Projeto IA – Relatório

Grupo 10 – André Martins Esgalhado (95533) e Filipe Ligeiro Silva (95574)

Obtenção de dados

Os resultados obtidos e apresentados nos gráficos 1 e 4 e tabelas 1 a 4 foram obtidos a partir da execução[1] do código no ficheiro numbrix.py, feito pelos membros do grupo, com os ficheiros fornecidos de teste. O uso da classe *InstrumentedProblem*, definida no ficheiro search.py fornecido, permitiu-nos obter dados extra para esta análise.

A partir daí, os resultados foram agrupados por categorias:

- Tempo
- Ações geradas
- Nós testados para solução (através da função *goal_test*)
- Nós gerados

tendo sido posteriormente colocados em gráficos, um para cada categoria, para poderem ser comparados por teste, por algoritmo de procura. Nos algoritmos de procura, decidimos incluir as estatísticas para RBFS, visto que este algoritmo foi o utilizado na nossa implementação do numbrix.py, por assegurar melhor performance nos testes a que o código foi sujeito na restante avaliação.

Análise

De acordo com os gráficos e tabelas obtidos, podemos constatar que:

1. Para inputs de tamanho pequeno, como os testes 1 e 3, as procuras não informadas (DFS, BFS) se comportam de forma semelhante às procuras informadas (A*, procura gananciosa, RBFS) em quase todos os parâmetros.
2. No seguimento do ponto anterior, podemos observar que BFS é a procura que obtém piores resultados em todas as categorias, chegando a casos em que nem sequer consegue completar a sua execução em tempo útil (nos testes 8, 9 e 10). Gera consistentemente uma maior quantidade de nós, o que leva a um maior número de comparações, e a uma maior quantidade de tempo despendida para execução. Dado o funcionamento deste algoritmo, faz sentido que este seja o pior dos algoritmos comparados.
3. O algoritmo DFS, apesar de gerar e testar mais nós que as pesquisas informadas, comporta-se equivalentemente bem a estas para testes médios/grandes (testes 4, 5, 8, 9 por exemplo). Dada a forma como as ações possíveis são retornadas no código, esta conclusão faz sentido, visto que o algoritmo usado busca retornar uma ação ótima e que possa ser usada sem necessidade de *backtracking*.
4. As procuras informadas, a nível de tempo, variam bastante de teste para teste (sendo a procura gananciosa ou a RBFS as melhores na maioria dos casos), no entanto a procura A* é a que gera mais nós, o que leva a conclusões

[1] Especificações do computador utilizado: laptop com processador Intel i7-9750H e 16Gb de RAM.

semelhantes às do ponto anterior (é das procuras informadas mais lentas, à exceção de um teste, o 6).
A procura RBFS e a procura gananciosa, tirando algumas exceções, comportam-se de forma semelhante quanto ao número de nós gerados e testados, e ações geradas.

5. No geral, as procuras informadas comportam-se melhor que as procuras não informadas, devido ao uso de heurística. Esta verifica condições nas quais uma *Board* não é válida, e retorna um valor alto, para que esta tenha menor probabilidade de ser escolhida, pondo outras ações melhores na linha da frente para serem executadas.
   Nos testes pequenos pode-se observar um comportamento semelhante (como dito no primeiro ponto, nos testes 1 e 3 verifica-se isto), e isto pode-se dever ao facto de que para uma *Board* pequena, o número de ações dadas será reduzido, e de acordo com a nossa implementação, muitas vezes retorna apenas uma ação, a correta a tomar, por forma a evitar *backtracking* (como referido no ponto 4). Assim, o papel da heurística torna-se irrelevante nestes casos, o que leva a um comportamento semelhante entre procuras.
6. Todas as procuras são completas, dados recursos suficientes (RAM, tempo, por exemplo) para que estas terminem, no entanto não existem dúvidas de que, com uma heurística adequada, as procuras informadas estejam noutro patamar de eficiência.
7. Podemos afirmar que a heurística definida por nós não é admissível: apesar do valor desta ser 0 para um estado que seja o final, pode retornar estimativas menores que o custo de chegar à solução. Assim sendo, esta também não é consistente.
8. As ações para um dado estado, na nossa implementação, consistem na posição adjacente a uma posição já colocada, caso apenas exista uma adjacente livre, ou na interseção das posições livres de dois valores cuja subtração seja 2 (caso seja uma opção viável, ou seja, estejam a 2 unidades de distância), ou, se não existirem as opções acima, no conjunto das adjacentes das posições já colocadas, dando mais opções para a heurística filtrar.

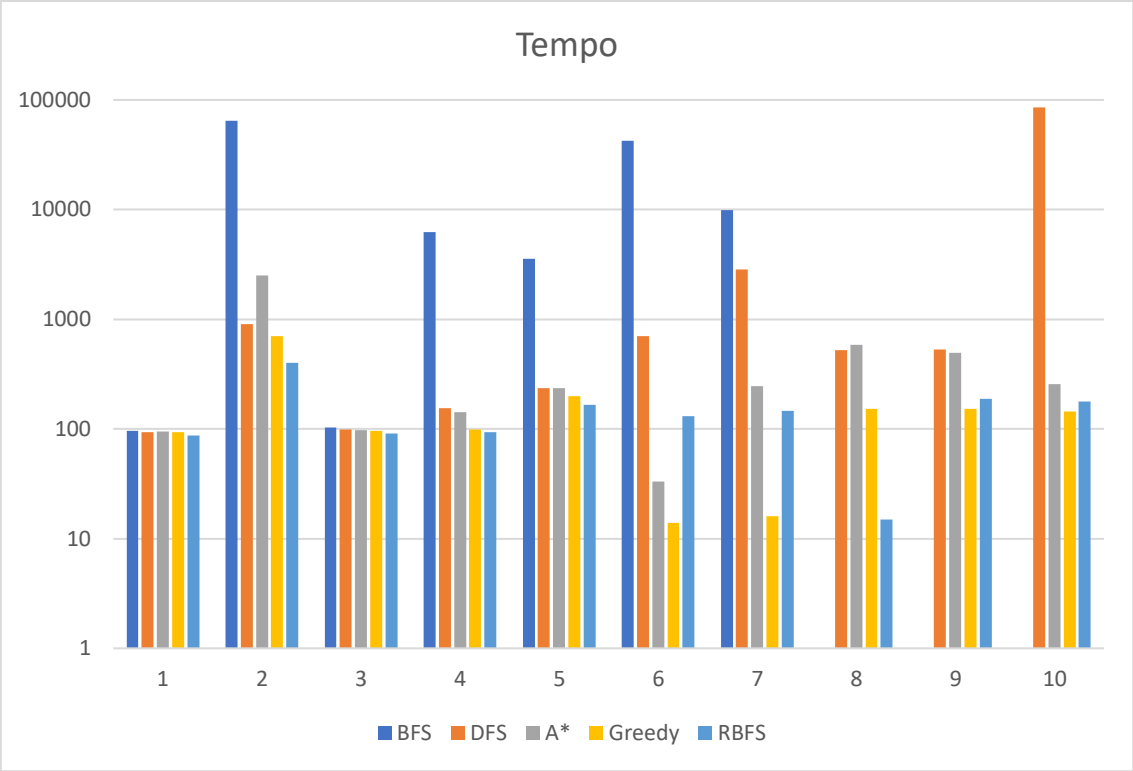

Gráfico 1 - Tempo por teste, por algoritmo de procura

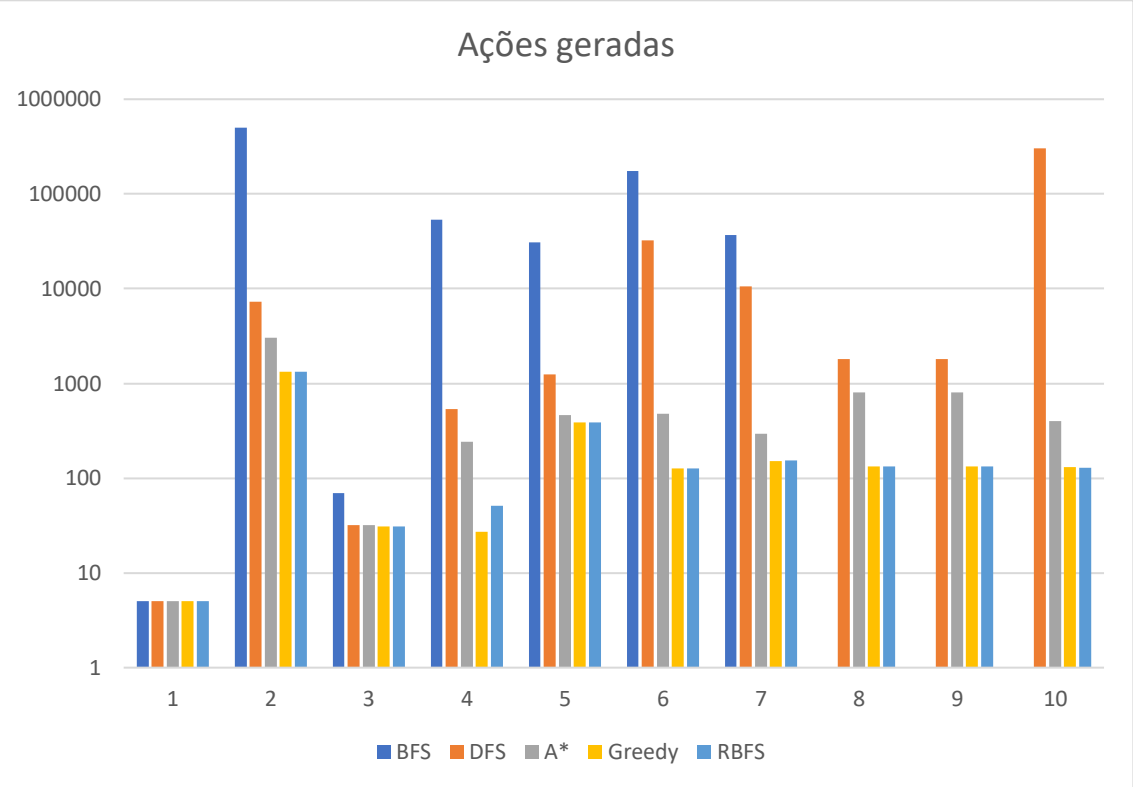

Gráfico 2- Número de ações geradas por teste, por algoritmo de procura

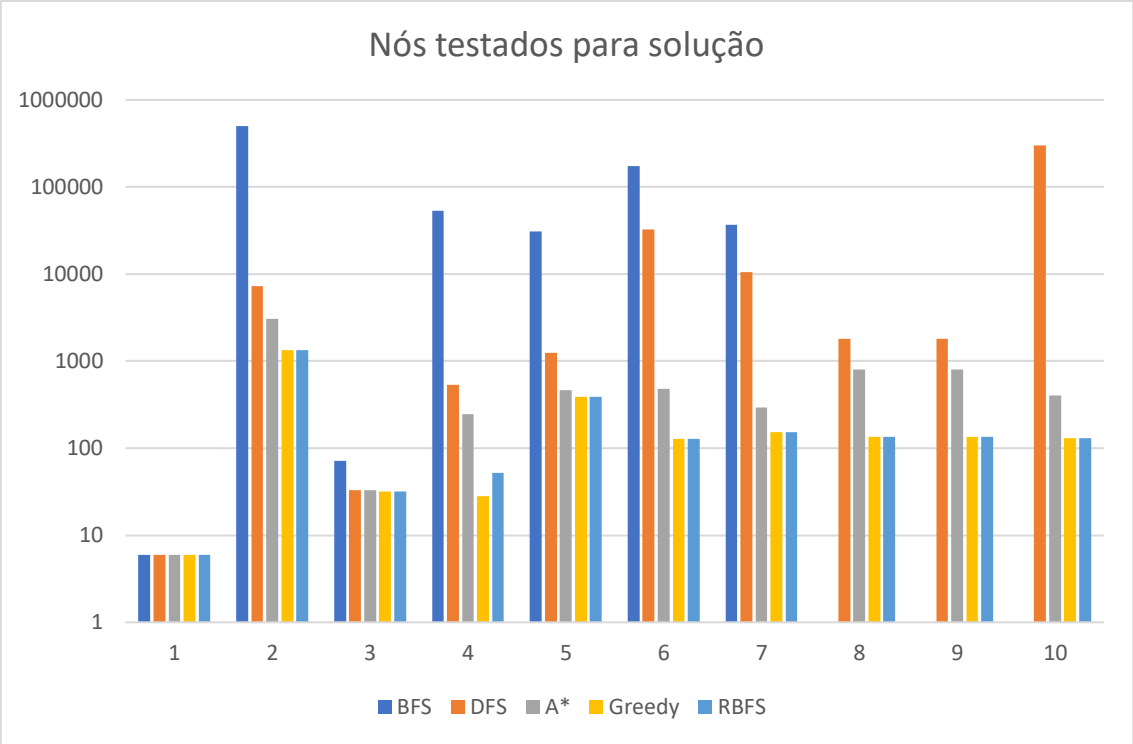

Gráfico 3-Nós testados para solução por teste, por algoritmo de procura

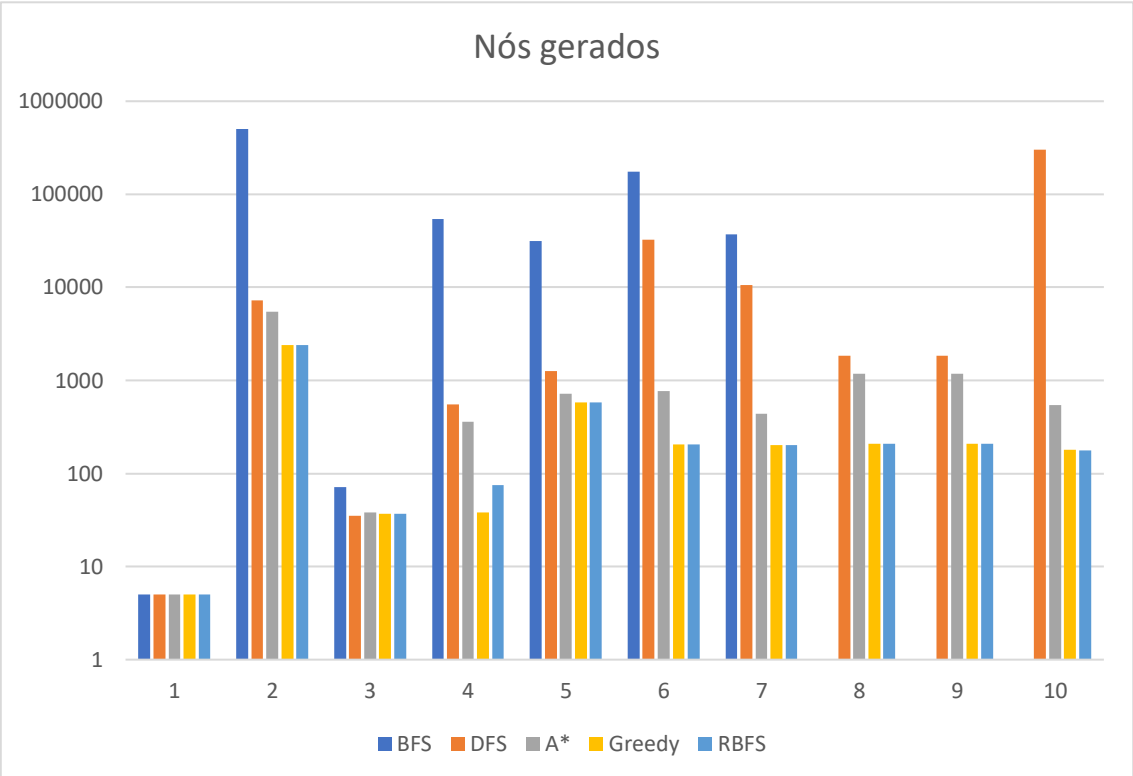

Gráfico 4-Nós testados para solução por teste, por algoritmo de procura

Tempo	BFS	DFS	A*	Greedy	RBFS
1	96	94	95	93	87
2	64439	907	2497	704	399
3	103	99	98	96	91
4	6233	155	143	99	94
5	3575	235	234	198	165
6	42537	702	33	14	131
7	9844	2827	245	16	147
8	0	524	585	152	15
9	0	531	493	152	188
10	0	85663	257	144	178
Ações geradas	BFS	DFS	A*	Greedy	RBFS
1	5	5	5	5	5
2	501804	7229	3034	1325	1325
3	70	32	32	31	31
4	53754	535	244	27	51
5	30990	1252	466	387	387
6	174522	32348	480	126	126
7	36996	10604	294	152	153
8	0	1818	800	134	134
9	0	1818	800	134	134
10	0	301436	401	130	129
Nós testados	BFS	DFS	A*	Greedy	RBFS
1	6	6	6	6	6
2	501805	7230	3035	1326	1326
3	71	33	33	32	32
4	53755	536	245	28	52
5	30991	1253	467	388	388
6	174523	32349	481	127	127
7	36997	10605	295	153	154
8	0	1819	801	135	135
9	0	1819	801	135	135
10	0	301437	402	131	130
Nós gerados	BFS	DFS	A*	Greedy	RBFS
1	5	5	5	5	5
2	502625	7251	5478	2381	2381
3	71	35	38	37	37
4	53915	551	357	38	75
5	31231	1261	715	585	585
6	174601	32366	770	204	204
7	37011	10615	438	201	203
8	0	1838	1174	208	208
9	0	1838	1174	208	208
10	0	301453	544	179	178

Tabelas 1 a 4 – Dados obtidos a partir da execução do programa. Estes foram usados para obter os gráficos acima.