universidade de aveiro

Rush Hour

Mariana Andrade, 103823
Vicente Barros, 97787

Inteligência Artificial 2022/23

Algoritmos

- A escolha de Algoritmo é feita com base no tamanho do tabuleiro a jogar
- Para tabuleiros de tamanho inferior a 7 usou-se o Breadth First com custos pela sua velocidade de cálculo
- Para tabuleiros com tamanhos maiores o A* tem uma performance melhor com recurso a heurísticas do que o algoritmo anteriormente apresentado
- Ao receber um nível, é calculado através dos algoritmos de pesquisa acima descritos o melhor caminho possível. Caso não haja nenhum movimento aleatório é efetuado a jogada movimentando o cursor pelo caminho mais curto da localização atual, até ao centro do carro a movimentar. As estratégias usadas para lidar com o movimento aleatório de carros, crazy cars, são descritas mais à frente.

Heurísticas

1-Distância Saída

Foi a primeira heurística a ser testada, sendo a mais simples de todas, que simplesmente avalia a distância do carro vermelho à saída

2-Distância Saída + Carros Bloqueados

É a combinação de duas heurísticas mais simples que são a distância do carro vermelho à saída (parece do lado direito) e o número total de carros a bloquear

3-Pior caso

Tendo em conta a heurística da distância do carro vermelho (“A”), e o número de carros que bloqueiam este, podemos prever que no pior dos casos haverá o mesmo exato número de carros à frente de “A”. Pelo que, a heurística é dada por 2 vezes a distância à Saída

4-Heurística Avançada

A heurística que apresentou performance mais próxima ao breadth first para níveis de maior dimensão consiste na multiplicação do número de carros do tabuleiro pela distância do carro.

5 - Distância de Manhattan

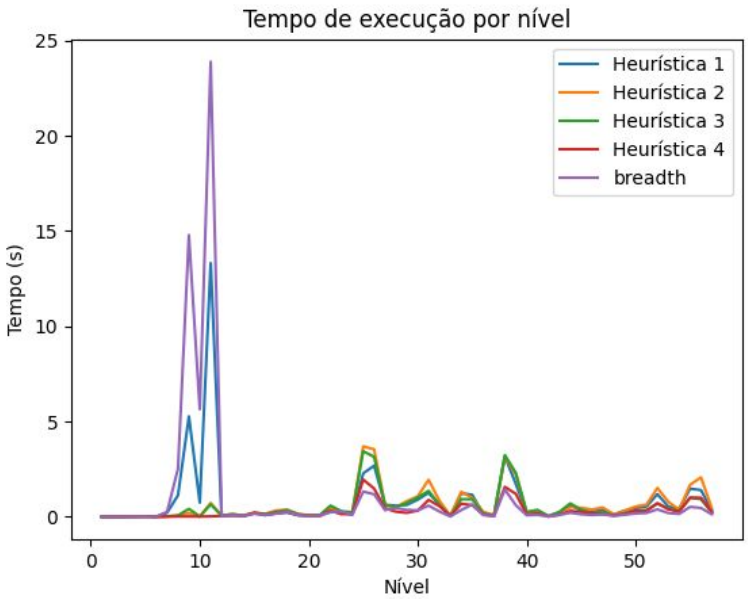
A distância de Manhattan define que a distância entre dois pontos é a soma das diferenças absolutas das suas coordenadas. Esta é usada para calcular custos e para os crazy cars não como heurística do A*

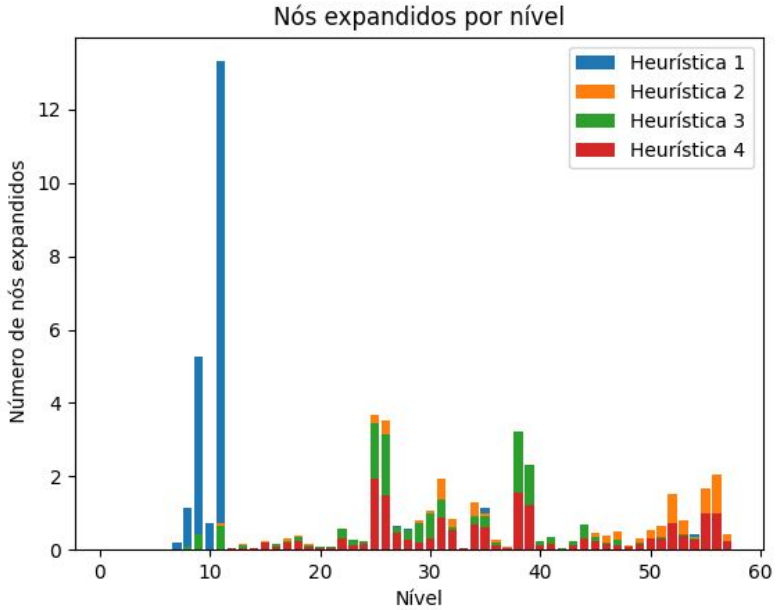
Crazy Cars

Para lidar com os crazy cars, foram testadas várias estratégias

- **Reposição:** Caso um carro se mova aleatoriamente, o agente volta a colocar o carro no sítio original para depois retomar a resolução do problema. A dificuldade que advém desta solução é que ao efetuar movimentos, a probabilidade de acontecer novamente crazy car é maior.
- **Thread:** Como forma de otimização à estratégia acima descrita foi implementada uma thread que quando um carro se move é calculado com um limite de tempo determinado pela game_speed e caso o cálculo não consiga ser efetuado dentro desse limite, a peça é reposta
- **Simulate:** Posteriormente foi implementada uma função para permitir ver se, através do movimento aleatório do carro é possível concluir o nível. Por razões de performance e de problemas de dessincronização com o servidor esta não foi usada para a implementação final, tendo ficado apenas comentada, bem como o local onde é usada.

Benchmarks





Os resultados usados para os gráficos podem ser encontrados aqui