Universidade Federal de São Carlos  
Departamento de Computação  
Inteligência Artificial

Trabalho 1

Prof. Lúcia Machado Rino

Turma A

Cristiano Faustino

Matheus Takata

Pedro Henrique B. Zago

São Carlos – SP  
2013

**Indicação do problema eleito**

Neste trabalho, abordou-se o Sliding-Tile Puzzle. Nele, o jogador dispõe de três fichas brancas e três pretas, em um espaço com sete posições cujo objetivo é deixar todas as fichas brancas à esquerda e as pretas à direita. É possível deslizar uma ficha para um espaço vazio adjacente ou pular sobre uma ou duas fichas para uma posição vazia.

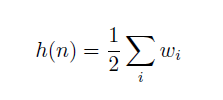
**Definição de f1 e f2**

As funções heurísticas utilizadas foram:

**f1:** Considera-se, para o cálculo de h1(n) a subtração das fichas pretas das brancas situadas nas quatro primeiras posições do tabuleiro, podendo assumir valores negativos se a quantidade de fichas pretas for maior. A esse resultado, é somado g1(n), que considera o peso de cada jogada da raiz até o nó analisado. Esse peso tem valor igual a 1 para arrastar e pular uma peça e igual a 2 se pular duas peças.

**f2:** Utiliza-se metade da soma do número de peças brancas à direita de cada peça preta *wi*, sendo que ao menos um movimento sobre uma peça preta é possível. Dividindo o valor da soma pela metade garante que o resultado não seja superestimado, uma vez que uma peça branca pode pular duas peças pretas com um único movimento.

O cálculo de h2(n) é dado por:



Ao valor de h2(n), soma-se g2(n), que considera o peso de cada jogada da raiz até o nó analisado, assim como g1(n).

**Métodos de Busca e Solução**

Descrição breve dos métodos de busca de solução - direção, tipo de atualização da open, ...

Os métodos de

**Ambiente e Instruções de Uso**

O programa foi desenvolvido na IDE Visual Studio 2012 professional na linguagem C#, utilizando o Monogame, uma implementação *open-source* do *framework* XNA da Microsoft para desenvolvimento de jogos que tem como objetivo permitir que os desenvolvedores deste framework possam produzir jogos para as diversas plataformas existentes hoje em dia, como Linux, Sony PlayStation 3, OUYA com o mesmo código utilizado para desenvolver os jogos para as plataformas da Microsoft.

Para executar o programa, será necessário instalar o Visual Studio 2012 em uma plataforma Windows (como o programa foi desenvolvido no Windows 7, deve-se dar preferência a esta plataforma). Além disso, é necessário instalar o Monogame, cujo instalador já está inserido no arquivo .rar. Instalado o *framework*, é necessário executar o arquivo oalinst.exe que se encontra no diretório “C:\Program Files (x86)\MonoGame\v3.0”. É importante observar que o diretório C: e Program Files (x86) são diretórios que variam de acordo com a partição e a versão do Windows instalado.

Feito isso, abra execute o arquivo GameName1.sln localizado dentro da pasta GameName1. Com o ambiente de desenvolvimento aberto, basta apertar F5 para executar o jogo. Durante a execução, é possível acompanhar cada passo que o robô executou até chegar no resultado apertando a tecla Espaço. Para fechar o programa, basta apertar a tecla de Escape(Esc). No código enviado, o programa executará a heurística do primeiro robô. Para acompanhar a heurística do segundo robô, é necessário fazer algumas mudanças no código, mais especificamente no arquivo LogicaIA.cs, sendo elas:

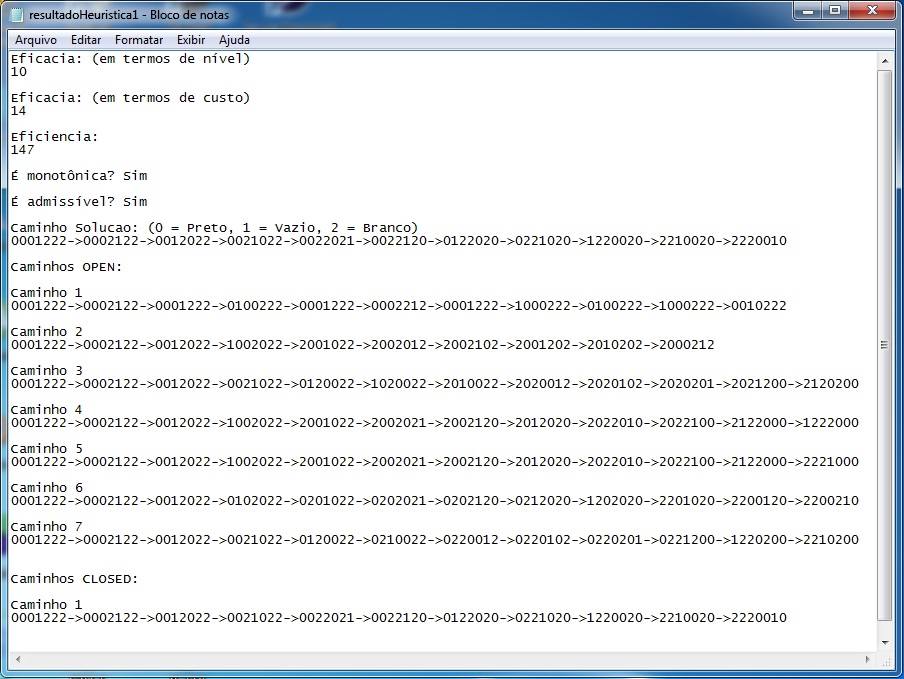
* Na linha 30, trocar “calculaHeuristica1” por “calculaHeuristica2”;
* Na linha 177, trocar “calculaHeuristica1” por “calculaHeuristica2”;
* Na linha 383, trocar “calculaHeuristica1” por “calculaHeuristica2”;

Para gerar os resultados no arquivo correto é necessário trocar, na linha 244, StreamWriter("resultadoHeuristica1.txt") por StreamWriter("resultadoHeuristica2.txt").

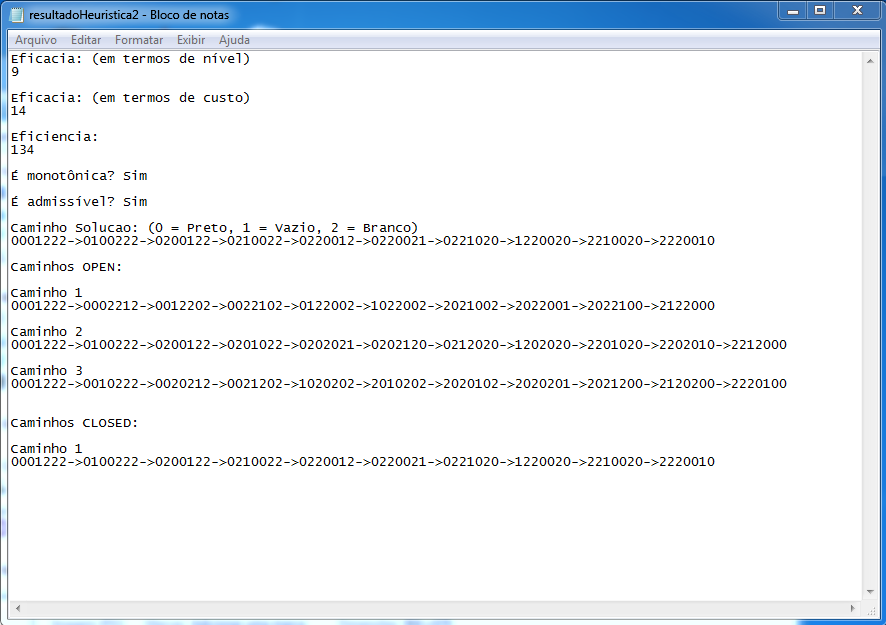
**Resultados**

Os resultados gerados por ambos os robôs foram escritos em arquivo, que acompanha o pacote de execução do programa. Nas figuras, são exibidos os *dumps* de tela com o conteúdo dos arquivos correspondentes a cada heurística.

Resultado – Robô 1:



Resultado – Robô 2:



**Comparação entre os robôs**

Analisando as propriedades dos robôs, constatou-se que ambos são monotônicos, o que implica que os mesmos são admissíveis. Do ponto de vista do peso das jogadas, ambos possuem o mesmo custo. Por outro lado, o caminho de solução encontrado pelo robô 2 é mais curto, o que caracteriza um custo menor em relação ao tamanho do caminho. Sendo assim, considera-se que o robô 2 é o mais informado.

O modelo de raciocínio do primeiro robô analisa as quatro primeiras posições e compara o número de peças brancas em relação às pretas nesse espaço. É deduzido que, quanto maior o número de peças brancas, mais promissor é o caminho.

Já o segundo robô compara a quantidade de peças brancas à direita das pretas no tabuleiro. Quão menor for esse valor, mais próximo da configuração desejada o nó está e, portanto, é considerado o mais promissor.

Deficiencias e potencialidades

Desdobramento dessa pratica