UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS REITOR JOÃO DAVID FERREIRA LIMA
PROGRAMA DE GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Marcelo Emilio Vendramin

Algoritmos para encontrar anéis cromáticos em redes de parentesco

Florianópolis
2021

Marcelo Emilio Vendramin

Algoritmos para encontrar anéis cromáticos em redes de parentesco

Trabalho de Conclusão de Curso submetida ao Programa de Graduação em Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Ciências da Computação.
Orientador: Prof. Álvaro Junio Pereira Franco, Dr.

Florianópolis
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Vendramin, Marcelo Emilio
Algoritmos para encontrar anéis cromáticos em redes de parentesco / Marcelo Emilio Vendramin ; orientador, Álvaro Junio Pereira Franco, 2021.
56 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2021.

Inclui referências.

1. Ciências da Computação. 2. Anéis cromáticos. 3. Algoritmos em grafos. 4. Redes reais. I. Franco, Álvaro Junio Pereira. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

Marcelo Emilio Vendramin

Algoritmos para encontrar anéis cromáticos em redes de parentesco

O presente trabalho em nível de bacharelado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Alvaro Junio Pereira Franco, Dr.
Instituição Universidade Federal de Santa Catarina

Prof. Márnio Teixeira-Pinto, Dr.
Instituição Universidade Federal de Santa Catarina

Prof. Rafael de Santiago, Dr.
Instituição Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Bacharel em Ciências da Computação.

_____

Coordenação do Programa de Graduação

_____

Prof. Álvaro Junio Pereira Franco, Dr.
Orientador

Florianópolis, 2021.

RESUMO

O trabalho tem como objetivo possibilitar a verificação de hipóteses sobre povos através da disponibilização de anéis cromáticos derivados de redes reais. Para isto, foi definido a utilização de grafos para representação das redes de parentesco reais. Um modo de colorir vértices deste grafo foi proposto com base em uma hipótese antropológica. Também foram propostos algoritmos que calculam propriedades derivadas das cores. Utilizando as propriedades obtidas, o trabalho desenvolveu formas de enumerar anéis cromáticos, implementando cada uma delas para execuções sequenciais e paralelas. Uma análise sobre o tempo de execução das buscas implementadas foi realizada sobre a rede real Arara. Uma análise numérica dos resultados também foi realizada sobre os 520 anéis com uma afinidade e 110590 anéis com duas afinidades encontrados para a rede.

Palavras-chave: Anéis cromáticos. Algoritmos em grafos. Redes reais.

ABSTRACT

The goal of this work is to enable the verification of hypotheses about people by providing chromatic rings obtained from real networks. To achieve this goal, the use of graphs to represent real kinship networks was defined. A way of coloring vertices of this graph was proposed based on an anthropological hypothesis. Algorithms that calculate properties derived from colors have also been proposed. Using this properties, the work developed ways of enumerate all the chromatic rings, implementing each one to execute sequentially and parallelaly. An analysis of the execution time of the implemented searches was carried out on the Arara network, a real network. An analysis of numeric results was also performed on the 520 rings with one affinity and 110590 rings with two affinities found on the network.

Keywords: Chromatic rings 1. Algorithms in graphs. Real networks.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Uma rede de parentesco representa indivíduos de um povo e suas relações internas, como casamentos e filiações (AGHASSIAN; AUGÉ; BESSA, 1975). Uma rede de parentesco é um grafo misto, onde os arcos representam relações de filiação entre indivíduos e as arestas representam os casamentos. Um anel é uma estrutura formada por relações de parentesco e casamentos, arranjados de uma forma a lembrar um ciclo (HAMBERGER *et al.*, 2004).

Existe uma área na Antropologia que se interessa em analisar anéis obtidos em redes de parentesco, porém eles podem existir em grande quantidade, gerando um desafio sobre sua enumeração. Este desafio se encontra em enumerá-los de maneira rápida (enumerar todos os anéis no menor intervalo de tempo), simples (com pouco consumo de espaço) e clara (para cada anel inserir informações que auxiliem na análise).

Algumas propriedades da rede podem ser modeladas usando cores nos vértices, nas arestas e nos arcos. Por exemplo, adicione uma cor distinta para cada ancestral muito antigo, ou seja, um ancestral sem registro de quem seriam seus pais. Quem são os descendentes dos ancestrais antigos? Aqueles que possuem as cores de tais ancestrais desde que coloquemos cores em indivíduos a partir dos ancestrais mais antigos e seguimos para os descendentes usando as relações de filiação. Essa é apenas uma regra simples que demonstra a utilidade das cores nos vértices para questões antropológicas.

As questões que queremos resolver neste trabalho estão relacionadas à enumeração de anéis cromáticos com o objetivo de decifrar aspectos da estrutura social de um povo através dos casamentos entre os indivíduos. Para isso, desenvolvemos e implementamos algoritmos para enumerar anéis cromáticos. Os algoritmos desenvolvidos foram aplicados sobre a rede Arara para um levantamento quantitativo dos anéis cromáticos encontrados e do tempo de execução dos algoritmos. Este trabalho contou com o auxílio de uma bolsa PIBIC do CNPq e um dos algoritmos desenvolvidos foi apresentado no Encontro de Teoria da Computação (ETC) em 2021.

1.1 OBJETIVOS

O trabalho envolverá um levantamento bibliográfico em duas disciplinas: Antropologia e Computação. Dessa forma, uma fundamentação na *Antropologia Computacional* e nos *Algoritmos em Grafos* será realizada. Uma implementação de cada algoritmo estudado/proposto foi realizada. O objetivo principal é enumerar anéis cromáticos de redes de parentesco reais.

1.1.1 Objetivo Geral

O objetivo do trabalho é possibilitar a verificação de hipóteses sobre *povos* através da disponibilização de anéis cromáticos derivados de redes reais.

1.1.2 Objetivos Específicos

- Pesquisar, elaborar e implementar algoritmos que encontrem anéis cromáticos em redes de parentesco.
- Analisar os resultados obtidos.

1.1.3 Organização do Trabalho

O trabalho é dividido em sete capítulos. O primeiro apresenta uma introdução. O segundo descreve uma teoria fundamental para este trabalho e apresenta algumas ferramentas que auxiliam interessados da área. No terceiro são descritos algoritmos fundamentais em grafos, enquanto que no quarto são descritos algoritmos que encontram anéis. No quinto são apresentados algoritmos para enumerar anéis cromáticos e no sexto são descritos os resultados númericos obtidos. Por último, no sétimo capítulo, são apresentados as conclusões e os trabalhos futuros.

1.1.4 Metodologia

O desenvolvimento deste trabalho se iniciou com uma pesquisa sobre os temas associados a área da antropologia que estuda redes de parentesco e a teoria das alianças matrimoniais, assim como uma pesquisa conjunta com implementação sobre algoritmos essenciais para compreensão de grafos, que possibilitaram validar a transposição das redes de parentesco para o programa, assim como facilitar a implementação de algoritmos mais complexos.

Em seguida foi realizada uma etapa de implementação de um algoritmo busca por anéis extensivamente, isto é, realiza a busca testando todas as combinações possíveis. Após esta etapa o trabalho com as cores do grafo foi iniciado. Ele foi divido em estudar, propor e implementar idéias sobre utilização das cores com objetivo final de encontrar anéis cromáticos. A implementação se utilizou do algoritmo extensivo como base.

Após concluidos os algoritmos, foram propostos modos de paralelizar a busca pelos anéis cromáticos. Um estudo de caso sobre a rede Arara foi realizado com todos os modos de busca a anéis implementados.

2 FUNDAMENTAÇÃO DA ANTROPOLOGIA COMPUTACIONAL: ELEMENTOS BÁSICOS E FERRAMENTAS

Nas seções seguintes, os conceitos fundamentais que estão presentes no trabalho são descritos, assim como é realizado um breve relato de soluções existentes que tem como objetivo auxiliar uma área da Antropologia a analisar redes de parentesco.

2.1 REDES DE PARENTESCO

As redes de parentesco são construídas a partir de relações sociais de filiação ou casamento que determinados indivíduos apresentam. Relações de filiação indicam indivíduos que se consideram pais/mães, filhos/filhas, enquanto que as de casamentos são relações reconhecidas de alguma forma pela sociedade (AGHASSIAN; AUGÉ; BESSA, 1975).

Dentro de uma rede de parentesco será sempre identificado um Ego a partir do qual se procederá a análise. Todas as relações são descritas tomando Ego como referência. Utiliza-se a notação "M"(Mãe), "F"(Pai), "Z"(Irmã), "B"Irmão, "D"(Filha), "S"(Filho), "W"(Esposa) e "H"(Marido) para denotar uma ligação em relação ao Ego, como, por exemplo, utilizando ”MBS” para representar um primo do Ego (filho do irmão da mãe), utilizando a forma inglesa de nomeação, conforme a literatura da área. A Figura 1 mostra uma rede de parentesco e as denominações dos seus indivíduos a partir de Ego.

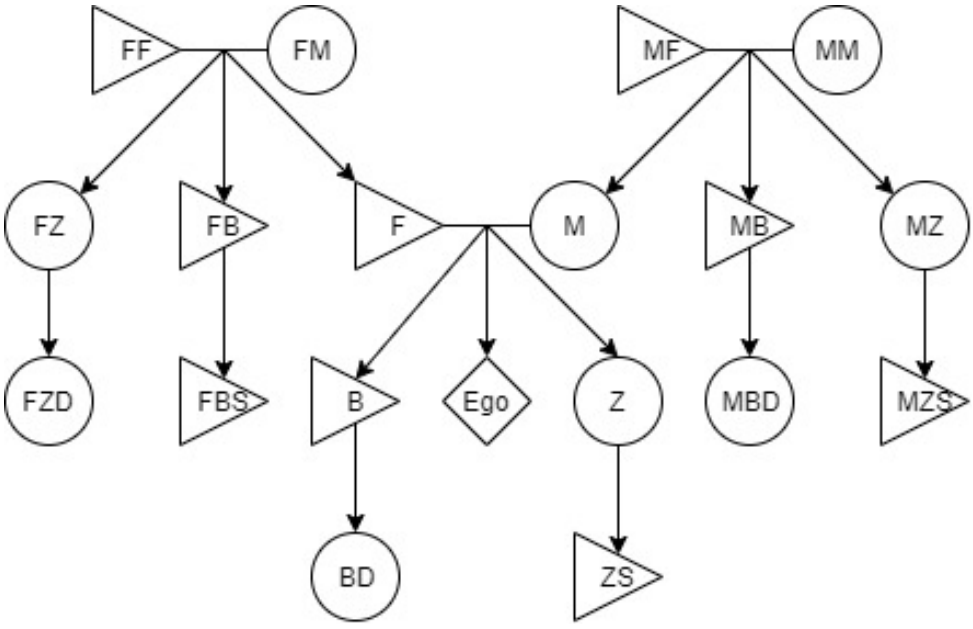


Figura 1 – Uma representação gráfica de Ego e seus parentes

2.2 TEORIA DAS ALIANÇAS MATRIMONIAIS

Um casamento entre dois indivíduos pode representar desde um complexo de normas que sancionam as relações entre eles, formalizados por cerimônias rituais dirigidas por membros da comunidade (AGHASSIAN; AUGÉ; BESSA, 1975) até quaisquer uniões que tenham prole reconhecidas, a depender da sociedade que é estudada. A hipótese que consiste na teoria das alianças matrimoniais afirma que casamentos não apenas criam laços, mas também são determinados por eles, sendo uma possibilidade a de que os padrões de ciclos (anéis) que acontecem em uma rede indicam um padrão para formação dos casamentos (HAMBERGER *et al.*, 2004). Dessa forma, uma proposta é enumerar os anéis para depois realizar uma análise sobre eles, procurando por padrões que refletem no comportamento social dos indivíduos.

2.3 ANÉIS

Um anel é uma subrede que forma um ciclo e contém pelo menos um casamento. Cada anel descreve uma forma de relacionar indivíduos casados por relações de consanguinidade, definindo o interesse de estudo da área de antropologia do parentesco (HAMBERGER *et al.*, 2004). Um anel é formado por relações de consanguinidade e afinidade, sendo denotado neste texto por A*i*C*j*, com *i* indicando o número de relações de afinidade (casamentos) e *j* indicando o número de vínculos de consanguinidade (filiação) no anel. De maneira didática, pode-se dizer que um anel é um *caminho especial* na rede de parentesco que

- Forma um ciclo;
- Começa e termina no Ego;
- Não há repetições de vértices, a menos do Ego; e
- Não possui vértices de grau de entrada igual a 2.

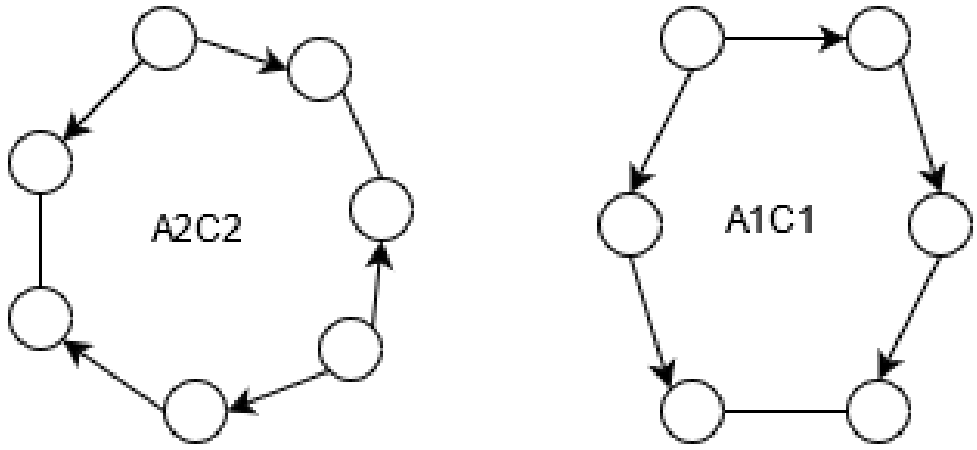


Figura 2 – Exemplos de anéis A2C2 e A1C1

2.4 ARARA: UMA REDE DE PARENTESCO REAL

Arara é um povo que habita a região do vale médio Xingu, no Pará, atualmente com duas áreas legalmente definidas: Terra Indígena Arara e Terra Indígena Cachoeira Seca do Iriri. Foram considerados extindos por volta da década de 1940, retornando ao conhecimento no início dos anos de 1970, após construção da rodovia Transamazônica. Tradicionalmente estabeleciam vários subgrupos, embora atualmente a maioria viva em uma aldeia na Terra Indígena Arara (TEIXEIRA-PINTO, 1998). A rede utilizada, coletada pelo antropólogo Prof. Dr. Márnio Teixeira-Pinto conta com 389 membros, 104 casamentos e 727 relações de filiação. Utilizaremos os dados desta rede para os experimentos e análises.

2.5 ALGUMAS FERRAMENTAS

Nesta seção são apresentadas algumas ferramentas utilizadas por antropólogos para analisar redes, realizando uma breve introdução sobre suas funcionalidades.

2.5.1 MaqPar

A MaqPar é uma ferramenta que enumera anéis de redes de parentesco. Na sua primeira versão a ferramenta utilizou a linguagem SQL sobre MS-Acess (NETO; SILVA, 2009). Hoje, a ferramenta é multiplataforma, mais eficiente, e disponível via WEB (`www.maqpar.ufsc.br`). Um dos objetivos deste trabalho é desenvolver uma nova funcionalidade para esta ferramenta, a enumeração de anéis cromáticos.

2.5.2 Pajek

Pajek é um programa muito utilizado para analisar redes grandes, disponibilizado gratuitamente. Possui várias atualizações e melhorias desde o seu projeto inicial (BATAGELJ; MRVAR, 1998), (BATAGELJ; MRVAR, 2014). Esta ferramenta possibilita operações sobre a rede, como por exemplo o particionamento de indivíduos por geração, onde os indivíduos são atribuídos a uma geração de acordo com a de seus pais ou o particionamento da rede em subredes contidas na rede maior mas com propriedades específicas, como por exemplo subredes conexas. A ferramenta também auxilia no desenho das redes.

2.5.3 Kinship2

A ferramenta kinship2 é utilizada para trabalhar com redes de parentesco que possuam informações sobre pedigree (identidade por descendente). É descrita em (SINNWELL; THERNEAU; SCHAID, 2014) e utiliza estes valores para descrever relações familiares de parentesco e descendência, assim como para desenhar *layouts*

gráficos. A ferramenta é escrita em linguagem R e possui compatibilidade com outras ferramentas.

No próximo capítulo alguns algoritmos fundamentais em grafos são descritos. Estes algoritmos darão a base necessária para o desenvolvimento de novos algoritmos em grafos.

3 FUNDAMENTAÇÃO COMPUTACIONAL: ALGORITMOS EM GRAFOS

Neste capítulo descrevemos como representar um grafo em um computador utilizando os modos de matriz de adjascência e lista de adjascência. Grafos é uma estrutura escolhida para modelar relações entre pares de objetos. Para o nosso caso, onde trabalhamos com redes de parentesco, alguns algoritmos conhecidos são utilizados para encontrar outras estruturas de interesse. Além disso, os grafos conseguem representar todas as propriedades e características de uma rede de parentesco com facilidade.

Posteriormente são descritos os algoritmos de busca em largura, busca em profundidade, busca por componentes conexas e ordenação topológica, algoritmos básicos da área de grafos. A busca em largura é utilizada como base para implementar a busca por componentes conexas na seção *3.2.4*. A busca em profundidade é a base para implementação da ordenação topológica, que é utilizada para encontrar todas as junções do grafo, um passo fundamental para encontrar todos os anéis com descrição na seção *4.1*. A busca em profundidade também é utilizada para colorir o grafo, na seção *5.1.2* e para um algoritmo que encontra atributos relacionados as cores do grafo, na seção *5.2.2*. Um grafo conexo é um grafo onde existem um caminho entre quaisquer pares de vértices. Uma componente conexa é um subgrafo maximal, isto é, com a maior quantidade de vértices possíveis, em que o subgrafo seja conexo. Componentes conexas de um grafo é outro tópico importante para este trabalho pois se em um grafo existem mais de uma componente conexa, a procura por anéis pode ser dividida entre elas sem perda de anéis, apenas eliminando buscas que não gerariam resultados.

3.1 REPRESENTANDO GRAFOS COMPUTACIONALMENTE

Um grafo é uma estrutura descrita por um conjunto de vértices e um conjunto de arestas (pares de vértices). Um grafo dirigido (ou digrafo) é um grafo que possui vértices e arcos (pares ordenados de vértices) (KLEINBERG; TARDOS, 2006). Um grafo com vértices, arcos e arestas é denominado *grafo misto* e foi essa a estrutura escolhida para representar as redes de parentesco que serão utilizadas como entrada para os algoritmos. É importante notar que um indivíduo é representado como um vértice, um casamento como uma aresta e uma relação de filiação como um arco.

Duas formas frequentemente utilizadas para representar em um programa um grafo G com n vértices, são representações por matriz de adjacência e lista de adjacência. A primeira representação corresponde a uma matriz com linhas e colunas ($n \times n$) em que cada entrada (i,j) possui informações sobre qual o tipo da relação que i possui com j. A segunda representação consiste em uma lista com elementos, onde cada elemento i contém seus adjacentes (CORMEN *et al.*, 2002).

Como ambas as representações possuem vantagens em relação a sua implementação, ambas foram utilizadas, sendo que os algoritmos iniciais foram implementados em relação aos dois modos, com os algoritmos seguintes utilizando a representação mais adequada ao algoritmo.

Como o grafo de entrada é um grafo misto, cada entrada da matriz de adjacência deste grafo possuirá 1 valor de 6 possíveis. A entrada (*i*,*j*) pode ser:

- uma aresta entre *i* e *j*;
- um arco de *i* para *j*;
- um arco de *j* para *i*;
- uma aresta entre *i* e *j*, e arco de *i* para *j* – caso onde temos um casamento entre filhos e pais;
- uma aresta entre *i* e *j*, e arco de *j* para *i* – idem caso anterior; ou
- uma entrada vazia, se não houver relação entre *i* e *j*.

3.2 ALGORITMOS CLÁSSICOS

Alguns algoritmos clássicos sobre grafos foram implementados para obter familiaridade com a manipulação desta estrutura, assim como para manipular procedimentos simples utilizados posteriormente em algoritmos mais complexos. Foram implementados e descritos nesta seção algoritmos de busca em largura, busca em profundidade, busca de componentes conexas e ordenação topológica.

Estes algoritmos apresentam complexidade de tempo na ordem de $O(n + m)$, com n representando o número de vértices e m o número de arcos em um grafo. Para a busca por componentes conexas e ordenação topológica esta é a complexidade típica. Para ambas as buscas, a complexidade é $O(m)$, quando os grafos de entrada são *conexos*. Porém, como não há garantia de conexidade dos grafos de entrada, as buscas foram ajustadas para devolver florestas, acarretando na necessidade de adicionar n no seu tempo de execução, justamente para cobrir os casos em que o grafo possui mais vértices que arcos.

3.2.1 Busca em Largura

A busca em largura é um algoritmo que atua sobre um subgrafo induzido pelos arcos *G* e um vértice *v*, devolvendo uma árvore de busca em largura sobre os descendentes de *v*. A árvore resultante contém todos os vértices em *G* que são alcançáveis a partir de *v*, organizados em camadas relacionadas com a distância, isto é, o menor número de vértices possíveis entre *v* e um descendente. Este resultado é obtido percorrendo uniformemente os novos descendentes derivados de *v*, os marcando para não ocorrer visitas duplicadas.

Uma árvore pode ser representada como um vetor, onde cada índice *i* contém o valor correspondente ao pai do vértice representado por *i* na árvore. Utiliza-se um valor fictício para representar *o pai* da raiz. Uma árvore pode também ser representada utilizando nós, que contém informações sobre o índice e quais são seus próprios filhos e pai. Uma floresta é um conjunto de árvores disjuntas, podendo ser representada como um vetor com mais de um valor representante de raiz ou um conjunto de nós raízes.

Algoritmo 1: *Busca em Largura*

```
Entrada : Um grafo G e um vértice x ∈ G
Saída   : Para todos os vértices v alcançáveis a partir de x, v.pai contém o
          índice do pai de v na árvore
para ∀ v ∈ G faça
    se v == x então
        v.visitado = verdadeiro
    senão
        v.visitado = falso
x.pai = NULO
Q = FILA
Q.adicionar(x)
enquanto Q.tamanho > 0 faça
    u = Q.remover
    para v ∈ u.adjascentes faça
        se v.visitado == falso então
            v.visitado = verdadeiro
            v.pai = u
            Q.adicionar(v)
```

As buscas em largura foram implementadas de tal forma a devolver florestas contendo todos os vértices do grafo, realizando buscas em larguras sucessivas. A busca em largura em relação a matriz de adjacências devolve uma floresta representada por um vetor, enquanto que a busca em relação a lista de adjacência devolve uma nova lista contendo nós que representam as raízes das árvores encontradas. A busca se inicia com o recebimento da estrutura que representa o grafo e criação de novas estruturas para armazenar os atributos utilizados neste procedimento. Todo vértice de *G* é inserido em uma lista de vértices livres *l*, que é percorrida. Esta procura ocorre retirando o primeiro vértice *v* de *l*, onde *v* tem seus atributos atualizados e descendentes com atributos inalterados adicionados a uma fila *f*. A fila *f* é então percorrida até ficar vazia, alterando os atributos de cada vértice *u* analisado, inserindo descendentes de *u* inalterados no final de *f* e os removendo de *l*. Todo vértice analisado tem também seu valor em relação a floresta atualizado com base nos valores contidos em seus atributos. No final da busca *l* está vazia e todos os vértices de *G* já foram percorridos e estão inseridos na floresta.

3.2.2 Busca em Profundidade

A busca em profundidade atua também sobre um grafo e um vértice, devolvendo uma árvore, porém seu método consiste em ir mais "ao fundo" a cada iteração da busca, isto é, em cada passo um vértice não selecionado anteriormente é escolhido e percorrido. Este processo continua até não haver mais nenhum vértice a ser escolhido. Dessa forma, o algoritmo pode voltar a um passo anterior e tentar escolher outro vértice ainda não selecionado.

Algoritmo 2: *Busca em Profundidade*

Entrada : Um grafo G
Saída : Para todos os vértices v em G, $v.pai$ contém o índice do pai de v na floresta

```
para ∀ v ∈ G faça
    v.pai = NULO
para ∀ v ∈ G faça
    se v.pai == NULO então
        Busca em Profundidade(G, v)
```

Algoritmo 3: *Busca em Profundidade*

Entrada : Um grafo G e um vértice $x \in G$
Saída : Para todos os vértices v alcançaveis a partir de x e ainda não análisados, $v.pai$ contém o índice do pai de v na floresta

```
para v ∈ x.adjascentes faça
    se v.pai == NULO então
        v.pai = x
        Busca em Profundidade(G, v)
```

Implementada para devolver florestas, busca em profundidade realiza chamadas a procedimentos de busca em profundidade até uma floresta contendo todos os elementos do grafo ter sido formada. A representação do grafo G é recebida pelo procedimento e a floresta devolvida tem formato de vetor caso a representação seja matriz de adjacência, ou uma lista de nós raízes, caso lista de adjacência. O procedimento inicializa estruturas para armazenar atributos sobre cada vértice de G, inseridos em uma lista I. Esta lista é percorrida até ficar vazia, realizando uma chamada recursiva a um método, passando como argumento o primeiro elemento v de I. Este método recursivo atualiza os atributos de v e chama recursivamente todo descendente com atributos ainda inalterados, até algum vértice não possuir mais descendentes aptos a serem chamados. Cada chamada ao procedimento retira o vértice principal de I, assim como o insere na floresta resultante, com base nos atributos do vértice.

3.2.3 Busca de Componentes Conexas

Neste trabalho uma componente conexa de um grafo misto G é um subgrafo de G em que todos os vértices possuem ligações entre si, sejam elas arestas ou arcos (CORMEN *et al.*, 2002). Para implementar a busca de componentes conexas foi utilizado matriz de adjacências como entrada e listas de listas de inteiros como saída, cada lista contendo os índices de vértices que representam uma componente de G.

Algoritmo 4: *Busca de Componentes Conexas*

```
Entrada : Um grafo G
Saída   : G.componentes contendo todos os conjuntos de componentes
          conexas
L = LISTA
G.componentes = ∅
para ∀ v ∈ G faça
    L.adicionar(v)
    v.visitado = falso
enquanto L.tamanho > 0 faça
    Q = FILA
    C = CONJUNTO
    v = L.primeiro
    v.visitado = verdadeiro
    Q.adicionar(v)
    enquanto Q.tamanho > 0 faça
        v = Q.remover
        C ∪ v
        para x ∈ v.adjascentes faça
            se x.visitado == falso então
                Q.adicionar(x)
                L.remover(x)
                x.visitado = verdadeiro
    G.componentes ∪ {C}
```

O primeiro passo da implementação foi adicionar todos os vértices de G em uma lista l. Para o primeiro elemento da lista é realizado uma busca em largura especial, que procura por vértices, arcos e arestas deste elemento e devolve os índices encontrados. Após busca em largura finalizar, os índices encontrados são retirados de l. Este processo é repetido até l ficar vazia, momento em que todos os vértices de G estarão em alguma componente.

3.2.4 Ordenação Topológica

A ordenação topológica é uma ordenação linear de todos os vértices em G, de forma que se G possui um arco (i,j) então i aparece antes de j na ordenação (CORMEN *et al.*, 2002). Uma ordenação topológica é possível somente em grafos

dirigidos e acíclicos. Um grafo acíclico é um grafo onde não existem ciclos dirigidos, critério coerente com os grafos dirigidos utilizados, já que em uma rede de parentesco induzida pelos arcos nenhum ciclo dirigido pode existir, pois um indivíduo não pode ser ancestral de seu pai (ou dele próprio).

Algoritmo 5: *Ordenação Topológica*

```
Entrada : Um grafo G
L = LISTA
para v ∈ G faça
    v.visitado = Falso
para v ∈ G.fontes faça
    Ordenação Topológica(v, L)
```

Algoritmo 6: *Ordenação Topológica*

```
Entrada : Um vertice v e uma lista L
v.visitado = Verdadeiro
para x ∈ v.adjascentes faça
    se x.visitado == Falso então
        Ordenação Topológica(x, L)
L.adicionar(v)
```

A ordenação topológica foi implementada utilizando um procedimento que opera com o grafo no formato de lista de adjacência e utiliza *templates* para a estrutura que armazena o resultado, terminando com ordenações topológicas para pilhas e ordenações topológicas inversas para listas, filas, vetores e conjuntos, ou seja, para percorrer a ordenação topológica nestas estruturas com ordenação inversa as estruturas precisam ser percorridas de forma inversa. Este procedimento recebe um grafo G e um vértice v, como ponto de partida, inserindo na estrutura destino os vértices já ordenados. A implementação inicializa os atributos dos vértices de G que serão utilizados e aplica sobre v um algoritmo recursivo que recebe um vértice i como argumento, marca i e aplica o algoritmo sobre os descendentes não marcados de i. Quando todos os filhos de i estão marcados (tiveram o algoritmo aplicado sobre eles), i é inserido na estrutura.

No próximo capítulo serão apresentados os passos utilizados para enumerar anéis em geral. Isso significa que os vértices podem ter informações de cores ou não. Se tiverem, tais informações não serão utilizadas pelo processo de enumeração.

4 SOBRE A ENUMERAÇÃO DE ANÉIS

O procedimento para encontrar anéis implementado foi apresentado em (FERREIRA; FRANCO; SILVA, s.d.). Ele consiste em encontrar todas as junções dos pares de vértices na entrada, construir conjuntos sobre as combinações de *k* casamentos (sendo *k* o número de casamentos que se deseja encontrar o anel) e procurar anéis sobre cada conjunto especificado. A primeira parte da busca consiste em criar estruturas que contém as informações de um destes conjuntos. Seguindo pelos conjuntos, cada informação é combinada e é realizado uma verificação sobre a existência de um anel na combinação. Se um anel for encontrado, ele é cortado de acordo com um método para ser representado por uma reta. A representação linear de um anel é utilizada para facilitar a análise sobre os anéis. O corte dos anéis é um problema novo, onde uma solução está sendo proposta neste trabalho. Ele ocorre em anéis com dois ou mais casamentos onde é cortado o casamento o qual acreditamos ser determinado pelo outro, ou seja, o mais recente.

4.1 JUNÇÕES

Uma junção em um grafo depende de um tripla de vértices *s*, *u* e *v*, onde *s* é denominado uma junção de *u* e *v* se existirem caminhos com origem em *s* e destinos em *u* e *v* em que o único vértice em comum seja *s*, em outras palavras, existe pelo menos um caminho de *s* até *u* excluindo origem que não compartilha vértices com, pelo menos, um caminho de *s* até *v*.

A Figura 3 mostra um grafo com 7 vértices, em que para a dupla de vértices *u* e *v* existem 2 junções, v_1 e v_3. Podemos ver que os vértices v_0, v_1, v_2 e v_3 possuem pelo menos dois caminhos até *u* e *v*, mas somente os vértices junções de *u* e *v* possuem mais de um caminho disjunto até eles, pois para os vértices v_0 e v_2 todas as possibilidades de caminho obrigatoriamente passam pelo vértice v_3 e v_4 não possui caminho para *u*.

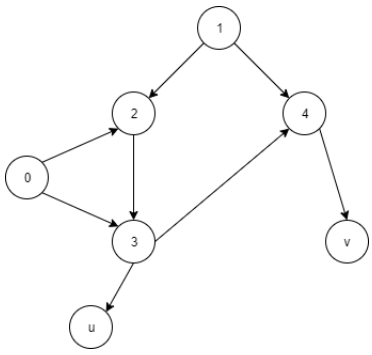


Figura 3 – Exemplo de uma junção

O procedimento implementado recebe a representação do grafo e utiliza lista de

adjacência para devolver uma nova estrutura, que contém todas as junções de todas as possibilidades de pares de vértices do grafo.

A implementação do procedimento para encontrar as junções de um grafo G (com n vértices, m arcos e k arestas) pode ser dividida em duas partes. Na primeira ocorre a implementação do algoritmo descrito em (FERREIRA; FRANCO; SILVA, s.d.), percorrendo todos os vértices v de G e utilizando uma fila f para receber a ordem topológica com v sendo uma fonte. A cada iteração com um vértice v, são descobertos todos os pares de vértices para as quais v é junção. Esta descoberta ocorre percorrendo f e atribuindo para cada vértice uma partição, com base nas partições de seus pais. Como f está ordenada topologicamente, os pais de um vértice que está sendo analisado certamente já foram analisados ou não são descendentes de v e portanto, não serão analisados. Ao final da análise, todos os vértices que estão em partições diferentes e foram encontrados são considerados elementos de uma junção v.

A segunda parte da implementação consiste em organizar as junções encontradas em uma estrutura que facilita o acesso. A organização acontece utilizando uma matriz, indexada por pares de vértices, para adicionar toda junção encontrada ao índice correspondente aos vértices sobre os quais a junção é aplicada.

A complexidade para encontrar todos os pares de vértices para os quais v é junção é $O(t)$, com t igual ao número de arcos do subgrafo que contém todos os descendentes de v. Como a primeira parte do procedimento encontra todas as junções de um grafo G, sua complexidade é $O(nm)$, pois o algoritmo cujo tempo é $O(t)$ será executado para cada vértice, ou seja, n vezes. E como $O(t) = O(m)$, o tempo total gasto é $O(nm)$. A complexidade da segunda parte da implementação é $O(n^3)$, derivado do número de junções que é possível encontrar para todo par.

4.2 ENUMERAÇÃO DE ANÉIS

Para encontrar os anéis a implementação busca todas as junções para os pares de vértices do grafo, todos os casamentos (relações através de arestas) e executa um procedimento de acordo com o tipo de anel que se deseja encontrar, passando como argumento o número de casamentos k que este anel terá. O procedimento combina os casamentos de acordo com k, operando sobre todas as combinações possíveis com tamanho k. Por exemplo, se na busca por anéis com dois casamentos tivermos os casamentos $c1$, $c2$ e $c3$, as combinações formadas serão $c1$ com $c2$, $c1$ com $c3$ e $c2$ com $c3$.

Para cada combinação encontrada, agrupamentos (pares de vértices casados não necessáriamente entre si e que podem ajudar a formar anéis) são definidos. Sobre cada par presente em um agrupamento uma estrutura auxiliar também é definida, estrutura que contém as junções e os caminhos disjuntos de cada junção até o par. As estruturas auxiliares de todos os pares que estão em um determinado agrupamento

são submetidas a um procedimento recursivo, que verifica todas as possíveis combinações de caminhos, retornando um anel quando uma combinação que contém todos os casamentos é encontrada e tal combinação não possui vértices em comum.

4.2.1 Encontrando casamentos, combinações e agrupamentos no grafo de entrada

Para encontrar os casamentos, a matriz de adjacência triangular superior do grafo é percorrida e os casamentos encontrados são adicionados a uma lista. As combinações são grupos de casamentos com tamanho igual ao número de afinidades do anel, encontrados por um método que recebe a lista com os casamentos e os combina conforme o número de afinidades da busca. Por último, sobre cada uma destas combinações é utilizado linhas de código manuais e simples para transformá-las em agrupamentos de pares de vértices, agrupando os vértices de forma que eles podem ajudar a formar anéis. Na Figura 4 podemos ver os casamentos transformados em combinações com duas afinidades e por último, em agrupamentos. Os casamentos de v_1 com v_2, v_3 com v_4 e v_5 com v_6 são transformados em três combinações de vértices, $v_1 - v_2 - v_3 - v_4$, $v_1 - v_2 - v_5 - v_6$ e $v_3 - v_4 - v_5 - v_6$. Cada uma destas combinações é transformada em dois agrupamentos, como por exemplo a combinação $v_1 - v_2 - v_3 - v_4$ sendo transformada nos agrupamentos de $v_1 - v_3$ com $v_2 - v_4$ e $v_1 - v_4$ com $v_2 - v_3$.

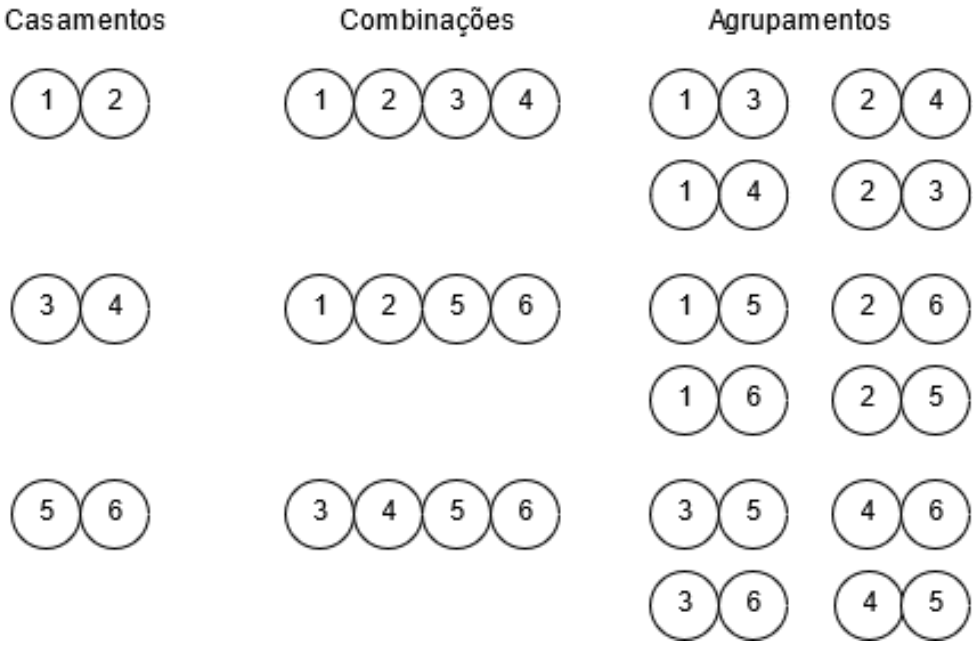


Figura 4 – Exemplos de casamentos, combinações e agrupamentos

4.2.2 Encontrando uma estrutura auxiliar para encontrar anéis

A estrutura utilizada para auxiliar na busca de um anel contém um conjunto com todas as junções de um par de vértices $J(i,j)$, assim como para cada junção $x \in J(i,j)$ todas as combinações de caminhos de x até i e j com apenas x em comum. Para armazenar estas informações, todos os caminhos de x até i e j são encontrados por uma busca. Esta busca utiliza um algoritmo recursivo que busca profundamente pelas possibilidades de caminho, fixando x e os destinos i ou j. Os caminhos até i são confrontados com os caminhos até j, adicionando na estrutura aqueles que são disjuntos, isto é, não possuem vértices iguais.

4.2.3 Encontrando anel

Para encontrar um anel o procedimento recebe uma lista com as estruturas encontradas anteriormente e informações sobre o anel até o momento (vértices, casamentos e junções). A cada chamada recursiva do procedimento a primeira estrutura da lista é removida, e para cada uma de suas combinações de caminhos disjuntos diferentes, os dados são adicionados as informações recebidas sobre o anel, sendo realizada uma nova chamada ao procedimento. O procedimento é realizado recursivamente até a lista das estruturas ficar vazia, momento no qual o procedimento de verificação de um anel é chamado.

4.2.4 Verificando anel

A verificação se um anel foi encontrado ocorre analisando os caminhos e as junções recebidas, sendo que um novo anel é formado quando nenhum vértice nos caminhos se repete. A estrutura que forma um anel contém os vértices que o formam ordenados por um corte, realizado sobre um homem (Ego) de algum dos casamentos de acordo com um critério definido, como podemos ver na Figura 5. Esta estrutura também armazena as junções, os caminhos disjuntos destas junções e os casamentos.

Após armazenamento das junções, casamentos e caminhos, a ordem do anel é definida escolhendo um homem do primeiro casamento. A partir deste homem o caminho do qual ele faz parte é inserido no anel, e o próximo caminho a ser procurado é o caminho que contém o vértice casado com o último elemento inserido. Se o homem escolhido não é o correto de acordo com o método de corte, o anel é calculado novamente.

4.3 CORTANDO ANÉIS

Para possibilitar uma análise de um anel com 2 ou mais casamentos um corte em um dos casamentos é realizado, para representar o anel linearmente durante a

análise. Visto que o objetivo do estudo antropológico dos anéis é estudar como os casamentos recentes foram influenciados pelos antigos, surgiu um novo problema: Qual casamento realizar este corte? Idealmente ele seria realizado sobre os casamentos mais recentes, mas para redes sem datas em seus casamentos, como identificar qual casamento é o mais recente?

A solução proposta por este trabalho consiste em contar o número de ancestrais de cada casamento até a junção e cortar o anel no casamento que apresenta maior soma dos ancestrais, supondo que indivíduos que tenham mais conexões até os antepassados em comum seriam indivíduos mais novos. Esta maneira de cortar anéis foi testada em relação aos anéis A2C2 da rede *Marcapata*, rede que possui datas em 701 de seus 1587 casamentos. Nesta rede foram encontrados 188 anéis A2C2 com datas em ambos casamentos. Destes 188, 78 (aproximadamente 42%) apresentavam somas entre os casamentos iguais, não sendo cortados pelo método e realizando um corte no primeiro casamento que aparece no arquivo. Dos 110 anéis com datas reais em que o corte foi efetivamente realizado, 98 (aproximadamente 89%) obtiveram *acertos* e 12 (aproximadamente 11%) obtiveram *erros*.

A Figura 5 demonstra um exemplo de corte realizado em um anel com 7 vértices, com casamentos entre os vértices v_0,v_6 e v_3,v_4. O corte transforma o anel em uma linha, tornando mais fácil a análise. No exemplo abaixo, o vértice v_0 tem 2 ancestrais, enquanto os outros vértices casados, v_3, v_4 e v_6 tem somente 1. Como a soma dos ancestrais do casamento v_0,v_6 é maior que a soma dos ancestrais do casamento v_3,v_4, o anél é cortado no casamento v_0,v_6.

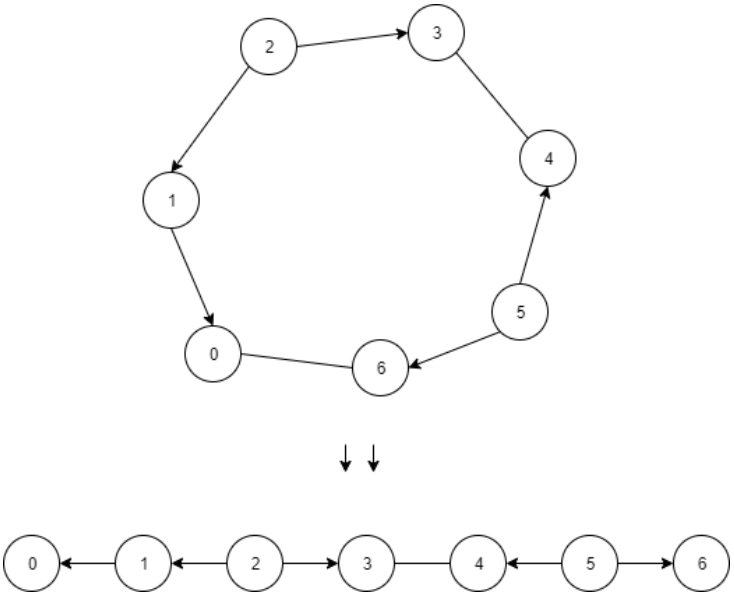


Figura 5 – Corte de um anel

No próximo capítulo é apresentado como a coloração foi tratada neste trabalho, alguns modos de coloração de vértices implementados, como os anéis cromáticos

foram encontrados e finalmente como foram paralelizados.

5 ENCONTRANDO ANÉIS CROMÁTICOS

Uma forma de encontrar os anéis cromáticos é encontrar anéis que possuam caminhos com um determinado número de cores, otimizando a busca para coincidir com este número. Para isto, este capítulo propõe alguns modos de busca, assim como os algoritmos utilizados para calcular as informações utilizadas por estes modos. Neste capítulo também é descrito o modo de implementação das cores, como os vértices foram coloridos a partir de uma hipótese sobre a rede Arara e algumas formas utilizadas para encontrar os anéis cromáticos. Por último, são descritos como os algoritmos que buscam anéis foram paralelizados.

5.1 COLORAÇÃO

Um atributo utilizado sobre os grafos é sua coloração, através da qual cada vértice possui uma cor. Esta cor pode ser derivada de propriedades relacionadas as redes de parentesco reais ou calculada através de hipóteses antropológicas. Além de utilizar as cores para esta representação, podemos utilizá-las para acelerar a enumeração dos anéis.

5.1.1 Cores no grafo

A implementação de cores no grafo é realizada armazenando para cada vértice um conjunto de inteiros, que representa sua cor. Para o desenho de grafos utilizamos a ferramenta *Graphviz* que utiliza cores no formato textual. Por isso, foram criadas estruturas para relacionar os conjuntos de inteiros de um vértice com valores textuais de cores.

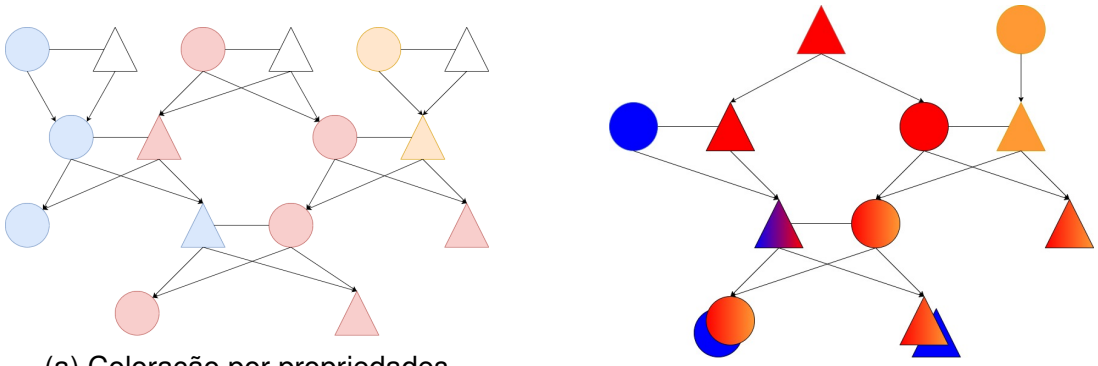
5.1.2 Modos de colorir - Uma hipótese antropológica

Na rede Arara, a filiação segue uma regra *bilateral* (podendo seguir ora pelo lado da mãe ora pelo lado do pai). Portanto os modos de coloração propostos procuram se adequar a esta propriedade.

Foram definidos dois modos de coloração, ambos com origens nas fontes do grafo, que começam os procedimentos com conjunto de cores contendo apenas valor igual ao número que o vértice representa. No primeiro modo, alguma fonte adiciona sua cor aos seus filhos, que continuam a propagar esta mesma cor até terminar em algum sorvedouro, procedimento que é repetido para todas as fontes do grafo. No outro modo, somente as fontes de um determinado sexo são coloridas, e somente elas passam a cor para os filhos, que continuam a propagação somente se também possuírem este atributo. A implementação de ambos os métodos é baseada na busca em profundidade,

onde a propagação de uma cor vai “ao fundo”, chegando aos sorvedouros para então continuar para os outros vértices.

A Figura 6 mostra os dois exemplos de coloração, com a coloração pela propriedade de elipse representada na Figura 6a e coloração geral representada na Figura 6b.



(a) Coloração por propriedades

(b) Coloração geral

Figura 6 – Modos de Coloração

5.2 ALGORITMOS SOBRE GRAFOS COLORIDOS

Para obter mais informações sobre os vértices do grafo e ter mais capacidade de melhorar a procura por anéis, alguns algoritmos que calculam atributos com relação às cores dos vértices foram propostos. Um dos atributos calculados para todo vértice, é o número de cores máximo e mínimo em caminhos até todos os vértices alcançáveis. Para este atributo são propostos dois algoritmos, 8 e 10. O algoritmo 8 conta com tempo exponencial no número de vértices enquanto que o algoritmo 10 conta com tempo exponencial no número de cores, mas polinomial no número de arcos e vértices quando olhamos somente para estes valores. O algoritmo 10 é a evolução do algoritmo 7, que encontra para todos os vértices o maior número de cores até sorvedouros. Estes atributos ajudam na escolha por uma busca de anéis mais efetiva. Um artigo sobre o algoritmo 7 foi desenvolvido e apresentado no Encontro de Teoria da Computação 2021. Neste artigo apresentamos o problema que o algoritmo 7 resolve, denominado por *caminhos supercoloridos*, mostramos que ele é NP-difícil quando a entrada é um grafo dirigido qualquer e descrevemos o presente algoritmo 7 demonstrando que o problema é tratável quando o grafo dirigido de entrada é acíclico e o número de cores é pequeno. Na próxima subseção apresentamos o problema dos *caminhos supercoloridos*.

5.2.1 Maior número de cores até sorvedouros

Um dos atributos que se deseja calcular é sobre o maior número de cores possíveis até um sorvedouro (vértice com grau de saída igual a 0), atributo denominado neste trabalho de valor super de um vértice. Para este cálculo foi proposto um algo-

ritmo, que recebe um grafo direcionado acíclico (DAG) e trabalha através da ordem topológica inversa, enviando para camadas superiores do grafo as cores e os conjuntos encontrados. Armazenar todos os subconjuntos é necessário pois se cada vértice realizar escolha do melhor conjunto local, vértices que ainda serão processados podem ter valores errôneos, como podemos ver na Figura 7, onde o vértice v_3 armazena um conjunto com as cores verde e laranja, conjunto que não é a melhor escolha para ele próprio, porém se torna essencial para escolha correta do valor super do vértice v_1.

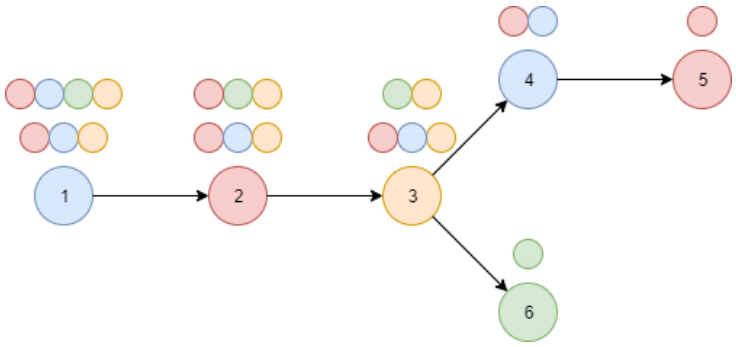


Figura 7 – Exemplo sobre a importância de armazenar todos os subconjuntos

Algoritmo 7: *Encontra o valor super para cada vértice de um DAG*

Entrada : um DAG D com n vértices coloridos e m arcos.
Saída : Um vetor $super(v)$ que contém o maior número de cores em caminhos iniciando em v para todo $v \in D$

Sejam $v_1, v_2, \ldots, v_n$, os vértices de D ordenados topologicamente (começando em sorvedouros e em direção às fontes).
Para cada vértice v, se v é sorvedouro então $cores(v) = \{cor(v)\}$; caso contrário, $cores(v) = \{\emptyset\}$.
para $i = 1, 2, \ldots, n$ **faça**
    $super(v_i)$ = maior conjunto em $cores(v_i)$
    para *cada pai* w *de* v_i **faça**
        para *cada conjunto* $p \in cores(v_i)$ **faça**
            $q = cor(w) \cup p$
            $cores(w) = cores(w) \cup q$

O algoritmo considera $super(v)$ como o maior número de cores possíveis a partir de v até algum sorvedouro e $cores(v)$ como os maiores conjuntos de cores que qualquer caminho a partir de v possui. Caminhos a partir de v com menos cores existem, mas em nenhum deles o conjunto de cores não é subconjunto de algum outro conjunto presente em $cores(v)$.

O algoritmo está correto pois um vértice w recebe todos os subconjuntos de cores de caminhos a partir de seus filhos, que são acrescidos pela cor de w (no algoritmo, última atribuição). Portando, se os filhos de um vértice w conhecem corretamente todos os subconjuntos de cores de caminhos iniciando neles, então w também conhecerá.

Para concluir a prova basta notar que os subconjuntos de cores são construídos em ordem topológica inversa (começando em sorvedouros e subindo em direção às fontes), portanto, todos os filhos de w já foram avaliados e possuem os subconjuntos de cores de caminhos iniciando neles.

O tempo de execução do algoritmo depende do número de subconjuntos que cada vértice pode guardar (primeira atribuição e terceiro laço). Este número pode ser considerado no processamento de cada arco (primeiro e segundo laços). Logo, o tempo de execução é no máximo $O(m2^k t)$, onde k é o número total de cores e t é o tempo de execução do comando de seleção e da segunda atribuição do laço mais interno[1]. Se cada subconjunto é armazenado em uma árvore binária de busca balanceada, t é da ordem de $O(k \log(2^k)) = O(k^2)$. Dessa forma, o tempo de execução do algoritmo é da ordem de $O(m2^k k^2)$. Se k é uma constante pequena, então o problema é tratável quando o número de cores do grafo é pequeno. Para obter o tempo $t = O(k^2)$, note que dois subconjuntos podem ser comparados quando consideramos os elementos de cada subconjunto em ordem lexicográfica. Com isso, uma comparação de duas chaves custa tempo $O(k)$ onde k é o total de cores no dígrafo. Como agora os subconjuntos são comparáveis, podemos armazená-los em uma árvore binária de busca balanceada. Portanto, o tempo de execução da operação de união da última atribuição no algoritmo é formado pelo tempo de comparar chaves em nós da árvore por cada nível da árvore que possui altura $O(\log(2^k))$. Logo, o tempo para adicionar um subconjunto na estrutura é $O(k \log(2^k)) = O(k^2)$.

5.2.2 Máximo e o mínimo número de cores até todo vértice do grafo

Outros atributos são os valores mínimos e máximos de cores presentes em caminhos de um vértice para qualquer outro vértice do grafo, atributos que auxiliam na poda de caminhos erroneos durante a busca por caminhos de uma junção até os vértices relacionados. Um exemplo de poda por caminhos pode ser observado na Figura 8, onde uma busca por caminhos com 2 cores de v_0 até v_5 podaria a busca para os vértices v_1 e v_3 pois ambos tem valores máximos e mínimos para v_5 iguais a respectivamente 3 e 0, nunca satisfazendo o critério de busca. Estes valores são encontrados realizando buscas em profundidade sobre todos os vértices do grafo, procurando ajustar os valores a cada iteração.

[1] Podemos assumir que o tempo da primeira atribuição do laço mais interno é dominado pelo tempo da segunda.

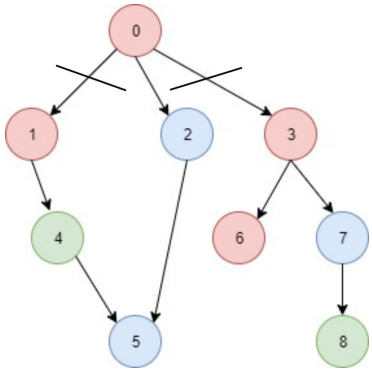


Figura 8 – Exemplo de poda na busca por caminhos

Algoritmo 8: *Máximo e mínimo*

Entrada : Um DAG D com n vértices coloridos e m arcos.

Saída : Para cada $v \in D$, vetores max e min, cada um contendo respectivamente o máximo e o mínimo número de cores até todos vértices $\in D$

para *cada* $v \in D$ **faça**
- $v.max = 0$
- $v.min = +\infty$
- $cores = \{\emptyset\}$
- *Encontrar max e min(v, v, cores)*

O algoritmo 8 recebe um grafo e atualiza para todo vértice do grafo os vetores max e min, inicializando-os respectivamente com valores mínimos e máximos possíveis. Depois ele inicializa um conjunto que armazena as cores utilizadas na computação, realizando uma chamada a um procedimento para o vértice.

Algoritmo 9: *Encontrar max e min*

Entrada : Um vértice v que é a fonte, w que é o vértice que está sendo analisado e uma árvore de busca binária *cores* que contém as cores encontradas até o momento.

Saída : v, com os vetores max e min atualizados

$cores \cup cor(w)$

se $|cores| \geq v.max(w)$ **então**
- $v.max(w) = |cores|$

se $|cores| \leq v.min(w)$ **então**
- $v.min(w) = |cores|$

para *cada filho x de w* **faça**
- *Encontrar max e min(v, x, cores)*

O algoritmo 9 procura de maneira recursiva sobre todos os descendentes de v, enquanto mantém um conjunto das cores para cada linha de execução. Se uma linha encontra um valor mais atual para os vetores em v, eles são atualizados.

O tempo de execução deste algoritmo é $\Omega(2^n)$, pois ele passa por todos os caminhos possíveis presentes em um grafo com inicio em v (para todo vértice v), propriedade que confirma sua corretude.

Estes valores também podem ser encontrados através de um outro algoritmo, alteração do algoritmo 7 e resultando no algoritmo 10. A alteração consiste em armazenar juntamente com os subconjuntos de um vértice a informação sobre o sorvedouro que originou aquele subconjunto. Após esta adição, os valores mínimos e máximos para os sorvedouros do grafo podem ser calculados percorrendo todos os conjuntos presentes em um determinado vértice. Para calcular o valor mínimo e máximo para todos os vértices do grafo é realizado várias iterações, cada uma podando os sorvedouros do grafo para calcular os valores para todos os vértices alcançáveis.

Algoritmo 10: *Encontra max e min até os sorvedouros para cada vértice de um DAG*

Entrada : Um DAG D com n vértices coloridos, m arcos e $\forall v \in D$ vetores *max* e *min* inicializados
Saída : Para todo $v \in D$ o array *max* e *min* atualizado com respectivamente o maior e o menor número de cores em caminhos iniciando em v até os sorvedouros $\in D$

Sejam $v_1, v_2, \ldots, v_n$, os vértices de D ordenados topologicamente (começando em sorvedouros e em direção às fontes).
Para cada vértice v, se v é sorvedouro então $cores(v) = \{cor(v)\}$ e $sorvedouro(cores(v)) = v$; caso contrário, $cores(v) = \{\emptyset\}$ e $sorvedouro(cores(v)) = 0$.

```
para i = 1, 2, ..., n faça
    para cada conjunto p ∈ cores(v_i) faça
        se max(sorvedouro(p)) ≤ |p| então
            max(sorvedouro(p)) = |p|
        senão se min(sorvedouro(p)) ≥ |p| então
            min(sorvedouro(p)) = |p|
    para cada pai w de v_i faça
        para cada conjunto p ∈ cores(v_i) faça
            q = cor(w) ∪ p
            cores(w) = cores(w) ∪ q
            sorvedouro(q) = sorvedouro(p)
```

O tempo de execução de todo este processo é $O(nm2^k k^2)$ com m representando o número de arcos, k o número de cores e n o número de vértices do grafo. A adição de informação sobre qual sorvedouro um conjunto se origina pode ser realizada em tempo constante, enquanto a operação de alteração dos vetores *max* e *min* (primeiro laço interno) é dominada pela operação de envio dos subconjuntos para os pais. O aumento n na complexidade, quando comparado ao tempo do algoritmo 7, vem da necessidade de remover os vértices sorvedouros do grafo a cada iteração mais

externa deste procedimento. No pior caso realizando uma remoção por iteração.

5.3 ANÉIS CROMÁTICOS

A melhoria da busca por anéis utilizando os valores min-max obtidos pela coloração dos vértices foi implementada utilizando tais valores para encontrar os caminhos de uma junção x até seus elementos i e j de forma mais efetiva. A busca por caminhos foi dividida por k buscas, cada uma procurando caminhos com número de cores específicas, variando de $x.min[i]$ e $x.min[j]$ até $k = x.max[i]$ e $k = x.max[j]$.

Algoritmo 11: *Encontrar caminhos*

Entrada : Vértice d destino, vértice v atual, inteiro k número de cores no caminho, inteiro n número de cores atual, conjunto l conjunto de cores atual, conjunto a de vértices do caminho atual e conjunto c destino

Saída : Conjunto c com os caminhos encontrados que tenham n cores

se $v.cor \notin l$ **então**
  $l \cup v.cor$
  $n = n - 1$
se $n \leq -1$ **então**
  retornar
se $d = v$ **então**
  se $k = n$ **então**
    $c \cup a$
  retornar
para $\forall$ *filho x de v* **faça**
  se *Não chega até d ou máximo de cores não chega em k* **então**
    continuar
  $a \cup x$
  Encontrar caminhos(d, x, k, n, l, a, c)

O procedimento descrito pelo algoritmo 11 é aplicado utilizando x como primeiro vértice atual, i e j como destino e todos os valores k para i e j, substituindo a busca por caminhos apresentada em *4.2.2*. O procedimento que realiza esta troca será denominado *Algoritmo 1 Colorido Sequencial*, enquanto o anterior será *Algoritmo 1 Não Colorido*. Utilizando este procedimento para encontrar os caminhos com cores um novo procedimento que busca por anéis foi desenvolvido, denonimado de *Algoritmo 2 Sequencial*. O *Algoritmo 2 Sequencial* armazena os caminhos em uma matriz previamente à procura das estruturas auxiliares, alterando a busca por estas estruturas para copiar os resultados já armazenados na matriz.

Para encontrar todos os caminhos o primeiro passo é encontrar os k pares de vértices entre os quais caminhos são necessários, onde o par de vértices u e v representa u como uma junção de v com outro vértice j, com u e j casados (não

necessáriamente entre si). Utilizando estes pares, a busca por caminhos é dividida em *i threads*. Cada uma destas *threads* procura por caminhos entre pares de vértices, inserindo os caminhos encontrados em uma matriz. Os pares em que uma thread trabalha são divididos em grupos de pares com tamanho *j*, onde $j = k / 10i$. Uma thread que termine de trabalhar em seu grupo de pares procura um novo grupo para trabalhar, até que todos os *k* pares tenham seus caminhos analizados e armazenado. Na Figura 9 vemos um exemplo em que *k* é igual a 12, *i* é igual a 2 e *j* é ajustado de 1 (arredondamento para cima de 12/20) para 4. Cada thread pega um número de pares igual a *j* e realiza o trabalho sobre ele, sendo que a primeira thread que terminar este trabalho irá pegar os *j* pares restantes. Pela Figura, vemos que o tempo de trabalho da thread 1 é menor que o tempo da thread 2, portanto após terminar seus pares atuais, ela trabalharia sobre os *j* pares restantes. Porém, ao olharmos para os tempos de execução dos pares restantes vemos que sua soma é igual a 8 unidades, deixando a thread 2 ociosa por 7 unidades de tempo enquanto a thread 1 termina a execução. Casos desta maneira mostram o motivo da divisão de $k/10i$ em detrimento de divisões simples de k/i.

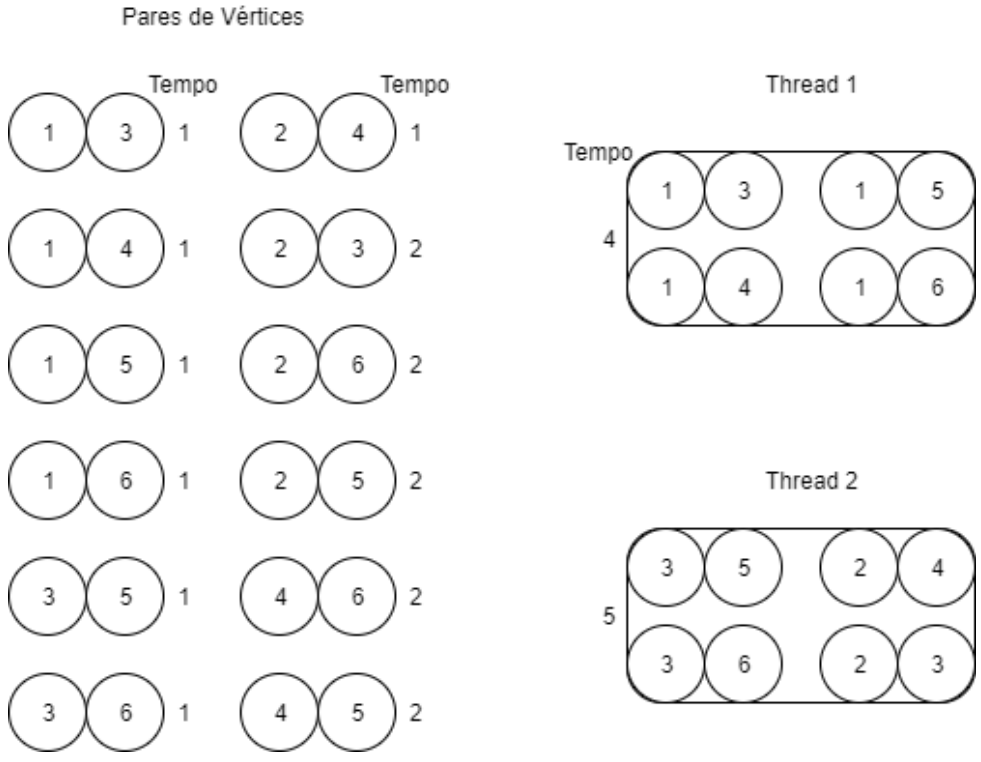


Figura 9 – Exemplo sobre a divisão do trabalho entre as threads

5.4 ANÉIS COLORIDOS PARALELOS

A busca por anéis coloridos paralelos representa a paralelização dos *Algoritmo 1 Colorido Sequencial* e *Algoritmo 2 Sequencial*, criando respectivamente os algoritmos

Algoritmo 1 Colorido Paralelo e *Algoritmo 2 Paralelo*, e também a criação de um novo algoritmo, denominado *Algoritmo 3* que realiza a busca pelos anéis cromáticos com um número específico de cores. Estes algoritmos utilizam *threads* para paralelizar a busca pelos anéis, tornando possível dividir as buscas em várias linhas e realizá-las simultaneamente. Os algoritmos, assim como suas especificações e siglas podem ser vistos na tabela 1.

Tabela 1 – Algoritmos de busca a anéis

Nome	Sigla	Utiliza Cores	Utiliza matriz com caminhos	Utiliza threads para os anéis
Algoritmo 1 Não Colorido	A1LSC	Não	Não	Não
Algoritmo 1 Colorido Sequencial	A1LCC	Sim	Não	Não
Algoritmo 1 Colorido Paralelo	A1P	Sim	Não	Sim
Algoritmo 2 Sequencial	A2L	Sim	Sim	Não
Algoritmo 2 Paralelo	A2P	Sim	Sim	Sim
Algoritmo 3	A3	Sim	Sim	Sim

5.4.1 Paralelizando a busca por anéis cromáticos

A paralelização dos procedimentos de busca ocorre dividindo a procura de anéis entre *i* threads, cada uma delas responsável por realizar pequenas buscas. As threads recebem um valor *g* que representa o número de combinações de casamentos sobre o qual elas efetuam uma busca por anéis. Sempre que uma thread fica disponível ela retira as combinações de uma estrutura comum a todas as threads e realiza sua própria busca sobre elas, ficando disponível novamente após a busca atual ser encerrada. Os valores de *i* e *g* são recebidos pelo procedimento.

Podemos ver na Figura 10 um exemplo de paralelização com *g* igual a 20 e *i* igual a 3. No exemplo temos 100 combinações e as threads retiram uma quantidade igual a *g* para realizar a busca. A primeira thread que terminar a busca sobre seu grupo irá buscar no conjunto de combinações restantes uma quantidade igual a *g* de novas combinações e operará sobre elas. Este ciclo continua até todas as combinações terem sido trabalhadas e seus anéis encontrados.

5.4.2 Encontrando Anéis Com Número Específico de Cores

Para implementação do algoritmo *A3* a busca é dividida entre *i* threads, onde $i = \min(x, y)$, com *y* igual ao número de cores do grafo e $x = 2kj$, com *k* representando o número de relações de afinidade do anel e *j* representando o maior valor *super* do

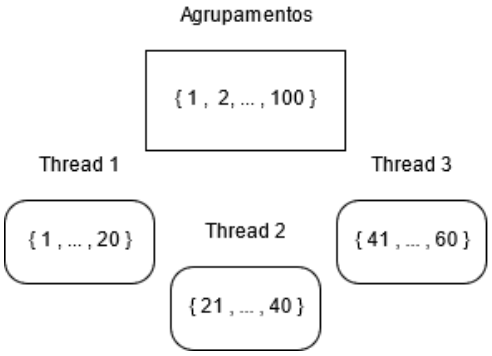


Figura 10 – Exemplo sobre a paralelização das buscas

grafo. O valor *i* representa o maior número de cores que um anel no grafo pode ter, pois ou é igual ao número total de cores do grafo ou é um valor obtido pela maior possibilidade de cores em caminhos que um anel pode apresentar. Para cada thread o procedimento continua a realizar as combinações e os agrupamentos de acordo com o número de casamentos, mas ao invés de procurar por todos os anéis ele procura por anéis que contenham um número de cores específico, cortando a busca quando este número não pode ser alcançado ou foi ultrapassado.

O procedimento do algoritmo *A3* altera a busca pela estrutura auxiliar para pré-calcular a matriz de caminhos, assim como para realizar união entre os conjuntos de cores dos caminhos disjuntos unidos por alguma junção. A estrutura também armazena qual o maior e o menor número de cores possíveis para algum caminho disjunto encontrado, informação que depois é transposta para o maior e menor número de cores em caminhos para qualquer estrutura auxiliar presente na busca, utilizado para verificar se o número de cores desejado não pode ser alcançado.

Todas as estruturas encontradas para uma combinação são enviadas para uma busca separada, que mantém também informações sobre os caminhos, casamentos e junções utilizadas, assim como um conjunto das cores presentes na busca atual e as informações sobre maior e menor caminho possível. Esta busca é recursiva e para cada estrutura auxiliar ela realiza novas execuções para todos os caminhos de todas as junções, realizando um corte na busca por caminhos se o número de cores atual ultrapassou *x* ou se *x* nunca pode ser alcançado.

No próximo capítulo é descrito como os resultados obtidos pelo trabalho foram apresentados, os tempos de execução dos algoritmos para a rede Arara na busca por anéis A1C1 e A2C2 e um levantamento númerico sobre os anéis cromáticos encontrados.

6 SOBRE A SAÍDA DOS DADOS E OS RESULTADOS OBTIDOS

Para analisar os resultados foram utilizados três modos de saída. O primeiro corresponde a escrita dos grafos, árvores e componentes em um formato especializado para ferramenta graphviz. O segundo corresponde a escrita textual em um arquivo sobre alguns dos resultados obtidos, como escrita dos máximos e mínimos de cores de um vértice até outros. O último modo corresponde a escrita em um arquivo em formato texto puro para ser aceito por leitores de planilhas, facilitando seu posterior manuseio por bancos de dados. Esta escrita contém a cada linha os anéis encontrados, assim como alguns atributos calculados sobre o anel.

Neste capítulo também apresentamos os tempos de execução dos algoritmos aplicados sobre a rede Arara na busca por anéis A1C1 e A2C2 assim como um levantamento numérico sobre os anéis cromáticos e suas cores, utilizando a coloração por propriedades.

6.1 GRAPHVIZ

Graphviz é um programa que recebe descrições de grafos em uma linguagem de texto específica e as transforma em diagramas ou desenhos de grafos. Neste trabalho foi utilizado a especificação de linguagem *dot* para descrever as estruturas escolhidas, especificação recomendada para grafos direcionados. Esta especificação realiza desenhos em camadas dos grafos dirigidos, procurando direcionar arcos para mesma direção enquanto tenta reduzir o tamanho de arcos e retirar sobreposições (GANSNER, 2009).

As estruturas que foram descritas utilizando a especificação *dot* foram as representações de árvores, componentes e grafos. Para isto algumas funções auxiliares foram utilizadas para instanciar as propriedades dos vértices que estão presentes neste componente, descrevendo o número do vértice, sua cor e seu formato (triângulo para homens e elipse para mulheres). Após inicialização as ligações da estrutura são transpostas para representação dot, utilizando ”ID – ID” para arestas, ”ID -> ID” para arcos simples e ”ID -> ID [COR]” para arcos com alguma propriedade, definida pela cor.

6.2 ANÉIS

Para escrita dos anéis em um arquivo no formato de planilhas um arquivo é aberto e todos os anéis encontrados no grafo são enumerados, escrevendo para cada anel uma lista de atributos. Os atributos descritos são:

- Um indice para o anel (Anel);

- Os números dos vértices que formam os casamentos do anel, com o sexo e sua cor (Ego, SxEgo, CorEgo, Alter, SxAlter, CorAlter, Ego1, ...);
- O Percurso realizado pelo anel, escrito de três formas: utilizando apenas os números dos vértices, utilizando os números, sexo e destacando a junção e utilizando os números e as cores de cada vértice (Percurso, BarryPercurso e PercursoColorido);
- As relações entre os elementos do anel, segundo a ordem descrita nos percursos (Parente, Parente1, ...);
- Os casamentos presentes no anel (Casal, Casal1, ...);
- A diferença de gerações entre os elementos da junção (Geração, Geração1, ...);
- O maior número de conexões realizadas em um dos lados da junção (Lateral, Lateral1, ...)
- O número de conexões realizadas na junção (Cnx, Cnx1, ...);
- O número de cores do anel (Número de Cores);
- As cores presentes no anel (Cores).

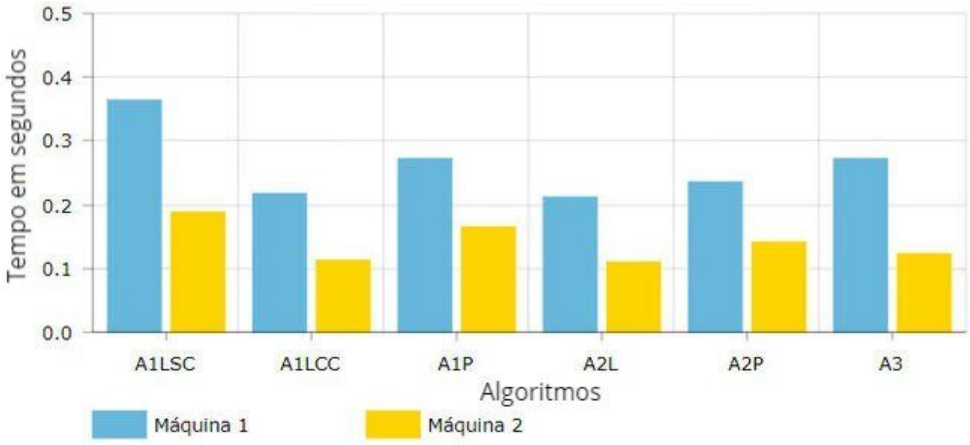
Uma entrada para o arquivo para os anéis A1C1 com coloração paterna contém, em ordem, as informações: Anel, Ego, SxEgo, CorEgo, Alter, SxAlter, CorAlter, Percurso, BarryPercurso,PercursoColorido, Parente, Casal, Geração, Lateral, Cnx, Numero de Cores, Cores, sendo que um exemplo de entrada desta tabela é: 3, 1008, m, 1001 , 1011, f, 1001 , 1008 1002 1011 , m1008 (f1002) f1011 , 1008-1001 (1002- -1) 1011-1001 , MD, 1008-0 1011-0, G0, 1, 2, 2, (1001) (-1).

6.3 TEMPOS DE EXECUÇÃO DOS ALGORITMOS

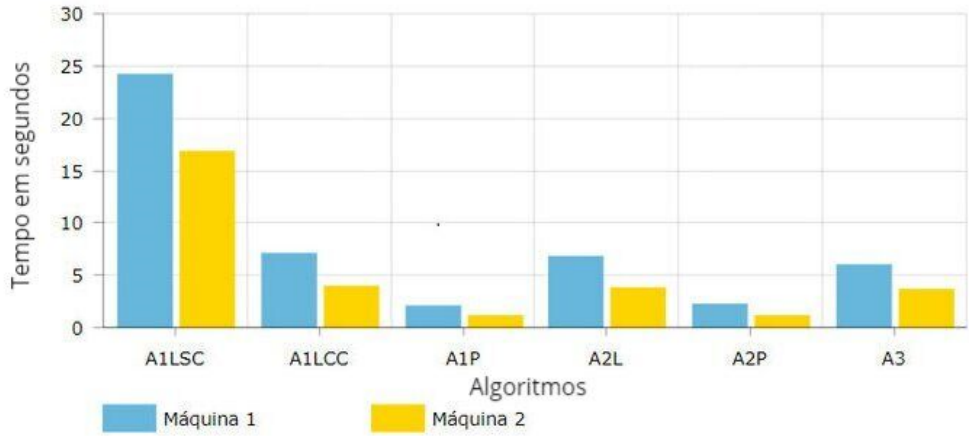
Os algoritmos para encontrar anéis foram testados em duas máquinas denominadas *Máquina 1* (com 10 unidades de processamento de 2.4GHz, 9,74GB de memória RAM e Ubuntu 18.04.1), e *Máquina 2* (com 20 unidades de processamento de 2.4GHz, 128GB de memória RAM e Ubuntu 16.04.7). Para os algoritmos paralelos o número de threads foi especificado para o número de processadores disponíveis para a máquina, enquanto o número de arranjamentos foi fixado em 40. O número 40 foi definido arbitrariamente para possibilitar o paralelismo tanto para as 104 combinações presentes nos anéis A1C1 quanto para as 10712 presentes em anéis A2C2 da rede Arara. Os resultados apresentados na Figura 11 foram os tempos médios de execução em segundos obtidos pela busca de anéis utilizando a rede Arara realizados cinco vezes.

Podemos ver que a *Máquina 2* foi em média mais rápida que a *Máquina 1*, chegando a executar na metade do tempo para o algoritmo *A1LCC* e algoritmo *A2L*

para todos os casos. A execução da busca pelos anéis A2C2 também foi executada na metade do tempo para os algoritmos *A1P* e *A2P*. A melhora no algoritmo *A1LCC* em relação ao algoritmo *A1LSC* obtida através da utilização dos valores min e max pode ser notada em ambas as máquinas nas duas buscas, chegando a dividir o tempo por mais de três unidades na busca pelos anéis com duas afinidades. A paralelização dos algoritmos apresentou ganho somente na busca por anéis A2C2, terminando com tempo maior na busca pelos anéis A1C1. As buscas paralelas nos anéis A2C2 obtiveram ganho temporal, porém não foi um ganho proporcional ao número de unidades de processamento utilizadas. Por fim, os algoritmos que obtiveram melhores resultados para ambas as máquinas foram o algoritmo *A2L* para os anéis A1C1 e o algoritmo *A1P* para os anéis A2C2.



(a) Tempo da execução dos anéis A1C1



(b) Tempo da execução dos anéis A2C2

Figura 11 – Tempo de execução dos algoritmos

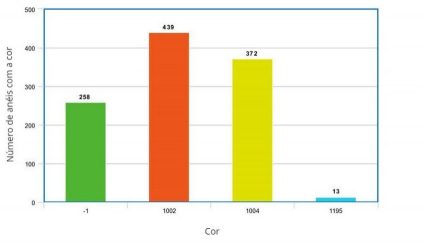
6.4 RESULTADOS QUANTITATIVOS SOBRE OS ANÉIS COLORIDOS

Nesta secção serão apresentados alguns resultados estatísticos sobre os anéis cromáticos A1C1 e A2C2 encontrados na rede Arara. Para obtenção destes números

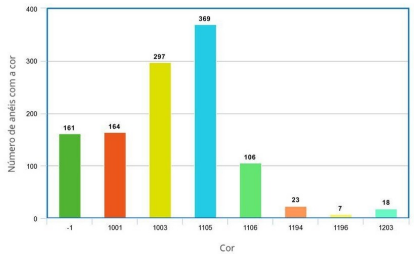
foi utilizada a forma de coloração sugerida pelo antropólogo Prof. Dr. Marnio Teixeira-Pinto onde os vértices passam as cores para os descendentes de um sexo específico primeiro para os homens depois para as mulheres. Cada cor é representada por um número, o número que representa o vértice fonte que iniciou a coloração. Para os vértices que não possuíram cores após execução do algoritmo que colore os vértices (vértices fontes ou intermediários sem descendentes do sexo da propriedade) o número -1 foi atribuído como sua cor.

Na procura por anéis cromáticos foram encontrados 520 anéis A1C1 e 110590 anéis A2C2 na rede Arara. As cores que aparecem para a coloração materna são representadas pelos números 1002, 1004, 1195, 1201, 1204 e -1, com as cores 1001, 1003, 1105, 1106, 1194, 1196, 1197, 1203 e -1 aparecendo para a coloração paterna. Os números -1 aparecem em vértices os quais não são coloridos pelo modo de coloração analisado.

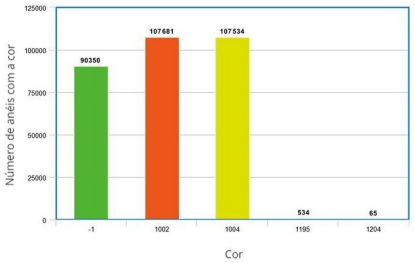
Os números de aparições das cores nos anéis estão presentes na Figura 12. Podemos notar que para ambos os modos de coloração duas cores se destacaram, 1002 e 1004 para materna e 1003 e 1105 para paterna. Vemos também que algumas cores não aparecem em nenhum dos anéis, como as cores 1201 e 1204 para os anéis A1C1 maternos, 1201 para anéis A2C2 maternos e 1197 para anéis paternos. Vale notar que a taxa de aparições dos vértices não coloridos diminuiu para os anéis paternos em relação aos maternos, variando de 49.61% para 30.96% em anéis A1C1 e de 81.70% para 61.54% em anéis A2C2.



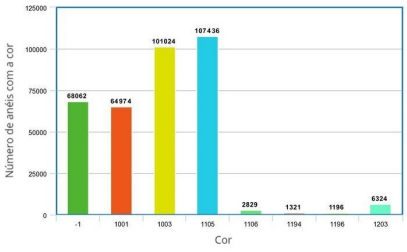
(a) Cores maternas em anéis A1C1



(b) Cores paternas em anéis A1C1



(c) Cores maternas em anéis A2C2



(d) Cores paternas em anéis A2C2

Figura 12 – Número de aparições das cores nos anéis

Pela Figura 13 podemos ver que os anéis A1C1 coloridos de forma materna formam anéis com no maximo três cores distintas. Os anéis são relativamente bem

distribuídos, com a maior taxa de 46.2% atribuída aos anéis com duas cores. Pela Figura 14 vemos que uma das cores com mais aparições, 1002, aparece em todos os anéis com três cores, 75% dos anéis com duas cores e 69% dos anéis com uma cor, totalizando uma taxa de aparição entre os 520 anéis de 84,42%. A cor 1002 também está presente em todos os anéis que contém a cor 1195, de baixa taxa de manifestação (2.5%).

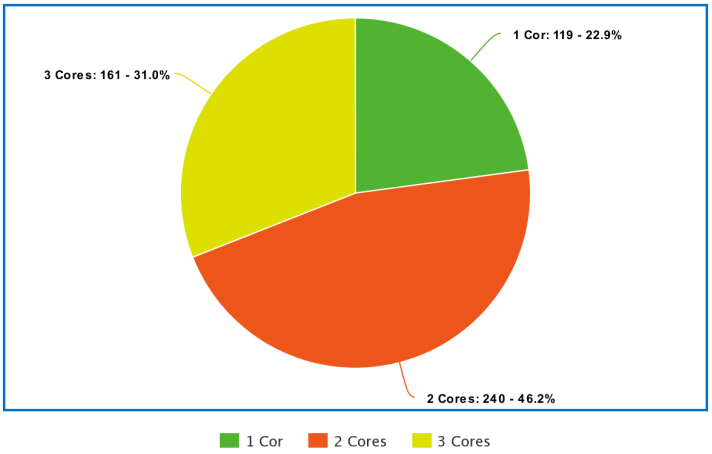
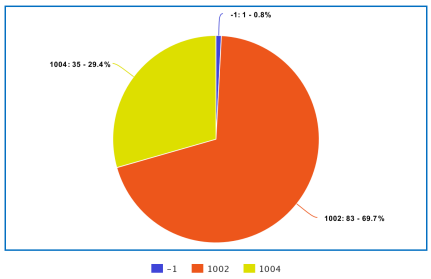
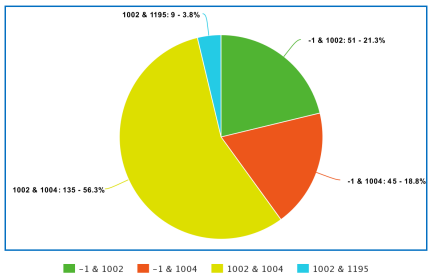


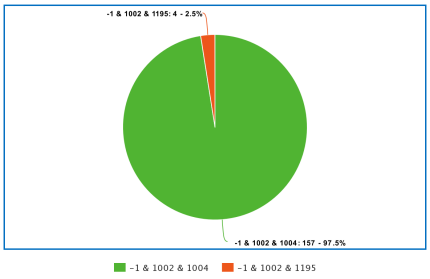
Figura 13 – Distribuição dos anéis A1C1 coloridos por propriedade materna



(a) Distribuição dos 119(22.89%) anéis com uma cor



(b) Distribuição dos 240(46.15%) anéis com duas cores



(c) Distribuição dos 161(30.96%) anéis com três cores

Figura 14 – Distribuição dos grupos de cores dos anéis A1C1 coloridos por propriedade materna

Pela Figura 15 vemos que os anéis A1C1 com coloração paterna formam anéis

com até 4 cores distintas e ao contrário dos anéis A1C1 maternos, sua distribuição de cores apresenta maior diferença, com os anéis com 4 cores aparecendo em apenas 3.1% dos anéis, enquanto os anéis com duas cores aparecem em 42.5% das ocasiões. Através da Figura 16 vemos que em todos os anéis com três cores, algum dos vértices entre 1001 e 1105 aparecem. Como acontece nos A1C1 maternos com as cores 1002 e 1195, nesta coloração também existem cores que aparecem somente em conjunto, porém todos eles aparecem com baixa taxa, caso contrário do vértice 1002 que está em 84,42% dos anéis. As cores que aparecem em conjunto são 1106, 1194 e 1196, onde as cores 1194 (4.42% taxa de manifestações) e 1196 (1.35% taxa de manifestações) sempre aparecem em conjunto com a cor 1106 (20.38% taxa de manifestações).

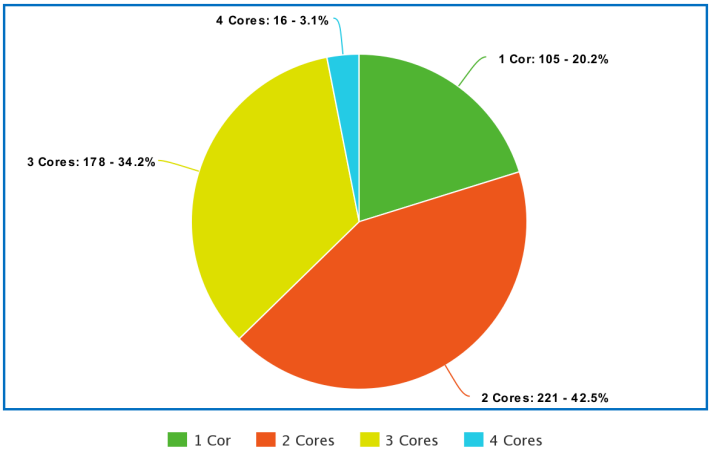
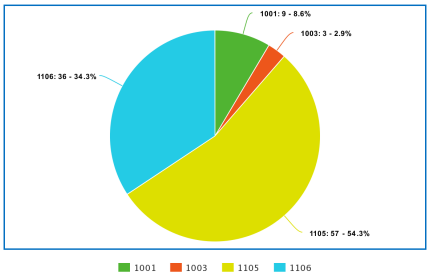
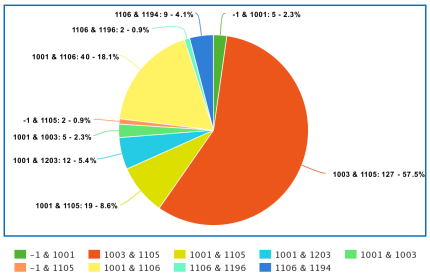


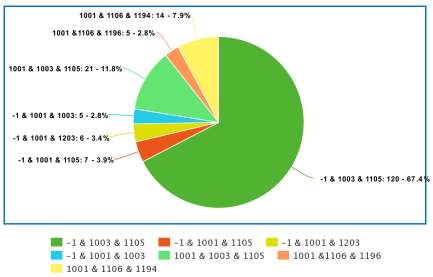
Figura 15 – Distribuição dos anéis A1C1 coloridos por propriedade paterna



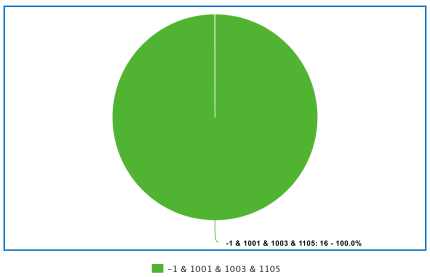
(a) Distribuição dos 105 (20.19%) anéis com uma cor



(b) Distribuição dos 221(42.5%) anéis com duas cores



(c) Distribuição dos 178(34.23%) anéis com três cores



(d) Distribuição dos 16(3.08%) anéis com quatro cores

Figura 16 – Distribuição dos grupos de cores dos anéis A1C1 coloridos por propriedade paterna

Para os anéis A2C2 com coloração materna notamos pela Figura 12c que as cores 1002 e 1004 aparecem respectivamente em 97.37% e 97.24% dos anéis encontrados, em contraste com as outras cores que representam vértices fontes femininos 1195, 1201 e 1204, que aparecem em respectivamente 0.48%, 0.0% e 0.06% dos anéis. Pelas figuras 17 e 18 percebe-se uma alta porcentagem de anéis com três cores, que aparecem em 77.85% dos casos, com anéis formados pelo trio 1002, 1004 e -1 representando 99.5% destes casos. Pode-se notar que do mesmo modo que acontece nos anéis A1C1 maternos, sempre que as cores com poucas aparições ocorrem, o vértice 1002 está com elas. Outro ponto interessante é que todos os anéis possuem a cor 1002 ou a cor 1004, ou seja, nenhum dos anéis é formado sem a presença de alguma destas duas cores.

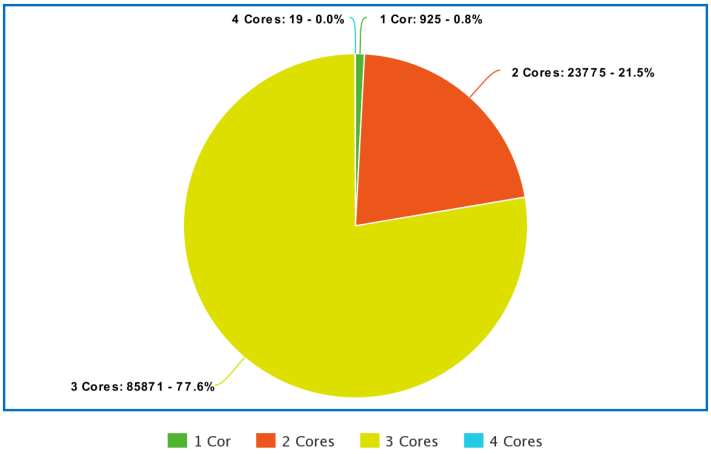
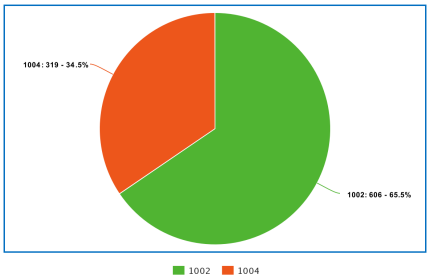
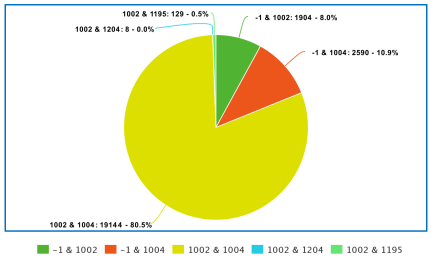


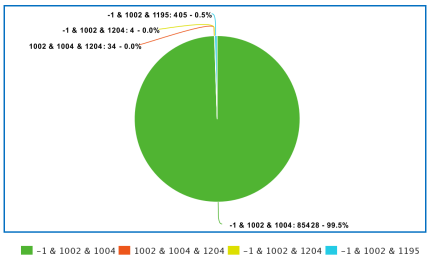
Figura 17 – Distribuição dos anéis A2C2 coloridos por propriedade materna



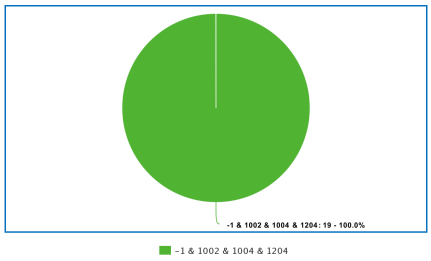
(a) Distribuição dos 925(0.84%) anéis com uma cor



(b) Distribuição dos 23775(21.5%) anéis com duas cores



(c) Distribuição dos 85871(77.65%) anéis com três cores



(d) Distribuição dos 19(0.02%) anéis com quatro cores

Figura 18 – Distribuição dos grupos de cores dos anéis A2C2 coloridos por propriedade materna

Para os anéis A2C2 obtidos pela coloração paterna podemos observar pelas figuras 12d e 20 que existem mais cores, e que embora elas também não sejam distribuídas igualmente quanto sua aparição em anéis (97.15% para 1105, 91.35% para 1003, 58.75% para 1001, 2.56% para 1106, 1.2% para 1194, 0.31% para 1196, 0.0% para 1197 e 5.72% para 1203), existem anéis em que as cores mais dominantes 1105 e 1003 não aparecem. Nos anéis A2C2 paternos, assim como nos anéis A1C1 paternos, sempre que as cores de baixa taxa de manifestação 1194 (1.2%) e 1196

(0.31%) aparecem, a cor 1106 (2.56%) também aparece. Por fim vemos pela Figura 19 que existem anéis com até 5 cores, com os anéis com 3 cores aparecendo em 45.1% dos casos.

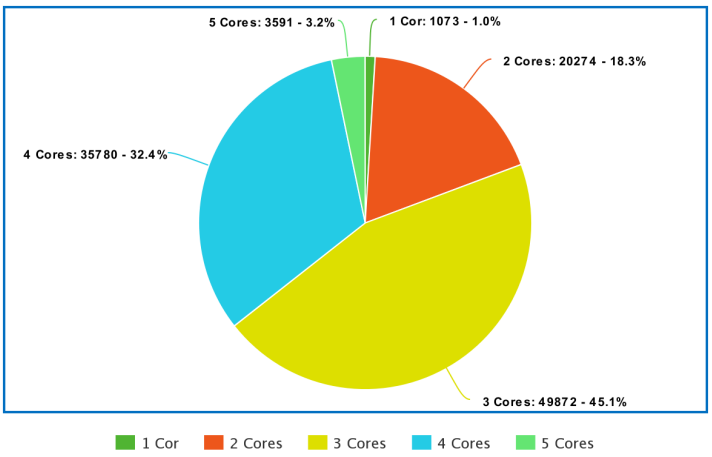
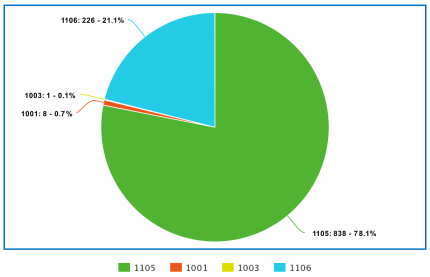
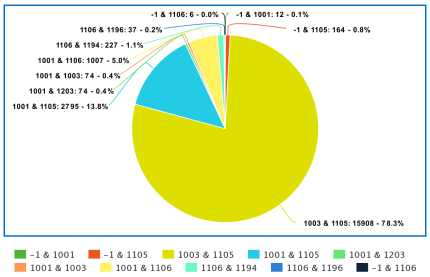


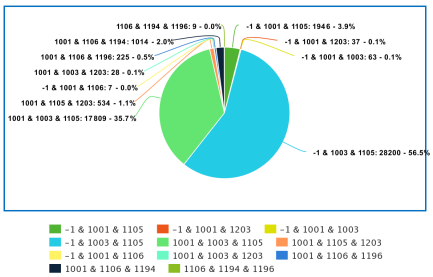
Figura 19 – Distribuição dos anéis A2C2 coloridos por propriedade paterna



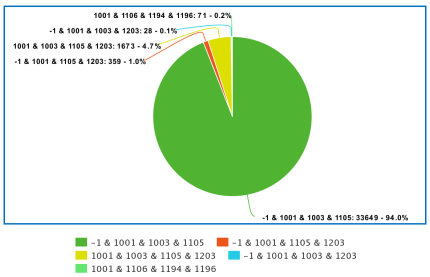
(a) Distribuição dos 1073 (0.97%) anéis com uma cor



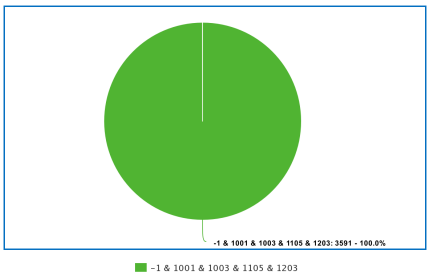
(b) Distribuição dos 20274(18.33%) anéis com duas cores



(c) Distribuição dos 49872(45.1%) anéis com três cores



(d) Distribuição dos 35780(32.35%) anéis com quatro cores



(e) Distribuição dos 3591(3.25%) anéis com cinco cores

Figura 20 – Distribuição dos grupos de cores dos anéis A2C2 coloridos por propriedade paterna

Pela análise dos gráficos podemos perceber que as duas colorações de anéis A1C1 apresentam maior quantidade de anéis com 2 cores, enquanto as duas colorações A2C2 apresentam mais anéis com 3 cores. Todas as distribuições possuem duas cores que aparecem mais que as outras, 1003 e 1105 para coloração paterna e 1002 e 1004 para coloração materna. Estas duas cores sempre formavam mais que 50% dos anéis que tinham mais de duas cores, sendo que uma delas formava mais que 50% dos anéis com apenas uma cor. A cor que representa um indivíduo do sexo oposto regra de transmissão de cores, -1, apareçe em grande quantidade, porém ela somente aparece nesta grande quantidade em anéis com mais de 2 cores, em conjunto com as duas cores dominantes. Finalmente, a cor -1 aparece somente em 1 anel sozinha, nos anéis A1C1 maternos.

Isso conclui nossa análise quantitativa dos anéis cromáticos da rede Arara. No

capítulo final é apresentado a conclusão do trabalho.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho desenvolveu e implementou algoritmos que realizam a busca por anéis cromáticos em redes de parentesco. Para alcançar este objetivo o trabalho realizou uma implementação para transpor redes de parentesco reais para grafos, um sistema de coloração com dois modos de colorir, baseados em hipóteses antropológicas e algoritmos que percorrem o grafo para obter informações de máximo e mínimo cores entre vértices, utilizadas para melhorar a busca por caminhos no grafo. Uma análise de complexidade de tempo foi realizada sobre os algoritmos desenvolvidos, sendo que um artigo sobre o algoritmo 7 foi apresentado no Encontro de Teoria da Computação 2021 (ETC), um evento satélite do Congresso da Sociedade Brasileira de Computação.

Utilizando estas implementações três modos de busca pelos anéis foram produzidos, sendo que para dois modos foram desenvolvidos versões que executam o processo de forma paralela e para um deles a versão executa somente em paralelo. Por fim a busca pelos anéis cromáticos foi aplicada sobre a rede Arara, com os resultados obtidos utilizados para analisar a rede. As versões que utilizavam a informação das cores tiveram um tempo de execução menor para os anéis A2C2 se comparadas com sua contraparte que não utilizava a informação, porém na busca pelos anéis A1C1 a diferença de tempo não se mostrou significativa. Os algoritmos paralelos implementados possuem desempenho melhor nas buscas de anéis A2C2, porém não obtiveram ganho temporal em relação a busca A1C1, assim como não apresentaram ganho proporcionalao número de unidades de processamento utilizadas.

Foram apresentados levantamentos numéricos sobre as cores presentes nos anéis utilizando a coloração por propriedades feminina e masculina, onde as cores apresentadas pelos números 1002 e 1004 foram dominantes nos anéis maternos, enquanto 1003 e 1105 foram dominantes nos anéis paternos. A maior parte dos anéis A2C2 maternos, 77,65%, apresentou 3 cores, com 99,5% destes anéis contendo o trio 1002, 1004 e -1. Para os anéis A2C2 e A1C1 paternos, nenhuma quantidade alcançou maioria, embora a distribuição não tenha sido igualitária. Os anéis A1C1 maternos apresentaram melhor distribuição, com no máximo os anéis com duas cores representando 46% dos anéis totais.

Para disponibilizar mais informações sobre uma rede e possibilitar melhor análise, o trabalho implementou modos de visualização para as estruturas obtidas, utilizando a ferramenta de descrição de linguagens Graphviz.

Para os trabalhos futuros seguímos com análise sobre os anéis cromáticos encontrados, com uma proposta de artigo sobre sua análise númerica sendo enviada para o evento Rede de Antropologia da Ciência e da Tecnologia (ReACT) em 2021. Outros trabalhos futuros incluem os algoritmos paralelos, com a procura de valores calculados

pela rede para os parâmetros utilizados nas threads, assim como a enumeração dos anéis A3C3 e redes de parentesco diferentes da rede Arara.

REFERÊNCIAS

AGHASSIAN, Michel; AUGÉ, Marc; BESSA, Ana Maria. **Os domínios do parentesco: Filiação, aliança matrimonial, residencia**. [*S.l.*: *s.n.*], 1975.

BATAGELJ, Vladimir; MRVAR, Andrej. Pajek-program for large network analysis. **Connections**, v. 21, n. 2, p. 47–57, 1998.

BATAGELJ, Vladimir; MRVAR, Andrej. **Pajek.** [*S.l.*: *s.n.*], 2014.

CORMEN, Thomas H; LEISERSON, Charles E; RIVEST, Ronald L; STEIN, Clifford. Algoritmos: teoria e prática. **Editora Campus**, v. 2, p. 296, 2002.

FERREIRA, Carlos Eduardo; FRANCO, Álvaro Junio Pereira;
SILVA, Marcio Ferreira da. The kinship as a computational question.

GANSNER, Emden R. Drawing graphs with Graphviz. **Technical Report, Technical Report**, 2009.

HAMBERGER, Klaus; HOUSEMAN, Michael; DAILLANT, Isabelle; WHITE, Douglas R; BARRY, Laurent. Matrimonial ring structures. **Mathématiques et sciences humaines. Mathematics and social sciences**, Centre d'analyse et de mathématique sociales de l'EHESS, n. 168, 2004.

KLEINBERG, Jon; TARDOS, Eva. **Algorithm design**. [*S.l.*]: Pearson Education India, 2006.

NETO, João Dal Poz; SILVA, Marcio Ferreira da. MaqPar. A Homemade Tool for the Study of Kinship Networks. **VIBRANT-Vibrant Virtual Brazilian Anthropology**, Associação Brasileira de Antropologia, v. 6, n. 2, p. 29–51, 2009.

SINNWELL, Jason P; THERNEAU, Terry M; SCHAID, Daniel J. The kinship2 R package for pedigree data. **Human heredity**, Karger Publishers, v. 78, n. 2, p. 91–93, 2014.

TEIXEIRA-PINTO, Márnio. **Povos Indígenas no Pará: Arara**. Abr. 1998. Disponível em: `https://pib.socioambiental.org/pt/Povo:Arara#Fontes_de_informa.C3.A7.C3.A3o`. Acessado: 14/08/2021.

ANEXO A – CÓDIGO

O código desenvolvido está disponível no endereço `codigos.ufsc.br/m.emilio/aneis-cromaticos`. Um arquivo makefile está incluido no projeto contendo as informações para compilação e execução do programa. Para acessar estas informações utilizar o comando ”make info”. Na pasta ”entrada” deve ser colocado o arquivo contendo a rede de parentesco que será utilizada, com os resultados gerados sendo inseridos nas pastas ”aneis”, ”desenhos” e ”outros”, a depender do tipo de execução realizada. As atualizações e o código desenvolvido para trabalhos futuros estão disponíveis no endereço `github.com/mevendra/algoritmos`.

ANEXO B – ARTIGO

Algoritmos para encontrar anéis cromáticos em redes de parentesco

Marcelo E. Vendramin
Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC –Brazil

Abstract. The goal of the work is to enable the verification of hypotheses about people by providing chromatic rings obtained from real networks. To achieve this goal, the work use graphs to represent real kinship. A way of coloring vertices of this graph was proposed based on an anthropological hypothesis. Algorithms that calculate properties derived from colors have also been proposed. Using this properties, the work developed ways of enumerate all the chromatic rings, implementing each one to execute sequentially and parallelaly. An analysis of the execution time of the implemented searches was carried out on the Arara network, a real network. An analysis of numeric results was also performed on the 520 rings with one affinity and 110590 rings with two affinities found on the network.

Resumo. O trabalho tem como objetivo possibilitar a verificação de hipóteses sobre povos através da disponibilização de anéis cromáticos derivados de redes reais. Para isto, o trabalho define a utilização de grafos para representação de redes de parentesco reais. Um modo de colorir vértices deste grafo foi proposto com base em uma hipótese antropológica. Também foram propostos algoritmos que calculam propriedades derivadas das cores. Utilizando as propriedades obtidas, o trabalho desenvolveu formas de enumerar anéis cromáticos, implementando cada uma delas para execuções sequenciais e paralelas. Uma análise sobre o tempo de execução das buscas implementadas foi realizada sobre a rede real Arara. Uma análise numérica dos resultados também foi realizada sobre os 520 anéis com uma afinidade e 110590 anéis com duas afinidades encontrados para a rede.

Introdução

Uma rede de parentesco representa indivíduos de um povo e suas relações internas, como casamentos e filiações. Uma rede de parentesco é um grafo misto, onde os arcos representam relações de filiação e arestas representam os casamentos. Dentro desta rede, existe uma estrutura denominada de anéis, que é formada pelas relações de parentesco e casamentos e é arranjada para formar um ciclo.

Existe uma área na Antropologia que se interessa em analisar os anéis obtidos em redes de parentesco, porém eles podem existir em grande quantidade, o que gera um desafio sobre sua análise e enumeração. Algumas propriedades de redes de parentesco podem ser modeladas utilizando cores nos vértices, arestas e arcos. Estas cores podem representar diversas propriedades e dois modos de coloração, que procuram solucionar uma propriedade específica da rede Arara, uma rede real, foram propostos.

O trabalho procura resolver questões relacionadas a enumeração de anéis cromáticos, com o objetivo de decifrar aspectos da estrutura social de um povo através dos casamentos entre os indivíduos. Para atingir este objetivo, o trabalho também desenvolve e implementa algoritmos que enumeram anéis cromáticos. Os algoritmos foram aplicados sobre a rede Arara e um levantamento quantitativo sobre os anéis cromáticos encontrados foi realizado. Este trabalho contou com o auxílio de uma bolsa PIBIC do CNPq e um dos algoritmos desenvolvidos foi apresentado no Encontro de Teoria da Computação (ETC) em 2021.

Desenvolvimento do Trabalho

O trabalho iniciou com uma introdução sobre os conceitos sobre a área da antropologia relacionados com o trabalho (AGHASSIAN; AUGÉ; BESSA, 1975) e (HAMBERGER *et al.*, 2004), seguindo para uma informação sobre algumas soluções existentes para problemas antropológicos.

Com os conceitos iniciais desenvolvidos, o trabalho apresentou algoritmos que serviram de base para os algoritmos relacionados a busca de anéis cromáticos, alguns dos algoritmos descritos em (KLEINBERG; TARDOS, 2006) e (CORMEN *et al.*, 2002).

A busca pelos anéis foi primeiro apresentada sem utilizar a informação das cores, com a descrição sobre a implementação do algoritmo descrito em (FERREIRA; FRANCO; SILVA, s.d.). A coloração dos vértices foi introduzida nas redes de parentesco para se adequar a uma propriedade relativa a rede Arara, mas os algoritmos desenvolvidos não dependem do modo de coloração.

Utilizando os vértices coloridos foram propostos algoritmos que calculam as propriedades de valores máximo e mínimo de cores de um vértice para todos os outros, assim como o calculo do valor super (maior número de cores em caminhos) para todos os vértices.

Utilizando as propriedades obtidas pelas cores, três algoritmos que encontram anéis cromáticos foram apresentados. Para estes algoritmos também foram desenvolvidas versões que realizam sua execução de forma paralela. Estes algoritmos foram adaptados para diferentes maneiras de execução, resultando nos algoritmos denominados de *Algoritmo 1 Não Colorido* (A1LSC), *Algoritmo 1 Colorido Sequencial* (A1LCC), *Algoritmo 1 Colorido Paralelo* (A1P), *Algoritmo 2 Sequencial* (A2L), *Algoritmo 2 Paralelo* (A2P) e *Algoritmo 3* (A3).

Análise dos resultados e Trabalhos Futuros

Os tempos de execução em segundos para os algoritmos podem ser vistos na Figura B.0.1 para anéis com uma afinidade e um casamento, e na Figura B.0.2 para os anéis com duas afinidades e dois casamentos. Os tempos foram calculados pela média de 5 execuções, realizadas nas máquinas 1 e 2. A máquina 1 tem 10 unidades de processamento de 2.4GHz, 9,74GB dememória RAM e Ubuntu 18.04.1. A máquina 2 tem 20 unidades de processamentode 2.4GHz, 128GB de memória RAM e Ubuntu 16.04.7.

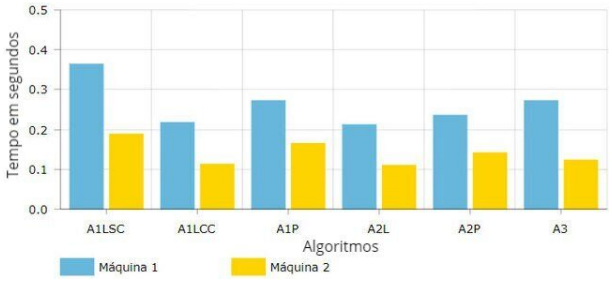


Figura B.0.1 – Busca por anéis com uma afinidade e um casamento

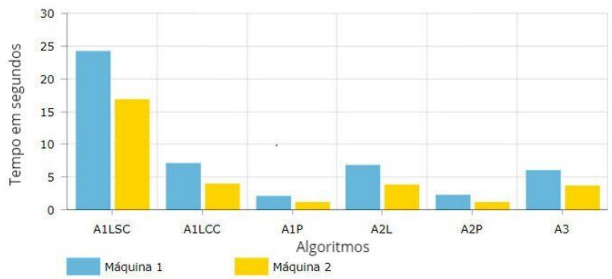


Figura B.0.2 – Busca por anéis com duas afinidades e dois casamentos

Para a rede Arara foram encontrados 520 anéis com uma afinidade e um casamento e 110590 anéis com duas afinidades e dois casamentos. O trabalho realizou uma análise sobre estes anéis utilizando uma das formas de coloração propostas, apresentando alguns resultados númericos.

Os trabalhos futuros seguem com uma análise mais profunda sobre os anéis cromáticos encontrados, melhoras nos algoritmos paralelos, análises de anéis com três afinidades e três casamentos e por fim trabalhos sobre redes de parentesco diferentes da rede Arara.

REFERÊNCIAS

AGHASSIAN, Michel; AUGÉ, Marc; BESSA, Ana Maria. **Os domínios do parentesco: Filiação, aliança matrimonial, residencia**. [*S.l.*: *s.n.*], 1975.

CORMEN, Thomas H; LEISERSON, Charles E; RIVEST, Ronald L; STEIN, Clifford. Algoritmos: teoria e prática. **Editora Campus**, v. 2, p. 296, 2002.

FERREIRA, Carlos Eduardo; FRANCO, Álvaro Junio Pereira;
SILVA, Marcio Ferreira da. The kinship as a computational question.

HAMBERGER, Klaus; HOUSEMAN, Michael; DAILLANT, Isabelle; WHITE, Douglas R; BARRY, Laurent. Matrimonial ring structures. **Mathématiques et sciences humaines. Mathematics and social sciences**, Centre d'analyse et de mathématique sociales de l'EHESS, n. 168, 2004.

KLEINBERG, Jon; TARDOS, Eva. **Algorithm design**. [*S.l.*]: Pearson Education India, 2006.