Universidade de Lisboa - Instituto Superior Técnico
Licenciatura em Engenharia Informática e de Computadores
Inteligência Artificial

1º Projeto - Grupo 22

Gonçalo Marques, 84719 Manuel Sousa, 84740

A*

A procura por $A*$ é um algoritmo de procura que oferece optimalidade. No entanto isso apenas acontece se garantirmos que a heuristica utilizada é admissível. Um dos principais problemas da utilização desta procura é que exige guardar todos os nós em memória, tornando a complexidade espacial exponencial.

Comparando o Exemplo 2 e o Exemplo 4, conclui-se que o tamanho do tabuleiro tem um impacto bastante significativo na procura, visto que duplicando o tamanho do tabuleiro, o tempo de procura aumentou para 1000% no Exemplo 4. Através de outra comparação entre o Exemplo 4 e o Exemplo 5, concluímos que o aumento do número de cores tem um impacto no tempo de procura, mas não tão significativo. Quase duplicando o número de cores, obteve-se um aumento do tempo de procura de cerca de 250% no Exemplo 5.

Tabela 1: Desempenho do A*

	Exemplo 1	Exemplo 2	Exemplo 3	Exemplo 4	Exemplo 5
Tempo de Execução (s)	0.000266	0.001522	25.653358	0.015687	0.040954
Nº de nós expandidos	1	4	74702	24	16
Nº de nós gerados	0	7	74701	43	91

Procura em Profundidade Primeiro

A Procura em Profundidade Primeiro é um algoritmo de procura não óptimo. Este algoritmo não encontra soluções em espaços de estados com profundidade infinita ou com ciclos, e por isso concluimos que não é completo.

Comparando o Exemplo 2 e o Exemplo 4, conclui-se que o tamanho do tabuleiro tem um impacto significativo na procura, aumentando o tempo de execução para 2000% no Exemplo 4, comparativamente ao Exemplo 2. Através de outra comparação entre o Exemplo 4 e o Exemplo 5, concluímos que o aumento do número de cores tem um impacto extremamente significativo no desempenho, aumentando o tempo de execução para cerca de 3000000% no Exemplo 5.

Tabela 2: Desempenho da Procura em Profundidade Primeiro

	Exemplo 1	Exemplo 2	Exemplo 3	Exemplo 4	Exemplo 5
Tempo de Execução (s)	0.000291	0.001387	18.795010	0.027842	795.923310
Nº de nós expandidos	1	4	74702	54	3123308
Nº de nós gerados	0	7	74701	85	3123363

Procura Gananciosa

A Procura Gananciosa é um algoritmo batante semelhante à procura em profundidade primeiro, no entanto mais exigente em memória. Não é óptimo, e tambem não é completo, visto que pode entrar em ciclo. A Procura Gananciosa vai procurar sempre o nó que julga estar mais perto do objetivo final.

Relativamente ao tamanho do tabuleiro, o impacto é semelhante ao algoritmo de Procura em Profundidade Primeiro, apesar de ser ligeiramente mais significativo. Em relação ao impacto no aumento do número de cores, conclui-se que este é significativo, aumentando o tempo de execução para cerca de 150% entre o Exemplo 4 e 5.

Tabela 3: Desempenho da Procura Gananciosa

	Exemplo 1	Exemplo 2	Exemplo 3	Exemplo 4	Exemplo 5
Tempo de Execução (s)	0.000331	0.001410	26.159087	0.016135	0.119890
Nº de nós expandidos	1	3	74702	42	256
Nº de nós gerados	0	6	74701	59	319

Conclusão

Sendo b o fator de ramificação, e m a profundidade máxima, podemos descrever as várias propriedades dos algoritmos aqui testados:

Tabela 4: Propriedades das várias procuras

	C. Temporal	C. Espacial	Completo	Ótimo
Profundidade Primeiro	$O(b^m)$	$O(b * m)$	Não	Não
Procura Gananciosa	$O(b^m)$	$O(b^m)$	Não	Não
A*	Exponencial	Exponencial	Sim(Exceto se for infinito)	Sim

A heurística usada neste projeto (nº de grupos restantes) não é admissível, porque em certos casos vai sobrestimar o custo de atingir o objetivo. Por exemplo, se no tabuleiro restarem 3 grupos, e ao apagar um grupo, passar a restar apenas um (dois grupos juntaram-se num só), a heurística vai sobreestimar a distância ao objetivo, visto que na realidade restavam apenas 2 grupos e não 3. Como o problema é NP-Completo, podemos concluir pela conjetura $P \neq NP$, que não existe uma heuristica admissivel para este problema.