Aspectos de complexidade de problemas de rotulação em grafos

Candidato: Celso Aimbiré Weffort Santos

Orientadores: Christiane Neme Campos
Rafael Crivellari Saliba Schouery

18 de setembro de 2017

Resumo

Este documento apresenta o projeto de dissertação de mestrado do aluno Celso A. W. Santos no Instituto de Computação (IC) da Universidade Estadual de Campinas (UNICAMP), sob orientação da Prof.ª Christiane Neme Campos e do Prof. Rafael C. S. Schouery. Este projeto de pesquisa se insere na área de Teoria da Computação, abordando aspectos estruturais e de complexidade de problemas de rotulação em grafos. A pesquisa será desenvolvida nos anos de 2016 e 2017.

1 Introdução

Este texto apresenta a descrição do projeto de pesquisa para obtenção do título de Mestre em Ciência da Computação do aluno Celso Aimbiré Weffort Santos, do Instituto de Computação (IC) da Universidade Estadual de Campinas (UNICAMP), sendo parte integrante do pedido de concessão de bolsa de mestrado à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Este projeto se insere na área de Teoria da Computação, abordando aspectos estruturais e questões de análise de complexidade de problemas de rotulação em grafos.

O texto está estruturado como segue. Esta seção apresenta e contextualiza a área de pesquisa; na Seção 2, apresentamos os problemas que serão estudados, o seu estado da arte e alguns resultados preliminares já obtidos pelo aluno; na Seção 3, apresentamos o resumo do plano de trabalho e o cronograma de atividades previsto; na Seção 4, listamos os materiais e métodos que serão utilizados no projeto. O texto é finalizado com a Seção 5, que expõe a forma de análise do trabalho a ser desenvolvido.

As definições e notações utilizadas são introduzidas ao longo do texto, na medida em que se fizerem necessárias. De uma maneira geral, a terminologia apresentada na parte de Complexidade de Algoritmos segue os livros clássicos de T. H. Cormen et al. [4] e M. R. Garey e D. S. Johnson [8]. Já na parte de Teoria de Grafos, estão de acordo com o livro de J. A. Bondy e U. S. R. Murty [3].

1.1 Análise de Complexidade

Um *problema computacional* é uma questão geral acompanhada de parâmetros não especificados, chamados de *entrada*, para o qual se deseja obter uma resposta específica, chamada *saída*. O *enunciado* do problema informa a relação entre a entrada e a saída esperada. Um exemplo simples é o *problema da primalidade*, que consiste em determinar se um número é primo ou não. Outro exemplo é o *problema de ordenação* de um conjunto de elementos $\{n_1, n_2, ..., n_m\}$. Neste caso, a entrada é a m-tupla $A = (n_1, n_2, ..., n_m)$, e a saída, a m-tupla $A' = (n'_1, n'_2, ..., n'_m)$, que é uma permutação dos dados de entrada, satisfazendo $n'_1 \leq n'_2 \leq ... \leq n'_m$. Como último exemplo, citamos um problema muito conhecido na Teoria da Computação, chamado *problema do caixeiro viajante*, para o qual são dados um conjunto de cidades e as distâncias entre elas, e a resposta é

um percurso que, saindo de uma cidade arbitrária, passa por todas as cidades e retorna ao ponto de partida, percorrendo a menor distância possível.

Os problemas computacionais podem ser divididos em três classes: problemas de decisão, localização e otimização. Um *problema de decisão* é aquele cuja saída esperada é apenas sim ou não. A saber, o problema da primalidade é um problema de decisão. Já um *problema de localização* tem por objetivo encontrar uma saída S que satisfaça as condições requeridas no enunciado. No exemplo do problema da ordenação, encontramos uma permutação A' de A com os elementos em ordem não decrescente. Por fim, um *problema de otimização* consiste em encontrar uma solução, dentre todas as soluções possíveis, que seja considerada *ótima*, segundo algum critério especificado na entrada. No problema do caixeiro viajante, podem existir diversos percursos que passem por todas as cidades e voltem para a cidade inicial, porém estamos interessados naqueles que possuem a menor distância. Neste projeto de pesquisa, voltamos nossa atenção apenas para problemas de decisão.

O conceito de *algoritmo* é difícil de se definir precisamente. De acordo com Knuth [14], um algoritmo é um conjunto finito de regras que fornecem uma sequência de operações para resolver um tipo específico de problema computacional, e que possui cinco características importantes: (i) o algoritmo sempre termina, ou seja, é *executado* em tempo finito; (ii) cada passo do algoritmo é rigorosamente definido, sem ambiguidades ou dúvidas sobre qual operação deve ser realizada em cada etapa; (iii) possui um conjunto de dados de entrada; (iv) possui, também, um conjunto de dados saída; e (v) o algoritmo deve ser realizável, ou seja, suas operações devem ser suficientemente básicas para que qualquer pessoa possa realizá-las. Chamamos de *instância* do problema um conjunto de dados de entrada para um problema computacional. O *tamanho da entrada* é o valor n que reflete a quantidade de dados que é necessária para descrever cada instância do problema. No exemplo do problema de ordenação, o tamanho da entrada n é o número de elementos do conjunto A, enquanto no problema da primalidade, é o número de bits necessários para representar o número em questão.

A *análise de complexidade de algoritmos* consiste em estimar os recursos necessários para a execução de um algoritmo. Estes *recursos* podem ser o uso de memória, a largura de banda, o *hardware* computacional e o tempo de execução; este último sendo nosso principal foco de estudo. Definimos, então, $T(n)$ como o tempo de execução de um determinado algoritmo, para uma entrada de tamanho n. Um algoritmo é dito *eficiente* se, para cada entrada de tamanho n, $T(n)$ for limitado superiormente por algum polinômio $f(n)$. Os problemas que podem ser resolvidos por algoritmos eficientes são chamados de *tratáveis*, e os problemas de decisão tratáveis formam a classe denotada por P. Este critério de eficiência é uma forma de expressar o comportamento do algoritmo para instâncias de tamanho arbitrário. Por exemplo, consideremos os tempos de execução $T_1(n) = 2^n$ e $T_2(n) = n^3$. Vemos que para $n < 10$, $T_1(n)$ executa mais rapidamente, apesar de ser exponencial. Estamos interessados, porém, em conhecer o desempenho do algoritmo para instâncias de tamanho arbitrariamente grande. Para isso, comparamos o comportamento das funções $T_1(n)$ e $T_2(n)$ para valores crescentes de n. Percebe-se, facilmente, que o crescimento de $T_1(n)$ acontece muito mais rapidamente do que o de $T_2(n)$. A Tabela 1 retrata a diferença entre quatro algoritmos, de complexidades diferentes, executando em uma máquina capaz de realizar um milhão de instruções por segundo. Esta tabela foi retirada e traduzida do livro de D. Harel [9] e demonstra que pequenos aumentos no tamanho da entrada produzem tempos de execução muito maiores para algoritmos exponenciais, quando comparados aos polinomiais. Logo, algoritmos com $T(n)$ polinomial são muito desejados, e o conhecimento e desenvolvimento destes algoritmos é de grande valia. No entanto, existe uma grande quantidade de problemas computacionais para os quais não se conhece solução em tempo polinomial.

Consideremos um problema computacional arbitrário $\mathcal{P}$ e chamemos de $\mathscr{A}^{\mathcal{P}}$ a coleção de todos os algoritmos (conhecidos ou não) que resolvem o problema $\mathcal{P}$. Como dito anteriormente, se existe algum algoritmo eficiente A em $\mathscr{A}^{\mathcal{P}}$, dizemos que o problema é tratável. Por outro lado, se esta coleção for não vazia, mas nenhum dos seus algoritmos for eficiente, dizemos que o problema é *intratável*. Neste último caso, instâncias destes problemas podem ser, no máximo, *verificadas* em tempo polinomial. Esta verificação é realizada por

	Tamanho de Entrada				
	10	20	50	100	200
n^2	1/10000 segundo	1/2500 segundo	1/400 segundo	1/100 segundo	1/25 segundo
n^5	1/10 segundo	3.2 segundos	5.2 minutos	2.8 horas	3.7 dias
2^n	1/1000 segundo	1 segundo	35.7 anos	400+ trilhões de séculos	n^o. de 45 dígitos de séculos
n^n	2.8 horas	3.3 trilhões de anos	n^o. de 70 dígitos de séculos	n^o. de 185 dígitos de séculos	n^o. de 445 dígitos de séculos
Para comparação, o *Big Bang* foi há 12-15 bilhões de anos atrás					

Tabela 1: Comparação entre $T(n)$ para algoritmos polinomiais e exponenciais

um algoritmo, chamado de *verificador*, que recebe, como entrada, dois objetos: uma instância do problema e um conjunto de argumentos referente a esta instância, chamado de *certificado*. O algoritmo verificador pode devolver como saída apenas as respostas *sim* e *não*. Se a resposta for *sim*, dizemos que o verificador *aceitou* o certificado. Segundo P. Feofiloff [7], um verificador *polinomial* de um problema de decisão é tal que: (i) para cada instância do problema cuja resposta é *sim*, existe um certificado que o verificador aceita em tempo polinomial no tamanho da instância; e (ii) para cada instância cuja resposta é *não*, não existe certificado que o verificador aceite. Para ilustrar a definição, consideremos a versão de decisão do problema do caixeiro viajante que, em vez de pedir o menor percurso possível, pergunta apenas se existe algum percurso com custo total menor ou igual a um valor k especificado. Um possível verificador da resposta *sim* para este problema receberia como entrada uma instância arbitrária do problema e uma sequência das cidades $c_1, c_2, ..., c_n, c_1$. Seguindo a sequência dada, tal verificador soma as distâncias percorridas entre cidades consecutivas. Ao final, se a soma das distâncias for menor ou igual ao valor k especificado na entrada, cada cidade é visitada e a sequência termina na mesma cidade de origem, o verificador devolve *sim*, indicando que um algoritmo que decidisse este problema devolveria *sim* para esta instância. Ademais, note que a verificação deste certificado foi realizada de forma eficiente, isto é, em tempo polinomial no tamanho da entrada.

Definimos, então, a classe NP como aquela que compreende todos os problemas de decisão cuja resposta afirmativa pode ser verificada em tempo polinomial. Note que um problema $\mathcal{P}$ que pertence à P também pertence à NP - basta utilizar, como verificador de $\mathcal{P}$, um algoritmo eficiente existente para o problema, fornecendo como entrada a instância de $\mathcal{P}$ e um certificado qualquer. Logo, $\mathsf{P} \subseteq \mathsf{NP}$. O estudo de problemas pertencentes à classe NP é essencial, pois muitas situações do cotidiano podem ser modeladas computacionalmente usando problemas pertencentes a essa classe.

Considere, agora, $\mathcal{P}_1$ e $\mathcal{P}_2$ dois problemas de decisão. Suponha que se conheça um algoritmo A_2 para resolver o problema $\mathcal{P}_2$, e seja I_1 uma instância arbitrária do problema $\mathcal{P}_1$ com resposta R_1. Construímos uma função f que transforma a instância I_1 em uma instância do problema $\mathcal{P}_2$, que denotamos por I_2. Aplicamos, então, o algoritmo A_2 à instância I_2, obtendo a resposta R_2. Se a resposta R_1 é *sim* se e somente se a resposta R_2 for *sim*, então, usando estes passos, construímos um algoritmo A_1 para resolver o problema $\mathcal{P}_1$. Este algoritmo consiste em: (i) formular $I_2 = f(I_1)$; (ii) encontrar $R_2 = A_2(I_2)$; e (iii) obter $R_1 = R_2$. Chamamos este processo de transformação de problemas e mapeamento de soluções de *redução*. O diagrama da Figura 1 demonstra o processo de redução e criação do algoritmo A_1.

Se a transformação f for realizada em tempo polinomial, dizemos que $\mathcal{P}_1$ é *polinomialmente redutível* à $\mathcal{P}_2$. Denotamos tal redução por $\mathcal{P}_1 \preceq \mathcal{P}_2$. Essa relação entre problemas implica em duas consequências fundamentais: primeramente, se A_2 for um algoritmo polinomial, então $\mathcal{P}_1$ pode ser resolvido em tempo polinomial também, logo $\mathcal{P}_1 \in \mathsf{P}$; em segundo lugar, se tivermos uma noção de quão difícil é resolver $\mathcal{P}_1$, ou seja, se soubermos a classe de problemas à qual $\mathcal{P}_1$ pertence, e soubermos também que $\mathcal{P}_1 \preceq \mathcal{P}_2$, então resolver $\mathcal{P}_2$ é, no mínimo, tão difícil quanto resolver $\mathcal{P}_1$.

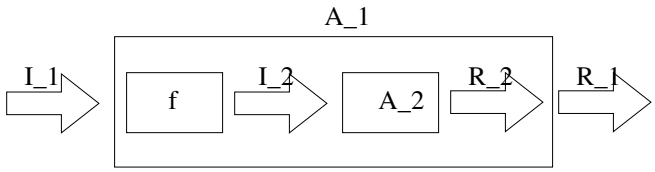


Figura 1: Diagrama que representa $\mathcal{P}_1 \preceq \mathcal{P}_2$.

Com este conceito em mente, um problema $\mathcal{P}$ é dito NP-*completo* se: (i) $\mathcal{P} \in \mathsf{NP}$; e (ii) para todo problema $\mathcal{P}' \in \mathsf{NP}$, $\mathcal{P}' \preceq \mathcal{P}$. A classe NP-completo contém os problemas considerados "mais difíceis" da classe NP, e esta dificuldade intrínseca é alvo de estudo de inúmeros pesquisadores. Ao provar que um determinado problema pertence a esta classe, provamos também que todos os problemas que pertencem à classe NP são polinomialmente redutíveis a ele. Dessa forma, se for encontrado um algoritmo A polinomial para resolver um problema $\mathcal{P} \in$ NP-completo qualquer, todos os problemas $\mathcal{P}' \in \mathsf{NP}$ também poderiam ser resolvidos eficientemente: reduzimos $\mathcal{P}' \preceq \mathcal{P}$ e aplicamos o algoritmo A para resolver $\mathcal{P}$ em tempo polinomial, assim resolvendo o problema original $\mathcal{P}'$. Isto provaria que $\mathsf{NP} \subseteq \mathsf{P}$ e, consequentemente, $\mathsf{P} = \mathsf{NP}$. Até hoje, porém, não se conhecem algoritmos polinomiais para nenhum problema da classe NP-completo.

A dificuldade de encontrar tais algoritmos reforça a possibilidade de que $\mathsf{P} \neq \mathsf{NP}$, abrindo um novo leque de pesquisa em teoria computacional para lidar com estes problemas, dada a grande quantidade de situações reais que podem ser associados a eles. Esta nova linha de pesquisa abrange Algoritmos de Aproximação, Algoritmos Probabilísticos, Algoritmos Genéticos, Heurísticas e Metaheurísticas, Computação Quântica e Molecular, entre muitos outros. Apesar de grandes avanços obtidos nestas áreas, o estudo da dificuldade intrínseca destes problemas ainda merece atenção, pois o questionamento "$\mathsf{P} = \mathsf{NP}$?" continua em aberto.

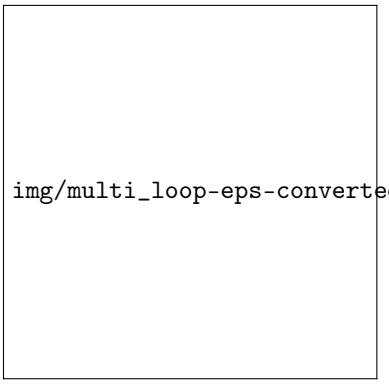
1.2 Teoria de Grafos

Muitos problemas de diversas áreas podem ser representados e interpretados como problemas em grafos. Como exemplos, citamos: o posicionamento de antenas de rádio para cobertura de espaços geográficos, o escalonamento de processos em um computador, a construção de rotas eficientes para distribuição de produtos, a alocação de registradores para variáveis em um programa computacional e a distribuição de faixas de rádiofrequência para emissoras. O estudo de Teoria de Grafos é essencial para uma melhor compreensão das propriedades e dificuldades de tais problemas, assim como para o desenvolvimento de algoritmos para solucioná-los.

Um *grafo* $G = (V(G), E(G))$ é um par ordenado tal que $V(G)$ é o seu conjunto de *vértices* e $E(G)$ é o seu conjunto de *arestas*, disjunto de $V(G)$; os conjuntos $V(G)$ e $E(G)$ são relacionados por uma *função de incidência* ψ_G que associa a cada aresta e um par não ordenado de vértices u e v de G (não necessariamente distintos). Dizemos, neste caso, que a aresta e *liga* os vértices u e v, e que u e v são *extremidades* de e. Os *elementos* do grafo são as suas arestas e os seus vértices. Definimos *adjacência* como uma relação entre elementos do mesmo conjunto. Havendo uma aresta ligando dois vértices u e v, estes vértices são chamados de *adjacentes*. De modo semelhante, se duas arestas possuem um mesmo vértice como extremidade, elas também são *adjacentes*. O conjunto de vértices adjacentes a um vértice v de G compõe a *vizinhança* de v, denotada por $N(v)$. Dizemos que uma aresta *incide* em um vértice (ou vice-versa) se ela possui este vértice como uma extremidade. Logo, *incidência* é uma relação entre elementos de conjuntos distintos.

Um *laço* é uma aresta que possui ambas as extremidades incidindo no mesmo vértice. Caso existam duas arestas $e, f \in E(G)$ tais que $\psi_G(e) = \psi_G(f)$, dizemos que e e f são *arestas múltiplas*. Um *grafo simples* é um grafo que não possui laços e arestas múltiplas, de modo que a aresta com $\psi_G(e) = \{u, v\}$ é única e

pode ser denotada por $e = uv$. A Figura 2 exibe representações visuais de dois grafos, um deles com arestas múltiplas e laços, e o outro simples. Na Figura 2(a), o grafo possui cinco vértices e oito arestas. Note que, neste exemplo, a função de incidência está representada implicitamente, podendo ser explicitada usando os nomes de seus vértices e arestas (como exemplos, citamos $\psi_G(e_1) = \{v_1, v_2\}$ e $\psi_G(e_7) = \{v_5, v_5\}$). Já na Figura 2(b), o grafo G, conhecido como *Grafo de Petersen*, é representado sem nomear seus vértices e arestas[1].



(a)

(b)

Figura 2: Alguns exemplos de grafos.

O *grau* de um vértice $d(v)$, $v \in V(G)$, é o número de vezes em que v é extremidade de arestas de G. Na Figura 2(a), observamos $d(v_4) = 3$, e $d(v_5) = 5$. Definimos o *grau máximo* do grafo G como $\Delta(G) := max\{d(v) : v \in V(G)\}$. Além disso, G é *k-regular* se $d(v) = k$, para todo $v \in V(G)$. De particular interesse é o caso em que $k = 3$, quando dizemos que o grafo G é *cúbico*. O Grafo de Petersen (Figura 2(b)) é um exemplo de grafo cúbico.

Um *passeio* em G é uma sequência alternada de vértices e arestas $P = v_0e_1v_1..e_kv_k$, tal que $\psi_G(e_i) = \{v_{i-1}, v_i\}, 1 \leq i \leq k$ e $v_j \in V(G), 0 \leq j \leq k$. Se não existem repetições de vértices em P, dizemos que P é um *caminho* em G. Se existir um caminho entre dois vértices u e v de G, dizemos que u e v estão *conectados*. O grafo G é dito *conexo* se, para todo par de vértices $u, v \in V(G)$, u e v estão conectados. Os subconjuntos maximais de vértices que satisfazem esta propriedade formam os conjuntos de vértices das *componentes conexas* de G, e o conjunto de arestas de cada componente é formado pelas arestas de G cujas extremidades são vértices pertencentes à componente.

Definimos uma família $\mathcal{F}$ de grafos como uma coleção de grafos G_i que satisfazem uma determinada propriedade. Como exemplos, todos os grafos cúbicos compõem uma família, ou todos os grafos conexos. A análise de famílias de grafos nos permite estender os resultados encontrados para um grafo $G_i \in \mathcal{F}$ a todos os grafos $G_j \in \mathcal{F}, i \neq j$. A seguir, definimos algumas famílias especiais de grafos que são utilizadas ao longo do texto.

Um *grafo completo*, K_n, é um grafo simples, com n vértices, tal que para qualquer par de vértices distintos $u, v \in V(G)$, existe a aresta $e = uv$. Um *ciclo* é um grafo simples G, com conjunto de vértices $V(G) = \{v_0, v_1, ..., v_{n-1}\}$ e conjunto de arestas $E(G) = \{v_0v_1, v_1v_2, ..., v_{n-2}v_{n-1}, v_{n-1}v_0\}$. Denotamos um ciclo com n vértices, $n \geq 3$, por C_n. Um *grafo bipartido* $G[X, Y]$ é um grafo cujo conjunto de vértices $V(G)$ é particionado em dois conjuntos $X, Y \subset V(G)$, de modo que toda aresta de G possui uma extremidade em X e a outra em Y. Finalmente, se um grafo bipartido $G[X, Y]$ tem todo vértice de X ligado a todos os vértices de Y, esse recebe o nome de *grafo bipartido completo*, e é denotado por $K_{m,n}$, com $m = |X|$ e $n = |Y|$.

[1] Grafos representados desta forma são denominados grafos *não rotulados*.

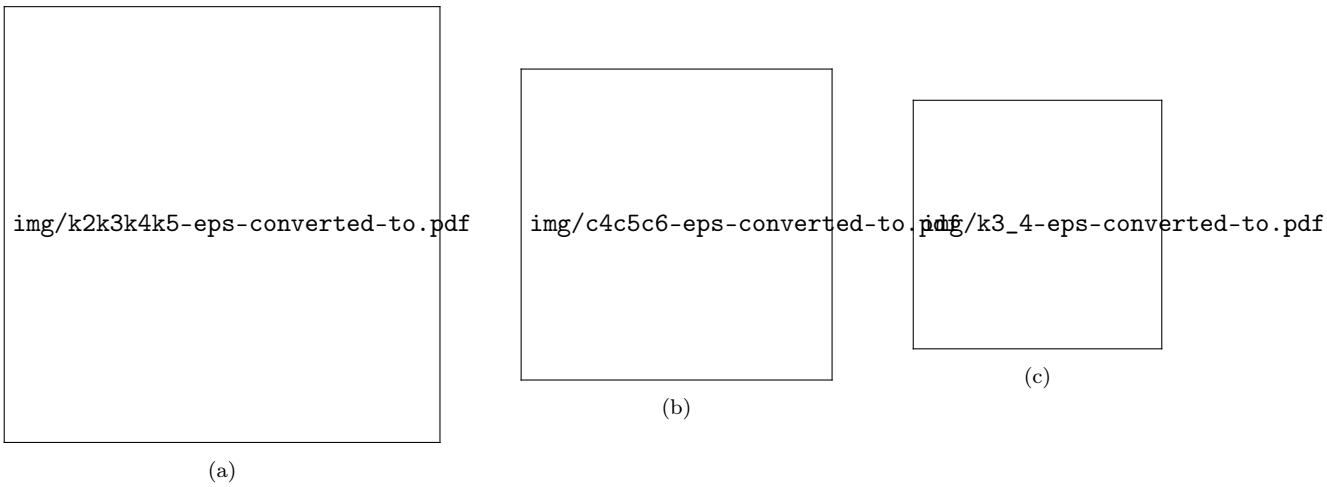


Figura 3: (a) Os grafos completos K_2, K_3, K_4, K_5; (b) os ciclos C_4, C_5, C_6; e (c) o grafo bipartido completo $K_{3,4}$ e suas partições X e Y. Note que o ciclo C_3 e o grafo K_3 são iguais.

Coloração é uma área de Teoria de Grafos que estuda a atribuição de *cores* a elementos do grafo, sujeitas a certas condições. Dado um grafo G, uma *coloração própria de vértices* é uma atribuição de cores aos vértices do grafo de modo que nenhum par de vértices adjacentes receba a mesma cor. Se o número de cores utilizadas para obter a coloração própria do grafo for menor ou igual a um inteiro k, esta recebe o nome de *k-coloração própria de vértices*. O menor número k de cores para o qual G admite uma k-coloração própria de vértices é chamado de *número cromático* de G, e é denotado por $\chi(G)$. O *problema da coloração própria de vértices* consiste em determinar $\chi(G)$ para um grafo G. A versão de decisão deste problema consiste em decidir se um grafo arbitrário G admite uma k-coloração própria de vértices. Denotamos tal problema de decisão por kCol. É um resultado conhecido, e importante, que o problema kCol, $k \geq 3$, é NP-completo [13]. Por outro lado, o problema 2Col pode ser resolvido em tempo polinomial[2], ou seja, 2Col $\in$ P. Problemas de coloração possuem diversas aplicações práticas. Dentre os exemplos citados anteriormente, a alocação de registradores para variáveis em programas computacionais e a atribuição de faixas de radiofrequência são aplicações deste problema.

Coloração de vértices pode ser visto como um problema de rotulação, em que *rótulos*, as cores, são atribuídos aos vértices do grafo. Na década de 1960, foi definido um outro tipo de rotulação de grafos [20]. Como em coloração, são atribuídos rótulos aos elementos do grafo. Agora, porém, estes rótulos são necessariamente numéricos e as condições que eles devem satisfazer envolvem propriedades entre estes números. Estas condições, geralmente funções matemáticas, podem ser interpretadas de maneira a *induzir* uma coloração própria de vértices. Para exemplificar, considere uma rotulação π das arestas de um grafo G, tal que $\pi : E(G) \to \{1, 2, 3\}$, e defina uma coloração $c_\pi : V(G) \to \mathbb{N}$, tal que $c_\pi(v) = \sum_{u \in N(v)} \pi(uv), v \in V(G)$. Neste caso, a rotulação π mapeia as arestas nos rótulos inteiros 1, 2 e 3, e a cor de cada vértice é dada pela soma dos rótulos das suas arestas incidentes. A Figura 4 apresenta um exemplo desta rotulação para o Grafo de Petersen.

Formalmente, definimos uma *rotulação própria* como um par ordenado (π, c_π), formado por um mapeamento $\pi : \mathcal{D} \to \mathcal{L}$, tal que $\mathcal{D}$ é o conjunto de elementos do grafo a ser rotulado, $\mathcal{L}$ é o conjunto de rótulos, acompanhado de uma coloração própria de vértices c_π, definida a partir da rotulação π. Em uma *rotulação*

[2]Decidir se um grafo arbitrário G admite uma 2-coloração própria pode ser feito por meio de uma adaptação do algoritmo de *busca em profundidade*, que executa em tempo linear em $|V(G)| + |E(G)|$ [4].

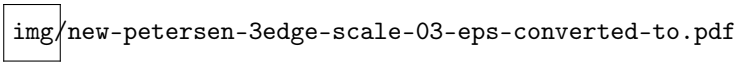


Figura 4: Grafo de Petersen com as arestas rotuladas com elementos do conjunto $\{1, 2, 3\}$ e coloração de vértices indicada pelos números dentro dos vértices.

própria de vértices, o domínio $\mathcal{D}$ é o conjunto de vértices $V(G)$. De modo semelhante, em uma *rotulação própria de arestas*, o domínio $\mathcal{D}$ é o conjunto de arestas $E(G)$. Finalmente, em uma *rotulação própria total*, o domínio $\mathcal{D}$ é dado por $V(G) \cup E(G)$. Deste ponto em diante, restringimos o uso da palavra rótulo aos elementos da imagem da função π, e da palavra cor, aos elementos da imagem da função c_π.

O problema da rotulação própria possui diversas variantes, dependendo de quais elementos do grafo são rotulados e de como a coloração própria é obtida - por soma, subtração, mínimo ou máximo, entre diversas outras funções. As propriedades de cada rotulação, suas aplicações em situações reais, a forma como as colorações são obtidas por meio de diferentes funções matemáticas e também a complexidade computacional de cada problema de rotulação têm despertado o interesse de muitos pesquisadores [19, 5, 6, 12, 1, 2].

O objetivo deste projeto de mestrado é o estudo aprofundado da análise de complexidade de tempo dos problemas de rotulação em grafos apresentados na Seção 2.

2 Tópicos e objetivos

Nesta seção, apresentamos alguns problemas de rotulação própria introduzidos nos artigos *On a 1,2 Conjecture*, de J. Przybyło e M. Woźniak [19], e *Algorithmic complexity of proper labeling problems*, de A. Dehghan et al. [5], que compreendem a bibliografia fundamental deste projeto. O primeiro artigo aborda um tipo específico de rotulação própria denominada rotulação total semiforte; o segundo artigo introduz diversas rotulações próprias interessantes, algumas das quais serão abordadas neste projeto de mestrado.

Nesta seção, inicialmente, apresentamos o problema de decisão associado a uma conjetura abordada no trabalho de Przybyło e Woźniak [19], denominada Conjetura 1,2, e alguns desdobramentos do seu estudo preliminar. Posteriomente, apresentamos o problema da k-coloração por gap, bem como alguns resultados iniciais obtidos para os grafos ciclo.

2.1 A Conjectura 1,2

Considere a seguinte rotulação própria de arestas (π, c_π), tal que $\pi : E(G) \to \{1, 2, ..., k\}$, e $c_\pi(v) = \sum_{uv \in E(G)} \pi(uv)$. Neste caso, dizemos que (π, c_π) é uma *k-rotulação de arestas semiforte*. O menor valor k para o qual G admite uma k-rotulação de arestas semiforte é denotado por $\mu(G)$. Esta rotulação foi proposta em 2004 por Karoński et al. [12]. Nesse trabalho, os autores provaram que todos os grafos 3-coloráveis, grafos completos e grafos com no máximo 10 vértices admitem uma 3-rotulação de arestas semiforte. Baseados nestes resultados, eles propuseram a seguinte conjectura:

Conjectura 1 (Conjectura 1,2,3)**.** *Todo grafo simples, sem componentes conexas isomorfas ao K_2, admite uma 3-rotulação de arestas semiforte.*

Entre os diversos resultados discutidos em seu artigo, Karoński et al. [12] perceberam que, para muitos dos grafos estudados, bastavam os rótulos 1 e 2 para obter a rotulação de arestas semiforte. Com isso em mente, os autores levantaram a seguinte questão: é verdade que todo grafo simples admite uma 2-rotulação de arestas semiforte? Em 2011, Dudek e Wajc [6] provaram que decidir se um grafo simples G admite uma

2-rotulação de arestas semiforte é um problema NP-completo.

Inspirados no problema da k-rotulação de arestas semiforte, em 2007, Przybyło e Woźniak [17] introduziram uma nova rotulação própria. Semelhante à rotulação de arestas apresentada anteriormente, a rotulação própria total (π, c_π) é dada por: $\pi : V(G) \cup E(G) \to \{1, 2, ..., k\}$, e $c_\pi(v) = \pi(v) + \sum_{uv \in E(G)} \pi(uv)$. Neste caso, dizemos que (π, c_π) é uma *k-rotulação total semiforte*. A Figura 5 exemplifica uma 2-rotulação total semiforte para o Grafo de Petersen.

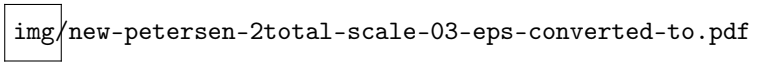


Figura 5: Uma 2-rotulação total semiforte do Grafo de Petersen. O valor dentro de cada vértice é sua cor, e os valores enquadrados são os rótulos das arestas e dos vértices.

O menor valor k para o qual um grafo admite uma k-rotulação total semiforte é denotado por $\tau(G)$. Este parâmetro possui forte relação com $\mu(G)$, como mostramos a seguir.

Lema 2. *Seja G um grafo simples. Se G admite uma k-rotulação de arestas semiforte, então G também admite uma k-rotulação total semiforte.*

Demonstração. Seja (π, c_π) uma k-rotulação de arestas semiforte de G. Construímos a k-rotulação total (π', c'_π) de G, tal que $\pi'(e) = \pi(e), e \in E(G)$, e $\pi'(v) = 1, v \in V(G)$. A coloração c'_π é própria, pois $c'_\pi(v) = c_\pi(v) + 1$ para todo $v \in V(G)$, e c_π é própria. □

Como corolário do resultado anterior, temos a seguinte relação entre $\tau(G)$ e $\mu(G)$.

Corolário 3. *Seja G um grafo simples. Então, $\tau(G) \leq \mu(G)$.* □

No mesmo trabalho em que introduziram a rotulação total semiforte, Przybyło e Woźniak [17] propuseram a seguinte conjectura:

Conjectura 4 (Conjectura 1,2)**.** *Todo grafo simples G admite uma 2-rotulação total semiforte.*

Os autores verificaram a conjectura para grafos 3-coloráveis, grafos completos e grafos 4-regulares. Também estabeleceram que $\tau(G) \leq min\{\lfloor \frac{\chi(G)}{2} \rfloor + 1, 11\}$. Um pouco depois, J. Przybło [18] mostrou que $\tau(G) \leq 7$ para grafos regulares. Em 2008, M. Kalkowski [10] anunciou que $\tau(G) \leq 3$, porém este trabalho não foi publicado. Apesar disso, é possível obter este resultado a partir de um lema, que se encontra em outro artigo publicado, posteriormente, em coautoria com M. Karónski e F. Pfender [11]. Atualmente, este é o melhor limitante superior para o parâmetro $\tau(G)$. Mais recentemente, A. Luiz et al. [15] verificaram a Conjectura 1,2 para duas classes de grafos. Desde que foi proposto, o problema da k-rotulação total semiforte tem atraído a atenção de pesquisadores da área, e limitantes superiores para $\tau(G)$ têm sido melhorados, porém a Conjectura 1,2 continua aberta para grafos arbitrários.

Sabe-se que todo grafo simples G admite uma 3-rotulação total semiforte. Além disso, a Conjectura 1,2 afirma que todo grafo simples G admite uma 2-rotulação total semiforte. Naturalmente, surge o questionamento se todo grafo simples G admite uma 1-rotulação total semiforte, qual seja: seria possível definir uma rotulação própria total de G utilizando apenas o rótulo 1? A Figura 6 apresenta um exemplo de grafo com uma 1-rotulação total semiforte.

Ao estudar esse problema, nota-se que, para cada vértice v de G, sua cor $c_\pi(v)$ é dada por $1 + d(v)$. Com base nestas observações, estabelecemos a propriedade estrutural a seguir.

Proposição 5. *Um grafo simples G admite uma 1-rotulação total semiforte se e somente se, para toda aresta $e = uv, e \in E(G)$, $d(u) \neq d(v)$.*

img/new-1total-scale-03-eps-converted-to.pdf

Figura 6: Um grafo com uma 1-rotulação total semiforte. O número no interior de cada vértice representa sua cor, e os valores enquadrados são os rótulos dos vértices e arestas.

Demonstração. ($\Rightarrow$) Seja G um grafo simples e seja (π, c_π) uma 1-rotulação total semiforte de G. Por construção, para todo $v \in V(G)$, $c_\pi(v) = 1 + d(v)$. Como c_π é uma coloração própria, para todo par de vértices adjacentes $u, v \in V(G)$, $c_\pi(u) \neq c_\pi(v)$. Portanto, $d(u) \neq d(v)$ para todo par de vértices adjacentes.

($\Leftarrow$) Seja G um grafo simples. Suponha que G não admita uma 1-rotulação total semiforte. Considere a atribuição π de rótulos 1 aos elementos do grafo G, e defina c_π como $c_\pi = \pi(v) + \sum_{uv \in E(G)} \pi(uv)$. Como (π, c_π) não é uma rotulação própria, então existem $u, v \in V(G)$ adjacentes tais que $c_\pi(u) = c_\pi(v)$. Por construção, $c_\pi(v) = 1 + d(v)$ para todo $v \in V(G)$. Então $d(u) = d(v)$ para u e v adjacentes, finalizando a demonstração. □

Motivados pela Proposição 5, no contexto deste projeto, de estudar a complexidade computacional de problemas de rotulação própria, elaboramos o Algoritmo 1, descrito a seguir, que decide se um grafo simples G admite uma 1-rotulação total semiforte.

Algoritmo 1 Decide se um grafo G admite uma 1-rotulação total semiforte.

```
1: para e = uv ∈ E(G) faça
2:     se d(u) = d(v) então
3:         devolve não.
4:     fim se
5: fim para
6: devolve sim.
```

A correção do Algoritmo 1 segue da Proposição 5, pois a linha 2 do algoritmo verifica se, para cada par de vértices adjacentes $u, v \in V(G)$, $d(u) = d(v)$. Se, em algum momento, esta verificação devolver verdadeiro, então o algoritmo para com resposta *não*, como esperado.

Quanto à complexidade computacional do Algoritmo 1, ao escolher uma estrutura de dados apropriada para armazenar o grafo G, seja por lista de adjacências ou por matriz de adjacências, podemos calcular $d(v)$ para todos os vértices $v \in V(G)$ de maneira eficiente, e armazená-los de forma que o acesso à $d(v)$ pelo algoritmo seja realizado em tempo constante. Como o laço da linha 1 é executado, no máximo, $|E(G)|$ vezes, o Algoritmo 1 executa em tempo polinomial no tamanho da entrada. Consequentemente, o problema de decidir se um grafo simples G admite uma 1-rotulação total semiforte - denotado a partir de agora por 1RoTSF - está em P. No contexto da Conjectura 1,2, resta considerar o problema de decidir se um grafo simples possui uma 2-rotulação total semiforte, definido a seguir.

Problema 6 (2RoTSF)**.** *Decidir se um grafo arbitrário G admite uma 2-rotulação total semiforte.*

É importante ressaltar a similaridade dos enunciados dos problemas 1RoTSF e 2RoTSF. Em computação, frequentemente observa-se que modificações aparentemente simples no enunciado de um problema podem gerar grandes diferenças quando analisamos as complexidades computacionais dos problemas derivados. Estas observações explicitam a sutileza da relação entre as classes de problemas P e NP. Para exemplificar, citamos o problema da satisfatibilidade booleana em suas versões 2SAT[3] e 3SAT[4], para os

[3] 2SAT: Sejam X um conjunto de literais, ϕ uma fórmula na forma normal conjuntiva, e C um conjunto de cláusulas, tal que cada cláusula tem exatamente dois literais. Existe uma atribuição verdade para X, tal que ϕ é satisfatível?

[4] 3SAT: Sejam X um conjunto de literais, ϕ uma fórmula na forma normal conjuntiva, e C um conjunto de cláusulas, tal

quais sabemos que 2SAT $\in$ P, e 3SAT $\in$ NP-completo. Retomemos o problema de coloração de grafos como outro exemplo, a saber 2Col $\in$ P, enquanto 3Col $\in$ NP-completo[5].

Pelas observações explicitadas no parágrafo anterior, apesar de sabermos que 1RoTSF $\in$ P, as semelhanças da 2-rotulação total semiforte com outros problemas de rotulação própria conhecidamente NP-completos [5, 6] levou-nos a conjecturar que 2RoTSF $\in$ NP-completo. Para provar tal conjectura, basta encontrar uma redução $\mathcal{P} \preceq$ 2RoTSF, sendo $\mathcal{P}$ um problema NP-completo. Conforme o problema foi sendo estudado, levantamos o seguinte questionamento: o que aconteceria com o problema 2RoTSF se a Conjectura 1,2 fosse verificada para todos os grafos? A discussão desta questão levou-nos a duas hipóteses: (i) 2RoTSF $\in$ NP-completo; e (ii) a Conjectura 1,2 é verdadeira. Analisamos estas hipóteses a seguir.

Supomos, a partir de agora, que a Conjectura 1,2 seja verdadeira, ou seja, todo grafo G admite uma 2-rotulação total semiforte.

Naturalmente, a resposta do problema de decisão 2RoTSF seria *sim* para toda instância. Sabemos, porém, que problemas que pertencem à classe NP-completo têm instâncias cuja resposta é *não*. Retomando novamente o problema do caixeiro viajante para nos auxiliar, existem instâncias deste problema para os quais não há nenhum percurso que, partindo de uma cidade inicial c_1, passa por todas as cidades fornecidas na entrada e retorna à cidade c_1 percorrendo uma distância menor ou igual ao valor k especificado na entrada. Outro exemplo de instância para o 3Col cuja resposta é *não* seria um grafo que não admite uma 3-coloração (o K_4, por exemplo). Sendo assim, e lembrando que a redução em tempo polinomial $\mathcal{P} \preceq$ 2RoTSF precisa devolver a resposta *sim* para $\mathcal{P}$ se e somente se a resposta de 2RoTSF for *sim*, seria impossível mapear a resposta *não* de $\mathcal{P}$ em uma resposta *não* de 2RoTSF, pois esta não existiria. Logo, 2RoTSF $\notin$ NP-completo.

Dessa forma, caso a conjectura seja verdadeira, é necessário, então, estudar um problema diferente associado à Conjectura 1,2 - um problema que permita instâncias *sim* e instâncias *não*. Voltemos nossa atenção para a 1-rotulação total semiforte novamente.

Por definição, toda 1-rotulação total semiforte é também uma 2-rotulação total semiforte. Ademais, a formulação do problema 1RoTSF permite instâncias que não são 1-rotuláveis, porém são 2-rotuláveis - o grafo K_2, por exemplo. Dessa forma, seria possível reduzir as instâncias *sim* e *não* de um problema conhecidamente NP-completo para as instâncias *sim* e *não* de 1RoTSF, respectivamente. Contudo, se fosse encontrada uma redução $\mathcal{P} \preceq$ 1RoTSF, então P = NP.

O resultado principal desta discussão é que nossas hipóteses iniciais são mutuamente exclusivas: não é possível que a Conjectura 1,2 seja válida ao mesmo tempo que o problema 2RoTSF $\in$ NP-completo. Ademais, se for encontrada uma redução $\mathcal{P} \preceq$ 2RoTSF ou $\mathcal{P} \preceq$ 1RoTSF, provamos que a Conjectura 1,2 é falsa, ou que P = NP, respectivamente. Dado que ambos os questionamentos continuam em aberto após anos de pesquisa, encontrar uma tal redução seria, no mínimo, tão difícil quanto desprovar a Conjectura 1,2, ou provar que P = NP.

2.2 Coloração por Gap

Uma *coloração por gap* é uma atribuição de rótulos inteiros positivos aos vértices de um grafo simples G, de maneira que, para cada $v \in V(G)$, a cor $c_\pi(v)$ é definida como a diferença (o *gap*) entre os rótulos de valor máximo e mínimo dos vértices adjacentes a v. Formalmente, a k-coloração por gap é um rotulação própria (π, c_π), tal que $\pi : V(G) \to \{1, 2, ..., k\}$, e $c_\pi(v) : V(G) \to \{0, 1, ..., k\}$, é dada por:

que cada cláusula tem exatamente três literais. Existe uma atribuição verdade para X, tal que ϕ é satisfatível?

[5]Para maiores informações sobre os problemas computacionais apresentados neste parágrafo, consultar o livro de M. R. Garey e D. S. Johnson [8].

$$c_\pi(v) = \begin{cases} 1, & \text{se } d(v) = 0; \\ \pi(u)_{u \in N(v)}, & \text{se } d(v) = 1; \\ \max\limits_{u \in N(v)} \{\pi(u)\} - \min\limits_{u \in N(v)} \{\pi(u)\}, & \text{se } d(v) \geq 2. \end{cases}$$

As Figuras 7(a) e 7(b) exibem uma 4-coloração por gap do C_3 e uma 2-coloração por gap do C_8, respectivamente. É importante ressaltar que nem todo grafo simples G admite uma k-coloração por gap. Por exemplo, o grafo completo K_n não admite tal rotulação para qualquer k, quando $n \geq 4$.

img/new-4gap-c3-2gap-c8-eps-converted-to.pdf

Figura 7: Uma 4-coloração por gap do C_3 e uma 2-coloração por gap do C_8. O valor no interior de cada vértice é sua cor c_π, e os valores enquadrados são os rótulos dos vértices.

A coloração por gap foi introduzida por A. Dehghan et al. [5] e é uma variante de uma rotulação própria definida por M. Tahraoui et al. [22] em 2012. Dehghan et al. [5] provaram que toda árvore T admite uma 2-coloração por gap. Os autores analisaram, também, a complexidade do problema de decisão associado a esta rotulação própria para algumas famílias de grafos, concluindo que é NP-completo: (i) decidir se um grafo bipartido admite uma 2-coloração por gap; (ii) para todo k, $k \geq 3$, decidir se um grafo admite uma k-coloração por gap; e (iii) decidir se um grafo 3-colorável admite uma 2-coloração por gap. Por outro lado, provaram que decidir se um grafo planar bipartido admite uma 2-coloração por gap pode ser resolvido eficientemente. Ainda nesse trabalho, os autores perguntam se existe um algoritmo polinomial que decide se um grafo admite uma k-coloração por gap.

Iniciamos nosso estudo da k-coloração por gap analisando os ciclos C_n, $n \geq 3$. Apresentamos, inicialmente, uma propriedade estrutural desta rotulação própria. Em seguida, mostramos os resultados obtidos para $k = 2$ e $k = 3$.

Propriedade 7. *Seja C_n, $n \geq 3$, e (π, c_π) uma k-coloração por gap do C_n. Então, c_π é uma k-coloração própria de vértices do C_n.*

Demonstração. Seja (π, c_π) uma k-coloração por gap do C_n. Para todo vértice $v \in V(C_n)$, temos $c_\pi(v) \neq k$, pois $d(v) = 2$ e o conjunto de rótulos é $\{1, 2, ..., k\}$. De fato, as cores possíveis para v são $\{0, 1, ..., k-1\}$. Então, c_π é uma k-coloração própria de vértices. □

Teorema 8. *Um ciclo C_n, $n \geq 3$, admite uma 2-coloração por gap se e somente se $n \equiv 0 \pmod 4$.*

Demonstração. Seja C_n, $n \geq 3$, com $V(C_n) = \{v_0, v_1, ..., v_{n-1}\}$. A demonstração está dividida em três casos de acordo com o valor de n mod 4. As operações nos índices dos vértices, quando ocorrem, são módulo n.

Caso 1. $n \equiv 1, 3 \pmod 4$.
Nesse caso, o ciclo C_n é ímpar. Dado que $\chi(C_n) = 3$, quando $n \equiv 1, 3 \pmod 4$, qualquer coloração por gap (π, c_π) usa, pelo menos, três cores. Entretanto, o número de cores em qualquer 2-coloração por gap do C_n é limitado a dois, pela Propriedade 7. Portanto, C_n não admite uma 2-coloração por gap quando $n \equiv 1, 3 \pmod 4$.

Caso 2. $n \equiv 2 \pmod 4$.
Suponha que exista uma 2-coloração por gap (π, c_π) para o C_n. Considerando as vizinhanças dos vértices do grafo C_n, e o fato de que apenas duas cores são usadas, concluímos que $c_\pi(v_i) \neq c_\pi(v_{i+1})$, para $0 \leq i \leq n-1$.

Em outras palavras, as cores se alternam nos vértices, percorrendo o grafo ciclicamente. Ajuste a notação para que $c_\pi(v_i) = 0$ se e somente se i for par.

Seja $v_i \in V(C_n),\ i \equiv 0 \pmod 2$. Como $c_\pi(v_i) = 0$, temos que $\pi(v_{i-1}) = \pi(v_{i+1})$ na rotulação π. Além disso, $N(v_i) \cap N(v_{i+2}) = \{v_{i+1}\}$, para todo $0 \leq i \leq n-1$. Portanto, existe $a \in \{1, 2\}$ tal que $\pi(v_j) = a$, para todo $j \equiv 1 \pmod 2$. Como $j \equiv 1 \pmod 2$, e sabemos que $c_\pi(v_i) = 0$ se e somente se i for par, $c_\pi(v_j) = 1$. Logo, na rotulação π, temos que $\pi(v_{j-1}) \neq \pi(v_{j+1})$. Analogamente ao caso dos vértices de índices ímpares, concluímos que os rótulos dos vértices de índices pares alternam seus valores ao longo do ciclo. Ajuste os índices para que $\pi(v_0) = a$. Isto implica que $\pi(v_i) = a$ se $i \equiv 0 \pmod 4$, e $\pi(v_i) = b$, $b \in \{1, 2\}$, $b \neq a$, se $i \equiv 2 \pmod 4$. Entretanto, como $n \equiv 2 \pmod 4$, então $\pi(v_{n-2}) = a$. Sendo assim, $c_\pi(v_{n-1}) = 0$, que é uma contradição, pois a notação foi ajustada para que $c_\pi(v_i) = 0$ se e somente se $i \equiv 0 \pmod 2$. A Figura 8 ilustra esta conclusão. Portanto, C_n não admite uma 2-coloração por gap quando $n \equiv 2 \pmod 4$.

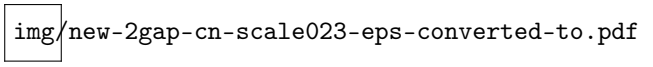


Figura 8: A contradição é observada no vértice v_{n-1}. O ajuste de notação define $c_\pi(v_{n-1}) = 1$. Entretanto, pelos rótulos $\pi(v_0)$ e $\pi(v_{n-2})$, sua cor $c_\pi(v_{n-1}) = 0$.

Caso 3. $n \equiv 0 \pmod 4$.
Considere a rotulação própria (π, c_π) do ciclo C_n como segue. Para todo $i \equiv 0 \pmod 4$, $\pi(v_i) = 2$, e para todo $i \equiv 1, 2, 3 \pmod 4$, $\pi(v_i) = 1$. A Figura 7(b) ilustra esta rotulação para o C_8. Note que, para todo v_i, i par, $\pi(v_{i-1}) = \pi(v_{i+1}) = 1$. Portanto, $c_\pi(v_i) = 0$. Analogamente, para todo v_j, j ímpar, $c_\pi(v_j) = 1$, pois $\{\pi(v_{j-1}), \pi(v_{j+1})\} = \{1, 2\}$. Logo, (π, c_π) é uma 2-coloração por gap do C_n para todo $n \equiv 0 \pmod 4$. □

Analisamos, agora, o problema da 3-coloração por gap de ciclos. Para uma 3-coloração por gap (π, c_π), por definição, o conjunto de rótulos é $\{1, 2, 3\}$ e, pela Propriedade 7, c_π é uma 3-coloração própria de vértices do C_n. Note, inicialmente, que o ciclo C_3 não admite uma 3-coloração por gap. Para ver isso, suponha que exista uma 3-coloração por gap de C_3. Então, $\{c_\pi(v_0), c_\pi(v_1), c_\pi(v_2)\} = \{0, 1, 2\}$. Ajuste a notação para que $c_\pi(v_0) = 0$. Então, temos que $\pi(v_1) = \pi(v_2)$. Sendo assim, $c_\pi(v_1) = |\pi(v_2) - \pi(v_0)| = |\pi(v_1) - \pi(v_0)| = c_\pi(v_2)$, o que é uma contradição pois v_1 e v_2 são adjacentes. Resta, então, considerar o problema da 3-coloração por gap dos ciclos C_n, $n \geq 4$.

Teorema 9. *Seja C_n, $n \geq 4$. Então, C_n admite uma 3-coloração por gap.*

Demonstração. Para demonstrar o resultado, provamos, por indução no número de vértices, a seguinte afirmação (mais forte).

Seja C_n, $n \geq 4$, e $V(C_n) = \{v_0, v_1, ..., v_{n-1}\}$. Então, existe uma 3-coloração por gap (π, c_π) do C_n tal que:

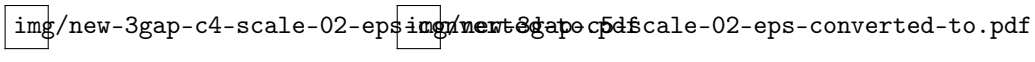
(i) $\pi(v_{n-2}) = 2$, $\pi(v_{n-1}) = 3$, $\pi(v_0) = 2$, e $c_\pi(v_{n-2}) = 2$, $c_\pi(v_{n-1}) = 0$ e $c_\pi(v_0) = 2$; ou

(ii) $\pi(v_{n-2}) = 3$, $\pi(v_{n-1}) = 1$, $\pi(v_0) = 3$, e $c_\pi(v_{n-2}) = 1$, $c_\pi(v_{n-1}) = 0$ e $c_\pi(v_0) = 1$; ou

(iii) $\pi(v_{n-2}) = 1$, $\pi(v_{n-1}) = 1$, $\pi(v_0) = 1$, e $c_\pi(v_{n-2}) = 2$, $c_\pi(v_{n-1}) = 0$ e $c_\pi(v_0) = 2$; ou

(iv) $\pi(v_{n-2}) = 1$, $\pi(v_{n-1}) = 2$, $\pi(v_0) = 1$, e $c_\pi(v_{n-2}) = 1$, $c_\pi(v_{n-1}) = 0$ e $c_\pi(v_0) = 1$.

A Figura 9 exibe uma 3-coloração por gap dos ciclos C_4 e C_5. Por inspeção, verificamos que ambas as colorações satisfazem a condição (iv).



(a) (b)

Figura 9: Os ciclos C_4 e C_5 e suas 3-colorações por gap em (a) e (b), respectivamente.

Considere um ciclo C_n, $n \geq 4$, e uma coloração por gap (π, c_π) que satisfaz uma das condições (i), (ii), (iii), (iv). Seja C_{n+2}, tal que $V(C_{n+2}) = \{v_0, v_1, ..., v_{n+1}\}$.

Defina $\pi' : V(C_{n+2}) \to \{1, 2, 3\}$, tal que:

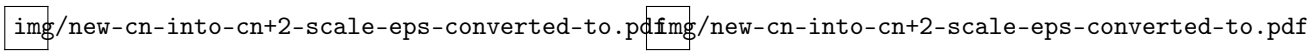
$$\pi'(v_i) = \begin{cases} \pi(v_i), & \text{se } 0 \leq i \leq n-2; \\ \pi(v_{n-1}), & \text{se } i = n-1, \text{ ou } i = n+1; \\ a, & \text{se } i = n. \end{cases}$$

O valor de a é dependente de qual das condições (i), (ii), (iii), (iv) é satisfeita e está explicitado a seguir.

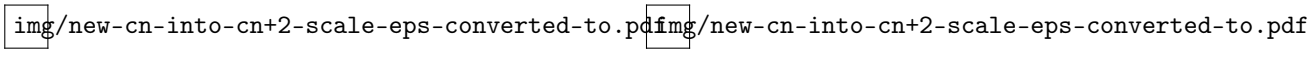
$$a = \begin{cases} 1, & \text{se a condição (i) ou (ii) é satisfeita;} \\ 2, & \text{se a condição (iii) é satisfeita;} \\ 3, & \text{se a condição (iv) é satisfeita.} \end{cases}$$

Defina a coloração c'_π de forma usual. Inicialmente, mostramos que (π', c'_π) é uma 3-coloração por gap do C_{n+2}.

Seja $w \in V(C_{n+2}) \backslash (N(v_{n-1}) \cup N(v_n) \cup N(v_{n+1}))$. Como $N_{C_{n+2}}(w) = N_{C_n}(w)$ e a rotulação π' preserva os rótulos de π para estes vértices, concluímos que $c'_\pi(w) = c_\pi(w)$. Isto implica que, para $v_i, v_{i+1} \in V(C_{n+2}) \backslash (N(v_{n-1}) \cup N(v_n) \cup N(v_{n+1}))$, temos $c'_\pi(v_i) \neq c'_\pi(v_{i+1})$. Consideremos, agora, $w \in N(v_{n-1}) \cup N(v_n) \cup N(v_{n+1})$. Estes vértices possuem seus rótulos explicitados na Figura 10, que detalha cada um dos quatro casos possíveis. Esta figura também exibe as cores dos vértices. Por inspeção, concluímos que a cor $c'_\pi(w)$ é diferente da cor de seus vizinhos.



(a) 3-coloração por gap referente à condição (i). (b) 3-coloração por gap referente à condição (ii).



(c) 3-coloração por gap referente à condição (iii). (d) 3-coloração por gap referente à condição (iv).

Figura 10

Para concluir a demonstração, precisamos mostrar que a rotulação própria (π', c'_π) do C_{n+2} satisfaz uma das condições dentre (i), (ii), (iii), (iv). Para isso, renomeie os vértices do ciclo C_{n+2} de modo que $v_i \leftarrow v_{(i+1) \bmod (n+2)}$. Observe que, após a renomeação dos vértices:

(a) Se C_n satisfaz (i), então C_{n+2} satisfaz (ii);

(b) Se C_n satisfaz (ii), então C_{n+2} satisfaz (iii);

(c) Se C_n satisfaz (iii), então C_{n+2} satisfaz (iv);

(d) Se C_n satisfaz (iv), então C_{n+2} satisfaz (i),

concluindo essa demonstração. □

Uma implicação imediata do Teorema 9 é a existência de um algoritmo polinomial para encontrar uma 3-coloração por gap do C_n, $n \geq 4$. Os resultados obtidos para os ciclos são produto dos nossos estudos preliminares a respeito do problema da k-coloração por gap, que possui propriedades muito interessantes. Devido a isso, pretendemos abordar este problema para outras classes de grafos.

Finalizamos esta seção analisando o problema da k-coloração por gap pelo viés da análise de complexidade. Consideremos, então, o problema de decidir se um determinado grafo G admite uma k-coloração por gap. Da bibliografia fundamental [5], sabemos que decidir se G admite uma 2-coloração por gap é NP-completo para grafos bipartidos e para grafos planares 3-coloráveis. Neste trabalho, os autores mostram, ainda, que, para todo $k \geq 3$, decidir se um grafo arbitrário G admite uma k-coloração por gap é NP-completo. Os autores, ainda, propõem a seguinte questão acerca da complexidade computacional da k-coloração por gap.

Problema 10. *Determinar a complexidade computacional de decidir se um determinado grafo G 3-regular e bipartido admite uma 2-coloração por gap.*

Considere o problema de decisão associado ao Problema 10, denominado por 3RB-2ColGap: decidir se um determinado grafo 3-regular e bipartido G admite uma 2-coloração por gap. De maneira semelhante à k-rotulação total semiforte, e dada a proximidade do enunciado do Problema 10 com diversos outros problemas de decisão NP-completos associados a rotulações próprias, conjecturamos que 3RB-2ColGap $\in$ NP-completo.

Inicialmente, provamos que 3RB-2ColGap $\in$ NP com o seguinte algoritmo verificador. Recebemos como entradas uma instância de 3RB-2ColGap e uma sequência de rótulos π_i para cada vértice v_i de G. Para todo $v \in V(G)$, calculamos sua cor $c_\pi(v)$ de acordo com a definição. Comparamos, para cada aresta $e = uv$, as cores $c_\pi(u)$ e $c_\pi(v)$. Finalmente, devolvemos *sim* se, para todo $u, v \in V(G)$ adjacentes, $c_\pi(u) \neq c_\pi(v)$, ou seja, G admite tal rotulação própria. Nota-se que este algoritmo verificador executa em tempo polinomial e, portanto, 3RB-2ColGap $\in$ NP. Portanto, resta achar uma redução para provar que 3RB-2ColGap $\in$ NP-completo, o que seria um resultado interessante para este problema.

3 Plano de trabalho e cronograma

O cronograma de trabalho foi elaborado considerando as exigências do programa de mestrado do Instituto de Computação da UNICAMP e está exposto na Tabela 2.

O aluno deve cursar cinco disciplinas ao longo do primeiro ano, sendo uma destas na área de Teoria de Computação. As disciplinas escolhidas em comum acordo com os orientadores foram: Biologia Computacional, Segurança de Computadores, Bancos de Dados (estas sendo escolhidas para cumprir as normas impostas pelo programa de pós-graduação do instituto), Complexidade de Algoritmos, Teoria de Grafos e Algoritmos Parametrizados (estas escolhidas para aprofundar os conceitos do aluno nas áreas do projeto de pesquisa). O aluno já havia cursado as três primeiras disciplinas como aluno especial (aluno regularmente matriculado, porém fora do programa de pós-graduação do Instituto), restando apenas duas a serem cursadas até o final de 2016. O aluno também participa semestralmente da série de seminários em Teoria da Computação.

O levantamento bibliográfico será realizado durante todo o decorrer do mestrado, de modo a sempre manter o aluno atualizado com o estado da arte dos problemas de rotulação e coloração em grafos. Como estes problemas possuem diversas variantes, é necessário um esforço constante para que o aluno se mantenha a par dos principais resultados sobre o tema. Artigos relacionados diretamente com os problemas de rotulação selecionados para aprofundamento terão prioridade, de modo que os avanços encontrados possam auxiliar

Cronograma de Execução									
Atividade	2016			2017					
	bimestre			bimestre					
	4	5	6	1	2	3	4	5	6
Disciplinas obrigatórias	•	•	•						
Levantamento bibliográfico	•	•	•	•	•	•	•	•	•
Estudo de problemas de rotulação própria	•	•	•	•	•	•	•	•	•
Estudo de artigos selecionados	•	•	•						
Preparação para Exame de Qualificação	•	•							
Programa de Estágio Docente (PED)				•	•	•			
Escrita da dissertação				•	•	•	•	•	•
Defesa									•

Tabela 2: Cronograma inicial de atividades

no desenvolvimento da pesquisa corrente.

O Instituto de Computação estimula a participação no Programa de Estágio Docente durante o período de mestrado. Os alunos atuam como monitores de uma disciplina oferecida para os alunos de graduação. Se possível, o objetivo é atuar como monitor da disciplina de MC558 - Projeto e Análise de Algoritmos II, que aborda conceitos básicos de Teoria de Grafos, problemas e algoritmos em grafos, tópicos em NP-completude e redução entre problemas. De modo a não conflitar com as disciplinas cursadas no primeiro ano e com o Exame de Qualificação, o período previsto para esta atividade é no primeiro semestre de 2017. Embora o programa não seja de caráter obrigatório, esta oportunidade tem muito a agregar na formação profissional e científica do aluno.

A escrita da dissertação está prevista para o ano de 2017, assim como também a escrita de artigos que consolidem resultados obtidos durante o período de pesquisa. A defesa está programada para o final do ano de 2017, ou início do ano de 2018. É importante ressaltar que, por ser um projeto de natureza teórica, diversos outros problemas e questionamentos relacionados aos tópicos tratados podem surgir. Caso seus resultados sejam considerados interessantes e promissores, estes poderão ser incorporados ao projeto.

4 Material e métodos

Para o desenvolvimento do projeto de pesquisa, o aluno necessitará de um ambiente para estudo (comunitário ou exclusivo), acesso à rede de computadores do IC-Unicamp e a material bibliográfico especializado. A infraestrutura oferecida pelo Instituto de Computação atende todas estas necessidades. O aluno terá acesso às bilbiotecas da universidade, que possuem um grande acervo de livros, teses, dissertações, revistas e periódicos. O Instituto de Computação também disponibiliza ao aluno uma cota mensal de impressão, facilitando o acesso a materiais impressos, quando necessário. Além disso, o aluno já é membro do Laboratório de Otimização e Combinatória (LOCo), com acesso à rede computacional do IC, assim como salas para estudo e reuniões.

Os métodos que serão utilizados no projeto são usuais nesta área. Os estudos serão realizados de forma individual, com reuniões semanais com os orientadores para discutir os avanços no projeto e o desenvolvimento do aluno nas disciplinas cursadas. Durante todo o período, o aluno participará da série de seminários, assim como assistirá à exames de qualificação de colegas, defesas de tese e dissertação, a fim de adquirir experiência e maturidade para a sua própria defesa.

5 Forma de análise dos resultados

Os problemas abordados são de natureza teórica, logo resultados encontrados também o serão, e, quando possível, serão divulgados em relatórios técnicos e artigos. Como este é um projeto de mestrado, objetiva-se principalmente que, ao longo destes dois anos, o aluno amadureça do ponto de vista científico. Este desenvolvimento deve se refletir na capacidade do aluno de extrair e interpretar novos resultados e conclusões de forma autônoma e independente. Em particular, espera-se que o aluno seja capaz de determinar a complexidade computacional de problemas de rotulação própria cuja dificuldade ainda é desconhecida.

Referências

[1] L. Addario-Berry, K. Dalal, C. McDiarmid, B. A. Reed and A. Thomason *Vertex-colouring edge-weightings* Combinatorica **24(1)** (2007), 1-12

[2] O. Baudon, J. Bensmail and E. Sopen, *On the complexity of determining the irregular chromatic index of a graph*, Journal of Discrete Algorithms **30** (2015), 113 - 127.

[3] J. A. Bondy and U. S. R. Murty, *Graph Theory*, volume 244 of *Graduate Texts in Mathematics*. Springer, London, 2008.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms, 3rd ed.*, The MIT Press, Massachussetts, USA, 2009.

[5] A. Dehghan, M.-R. Sadeghi and A. Ahad *Algorithmic complexity of proper labeling problems*, Theoretical Computer Science **495** (2013), 25 - 36.

[6] A. Dudek and D. Wajc, *On the complexity of vertex-coloring edge-weightings*, Discrete Mathematics and Theoretical Computer Science **13** (2011), 45-50.

[7] P. Feofiloff, *Complexidade computacional e problemas* NP*-completos*, `http://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/NPcompleto2.html`, 25-08-2016.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, USA, 1979.

[9] D. Harel, *Computers Ltd. - What they really can't do*, Oxford University Press, New York, USA, 2000.

[10] M. Kalkowski, Unpublished manuscript (2008).

[11] M. Kalkowski, M. Karoński and F. Pfender, *Vertex coloring edge weightings with integer weights at most 6*, Journal of Combinatorial Theory, Series B **100** (2010), 347-34

[12] M. Karoński, T. Luczak and A. Thomason, *Edge weights and vertex colours*, Journal of Combinatorial Theory, Series B, **91(1)** (2004), 151-157

[13] R. M. Karp, *Reductibility among combinatorial problems*, Complexity of Computer Computations (1972), 85-103.

[14] D. E. Knuth, *The Art of Computer Programming, 3rd ed.*, Addison-Wesley Publishing Company, USA, 1997.

[15] A. G. Luiz, C. N. Campos, S. Dantas, D. Sasaki, *The 1,2-Conjecture for powers of cycles*, Electronic Notes in Discrete Mathematics **50** (2015), 83 - 88.

[16] M. Pilśniak and M. Woźniak, *On the total-neighbor-distinguishing index of sums*, Graphs and Combinatorics **31** (2013), 771-782

[17] J. Przybyło and M. Woźniak, *1, 2 Conjecture, II*, Preprint MD **026** (2007)

[18] J. Przybyło, *A note on neighbour-distinguishing regular graphs total-weighting*, The Electronic Journal of Combinatorics **15** (2008), #N35

[19] J. Przybyło and M. Woźniak, *On a 1, 2 Conjecture*, Discrete Mathematics and Theoretical Computer Science **12** (2010), 101-108

[20] A. Rosa, *On certain valuations of the vertices of a graph*, Artigo, McMaster University (1967)

[21] J. Skowronek-Kaziów, *Multiplicative vertex-colouring weightings of graphs*, Information Processing Letters **112** (2012), 191-194

[22] M. A. Tahraoui, E. Duchene, H. Kheddouci, *Gap vertex-distinguishing edge colorings of graphs*, Discrete Mathematics **312** (20) (2012), 3011 - 3025.