Por fim, depois de estudar sobre planejamento[2], entender seus conceitos e aplicações percebeu-se que ao longo do trabalho, que alguns planejadores executavam mais rapidamente que outros e isso depende da máquina ou do servidor e da heurística usada. Em trabalhos futuros pretende-se aprimorar mais o conhecimento em planejadores, aprendendo como colocar a duração de uma ação, alterar o custo, e outras funções que podem ajudar a otimizar o domínio. A história inventada para esse trabalho representa a complexidade de escolher uma sequência de ações a serem seguidas em situações de risco.

Referências

[1] Malte Helmert. “The Fast Downward Planning System”. Em: *J. Artif. Int. Res.* 26.1 (jul. de 2006), pp. 191–246. ISSN: 1076-9757. URL: http://dl.acm.org/citation.cfm?id=1622559.1622565.

[2] Stuart Russell, Peter Norvig e Ernest Davis. *Artificial intelligence: a modern approach*. English. 3ª ed. Upper Saddle River, NJ: Prentice Hall, 2010.