Manual Tecnico

Blokus Uno


```
Disciplina - Inteligencia Artificial
Turma - SW-01
Realizadores - Joana Guerreiro 202001733 | Andreia Novas - 201400498
Docente - Filipe Mariano
```


Índice

Introdução

A aplicação desenvolvida em Common Lisp, para a Unidade Curricular de Inteligência Artificial, é baseda numa versão simplificada do jogo Blokus, denominada Blokus Uno. Esta primeira parte do projeto tem como objetivo encontrar a uma solução para umdeterminado problema,preenchendo o tabuleiro existente com um numero de espaços minimos com um certo numero maximo de peças á sua disposição. Em cada movimento com uma peça deverá através da procura em espaços de estados, que será baseado num critério de menor custo ou caminho. O utilizador escolhe dentro das opções que aparecem no menu, qual o problema e o algoritmo a utilizar, em que estes mesmos problemas são lidos diretamente de um ficheiro. Assim que o jogo termine será gravado no ficheiro, e será mostrado no ecrã o resultado conseguido ao longo do jogo pelo utilizador.

Arquitetura do sistema

Para uma melhor manutenção, organização, e compreensão do programa desenvolvido, a aplicação foi dividida em três ficheiros lisp. Estes são denominados por puzzle, procura e projeto. Além destes ainda existe o ficheiros problemas.dat, que contém os problemas, e o resultados.dat, que guarda o resultado do jogo.

Procura

O ficheiro denominado por **"procura"** tem toda a parte do desenvolvimento de funções auxiliares, de associação aos algoritmos e de procura em espaço de estados. Como :

- Algoritmo de Procura de Largura Primeiro (BFS)
- Algoritmo de Procura do Profundidade Primeiro (DFS)
- Algoritmo de Procura do Melhor Primeiro (A*)

Puzzle

O ficheiro denominado por **"puzzle"** tem toda a parte do desenvolvimento referente a tudo o que esteja interligado com o domínio da aplicação, toda a manipulação da

escolha das peças dos movimentos e validações.

Projeto

O ficheiro denominado por **"projeto"** tem as restantes funções, como de interação com o utilizador,funções de teste como auxilio aos testes aos algoritmos, iniciar a aplicação e funções de I/O.

##Problemas O ficheiro denominado por **"problemas"** tem todos os tabuleiros que seram dados no menu como hipóteses de jogo ao utilizador, para que este posso escolher um, e iniciar o seu jogo.

##Resultados O ficheiro denominado por **"resultados"** é gerado no final do jogo, com as caracteristicas do problema que foi resolvido, mostrando o algortimo escolhido, heuristica utilizada, profunidade, problema escolhido, resultados finais.Estes são gerados, expandidos, definida a penetrancia, o fator de ramificação, a profundidade da solução e o caminho do nó inicial até ao nó solução.

Entidades e sua implementação

Limitações tecnicas impostas para o desenvolvimento. A implementação tinha foco na recursividade, sem a possibilidade de exitir ciclo, na mesma. Ainda foram dadas limitações como o impedimento para utilização de sequênciação e funções com efeitos secundários.

Algoritmos e sua implementação

No desenvolvimento deste projeto, foram desenvolvidos algortimos de procura em espaço de estados, como o breadth first search (bfs), o depth first search (dfs) e o A*. Para obter um melhor desempenho dos algoritmos a resolver os problemas fornecidos. O desempanho dos algoritmos é verificado através do fator de ramificação,penetrância, número de nós gerados e expandidos e o tempo de execução (em milissegundos).

- **Penetrância**: Dá uma boa percepção do número de nós, desnecessários à resolução do problema que foramgerados até se encontrar o nó objetivo.
- **Número de nós gerados**: número de nós gerados desde nó inicial até ao nó solução. Quantos menos gerarmais rapidamente encontra a solução, mais eficiente é.
- **Número de nós expandidos**: número de nós expandidos desde nó inicial até ao nó solução. Quantos menos expandir mais rapidamente encontra a solução, mais eficiente é.
- **Tempo de execução**: Este é o número de milissegundos que passaram desde que o algoritmo começou atéque acabou. Quanto menor for este tempo, mais eficiente é. Contudo inicialmente podemos assumir que o A* será o mais eficiente, pois este resolve o problema, tendo em conta a explusão combinatória, senda a heurística utilizada indicada ao problema.

Descrição das opções tomadas

Ao longo do desenvolvimento do projeto deparamo-nos com alguns problemas, onde tivemos de tomar algumas decisões . Como por exemplo , em vez de desenvolvermos operadores por cada posição de peça optamos por criar operadores referentes aos vertices possiveis. Outra decisão foi a nova heuristica que criamos, além de basear o algoritmo a* na

heuristica dada , criamos uma melhor heuristica baseada em que cada peça e adiciona-mos lhe um peso , ou seja :

- Quadrado grande -> consideramos o mais importante então o valor dela é 3
- Peça S -> será mediana pois mesmo tendo a mesma dimensão é necessário executar mais operações, por isso o seu valor 2
- Quadrado pequeno -> tem a menos dimensão e gera vários espaços em branco com o seu valor 1

A heuristca será especificamente a soma da multiplicação do peso das peças ainda não jogadas , ou seja quanto menos tiver quadrados grandes melhor.

Limitações técnicas e ideias para desenvolvimento futuro

Encontrámos algumas limitações na realização de alguns algoritmos aplicados aos problemas, devido aspetos de limitação de memória do programa. Vamos em seguida especifica-los:

- Bfs -> só conseguiu resolver o primeiro problema corresponde a alinea a).
- Dfs -> não conseguiu resolver alguns problemas devido a uma baixa profundidade, por isso vamos especificar o minimo de profundidade que cada problema necessita para encontrar uma solução.
  - a) Primeiro problema - minimo de profundidade 2.
  - b) Segundo Problema - minimo de profundidade 5.
  - c) Terceiro Problema - minimo de profundidade 7.
  - d) Quarto Problema - minimo de profundidade 9.
  - e) Quinto Problema - minimo de profundidade 11.
  - f) Sexto Problema - minimo de profundidade 18.

Na nossa visão no futuro poderia ser interessante desenvolver os algoritmos bonus que seriam SMA, IDA e/ou RBFS.