Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
SCC0230 - Inteligência Artificial
Profa. Solange Oliveira Rezende

Projeto 1 - Sokoban

Bernardo Maia Coelho 12542481
João Gabriel Sasseron Roberto Amorim 12542564
João Pedro Buzzo Silva 10425191
Marcos Patrício Nogueira Filho 11819063
Pedro Guilherme dos Reis Teixeira 12542477
Rogério Lopes Lübe 10770113

Outubro de 2023

1 Sokoban

O tema escolhido para o projeto é o *Sokoban*: um jogo de quebra-cabeças japonês, criado nos anos 80. O jogo consiste em um tabuleiro com posições bem definidas e alguns elementos fundamentais:

- Jogador: pode se mover e empurrar caixas espalhadas pelo tabuleiro (porém não pode puxar uma caixa).
- Caixa: objeto movido pelo jogador.
- Botão: fixo no tabuleiro e não oferece resistência aos movimentos do jogador e da caixa.
- Parede: limita os movimentos do jogador e da caixa.

O objetivo principal é mudar a posição das caixas até que todos os botões estejam pressionados por uma caixa e, assim, finalizar o nível.

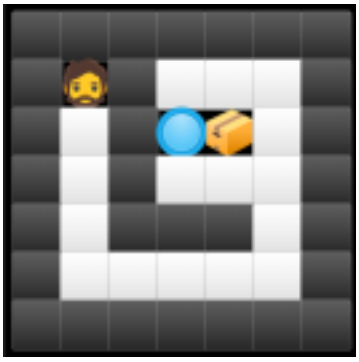

Figura 1: Exemplo de tabuleiro de *Sokoban*.

Em termos de computação, apesar de não parecer num primeiro momento, o *Sokoban* é um problema complexo de se resolver. De fato, ao se considerar os movimentos do jogador, deve ser possível voltar um movimento e, para tabuleiros maiores, as possibilidades crescem significativamente, sendo esse categorizado como um problema P-SPACE completo.

2 Modelagem

Para o projeto, estamos interessados em modelar o jogo de *Sokoban* de forma que não seja necessário um jogador para solucionar o nível. Usaremos um algoritmo de busca para encontrar uma solução ótima para o jogo e

mostrar o caminho a ser feito para que o nível possa ser finalizado. Portanto, devemos modelar o espaço de estados para efetuar uma busca.

A modelagem escolhida foi tratar cada configuração do tabuleiro como um estado. Assim, quando o jogador se move em um espaço, temos um novo estado. O estado final depende de cada botão estar sendo pressionado por uma caixa.

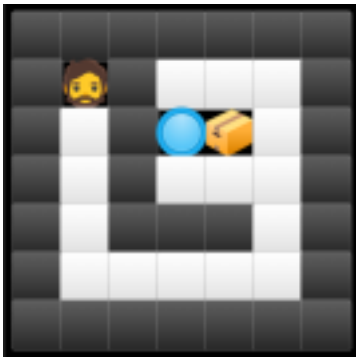

Figura 2: Exemplo de estado inicial.

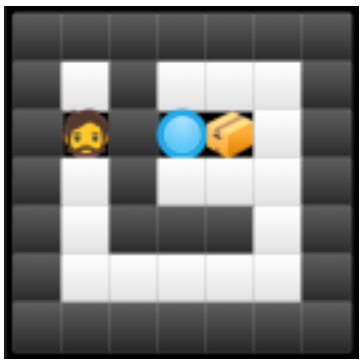

Figura 3: Estado imediatamente após movimento do jogador.

Além disso, em virtude da complexidade do problema tratada na seção anterior, optou-se por modelar apenas uma caixa e um botão ao buscar pela solução.

3 Implementação

Para resolver o problema do *Sokoban*, utilizamos duas estratégias: o *Breadth-First Search* e o *A* Search*.

3.1 Breadth-First Search (BFS)

A nossa primeira estratégia é o algoritmo BFS, uma estratégia de busca não informada. A escolha desse algoritmo em relação ao *Depth-First Search* (ou DFS) se deu por conta da complexidade do problema e da rápida expansão da sua árvore de estados. Além disso, o DFS apenas garante encontrar uma solução – não necessariamente a solução ótima. No entanto, estamos também interessados na solução ótima do jogo, que pode ser encontrada utilizando o BFS.

3.2 A* Search

A nossa segunda estratégia é o algoritmo A*, uma estratégia de busca informada. Esse algoritmo se destaca por garantir uma solução ótima, dado que a heurística escolhida para avaliar o espaço de estados seja admissível – isto é, a avaliação seja menor ou igual ao custo real para se chegar à solução. Portanto escolhemos uma função heurística composta por 3 avaliações distintas e a função de custo:

- $h_1(n)$ - A *distância de Manhattan* da caixa até o botão;
- $h_2(n)$ - A *distância de Manhattan* do jogador até a caixa;
- $h_3(n)$ - Uma função corretiva: caso o jogador esteja em uma posição adjacente à caixa, a estimativa é decrementada em 1; caso o jogador, a caixa e o botão estejam alinhados em uma mesma linha ou coluna, a estimativa é decrementada em 2;
- $g(n)$ - Custo associado à transição de um estado para outro, que no caso é 1 para qualquer transição.

Pensando apenas nas avaliações 1 e 2, dado que a *distância de Manhattan* é uma heurística admissível, elas são, portanto, também admissíveis. No caso da avaliação 3, ela tem por finalidade atribuir uma maior prioridade a estados próximos à caixa, pois eles devem se aproximar mais de uma solução. Além disso, a avaliação 3, no pior caso, mantém a estimativa igual à *distância de Manhattan* das avaliações 1 e 2 (que é admissível) e, em outros casos, apenas diminui o valor da estimativa, obedecendo o critério de admissibilidade da heurística.