Grupo Nº 21



**Inteligência Artificial**

1.º Semestre 2015/2016

**IA-Tetris**

Relatório de Projecto

Índice

[1 Implementação Tipo Tabuleiro e Funções do problema de Procura 3](#_Toc436330760)

[1.1 Tipo Abstracto de Informação Tabuleiro 3](#_Toc436330761)

[1.2 Implementação de funções do problema de procura 3](#_Toc436330762)

[2 Implementação Algoritmos de Procura 4](#_Toc436330763)

[2.1 Procura-pp 4](#_Toc436330764)

[2.2 Procura-A\* 4](#_Toc436330765)

[2.3 Outros algoritmos 4](#_Toc436330766)

[3 Funções Heurísticas 5](#_Toc436330767)

[3.1 Heurística 1 5](#_Toc436330768)

[3.1.1 Motivação 5](#_Toc436330769)

[3.1.2 Forma de Cálculo 5](#_Toc436330770)

[3.2 Heurística n 5](#_Toc436330771)

[4 Estudo Comparativo 6](#_Toc436330772)

[4.1 Estudo Algoritmos de Procura 6](#_Toc436330773)

[4.1.1 Critérios a analisar 6](#_Toc436330774)

[4.1.2 Testes Efectuados 6](#_Toc436330775)

[4.1.3 Resultados Obtidos 6](#_Toc436330776)

[4.1.4 Comparação dos Resultados Obtidos 6](#_Toc436330777)

[4.2 Estudo funções de custo/heurísticas 6](#_Toc436330778)

[4.2.1 Critérios a analisar 6](#_Toc436330779)

[4.2.2 Testes Efectuados 6](#_Toc436330780)

[4.2.3 Resultados Obtidos 6](#_Toc436330781)

[4.2.4 Comparação dos Resultados Obtidos 7](#_Toc436330782)

[4.3 Escolha da procura-best 7](#_Toc436330783)

# Implementação Tipo Tabuleiro e Funções do problema de Procura

## Tipo Abstracto de Informação Tabuleiro

Para representar o tipo tabuleiro optámos por usar um array bidimensional. A escolha desta estrutura de dados advém da simplicidade em operações, de criação e modificação em comparação a outras alternativas, como por exemplo nas listas, onde estas operações não seriam tão triviais.

## Implementação de funções do problema de procura

A função accoes é responsável por devolver uma lista com todas as accoes possíveis que podem ser feitas com a próxima peça a ser colocada. Para tal, verifica todas as rotações válidas e todas as posições onde estas podem ser colocadas.

A função resultado vai retornar um novo estado que resulta de aplicar a accao recebida ao estado original. Primeiro verifica em que linha vai colocar a peça da accao, criando um estado resultante da aplicação da peça no tabuleiro do estado inicial. De seguida verifica se o tabuleiro esta preenchido até ao topo, se não estiver verifica se existem linhas totalmente preenchidas e remove-as do tabuleiro, atualizando a pontuação de acordo com a quantidade de linhas removidas.

# Implementação Algoritmos de Procura

## Procura-pp

A nossa procura em profundidade primeiro foi implementada de forma recursiva. Para isso começa por verificar se o nó atual é solução do problema, se for devolve as accoes necessárias ate chegar ao nó solução; se não for, vai expandir o nó actual e aplica uma acção ao estado actual criando assim outro estado. Repete enquanto não encontrar solução, caso não exista retorna uma lista vazia.

## Procura-A\*

Na procura A\* é guardado na fronteira não só os estados a ainda serem explorados como também as acções desde o estado inicial ate ao estado atual e o valor da função de avaliação associado. Cada vez que que é guardado um valor na fronteira é necessário reorganizar toda a fronteira. Como a fronteira está sempre organizada o próximo nó a ser explorado vai ser sempre o primeiro da fronteira.

## Outros algoritmos

# Funções Heurísticas

**NOTA IMPORTANTE:** As várias heurísticas apresentadas devem ter como base uma ideia diferente ou serem variações *relevantes* da mesma ideia, e não simples parametrizações da mesma. Heurísticas com base na mesma ideia que apenas diferem em pequenos factores *são passos na depuração da melhor heurística encontrada para ela e serão considerados como tal.*

## Heurística 1

### Motivação

Abordar de forma detalhada, *pelo menos*, os seguintes aspectos:

* Ideia inicial, informal, que levou à criação da heurística.
* Informação relevante contida no estado usada no cálculo da heurística.
* Processo de depuração que conduziu à forma final da heurística (outras variantes ensaiadas, etc.)

### Forma de Cálculo

Descrição *detalhada* da forma final da heurística (incluindo, eventualmente, uma fórmula que permita calcular o seu valor). É *muito importante* que esta descrição seja *concreta*, de modo a que outra pessoa consiga reproduzir a heurística se necessário. Deve também ser *clara* de modo a que qualquer pessoa que tenha lido a motivação da heurística compreenda a sua forma de cálculo.

Esta descrição deve ser complementada com exemplos da sua aplicação, recorrendo a figuras ou outros meios similares que tornem *perceptível* e *imediata* a compreensão da ideia subjacente à heurística.

**.**

**.**

**.**

## Heurística n

# Estudo Comparativo

NOTA: Embora nas várias secções seguintes seja pedida a descrição dos testes efectuados, não é necessário repetir essa descrição se os testes já tiverem sido descritos anteriormente, bastando referir onde esta descrição se encontra.

Pretende-se, neste projecto, efectuar quer os algoritmos de procura quer as heurísticas implementadas para o jogo IA-Tetris.

## Estudo Algoritmos de Procura

O primeiro estudo foca-se em comparar as duas implementações pedidas para os algoritmos de procura, a procura em profundidade primeiro e a procura A\*. Caso tenham implementado outro algoritmo de procura (ex. procura local)[[1]](#footnote-1), deverão também compará-lo com as outras duas procuras.

### Critérios a analisar

Descrever quais os critérios a ser usados para comparar os algoritmos e justificar a sua escolha. Alguns exemplos de critérios possíveis: qualidade das jogadas, número máximo de jogadas conseguidas, nós visitados, tempo de execução.

### Testes Efectuados

Descrever os problemas utilizados para efectuar o estudo. Justificar a sua *relevância* para os critérios a comparar.

### Resultados Obtidos

Enumerar os resultados obtidos, sob a forma mais adequada (tabela e gráficos, se necessário).

### Comparação dos Resultados Obtidos

Analisar, caso a caso, os resultados, focando a diferença de desempenho dos vários algoritmos entre si. A análise não deve ser somente uma descrição dos resultados obtidos, mas sim uma análise dos factores que possam ter conduzido a esses resultados.

## Estudo funções de custo/heurísticas

O segundo estudo foca-se em comparar as várias funções de custo e heurísticas implementadas, e perceber qual o custo e heurística com melhores resultados para a qualidade do jogador. **Atenção:** os testes com custos e heurísticas devem ser todos feitos com o mesmo algoritmo de procura.

### Critérios a analisar

Descrever quais os critérios a ser usados para comparar as várias heurísticas e justificar a sua escolha.

### Testes Efectuados

Descrever os problemas utilizados para efectuar o estudo. Justificar a sua *relevância* para os critérios a comparar.

### Resultados Obtidos

Enumerar os resultados obtidos, sob a forma mais adequada (tabela e gráficos, se necessário).

### Comparação dos Resultados Obtidos

Analisar, caso a caso, os resultados, focando a diferença de desempenho dos vários custos e das várias heurísticas entre si. A análise não deve ser somente uma descrição dos resultados obtidos, mas sim uma análise dos factores que possam ter conduzido a esses resultados.

## Escolha da procura-best

Com base nos resultados obtidos nas duas comparações anteriores, devem descrever e justificar qual o algoritmo e respectivas funções custo/heurística para implementar a procura-best criando assim o vosso melhor jogador.

1. Atenção, para que fique claro, não é necessária a implementação de outros algoritmos de procura, mas caso o tenham feito, deverão descrevê-los e compará-los. [↑](#footnote-ref-1)