Passeio do Cavalo - Heurística de Warnsdoff

Samuel Brum Martins

Introdução

O problema do passeio do cavalo é um quebra cabeças milenar, útil como exemplo e simplificação de problemas em teoria de grafos.

O problema original consiste em percorrer, com o cavalo, todo o tabuleiro de xadrez, passando apenas uma vez por cada casa. Podem existir mais de uma solução para uma casa inicial específica e, em alguns tabuleiros menores, podem haver casas sem uma única solução.

O algorítmo de *backtracking* consiste em uma abordagem de força bruta para solucionar o problema, essecialmente realizando "caminhadas cegas"pelo tabuleiro e retrocedendo um passo quando não há mais movimentos. Tal abordagem é extremamente ineficiente, já que o problema cresce exponencialmente com o tamanho do tabuleiro, pois cada passo possui multiplos destrichamentos, cada qual com suas multiplas n-furcações.

Uma forma de contornar tal dificuldade é através de heurísticas, regras gerais que abrem mão de algum aspecto de uma solução completa, como precisão, optimização, ou completeza por velocidade.

Uma Heurística notável para o problema do passeio do cavalo é a de Warnsdoff, que consiste em escolher o próximo movimento baseado no número de movimentos possíveis da casa.

Implementação

A específicação do projeto requeria que uma biblioteca `passeio.h` possuisse uma função `void passeio(int x, int y)` que, ao ser chamada, realizasse o algorítmo de *backtracking* para o passeio do cavalo em um tabuleiro 8x8, começando da casa de linha x e coluna y.

A função passeio como foi implementada serve como casca para a chamada da função, onde são ajustados todos os parâmetros necessários, como um tabuleiro vazio e uma tabela de consulta para implementar a heurística de Warnsdoff. Em seguida, é chamada a função `passo`, que realiza o grosso da implementação. Nela é realizado iterativamente o algorítmo de *backtracking* a partir da posição inicial da chamada de `passeio`. Em pseudocódigo a ideia é apresentada no Algorítmo 1.

Algorithm 1 Algorítmo de backtracking para passeio do cavalo

```
1: function BACKTRACKING(casa, turno)
2:     for movimento in movimentovalidos do
3:         tabuleiro[casa] ← turno
4:         if Backtracking(casa + movimento, turno + 1) then
5:             return True
6:         else
7:             RetocedeCasa()
8:         end if
9:     end for
10: end function
```

Mesmo assim, essa implementação ingênua apresenta um problema de performânce, uma vez que a implementação possui $\mathcal{O}(8^{n^2})$, já que para cada casa há até 8 movimentos que o cavalo pode fazer e cada permutação de visitas configura um passeio diferente.

Sendo assim, implementou-se também a heurística de Warnsdoff, que consiste em ordenar os movimentos possíveis dos quadrados que possuem o menor número de movimentos válidos até o maior número de movimentos válidos. O ordenamento deu-se por `BubbleSort`, já que o número de elementos a serem ordenados era sempre 8, não justificando a implementação de um sorteamento mais sofisticado.

O conceito é intuitivo, já que as casas que possuem o menor número de movimentos válidos também são as que tem menos pontos de acesso, e portanto são mais prováveis de apresentarem gargalos nas tentativas e erro. Priorizando o acesso a essas casas, reduzimos a chance de termos que retroceder um passo.

Conclusões

Na implementação ingênua, o programa é até capaz de rodar algumas posições iniciais em tempo hábil, com o melhor caso tendo sido em torno de 30 segundos e com casos nos quais o programa rodava por mais de 10 minutos sem sucesso.

Após implementação da heurística o caso médio caiu para frações de segundo, com os piores casos na ordem de alguns poucos segundos, claramente uma melhoria de várias ordens de magnitude, demonstrando que, em termos práticos, uma solução incompleta pode ser boa o suficiente, a depender da aplicação pretendida.

Os resultados com o formato de saída exigidos e o tempo de cada uma das soluções encontram-se no arquivo **saida.txt**, seguidas do tempo de execução de cada uma delas, em ordem, ambos gerados pelo teste **main.c**. Ambos arquivos encontram-se na pasta "Apêndices".