UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO GLEYSON DA SILVA GOMES

ALGORITMOS DE COLORAÇÃO DE GRAFOS PARA O ESCALONAMENTO EM REDES SEM FIO

FORTALEZA – CEARÁ

2018

FRANCISCO GLEYSON DA SILVA GOMES

ALGORITMOS DE COLORAÇÃO DE GRAFOS PARA O ESCALONAMENTO EM REDES SEM FIO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Leonardo Sampaio Rocha

Co-Orientador: Prof. Dr. Gerardo Valdísio Rodrigues Viana

FORTALEZA – CEARÁ

2018

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Sistema de Bibliotecas

Gomes, Francisco Gleyson da Silva.
Algoritmos de Coloração de Grafos para o Escalonamento em Redes sem Fio [recurso eletrônico] / Francisco Gleyson da Silva Gomes. - 2018.
1 CD-ROM: il.; 4 ¾ pol.

CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico com 84 folhas, acondicionado em caixa de DVD Slim (19 x 14 cm x 7 mm).

Dissertação (mestrado acadêmico) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Mestrado Acadêmico em Ciência da Computação, Fortaleza, 2018.
Área de concentração: Otimização e Matemática Aplicada.
Orientação: Prof. Dr. Leonardo Sampaio Rocha.
Coorientação: Prof. Dr. Gerardo Valdísio Rodrigues Viana.

1. Teoria dos Grafos. 2. Modelo de Coloração BPRN. 3. Algoritmos de Coloração BPRN. 4. Redes Sem Fio. 5. Intervalos de Tempo. I. Título.

FRANCISCO GLEYSON DA SILVA GOMES

ALGORITMOS DE COLORAÇÃO DE GRAFOS PARA O ESCALONAMENTO EM REDES SEM FIO

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Aprovada em: 21 de Junho de 2018

BANCA EXAMINADORA

Prof. Dr. Leonardo Sampaio Rocha
(Orientador/UECE)

Prof. Dr. Gerardo Valdísio Rodrigues Viana
(Coorientador/UECE)

Prof. Dr. Fábio Carlos Sousa Dias
(UFC)

Profa. Dra. Ana Shirley Ferreira da Silva
(UFC)

Dedico este trabalho aos meus pais, Eginaldo e Ivanilda, ao meu irmão Gleiton e ao professor Davi.

AGRADECIMENTOS

A Deus pelo dom da vida, pelas graças alcançadas, e por toda a força recebida em todos os momentos e, principalmente, nos momentos de dificuldades. Ainda quero agradecer por todas as oportunidades concedidas para o meu crescimento.

Aos meus pais, por acreditarem em mim, e estarem do meu lado nas diferentes etapas da vida. Agradeço também por todos os ensinamentos, que são fundamentais para mim. Ao meu irmão, pelo seu respeito, confiança, e por saber que posso contar com você. Pai, mãe, e irmão, amo vocês!

Ao professor Dr. Davi Romero de Vasconcelos por todo o apoio recebido, em especial, quando comecei os estudos para o vestibular. Por todos os conselhos e palavras de incentivo. A confiança do senhor no meu trabalho me motiva a superar os desafios. O trabalho do senhor transformou a minha vida. E tenho certeza que vai transformar a realidade de outros estudantes. Gosto muito do senhor!

Ao meu orientador professor Dr. Leonardo Sampaio Rocha por todo o conhecimento compartilhado, e pelas reuniões, que foram importantes para a melhoria do nosso trabalho. Gostaria de agradecer o senhor pela confiança, conselhos, e por toda a dedicação em me orientar.

Ao professor Dr. Gerardo Valdísio Rodrigues Viana pela coorientação.

Aos professores Dr. Fábio Carlos Sousa Dias e Dra. Ana Shirley Ferreira da Silva por participarem da minha banca examinadora.

A todos os amigos pelas contribuições e amizade.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro para a realização deste mestrado.

“You can always give it more than what you think you have.”

(Mo Farah)

RESUMO

Uma rede sem fio consiste em um conjunto de nós que compartilham um canal de difusão para a transmissão e recepção de informações. Em um canal compartilhado, as transmissões estão sujeitas à colisões por causa dos problemas de interferências. Uma forma de resolver esses problemas é através de um escalonamento usando a técnica TDMA, que atribui aos nós ou enlaces da rede intervalos de tempo de um quadro TDMA. Uma rede pode ser modelada como um digrafo e o escalonamento TDMA de enlaces ser encontrado com o modelo de coloração PRN. O modelo de coloração PRN é usado na coloração dos arcos do digrafo da rede, para evitar as interferências primárias e secundárias. Este trabalho propõe algoritmos e uma modelagem para o problema de coloração BPRN. O modelo de coloração BPRN é uma extensão do modelo de coloração PRN, pois considera apenas à coloração de um subconjunto de arcos do digrafo da rede, denominado de digrafo *backbone*. Um digrafo *backbone* pode representar diferentes topologias de redes, mas neste trabalho, ele é representado por uma árvore geradora orientada em direção a um vértice raiz. Os algoritmos propostos, denominados de Bsatur e Bswap, adaptam os conceitos de grau e grau de saturação de um vértice usados no algoritmo Dsatur, para grau de conflitos e grau de saturação de um arco. As informações de grau de conflitos e grau de saturação de um arco são usadas por esses algoritmos para determinar à ordem de coloração dos arcos de um digrafo *backbone*. O algoritmo Bswap incorpora um mecanismo de reutilização de cores a ser utilizado quando é preciso criar uma nova cor para colorir um arco. Os algoritmos foram avaliados em cenários com instâncias de baixa, média e alta densidade. Os resultados mostraram que em instâncias com maior densidade, os algoritmos propostos obtiveram resultados melhores em relação aos resultados dos algoritmos BF e DFOr usados para comparação. O tempo de execução do algoritmo Bswap é maior do que o tempo de execução do algoritmo Bsatur devido ao mecanismo de reutilização de cores adotado por esse algoritmo. Os algoritmos BF e DFOr apresentaram o mesmo tempo de execução, pois a principal diferença entre esses algoritmos está na ordem em que os vértices do digrafo *backbone* são visitados.

Palavras-chave: Teoria dos Grafos. Modelo de Coloração BPRN. Algoritmos de Coloração BPRN. Redes Sem Fio. Intervalos de Tempo.

ABSTRACT

A wireless network consists of a set of nodes that share a broadcast channel for transmitting and receiving information. In a shared channel, transmissions are subject to collisions because of interferences problems. One way to solve these problems is through a scheduling using the TDMA technique, which assigns nodes or network links time slots of a TDMA frame. A network can be modeled as a digraph and TDMA links scheduling can be found with the PRN coloring model. The PRN coloring model is used to color the digraph arcs of the network to avoid primary and secondary interferences. This work proposes algorithms and a modeling for the BPRN coloring problem. The BPRN coloring model is an extension of the PRN coloring model, because it considers only the coloring of a subset of arcs of the network digraph, called the *backbone* digraph. A *backbone* digraph can represent different network topologies, but in this work, it is represented by a generating tree oriented towards a root vertex. The proposed algorithms, named Bsatur and Bswap, adapt the concepts of degree and degree of saturation of a vertex used in the Dsatur algorithm, for degree of conflicts and degree of saturation of an arc. The information of degree of conflict and degree of saturation of an arc are used by these algorithms to determine the order of coloring of the arcs of a *backbone* digraph. The Bswap algorithm incorporates a color reuse mechanism to be used when is need to create a new color to color an arc. The algorithms were evaluated in scenarios with low, medium and high density instances. The results showed that in higher density instances, the proposed algorithms obtained better results in relation to the results of the BF and DFOr algorithms used for comparison. The execution time of the Bswap algorithm is greater than the execution time of the Bsatur algorithm due to the color reuse mechanism adopted by this algorithm. The algorithms BF and DFOr presented the same execution time, since the main difference between these algorithms is in the order in which the vertices of the *backbone* digraph are visited.

Keywords: Graph Theory. BPRN Coloring Model. BPRN Coloring Algorithms. Wireless Networks. Time Intervals.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

BF	Breadth First Algorithm
BPRN	Backbone Packet Radio Network
CDMA	Code Division Multiple Access
DFOr	Depth First Ordering Algorithm
Dsatur	Degree of Saturation
ECDiG	Edge Coloring on Directed Graphs
FDMA	Frequency Division Multiple Access
MAC	Medium Access Control
PRN	Packet Radio Network
TDMA	Time Division Multiple Access

SUMÁRIO

1 INTRODUÇÃO

As redes sem fio são formadas por um conjunto de nós transceptores, isto é, capazes de transmissão e recepção de informações, onde cada nó possui um raio de transmissão para se comunicar com outros nós através do uso de rádio de transmissão. O conjunto de nós vizinhos de um nó, correspondem aos nós pertencentes ao seu raio de transmissão. Assim, um nó v é vizinho de um nó u, se v estiver no raio de transmissão de u. A maioria dessas redes são de múltiplos saltos, ou seja, existem nós entre um nó origem e um nó destino, com capacidade de roteamento. O uso de múltiplos saltos é motivado pelos custos de transmissão decorrentes da distância e o consumo de recursos para a comunicação entre um nó remetente e um nó destinatário. Nessas redes, as comunicações acontecem através de um canal de difusão compartilhado. Assim, as transmissões estão sujeitas a colisões, por causa de interferências primárias ou interferências secundárias (ARIKAN, 1984; HUSON, 1995; DJUKIC, 2008).

Uma interferência primária ocorre quando um nó realiza tarefas simultâneas de transmissão ou recepção. Já uma interferência secundária ocorre quando um par de nós u e a compartilham um nó vizinho v e transmitem simultaneamente, onde u transmite para v e a transmite para um nó b. Neste caso, as transmissões de u e a colidem em v, pois v está no alcance de transmissão de a. O escalonamento em redes sem fio possibilita a transmissão e recepção de informações sem a ocorrência de interferências primárias e secundárias.

Uma rede pode ser representada por um digrafo $D = (V(D), A(D))$, onde $V(D)$ é o conjunto de vértices denotando os nós, e $A(D)$ é o conjunto de arcos entre pares de vértices $u, v \in V(D)$, tal que, $uv \in A(D)$, se o nó v estiver no alcance de transmissão do nó u. Uma forma de interpretar o escalonamento é observando a equivalência com a coloração de D. Assim, o escalonamento de enlaces pode ser visto como a coloração dos arcos de D, usando o modelo de coloração PRN (do inglês, Packet Radio Network). Uma coloração PRN de D pode ser descrita como uma função $c : A(D) \to \mathbb{N}$, tal que, se $c(uv) = c(ab)$, então:

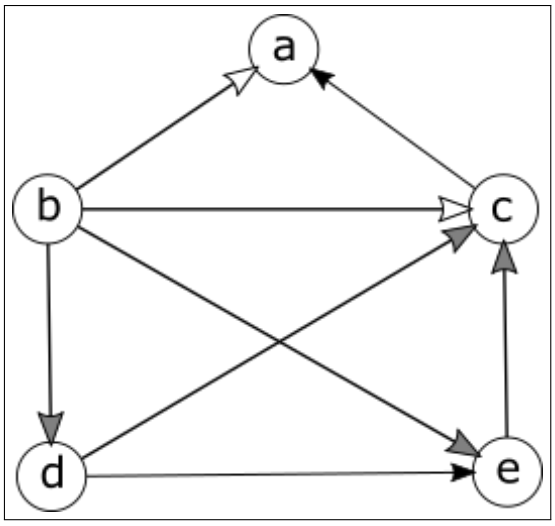
a) a, b, u e v são vértices mutuamente distintos; e

b) $av \notin A(D)$ e $ub \notin A(D)$.

As restrições em (a) evitam as interferências primárias e as restrições em (b) evitam as interferências secundárias. Um arco causa conflito primário com outro arco quando não respeita as restrições em (a). De forma semelhante, um arco causa conflito secundário com outro arco quando não satisfaz as condições em (b). O índice PRN-cromático de D representa o número mínimo de cores usadas em uma coloração PRN de seus arcos. A Figura 1 ilustra

uma rede modelada como um digrafo, destacando os arcos que causam conflitos primários e secundários em relação ao arco *de*, onde os arcos com a seta colorida de cinza causam conflitos primários e os arcos com a seta colorida de branco causam conflitos secundários. No entanto, o arco *ca* com a seta de cor preta não causa conflito com o arco de.

Figura 1 – Exemplo de conflitos primários e secundários de um arco.



Fonte – Elaborado pelo autor

Este trabalho investiga uma extensão do modelo de coloração PRN, denominada de Backbone PRN (BPRN), para o escalonamento TDMA de enlaces em redes sem fio (ARIKAN, 1984; ROCHA; SASAKI, 2017). A diferença entre esses modelos é que o modelo de coloração BPRN considera apenas um subconjunto de arcos a serem coloridos, sendo que esse subconjunto de arcos é denominado de digrafo *backbone*. Os outros arcos do digrafo são usados com o intuito de evitar os conflitos secundários ao colorir os arcos do digrafo *backbone*. Já no modelo de coloração PRN todos os arcos do digrafo são coloridos. Esse modelo foi abordado em alguns trabalhos para o escalonamento TDMA de enlaces (ARIKAN, 1984; RAMANATHAN; LLOYD, 1993; HUSON, 1995).

Seja $D = (V(D), A(D))$ um digrafo, dizemos que $B = (V(B), A(B))$ é um digrafo *backbone* de D, se $V(B) = V(D)$ e $A(B) \subseteq A(D)$. Uma coloração BPRN de (D, B) é uma coloração de arcos $c : A(B) \to \mathbb{N}$, tal que, se $c(uv) = c(ab)$, então:

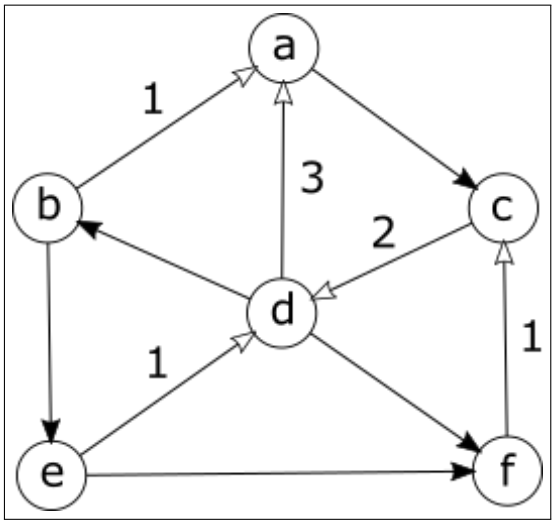
a) a, b, u e v são vértices mutuamente distintos; e

b) $av \notin A(D)$ e $ub \notin A(D)$.

Um par de arcos no digrafo *backbone* terão a mesma cor se as restrições do modelo forem satisfeitas. O índice BPRN-cromático de D e B representa o número mínimo de cores

usadas em uma coloração BPRN dos arcos de (D, B). A Figura 2 ilustra o número mínimo de cores usadas em uma coloração BPRN, onde os arcos do digrafo *backbone* estão com a seta colorida de branco.

Figura 2 – Número mínimo de cores usadas em uma coloração BPRN.



Fonte – Elaborado pelo autor

O modelo de coloração BPRN procura capturar as características principais de topologias no domínio das redes sem fio. Esse modelo é mais realista, pois considera que na maioria dessas redes, um nó precisa se comunicar apenas com alguns de seus vizinhos. Desse modo, as propriedades intrínsecas do digrafo *backbone* podem se adaptar às condições da aplicação desenvolvida. Além disso, permite investigar o uso de topologias conhecidas no intuito de capturar os aspectos relevantes e explorar a complexidade do problema de escalonamento TDMA de enlaces com mais precisão (ROCHA; SASAKI, 2017).

Neste trabalho, são propostos dois algoritmos baseados no algoritmo Dsatur (BRÉLAZ, 1979) para a coloração BPRN. Os conceitos de grau e grau de saturação de um vértice, vistos no algoritmo Dsatur, são adaptados nos algoritmos propostos para grau de conflitos e grau de saturação de um arco. Assim, esses algoritmos denominados de Bsatur e Bswap, colorem os arcos do digrafo *backbone* seguindo estratégias semelhantes com as estratégias adotadas pelo algoritmo Dsatur durante a coloração dos vértices de um grafo. O algoritmo Bswap representa uma extensão do algoritmo Bsatur, pois durante a coloração dos arcos do digrafo *backbone* é verificado a possibilidade de reutilizar uma cor, quando é preciso criar uma nova cor para colorir um arco. Para os experimentos computacionais, foram utilizadas instâncias com pouca, média e alta densidade. Essas instâncias foram criadas seguindo certas configurações, sendo inspiradas

em topologias de redes sem fio.

1.1 MOTIVAÇÃO

A coloração BPRN está diretamente relacionada com o escalonamento TDMA de enlaces, pois uma rede sem fio pode ser modelada como um digrafo. Dessa forma, a coloração BPRN encontrada pode ser interpretada como um escalonamento TDMA de enlaces, onde as cores dos arcos representam os intervalos de tempo no quadro TDMA. Os algoritmos propostos podem ser usados em protocolos MAC, com o objetivo de otimizar o tempo de trabalho dos nós, para que mais informações sejam recebidas pela estação base através da reutilização de intervalos de tempo.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral propor e avaliar algoritmos para o problema de coloração BPRN, além de definir uma modelagem para resolver esse problema. Além disso, incrementar os conceitos usados no algoritmo Dsatur a fim de encontrar uma ordem de coloração dos arcos no digrafo *backbone* e usar um mecanismo de reutilização de cores com o intuito de minimizar o número de cores usadas em uma coloração BPRN. Por fim, avaliar os algoritmos propostos e os algoritmos usados para comparação através de experimentos em cenários com instâncias de diferentes níveis de densidade, motivadas em topologias do domínio das redes sem fio.

1.2.2 Objetivos Específicos

a) Revisar a bibliografia sobre algoritmos relacionados com o modelo de coloração PRN;
b) Propor os algoritmos Bsatur e Bswap para a coloração BPRN baseados no algoritmo Dsatur;
c) Adaptar os algoritmos BF e DFOr de coloração PRN para a coloração BPRN;
d) Avaliar os algoritmos em três cenários com instâncias de baixa, média e alta densidade;
e) Comparar os algoritmos Bsatur, Bswap, BF e DFOr em relação ao número de

cores individual de cada instância, menor número de cores, média e desvio padrão dos números de cores;

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em seis capítulos, com os capítulos posteriores à introdução, sendo descritos a seguir.

a) O Capítulo 2 apresenta conceitos de teoria dos grafos vistos nos algoritmos apresentados e nos modelos de coloração estudados. Em seguida, o problema de coloração de grafos é discutido. Posteriormente, explanam-se características da camada de enlace em redes sem fio para fundamentar a relação dos modelos de coloração com o problema de escalonamento TDMA de enlaces. Por fim, apresentam-se os principais fundamentos relacionados com os modelos de coloração PRN e BPRN.
b) O Capítulo 3 destaca os principais trabalhos relacionados a esta pesquisa. Esses trabalhos abordam o problema de escalonamento TDMA em redes sem fio, onde propuseram algoritmos de coloração e algoritmos que combinam otimização com o algoritmo de Bellman-Ford, além de um estudo da complexidade da coloração BPRN em diferentes classes de grafos, considerando o digrafo *backbone* como uma árvore geradora orientada em direção a um vértice raiz.
c) O Capítulo 4 expõe os algoritmos propostos para a coloração BPRN. Além disso, são apresentados os algoritmos usados para comparação com os algoritmos propostos.
d) O Capítulo 5 apresenta os resultados encontrados através de uma avaliação empírica dos algoritmos de coloração BPRN.
e) O Capítulo 6 destaca as contribuições alcançadas e elenca os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos importantes para o melhor entendimento deste trabalho. Esses conceitos estão relacionados com a teoria dos grafos, coloração de grafos, redes sem fio e modelos de coloração.

2.1 TEORIA DOS GRAFOS

Um grafo $G = (V(G),\ E(G))$ é um par $(V(G),\ E(G))$, onde os elementos de $V(G)$ são denominados de vértices e os elementos de $E(G)$ são denominados de arestas. O conjunto $E(G)$ é um subconjunto de $[V(G)]^2$, com $[V(G)]^2$ representando todos os pares não-ordenados de elementos de $V(G)$. Os elementos de $[V(G)]^2$ são subconjuntos de $V(G)$ com cardinalidade 2, onde cada elemento tem a forma $\{u,\ v\}$, sendo u e v dois elementos distintos de $V(G)$ (DIESTEL, 2000).

Uma aresta $uv \in E(G)$ significa que u e v são suas extremidades e que os vértices u e v são vizinhos ou adjacentes em $V(G)$. Além disso, a aresta uv é incidente em u e em v. O conjunto $N_G(v)$ de todos os vizinhos de um vértice $v \in V(G)$ é chamado de vizinhança de v. Um laço consiste em uma aresta formada por um par de vértices iguais. Uma aresta é paralela com outra se elas têm as mesmas extremidades. Um grafo G é simples se não tem arestas paralelas, laços e seu conjunto de vértices é finito. Duas arestas são adjacentes se possuem um vértice em comum. Um grafo H é um subgrafo de G se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. Um subconjunto S de $V(G)$ é chamado de um conjunto independente se nenhum par de vértices de S é uma aresta em $E(G)$. Um emparelhamento é um subconjunto de arestas em $E(G)$, que não são adjacentes entre si.

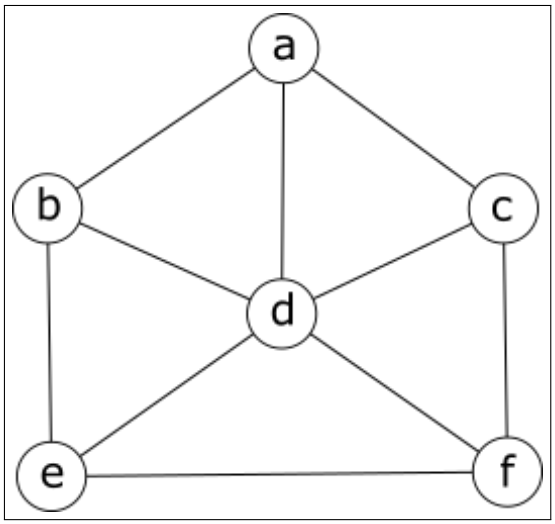
O grau de um vértice $v \in V(G)$ é a cardinalidade de $N_G(v)$ e o grau máximo dos vértices de $V(G)$ será denominado de $\Delta(G)$. Um caminho em um grafo G é uma sequência de vértices distintos $p =< v_1,\ v_2,\ ...,\ v_k >$ de modo que $\{v_1v_2, v_2v_3, v_{k-1}v_k\} \subseteq E(G)$, onde v_1 e v_k são as extremidades do caminho p. Um grafo é conexo se existe um caminho entre quaisquer dois vértices distintos em $V(G)$. Um ciclo em um grafo consiste de uma sequência de vértices distintos $c =< v_1, v_2, ..., v_k >$ de forma que $\{v_1v_2, v_2v_3, v_{k-1}v_k, v_kv_1\} \subseteq E(G)$. Um grafo é acíclico se não possui um ciclo como subgrafo.

Uma árvore é um grafo acíclico e conexo. Os vértices de grau 1 em uma árvore são denominados de folhas. Já os vértices que não são folhas recebem a denominação de vértices internos. Uma árvore $T_{v_r} = (V(T_{v_r}),\ A(T_{v_r}))$ geradora de um grafo G, orientada em direção a um

vértice raiz $v_r \in V(G)$, é uma árvore formada por um conjunto de caminhos $u \rightsquigarrow v_r$, onde cada um consiste em uma sequência de vértices $v_1, v_2, ..., v_k$, com $u = v_1$, $v_r = v_k$ e $v_i v_{i+1} \in A(T_{v_r})$, para $1 \leq i \leq k-1$.

A Figura 3 apresenta a estrutura de um grafo, onde a, b, c, d, e, e f são os vértices e, ab, ac, ad, bd, be, cd, cf, de, df e ef representam as arestas. Apenas o vértice d possui o grau 4, e todos os outros vértices possuem o grau 3. Um caminho do vértice a ao vértice c pode ser representado pelos vértices a, b, d e c. Da mesma forma, um ciclo pode ser dado pelos vértices a, b, d, c e a.

Figura 3 – Exemplo de um grafo.



Fonte – Elaborado pelo autor

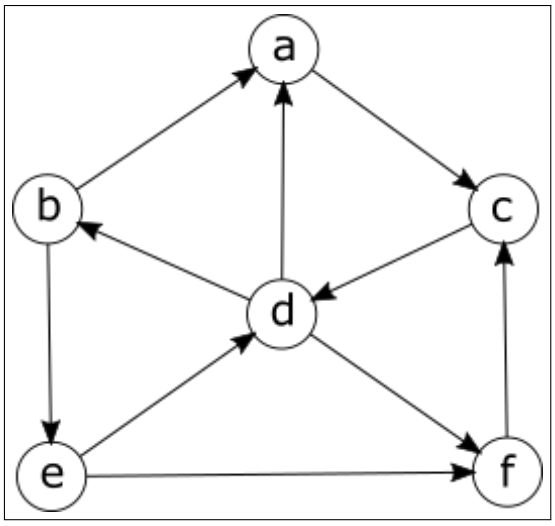
Um digrafo $D = (V(D), A(D))$ consiste em um conjunto de vértices $V(D)$ e um conjunto de arestas $A(D)$ de pares ordenados de vértices distintos (BANG-JENSEN; GUTIN, 2009). Os pares $uv \in A(D)$ são chamados de arcos. Um arco em $A(D)$ de um vértice u para um vértice v é denotado por uv. Em um arco uv, o primeiro vértice u é chamado de cauda e o segundo vértice v é chamado de cabeça. A cauda e a cabeça de um arco são chamadas de pontos de extremidade.

Um arco a é um arco de entrada de um vértice v se $a = uv$ para algum vértice cauda u. Da mesma forma, um arco a é um arco de saída de um vértice u, se $a = uv$ para algum vértice cabeça v. Denotamos por $N^+(u) = \{b \mid ub \in A(D)\}$ o conjunto de arcos que saem de u e $N^-(u) = \{b \mid bu \in A(D)\}$ o conjunto de arcos que entram em u. O grau de saída de u é $d^+(u) = |N^+(u)|$ e o grau de entrada de u é $d^-(u) = |N^-(u)|$ (ROCHA; SASAKI, 2017).

A Figura 4 ilustra um digrafo, com a, b, c, d, e e f sendo os vértices e, ba, da, db,

ac, fc, cd, ed, be, df, e ef representado os arcos. No arco ba, o vértice b é a cauda e o vértice a é a cabeça. O arco ba é um arco de saída do vértice b, e é um arco de entrada do vértice a. O grau de entrada do vértice a é 2, pois os arcos ba e da entram em a. O grau de saída do vértice a é 1, pois sai apenas o arco ac de a.

Figura 4 – Exemplo de um digrafo.



Fonte – Elaborado pelo autor

2.2 COLORAÇÃO DE GRAFOS

Os problemas de coloração de grafos aparecem com frequência em diferentes domínios. Como exemplo, considere o problema de colorir o mapa de um país, onde estados vizinhos não podem ser coloridos com a mesma cor. Diante deste problema, podem surgir algumas perguntas em relação ao número de cores necessárias para colorir tal mapa, conforme as restrições do problema. O problema de coloração conhecido como o Problema das Quatro Cores, criado por Francis Guthrie em 1852, deu origem as pesquisas direcionadas para uma área de teoria dos grafos, denominada de coloração de grafos (BONDY; MURTY, 2008).

Esse problema pode ser visto como um problema de decisão, onde verifica-se a possibilidade de colorir qualquer mapa usando no máximo quatro cores. O Problema das Quatro Cores foi resolvido por Appel e Haken (1976) com o auxílio de um computador. Esse resultado afirma que qualquer mapa pode ser colorido com apenas quatro cores, ficando conhecido como o Teorema das Quatro Cores (ROBERTSON *et al.*, 1997).

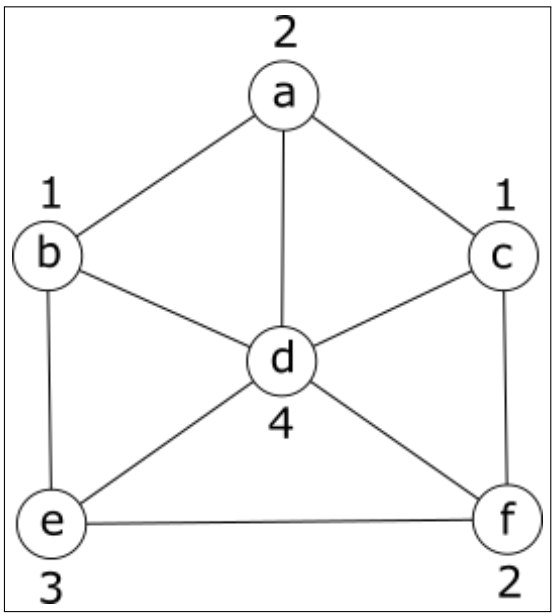
A coloração de grafos consiste na partição de vértices ou arestas em coleções chamadas classes de cores respeitando certas restrições. Na coloração de vértices, dois vértices

adjacentes não recebem a mesma cor. De forma semelhante, na coloração de arestas, duas arestas adjacentes recebem cores diferentes.

2.2.1 Coloração de Vértices

Uma k-coloração dos vértices de um grafo $G = (V(G), E(G))$ corresponde a uma função $\varphi : V(G) \rightarrow \{1, ..., k\}$, onde $\varphi(v)$ é denotado como a cor do vértice $v \in V(G)$. A coloração dos vértices de G é própria se não houver dois vértices adjacentes com a mesma cor, ou seja, se $\varphi(u) \neq \varphi(v)$, para toda aresta uv em $E(G)$. A Figura 5 ilustra uma coloração própria dos vértices de um grafo.

Figura 5 – Exemplo de uma coloração própria dos vértices de um grafo.



Fonte – Elaborado pelo autor

Uma k-coloração pode também ser vista como uma partição dos vértices $V_1 \cup V_2 \ ... \ \cup V_k$ de $V(G)$, onde os conjuntos V_i são denominados de classes de cores. Uma k-coloração é própria se os conjuntos V_i forem independentes. Um grafo é k-colorível se admite uma k-coloração própria. O número cromático $\chi(G)$ de G corresponde ao menor número de cores usadas em uma coloração própria de seus vértices. O problema de determinar o número cromático de um grafo arbitrário é NP-completo (GAREY; JOHNSON, 1990). Um grafo G é k-cromático se $\chi(G) = k$.

Um algoritmo guloso para a coloração de vértices, consiste em ordenar os vértices com base em algum critério, para indicar a ordem em que serão coloridos. Logo após, cada

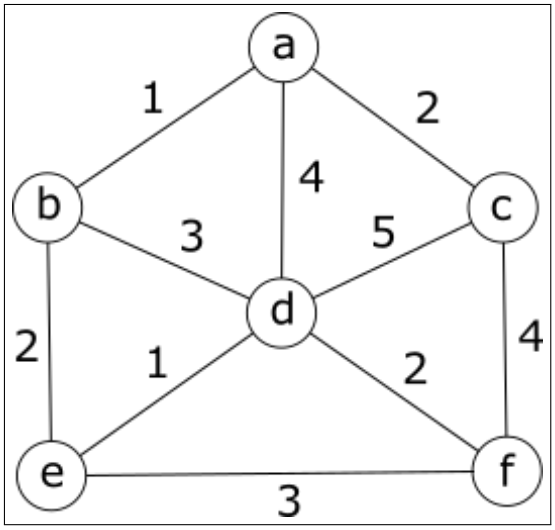
vértice é colorido em ordem com a menor cor que não é usada para colorir nenhum de seus vizinhos (LUIZ, 2015). Esse algoritmo guloso pode encontrar diferentes colorações, isso vai depender da ordem escolhida para colorir os vértices. Essa observação explica a complexidade do problema de coloração de vértices. O uso desse algoritmo permite obter um limite superior para $\chi(G)$, pois o número cromático de um grafo G não será maior do que $\Delta(G)+1$. Esse resultado faz sentido porque os vértices adjacentes de um vértice a ser colorido usam no máximo $\Delta(G)$ cores. Além disso, o número de cores usadas pelos vizinhos de um vértice a ser colorido não é maior do que o grau desse vértice. Assim, uma das cores $1,2,...,\Delta(G)+1$ estará disponível para cada vértice do grafo. Há uma ordem dos vértices que leva a uma coloração ótima. O problema é justamente saber qual ordenação dos vértices irá produzir uma solução ótima (BONDY; MURTY, 2008).

2.2.2 Coloração de Arestas

Uma coloração de arestas de um grafo $G = (V(G),\ E(G))$ consiste na atribuição de uma cor para cada aresta de G. Desse modo, se em uma coloração de arestas, as arestas adjacentes forem coloridas com cores diferentes, o resultado será uma coloração própria de arestas. Uma coloração própria de arestas que usa cores de um conjunto de k cores é denominada de uma k-aresta-coloração.

Uma k-aresta-coloração de G pode ser descrita como uma função $\varphi : E(G) \rightarrow \{1,...,k\}$, onde $\varphi(e) \neq \varphi(e')$, para todo par de arestas e e e' adjacentes em $E(G)$. A Figura 6 ilustra uma coloração própria das arestas de um grafo. Um grafo G é k-aresta-colorível se houver uma k-aresta-coloração de G. O índice cromático $\chi^{'}(G)$ de G é o menor inteiro k, para o qual G é k-aresta-colorível. Se G é um grafo k-aresta-colorível para algum inteiro positivo k, então $\chi^{'}(G) \leq k$.

Figura 6 – Exemplo de uma coloração própria das arestas de um grafo.



Fonte – Elaborado pelo autor

Uma k-aresta-coloração com as cores $1, 2, ..., k$ particiona as arestas de um grafo G em conjuntos E_i de arestas que têm a cor i. Os conjuntos não-vazios entre $E_1, E_2, ..., E_k$ são as classes de cores de $E(G)$, para uma determinada k-aresta-coloração. O índice cromático também pode ser visto como o número mínimo de emparelhamentos em que as arestas de G podem ser particionadas (CHARTRAND, 2009).

O resultado mais expressivo em relação a coloração de arestas é o Teorema de Vizing (VIZING, 1964). Esse Teorema determina que o número mínimo de cores usadas para colorir as arestas de um grafo G corresponde aos valores $\Delta(G)$ ou $\Delta(G)+1$. Assim, os grafos podem ser classificados em Classe 1 ou Classe 2. A Classe 1 é composta por grafos que precisam apenas de $\Delta(G)$ cores para serem coloridos. Na Classe 2 estão os grafos que têm o índice cromático igual a $\Delta(G)+1$. O problema de determinar o índice cromático de um grafo é NP-completo (HOLYER, 1981). Dessa forma, é justificável o uso de algoritmos que encontrem boas soluções para esse problema em tempo polinomial.

2.3 REDES SEM FIO

Uma rede sem fio é composta por um conjunto de nós que se comunicam entre si usando ondas de rádio ou de luz com o espaço como meio de propagação (GEIER, 2004). Essas redes podem ser dotadas de nós móveis e possuírem uma infraestrutura de rede, onde os principais componentes são (KUROSE; ROSS, 2013):

- Hospedeiro sem fio (nó): um nó em uma rede sem fio pode ser um notebook, um

palmtop, um smartphone ou um computador de mesa, entre outros.

- Estação base: uma estação base pode transmitir dados para um nó ou receber dados do mesmo. Em geral, uma estação base é responsável por coordenar as transmissões dos nós associados a ela, ou seja, os nós que estão dentro de seu alcance de comunicação. Além disso, os nós podem usar a estação base para retransmitir dados entre ele e a rede maior. De forma geral, o modo de infraestrutura representa uma estação base com os nós associados a ela, onde os serviços de rede, como a atribuição de endereçamento e roteamento, são de responsabilidade da rede maior na qual os nós estão conectados através da estação base. Uma torre de transmissão em uma rede de celulares é um exemplo de estação base.
- Enlaces sem fio: um enlace de comunicação sem fio conecta um nó a outro nó ou um nó à estação base.
- Infraestrutura de rede: consiste na rede maior (por exemplo, à Internet) que um nó pode se comunicar.

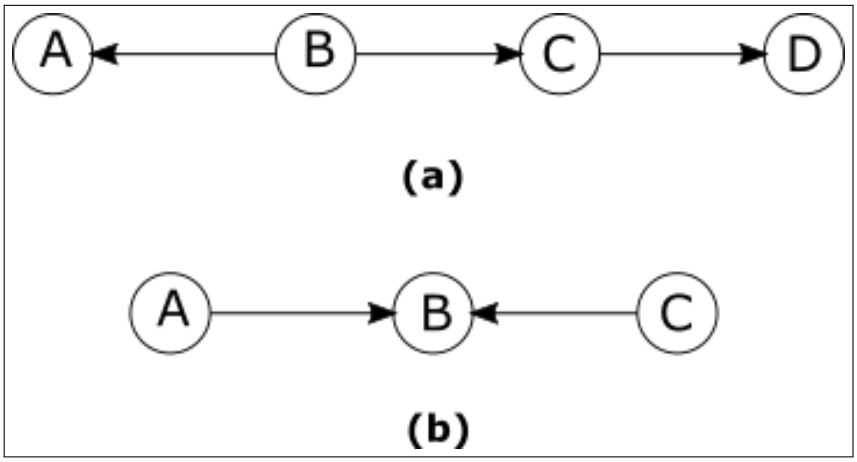
2.3.1 Conflitos de Acesso Múltiplo

Os enlaces em redes são caracterizados por serem de ponto a ponto e enlaces de difusão. Um enlace ponto a ponto é caracterizado por existir um único nó remetente em uma extremidade e um único nó receptor na outra. Já um enlace de difusão consiste em vários nós transmissores e receptores conectados a um único canal de transmissão compartilhado. Assim, quando um nó transmite um quadro, os nós em seu alcance de comunicação irão receber uma cópia. Esse modo de transmissão é conhecido como transmissão por difusão ou transmissão *broadcast*. A transmissão por difusão na camada de enlace é usada em tecnologias como a Ethernet e LANs sem fio (KUROSE; ROSS, 2013).

Um problema a ser tratado na camada de enlace é o problema de acesso múltiplo, ou seja, como coordenar o uso de um canal de difusão compartilhado por vários nós transmissores e receptores. O acesso múltiplo ao canal de difusão compartilhado causa alguns problemas, tais como o problema do terminal exposto e o problema do terminal oculto (PEIXOTO, 2009). O problema do terminal exposto acontece quando um nó B transmite para um nó A, e um nó C no raio de transmissão de B quer transmitir para um nó D, mas acaba não transmitindo ao perceber a transmissão de B e, por isso, considera que o canal está ocupado. O problema neste caso é que

C poderia ter transmitido para *D* sem problema de colisão. O problema do terminal oculto ocorre porque um nó *A* transmite para um nó *B*, e um nó *C* decide transmitir para *B*, por acreditar que nenhum nó está transmitindo no seu alcance de transmissão. Assim, os quadros transmitidos de *A* e *C* colidem em *B*. A Figura 7(a) ilustra o problema do terminal exposto e a Figura 7(b) apresenta a situação em que ocorre o problema do terminal oculto.

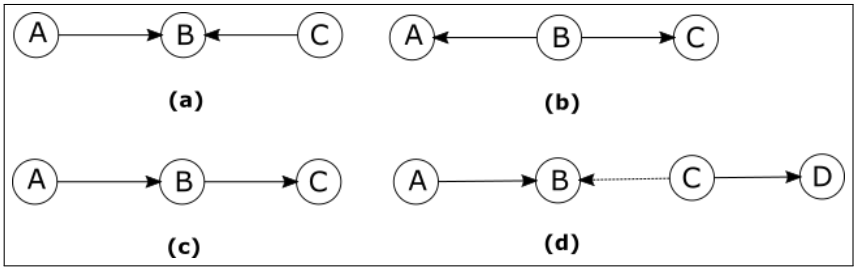
Figura 7 – Problema do terminal exposto e problema do terminal oculto.



Fonte – Elaborado pelo autor

Os enlaces sem fio causam interferências com outros enlaces quando seus quadros colidem em um receptor em transmissões simultâneas (DJUKIC; VALAEE, 2009). Os três tipos de interferências primárias ocorrem por causa dos enlaces adjacentes. No primeiro tipo, um nó *A* não consegue decodificar os quadros dos nós *B* e *C* em transmissões paralelas, conforme apresentado na Figura 8(a). A Figura 8(b) ilustra o segundo tipo, onde um nó *A* não consegue separar quadros para dois receptores diferentes. O terceiro tipo apresentado na Figura 8(c), caracteriza o fato de um nó *A* não poder transmitir e receber ao mesmo tempo. Ainda existem as interferências entre enlaces não adjacentes, sendo denominadas de interferências secundárias. As interferências secundárias ocorrem quando um nó *A* transmite para um nó *B*, e um nó *C* transmite ao mesmo tempo para um nó *D*, com *B* dentro do raio de transmissão de *C*, conforme a Figura 8(d).

Figura 8 – Exemplo de interferências primárias e secundárias.



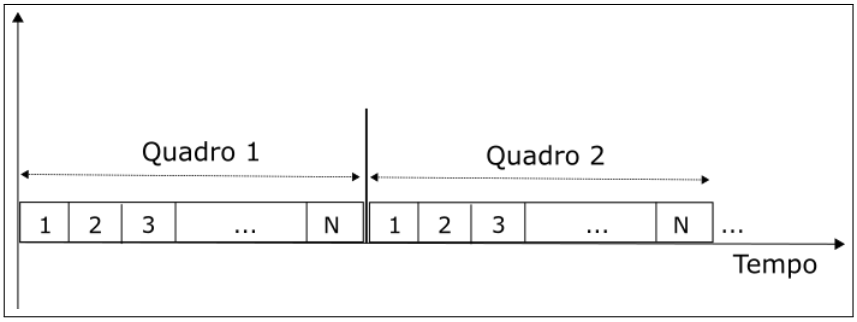
Fonte – Elaborado pelo autor

2.3.2 Escalonamento TDMA

O escalonamento TDMA representa uma forma de coordenar os nós no uso do canal de difusão compartilhado através da atribuição de intervalos de tempo. O objetivo é proporcionar acesso livre de colisões ao canal, além de possibilitar que os nós minimizem as perdas na transmissão e recepção de informações em função de problemas causados por colisões e interferências na rede.

Existem várias técnicas de acesso múltiplo que permitem o escalonamento do canal, por exemplo, acesso múltiplo por divisão de frequência (FDMA), acesso múltiplo por divisão de tempo (TDMA) e acesso múltiplo por divisão de código (CDMA) (STOJMENOVIC, 2005). Neste trabalho, será dado ênfase a técnica TDMA, por causa da relação com a coloração BPRN. Essa técnica divide o tempo em quadros, onde cada quadro é divido em um determinado número de intervalos de tempo (NELSON; KLEINROCK, 1985). Um quadro TDMA é ideal quando possui um número mínimo de intervalos de tempo, pois diminui o ciclo de trabalho dos nós e aumenta a quantidade de informações trafegadas na rede. Um escalonamento TDMA representa uma atribuição dos intervalos de tempo aos nós ou enlaces da rede a fim de evitar a ocorrência de interferências e colisões em transmissões simultâneas. A Figura 9 ilustra a estrutura de um quadro TDMA com N intervalos de tempo.

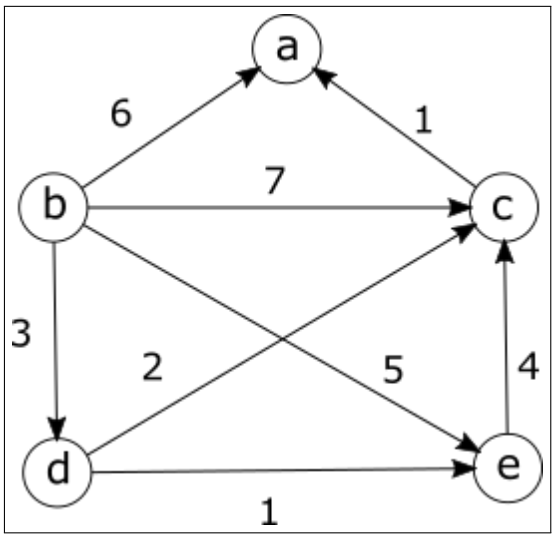
Figura 9 – Exemplo de um quadro TDMA.



Fonte – Elaborado pelo autor

No escalonamento de nós, também conhecido como escalonamento *broadcast*, cada nó recebe um intervalo de tempo no quadro TDMA. Além disso, a transmissão de um nó alcança todos os seus vizinhos sem a ocorrência de colisões. Já no escalonamento de enlaces, cada enlace possui um intervalo de tempo no quadro TDMA. Assim, se existir um enlace *ab*, o nó *a* poderá transmitir informações para o nó *b* livres de colisões. No escalonamento *broadcast*, ou no escalonamento de enlaces, uma mesma posição no quadro TDMA pode ser alocada para mais de um nó ou enlace sem a ocorrência de interferências e colisões em decorrência da distribuição espacial entre os nós na rede. A Figura 10 ilustra a reutilização de intervalos de tempo nos enlaces *de* e *ca*. O problema de escalonamento TDMA de enlaces é NP-difícil quando o objetivo é encontrar um quadro TDMA de tamanho mínimo (ARIKAN, 1984; EVEN *et al.*, 1984).

Figura 10 – Exemplo do reuso de intervalos de tempo em um escalonamento TDMA de enlaces da rede.



Fonte – Elaborado pelo autor

A tarefa de encontrar um escalonamento livre de colisões com base no modelo de

interferências não é tão complicada. O difícil é encontrar um escalonamento com o número mínimo de intervalos de tempo. O uso eficiente de um escalonamento com base na técnica TDMA está relacionado com o aumento na taxa de dados trafegados na rede, diminuição de atrasos e perda de dados (ROCHA; SASAKI, 2017).

2.4 MODELOS DE COLORAÇÃO PARA O ESCALONAMENTO TDMA

Os modelos capturam as características essenciais e representam os aspectos de interesse no domínio do problema em estudo. Um dos objetivos com o uso de modelos é ter uma boa representação do sistema para resolver o problema. A qualidade de um modelo pode ser medida por sua fidelidade em retratar as informações do domínio de um problema, deixando de lado aquelas que não precisam ser representadas. Um fator importante é a eficiência no uso de um modelo em relação ao tempo gasto para se chegar a uma solução (HUSON, 1995).

Os grafos podem ser usados para modelar problemas em diferentes domínios, por exemplo, o problema de escalonamento de enlaces em redes sem fio, onde o escalonamento é alcançado com a coloração dos arcos do digrafo que representa a rede. Essa relação retratada em alguns trabalhos direcionados para o escalonamento de enlaces é modelada como coloração PRN (ARIKAN, 1984; RAMANATHAN; LLOYD, 1993; HUSON, 1995).

O modelo de coloração PRN foi apresentado pela primeira vez no trabalho de (ARIKAN, 1984), sendo referenciado em outros trabalhos voltados para o escalonamento de enlaces. O modelo PRN é um modelo clássico para o escalonamento TDMA de enlaces em redes sem fio. Uma coloração PRN de um digrafo $D = (V(D), A(D))$ consiste na atribuição de cores para os arcos de D, visando encontrar um escalonamento TDMA de enlaces sem interferências primárias e secundárias. De outra forma, uma coloração PRN pode ser descrita como uma função $c : A(D) \to \mathbb{N}$, onde $c(uv) = c(ab)$, se as restrições apresentadas a seguir forem satisfeitas.

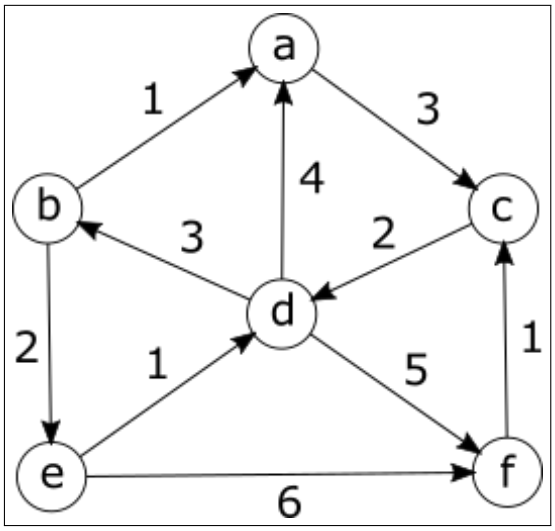
a) a, b, u e v são vértices mutuamente distintos; e

b) $av \notin A(D)$ e $ub \notin A(D)$.

As restrições em (a) indicam que arcos adjacentes recebem cores diferentes. Essas restrições têm a finalidade de evitar os conflitos primários. As restrições em (b) asseguram que um par de arcos não adjacentes recebam cores diferentes, quando a cauda de um arco e a cabeça do outro coincidirem com as extremidades de um terceiro arco dos quais são adjacentes, conforme os arcos be, da e ba na Figura 11. Essas restrições evitam que os conflitos secundários ocorram. O objetivo é encontrar uma coloração PRN que minimize o número de cores usadas

na coloração dos arcos de D. Uma coloração PRN pode ser traduzida como um escalonamento TDMA de enlaces, onde as cores dos arcos de D são associadas aos intervalos de tempo de um quadro TDMA. O índice cromático de uma coloração PRN representa o número mínimo de cores usadas em uma coloração PRN dos arcos de D. A Figura 11 ilustra o número mínimo de cores usadas em uma coloração PRN.

Figura 11 – Número mínimo de cores usadas em uma coloração PRN.

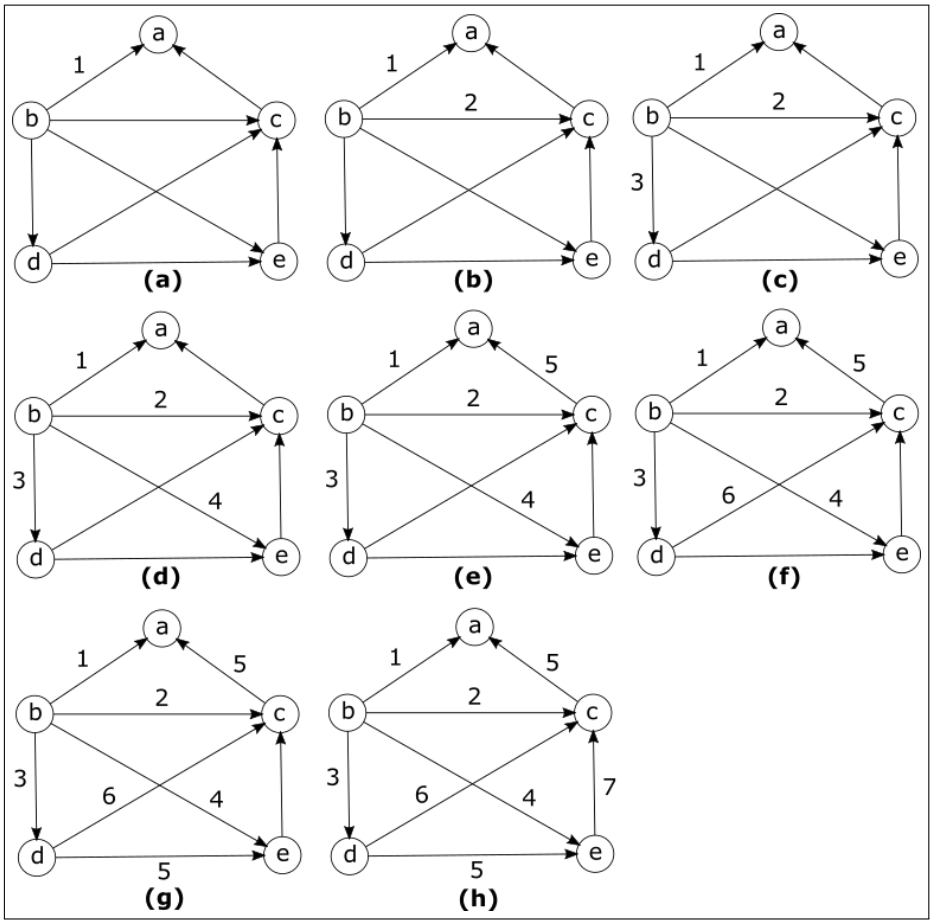


Fonte – Elaborado pelo autor

Observe a Figura 12 em que os arcos foram coloridos na seguinte ordem: ba, bc, bd, be, ca, dc, de e ec. O arco ba foi colorido com a cor 1, pois antes de colorir o primeiro arco, todas as cores estão disponíveis (Figura 12(a)). O arco bc não pode receber a cor 1, por causa de conflito primário. Nesse caso, podemos atribuir a cor 2 para esse arco, sem causar conflito primário com o arco ba (Figura 12(b)). O arco bd causa conflitos primários com os arcos ba e bc, por isso irá receber a cor 3 (Figura 12(c)). De forma semelhante ao arco bd, o arco be será colorido com a cor 4, por causa de conflitos primários com os arcos ba, bc e bd (Figura 12(d)). O arco ca não pode receber as cores 1 e 2, por causa de conflitos primários. Além de não poder receber as cores 3 e 4, por causa de conflitos secundários com os arcos bd e be. Nesse caso, o arco ca pode receber a cor 5, sem causar conflitos com os arcos que estão coloridos (Figura 12(e)). A cor atribuída ao arco dc deve ser diferente das cores dos arcos ba, bc, bd, be e ca (Figura 12(f)). Note que apesar dos arcos dc e ba não serem adjacentes, eles se conflitam por causa do arco bc. Esse é um exemplo de conflito secundário. Assim, o arco dc deve ser colorido com a cor 6. O arco de deve receber uma cor que seja diferente das cores dos arcos ba, bc, bd, be e dc. Dessa forma, o arco de será colorido com a cor 5 (Figura 12(g)). Por fim, o arco ec vai

receber a cor 7, que é diferente das cores dos arcos ba, bc, bd, be, ca, dc e de (Figura 12(h)).

Figura 12 – Exemplo de uma coloração PRN.



Fonte – Elaborado pelo autor

O modelo de coloração PRN está relacionado com o problema de coloração de arestas visto em teoria dos grafos. Além disso, como já foi mencionado os problemas de coloração de vértices e coloração de arestas são problemas NP-completos. A coloração PRN fornece um escalonamento livre de interferências para o problema de escalonamento TDMA de enlaces. No entanto, considera que todos os arcos do digrafo da rede serão coloridos. Dessa forma, encontrar um escalonamento pode ficar ainda mais complexo, pois em uma situação real no domínio das redes sem fio, apenas um subconjunto de enlaces são realmente usados no roteamento da rede (ROCHA; SASAKI, 2017). Um grafo planar é um exemplo de que o uso de todos os arcos torna o problema de coloração PRN NP-completo quando o objetivo é encontrar uma coloração PRN com o menor número de cores (LLOYD; RAMANATHAN, 1992).

Em contra partida, o modelo de coloração BPRN, que representa uma extensão do modelo de coloração PRN, considera apenas um subconjunto de arcos do digrafo a serem

coloridos, onde esse subconjunto de arcos recebe o nome de digrafo *backbone*. Dessa forma, o digrafo *backbone* formado por um subconjunto de arcos, pode representar topologias mais simples, por exemplo, uma árvore ou um digrafo com graus limitados.

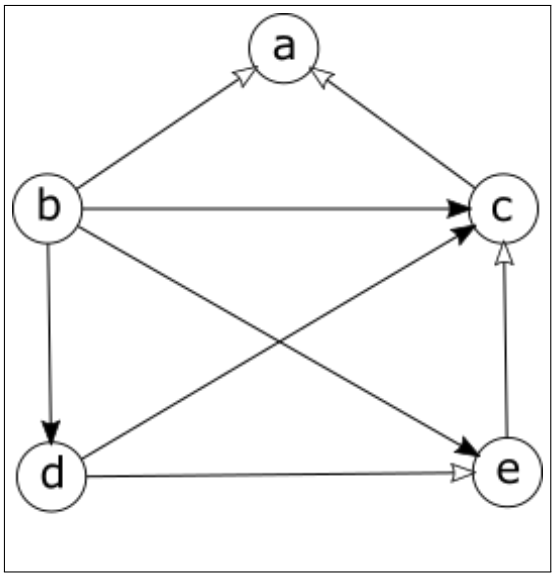
Uma coloração BPRN consiste na coloração dos arcos de um digrafo *backbone*, evitando os conflitos primários e secundários. Seja $D = (V(D), A(D))$ um digrafo, dizemos que $B = (V(B), A(B))$ é um digrafo *backbone* de D, se $V(B) = V(D)$ e $A(B) \subseteq A(D)$. Uma coloração BPRN de (D, B) é um mapeamento $c : A(B) \to \mathbb{N}$, que representa uma atribuição de cores aos arcos de B, tal que, se $c(uv) = c(ab)$, então:

a) a, b, u e v são vértices mutuamente distintos; e

b) $av \notin A(D)$ e $ub \notin A(D)$.

O índice cromático de uma coloração BPRN é o número mínimo de cores usadas em uma coloração BPRN dos arcos de (D,B). Nas restrições em (b), é preciso o uso de D para verificar os conflitos secundários, pois apenas com o uso de B não é possível verificar esses conflitos. A Figura 13 ilustra um exemplo em que usando apenas B não é possível verificar que o arco ba causa conflito secundário com o arco de, por causa do arco be que não está em B. Os arcos de B estão com a seta colorida de branco.

Figura 13 – Um exemplo em que usando apenas os arcos do digrafo backbone não é possível evitar a ocorrência de conflitos secundários.

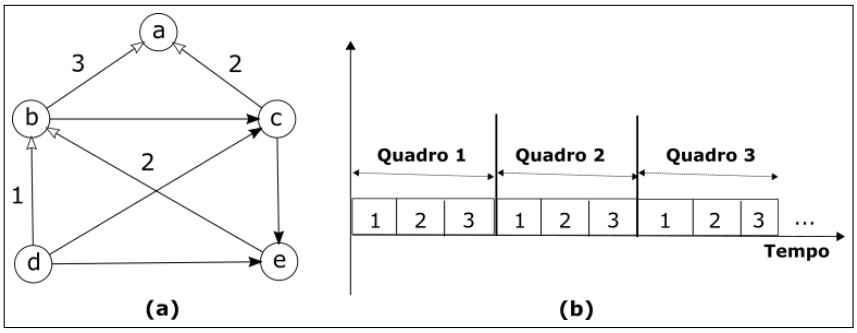


Fonte – Elaborado pelo autor

Uma rede sem fio pode ser modelada como um digrafo em que os vértices são associados aos nós da rede e os arcos representam os enlaces de comunicação entre esses nós.

Assim, um escalonamento TDMA de enlaces pode ser encontrado através de uma coloração BPRN, na qual as cores dos arcos são associadas aos intervalos de tempo de um quadro TDMA. A Figura 14(a) representa a rede, onde os arcos da rede *backbone* estão com a seta colorida de branco. Já a Figura 14(b), ilustra o mapeamento das cores dos arcos da rede *backbone* em intervalos de tempo no quadro TDMA.

Figura 14 – Mapeamento entre uma coloração BPRN e um quadro TDMA.



Fonte – Elaborado pelo autor

3 TRABALHOS RELACIONADOS

Neste capítulo, apresentam-se os principais trabalhos relacionados com este trabalho de pesquisa. Em parte, os conhecimentos expostos são referentes a um modelo de grafo de pontos, proposto em um dos trabalhos relacionados, além de um algoritmo de coloração PRN de grafo de pontos lineares. Em seguida, são mostrados alguns resultados da complexidade de coloração BPRN, quando o digrafo *backbone* é representado por uma árvore geradora orientada em direção a um vértice raiz. Além disso, são apresentadas algumas questões abertas sobre a complexidade de coloração BPRN. Posteriormente, é apresentado o algoritmo Dsatur, para a coloração de vértices de um grafo. Logo após, expõe-se um algoritmo denominado de ECDiG, para o escalonamento em redes de sensores, onde o mesmo usa coloração de arcos para a comunicação *unicast* e *broadcast*. Após isso, são apresentados algoritmos para o escalonamento TDMA de enlaces livre de conflitos, com atraso mínimo de ponta a ponta. Esses algoritmos mesclam otimização em conjunto com o algoritmo de Bellman-Ford, para o escalonamento de enlaces consciente de atraso em redes de malha. Por fim, são descritos de forma sucinta três trabalhos relacionados.

3.1 UM NOVO MODELO DE PROGRAMAÇÃO PARA REDES DE RÁDIO

O autor Huson (1995) propõe um modelo de grafo para redes sem fio e desenvolve alguns algoritmos de aproximação para o problema de escalonamento usando informações do modelo proposto. Na análise desses algoritmos, foi considerado o tempo de execução e a solução encontrada com relação as soluções ótimas para o problema. O modelo proposto considera a posição e o alcance dos nós transceptores, sendo chamado de grafo de pontos. Sejam p_i e r_i, pontos no espaço bidimensional e faixas de transmissão, onde cada faixa r_i está associada com um ponto p_i.

Um digrafo $D = (V(D), A(D))$ é formado por um conjunto de tuplas $< p_i,\ r_i >$, tal que, para cada p_i existe um vértice $v_i \in V(D)$. Dessa forma, se $d(p_i,\ p_j) \leq r_i$, então o arco $(v_i,\ v_j) \in A(D)$, onde $d(p_i,\ p_j)$ é a distância euclidiana entre os pontos p_i e p_j. Um digrafo construído a partir de um conjunto de pontos é denominado de grafo de pontos. Um grafo de pontos lineares é constituído por um conjunto de pontos no espaço unidimensional. Esse tipo de grafo pode ser desenhado como uma linha reta ao reproduzir uma rede. Além disso, não é necessário que os nós formem tal reta. O importante é representar os enlaces entre esses nós por um conjunto de pontos em uma linha com faixas associadas. Já um grafo de pontos planares é

formado a partir de um conjunto de pontos no espaço bidimensional.

Esse trabalho abordou a coloração de vértices em grafos dirigidos e não dirigidos para o escalonamento *broadcast*. A proposta de algoritmos e provas buscou enfatizar a complexidade de cada uma das abordagens de coloração em grafos de pontos lineares e grafos de pontos planares. O trabalho faz uma análise da complexidade do problema de escalonamento e desenvolve algoritmos com base nas propriedades de grafo de pontos. Além disso, investigou o escalonamento de enlaces, propondo algoritmos para a coloração PRN em grafo de pontos. O desempenho dos algoritmos propostos é comparado com algoritmos de coloração de outros trabalhos.

Os grafos de pontos modelam os nós de uma rede e os seus enlaces de comunicação entre esses nós. Esses grafos representam um subconjunto próprio de grafos e não são equivalentes a árvores ou grafos planares. Um grafo de pontos com intervalos uniformes é equivalente a um grafo de unidade de disco para o espaço bidimensional. Uma representação adequada de um grafo usado para modelar uma rede pode resultar em melhores resultados com relação ao tempo de execução ou nas soluções encontradas.

Uma coloração de um digrafo foi definida para o escalonamento *broadcast*, visto que a coloração de vértices de um grafo para o escalonamento *broadcast* introduzia dependências inexistentes entre os nós. Foi demonstrado a complexidade do problema de coloração de um grafo e do problema de coloração de um digrafo. A coloração de grafos de pontos unidimensional pode ser resolvida em tempo polinomial de forma ótima. Já o problema de coloração de grafos de pontos com dimensões maiores do que 1 é NP-completo.

Os algoritmos implementados para o escalonamento *broadcast* foram divididos em duas categorias. Os algoritmos geométricos buscaram incluir as informações de localização, com o intuito de compreender a relação espacial dos nós e o uso de algoritmos gulosos. A complexidade dos algoritmos geométricos foram melhores em comparação com outros algoritmos testados. No entanto, esses algoritmos usaram mais cores em média do que os outros algoritmos. Os algoritmos topológicos usam a topologia de interconexões da rede em contraste com os algoritmos geométricos que usam as localizações dos nós na rede. A complexidade do tempo de execução dos algoritmos topológicos foi melhor do que a complexidade dos algoritmos tradicionais que foram adaptados para o problema de coloração de um digrafo.

O problema de coloração PRN foi apresentado como sendo NP-completo para grafos de pontos de dimensão maior do que 1. Assim como no escalonamento *broadcast*, foram implementados dois tipos de algoritmos para o escalonamento de enlaces. Um algoritmo

geométrico foi desenvolvido, onde os enlaces são coloridos conforme suas localizações na rede. Os outros algoritmos desenvolvidos foram os algoritmos topológicos que dependem das conexões entre os nós. Os algoritmos desenvolvidos para a coloração de enlaces tiveram uma complexidade de tempo de execução $O(n^4)$, onde n é o número de vértices de um grafo de pontos.

Um algoritmo de tempo polinomial foi apresentado para a coloração PRN de digrafos de pontos lineares, sendo representado pelo Algoritmo 1. Esse algoritmo mantém quatro matrizes de tamanho NxN^2 durante a coloração dos arcos de entrada, onde cada linha está associada a um vértice e cada coluna representa uma cor c, com $N = |V(D)|$ e $1 \leq c \leq N^2$. O número de colunas de cada matriz é N^2, pois no caso de um digrafo totalmente conectado, cada arco recebe uma cor diferente. Essas matrizes denominadas de NOTIN, NOTOUT, NOTSIN e NOTSOUT são preenchidas conforme as equações descritas a seguir.

$$NOTIN[v][c] = \begin{cases} 0, & \text{se existir um arco } uv \text{ com a cor } c \\ 1, & \text{caso contrário} \end{cases} \tag{3.1}$$

$$NOTOUT[v][c] = \begin{cases} 0, & \text{se existir um arco } vu \text{ com a cor } c \\ 1, & \text{caso contrário} \end{cases} \tag{3.2}$$

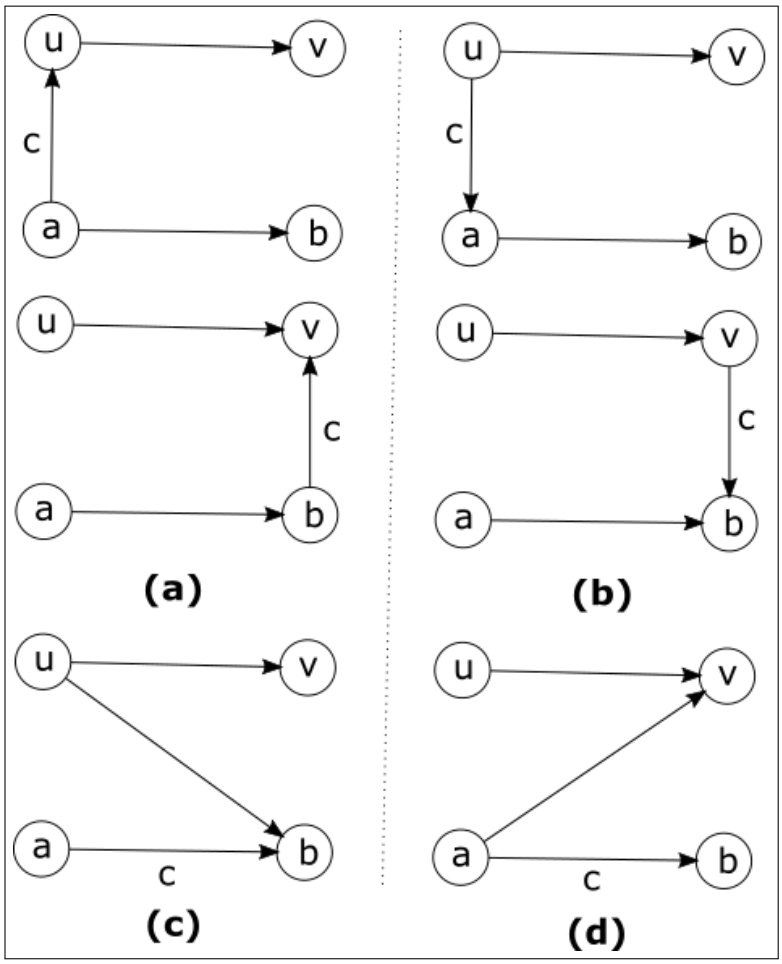
$$NOTSIN[b][c] = \begin{cases} 0, & \text{se existir um arco } uv \text{ com a cor } c \text{ e um arco } ub \\ 1, & \text{caso contrário} \end{cases} \tag{3.3}$$

$$NOTSOUT[a][c] = \begin{cases} 0, & \text{se existir um arco } uv \text{ com a cor } c \text{ e um arco } av \\ 1, & \text{caso contrário} \end{cases} \tag{3.4}$$

Após a coloração de um arco uv com a cor c, as matrizes são atualizadas para permitir a verificação de conflitos primários e secundários na próxima atribuição de cor para um arco a ser colorido. Assim, as matrizes são atualizadas da seguinte forma: NOTIN[v][c] = 0, NOTOUT[u][c] = 0, NOTSIN[b][c] = 0 e NOTSOUT[a][c] = 0, com b vizinho de u e v vizinho de a. Essas atualizações indicam que a cor c não será mais usada para colorir outros arcos de entrada ou saída dos vértices u e v. Além disso, a cor c não poderá mais ser usada para colorir arcos de entrada em b e arcos de saída em a. A Figura 15 ilustra as situações em que uma

cor *c* não pode ser usada para colorir um arco *uv*, por causa de conflitos primários ou conflitos secundários.

Figura 15 – Uma cor *c* não pode ser usada para colorir um arco *uv*, por causa de conflitos primários ou secundários.



Fonte – Elaborado pelo autor

A Figura 15 tem por objetivo mostrar as situações em que uma cor *c* não pode ser usada para colorir um arco *uv*, além de fazer uma relação com as matrizes usadas para verificar os conflitos primários e secundários. A matriz NOTIN garante que um arco *uv* não seja colorido com uma cor *c*, se um vértice *u* ou um vértice *v* tem um arco de entrada com essa cor (Figura 15(a)). A matriz NOTOUT assegura que uma cor *c* não seja atribuída a um arco *uv*, quando ela é usada em um arco de saída de um vértice *u* ou de um vértice *v* (Figura 15(b)). A matriz NOTSOUT evita que uma cor *c* seja usada para colorir um arco *uv*, se um arco de entrada em um vértice *b* estiver colorido com essa cor (Figura 15(c)). Por fim, a matriz NOTSIN possibilita que um arco *uv* não receba uma cor *c*, se ela é usada em um arco de saída de um vértice *a* (Figura 15(d)).

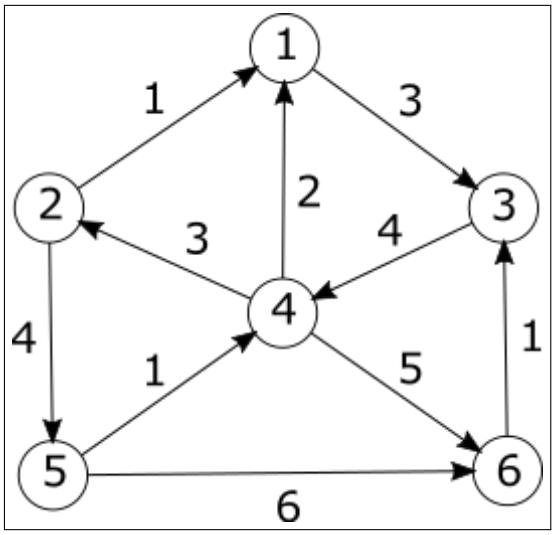
Algoritmo 1: Algoritmo de Coloração PRN em Grafos de Pontos Lineares

```
Entrada: Digrafo D = (V(D), A(D)) de pontos lineares;
Saída: Número de cores usadas na coloração PRN de D;
início
    para i = 1 até N faça
        para j = 1 até N² faça
            NOTIN[i][j] ← 1;
            NOTOUT[i][j] ← 1;
            NOTSIN[i][j] ← 1;
            NOTSOUT[i][j] ← 1;
        fim
    fim
    para u = 1 até N faça
        para v = 1 até N faça
            se vu ∈ A(D) então
                cor ← 1;
                enquanto vu ∈ A(D) não possuir uma cor permitida faça
                    se NOTIN[v][cor] ∧ NOTIN[u][cor] ∧ NOTOUT[v][cor] ∧
                     NOTOUT[u][cor] ∧ NOTSOUT[v][cor] ∧ NOTSIN[u][cor] então
                        ASGN[v][u] ← cor;
                        NOTIN[u][cor] ← 0;
                        NOTOUT[v][cor] ← 0;
                        para l = 1 até N faça
                            se vl ∈ A(D) então
                                NOTSIN[l][cor] ← 0;
                            fim
                            se lu ∈ A(D) então
                                NOTSOUT[l][cor] ← 0;
                            fim
                        fim
                    fim
                    cor ← cor + 1;
                fim
            fim
        fim
    fim
fim
```

No Algoritmo 1, os pontos são ordenados de acordo com suas posições. Assim, para dois pontos i e j, com $i < j$, os arcos de entrada do ponto i são coloridos antes dos arcos de entrada do ponto j. A pior complexidade ocorre no caso degenerado de um grafo totalmente conectado, onde cada arco recebe uma cor diferente, com um total de N^2 arcos a serem coloridos. Como o laço **"enquanto"** executa em média $N^2/2$ vezes, o teste para verificar se uma cor está livre, é executado $O(N^4)$ vezes. O laço mais interno é executado $O(N^3)$ vezes, pois ele executa

uma vez para cada arco colorido. A complexidade também pode ser expressa em termos do número de arcos m, grau máximo de entrada dos vértices ρ e número de vértices n, ou seja, a complexidade de tempo é $O(m\rho n)$. Portanto, a complexidade do tempo de execução desse algoritmo é $O(n^4)$. A Figura 16 ilustra uma coloração PRN de um grafo de pontos com o Algoritmo 1.

Figura 16 – Uma coloração PRN com o Algoritmo 1.



Fonte – Elaborado pelo autor

O trabalho também fez uma revisão da complexidade do problema de coloração PRN. Foi citado que esse problema é NP-completo quando é preciso decidir se existe uma coloração PRN de grafos arbitrários com k cores, para $k \geq 3$. A coloração PRN de grafos arbitrários é encontrada em tempo polinomial, para $k < 3$. Além disso, árvores e redes de anéis podem ser coloridas em tempo polinomial.

O escalonamento distribuído foi abordado através da implementação de um algoritmo de escalonamento *broadcast* que incorpora os resultados alcançados para os algoritmos de escalonamento centralizados. Esse algoritmo foi avaliado de maneira informal com relação à correção e não teve a sua complexidade de tempo de execução analisada.

3.2 REDES BACKBONE DE RÁDIO DE PACOTES

Os autores Rocha e Sasaki (2017), fizeram uma revisão do modelo de coloração PRN e ampliaram esse modelo para o modelo de coloração BPRN. Como já foi dito, a coloração BPRN considera apenas um subconjunto de arcos a serem coloridos. Os outros arcos são usados para verificar os conflitos secundários ao colorir os arcos do digrafo *backbone*. O trabalho

apresentou novos resultados e propôs questões abertas para o problema de escalonamento de enlaces, motivado por redes *backbone* bem estruturadas. Os resultados encontrados consideram a coloração BPRN quando o digrafo *backbone* é uma árvore T_{v_r} geradora orientada em direção a um vértice raiz v_r.

Para um ciclo $G = C_n$, com n vértices e uma árvore T_{v_r} de C_n, o índice BPRN-cromático de C_n é 2 quando n é impar, 3 se n é par e o grau de entrada de v_r é 1, e 2 caso contrário, para $n \geq 3$. Quando $G = K_n$ é um grafo completo, com $n \geq 3$, e T_{v_r} é uma árvore de K_n, o índice BPRN-cromático de K_n é $n-1$. Se um grafo G tem um ciclo C_n com os vértices x_1, x_2, x_3, x_4 e arestas $\{x_1x_2\}, \{x_2x_3\}, \{x_3x_4\}, \{x_4x_1\}$, onde os arcos x_1x_2 e x_3x_4 estão em uma árvore geradora, então esses arcos devem ser coloridos com cores diferentes. Um grafo $G = K_{n,n}$ bipartido completo e balanceado, com $n \geq 3$ e uma árvore T_{v_r} de $K_{n,n}$, tem o índice BPRN-cromático igual a n ou $n-1$.

O problema de coloração BPRN é NP-completo quando restrito a uma classe de grafos em que o problema de coloração PRN é NP-completo. Um resultado mostrou que o problema de coloração BPRN é NP-completo quando o grafo de entrada é um grafo planar. Outro resultado mostra que decidir se uma árvore T_{v_r} de um grafo G tem o o índice BPRN-cromático igual a 3 é um problema NP-completo, quando T_{v_r} tem grau máximo igual a 3 e G tem grau máximo menor ou igual a 6, ou seja, $\Delta(T_{v_r}) = 3$ e $\Delta(G) \leq 6$. Além disso, se o índice PRN-cromático de G é igual a 3, então o índice BPRN-cromático de T_{v_r} é igual 3. De forma semelhante, se o índice BPRN-cromático de T_{v_r} é igual a 3, então o índice PRN-cromático de G é igual 3.

Uma das questões abertas questiona se é possível encontrar uma coloração BPRN ótima em tempo polinomial para um grafo $G = K_{n_1,n_2}$ bipartido completo, com $n = n_1 + n_2$ vértices e uma árvore T_{v_r} de G, para $n_1, n_2 \geq 0$. Um questionamento semelhante, pergunta se é possível encontrar uma coloração BPRN ótima em tempo polinomial, quando um grafo G é bipartido conectado e T_{v_r} é uma árvore de G. Outras questões abertas estão relacionadas com a complexidade de decidir se é possível colorir uma árvore T_{v_r} de um grafo planar ou de um grafo de pontos, com no máximo k cores.

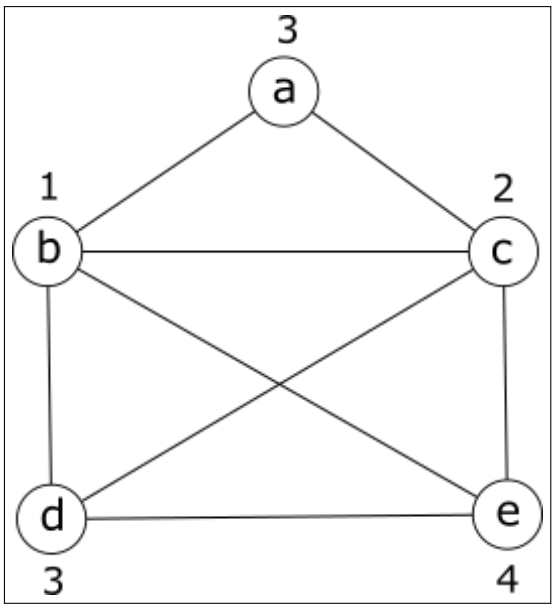
3.3 NOVOS MÉTODOS PARA COLORIR OS VÉRTICES DE UM GRAFO

O trabalho de Brélaz (1979) propõe algoritmos de coloração de vértices, tomando como informações a estrutura do grafo e o grau dos vértices. Um método foi proposto para grafos

bipartidos, sendo importante em algoritmos para encontrar cliques maximais em grafos gerais. Dentre os algoritmos propostos, um é denominado de Dsatur. O algoritmo Dsatur colore os vértices com base nas informações de grau e grau de saturação de um vértice, sendo responsáveis por indicar a ordem de coloração dos vértices. O grau de saturação de um vértice representa o número de vértices coloridos adjacentes desse vértice.

O algoritmo Dsatur começa colorindo o vértice de maior grau com a cor 1. Logo após, enquanto ainda existir vértices não coloridos, escolhe-se um vértice de maior grau de saturação, e atribui a menor cor própria a esse vértice. Em caso de empate, o vértice com o maior grau dentre os vértices não coloridos é o escolhido. Se o empate persistir, a escolha é feita de forma arbitrária. A Figura 17 apresenta a coloração dos vértices de um grafo com o algoritmo Dsatur.

Figura 17 – Uma coloração de vértices usando o algoritmo Dsatur.



Fonte – Elaborado pelo autor

Um procedimento a ser tratado no algoritmo Dsatur é o que encontra um vértice de maior grau entre os vértices não coloridos. Esse procedimento será denominado de VERTICE_MAIOR_GRAU_NAO_COLORIDO. Além desse procedimento, há também o responsável por selecionar um vértice de maior grau de saturação, sendo denominado de VERTICE_MAIOR_SATURACAO. Outra função no algoritmo Dsatur, é a responsável por colorir um vértice, sendo representada pelo Algoritmo 2. Após colorir um vértice v com a cor c, ela incrementa o atributo de saturação de cada vizinho de v que ainda não tem uma cor. Após isso, indica, através da matriz TAB, que os vizinhos de v sem uma cor não podem mais ser coloridos

com a cor c. Essa função será denominada de COLORIR_VERTICE. Uma posição TAB[v][c] da matriz TAB é 1 quando o vértice v puder ser colorido com a cor c, e 0 em caso contrário.

Algoritmo 2: Algoritmo para Colorir um Vértice

```
Entrada: Um grafo G = (V(G), E(G)), vértice v e cor c;
início
    COR[v] ← c;
    para cada vértice i ∈ V(G) faça
        se vi ∈ E(G) ∧ COR[i] = 0 então
            TAB[i][c] ← 0;
            SATUR[i] = SATUR[i] + 1;
        fim
    fim
fim
```

Algoritmo 3: Algoritmo Dsatur

```
Entrada: Um grafo G = (V(G), E(G));
início
    v ← VERTICE_MAIOR_GRAU_NAO_COLORIDO(V(G));
    COLORIR_VERTICE(G, v, 1);
    para i = 1 até i < G.n faça
        v ← VERTICE_MAIOR_SATURACAO(V(G));
        se ∃ u ∈ V(G) tal que SATUR[u] = SATUR[v] então
            v ← VERTICE_MAIOR_GRAU_NAO_COLORIDO(V(G));
        fim
        para c = 1 até c ≤ G.n faça
            se TAB[v][c] então
                corSelecionada = cor;
                pausa;
            fim
        fim
        COLORIR_VERTICE(G, v, corSelecionada);
    fim
fim
```

O tempo de execução para encontrar um vértice não colorido de grau máximo é $O(n)$, onde n é o número de vértices do grafo G. O mesmo tempo é gasto para encontrar um vértice de maior grau de saturação entre os vértices não coloridos. O tempo requerido para colorir um vértice também é $O(n)$. Concluímos então que o tempo de execução do algoritmo Dsatur é $O(n^2)$.

3.4 ESCALONAMENTO DE TRANSMISSÃO EM REDES DE SENSORES ATRAVÉS DA COLORAÇÃO DE ARESTAS ORIENTADAS

Os autores Cheng e Yin (2007), propuseram um algoritmo denominado de Edge Coloring on Directed Graphs (ECDiG), para o escalonamento em redes de sensores, onde cada nó recebe uma lista de intervalos de tempo a serem usados na comunicação *unicast* e *broadcast*. Esse algoritmo colore os arcos de um digrafo para obter um escalonamento *broadcast*. A abordagem de coloração empregada é diferente de outras abordagens apresentadas em trabalhos anteriores, que empregam a coloração de vértices para o escalonamento de nós, sendo também diferente de métodos que usam a coloração de arestas em grafos não-dirigidos para o escalonamento de enlaces. Além disso, esse algoritmo usa o menor número de intervalos de tempo em comparação com outros métodos de coloração, e evita o problema do terminal exposto, assim como, o problema do terminal oculto na comunicação *unicast* e *broadcast*.

A atribuição de um intervalo de tempo distinto para cada nó, evita os problemas decorrentes das colisões, porém, são consumidos muitos intervalos de tempo no escalonamento desses nós. De forma similar, se cada enlace receber um intervalo de tempo exclusivo, as colisões também podem ser evitadas, mas tem o mesmo problema de consumir muitos intervalos de tempo. Para evitar o desperdício de intervalos de tempo, são usadas técnicas de coloração de vértices, para o escalonamento de nós, e técnicas de coloração de arestas, para o escalonamento de enlaces. Desse modo, um escalonamento com o uso de coloração, pode diminuir o número de intervalos de tempo usados.

A coloração de vértices é indicada para o escalonamento *broadcast*, e a coloração de arestas é boa para o escalonamento de enlaces. Em redes de sensores os dois modos de transmissão podem ser intercalados. O algoritmo ECDiG visa atender algumas necessidades em redes de sensores. A seguir, são apresentadas essas necessidades.

a) Usar um número mínimo de intervalos de tempo, para diminuir o tempo de rotação de cada nó;

b) Oferecer condições para o tráfego *unicast* e *broadcast*;

c) Evitar o problema do terminal oculto e o problema do terminal exposto em toda a rede;

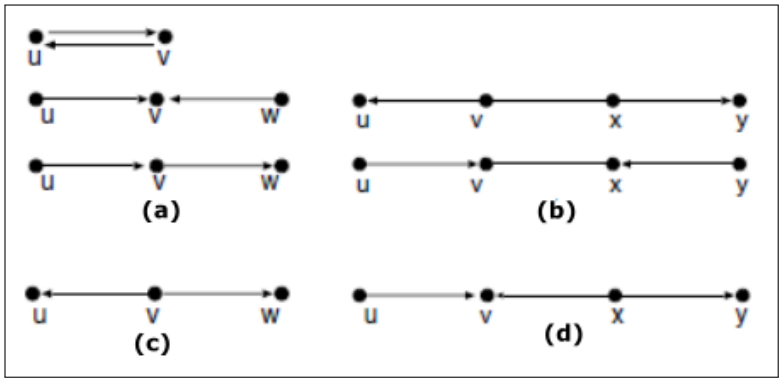
A verificação de conflitos visa eliminar o problema do terminal exposto e o problema do terminal oculto. Para dois arcos estarem em conflitos entre si, as seguintes condições devem ser satisfeitas:

a) Dois arcos compartilham um mesmo vértice e pelo menos um desses arcos é um arco de entrada desse vértice;

b) Um par de arcos não adjacentes estão em conflitos entre si, se houver um terceiro arco que compartilhe a cabeça com um arco e compartilhe a cauda com o outro;

Um par de arcos não são conflitantes, se compartilham somente a cauda, como ilustrado na Figura 18(c). Na Figura 18(b), os arcos disjuntos não são conflitantes, pois a transmissão de um arco não será ouvida pelo destino do outro arco. A Figura 18(a) e a Figura 18(d), representam as situações de conflitos em que dois arcos são conflitantes. O trabalho define arcos de distância 2 e arcos de distância 1. Os arcos de distância 1, são os arcos que incidem no mesmo vértice. Já os arcos de distância 2, são arcos que incidem sobre os vizinhos. Além disso, os arcos conflitantes e não conflitantes podem ser de ambas as distâncias.

Figura 18 – Critérios de verificação de conflitos no algoritmo ECDiG.



Fonte – Adaptado de (CHENG; YIN, 2007)

3.5 ESCALONAMENTO DE ENLACE CONSCIENTE DE ATRASO TDMA PARA REDES SEM FIO DE MÚLTIPLOS SALTOS

Os autores Djukic e Valaee (2009), propuseram um método para encontrar intervalos de tempo TDMA livres de conflitos com atraso mínimo de escalonamento em topologias de

redes de malha. O atraso de escalonamento pode ser visto como um custo em relação à ordem de transmissão dos enlaces obtidos em um ciclo no grafo de conflito. Dessa forma, uma otimização é usada para encontrar uma ordem de transmissão com atraso *min-max* em um conjunto de múltiplos caminhos. Além disso, foi apresentado um algoritmo que encontra uma ordem de transmissão com o mínimo de atraso em topologias de árvores de sobreposição através de uma versão modificada do algoritmo de Bellman–Ford, para encontrar intervalos de tempo de atraso mínimo em tempo polinomial.

O trabalho abordou o problema de escalonamento consciente de atraso, ou seja, dada uma atribuição de largura de banda de enlace, encontre um escalonamento TDMA de comprimento mínimo que também minimiza o atraso de ponta a ponta. O problema de escalonamento de enlace consciente de atraso é resolvido em duas etapas. Na primeira etapa, é desenvolvida uma classe de algoritmos que encontram intervalos de tempo TDMA livres de conflitos e uma ordem relativa de transmissão dos enlaces. Na segunda parte, mostra-se que a ordem de transmissão define um atraso de escalonamento de ponta a ponta. Assim, possibilita a formulação de um programa linear inteiro 0 e 1 que encontre um atraso *min-max* para um subconjunto de caminhos na rede. Nesse programa, o número de variáveis binárias é igual ao número de conflitos na rede, e coincide com a ordem de transmissão dos enlaces. A ordem relativa dos enlaces é usada em conjunto com o grafo de conflito no algoritmo de Bellman–Ford modificado, que pode encontrar um escalonamento viável em tempo polinomial.

As condições para um escalonamento sem conflitos são apresentadas através de um conjunto de desigualdades lineares. Essas desigualdades livres de conflitos correspondem a conflitos emparelhados na rede, onde cada desigualdade é determinada pela duração de transmissão dos enlaces no conflito, tempos de ativação dos enlaces no conflito e ordem de transmissão dos enlaces. Dessa forma, se as ordens de transmissão são fixas, as desigualdades podem ser resolvidas em tempo polinomial usando o algoritmo de Bellman–Ford.

A rede é modelada como um digrafo $G(V,E)$, onde $V = \{v_1, ..., v_n\}$ é o conjunto de nós, e $E = \{e_1, ..., e_m\}$ representa o conjunto de enlaces. Um protocolo de roteamento estabelece as rotas entre os nós, e os caminhos formam uma árvore de sobreposição para um subconjunto de enlaces. No modelo TDMA adotado, o tempo é divido em intervalos de tempo de duração fixa e agrupados em quadros. A duração de cada intervalo de tempo é de T_s segundos, sendo que existem N_f intervalos de tempo em cada quadro. Assim, um quadro tem duração de $T_f = N_f T_s$ segundos, onde N_c intervalos de tempo são reservados para o controle de tráfego e N_d intervalos de tempo são reservados para o tráfego de dados. Para um enlace e_j, Δ_j é o número de intervalos

alocados para esse enlace em cada quadro e $d_j = T_s\Delta_j$ indica o tempo de transmissão desse enlace no quadro.

O problema de escalonamento TDMA foi abordado em princípio sem considerar a existência de intervalos de tempo de controle e restrições relacionadas com o escalonamento. As restrições asseguram que um conjunto de taxas de enlace possam ser escalonadas em um quadro TDMA com N intervalos de tempo. Os intervalos de tempo são atribuídos de forma contínua, onde σ_j representa o intervalo de tempo de ativação de um enlace e_j e Δ_j os intervalos de tempo contínuos para as transmissões desse enlace. Os tempos de ativação da rede, são definidos por: $\sigma = [\sigma_1, ..., \sigma_m]$. O tempo de ativação σ_i de um enlace e_i vai se repetindo conforme a repetição de um quadro TDMA. Assim, os tempos de ativação de um enlace e_i, podem ser encontrados pela soma de σ_i com números múltiplos de N.

Para uma ordem de transmissão fixa, mostrou-se que o escalonamento pode ser encontrado em tempo polinomial com o uso do algoritmo Bellman-Ford. O algoritmo adiciona um vértice extra s no grafo de conflito e conecta esse vértice em cada um dos outros vértices com um arco de custo 0. Logo após, cada arco do grafo de conflito é substituído por um novo par de arcos, onde cada arco recebe um valor conforme a sua ordem de transmissão. Por fim, o algoritmo encontra a distância mais curta de s para cada $e_j \in E$. Se o algoritmo Bellman-Ford não encontrar os caminhos mais curtos, é porque as durações dos enlaces não são suportadas com relação à ordem de transmissão, para N intervalos de tempo. Neste caso, a ordem de transmissão pode ser alterada, ou o número de intervalos de tempo pode ser incrementado para suportar as durações dos enlaces.

Um algoritmo denominado de MINIMIZE-SCHEDULE-LENGTH encontra o número mínimo de intervalos de tempo suficientes para escalonar os enlaces da rede e reduz as taxas de enlace com o intuito de compensar o número de intervalos de tempo no quadro. A redução nas taxas de enlace é semelhante com o aumento no tamanho do quadro. A vantagem é que não existe impacto em relação ao atraso, ou o fato de protocolos MAC TDMA terem quadros de tamanho fixo.

O atraso de transmissão ocorre quando um enlace de saída em um nó transmite antes de um enlace de entrada no caminho em direção ao nó destino. Um algoritmo denominado de Algorithm-MM, representa um refinamento do algoritmo MINIMIZE-SCHEDULE-LENGTH. Esse algoritmo usa uma otimização *max-min* que encontra o número de intervalos de tempo suficientes, para que todos os enlaces sejam escalonados. Além disso, obtém uma ordem de transmissão com a propriedade de atraso de escalonamento *min-max*.

Um algoritmo de tempo polinomial foi desenvolvido para encontrar intervalos de tempo com atraso de escalonamento máximo de um quadro em caminhos de ida e volta nas topologias de árvores de sobreposição. Esse algoritmo encontra uma ordem de transmissão com atraso de escalonamento de ida e volta de um quadro em todos os caminhos, atribuindo uma classificação para cada enlace, representando a ordem de transmissão escolhida.

3.6 OUTROS TRABALHOS

Os autores Ramanathan e Lloyd (1993), propuseram algoritmos para o escalonamento *broadcast* e escalonamento de enlaces. As redes foram modeladas usando duas classes restritas de grafos: árvores e grafos planos. Os algoritmos propostos apresentaram bons resultados teóricos e experimentais. Também foi demonstrado que as redes de árvores podem ser escalonadas de forma ótima.

O trabalho de Wang *et al.* (2006), abordou o escalonamento de enlaces em redes sem fio para maximizar a taxa de transferência na rede. O trabalho assumiu que os nós poderiam ter diferentes faixas de transmissão e diferentes faixas de interferência. O modelo adotado também torna possível que um enlace de comunicação possa não existir por causa de barreiras ou porque não é usado por um protocolo de roteamento. Por meio de uma formulação matemática, foi possível encontrar um escalonamento TDMA que otimiza a taxa de transferência na rede. Esse trabalho usou o modelo de interferência RTS/CTS e o modelo de interferência de protocolo com potência de transmissão fixa. Foram propostos algoritmos centralizados e distribuídos para ambos os modelos.

Os autores Ma *et al.* (2009), trataram do escalonamento de sono em redes de sensores para reduzir o consumo de energia, motivado pelo desperdício de energia causado pelo estado de escuta ociosa dos nós. No escalonamento de sono tradicional, os nós precisam iniciar várias vezes, sendo que essas transições de estado leva a um consumo extra de energia. Esse trabalho projetou um escalonamento de sono para as redes de sensores com baixa taxa de dados. Foi usado a técnica TDMA na camada de enlace devido as vantagens de evitar colisões e escuta ociosa, com a proposta de um problema de escalonamento de sono TDMA livre de interferências. Nesse problema, os nós recebem intervalos de tempo consecutivos para reduzir a frequência de transições de estado. Para resolver esse problema foram propostos algoritmos centralizados e distribuídos.

4 ALGORITMOS PROPOSTOS

A proposta desta dissertação é desenvolver algoritmos de coloração BPRN. Os algoritmos propostos, denominados de Bsatur e Bswap, são inspirados no processo de coloração dos vértices de um grafo adotado pelo algoritmo Dsatur. Esses algoritmos usam os conceitos de grau e grau de saturação de um vértice, vistos no algoritmo Dsatur, mas relacionados aos arcos de um digrafo *backbone*. Assim, o grau e o grau de saturação de um vértice são interpretados como o grau de conflitos e o grau de saturação de um arco, respectivamente. O grau de conflitos de um arco, representa o número de arcos que causam conflitos primários ou secundários a esse arco. Já o grau de saturação de um arco, corresponde ao número de arcos coloridos que conflitam com esse arco.

O Bsatur colore os arcos de um digrafo *backbone* seguindo critérios semelhantes aos adotados pelo algoritmo Dsatur na coloração dos vértices de um grafo. O algoritmo Bswap segue uma coloração semelhante com a realizada no algoritmo Bsatur, mas difere no uso de um mecanismo de reutilização de cores, visto mais adiante. Este trabalho ainda propõe uma modelagem para resolver o problema de coloração BPRN. As definições propostas são usadas como base na implementação dos algoritmos propostos. Essas definições possibilitam a extração de informações importantes associadas a um arco, como o grau de conflitos e grau de saturação, além de permitirem a verificação de conflitos primários e secundários entre um arco a ser colorido e seus arcos conflitantes.

4.1 DEFINIÇÕES

Os conflitos primários e secundários associados a um arco estão entre as informações mais relevantes em uma coloração BPRN. Pois a partir dessas informações, uma cor pode ser escolhida para colorir um arco sem a ocorrência de conflitos com arcos coloridos e conflitantes a esse arco. Assim, a cor escolhida para colorir um arco deve ser diferente das cores de seus arcos conflitantes. Os arcos conflitantes de um arco são os arcos que causam conflitos primários ou secundários a ele. Seja $D = (V(D), A(D))$ um digrafo e $B = (V(B), A(B))$ um digrafo *backbone* de D, com $V(B) = V(D)$ e $A(B) \subseteq A(D)$. O conjunto $P(uv)$ de arcos que causam conflitos primários a um arco uv, pode ser definido da seguinte forma:

$$P_{in}(x) = \{ax \mid ax \in A(B)\}. \tag{4.1}$$

$$P_{out}(x) = \{xa \mid xa \in A(B)\}. \tag{4.2}$$

$$P(uv) = \{P_{in}(u) \cup P_{out}(u) \cup P_{in}(v) \cup P_{out}(v)\} - uv. \tag{4.3}$$

De forma semelhante, seja $S(uv)$ o conjunto de arcos que causam conflitos secundários com um arco uv. Esse conjunto pode ser definido da seguinte forma:

$$S_{out}(uv) = \{ib \mid ub \in A(D),\ ib \in A(B),\ b \neq v,\ i \neq u\ e\ i \neq v\}. \tag{4.4}$$

$$S_{in}(uv) = \{ai \mid av \in A(D),\ ai \in A(B),\ a \neq u,\ i \neq u\ e\ i \neq v\}. \tag{4.5}$$

$$S(uv) = \{S_{out}(uv)\ \cup\ S_{in}(uv)\}. \tag{4.6}$$

Dessa forma, o conjunto de arcos conflitantes de um arco uv é definido como:

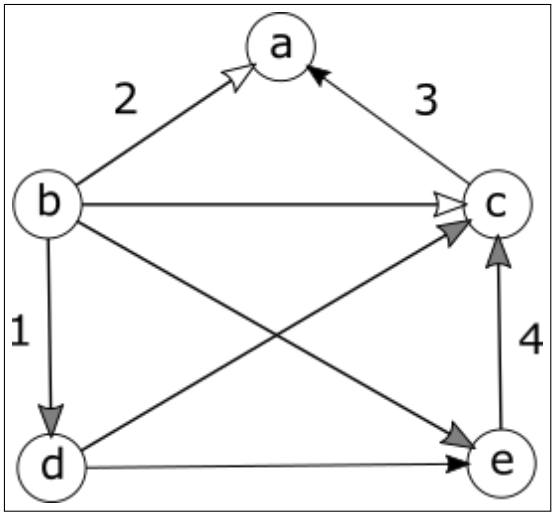
$$C(uv) = P(uv) \cup S(uv). \tag{4.7}$$

As informações de grau de conflitos e grau de saturação de um arco serão usadas nos algoritmos Bsatur e Bswap, para indicar a ordem de coloração dos arcos. Os arcos saturados de um arco uv são os arcos em $C(uv)$ que estão coloridos. Seja $c(uv)$ a função para retornar a cor de um arco uv, tal que, se um arco uv não estiver colorido, tem-se $c(uv) = 0$. Dessa forma, o conjunto $SATURADOS(uv)$ de arcos saturados de um arco uv é definido como:

$$SATURADOS(uv) = \{ab \mid ab \in C(uv)\ e\ c(ab) \neq 0\}. \tag{4.8}$$

O grau de conflitos de um arco uv será denotado por $GRAU(uv) = |C(uv)|$. Já o grau de saturação é definido como $SATUR(uv) = |SATURADOS(uv)|$. A Figura 19 ilustra o conjunto de arcos saturados do arco de, como sendo os arcos: ba, bd e ec.

Figura 19 – Exemplo da saturação de um arco.



Fonte – Elaborado pelo autor

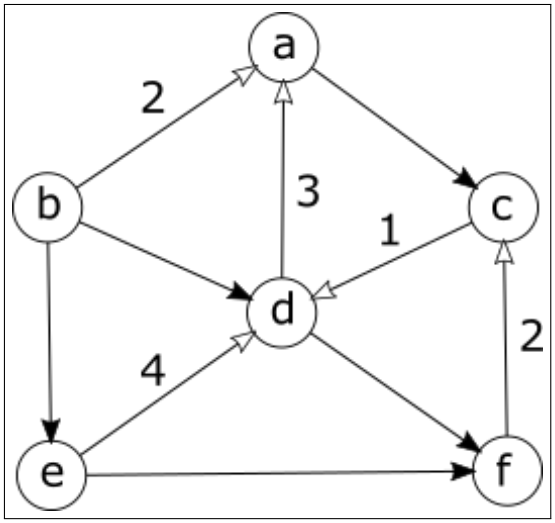
Após a identificação dos arcos saturados de um arco *uv* a ser colorido, a próxima etapa consiste em verificar a menor cor livre, no intervalo de 1 a *m*, que pode ser usada para colorir o arco *uv* sem causar conflitos primários ou secundários com os arcos saturados desse arco, onde *m* é o número de arcos do digrafo *backbone*. Assim, essa cor mínima pode ser usada para colorir o arco *uv*, se ela não pertencer ao conjunto $CORES(uv)$, ou seja, $c(uv) \notin CORES(uv)$.

$$CORES(uv) = \{c(ab) \mid ab \in SATURADOS(uv)\}. \tag{4.9}$$

4.2 ALGORITMO BSATUR

O algoritmo Bsatur representa uma versão adaptada do algoritmo Dsatur para a coloração BPRN. Dessa forma, esse algoritmo começa atribuindo a cor 1 para o arco de maior grau de conflitos no digrafo *backbone*. Enquanto existir arcos não coloridos no digrafo *backbone*, escolhe-se o arco de maior grau de saturação e a cor mínima livre é atribuída a esse arco. Em caso de empate, o arco de maior grau de conflitos entre esses arcos com o mesmo grau de saturação é selecionado. Se o empate permanecer, a escolha é feita de forma arbitrária. A Figura 20 ilustra uma coloração BPRN com o algoritmo Bsatur, onde os arcos são coloridos na seguinte ordem: cd, ba, da, ed e fc.

Figura 20 – Uma coloração BPRN com o algoritmo Bsatur.



Fonte – Elaborado pelo autor

Após o Bsatur selecionar um arco *uv* para ser colorido e escolher uma cor *c* mínima livre, a próxima etapa consiste em colorir esse arco. Na coloração de *uv*, as matrizes NOTIN e NOTOUT serão atualizadas para indicar que outros arcos de entrada ou saída dos vértices *u* e *v* não podem mais receber a cor *c*. As matrizes NOTSIN e NOTSOUT são atualizadas para evitar futuros conflitos secundários ao colorir um arco. Assim, a matriz NOTSIN indica que os arcos de entrada em um vértice *b*, com *b* vizinho de *u*, não podem mais ser coloridos com a cor *c*. De forma semelhante, a matriz NOTSOUT indica que os arcos de saída de um vértice *a*, com *v* vizinho de *a*, não podem mais receber a cor *c*. Essas quatro matrizes têm o mesmo significado das matrizes apresentadas na seção 3.1.

A matriz denominada de CADJ tem a função de guardar as cores dos arcos. A matriz NUMSIN indica o número de arcos coloridos que saem de *u* e causam conflitos secundários com arcos de entrada em *b*. De forma semelhante, a matriz NUMSOUT registra o número de arcos coloridos que chegam em *v* e causam conflitos secundários com arcos de saída em *a*. A variável *mcor* representa o número de cores usadas em uma coloração do digrafo *backbone*. O Algoritmo 4 apresenta o procedimento de coloração de um arco *uv*. Esse algoritmo denominado de COLORIR_ARCO, representa uma ampliação da etapa responsável por colorir um arco no Algoritmo 1, proposto no trabalho de Huson (1995), para a coloração de grafo de pontos lineares.

Algoritmo 4: Algoritmo para Colorir um Arco

```
Entrada: Digrafo D, digrafo backbone B, arco uv e cor c;
início
    B.NOTOUT[u][c] ← 0;
    B.NOTIN[v][c] ← 0;
    B.CADJ[u][v] ← c;
    se c > B.mcor então
        B.mcor ← c;
    fim
    para cada vértice i ∈ V(B) faça
        se ui ∈ A(D) então
            B.NOTSIN[i][c] ← 0;
            B.NUMSIN[i][c] ← B.NUMSIN[i][c] + 1;
        fim
        se iv ∈ A(D) então
            B.NOTSOUT[i][c] ← 0;
            B.NUMSOUT[i][c] ← B.NUMSOUT[i][c] + 1;
        fim
    fim
fim
```

No Algoritmo 4, as atribuições acontecem em tempo constante. No entanto, após colorir um arco uv com uma cor c, é preciso atualizar as matrizes NOTSIN, NOTSOUT, NUMSIN e NUMSOUT, para evitar futuros conflitos primários e secundários ao colorir outro arco. O tempo exigido na execução desse algoritmo é $O(n)$, onde n representa o número de vértices de um digrafo *backbone*.

A fase seguinte ao colorir um arco uv, é responsável por incrementar o grau de saturação dos arcos conflitantes de uv, que não estão coloridos. Essa fase utiliza duas matrizes denominadas de SAT e TABR, onde SAT tem a finalidade de armazenar o grau de saturação de um arco, e a matriz TABR é usada para que um arco não tenha o seu grau de saturação incrementado mais de uma vez em relação ao arco uv. O Algoritmo 5 representa essa fase, e será denominado de ATUALIZAR_SATURACAO_ARCO.

Algoritmo 5: Algoritmo para Atualizar a Saturação de Arcos Conflitantes Não Coloridos

```
Entrada: Digrafo D, digrafo backbone B e arco uv;
início
    para cada vértice i ∈ V(B) faça
        se iu ∈ A(B) ∧ c(iu) = 0 ∧ B.TABR[i][u] então
            B.SAT[i][u] ← B.SAT[i][u] + 1;
            B.TABR[i][u] ← 0;
        fim
        se i ≠ v ∧ ui ∈ A(B) ∧ c(ui) = 0 então
            B.SAT[u][i] ← B.SAT[u][i] + 1;
        fim
        se i ≠ u ∧ iv ∈ A(B) ∧ c(iv) = 0 então
            B.SAT[i][v] ← B.SAT[i][v] + 1;
        fim
        se vi ∈ A(B) ∧ c(vi) = 0 ∧ B.TABR[v][i] então
            B.SAT[v][i] ← B.SAT[v][i] + 1;
            B.TABR[v][i] ← 0;
        fim
    fim
    para cada vértice b ∈ V(B) faça
        se b ≠ v ∧ ub ∈ A(D) então
            para cada vértice i ∈ V(B) faça
                se i ≠ u ∧ i ≠ v ∧ ib ∈ A(B) ∧ c(ib) = 0 então
                    B.SAT[i][b] ← B.SAT[i][b] + 1;
                    B.TABR[i][b] ← 0;
                fim
            fim
        fim
    fim
    para cada vértice a ∈ V(B) faça
        se a ≠ u ∧ av ∈ A(D) então
            para cada vértice i ∈ V(B) faça
                se i ≠ u ∧ i ≠ v ∧ ai ∈ A(B) ∧ c(ai) = 0 ∧ B.TABR[a][i] então
                    B.SAT[a][i] ← B.SAT[a][i] + 1;
                fim
            fim
        fim
    fim
fim
```

A complexidade para atualizar o grau de saturação dos arcos conflitantes primários de um arco uv é $O(n)$. Já a complexidade para atualizar o grau de saturação dos arcos que causam conflitos secundários com uv é $O(n^2)$, pois é preciso atualizar o grau de saturação dos arcos que entram em b, e dos arcos que saem de a, com v vizinho de a e b vizinho de u. Assim, a complexidade do Algoritmo 5 responsável por atualizar o grau de saturação dos arcos conflitantes

de um arco é $O(n^2)$.

Uma etapa do algoritmo Bsatur consiste em encontrar a menor cor livre para colorir um arco *uv*. Quando uma cor *c* no intervalo de 1 a *m* é selecionada, verifica-se através do Algoritmo 6, se essa cor é diferente das cores dos arcos saturados de uv. Em caso positivo, ela será usada para colorir uv, onde *m* representa o número de arcos de um digrafo *backbone*. Esse algoritmo verifica se o vértice *u* não possui um arco de entrada ou saída com a cor *c*, onde o valor de NOTIN[u][c] é igual a 1, quando não existir um arco de entrada em *u* com a cor *c*. Caso contrário, o valor de NOTIN[u][c] é igual a 0. Da mesma forma, o valor de NOTOUT[u][c] é igual a 1 quando não existir um arco de saída em *u* com a cor *c*. No entanto, se existir um arco de saída em *u* com a cor *c*, o valor de NOTOUT[u][c] é igual a 0. Esse mesmo entendimento é válido para os valores de NOTIN[v][c] e NOTOUT[v][c]. Além disso, os valores de NOTSIN[v][c] e NOTSOUT[u][c] são verificados para que a atribuição da cor *c* ao arco *uv* não cause conflitos secundários com os arcos saturados de *uv*.

Quando o valor de NOTSIN[v][c] é igual a 1, é porque não existe um arco de saída em um vértice *a* com a cor *c*, onde *v* é vizinho de *a*. Se o valor de NOTSIN[v][c] é igual a 0, então existe um arco de saída em *a* com a cor *c*. O valor de NOTSOUT[u][c] é igual a 1, quando não existir um arco de entrada em um vértice *b* com a cor *c*, com *b* vizinho de *u*. Caso contrário, o valor de NOTSOUT[u][c] é igual a 0. O Algoritmo 6 denominado de NAO_CONFLITO e com um tempo de execução constante, representa uma parte do Algoritmo 1, onde retorna 1, se a cor c pode ser atribuída para *uv*, sem causar conflitos com os arcos saturados desse arco e, 0, caso contrário.

Algoritmo 6: Algoritmo para Verificar Conflitos Primários e Secundários

Entrada: Digrafo *backbone B*, arco *uv* e cor *c*;
Saída: Retorna 1, se a cor não causa conflito e, 0, caso contrário;
início
  se *B.NOTIN[u][c]* $\wedge$ *B.NOTOUT[u][c]* $\wedge$ *B.NOTIN[v][c]* $\wedge$ *B.NOTOUT[v][c]* $\wedge$ *B.NOTSOUT[u][c]* $\wedge$ *B.NOTSIN[v][c]* **então**
    retorna *1*;
  fim
  retorna *0*;
fim

O Algoritmo 7 usado para selecionar um arco a ser colorido, baseia-se nas infor-

mações de grau de conflitos e grau de saturação de um arco. Se existirem arcos com o mesmo grau de saturação, o arco selecionado entre esses arcos, é o que tem o maior grau de conflitos. Se mesmo assim, o empate persistir, escolhe-se o primeiro arco visitado entre esses arcos com o maior grau de conflitos. O Algoritmo 7 será chamado em outro algoritmo como ARCO_MAIOR_SATURACAO.

Algoritmo 7: Algoritmo para Selecionar o Arco a Ser Colorido

```
Entrada: Digrafo backbone B;
Saída: Retorna o arco a ser colorido;
início
    saturacaoAtual ← 0;
    maiorSaturacao ← −∞;
    arcoSelecionado ← ∅;
    para cada arco uv ∈ A(B) faça
        se c(uv) = 0 então
            saturacaoAtual ← B.SAT[u][v];
            se saturacaoAtual > 0 então
                se maiorSaturacao < saturacaoAtual) então
                    arcoSelecionado ← uv;
                    maiorSaturacao ← saturacaoAtual;
                fim
                se uv ≠ arcoSelecionado ∧ maiorSaturacao = saturacaoAtual então
                    se uv.grauConflito > arcoSelecionado.grauConflito então
                        arcoSelecionado ← uv;
                    fim
                fim
            fim
        fim
    fim
    retorna arcoSelecionado;
fim
```

O intuito do Algoritmo 7, é selecionar um arco uv a ser colorido. Assim, esse algoritmo visita cada arco $uv \in A(B)$, tal que, se o arco visitado não estiver colorido, a sua saturação é encontrada através de $SAT[u][v]$ e comparada com a maior saturação encontrada até

o momento. Caso a maior saturação seja menor do que a saturação atual, a maior saturação receberá o valor da saturação atual, e o arco selecionado passará a ser o arco *uv*. Como o laço é realizado m vezes, o tempo de execução desse algoritmo é $O(m)$.

A primeira etapa em ambos os algoritmos propostos, é a coloração do arco de maior grau de conflitos com a cor 1. Além disso, se existirem arcos com o maior grau de conflitos, o primeiro dentre esses arcos identificado no processo de pesquisa, será o arco escolhido para receber a cor 1. O Algoritmo 8 é o procedimento usado para colorir um arco de maior grau de conflitos com a cor 1. Esse algoritmo será denominado de COLORIR_PRIMEIRO_ARCO ao ser chamado nos algoritmos Bsatur e Bswap. Após a explicação de seus principais procedimentos, o algoritmo Bsatur será apresentado através do Algoritmo 9.

Algoritmo 8: Algoritmo para Colorir o Primeiro Arco

```
Entrada: Digrafo D e digrafo backbone B;
início
    maiorGrauConflito ← −∞;
    arcoSelecionado ← ∅;
    para cada arco uv ∈ A(B) faça
        se maiorGrauConflito < uv.grauConflito então
            arcoSelecionado ← uv;
            maiorGrauConflito ← uv.grauConflito;
        fim
    fim
    COLORIR_ARCO(D, B, arcoSelecionado, 1);
    ATUALIZAR_SATURACAO_ARCO(D, B, arcoSelecionado);
fim
```

O Algoritmo 8 responsável por colorir o primeiro arco, representa uma função no algoritmo Bsatur através da chamada COLORIR_PRIMEIRO_ARCO(D, B). O papel desse algoritmo consiste em selecionar e colorir o arco de maior grau de conflitos. O tempo requerido para encontrar esse arco é $O(m)$. Já foi apresentado que o tempo de execução do Algoritmo 4, correspondente à chamada COLORIR_ARCO(D, B, arcoSelecionado, 1) é $O(n)$. Também já foi demonstrado que o Algoritmo 5 correspondente à chamada ATUALIZAR_SATURACAO_ARCO(D, B, arcoSelecionado) tem complexidade $O(n^2)$. Portanto, o tempo de execução do Algoritmo 8 é $O(n^2)$.

Algoritmo 9: Algoritmo Bsatur

```
Entrada: Digrafo D e digrafo backbone B;
Saída: Número de cores usadas na coloração de B;
início
    COLORIR_PRIMEIRO_ARCO(D, B);
    para i = 2 até i ≤ B.m faça
        uv ← ARCO_MAIOR_SATURACAO(B);
        para cor = 1 até cor ≤ B.m faça
            se NAO_CONFLITO(B, uv, cor) então
                corSelecionada = cor;
                pausa;
            fim
        fim
        COLORIR_ARCO(D, B, uv, corSelecionada);
        ATUALIZAR_SATURACAO_ARCO(D, B, uv);
    fim
    retorna B.mcor;
fim
```

No algoritmo Bsatur, a chamada COLORIR_PRIMEIRO_ARCO(D, B) executa no tempo $O(n^2)$, como já havia sido demonstrado. Também já foi apresentado, que a chamada ARCO_MAIOR_SATURACAO(B), correspondente ao Algoritmo 7, consome o tempo $O(m)$. O laço mais interno desse algoritmo e, o pseudocódigo interno a esse laço, têm a finalidade de encontrar uma cor mínima livre no intervalo de 1 a m para colorir um arco uv. O tempo de execução para escolher uma cor e colorir um arco uv é $O(n+m)$, pois foi visto que a chamada NAO_CONFLITO(B, uv, cor) executa em tempo constante, e que a chamada COLORIR_ARCO(D, B, uv, cor) consome o tempo $O(n)$. Como a chamada ATUALIZAR_SATURACAO_ARCO(D, B, uv) tem complexidade $O(n^2)$ e o laço mais externo executa $m-1$ vezes, o tempo de execução do algoritmo Bsatur é $O(n^2m)$.

4.3 ALGORITMO BSWAP

O algoritmo Bswap representa uma ampliação do algoritmo Bsatur, por causa de um mecanismo de troca, que possibilita a oportunidade de reutilização de cores durante a coloração do digrafo *backbone*. O Bswap surgiu da observação de que no algoritmo Bsatur a coloração vai

acontecendo em subdigrafos disjuntos no processo de coloração dos arcos do digrafo *backbone*. A separação espacial desses subdigrafos que estão sendo coloridos, possibilita o reuso de cores de arcos saturados de um arco a ser colorido, uma vez que as cores desses arcos estão distribuídas no intervalo de 1 a k, com k representando o número de cores usadas em uma coloração parcial do digrafo *backbone*.

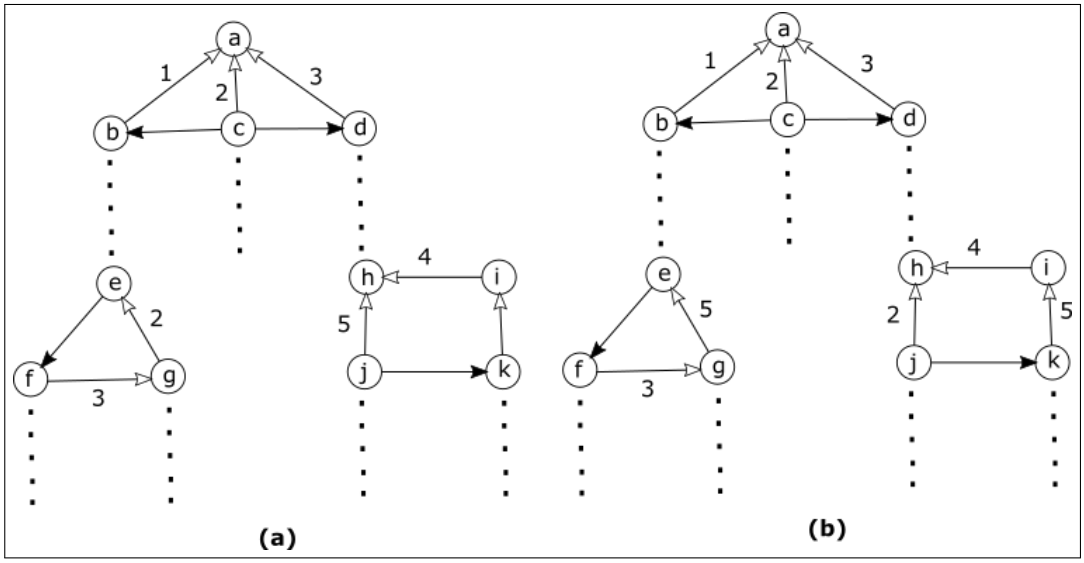
O mecanismo de troca é chamado quando a menor cor livre para colorir um arco uv é igual a $k+1$. Esse mecanismo tem o objetivo de reutilizar a cor de um arco do conjunto $SATURADOS(uv)$ para colorir um arco uv através da troca de cores entre um arco do conjunto $NAO\_SATURADOS(uv)$, onde o conjunto $NAO\_SATURADOS(uv)$ representa os arcos coloridos que não são arcos saturados de uv.

$$NAO\_SATURADOS(uv) = \{ab \mid ab \in A(B),\ ab \notin SATURADOS(uv)\ e\ c(ab) > 0\}. \quad (4.10)$$

Nesse mecanismo, para cada par de arcos $ab \in SATURADOS(uv)$ e $xy \in NAO\_SATURADOS(uv)$, verifica-se a possibilidade de trocar as cores entre esses dois arcos, com $c(ab) > c(xy)$, e sem a ocorrência de conflitos primários e secundários. Assim, para efetuar uma troca de cores entre esses dois arcos, as condições: $c(ab) \notin CORES(xy)$, $c(xy) \notin CORES(ab)$ e $c(ab) \notin CORES(uv)$ devem ser satisfeitas. Se essas condições forem satisfeitas, a troca de cores entre os arcos ab e xy é feita, com ab sendo colorido com $c(xy)$, xy recebendo a cor $c(ab)$ e uv sendo colorido com $c(ab)$. Se essas condições falharem para todos os pares de arcos ab e xy, o arco uv é colorido com a cor $k+1$.

A Figura 21 ilustra o procedimento de troca em subdigrafos do digrafo *backbone*, com os arcos do digrafo *backbone* possuindo as setas coloridas de branco. Na Figura 21(a), é ilustrado o digrafo *backbone* antes da chamada do mecanismo de troca. Já a Figura 21(b), apresenta o digrafo *backbone* após a execução desse mecanismo, onde o arco ki foi colorido com a cor 5, e os arcos jh e ge foram recoloridos com as cores 2 e 5, respectivamente.

Figura 21 – Exemplo do procedimento de troca em subdigrafos do digrafo *backbone*.



Fonte – Elaborado pelo autor

Um procedimento usado no algoritmo Bswap é o responsável por descolorir um arco. Esse procedimento é requerido quando no processo de coloração de um arco uv, procura-se reutilizar a cor c de um arco ab do conjunto $SATURADOS(uv)$, para colorir o arco uv. O reuso de cores acontece quando é possível trocar as cores entre dois arcos coloridos $ab \in SATURADOS(uv)$ e $xy \in NAO\_SATURADOS(uv)$. Assim, os arcos ab e xy são descoloridos, com o objetivo de verificar se esses arcos podem ser recoloridos com as cores $c(xy)$ e $c(ab)$, respectivamente.

No processo de descoloração de um arco uv, o valor de NOTIN[u][c] vai ser igual a 1, para indicar que o vértice u não mais possui um arco de saída com a cor c. Da mesma forma, o valor de NOTIN[v][c] passará a ser 1, para mostrar que o vértice v deixou de possuir um arco de entrada com a cor c. Já o valor de CADJ[u][v] será igual a 0, para indicar que o arco uv foi descolorido. Além disso, para cada vértice b vizinho de u, o valor de NUMSIN[b][c] é decrementado em uma unidade, com o intuito de mostrar que o arco uv não mais causará conflitos secundários, por causa da cor c com arcos de entrada em b. Da mesma forma, o valor de NUMSOUT[a][c] é decrementado em uma unidade, para indicar que o arco uv não mais causará conflitos secundários, por causa da cor c com arcos de saída em a, onde v é vizinho de a.

O valor de NOTSIN[b][c] será igual a 1, quando não existirem arcos com a cor c, que causem conflitos secundários com arcos de entrada em b. Já o valor de NOTSOUT[a][c] passará a ser 1, quando não existirem arcos com a cor c, que causem conflitos secundários com arcos de saída em a. O Algoritmo 10 apresenta esse procedimento de descoloração de um arco,

sendo denominado de DESCOLORIR_ARCO.

Algoritmo 10: Algoritmo para Descolorir um Arco

```
Entrada: Digrafo D, digrafo backbone B, arco uv e cor c;
início
    B.NOTOUT[u][c] ← 1;
    B.NOTIN[v][c] ← 1;
    B.CADJ[u][v] ← 0;
    para cada vértice i ∈ V(B) faça
        se ui ∈ A(D) então
            B.NUMSIN[i][c] ← B.NUMSIN[i][c] - 1;
            se B.NUMSIN[i][c] = 0 então
                B.NOTSIN[i][c] ← 1;
            fim
        fim
        se iv ∈ A(D) então
            B.NUMSOUT[i][c] ← B.NUMSOUT[i][c] - 1;
            se B.NUMSOUT[i][c] = 0 então
                B.NOTSOUT[i][c] ← 1;
            fim
        fim
    fim
fim
```

No algoritmo Bswap, quando uma cor c para colorir um arco uv é igual a $k+1$, o procedimento de troca é invocado. Esse mecanismo verifica se existe uma par de arcos $ab \in SATURADOS(uv)$ e $xy \in NAO\_SATURADOS(uv)$, com $c(ab) > c(xy)$. Após isso, esse mecanismo descolore os arcos ab e xy. Em seguida, verifica se é possível recolorir o arco ab com a cor $c(xy)$ e alterar a cor do arco xy para a cor $c(ab)$. Além disso, ainda é preciso verificar se o arco uv pode receber a cor $c(ab)$. Essas verificações têm por objetivo prevenir a ocorrências de conflitos primários e secundários. Se essas três condições forem positivas, o arco uv será colorido com a cor $c(ab)$ e, posteriormente, os arcos ab e xy serão recoloridos. O Algoritmo 11 apresenta esse mecanismo de troca, sendo denominado de TROCAR_COR_ARCO.

Algoritmo 11: Algoritmo para Troca de Cores

```
Entrada: Digrafo D, digrafo backbone B, arcos saturados e não saturados de um arco uv;
Saída: Retorna 1, se uma troca foi realizada, e 0 caso contrário;
início
    para cada arco xy ∈ NAO_SATURADOS(uv) faça
        para cada arco ab ∈ SATURADOS(uv) faça
            se c(ab) > c(xy) então
                DESCOLORIR_ARCO(D, B, ab, c(ab));
                DESCOLORIR_ARCO(D, B, xy, c(xy));
                se NAO_CONFLITO(B, ab, c(xy)) ∧ NAO_CONFLITO(B, xy, c(ab)) ∧
                 NAO_CONFLITO(B, uv, c(ab)) então
                    COLORIR_ARCO(D, B, uv, c(ab));
                    COLORIR_ARCO(D, B, ab, c(xy));
                    COLORIR_ARCO(D, B, xy, c(ab));
                    retorna 1;
                fim
                senão
                    COLORIR_ARCO(D, B, ab, c(ab));
                    COLORIR_ARCO(D, B, xy, c(xy));
                fim
            fim
        fim
    fim
    retorna 0;
fim
```

O Algoritmo 11 recebe como argumentos: o digrafo, o digrafo *backbone*, o conjunto de arcos saturados e o conjunto de arcos não saturados de um arco *uv* a ser colorido. O laço externo desse algoritmo percorre o conjunto de arcos não saturados, e o laço interno visita o conjunto de arcos saturados. Para cada par de arcos *ab* e *xy*, sendo *ab* pertencente ao conjunto de arcos saturados e *xy* contido no conjunto de arcos não saturados, objetiva-se trocar as cores entre esse par de arcos, e colorir o arco *uv* com a cor do arco *ab*. Na análise do tempo de execução desse algoritmo será considerado o caso em que o número de arcos saturados é igual a $m/2$ e o número de arcos não saturados é igual a $m/2-1$. Além disso, as cores dos arcos saturados são maiores do que as cores do arcos não saturados. Assim, esse mecanismo responsável pela troca

e reutilização de cores executa no tempo $O(m^2n)$.

Algoritmo 12: Algoritmo Bswap

```
Entrada: Digrafo D e digrafo backbone B;
Saída: Número de cores usadas na coloração de B;
início
    COLORIR_PRIMEIRO_ARCO(D, B);
    para i = 2 até i ≤ B.m faça
        uv ← ARCO_MAIOR_SATURACAO(B);
        para cor = 1 até cor ≤ B.m faça
            se NAO_CONFLITO(B, uv, cor) então
                corSelecionada = cor;
                pausa;
            fim
        fim
        se corSelecionada = B.mcor + 1 então
            se TROCAR_COR_ARCO(D, B, SATURADOS(uv), NAO_SATURADOS(uv),
              uv) = 0 então
                COLORIR_ARCO(D, B, uv, corSelecionada);
            fim
            ATUALIZAR_SATURACAO_ARCO(D, B, uv);
        fim
        senão
            COLORIR_ARCO(D, B, uv, corSelecionada);
            ATUALIZAR_SATURACAO_ARCO(D, B, uv);
        fim
    fim
    retorna B.mcor;
fim
```

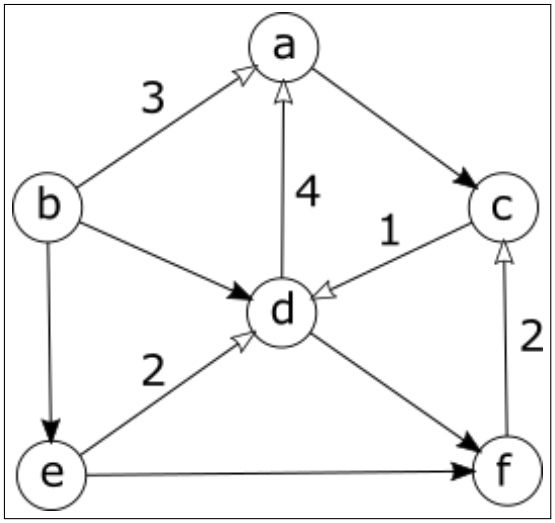
O Algoritmo 12 representando o algoritmo Bswap tem um tempo de execução maior do que o tempo de execução do algoritmo Bsatur, pois inclui um mecanismo de troca e reutilização de cores no processo de coloração dos arcos do digrafo *backbone*. Foi demonstrado que esse mecanismo correspondente à chamada TROCAR_COR_ARCO tem complexidade $O(m^2n)$. O tempo gasto para selecionar o conjunto de arcos saturados de um arco *uv* a ser colorido através da chamada SATURADOS(uv) é $O(n^2)$, e o tempo gasto na chamada NAO_SATURADOS(uv),

responsável por retornar o conjunto de arcos não saturados de *uv* é $O(m^2)$. A função COLORIR_ARCO que representa o Algoritmo 4 tem complexidade $O(n)$, e a função ATUALIZAR_SATURACAO_ARCO(D, B, uv) correspondente ao Algoritmo 5 tem complexidade $O(n^2)$. O tempo requerido para selecionar um arco com a função ARCO_MAIOR_SATURACAO é $O(m)$. O laço mais externo é executado $m-1$ vezes, pois um arco com o maior grau de conflitos já está colorido. Com essas informações pode-se concluir que o tempo de execução do algoritmo Bswap é $O(m^3n)$.

4.4 ALGORITMOS USADOS PARA COMPARAÇÃO

Nesta dissertação, escolheu-se dois algoritmos de coloração PRN do trabalho de Huson (1995), e adaptados para a coloração BPRN no intuito de comparar os resultados desses algoritmos com os resultados dos algoritmos Bsatur e Bswap. Os dois algoritmos selecionados são denominados de BF (*Breadth First Algorithm*) e DFOr (*Depth First Ordering Algorithm*). O algoritmo BF começa colorindo os arcos de entrada do vértice de maior grau no digrafo *backbone*. Após a coloração desses arcos, os vértices adjacentes a esse vértice, são colocados aleatoriamente em uma fila Q. Assim, enquanto existirem vértices em Q, remove-se um vértice v e colore os seus arcos de entrada, com os vértices adjacentes a v sendo adicionados de forma aleatória nessa fila. Dessa forma, quando essa fila não possuir mais vértices, os arcos do digrafo *backbone* estarão coloridos. Esse algoritmo é semelhante com o algoritmo de busca em largura. A Figura 22 apresenta uma coloração BPRN usando o algoritmo BF, onde os vértices são visitados na seguinte ordem: d, e, a, c, b e f.

Figura 22 – Uma coloração BPRN com o algoritmo BF.



Fonte – Elaborado pelo autor

O algoritmo BF gerencia um vetor que indica as cores dos vértices, sendo denominado de *cor*. Um vértice visitado é colocado em uma fila e sua cor passa a ser cinza. Pois no início desse algoritmo todos os vértices recebem a cor branca. A mudança de cor tem a finalidade de indicar que um vértice com a cor cinza já foi visitado e, portanto, não precisa ser visitado novamente. Uma fila Q adotada por esse algoritmo tem como objetivo estabelecer a ordem com que os vértices terão os seus arcos de entrada coloridos, ou seja, um vértice terá os seus arcos de entrada coloridos conforme a sua ordem de descoberta.

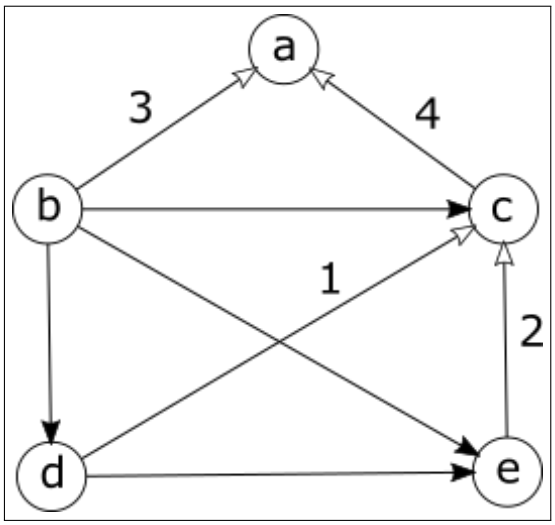
Após retirar um vértice u do início de Q, a etapa seguinte consiste em colorir os seus arcos de entrada. Em seguida, para cada arco vu de entrada em u, a menor cor livre no intervalo de 1 a m é selecionada para colorir esse arco. O tempo gasto para colorir os arcos de entrada de u é $O(n(n+m)) = O(nm)$. A função EMBARALHAR_VERTICES_ADJACENTES tem complexidade linear em termos do número de vértices adjacentes a um vértice. Essa função representa uma função denominada de "random_shuffle" disponível na biblioteca algoritmos do C++. Logo após, os vértices adjacentes de u são colocados aleatoriamente em Q. O tempo requerido para a execução desse procedimento é $O(n)$, pois a complexidade de pior caso ocorre quando um vértice tem $n-1$ vértices adjacentes. A complexidade do algoritmo BF será influenciada pelo consumo de tempo para colorir os vértices de entrada de um vértice. Como cada vértice é visitado uma vez, o tempo de execução do algoritmo BF é $O(n(nm+n)) = O(n^2m)$, conforme pode ser observado no Algoritmo 13 que representa esse algoritmo.

Algoritmo 13: Algoritmo BF

```
Entrada: Digrafo D e digrafo backbone B;
Saída: Número de cores usadas na coloração de B;
início
    INICIALIZAR_VETOR(cor, BRANCO, B.n);
    r ← VERTICE_MAIOR_GRAU(B);
    cor[r] ← CINZA;
    INSERIR_VERTICE(Q, r);
    enquanto Q ≠ ∅ faça
        u ← VERTICE_INICIO(Q);
        REMOVER_VERTICE_INICIO(Q);
        para cada vértice v ∈ V(B) faça
            se vu ∈ A(B) então
                para cor = 1 até cor ≤ B.m faça
                    se NAO_CONFLITO(B, vu, cor) então
                        corSelecionada = cor;
                        pausa;
                    fim
                fim
                COLORIR_ARCO(D, B, vu, corSelecionada);
            fim
        fim
        VA ← VERTICES_ADJACENTES(B, u);
        EMBARALHAR_VERTICES_ADJACENTES(VA);
        k ← NUMERO_VERTICES(VA);
        para i = 0 até i < k faça
            v ← VA[i];
            se cor[v] = BRANCO então
                cor[v] ← CINZA;
                INSERIR_VERTICE(Q, v);
            fim
        fim
    fim
    retorna B.mcor;
fim
```

O algoritmo DFOr começa colorindo os arcos de entrada do vértice de grau máximo no digrafo *backbone*. Esse algoritmo segue a descoberta de vértices de forma semelhante com a técnica adotada pelo algoritmo de busca em profundidade, porém os vértices adjacentes são visitados em ordem decrescente de grau. Nos algoritmos BF e DFOr, os arcos de entrada são coloridos conforme a descoberta dos vértices. Esses algoritmos consideram o digrafo *backbone* como um grafo subjacente durante a descoberta dos vértices. A Figura 23 ilustra uma coloração BPRN usando o algoritmo DFOr, com os vértices sendo visitados na seguinte ordem: c, a, b, d e e.

Figura 23 – Uma coloração BPRN com o algoritmo DFOr.



Fonte – Elaborado pelo autor

Uma função importante no algoritmo DFOr é a responsável por selecionar um vértice de maior grau não colorido entre os vértices adjacentes de um vértice, conforme apresentado no Algoritmo 14, sendo denominado de VERTICE_ADJACENTE_MAIOR_GRAU e com o tempo de execução $O(n)$.

Algoritmo 14: Algoritmo para Selecionar um Vértice Adjacente de Maior Grau

Entrada: Vetor *VA* de vértices adjacentes de um vértice *v*;
Saída: Vértice *k* de maior grau em *VA*;
início
    p $\leftarrow -1$, maiorGrau $\leftarrow -\infty$;
    para *i = 0 até i < NUMERO_VERTICES(VA)* **faça**
        se *VA[i].cor = BRANCO* $\wedge$ *maiorGrau < VA[i].grau* **então**
            p $\leftarrow$ i;
            maiorGrau $\leftarrow$ VA[i].grau;
        fim
    fim
    se *p = -1* **então**
        retorna *p*;
    fim
    senão
        retorna *VA[p]*;
    fim
fim

O algoritmo DFOr é composto por duas fases, conforme visto no Algoritmo 15. A primeira fase consiste na inicialização do vetor que representa as cores dos vértices com a cor branca. Logo após, o vértice de maior grau no digrafo *backbone* é selecionado. A segunda etapa simboliza uma busca em profundidade recursiva a partir desse vértice de maior grau. Essa segunda etapa chamada de DFS_VISIT_BPRN será retratada através do Algoritmo 16.

Algoritmo 15: Algoritmo DFOr

Entrada: Digrafo *D* e digrafo *backbone B*;
Saída: Número de cores usadas na coloração de *B*;
início
    INICIALIZAR_VETOR(cor, BRANCO, B.n);
    r $\leftarrow$ VERTICE_MAIOR_GRAU(B);
    DFS_VISIT_BPRN(D, B, cor, r);
    retorna *B.mcor*;
fim

Algoritmo 16: Algoritmo de Busca em Profundidade

```
Entrada: Digrafo D, digrafo backbone B, vetor cor de cores e vértice descoberto v;
início
    cor[u] ← CINZA;
    para cada vértice v ∈ V(B) faça
        se vu ∈ A(B) então
            para cor = 1 até cor ≤ B.m faça
                se NAO_CONFLITO(B, vu, cor) então
                    corSelecionada = cor;
                    pausa;
                fim
            fim
            COLORIR_ARCO(D, B, vu, corSelecionada);
        fim
    fim
    VA ← VERTICES_ADJACENTES(B, u);
    k ← NUMERO_VERTICES(VA);
    para i = 0 até i < k faça
        v ← VERTICE_ADJACENTE_MAIOR_GRAU(VA);
        se v ≠ −1 ∧ cor[v] = BRANCO então
            cor[v] ← CINZA;
            DFS_VISIT_BPRN(D, B, cor, v);
        fim
    fim
fim
```

A análise de complexidade da segunda parte do algoritmo DFOr é semelhante com a feita no algoritmo BF. O tempo gasto para colorir os arcos de entrada de um vértice é $O(nm)$, como foi visto anteriormente na análise do algoritmo BF. A etapa do algoritmo DFOr responsável por visitar os vértices adjacentes de um vértice em ordem de maior grau tem a complexidade $O(n^2)$, pois o pior caso ocorre quando um vértice tem $n-1$ vértices adjacentes. Após essa análise, conclui-se que a complexidade do algoritmo DFOr é $O(n(nm+n^2)) = O(n^2m)$.

5 RESULTADOS COMPUTACIONAIS

Neste capítulo, apresentam-se os cenários usados na avaliação de desempenho dos algoritmos com relação as colorações encontradas. Esses cenários contemplam instâncias com níveis diferentes de densidade, cujo objetivo é avaliar o desempenho dos algoritmos à medida que as instâncias ficam mais densas e, assim, tirar conclusões a respeito das abordagens de coloração adotadas nos algoritmos. Essas instâncias representam topologias de rede baseadas em grafo de pontos. Além disso, o algoritmo de busca em largura é usado para verificar se uma instância gerada representa uma árvore geradora orientada em direção a um vértice raiz denominado de estação base. Na criação dessas instâncias, os nós são colocados de forma aleatória em uma região, e a partir da informação do raio de transmissão de cada nó, o digrafo da rede é criado. Após uma explanação dos cenários usados e das abordagens adotadas para a criação das instâncias. A etapa seguinte consiste em apresentar os resultados, deixando evidente as diferenças de desempenho entre os algoritmos em cada cenário.

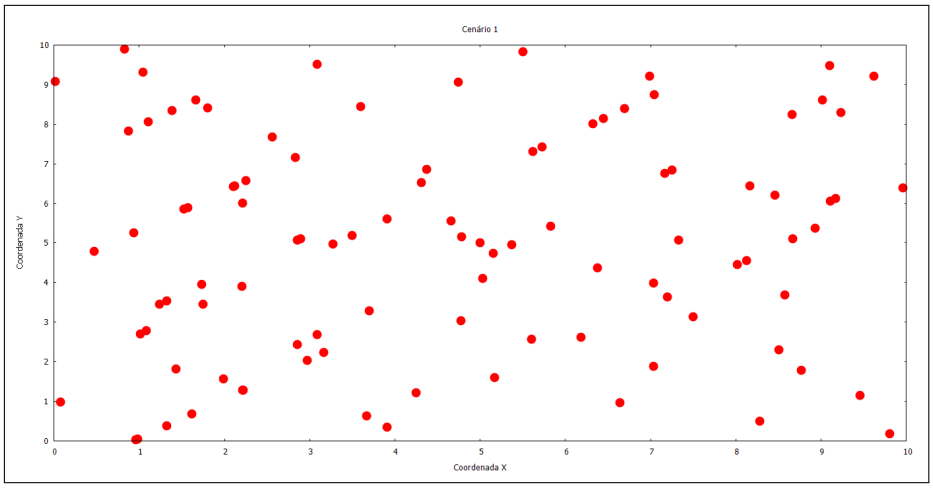
5.1 CENÁRIOS E MODELO DE REDE

Os cenários usados para comparar os resultados dos algoritmos têm o intuito de verificar o desempenho desses algoritmos na tarefa de encontrar colorações BPRN em instâncias com níveis variados de densidade. A utilização de instâncias com níveis diferentes de densidade, é para compreender se com o aumento da densidade, os algoritmos conseguem melhorar seus resultados. Assim, são usados três cenários com instâncias de baixa, média e alta densidade, respectivamente, onde cada cenário tem 30 instâncias de mesma densidade com relação ao número de vértices. Essas instâncias são baseadas em topologias de redes sem fio, onde cada uma tem um nó coletor denominado de estação base localizado no centro da região simulada, para onde todo o tráfego está direcionado. Elas são construídas usando o algoritmo de busca em largura, com a estação base representando o nó raiz, sendo uma abordagem semelhante à usada no trabalho (MA *et al.*, 2009).

O modelo de rede adotado para criar as instâncias é baseado em grafo de pontos, onde cada nó está relacionado com um ponto no espaço bidimensional e tem um raio de transmissão. Assim, um nó v é vizinho de um nó u, se a distância euclidiana entre u e v é menor ou igual ao raio de transmissão do nó u. De forma semelhante com o trabalho (WANG *et al.*, 2006), os nós são gerados de forma aleatória em uma região de $10x10$ unidades, com o raio de transmissão no intervalo de $1,8$ a 2 unidades, sendo que uma unidade equivale a 50 metros. No primeiro

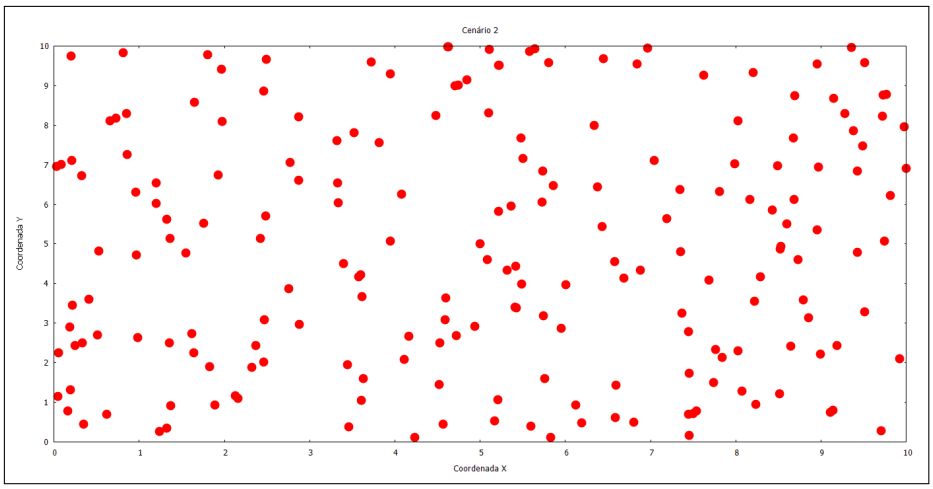
cenário, referente as instâncias de baixa densidade, cada instância possui 100 nós, conforme ilustrado na Figura 24. O segundo cenário representa as instâncias de densidade média, com cada uma possuindo 200 nós. A Figura 25 representa uma instância desse cenário. Por fim, o terceiro cenário consiste nas instâncias de maior densidade, onde cada uma tem 300 nós, como visto na Figura 26. A Tabela 2 apresenta as configurações dos três cenários adotados.

Figura 24 – Exemplo de uma instância do Cenário 1 com 100 nós.



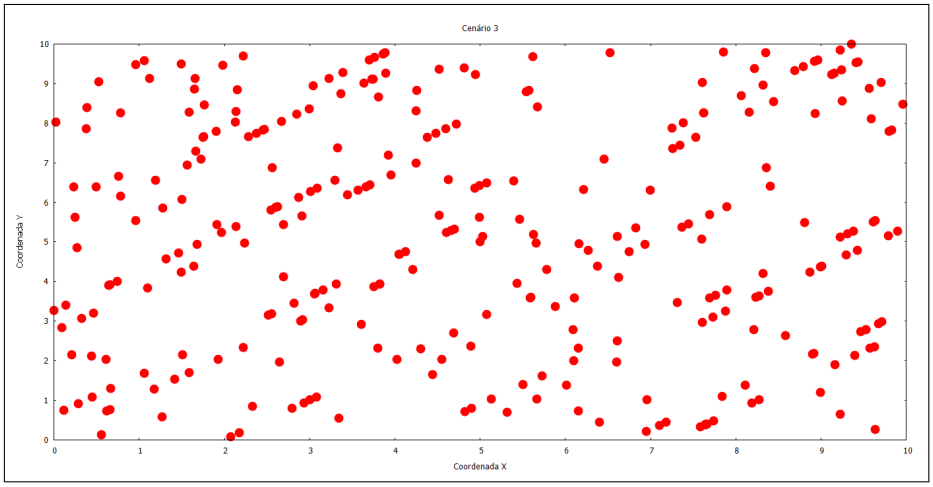
Fonte – Elaborado pelo autor

Figura 25 – Exemplo de uma instância do Cenário 2 com 200 nós.



Fonte – Elaborado pelo autor

Figura 26 – Exemplo de uma instância do Cenário 3 com 300 nós.



Fonte – Elaborado pelo autor

Tabela 2 – Cenários

Cenário	Quantidades de nós	Área (Unidades x Unidades)	Posição da Estação Base
1	100	10x10	(5,5)
2	200	10x10	(5,5)
3	300	10x10	(5,5)

Fonte – Elaborado pelo autor

5.2 ANÁLISE E VERIFICAÇÃO DOS RESULTADOS

Nesta seção, apresentam-se os resultados dos algoritmos propostos, e os resultados dos algoritmos usados para comparação. Além disso, realiza-se uma análise desses resultados, destacando o desempenho dos algoritmos através de comparações entre os números de cores das colorações encontradas em cada um dos cenários. Por exemplo, dado um cenário, indicar em quantas instâncias um algoritmo conseguiu melhores resultados em comparação com outro algoritmo. Esses algoritmos foram implementados em C++, com o uso do Eclipse IDE na versão Luna. Os testes foram realizados em um computador com 4GB de memória RAM, processador Intel(R) Core(TM) CPU i3-5005U @ 2.00GHz e Sistema Operacional Linux.

O algoritmo BF seleciona aleatoriamente os nós adjacentes de um nó e adiciona em uma fila. Depois retira cada um desses nós da fila, e colore seus arcos de entrada. Portanto, aplicando esse algoritmo em uma instância mais de uma vez, os resultados encontrados podem ser diferentes. Dessa forma, o algoritmo BF foi executado 10 vezes para cada instância dos três cenários, com a seleção do menor número de cores encontrado nessas execuções. Em cada

um dos três cenários são encontrados o número de cores na coloração de cada instância, menor número de cores, média e desvio padrão dos números de cores.

Nas Tabelas 3 e 4, são apresentados os resultados dos algoritmos em relação ao Cenário 1. Como pode ser visto na Tabela 3, o algoritmo DFOr é o que apresenta resultados menos expressivos em geral. Apesar do algoritmo BF não possuir uma ordem definida de visitação dos vértices, após a escolha do primeiro vértice de maior grau no digrafo *backbone*, os resultados encontrados no Cenário 1 com instâncias menos densas, superam os resultados encontrados pelo algoritmo DFOr nesse cenário.

Os algoritmos Bsatur e Bswap têm praticamente os mesmos resultados, pois em instâncias menos densas, as oportunidades de reutilização de cores diminuem e, assim, na instância em que o algoritmo Bswap não consegue reutilizar nenhuma cor, o número de cores encontrado em uma coloração é o mesmo do algoritmo Bsatur para essa instância. Os resultados encontrados no Cenário 1, indicam que as estratégias de coloração usadas nos algoritmos Bsatur e Bswap produzem, em geral, colorações mais eficientes em comparação com as colorações encontradas pelos algoritmos BF e DFOr.

Tabela 3 – Número de cores encontrado em cada instância do Cenário 1.

Cenário 1 - Instâncias de 1 a 15															
Algoritmos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Bsatur	14	18	17	17	16	19	16	17	14	16	16	14	16	18	17
Bswap	14	18	17	17	16	19	16	17	14	16	16	14	16	18	17
BF	15	18	17	17	17	19	16	18	15	16	16	14	16	18	17
DFOr	15	19	19	16	20	19	16	18	16	16	16	14	16	18	17
Cenário 1 - Instâncias de 16 a 30															
Algoritmos	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Bsatur	19	14	16	15	16	15	16	16	15	14	13	17	24	15	16
Bswap	19	14	16	15	16	15	16	16	15	13	13	17	24	15	16
BF	19	14	16	16	17	16	16	16	16	14	14	17	24	15	16
DFOr	19	14	16	16	18	16	17	17	15	14	13	17	24	16	17

Fonte – Elaborado pelo autor

Tabela 4 – Resultados dos algoritmos no Cenário 1.

Algoritmo	Resultado médio	Menor resultado	Desvio padrão
Bsatur	16.200	13	2.056
Bswap	16.167	13	2.099
BF	16.500	14	1.928
DFOr	16.800	13	2.135

Fonte – Elaborado pelo autor

As Tabelas 5 e 6 apresentam os resultados dos algoritmos no Cenário 2 composto por instâncias mais densas, conforme ilustrado na Figura 25, que representa uma instância desse cenário. No Cenário 2, a diferença de desempenho entre o algoritmo Bswap e Bsatur não está tão nítida, pois o número de conflitos ainda está proporcionando poucas oportunidades de reutilização de cores no processo de coloração dos arcos do digrafo *backbone*. Nesse cenário, o algoritmo Bswap obteve melhor desempenho em apenas 10% das instâncias, com relação ao algoritmo Bsatur. Os resultados alcançados no Cenário 2 indicam que aumentando a densidade das instâncias, os algoritmos propostos conseguem melhores resultados em comparação com os resultados dos algoritmos BF e DFOr.

O algoritmo BF foi superior em $36,66\%$ das instâncias do Cenário 2, quando comparado com o algoritmo DFOr. Já o algoritmo DFOr, conseguiu melhores resultados somente em 20% das instâncias, com relação ao algoritmo BF. O desempenho do algoritmo Bsatur, foi melhor em $66,66\%$ com relação ao algoritmo BF, e alcançou resultados superiores em 70% das instâncias ao ser comparado com o algoritmo DFOr. O algoritmo Bswap obteve melhor êxito em $66,66\%$ das instâncias, quando confrontado com o algoritmo BF, e alcançou melhores resultados em $73,33\%$ das instâncias, ao ser comparado com o algoritmo DFOr. Os resultados demonstram que mesmo as instâncias não sendo tão densas, os algoritmos Bsatur e Bswap alcançam resultados melhores em relação aos resultados dos algoritmos BF e DFOr.

Tabela 5 – Número de cores encontrado em cada instância do Cenário 2.

Cenário 2 - Instâncias de 1 a 15															
Algoritmos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Bsatur	22	32	28	32	26	30	29	30	35	27	28	26	30	29	27
Bswap	22	32	28	32	26	30	29	30	35	25	28	25	29	29	27
BF	23	32	29	33	27	30	30	31	35	28	29	28	31	30	27
DFOr	25	32	28	33	30	32	30	32	35	28	29	25	30	29	29
Cenário 2 - Instâncias de 16 a 30															
Algoritmos	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Bsatur	33	32	34	35	30	35	27	30	28	31	33	27	26	28	33
Bswap	33	32	34	35	30	35	27	30	28	31	33	27	26	28	33
BF	34	36	36	35	30	35	27	32	28	33	33	28	27	29	34
DFOr	34	33	34	35	32	36	28	32	30	35	33	28	28	29	34

Fonte – Elaborado pelo autor

Tabela 6 – Resultados dos algoritmos no Cenário 2.

Algoritmo	Resultado médio	Menor resultado	Desvio padrão
Bsatur	29.767	22	3.138
Bswap	29.633	22	3.261
BF	30.667	23	3.197
DFOr	30.933	25	2.909

Fonte – Elaborado pelo autor

As informações expostas nas Tabelas 7 e 8 representam os resultados alcançados pelos algoritmos no Cenário 3. Esse cenário é composto por instâncias mais densas, conforme visto na Figura 26, representando uma instância desse cenário. Em instâncias mais densas, o número de conflitos aumenta e, assim, possibilita a reutilização de cores durante a coloração dos arcos do digrafo *backbone*. Por isso, observa-se no Cenário 3, que os resultados do algoritmo Bswap apresentam melhor desempenho quando comparados com os resultados do algoritmo Bsatur.

Os resultados na Tabela 7 demonstram que o algoritmo BF obteve melhores colorações em 50% das instâncias em comparação com o algoritmo DFOr. Já o algoritmo DFOr, conseguiu melhores resultados em apenas 10% das instâncias com relação aos resultados alcançados pelo algoritmo BF. No Cenário 3, o algoritmo Bsatur obteve superioridade em $43,33\%$ das instâncias em comparação com o algoritmo BF, e quando comparado com o algoritmo DFOr, os resultados são melhores em $66,66\%$ das instâncias. O algoritmo Bswap alcançou maior êxito em $46,66\%$ das instâncias do Cenário 3, quando comparado com o algoritmo BF, e obteve colorações melhores em $73,33\%$ das instâncias desse cenário em comparação com o algoritmo DFOr. O Cenário 3 composto por instâncias mais densas, permitiu que o algoritmo Bswap conseguisse resultados melhores em $23,33\%$ das instâncias, com relação aos resultados do algoritmo Bsatur.

Tabela 7 – Número de cores encontrado em cada instância do Cenário 3.

Cenário 3 - Instâncias de 1 a 15															
Algoritmos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Bsatur	35	36	57	40	53	39	42	41	49	48	36	37	48	42	40
Bswap	35	35	57	40	53	39	42	41	49	48	35	36	48	42	40
BF	39	36	57	41	53	40	42	41	50	48	38	40	48	42	40
DFOr	42	36	57	45	56	42	43	44	49	48	38	42	48	42	40
Cenário 3 - Instâncias de 16 a 30															
Algoritmos	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Bsatur	51	45	48	37	47	40	45	41	39	35	34	41	38	38	41
Bswap	51	43	45	37	47	38	45	41	39	34	34	41	38	38	41
BF	51	48	44	37	47	38	45	41	40	38	35	45	42	38	42
DFOr	54	48	44	37	49	40	51	42	42	38	37	42	40	39	44

Fonte – Elaborado pelo autor

Tabela 8 – Resultados dos algoritmos no Cenário 3.

Algoritmo	Resultado médio	Menor resultado	Desvio padrão
Bsatur	42.100	34	5.706
Bswap	41.733	34	5.785
BF	42.867	35	5.246
DFOr	43.967	36	5.468

Fonte – Elaborado pelo autor

Uma forma de comparar o desempenho entre algoritmos é usando a função GAP = (SOLUCAO - MELHOR / MELHOR), onde SOLUCAO consiste na solução de um algoritmo para uma instância, e MELHOR representa a menor solução para essa instância entre as soluções dos algoritmos comparados. A função GAP foi aplicada em cada instância para possibilitar o cálculo da média dos GAPs de um algoritmo em cada cenário. A Tabela 9 retrata a média dos GAPs para os algoritmos comparados nos três cenários.

Tabela 9 – Média dos GAPs em cada cenário.

Algoritmo	Cenário 1	Cenário 2	Cenário 3
Bsatur	0.005	0.005	0.010
Bswap	0.002	0.000	0.001
BF	0.025	0.036	0.031
DFOr	0.043	0.046	0.057

Fonte – Elaborado pelo autor

Os resultados da Tabela 9 demonstram que o desempenho dos algoritmos Bsatur e Bswap é melhor em relação ao desempenho dos algoritmos BF e DFOr. No Cenário 2, o algoritmo Bswap obteve um número de cores menor ou igual ao número de cores obtido pelos

outros algoritmos em cada instância. No Cenário 1, o algoritmo DFOr foi melhor apenas na instância 4 em relação aos algoritmos Bsatur e Bswap. Nas outras instâncias, os algoritmos Bsatur e Bswap obtiveram um número de cores menor ou igual ao número de cores do algoritmo DFOr. Já no Cenário 2, o algoritmo DFOr foi melhor apenas na instância 12 em comparação com o algoritmo Bsatur. Na instância 18 do Cenário 3, os algoritmos BF e DFOr foram melhores em comparação com os algoritmos Bsatur e Bswap. Por fim, na instância 21 do Cenário 3, o algoritmo BF foi melhor em relação ao algoritmo Bsatur. Os resultados mostram que são poucas as instâncias em que os algoritmos BF e DFOr conseguem resultados melhores em relação aos resultados dos algoritmos Bsatur e Bswap. Após uma análise em relação aos resultados dos algoritmos, é possível dizer que os algoritmos Bsatur e Bswap obtiveram melhores resultados em comparação com os resultados dos algoritmos BF e DFOr nos cenários com instâncias de pouca, média e alta densidade.

Na coleta dos tempos de execução para cada instância dos três cenários, o menor tempo de execução encontrado pelo algoritmo BF em 10 execuções de uma mesma instância, é o tempo de execução associado a esse algoritmo para essa instância. A Tabela 10 apresenta os tempos de execução dos algoritmos para cada instância do Cenário 1. Os tempos de execução dos algoritmos BF e DFOr foram os mesmos para todas as instâncias. Os algoritmos Bsatur e Bswap apresentaram tempos de execução maiores em comparação com os tempos de execução dos algoritmos BF e DFOr. Essas diferenças refletem a complexidade de cada algoritmo.

Apesar da complexidade de cada algoritmo ser obtida em relação a função de maior complexidade, existem outras funções de menor complexidade que incrementam o tempo de execução desses algoritmos. Assim, é possível que algoritmos analisados com a mesma complexidade, apresentem tempos de execução diferentes ao serem executados em uma mesma instância. A Tabela 11 apresenta o tempo de execução médio, menor e maior tempo de execução dos algoritmos nas instâncias do Cenário 1.

Tabela 10 – Tempo de execução em segundos para cada instância do Cenário 1.

Cenário 1 - Instâncias de 1 a 30										
Algoritmos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Bsatur	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.012	0.013	0.012
Bswap	0.015	0.019	0.024	0.022	0.016	0.019	0.015	0.032	0.015	0.015
BF	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
DFOr	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Algoritmos	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Bsatur	0.013	0.012	0.011	0.014	0.011	0.012	0.012	0.012	0.012	0.013
Bswap	0.016	0.014	0.019	0.024	0.022	0.015	0.014	0.015	0.020	0.016
BF	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
DFOr	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Algoritmos	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Bsatur	0.013	0.013	0.013	0.012	0.013	0.012	0.012	0.013	0.013	0.012
Bswap	0.015	0.019	0.016	0.014	0.016	0.014	0.016	0.018	0.015	0.018
BF	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
DFOr	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001

Fonte – Elaborado pelo autor

Tabela 11 – Resultados dos tempos de execução no Cenário 1.

Algoritmo	Tempo médio	Menor tempo	Maior tempo
Bsatur	0.013	0.011	0.014
Bswap	0.018	0.014	0.032
BF	0.001	0.001	0.001
DFOr	0.001	0.001	0.001

Fonte – Elaborado pelo autor

A Tabela 12 expõe os tempos de execução dos algoritmos para cada instância do Cenário 2, onde cada uma tem 200 nós. Os tempos de execução dos algoritmos BF e DFOr obtiveram um leve aumento em relação aos tempos de execução obtidos nas instâncias do Cenário 1. Esses algoritmos conseguiram os mesmos tempos de execução no Cenário 2. Os algoritmos Bsatur e Bswap obtiveram tempos de execução nas instâncias do Cenário 2, com um certo crescimento em relação aos tempos de execução obtidos no Cenário 1. A Tabela 13 expõe os tempos de execução dos algoritmos nas instâncias do Cenário 2, em relação ao tempo de execução médio, menor e maior tempo de execução.

Tabela 12 – Tempos de execução em segundos para cada instância do Cenário 2.

Cenário 2 - Instâncias de 1 a 30										
Algoritmos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Bsatur	0.072	0.074	0.073	0.072	0.072	0.068	0.070	0.072	0.075	0.074
Bswap	0.080	0.240	0.086	0.199	0.121	0.114	0.100	0.087	0.092	0.098
BF	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
DFOr	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
Algoritmos	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Bsatur	0.073	0.072	0.072	0.071	0.073	0.075	0.071	0.071	0.070	0.070
Bswap	0.085	0.189	0.138	0.083	0.111	0.090	0.085	0.087	0.088	0.084
BF	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
DFOr	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
Algoritmos	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Bsatur	0.069	0.072	0.070	0.074	0.070	0.075	0.071	0.070	0.074	0.067
Bswap	0.086	0.083	0.130	0.101	0.101	0.090	0.093	0.089	0.148	0.215
BF	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
DFOr	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002

Fonte – Elaborado pelo autor

Tabela 13 – Resultados dos tempos de execução no Cenário 2.

Algoritmo	Tempo médio	Menor tempo	Maior tempo
Bsatur	0.072	0.067	0.075
Bswap	0.113	0.080	0.240
BF	0.002	0.002	0.002
DFOr	0.002	0.002	0.002

Fonte – Elaborado pelo autor

A Tabela 14 apresenta os tempos de execução dos algoritmos para cada instância do Cenário 3. Observa-se que os tempos de execução dos algoritmos Bsatur e Bswap aumentaram de forma mais expressiva em relação aos tempos de execução obtidos no Cenário 2. Os tempos de execução do algoritmo BF apresentaram poucas diferenças em relação aos tempos de execução do algoritmo DFOr nas instâncias desse cenário. A Tabela 15 mostra o tempo de execução médio, menor e maior tempo de execução dos algoritmos nas instâncias do Cenário 3.

Tabela 14 – Tempo de execução em segundos para cada instância do Cenário 3.

Cenário 3 - Instâncias de 1 a 30										
Algoritmos	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Bsatur	0.206	0.209	0.213	0.208	0.212	0.197	0.213	0.197	0.204	0.213
Bswap	0.273	0.333	0.305	0.666	0.298	0.227	0.249	0.230	0.339	0.258
BF	0.005	0.004	0.005	0.005	0.005	0.004	0.005	0.004	0.005	0.005
DFOr	0.005	0.004	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
Algoritmos	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Bsatur	0.211	0.215	0.213	0.203	0.209	0.218	0.204	0.215	0.210	0.209
Bswap	0.327	0.265	0.257	0.237	0.549	0.268	0.286	1.303	0.326	0.255
BF	0.005	0.004	0.005	0.004	0.004	0.005	0.005	0.005	0.004	0.005
DFOr	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
Algoritmos	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Bsatur	0.203	0.209	0.211	0.216	0.209	0.213	0.193	0.208	0.210	0.201
Bswap	0.358	0.298	0.392	0.241	0.349	0.261	0.237	0.262	0.286	0.321
BF	0.004	0.005	0.005	0.004	0.004	0.005	0.004	0.004	0.004	0.004
DFOr	0.004	0.005	0.005	0.005	0.004	0.005	0.004	0.005	0.005	0.004

Fonte – Elaborado pelo autor

Tabela 15 – Resultados dos tempos de execução no Cenário 3.

Algoritmo	Tempo médio	Menor tempo	Maior tempo
Bsatur	0.208	0.193	0.218
Bswap	0.342	0.227	1.303
BF	0.005	0.004	0.005
DFOr	0.005	0.004	0.005

Fonte – Elaborado pelo autor

Como pode ser observado nas Tabelas 10, 12 e 14, os tempos de execução dos algoritmos Bsatur e Bswap foram maiores em comparação com os tempos de execução dos algoritmos BF e DFOr. Os algoritmos Bsatur e Bswap consumiram mais tempo ao serem executados em cada uma das instâncias, por causa dos critérios adotados para indicar à ordem de coloração dos arcos, que se baseiam no grau de conflitos e grau de saturação de um arco. O algoritmo BF colore os arcos de entrada à medida em que os vértices são visitados por uma busca em largura. O algoritmo DFOr segue uma abordagem semelhante com o algoritmo BF, mas usando uma busca em profundidade. O esforço computacional requerido pelos algoritmos BF e DFOr para estabelecer à ordem de coloração dos arcos é menor em comparação com o tempo consumido pelos algoritmos Bsatur e Bswap. Os algoritmos Bsatur e Bswap são recompensados, pois em algumas instâncias conseguem resultados melhores com relação aos algoritmos BF e DFOr. Um artigo denominado de “Algorithms for BPRN Coloring of a Digraph” com os resultados deste trabalho foi aceito no ISCC 2018 (ISCC - International Symposium on Computers and Communications).

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho, propôs algoritmos e uma modelagem para resolver o problema de coloração BPRN. Os algoritmos propostos, denominados de Bsatur e Bswap, representam versões adaptadas do algoritmo Dsatur para a coloração BPRN. Esses algoritmos usam os principais conceitos envolvidos no algoritmo Dsatur, que são o grau e o grau de saturação de um vértice. Nos algoritmos Bsatur e Bswap, o conceito de grau de um vértice é traduzido como o grau de conflitos de um arco, significando o número de arcos que causam conflitos primários e secundários a esse arco. De forma semelhante, o grau de saturação de um vértice abordado no algoritmo Dsatur, é visto nos algoritmos propostos, como o grau de saturação de um arco, sendo representado pelo número de arcos saturados desse arco.

Os conceitos de grau de conflitos e grau de saturação de um arco, são importantes para indicar a ordem em que os arcos serão coloridos com a menor cor livre no intervalo de 1 a m, onde m é o número de arcos no digrafo *backbone*. O modelo de coloração BPRN representa uma extensão do modelo de coloração PRN. No modelo PRN todos os arcos do digrafo serão coloridos. Já no modelo BPRN, apenas um subconjunto de arcos do digrafo serão coloridos, onde esse subconjunto de arcos é denominado de digrafo *backbone*. Um digrafo *backbone* pode representar topologias mais simples, por exemplo, uma árvore ou um digrafo com graus reduzidos. Em conjunto com os algoritmos, foram propostas algumas definições, capazes de identificar no digrafo *backbone*, os arcos conflitantes, arcos saturados e não saturados de um arco. Essas definições ainda permitem verificar, se uma cor selecionada no intervalo de 1 a m, pode ser usada para colorir um determinado arco.

O algoritmo Bswap consiste em uma versão ampliada do algoritmo Bsatur, por causa de um mecanismo de reutilização de cores. Essa reutilização de cores acontece quando uma cor mínima para colorir um arco uv é igual a $k+1$, e o procedimento de reutilização de cores consegue trocar as cores entre um arco $ab \in SATURADOS(uv)$ e um arco $xy \in NAO\_SATURADOS(uv)$, satisfazendo as seguintes condições: $c(ab) \notin CORES(xy)$, $c(xy) \notin CORES(ab)$ e $c(ab) \notin CORES(uv)$, onde k é o número de cores usado em uma coloração parcial do digrafo *backbone*. Assim, se essas condições forem satisfeitas, a cor $c(ab)$ pode ser usada para colorir o arco uv.

Os algoritmos BF e DFOr de coloração PRN, foram adaptados para a coloração BPRN, e usados com o objetivo de comparar com os algoritmos Bsatur e Bswap. Na avaliação dos algoritmos, foram usados três cenários para avaliar o desempenho dos algoritmos em

instâncias com baixa, média e alta densidade. Os resultados demonstraram que no Cenário 1, com instâncias de baixa densidade, o número de instâncias em que os algoritmos Bsatur e Bswap obtiveram um número de cores menor em comparação com o número de cores dos algoritmos BF e DFOr é menos expressivo. Contudo, nos Cenários 2 e 3 com instâncias mais densas, o número de instâncias em que os algoritmos Bsatur e Bswap obtiveram resultados melhores em comparação com os resultados dos algoritmos BF e DFOr aumentou em relação ao Cenário 1.

Os resultados mostraram um desempenho melhor dos algoritmos Bsatur e Bswap em relação ao desempenho dos algoritmos BF e DFOr nos três cenários avaliados. O algoritmo Bswap obteve resultados melhores em instâncias mais densas, pois o número de conflitos aumenta e, consequentemente, possibilita mais oportunidades de reutilização de cores durante a coloração dos arcos do digrafo *backbone*. Os algoritmos Bsatur, BF e DFOr apresentaram a mesma complexidade. No entanto, o algoritmo Bswap apresentou uma complexidade maior devido ao uso de um mecanismo de reutilização de cores.

6.1 TRABALHOS FUTUROS

As oportunidades de trabalhos futuros podem está relacionadas com o aprimoramento dos algoritmos apresentados, além da proposta de outros algoritmos para o problema de coloração BPRN. A seguir são apresentadas algumas direções de possíveis trabalhos futuros.

a) Aperfeiçoar os algoritmos Bsatur e Bswap em relação à melhoria dos métodos usados para selecionar um arco a ser colorido. Considerando não apenas o grau de conflitos e o grau de saturação, mas estendendo a pesquisa para considerar outras relações entre os arcos;
b) Diminuir a complexidade do mecanismo de reutilização de cores usado no algoritmo Bswap;
c) Avaliar o desempenho dos algoritmos Bsatur e Bswap em instâncias com um número maior de nós;
d) Propor algoritmos usando uma determinada ordenação dos arcos a serem coloridos com base na informação de grau de conflitos;
e) Considerar outras topologias, por exemplo, topologias de anel e topologias com o grau dos nós reduzidos;
f) Desenvolver protocolos MAC centralizados, que incorporem os algoritmos desenvolvidos, para o escalonamento TDMA de enlaces em redes sem fio;

REFERÊNCIAS

APPEL, K.; HAKEN, W. Every planar map is four colorable. **Bull. Amer. Math. Soc.**, v. 82, n. 5, p. 711–712, set. 1976.

ARIKAN, E. Some complexity results about packet radio networks (corresp.). **IEEE Transactions on Information Theory**, v. 30, n. 4, p. 681–685, jul. 1984.

BANG-JENSEN, J.; GUTIN, G. Z. **Digraphs:** Theory, Algorithms and Applications. 2nd. ed. [S.l.]: Springer, 2009.

BONDY, J. A.; MURTY, U. S. R. **Graph theory, volume 244 of Graduate Texts in Mathematics**. [S.l.]: Springer, New York, 2008.

BRÉLAZ, D. New methods to color the vertices of a graph. **Commun. ACM**, ACM, New York, NY, USA, v. 22, n. 4, p. 251–256, abr. 1979.

CHARTRAND, P. Z. G. **Chromatic graph theory**. [S.l.]: Chapman & Hall/CRC, 2009. (Discrete mathematics and its applications).

CHENG, M.; YIN, L. Transmission scheduling in sensor networks via directed edge coloring. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 2007, Glasgow, Scotland. **Proceedings...** Glasgow, Scotland: IEEE, 2007. p. 3710–3715.

DIESTEL, R. **Graph Theory**. 2nd. ed. [S.l.]: Springer, 2000. (Graduate Texts in Mathematics).

DJUKIC, P. **Scheduling Algorithms for TDMA Wireless Multihop Networks**. 2008. 203 f. Tese (Doutorado em Filosofia) – Department Of Electrical And Computer Engineering, University Of Toronto, Toronto, 2008.

DJUKIC, P.; VALAEE, S. Delay aware link scheduling for multi-hop tdma wireless networks. **IEEE/ACM Transactions on Networking**, v. 17, n. 3, p. 870–883, jun. 2009.

EVEN, S.; GOLDREICH, O.; MORAN, S.; TONG, P. On the np-completeness of certain network testing problems. **Networks**, Wiley Online Library, v. 14, n. 1, p. 1–24, 1984.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability; A Guide to the Theory of NP-Completeness**. New York, NY, USA: W. H. Freeman & Co., 1990.

GEIER, J. **Wireless system architecture: how wireless works**. 2004. Disponível em: <http://www.wireless-nets.com/resources/sample_chapters/wireless_first_step_C1.pdf>. Acesso em: 09 mar. 2018.

HOLYER, I. The np-completeness of edge-coloring. **SIAM Journal on Computing**, v. 10, n. 4, p. 718–720, 1981.

HUSON, M. L. **A New Model for Scheduling Radio Networks**. 1995. 117 f. Tese (Doutorado em Filosofia) – Department Of The Air Force AFIT/CI, Arizona State University, Tempe, 1995.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a Internet:** uma abordagem top-down. 6. ed. São Paulo: Pearson, 2013.

LLOYD, E. L.; RAMANATHAN, S. **On the complexity of link scheduling in multi-hop radio networks**. 1992. Disponível em: <https://www.eecis.udel.edu/~elloyd/papers.d/ciss92.pdf>. Acesso em: 14 jun. 2017.

LUIZ, A. G. **Coloração de grafos e suas aplicações**. 2015. Disponível em: <https://www.ic.unicamp.br/~atilio/slidesWtisc.pdf>. Acesso em: 25 mar. 2018.

MA, J.; LOU, W.; WU, Y.; LI, X. Y.; CHEN, G. Energy efficient tdma sleep scheduling in wireless sensor networks. In: IEEE INFOCOM, 2009, Rio de Janeiro. **Proceedings...** Rio de Janeiro: [s.n.], 2009. p. 630–638.

NELSON, R.; KLEINROCK, L. Spatial tdma: A collision-free multihop channel access protocol. **IEEE Transactions on Communications**, v. 33, n. 9, p. 934–944, set. 1985.

PEIXOTO, J. L. S. **Estudo do Tempo de Acesso ao Meio em Redes em Malha sem Fio**. 2009. 86 f. Dissertação (Mestrado em Engenharia de Sistemas e Computação) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.

RAMANATHAN, S.; LLOYD, E. L. Scheduling algorithms for multihop radio networks. **IEEE/ACM Trans. Netw.**, IEEE Press, Piscataway, NJ, USA, v. 1, n. 2, p. 166–177, abr. 1993.

ROBERTSON, N.; SANDERS, D.; SEYMOUR, P.; THOMAS, R. The four-colour theorem. **journal of combinatorial theory, Series B**, Elsevier, v. 70, n. 1, p. 2–44, 1997.

ROCHA, L.; SASAKI, D. The backbone packet radio network coloring for time division multiple access link scheduling in wireless multihop networks. **Networks**, 2017.

STOJMENOVIC, I. **Handbook of sensor networks:** algorithms and architectures. [S.l.]: John Wiley & Sons, 2005.

VIZING, V. G. On an estimate of the chromatic class of a p-graph. **Diskret analiz**, v. 3, p. 25–30, 1964.

WANG, W.; WANG, Y.; LI, X.-Y.; SONG, W.-Z.; FRIEDER, O. Efficient interference-aware tdma link scheduling for static wireless networks. In: PROCEEDINGS OF THE 12TH ANNUAL INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, 2006, Los Angeles, CA, USA. **Proceedings...** New York, NY, USA: ACM, 2006. p. 262–273.