3º ano do Mestrado Integrado de Engenharia Informática e Computação, 2014/2015

Inteligência Artificial

Box World 2

Relatório Intercalar

Francisco Lopes, 201106912  
Miguel Mendes, 201105535

18/04/2015

**Objetivo:**

O objetivo deste trabalho consiste em desenvolver um programa que demonstre inteligência suficiente para resolver o problema em questão. Focando-nos no jogo “Box World 2”, pretendemos que, em vez de ser o jogador a resolver os puzzles propostos, que o computador seja capaz de o fazer autonomamente, encontrando caminhos através dos mesmos, e empurrando quaisquer blocos necessários para que tal seja possível.

**Especificação:**

1. **Análise**

O nosso trabalho é baseado no jogo “Box World 2”. Neste jogo, o objetivo do jogador é simplesmente chegar à saída, desta forma conseguindo avançar para o nível seguinte. No entanto, os caminhos para a saída poderão estar bloqueados por buracos que impeçam a travessia até ao objetivo. Para resolver isto, existirão vários blocos que podem ser deslocados pelo jogador, e colocados nestes buracos para que chegar ao objetivo seja ultimamente possível. O desafio neste jogo reside no quão complexos poderão ser os movimentos do jogador, em congruência com a colocação destes blocos, para que este consiga resolver os vários puzzles. Além dos blocos mais comuns, que apenas movem uma unidade quando o jogador os empurra, ainda existe uma segunda categoria de blocos, blocos de gelo, que se deslocam ininterruptamente na direção em que são empurrados até que encontrem uma parede ou bloco.

O nosso objetivo consiste em desenvolver inteligência para o programa, de forma a que não seja necessário um jogador, e o computador seja capaz de resolver os vários puzzles apresentados, presumindo que existe solução, respeitando todas as regras implícitas ao correr do jogo.

Tendo em conta as prioridades deste trabalho, pretendemos começar pelo desenvolvimento e implementação do algoritmo A\* que se adapta à resolução do problema, inicialmente apenas tendo em conta a existência do tipo de blocos mais comum. Assim que possuirmos um algoritmo estável que funcione de forma previsível, com resultados corretos, serão introduzidos os blocos de gelo e, se possível, o algoritmo será reajustado para que a resolução dos vários puzzles se mantenha correta. Finalmente, iremos desenvolver o ambiente gráfico para que toda a aplicação seja minimamente apresentável. Partindo do princípio que nesta etapa o algoritmo já se comporta de forma correta, sem falhas e com todas as condições de aresta testadas, concentrar-nos-emos apenas no melhoramento da componente gráfica do programa.

Considerando especificamente a aplicação do algoritmo, consideramos que será vantajosa a preferência de inserção de blocos em buracos, pelo que iremos dar o custo a este tipo de movimento um valor negativo. Considerando que a única heurística até agora considerada se baseia na distância entre o nó atual e a saída, um valor negativo na inserção de blocos em buracos irá garantir uma preferência do algoritmo por este tipo de escolha. No entanto, como a colocação de blocos em buracos irá alterar de forma um pouco significante a estrutura do puzzle a ser resolvido, consideramos melhor a reanálise do puzzle e reaplicação do algoritmo, para ter as alterações realizadas em consideração. Inicialmente, pretende-se encontrar o menor caminho possível até ao objetivo com o mínimo número de buracos. Após a seleção de blocos a colocar em cada buraco pode-se aplicar o algoritmo, não ao jogador, mas ao bloco. Fazendo parte da heurística a condição de que um movimento de um bloco possua um espaço vazio no sentido oposto ao seu movimento, assim garantido que serão preferenciais ao algoritmo movimentos em que seja possível o jogador se colocar detrás do bloco a empurrar para que a ação seja possível.

1. **Abordagem**

O algoritmo principal no qual o desenvolvimento deste projeto se foca é o A\*. Este algoritmo é vastamente utilizado para encontrar caminhos, principalmente em situações onde o deslocamento se efetua em grelhas bidimensionais, como é o caso. Cada nó da grelha, que poderá ser também um nó do algoritmo, tem um valor f(x), resultante da soma de dois outros valores. O primeiro, g(x), será correspondente à distância do nó inicial ao nó que se analisa na altura. O segundo valor, h(x), consiste na função heurística através da qual realizamos uma previsão da distância que será necessário percorrer do nó atual até ao objetivo.

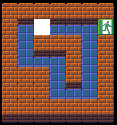
Partindo do nó inicial, calcula-se o f(x) dos seus vizinhos. Aquele que possuir o menor f(x), ou seja, o maior potencial de nos levar pelo caminho mais curto, terá os seus vizinhos analisados, que se juntam ao grupo de nós não selecionados até agora, de forma a que se encontre de novo o nó com o menor valor de f(x), e assim sucessivamente, até o nó selecionado ser o objetivo.

**Desenvolvimento:**

A nossa aplicação foi desenvolvida em C++, em Windows, com uma maioritária utilização de Visual Studio, fazendo uso da biblioteca SFML para a criação da interface gráfica.

A maior parte de todo o processo encontra-se concentrado no módulo relativo ao mapa, que alberga não só as estruturas que representam o problema no ecrã, assim como as funcionalidades relacionadas como algoritmo responsável pela resolução do problema. Este módulo contém também o módulo do “jogador”, que neste caso permitirá a representação do progresso da inteligência, que tenta resolver o problema proposto. Outro módulo importante será o módulo do nó, estrutura de conhecimento principal utilizada durante o algoritmo para criar a árvore que se torna crucial na procura da solução.

**Experiências:**

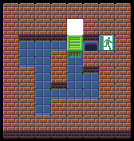
****Inicialmente, as experiências, têm que ser básicas, para confirmação da plena funcionalidade do algoritmo A\*. Neste exemplo, demonstramos como apresentamos ao programa a possibilidade de perseguir dois caminhos diferentes para chegar ao seu destino. Fora error técnicos iniciais que fariam com que o programa tomasse caminhos inválidos, confirma-se que o programa é capaz de selecionar o caminho mais curto. Neste caso, desloca-se para a direita.

**Conclusões:**

Através das experiências levadas a cabo, entendemos que o algoritmo implementado de A\* é perfeitamente capaz de encontrar o caminho mais curto para o objectivo de qualquer labirinto, posto que este tenha uma solução possível.

**Melhoramentos:**

Devido a condições fora do nosso controlo, não foi possível implementar mais do que a capacidade do programa de levar a cabo a viagem mais curta entre quaisquer dois pontos selecionados no mapa. Os algoritmos necessários à inclusão de caixas está existente. Apenas seria necessário adaptar a componente gráfica para que esta restante funcionalidade fosse suportada.

 Durante a maioria do trabalho foram apenas consideradas situações em que a colocação de caixas era bastante linear. De futuro, seria vantajoso considerar e testar situações em que o movimento de caixas necessita ser algo mais complexo. Como exemplo, temos a situação demonstrada na imagem, na qual seria necessário deslocar a caixa de forma a realizar uma espécie de ciclo, para que o único buraco que bloqueia a saída fosse eliminado.