Universidade Federal de São João del-Rei
Grafos
Trabalho Prático

Felipe Francisco Rios de Melo
Thales Mrad Leijoto

1. INTRODUÇÃO

O problema das n-rainhas é antigo tendo mais de dois séculos. Inicialmente era conhecido como o problema das 8-rainhas, tem sido estudado por matemáticos famosos, ao longo dos anos, incluindo o matemático alemão Karl Friedrich Gauss (1777-1855). O problema foi generalizado para n × n lugares (para qualquer $n \geq 4$) em 1850 por Franz Nauck. Desde a década de 1960, com a rápida evolução na ciência da computação, este problema tem sido usado como exemplo para algoritmos de busca por retrocesso backtracking, geração de permutação, paradigma dividir e conquistar, metodologia de desenvolvimento de programas, problemas de satisfação com restrições, programação inteira, e de especificação.

O problema consiste em encontrar todas as formas possíveis de posicionar as n-rainhas em um tabuleiro de xadrez de n lugares, de tal modo que as n-rainhas não se ataquem. De acordo com as regras de xadrez, a rainha ataca células em todas as direções para linha, coluna e diagonais (Figura 1). Assim, o objetivo é posicionar as n-rainhas em um tabuleiro de n × n de tal forma que duas rainhas nunca estejam na mesma linha, coluna e diagonais.

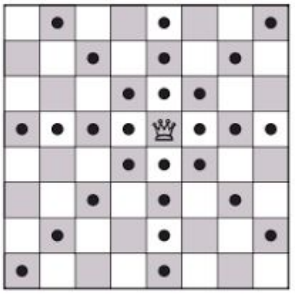

Figura 1 - Movimentos possíveis da Rainha do Xadrez.

O problema das 8 rainhas possui 92 soluções distintas, que podem ser obtidas a partir de um conjunto de 12 soluções únicas, aplicando operações de simetria (rotação e reflexão). A Figura 2 mostra uma solução para o problema clássico.

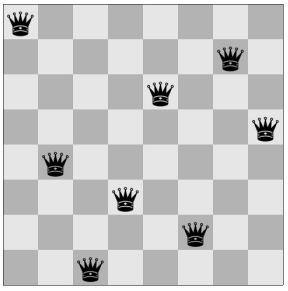

Figura 2 - Exemplo de solução para o problema das 8-rainhas.

Este trabalho prático, teve por objetivo, implementar um algoritmo em grafos que encontre todas as 92 soluções para o problema das 8-rainhas. Em nossa implementação, estendemos o problema para n-rainhas, inserindo, no momento da execução do programa um argumento referente ao valor de n.

2. IMPLEMENTAÇÃO

2.1. Modelagem do grafo

Seja um grafo T cujos vértices representam as posições de um tabuleiro (Figura 2). Para cada posição do tabuleiro, é interligado por uma aresta todas as posições que possam ser atingidas pela rainha a partir dela.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Figura 3 - posicionamento dos vértices em um tabuleiro 8x8.

O mapeamento de todas as arestas do grafo T foi realizado de acordo com a seguinte lógica:

Para todo vértice i em n*n
    Para todo vértice j em n*n
        Se i for diferente de j, cheque as seguintes condições:

1. (j-i) % n == 0
2. (j-i) % (n+1) == 0 and abs(i//n - j//n)*(n+1)+i == j
3. (j-i) % (n-1) == 0 and abs(i//n - j//n)*(n-1)+i == j
4. i//n == j//n

O item (1) condiz ao movimento vertical da rainha.
O item (2) condiz ao movimento diagonal da esquerda para a direita da rainha.
O item (3) condiz ao movimento diagonal da direita para a esquerda da rainha.
O item (4) condiz ao movimento horizontal da rainha.

Se pelo menos uma destas 4 condições forem verdadeiras, é adicionado uma aresta entre i e j.

Ao plotar o grafo T gerado com a função *plot()* da biblioteca *igraph* temos o seguinte resultado:

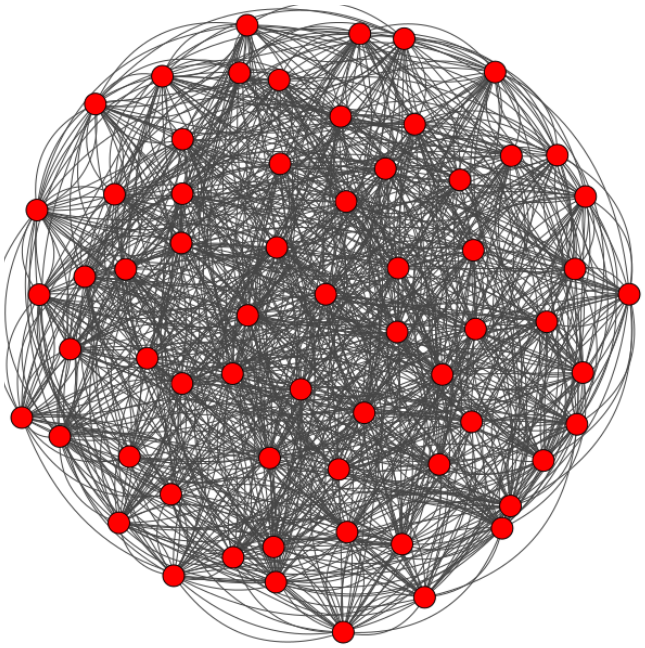

Figura 4 - Grafo de todos os movimentos possíveis por rainhas, em todas as posições.

2.2. Algoritmo para identificação de todas as soluções

O conjunto independente máximo representa a solução para o problema das n-rainhas.

Na teoria dos grafos, um conjunto independente de um grafo G é um conjunto S de vértices de G tal que não existem dois vértices adjacentes contidos em S. Em outras palavras, se a e b são vértices quaisquer de um conjunto independente, não há aresta entre a e b.

Aplicando ao nosso modelo que simula todos os movimentos das rainhas pelo tabuleiro, encontrar o conjunto máximo de vértices não adjacentes entre si, equivale a encontrar as posições das *n* rainhas de maneira que elas não ataquem umas às outras.

Dado um grafo G, seu conjunto independente é uma clique em seu grafo complementar.

Diante deste fato, foi gerado o grafo complementar de T e executado o algoritmo de Bron-Kerbosch**,** que a princípio encontra todas as cliques maximais, mas pelo fato de estarmos inserindo o grafo complementar, teremos como retorno a enumeração de todas os conjuntos independentes maximais.

2.2.1. Bron-Kerbosch

Proposto por Coenraad Bron e Joep Kerbosch em 1973, a base desse algoritmo é a busca exaustiva, mas utiliza-se um conhecimento maior sobre a natureza do problema da clique máxima. A grande vantagem desse algoritmo é que ele gera apenas cliques maximais, evitando assim que cada conjunto gerado tenha que ser comparado com os préviamente testados. O algoritmo de Bron-Kerbosch é caracterizado pelo backtracking, que procura por todas as cliques maximais em um dado grafo G. Existem três estruturas que são a base do algoritmo:

R: conjunto de vértices já definidos como parte da clique;

P: vértices que têm ligação com todos os vértices de R; esse é o conjunto de candidatos a entrar na clique

X: vértices que já foram analisados e não levam a uma extensão do conjunto de candidatos P, cujo objetivo é evitar comparação excessiva.

```
BronKerbosch2(R,P,X):
    if P and X are both empty:
        report R as a maximal clique
    choose a pivot vertex u in P ⋃ X
    for each vertex v in P \ N(u):
        BronKerbosch2(R ⋃ {v}, P ⋂ N(v), X ⋂ N(v))
        P := P \ {v}
        X := X ⋃ {v}
```

Algoritmo 1 - Bron-Kerbosch com pivoteamento

A execução do algoritmo é feita inicialmente com R e X vazios e P contendo todos os vértices do grafo. Dentro de cada chamada recursiva, se o conjunto P e X está vazio, o algoritmo encontra uma clique maximal ou realiza backtracking. Escolhe-se um “vértice pivô” *u* escolhidos de P U X, qualquer clique maximal deve incluir tanto *u* ou um dos vértices não adjacentes a ele, pois caso contrário a clique poderia ser aumentada. Para cada vértice v escolhido de P, faz-se uma chamada recursiva, adicionando em R. Na recursão, os conjuntos P e X são restritos aos vizinhos de v, possibilitando encontrar todas as extensões da clique que contém v. Continuando a execução, move-se v de P para X, uma vez que v já foi analisado. Várias recursões são chamadas, até que se encontrem todas as cliques maximais do grafo.

Vale ressaltar que no pior caso, o algoritmo de Bron-Kerbosch apresenta complexidade exponencial $O(3^{n/3})$.

3. RESULTADOS E DISCUSSÕES

Todas as 92 soluções foram encontradas com n = 8, como o esperado, o tempo gasto com esta entrada flutua em torno de 0,5s.

Para n = 11, o algoritmo levou um pouco mais que 3 minutos para calcular todas as 2680 soluções. Com n > 11 este tempo tende a aumentar exponencialmente, visto que a complexidade do Bron-Kerbosch é da ordem de $3^{n/3}$.

Porém o fato desta implementação, envolvendo grafos e conjuntos independentes, ter uma complexidade exponencial não vem a ser um grande problema neste caso, pois o problema das n-rainhas está na classe dos NP-Completos, ou seja, não se conhece, ainda, um algoritmo que o resolva em tempo polinomial.

Apenas a fim de curiosidade, o maior *n* que um computador conseguiu executar até hoje, foi n = 27, no ano de 2016. Para tal foi utilizado técnicas de computação paralela.

4. CONCLUSÃO

O resultado obtido foi o previsto: o algoritmo se comporta bem para n <= 11, e para n > 11 torna-se inviável a execução do algoritmo, devido a sua complexidade exponencial.

Quanto às dificuldades de implementação, a parte mais complexa se deu na lógica condicional de adição de arestas no grafo. Definir quais condições devem ser satisfeitas para haver uma aresta, como adicionar arestas referentes às movimentações horizontais, como adicionar arestas referentes às movimentações diagonais e etc. Em relação ao algoritmo de Bron-Kerbosch foi tranquilo a implementação, pois o algoritmo já existe, basta codifica-lo na linguagem escolhida.

Com este trabalho foi possível aplicar uma modelagem diferente da convencional para o problema das n-rainhas, por meio da estrutura de dados grafos aliado aos conceitos e relações entre o conjunto independente e a clique de um grafo.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Problema da oito damas. In: Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Problema_das_oito_damas>

Budke, Gerson Fernando, et al. "Aplicando Programação em Lógica com Restrições no Problema das N-Rainhas com Tabela de Pesos." *Anais do Computer on the Beach* (2012): 251-260. Disponível em: <http://docplayer.com.br/60607101-Problema-das-n-rainhas-com-tabela-de-pesos.html>

Fabiano C. Botelho. Trabalho Prático. Disponível em: <https://homepages.dcc.ufmg.br/~nivio/cursos/pa06/tp2/tp22/tp22.pdf>
Santos, Samuel Souza Brito Haroldo Gambini. "Pivotamento no Algoritmo Bron-Kerbosch para a

Detecçao de Cliques com Peso Máximo." Disponível em: <http://www.din.uem.br/sbpo/sbpo2011/pdf/87964.pdf>

The N Queens problem was recognized as an NP-complete problem. Disponível em: <https://sudonull.com/posts/7805-The-N-Queens-problem-was-recognized-as-an-NP-complete-problem>