Relatorio Projeto de IA

Descrição do problema

O problema abordado consiste numa adaptação do jogo PipeMania, em que o objetivo é conectar vários tubos de maneira a formar um caminho continuo. Existem 4 tipos de peças diferentes e, assim como na versão original, nesta adaptação, por meio de um algoritmo de pesquisa, tem de se conectar as peças de forma a formar um caminho continuo.

Ideia geral para abordar o problema

Para resolver o problema, adotamos a ideia de que cada rotação de uma peça forma um estado novo, ou seja, uma versão ligeiramente do tabuleiro dado como input, em que a peça em questão foi rodada. Sendo assim, a ideia passa então por tratar o tabuleiro como um estado, onde cada ação possível é a uma rotação de uma peça numa determinada pipe, e a ideia é aplicar diferentes algoritmos de busca para encontrar uma sequencia de ações que levem a um estado objetivo, ou, por outras palavras, o tabuleiro final todo conectado.

Ações no contexto do problema

No contexto deste problema, uma ação é então uma rotação de uma peça em um pipe específico do tabuleiro. Cada rotação de uma peça permite fazer conexões com diferentes peças adjacentes e assim, é possível formar vários tabuleiros diferentes.

Resultado de aplicar uma ações no estado

O resultado de aplicar uma rotação a um estado, é então a criação de outro estado, ou seja, outro tabuleiro, em que a peça que foi rodada esteja noutra posição. Este método permite que haja novas conexões e assim torna mais fácil a busca pelo caminho continuo.

Pré processamento e métodos de inferência

Neste projeto, decidimos implementar um conjunto de métodos de inferência para determinar as ações possíveis com base nos tubos adjacentes a um pipe especifico . Estes métodos ajudaram a restringir o conjunto de ações válidas, considerando apenas as rotações que poderiam potencialmente levar a uma solução, economizando assim tempo e recursos computacionais durante a busca. Aqui estão apresentados alguns desses métodos:

1. Deteção de Conexões Válidas

Os métodos de inferência verificam se as conexões entre tubos adjacentes são válidas. Para isso, foram definidas quatro conjuntos de tubos com base nas suas conexões:

- **lig_esq**: Tubos que podem se conectar à esquerda.
- **lig_dir**: Tubos que podem se conectar à direita.
- **lig_cima**: Tubos que podem se conectar acima.
- **lig_baixo**: Tubos que podem se conectar abaixo.

2. Ações Baseadas nas Conexões Adjacentes

Vários métodos foram implementados para determinar as ações possíveis com base nas conexões dos tubos adjacentes. Estes métodos consideram as diferentes combinações de conexões (ou a falta delas) nas direções cima, baixo, esquerda e direita.

- **top_connect_no_left_actions**: Determina as ações possíveis quando há uma conexão na parte superior, mas não na esquerda.
- **top_left_connect_actions**: Determina as ações possíveis quando há conexões nas partes superior e esquerda.
- **top_left_no_connect_actions**: Determina as ações possíveis quando não há conexões nas partes superior e esquerda.
- **left_connect_no_top_actions**: Determina as ações possíveis quando há uma conexão na parte esquerda, mas não na superior.
- **top_connect_nl_nr_actions**: Determina as ações possíveis quando há uma conexão na parte superior, mas não nas partes esquerda e direita.
- **top_left_connect_nr_actions**: Determina as ações possíveis quando há conexões nas partes superior e esquerda, mas não na direita.
- **top_connect_nl_nb_actions**: Determina as ações possíveis quando há uma conexão na parte superior, mas não nas partes esquerda e inferior.
- **top_left_connect_nb_actions**: Determina as ações possíveis quando há conexões nas partes superior e esquerda, mas não na inferior.
- **top_connect_nl_nb_nr_actions**: Determina as ações possíveis quando há uma conexão na parte superior, mas não nas partes esquerda, direita e inferior.
- **top_left_connect_nb_nr_actions**: Determina as ações possíveis quando há conexões nas partes superior e esquerda, mas não nas partes direita e inferior.

- **nt_nl_nr_actions**: Determina as ações possíveis quando não há conexões nas partes superior, esquerda e direita.
- **left_connect_nt_nr_actions**: Determina as ações possíveis quando há uma conexão na parte esquerda, mas não nas partes superior e direita.
- **nt_nl_nb_actions**: Determina as ações possíveis quando não há conexões nas partes superior, esquerda e inferior.
- **left_connect_nt_nb_actions**: Determina as ações possíveis quando há uma conexão na parte esquerda, mas não nas partes superior e inferior.
- **nt_nl_nb_nr_actions**: Determina as ações possíveis quando não há conexões nas partes superior, esquerda, direita e inferior.
- **left_connect_nt_nb_nr_actions**: Determina as ações possíveis quando há uma conexão na parte esquerda, mas não nas partes superior, direita e inferior.

3. Aplicação das Ações

Com base nas conexões válidas e nos métodos de inferência, a função actions retorna uma lista de ações possíveis que podem ser executadas a partir de um estado específico. Cada ação é representada por uma rotação de uma peça em um pipe específico. Esses métodos garantem que apenas ações que podem levar a uma solução válida estejam a ser considerados.

4. Verificação de Objetivos

A função goal_test utiliza a inferência para verificar se todas as peças estão conectadas corretamente formando um caminho contínuo. Ela usa uma DFS para verificar que todas as peças estão conectadas, garantindo que todas os pipes estejam visitados e conectados corretamente.

Avaliacao experimental

FALTA ISTO PORQUE JÁ NÃO ME LEMBRO QUAIS É QUE UTILIZAMOS SEM SER A ASTAR

Escolha da Busca Final

A função de pesquisa que acabou por ser a escolhida foi a DFS. As principais características dessa função incluem a exploração completa de cada ramo até o final antes de voltar e explorar outro ramo. Esta escolha foi feita porque a DFS pode ser mais eficiente em termos de memória para instâncias menores e pode rapidamente encontrar uma solução viável