Projeto Sokoban IA

Disciplina: Inteligência Artificial - 3º ano 1º Semestre

Curso: Licenciatura em Engenharia Informática

Diogo Filipe Amaral Carvalho 92969 Participação: 1/2

Rafael Ferreira Baptista 93367 Participação: 1/2

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Descrição do Algoritmo

1. Primeiramente após a leitura de um novo mapa construímos um conjuto de sets que irão ser utilizados na pesquisa. De seguida, é efetuada uma pesquisa inicial do tipo KeeperSearch (pesquisa que procura o caminho de uma dada posição inicial do Keeper até uma posição objetivo) em que lhe é passado a posição inicial do Keeper e um objetivo que é impossível. Através do conjunto de all_nodes que é verificado obtemos então todas as posições válidas do mapa.
2. De seguida, através de uma função de deteção de deadlocks estáticos, obtemos um conjunto com todas as posições deadlock do mapa. Obtemos ainda o conjunto de posições que dividem o mapa em partes distintas. Todos estes conjuntos são utilizados na pesquisa.
3. Nesta altura executamos uma fórmula que tem por base o número de caixas do nível e a heurística do estado inicial por forma a decidir se iniciamos uma pesquisa do tipo BoxSeach com uma estratégia A* ou com uma estratégia Greedy.
4. A estratégia de pesquisa passa pela separação da pesquisa geral BoxSearch cujo os estados são uma lista em que no índice 0 está o conjunto das caixas e no índice 1 a posição atual do keeper e as ações consistem na movimentação de uma caixa no mapa, da pesquisa KeeperSearch(já explicada anteriormente) que é utilizada para saber se o Keeper consegue alcançar uma determinada posição num dado estado.
5. Ambas as funções search das árvores de pesquisa seguem uma estratégia de pesquisa idêntica à que foi estudada na aula prática. Apresentam como diferenças o facto de apenas criarem um novo estado caso este não exista não só no caminho do estado atual até à raiz, mas sim em toda a árvore de pesquisa. Para além disto, na função search da arvore relativa à pesquisa BoxSearch, os estados são guardados sob a forma de um valor hash por forma a acelerar a verificação da sua existência.

Descrição do Algoritmo (cont.)

1. Relativamente à pesquisa BoxSearch a função actions do domínio que determina as possíveis ações num dado estado vai inicialmente fazer uma pesquisa do tipo KeeperSearch passando novamente um objetivo impossível mas desta vez tem em conta as posições atuais das caixas, para detetar todas as possíveis posições que o Keeper consegue atingir naquele estado.
2. De seguida, para cada caixa e para cada direção é verificada se é possível efetuar um determinado movimento com base em todos os sets que já foram abordados anteriormente, onde é verificada a existência de deadlocks não só estáticos mas também dinâmicos, i.e, que dependem das posições das outras caixas. Apenas ações que não provoquem deadlock são consideradas. É ainda efetuada uma verificação com base na existência de tuneis, isto é, locais onde as caixas só se deslocam num sentido, sendo considerados vários tipos de túneis (verticais, horizontais, que dividem o mapa ao meio).
3. Relativamente à função de result, tendo por base o estado atual e uma ação origina um novo estado. No que diz respeito à heurística utilizada nesta pesquisa, a mesma é dada pelo total das distâncias de Manhattan entre as caixas que não se encontrem em goals e os goals que ainda não estão preenchidos.
4. Relativamente à pesquisa KeeperSearch a função actions do domínio que determina as possíveis ações num dado estado verifica quais as posições adjacentes à atual estão livres para o Keeper se dirigir. A heurística é a distancia de Manhattan entre a posição atual e o objetivo.

Resultados obtidos

- Os resultados obtidos foram: o agente conseguiu passar todos os níveis até ao 146 (não inclusive), sendo que até ao nível 131 conseguiu encontrar uma solução em poucos segundos na maior parte dos casos, e a partir do nível 131 teve mais dificuldade a encontrar uma solução em alguns níveis, sendo o mais demorado o 131 que demorou cerca de 100-120 segundos (Tempo dos nossos dois computadores).
- Conseguimos também obter melhores soluções, i. e, relativas ao número de pushes, quando efetuamos uma pesquisa A*. Este fator permitiu-nos obter uma pontuação melhor, visto que a maioria dos níveis são ultrapassados de forma instantânea, pelo que com um menor número de pushes conseguimos melhor pontuação final.

Conclusão

- Concluindo, consideramos que o desenvolvimento do nosso projeto foi bastante positivo, uma vez que conseguimos aplicar os conhecimentos obtidos tanto nas aulas práticas como nas aulas teóricas num projeto que possuía uma maior dificuldade do que aquela que já tínhamos experienciado nas aulas práticas.
- Um dos pontos que achamos que poderia ter sido melhor efetuado, está relacionado com a limpeza, redução de complexidade de compreensão e falta de documentação do nosso código. Para além disso consideramos que a heurística utilizada na BoxSearch poderia ser melhor visto que é bastante simples e não muito eficaz em níveis com um maior número de caixas.
- À parte disso e tendo em conta a tabela de scores, consideramos que foi um projeto bem conseguido visto que o nosso agente ficou relativamente bem posicionado.