Inteligência Artificial

1º Projeto

Grupo 12

Guilherme Saraiva - 93717

Sara Ferreira - 93756

Introdução:

Neste relatório pretendemos analisar a solução proposta para o primeiro projeto da unidade curricular de Inteligência Artificial do primeiro semestre do ano letivo de 2020/2021. O problema apresentado tem por objetivo resolver o jogo de tabuleiro Ricochet Robots.

Avaliação Experimental dos Resultados:

Abaixo, encontram-se o tempo de execução, o número de nós expandidos e o número de nós gerados dados os 5 seguintes testes.

T1

```
4
Y 4 2
G 4 4
B 1 4
R 4 1
R 3 2
2
1 1 r
4 2 r
```

T2

```
5
R 2 1
G 1 5
B 3 2
Y 4 2
Y 5 4
4
1 3 d
3 1 r
3 2 r
4 4 r
```

T3

```
4
R 4 3
G 1 1
B 1 3
Y 2 4
G 3 2
3
3 1 d
2 3 r
3 3 r
```

T4

```
5
R 1 1
G 2 1
B 1 2
Y 5 1
R 5 2
6
2 2 d
2 3 d
4 5 d
3 4 r
4 4 r
5 4 r
```

T5

```
7
R 3 5
G 1 3
B 4 4
Y 4 7
B 6 1
6
2 2 d
2 7 d
3 1 r
4 1 r
4 3 r
7 1 r
```

	Procura em Profundidade Primeiro			**Procura Gananciosa**			**Procura A***			**Procura em Largura Primeiro**		
	Tempo de Execução	**Número de Nós Expandidos**	**Número de Nós Gerados**	**Tempo de Execução**	**Número de Nós Expandidos**	**Número de Nós Gerados**	**Tempo de Execução**	**Número de Nós Expandidos**	**Número de Nós Gerados**	**Tempo de Execução**	**Número de Nós Expandidos**	**Número de Nós Gerados**
T1	N.D.	N.D.	N.D.	N.D.	N.D.	N.D.	0.026	25	169	0.962	1450	9615
T2	N.D.	N.D.	N.D.	0.003	3	28	0.004	3	28	0.091	79	710
T3	N.D.	N.D.	N.D.	0.006	5	36	0.020	18	156	0.107	139	1072
T4	N.D.	N.D.	N.D.	0.015	11	72	0.161	73	520	1.5698	1494	11149
T5	N.D.	N.D.	N.D.	N.D.	N.D.	N.D.	1.387	179	1788	25.592	17326	172741

Na tabela acima estão presentes os resultados dos testes executados por diferentes algoritmos de procura. Nota: as parcelas que não possuem valores resultam do facto de que não foi encontrada solução para alguns testes executados por alguns algoritmos.

Por observação da tabela, é de notar que, no algoritmo de Procura em Largura Primeiro (BFS), conseguimos obter bons resultados apesar de não serem ideais. Na Procura em Largura Primeiro, começamos a partir da raiz e visitamos os nós nível por nível. É garantido que a solução ótima seja encontrada antes de qualquer solução subótima, porque o primeiro estado objetivo que encontramos é aquele com nível mais baixo em comparação com outros estados objetivo.

Por outro lado, na Procura em Profundidade Primeiro (DFS), percorremos um caminho até chegar a um beco sem saída, depois voltamos para o caminho mais próximo e assim adiante. Assim, não precisamos de manter todos os vértices de um nível quando comparado com o BFS. Dito isto, o DFS pode não conseguir encontrar a solução do problema como se pode observar na tabela.

Passando agora para os algoritmos que utilizam heurística, pode se observar que a Procura Gananciosa, nos casos em que é encontrada a solução, é mais eficiente que a Procura A* sendo ambas mais eficientes que a BFS. Estes dois algoritmos enquadram-se no algoritmo de busca melhor-primeiro que usa uma função de procura do nó ao destino. No caso da Procura Gananciosa a função de custo é dada pela função heurística, ou seja, f(n) = h(n), enquanto que na Procura A*, a função de custo é dada por f(n) = g(n) + h(n), sendo g(n) o número de nós visitados.

Assim, a Procura Gananciosa expande primeiro o nó cuja distância estimada ao objetivo é a menor e, portanto, é não completa podendo, em alguns casos, não chegar ao resultado pretendido como mostrado na tabela.

Por fim, na procura A*, devido à sua função custo, que utiliza uma função de heurística admissível pois esta nunca superestima o custo de atingir o *goal* a partir da *source*, pode se concluir que esta é completa e por isso a mais eficiente comparativamente aos outros algoritmos de procura, uma vez que encontra sempre a solução ideal entre o robot ativo e o seu alvo.

Em relação às heurísticas, no início começamos por utilizar a *Manhattan distance* desde a posição do robot ativo ao alvo onde muitas das vezes não era encontrada a solução ao problema. Após várias discussões, conseguimos implementar uma heurística que melhorou em cerca de 60% a rapidez de procura da solução que foi tratar os robots de igual forma, ou seja, em vez de só fazer a *Manhattan distance* do robot ativo ao alvo, calculamos a *Manhattan distance* de cada robot caso este pertencesse à *action* do *Node*.

Conclusão:

Através desta análise, é possível perceber que o algoritmo de procura que é ótimo comparativamente aos restantes, é o algoritmo de procura A*.