RUSH HOUR

Inteligência Artificial 2022

Daniel Ferreira - 102442
Guilherme Antunes - 103600

RUSH HOUR

Algoritmo e Heurísticas

- Para a resolução do problema, é feita uma pesquisa em árvore recorrendo ao algoritmo de pesquisa A* (função *search* do módulo *test.py*).
- Nesta implementação, são utilizadas funções auxiliares modulares, nomeadamente:
  - *default_heurisitic*: implementa a heurística utilizada na resolução do problema estático (sem *crazy cars*) - **distância euclidiana** do carro alvo ao destino e o **número de carros que lhe bloqueiam** o caminho
  - *default_check_goal:* implementa a condição de satisfação geral - presença do carro alvo na última coluna
- Para além destas, são também empregues outras funções auxiliares que manter-se-ão independentemente do contexto:
  - *reconstruct_path:* recolhe o conjunto de estados a seguir para atingir a solução encontrada
  - *neighbors:* permite encontrar os vizinhos do nó atual, sendo de notar a utilização do *yield* como meio de evitar operações fúteis

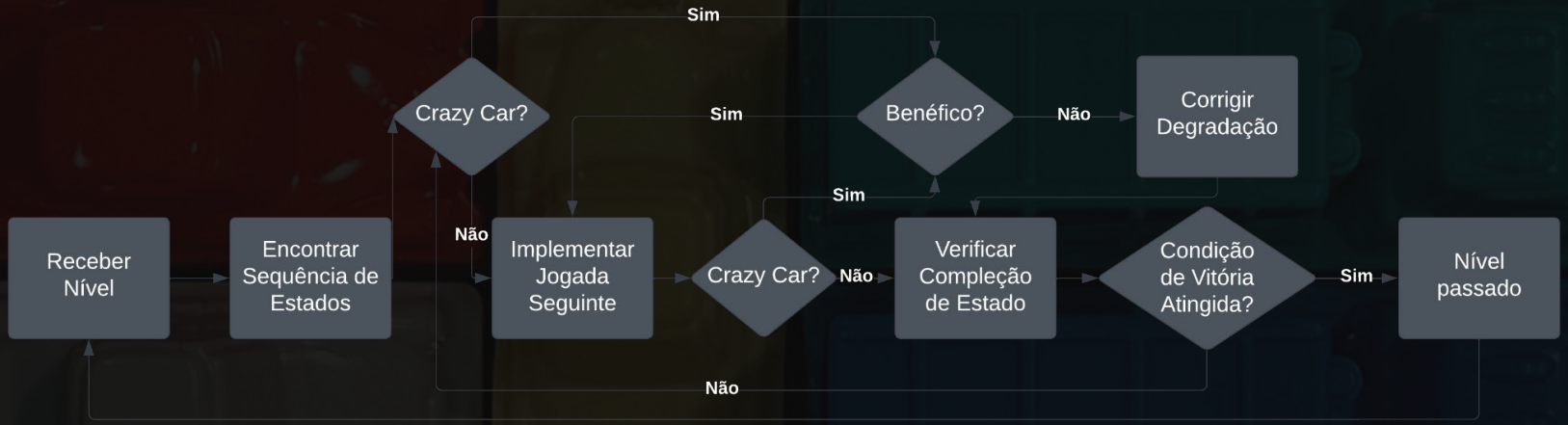
RUSH HOUR

Replaneamento

- O agente inteligente tem capacidade de **replaneamento**, na medida em que é capaz de discernir se o estado do sistema se degradou devido à ocorrência de um *crazy car* e alterar o fluxo de execução: ignorar se for benéfico [(resulta num estado desejável) ou que não afeta o movimento que se segue (*lookahead* de 1)] e corrigir se for prejudicial.
- Concretamente, isto resulta na execução duma **pesquisa parcial**, reutilizando as porções estanques do método *search* anteriormente referido, substituindo, por outro lado, as funções modulares:
    - *correct_heuristic:* passa a aferir a **distância de *Manhattan*** do carro a mover à posição desejada
    - *correct_check_goal:* verifica agora se o sistema voltou ao estado pré-degradado

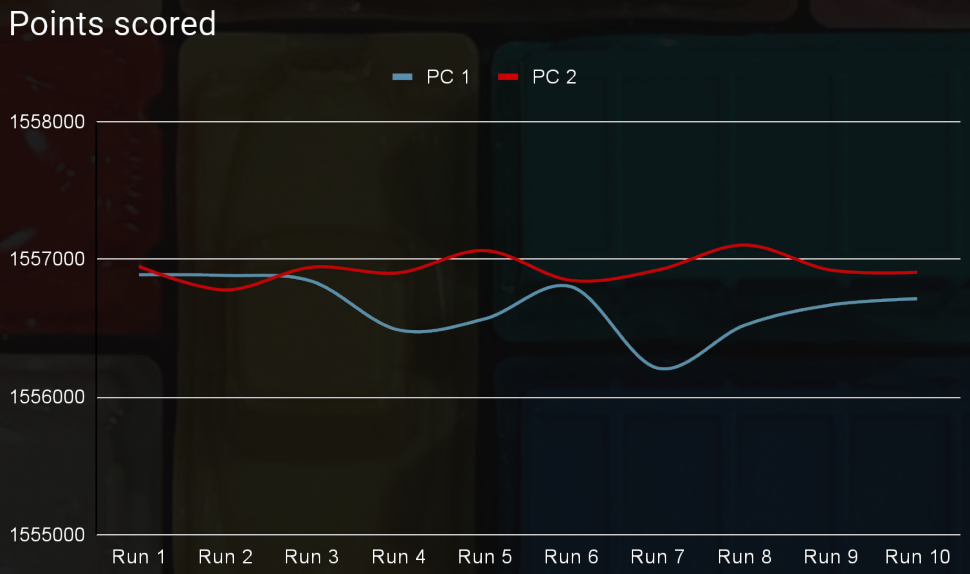
RUSH HOUR

Fluxo de Execução



RUSH HOUR

Benchmarks



Média PC1: 1556657
Desvio Padrão PC1: 217

Média PC2: 1556933
Desvio Padrão PC2: 94