Relatório do EP1 - MAC0425

Renato Lui Geh
NUSP: 8536030

1 Resultados

As variáveis na parte superior da tabela são as mesmas que aparecem na tela de estatísticas do jogador automático do EP no modo teste.

DFS			
Passos	Gerados	Expandidos	Ramificação
36	1701	1619	1.05
36	490	406	1.21
31	57693	57622	1.00
35	144	57	2.53
27	727255	727202	1.00

BFS			
Passos	Gerados	Expandidos	Ramificação
13	1229	1096	1.12
13	1234	1116	1.11
15	1606	1471	1.09
1	6	1	6.00
17	1912	1817	1.05

BestFS			
Passos	Gerados	Expandidos	Ramificação
13	1242	1058	1.17
13	1246	1123	1.11
15	1615	1418	1.14
1	6	1	6.00
17	1906	1691	1.13

A-star			
Passos	Gerados	Expandidos	Ramificação
13	691	542	1.27
13	798	611	1.31
15	867	704	1.23
1	6	1	6.00
17	1116	946	1.18

O algoritmo A-star foi rodado só com a heurística `manhattanDistanceAdmissible`, que é uma distância de Manhattan admissível para o problema. No caso descontamos a posição `y` da `manhattanDistance`, já que o usuário não escolhe ir uma distância para baixo de cada vez.

2 Comparação

Agora que temos todos os dados vamos compará-los. Para calcular a tabela abaixo, pegou-se cada instância i de um algoritmo da tabela acima e, para cada outra instância j de qualquer outro algoritmo, fizemos $r_{ij} = \frac{\sum x_i}{\sum x_j} = \sum \frac{x_i}{x_j}$, onde x_i é a instância i de uma variável x. Por exemplo, para compararmos o número de passos da DFS e BFS, fazemos $r_{dfs,bfs} = \sum \frac{passos_{dfs}}{passos_{bfs}} = \frac{165}{59} = 2.79$. Para não termos de escrever várias tabelas, comparamos um algoritmo com todos e passamos para o próximo. Deste modo temos todas as comparações de um jeito mais compacto. É fácil de ver que podemos calcular os valores restantes a partir dessa e das tabelas que vimos na seção passada.

A partir dessa tabela, podemos ver que a DFS faz muitos passos, pode acabar gerando e expandindo muitos nós e tem uma média de ramificação muito similar a cada iteração. Mas também dá para se ver que há casos que ela tem uma performance muito melhor que os outros algoritmos, chegando a ter menos nós gerados e expandidos que o A-star. No entanto, podemos ver que, na média, a DFS é bem pior que o resto dos algoritmos. Além disso, a DFS é sub-ótima, o que explica o número alto de passos para a solução final quando comparada aos outros.

A BFS, BestFS e A-star tem mesmo número de passos para a solução final. Isso é por que os três algoritmos são ótimos (se o custo for uniforme). Eles também tem uma média similar de ramificação, já que a BFS é, por definição, um algoritmo de largura e BestFS e A-star podem voltar atrás e partir de um caminho diferente da qual estava seguindo. Apesar de terem resultados um tanto parecidos, a BFS e BestFS geram e expandem mais em média do que o A-star. De fato, se olharmos para a tabela desta seção, pode-se ver que a A-star gera e expande mais ou menos metade dos nós que a BestFS e BFS, enquanto que estas duas tem média muito parecida. Isso ocorre por que em um espaço de estados onde o custo é uniforme, a BestFS comporta-se de forma parecida a uma BFS.

Variável	i\j	DFS	BFS	BestFS	A-star
Passos	DFS	1	2.79	2.79	2.79
Gerados	BFS	0.007	1	0.99	1.72
Expandidos	BestFS	0.006	0.96	1	1.88
Ramificação	A-Star	1.61	1.05	1.04	1.05

3 Melhorias

A princípio, quando foram implementadas, todos os algoritmos geravam e expandiam todos os nós que precisavam sem verificar se aquele nó já tinha sido visto em algum caminho anterior. Ao testar a BFS, BestFS e A-star, o tempo para calcular cada solução era muito grande, travando o browser. Para melhorar a performance, foi incluído um `Set` para tentar guardar os nós vistos e não entrarmos em um caminho já percorrido. Incluímos esse `Set` na BFS, BestFS e A-star para podermos rodar os testes normalmente. Os testes acima foram feitos já com o `Set` incluído.

A DFS, no entanto, ainda não guardava os nós já visitados. A tabela abaixo mostra a DFS com a memorização de nós implementada. Dá para ver que a quantidade de nós gerados e expandidos é muito menor, chegando a ser melhor que o A-star em termos de memória alocada. Contudo, ele ainda é sub-ótimo.

DFS-stored			
Passos	Gerados	Expandidos	Ramificação
36	363	293	1.24
36	234	162	1.44
31	698	637	1.10
35	129	56	2.30
27	935	889	1.05

Portanto, um jeito de se melhorar os algoritmos é memorizar os nós já vistos para não termos de percorrer um caminho já percorrido.

Uma outra possível melhoria é criar heurísticas melhores. A heurística usada pela BestFS é apenas o custo acumulado de cada nó. O A-star nas tabelas acima usa uma heurística que combina o custo acumulado de cada nó e a `manhattanDistanceAdmissible`, que é apenas o módulo da diferença das posições no eixo horizontal entre o tetraminó do nó atual com o tetraminó meta.

Para melhorarmos nossa heurística do A-star, podemos, além da distância de Manhattan admissível, incluir o quão distante da rotação final está o tetraminó atual. Portanto, a heurística resultante é:

```
var betterHeuristics = function(s1, s2) {
  return Math.abs(s1.tetromino.xpos - s2.tetromino.xpos) +
    Math.abs(s1.tetromino.type.next - s2.tetromino.type.next);
}
```

A-star-hr			
Passos	Gerados	Expandidos	Ramificação
13	514	375	1.37
13	596	421	1.42
15	648	488	1.33
1	6	1	6.00
17	891	709	1.26

Pela tabela `A-star-hr`, pode-se ver que o número de nós gerados e expandidos é menor que o do A-star original. Além disso, `betterHeuristics` é admissível, já que o custo real de se fazer a rotação é no máximo de custo `4*Problem.StepCost()`, enquanto que na heurística a estimativa é de no máximo 4 com relação a rotação.