Universidade do Minho
Departamento de Informática

Unidade Curricular de Inteligência Artificial

Algoritmos de procura
VectorRace

1ª Fase

Elaborado por:
António Afonso - A95180
Dinis Estrada- A97503
Gonçalo Lemos - A96484
Simão Matos - A96115

Grupo 11

Ano Letivo 2022/23
25 de dezembro de 2022

Índice

1 Descrição do problema

Este trabalho prático tem como objetivo a implementação de algortimos de procura para a resolução do jogo *VectorRace*, que consiste na simulação de carros de forma simplificada.

Como se trata de um jogo em espaço bidimensional, a movimentação do carro será bastante simples, dependendo apenas do conjunto de acelerações escolhidas. Este conjunto é composto por uma aceleração horizontal (a_l) e uma aceleração vertival (a_c), que poderão tomar os valores -1, 0 e 1.

Portanto velocidade instantânea ($v^j = (v_l, v_c)$, onde j representa uma jogada) será determinada a partir da equação:

$$v_l{}^{j+1} = v_l^j + a_l$$

$$v_c{}^{j+1} = v_c^j + a_c$$

Considerando p o tuplo que, numa determinada jogada, determina a posição atual do carro ($j(p^j = (p_l, p_c))$), podemos calcular a posição do carro da jogada seguinte:

$$p_l{}^{j+1} = p_l^j + v_l^j + a_l$$
$$p_c{}^{j+1} = p_c^j + v_c^j + a_c$$

Os circuitos serão representados como ficheiros de texto, onde o caracter '-' representa a pista, o caracter '#' representa uma posição fora da pista ou um obstáculo, o caracter 'P' representa a posição inicial e o caracter 'F' representa as posições finais.

2 Formulação do problema

Estamos perante um problema de pesquisa, pois pretendemos procurar o conjunto de ações que, a partir do nosso estado inicial, nos irá permitir alcançar o estado objetivo com o menor custo possível. Especificando para o presente problema, pretendemos encontrar o caminho a percorrer pelo carro para chegar, do ponto P, a um dos pontos F que implique o menor custo.

- **Estado inicial:** Qualquer posição representada no mapa, com v = (0,0).
- **Estado objetivo:** Umas das posições finais (F).
- **Operações:** Como carro dependerá apenas da sua aceleração para alterar o seu estado e o vetor aceleração pode tomar 9 valores distintos, existiram 9 estados para os quais o carro pode expandir.
- **Custo da solução:** Cada movimentação terá o custo de uma unidade, exceto quando esta implica sair dos limites da pista, neste caso terá o custo de 25 unidades.

3 Descrição de tarefas e decisões

3.1 Parse do Circuito

Optamos por guardar o circuito numa matriz com números inteiros, onde cada elemento irá representar uma linha. Para a distinção dos seus diferentes componentes utilizamos a seguinte numeração:

1. Pista
2. Metas
3. Ponto inicial
4. Obstáculos

3.2 Criar um grafo

A estratégia para criação de um grafo a partir do circuito, do qual fazemos parsing de um ficheiro .txt, utilizamos uma função de expansão, aplicada ao nodo pai, e esta função aplica todos os movimentos possíveis para o nodo, ou seja, aplica todas as acelarações possiveis e calcula a nova posição e velocidade do nodo. Se este novo nodo não estiver na lista de nodos e estiver dentro dos limites do circuito passará para outras verificações, nomeadamente, se o nodo filho for uma parede irá ser criada uma aresta do nodo pai para si próprio e com custo 25. Se o nodo for um "–"irá verificar se não atravessou nenhuma parede, através da nossa função de verificar caminho, e se não tiver atravessado nenhuma parede a jogada será válida e será criada uma aresta entre nodo pai e nodo filho com custo 1, caso tenha atravessado uma parede não será criada nenhuma aresta. Por fim, se o nodo for uma meta "F"será criada uma aresta entre nodo pai e nodo filho com custo 1.

3.3 Função de Procura

A estratégia de procura implementada nesta primeira fase do projeto foi a BFS(*Breadth-first search*), esta que é uma função de procura não informada. Esta opção tem como fundamento ser uma função de procura completa, ideal para problemas de pequena dimensão como o que nos deparamos.

3.4 Verificação do Caminho

A nossa estratégia para a verificação de caminho para esta fase foi simplesmente fazer uma verificação de posições em que uma parede poderia impedir o movimento do nodo consoante os diferentes tipos de movimentos que o carro pode fazer,ou seja , por exemplo , se a velocidade em x for maior que a velocidade em y este será um vetor diagonal e o que a nossa função faz é verificar as posições do percurso em que uma parede poderia impedir este movimento

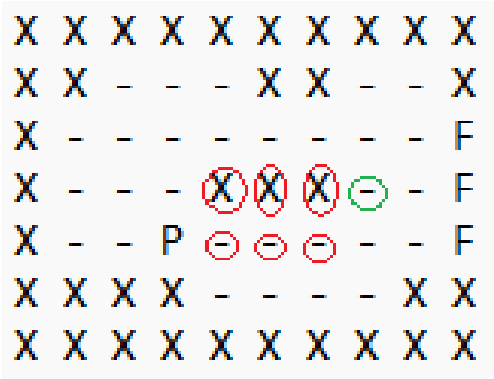

Figura 1: Exemplo de movimento e verificação do mesmo

e se a nossa função detetar uma parede nessas posições irá impedir que este movimento seja feito, o mesmo será feito para outros tipos de vetores.

Na figura o traço rodeado a verde representa o movimento que o carro faria e os carateres rodeados a vermelho as posições onde a nossa função verifica a existencia de paredes que impediriam este movimento de acontecer.

4 Sumário e Descrição de resultados

Com a realização da primeira fase deste trabalho prático, o grupo não só foi capaz de consolidar o conhecimento adquirido ao longo das aulas teóricas e práticas, como também aprofundar o conhecimento sobre algoritmos de procura.

Um dos resultados menos positivos nesta fase, e que se pretende melhorar na próxima, está relacionado com o aumento de velocidade na medida em que se torna cada vez mais dificil impedir que o carro passe paredes, e portanto achamos que nesse aspeto se o circuito tiver dimensões maiores que permitam um grande aumento de velocidade do nodo pode ser que a nossa função responsável por verificar caminho não funcione sempre como desejado.

Consideramos, no entanto, que concluímos esta fase com sucesso, na medida em que implementamos todas as funcionalidades obrigatórias, tendo sido feitos testes para várias posições iniciais do carro e tendo-se obtido em todos os casos, a solução desejada.