Guião do Relatório de IA

Descrição do Problema

O Pipe Mania é um jogo cujo objetivo é construir um sistema de canalização funcional. Cada peça do tabuleiro pode estar orientada de diferentes maneiras e deve ser conectada corretamente às peças adjacentes. O tabuleiro é uma grelha quadrada e as peças pertencem a uma destas quatro categorias.

O desafio é rodar as peças de forma a que todas se conectem e a água flua sem interrupções ou fugas, não podendo existir partições no tabuleiro.

Abordagem para Resolver o Problema

O tabuleiro é representado como uma matriz onde cada célula contém uma peça.
Cada peça é identificada pelo seu tipo e orientação. São guardadas em dicionários, onde a chave é o nome da peça e o valor da chave corresponde a um tuplo representando a direção das conexões da peça. (exemplo slide)

Inicialmente é realizado um pruning (pré-processamento) onde colocamos parte das peças logo nas posições corretas. Caso a peça tenha mais do que uma orientação possível, esta é colocada numa lista de peças incompatíveis, ou seja, peças onde é necessário realizar uma ação para ficar na sua posição final. Podemos afirmar que estamos a considerar a Heurística do Maior Grau, pois priorizamos a colocação de peças com mais restrições de conexão, reduzindo o fator de ramificação e aumentando a eficiência.

Por exemplo, neste tabuleiro todas a peças têm de obedecer às seguintes restrições:

- Não pode haver ligações para fora
- Caso um dos seus vizinhos esteja na sua posição final e tenha uma conexão, este tem de estar ligado a ele. A mesma lógica aplica-se no caso contrário

Assim, há uma propagação de restrições, tornando os domínios das variáveis mais restritos.

Nas actions optamos por retornar as ações possíveis de cada peça porque assim não temos que expandir a árvore toda e vamos calculando as ações da peça no estado recebido, reduzindo um pouco a eficiência, mas melhorando o custo de memória. Uma ação é representada por um tri-tuplo, onde as primeiras duas entradas representam a posição no tabuleiro (linha, coluna), e a última a peça. O resultado de uma ação é colocar a peça no tabuleiro numa cópia do estado.

No teste objetivo, verificamos que as peças estão todas conectadas e se não existem fugas. Através de uma DFS, verificamos também a ausência de partições no tabuleiro.

Conclusão

Após testar o programa com os diferentes algoritmos, observamos que, na procura cega, a DFS não só é mais eficiente, como também exige menos memória, em comparação à BFS, tal deve-se ao facto de que são expandidos mais nós numa procura em largura primeiro.

A heurísticas testadas para os algoritmos de procura informada não foram suficientes para eliminar os pontos negativos associados a cada algoritmo.

Concluímos que entre os algoritmos de procura informada a procura gananciosa é a eficiente, porém, ao estimar o uso de memória, verificamos que não compensava utilizá-la, pois o custo era extremamente elevado em comparação com a procura e profundidade.