FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

RELATÓRIO DE TRABALHO

Disciplina: Inteligência Artificial *(CC2006)*

Docente: Inês Dutra

Grupo: João Rodrigues up201405426

Pedro Almeida up201403403

Introdução

Um problema de pesquisa é composto por um conjunto de estados em que a partir de um estado inicial (problema) se vão gerando e se avaliam outros conjuntos até chegar ao estado final (objetivo). À medida em que é feita a geração de novos estados é feita a sua respetiva avaliação para que seja feita a melhor escola possível de cada estado a selecionar para a criação da próxima geração.

Neste tipo de problemas é muito usual usar-se métodos de armazenamento do tipo:

- *linked* lists;
- arrays;
- search trees;
- hash tables etc.

Métodos de pesquisa não guiada:

- Largura;
- Custo uniforme;
- Profundidade;
- Profundidade iterativa;
- Bidirecional;

Métodos de pesquisa guiada:

- *Greedy;*
- A*;

Estratégicas de procura

A *pesquisa em largura* (BFS) vai expandido todos os nós com a mesma profundidade e verificando em todos os níveis se é solução. Esta pesquisa exaustiva só passa para o próximo nível no fim de verificar todos os nós do nível atual.

A complexidade espacial é igual à temporal O (n^m), onde n é o fator de ramificação e m a profundidade máxima.

A *pesquisa em profundidade* (DFS) vai expandido de forma recursiva, ou seja, vai expandido um ramo até á profundidade máxima desse mesmo ramo e chegando á profundidade máxima desse ramo retrocede para profundidades inferiores expandido os ramos que ainda não foram visitados até encontrar solução.

A complexidade espacial é O (n * m), onde n é o fator de ramificação e m a profundidade máxima.

A complexidade temporal é dada por O(n^m) onde onde n é o fator de ramificação e m a profundidade máxima.

A *pesquisa iterativa limitada* em profundidade (iDFS) tem um funcionamento idêntico como a DFS mas como o próprio nome indica existe um limite que vai aumentado enquanto não for encontrada uma solução, ou seja, quando um ramo chega à profundidade limite não vai ser mais expandido verificando os ramos com profundidade menor ou igual ao limite estabelecido, não encontrando solução nesse limite incrementa 1 e volta a procurar com novo limite até chegar à solução pretendida ou até gerar todos os nós possíveis.

A complexidade espacial é O (n * m), onde n é o fator de ramificação e m a profundidade máxima.

Pesquisa guiada

A *pesquisa greedy* (“gulosa”) como o próprio nome indica é uma pesquisa guiada que escolhe sempre a “melhor” opção a tomar sem olhar para trás nem reverter decisões tomadas descartando as consequências negativas que alguma decisão possa gerar.

Vai ser sempre escolhido o nó mais perto do objetivo final e de seguida é expandido utilizando-se uma função que calcula o custo entre o nó atual e a solução final.

A direção está sempre dependente do custo dos nós adjacentes ainda não visitados. O custo é nos fornecido pelo somatório das distâncias entre cada célula do nosso nó atual até a respetiva célula no nosso nó final (distância de Manhattan).

A *pesquisa A estrela* (A*) é também uma pesquisa *greedy* que segue a procura tendo como objetivo minimizar o máximo possível o custo de cada nó gerado, mas ao contrário da *greedy* nesta pesquisa o custo é nos fornecido pelo somatório das distâncias entre cada célula do nosso nó atual até a respetiva célula no nosso nó final mais a profundidade do nó atual. Desta forma permite comparar os nós que tem o mesmo custo, mas que estão em profundidades diferentes, escolhendo assim aquele que tem menor heurística.

Descrição da implementação

A nossa linguagem escolhida foi o C++ não só porque é a linguagem que nos sentimos mais confortáveis a trabalhar, mas também porque nos permite gerir melhor a memória a ser utilizada que ao longo deste trabalho nos foi bastante útil para poder analisar a evolução do nosso programa.

As nossas estruturas foram criadas obedecendo a dois critérios principais: complexidade e facilidade da manipulação dos dados e como tal escolhemos as seguintes estruturas para a resolução deste problema:

- Uma estrutura a qual demos o nome de data, que é constituída por uma matriz de inteiros, um inteiro com a profundidade do nó, o custo do nó, a posição do nosso zero no eixo dos x e a posição do nosso zero no eixo y respetivamente. Utilizamos também uma variável char para guardar a ultima direção e um apontador a referir o pai e outro a referir o filho.

- Uma *hastable* com o nome de *table* que contém como chave uma matriz convertida em *string* e um inteiro permitindo um fácil e rápido acesso ao nosso nó e verificar se este já foi visitado ou ainda está por visitar.

- Uma *queue* com o nome de *bfs*, onde se insere no fim e se retira do inicio da nossa *queue* o nó a ser processado facilitando assim no processamento e desenvolvimento do nosso *BFS*.

- Uma *stack* com o nome de *dfs*, onde se remove e insere do topo da nossa *stack* o nó a ser processado facilitando assim no processamento e desenvolvimento do nosso *DFS* e *iDFS*.

- Uma *priority queue* para a pesquisa *greedy* e A*, pois o C++ permite colocar na *priority queue* por ordem de custo.

Resultados

Ver em anexo.

Conclusão

Após uma conclusiva análise dos resultados obtidos pelo nosso programa conseguimos perceber que os algoritmos mais eficazes são os algoritmos de pesquisa guiada segue-se o iDFS, BFS e por fim o DFS.

Este trabalho ajudou a perceber melhor os mecanismos dos vários tipos de pesquisa e a saber quando utilizá-los. Sem dúvida que vai ser um grande alicerce para nos ajudar a resolver este tipo de problemas na vida real.