IMPLEMENTAÇÃO E ANÁLISE DE ALGORITMOS PARA COLORAÇÃO DE ARESTAS

TIAGO DE OLIVEIRA JANUARIO

IMPLEMENTAÇÃO E ANÁLISE DE ALGORITMOS PARA COLORAÇÃO DE ARESTAS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Sebastián Alberto Urrutia

Belo Horizonte

Março de 2011

Januario, Tiago de Oliveira
J35i Implementação e Análise de Algoritmos para Coloração de Arestas / Tiago de Oliveira Januario. — Belo Horizonte, 2011
xx, 47 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Sebastián Alberto Urrutia

1. Computação. 2. Teoria dos Grafos-Teses. I. Orientador. II. Título.

CDU 519.6*62 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Implementação e análise de algoritmos para coloração de arestas

TIAGO DE OLIVEIRA JANUARIO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. SEBASTIÁN ALBERTO URRUTIA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. HAROLDO GAMBINI SANTOS
Departamento de Computação - UFOP

PROF. ANTONIO ALFREDO FERREIRA LOUREIRO
Departamento de Ciência da Computação - UFMG

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 18 de março de 2011.

Agradecimentos

Agradeço a Deus pela vida. Agradeço a meus pais por cuidarem de mim mesmo de longe e por estarem sempre presentes e acompanhando meus passos. Agradeço por ter duas irmãs tão lindas que sempre estão disponíveis quando eu preciso de uma orelha para puxar.

Agradeço ao meu orientador Sebastián, pela amizade, confiança, paciência, incentivos, compreenção nos momentos difíceis, ensinamentos e pela oportunidade de desenvolver este trabalho.

Ao Zilton, pelas noites em claro jogando conversa fora e tomando tereré, por todas as vezes em que saímos juntos e gastei todo o meu dinheiro, pelo show do *a-ha* que eu não vi, e pela honra de ter morado com um baiano tão gente fina.

Ao Mayron, que foi uma das pessoas que me motivou a estar aqui, com quem passei boa parte da graduação trabalhando, que sempre foi para mim um exemplo de dedicação e disciplina.

Aos professores do Departamento de Informática da Universidade Federal de Viçosa, que me forneceram uma forte base de conhecimento que me ajudou a chegar até aqui, em especial, aos professores André Gustavo e José Elias, pela amizade, companherismo e confiança em mim depositada.

Aos meus companheiros do LaPO, com quem passei maior parte do meu tempo ao longo desses dois anos de estrada.

A todos os amigos que fiz em Belo Horizonte, por todos os momentos de alegria e também a todos os amigos que fiz nos eventos dos quais participei durante o mestrado.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Universidade Federal de Minas Gerais e ao Departamento de Ciência da Computação, pelo apoio financeiro, por terem me acolhido e me proporcionado uma excelente infraestrutura de estudo e pesquisa.

A todos vocês, muito obrigado.

T.O.J.

Resumo

O problema de Coloração de Arestas, extensivamente estudado em Teoria dos Grafos, consiste em colorir as arestas de um grafo de tal forma que arestas incidentes em um mesmo vértice tenham cores distintas e que o número de cores utilizadas seja o menor possível.

O resultado mais importante a respeito do problema de coloração de arestas surgiu em 1964 com o Teorema de Vizing, que definiu que o número mínimo de cores necessárias para colorir um grafo, denominado de índice cromático χ', está limitado entre os valores Δ e $\Delta + 1$, onde Δ é o grau máximo do grafo.

Neste trabalho são apresentados duas implementações eficientes para coloração de arestas de grafos simples, baseados no Teorema de Vizing, utilizando não mais que $\Delta + 1$ cores. Várias adaptações em estruturas de dados e na ordem de processamento das arestas e cores foram propostas com o intuito de reduzir o tempo de execução das operações realizadas com maior frequência. Também foram desenvolvidas duas estratégias de pré-processamento que fornecem um grafo parcialmente colorido como entrada para o algoritmo de coloração de arestas. Os resultados experimentais mostram que as estratégias desenvolvidas permitem uma redução de 63,92% no tempo de execução das implementações dos algoritmos de coloração de arestas aqui estudados.

Palavras-chave: Teoria dos Grafos, Coloração de Arestas, Teorema de Vizing, Implementações eficientes.

Abstract

The Edge Coloring Problem, extensively studied in Graph Theory, concerns coloring the edges of a graph such that edges incident on the same vertex have distinct colors and the number of used colors is minimized.

The most important result on the edge coloring problem was proposed in 1964 with Vizing's Theorem, which established that the minimum number of colors needed to color a graph, the chromatic index χ', is limited between the values Δ and $\Delta + 1$, where Δ is the maximum degree of the graph.

This work presents two efficient edge coloring implementations for simple graphs based on Vizing's Theorem using no more than $\Delta + 1$ colors. Several adaptations in data structures and in the processing order of edges and colors have been proposed in order to reduce the runtime of the operations more frequently performed. We also developed two preprocessing strategies that provide a partially colored graph as input to the edge coloring algorithm. Experimental results show that the proposed strategies allow gains up to 63.92% in terms of performance in the edge coloring algorithms studied here.

Keywords: Graph theory, Edge coloring, Vizing's Theorem, Efficient implementations.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Capítulo 1

Introdução

O problema de Coloração de Arestas, extensivamente estudado em Teoria do Grafos, consiste em colorir as arestas de um grafo de tal forma que arestas incidentes em um mesmo vértice tenham cores distintas e que o número de cores utilizadas seja o menor possível. Soluções para problemas reais podem ser obtidas com o auxílio da coloração de arestas, de tal forma que os vértices podem representar: processadores, professores e alunos, sensores, equipes esportivas, etc., enquanto as cores das arestas podem representar: processos a serem executados, salas de aula, proximidades, estádios, etc.

O seguinte exemplo, ilustrado pela Figura 1.1, pode ser modelado através de coloração de arestas: suponha que seja necessário organizar uma série de pequenas reuniões, entre duas pessoas, em um grupo com dez pessoas. Com o objetivo de eliminar qualquer conflito de horários, todas as entrevistas poderiam ser agendadas em momentos distintos, no entanto, menos recursos serão desperdiçados (salas de reuniões,

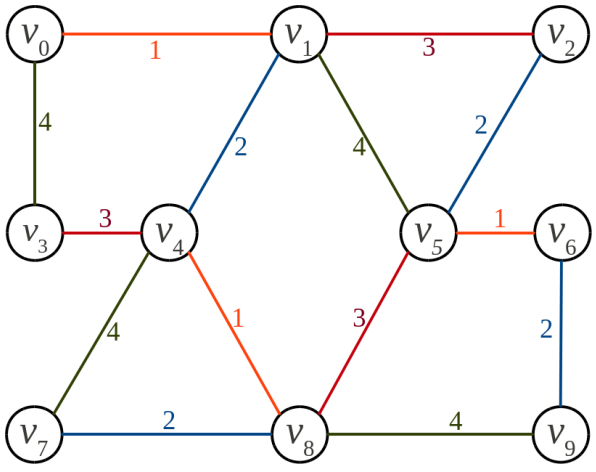


Figura 1.1. Solução ótima para o exemplo do problema de agendamento de reuniões modelado através de coloração de arestas.

horários, etc.) se várias entrevistas forem realizadas ao mesmo tempo. Neste caso, seja um grafo cujos vértices são as pessoas e cujas arestas representam as reuniões a serem agendadas. Uma coloração de arestas deste grafo define a agenda de reuniões, tal que as diferentes cores representam diferentes horários no cronograma, com todas as reuniões da mesma cor acontecendo simultaneamente. A Figura 1.1 apresenta uma solução ótima para o problema, onde nesse resultado, são necessários quatro horários, representados pelos números de 1 a 4, para agendar as reuniões entre as dez pessoas em questão, tomadas duas a duas.

O resultado mais importante a respeito do problema de coloração de arestas surgiu em 1964 com o Teorema de Vizing, que definiu que o número mínimo de cores necessárias para colorir um grafo, denominado de índice cromático χ', está limitado entre os valores Δ e $\Delta + 1$, onde Δ é o grau máximo do grafo.

A partir do Teorema de Vizing é possível classificar os grafos em Classe 1 ou Classe 2. Um grafo é dito Classe 1 se $\chi' = \Delta$, e dito Classe 2 se $\chi' = \Delta + 1$. O Teorema de Vizing impõe claramente que não existe outra possibilidade. Decidir a qual classe um grafo pertence é um problema de classificação. Para determinar se o grafo G pertence à Classe 1, primeiro procura-se uma coloração que utilize apenas Δ cores para G. Se esta coloração é encontrada, então G pertence à Classe 1. No entanto, encontrar uma coloração que utilize $\Delta + 1$ cores não garante que o grafo G pertença à Classe 2, uma vez que qualquer grafo que pertence à Classe 1 também pode utilizar uma cor extra em sua coloração e assim ser colorido com $\Delta + 1$ cores. Portanto, para provar que um grafo pertence à Classe 2, é preciso provar que este grafo não pertence à Classe 1. Holyer [1981] demonstra que o problema de classificação é $\mathcal{NP}$-Completo, o que torna a busca por algoritmos polinomiais cada vez mais eficientes na obtenção de uma coloração com $\Delta + 1$ cores, um tema de interesse para a comunidade acadêmica.

A demonstração construtiva do Teorema de Vizing utiliza indução na coloração das arestas do grafo. Essa demonstração serviu como base para vários algoritmos polinomiais de coloração de arestas que utilizam no máximo $\Delta + 1$ cores. Gabow et al. [1985] apresentam uma revisão da bibliografia sobre o tema, com os melhores algoritmos de coloração de arestas, suas evoluções e um estudo teórico quanto à análise de suas complexidades assintóticas. Porém, mesmo após vários anos de pesquisa a respeito desse tema, até o momento não se conhecem resultados experimentais expressivos para o problema de coloração de arestas de grafos simples que utilize no máximo $\Delta + 1$ cores, ilustrando os reais desempenhos dos algoritmos encontrados na literatura. Tão pouco se tem conhecimento a respeito de publicações de estratégias de execução ou heurísticas que poderiam ser utilizadas para se obter melhores resultados com relação aos tempos de execução de suas implementações. Além disso, não há sinais de que

os algoritmos propostos na literatura tenham sido testados para grafos com grandes valores de número de vértices ou arestas.

Este trabalho apresenta propostas de adaptações em estruturas de dados utilizadas para o armazenamento de informações sobre as cores, vértices e arestas dos grafos. Diversas análises experimentais realizadas consideraram a ordem de escolha das arestas e das cores para investigar a influência destas variáveis no tempo de execução dos algoritmos. Também foram desenvolvidas duas estratégias de pré-processamento que fornecem um grafo parcialmente colorido como entrada para o algoritmo de coloração de arestas.

1.1 Objetivos e Resultados desta Dissertação

Nesta dissertação, o problema de coloração de arestas com $\Delta + 1$ cores para grafos simples é estudado e tem-se como objetivo propor uma implementação mais eficiente de algoritmo de coloração de arestas para resolver este problema. Foram propostas várias versões de um algoritmo de coloração de arestas baseadas no Teorema de Vizing. Com o objetivo de executar as operações mais frequentes com maior eficiência, foram projetadas estruturas de dados adaptadas para o problema a fim de reduzir o tempo de execução dessas operações em várias etapas do algoritmo implementado. Também foram desenvolvidas duas estratégias de pré-processamento para obter um grafo parcialmente colorido como passo inicial do algoritmo de coloração.

Os resultados obtidos pelas implementações apresentadas neste trabalho foram experimentalmente comparadas com os resultados das implementações disponíveis na literatura e obtiveram um desempenho expressivo em termos de tempo de execução. Os resultados experimentais mostram que a combinação das estratégias de pré-processamento e o projeto das estruturas de dados para resolver o problema em questão permitem uma redução de 63,92% no tempo de execução das implementações dos algoritmos de coloração de arestas aqui estudados.

1.2 Organização do Texto

Neste capítulo foi apresentada uma breve introdução do assunto a ser tratado no restante desta dissertação. O Capítulo 2 apresenta as principais definições necessárias para o bom entendimento do texto, uma revisão bibliográfica atualizada do problema de coloração de arestas com diversas variações da formulação do problema e, finalmente, a demonstração tradicional do Teorema de Vizing.

O Capítulo 3 apresenta todos os detalhes do algoritmo de coloração de arestas proposto, a implementação das estruturas de dados e seus métodos de atualização, as estratégias de execução e de pré-processamento e, por fim, um comentário a respeito das implementações disponíveis na literatura.

Resultados computacionais, descrição das instâncias de teste e análises experimentais do algoritmo proposto são apresentadas no Capítulo 4.

Conclusões a respeito das principais contribuições deste trabalho, juntamente com propostas para atividades futuras, são apresentadas no Capítulo 5.

Capítulo 2

Coloração de Arestas e o Teorema de Vizing

Este capítulo apresenta os principais conceitos em Teoria dos Grafos, necessários para o entendimento do trabalho apresentado nesta dissertação, relacionados ao problema de coloração de arestas, incluindo a demonstração do Teorema de Vizing que, além do principal resultado para os limites de coloração, foi utilizada como base para o desenvolvimento dos algoritmos aqui estudados. Este capítulo também expõe uma revisão dos trabalhos mais importantes relacionados ao problema clássico de coloração de arestas e suas diversas variações.

2.1 Definições em Teoria dos Grafos

As definições apresentadas abaixo, que serão utilizadas ao longo do texto, podem ser encontradas em West [2001] e Diestel [2005].

Definição 1 *$G(V, E)$, ou simplesmente G, denota um grafo simples, sem laço, sem múltiplas arestas, não direcionado e finito. $V(G)$ e $E(G)$ representam os conjuntos de* ***vértices*** *e* ***arestas*** *de G, respectivamente. Dois vértices são ditos* ***adjacentes*** *quando existe uma aresta que os conecta. Para fins de simplicidade, serão utilizados os termos V e E quando o grafo em questão estiver implícito. Será utilizada a letra n para representar a cardinalidade do conjunto de vértices $|V|$ e a letra m para representar a cardinalidade do conjunto de arestas $|E|$.*

Definição 2 *Um* ***multigrafo*** *é um grafo que pode possuir até μ arestas conectando cada par de vértices, onde a letra μ é denominada* ***multiplicidade*** *do grafo. Em um grafo simples, $\mu = 1$.*

Definição 3 *Um* ***subgrafo*** *de G é um grafo G' com $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$.*

Definição 4 *Para cada vértice $v \in V$, $Adj(v)$ denota o conjunto de vértices que são adjacentes a v.*

Definição 5 *O* ***grau*** *de um vértice v é $\delta(v) = |Adj(v)|$. O* ***grau máximo*** *de um vértice é $\Delta(G) = max_{v \in V(G)}\{d(v)\}$, ou simplesmente Δ quando o grafo em questão estiver implícito. Um vértice é dito* ***universal*** *se este vértice está conectado a todos os demais vértices do grafo.*

Definição 6 *Uma atribuição de cores às arestas de um grafo G, ou simplesmente* ***coloração****, é uma função de coloração $\lambda : E \to C$. Os elementos do conjunto C são chamados de cores.*

Definição 7 *Uma cor $c \in C$* ***incide*** *em um vértice v se existe um vértice $w \in Adj(v)$ tal que a aresta $e(v, w)$ é colorida com a cor c, ou seja, $\lambda(e(v, w)) = c$, caso contrário a cor encontra-se* ***disponível*** *no vértice v. Um conflito em uma atribuição de cores ocorre quando duas cores incidem em um vértice comum.*

Definição 8 *Um vértice u é dito ser* ***satisfeito*** *quando $\lambda(uv) = \lambda(uw)$ implica em $v = w$, para todas as arestas em $Adj(u)$. Uma coloração de arestas é uma atribuição de cores de forma que todo vértice é satisfeito, ou de forma equivalente, que não existam conflitos.*

Definição 9 *Uma* ***coloração válida*** *de G é um atribuição de cores a todas as arestas, sem conflitos. Uma* ***coloração parcial*** *é uma coloração válida onde algumas arestas de G se encontrarão descoloridas. Um grafo é dito* ***parcialmente colorido*** *se esse possui uma coloração parcial.*

Definição 10 *O número mínimo de cores necessárias para produzir uma coloração de arestas válida para G é chamado de* ***índice cromático****, definido por $\chi'(G)$.*

Definição 11 *$H \subseteq G$ é um subgrafo induzido pelo conjunto de arestas $E(H) \subseteq E(G)$, se o seu conjunto de vértices é dado pelos vértices que são extremidades das arestas em $E(H)$, e o seu conjunto de arestas é dado por $E(H)$. O grafo $H(\alpha, \beta) \subseteq G$ é um subgrafo induzido pelas arestas coloridas com as cores α e β.*

Definição 12 *Um $\alpha/\beta - path \in H(\alpha, \beta)$ é um caminho simples ou um ciclo de tamanho par formado pelas arestas de cores α e β. A troca das cores ao longo desse caminho (β por α e α por β) é chamada de* ***inversão****. Se a coloração inicial de G é válida, a mesma continua válida após a inversão de um $\alpha/\beta - path$ Misra & Gries [1992].*

Tabela 2.1. Principais contribuições de algoritmos para coloração de arestas disponíveis na literatura.

Autores	**Classes de Grafos**	**Número de Cores**	Δ	**Complexidade**
Skulrattanakulchai [2002]	Planares	$\Delta + 1$	3	$\mathcal{O}(n)$
Gabow et al. [1985]	Planares	$\Delta + 1$	$4, 5, 6, 7$	$\mathcal{O}(n \cdot \log n)$
Cole & Kowalik [2008]	Planares	$\Delta + 2$	$4, 5, 6, 7$	$\mathcal{O}(n)$
Gabow et al. [1985]	Planares	Δ	8	$\mathcal{O}(n^2)$
Cole & Kowalik [2008]	Planares	$\Delta + 1$	8	$\mathcal{O}(n)$
Cole & Kowalik [2008]	Planares	Δ	≥ 9	$\mathcal{O}(n)$
Chrobak & Yung [1989]	Planares	Δ	≥ 19	$\mathcal{O}(n)$
Gabow et al. [1985]	Simples	$\Delta + 1$	-	$\mathcal{O}(m \cdot \sqrt{n \cdot \log n})$
Takabatake [2005]	Bipartidos	Δ	-	$\mathcal{O}(m \cdot \log \Delta + (\frac{m}{\Delta}) log(\frac{m}{\Delta}))$
Kirkman [1847]	K_n, n é par	Δ	$n - 1$	$\mathcal{O}(m)$
Kirkman [1847]	K_n, n é ímpar	$\Delta + 1$	$n - 1$	$\mathcal{O}(m)$
Januario [2011]	Com nó universal	$\Delta + 1$	$n - 1$	$\mathcal{O}(m)$

2.2 Trabalhos Relacionados

Como provado por Holyer [1981], determinar o *índice cromático* de um grafo é um problema $\mathcal{NP}$-Completo, e assim é incerto dizer se existe um algoritmo em tempo polinomial que o computa. Portanto, um algoritmo polinomial que dê um bom limite superior para χ' pode ser muito útil, uma vez que a busca por algoritmos computacionais eficientes e de baixa complexidade para problemas em grafos é muito importantes e tem sido tema de pesquisa há vários anos.

De acordo com Scheide & Stiebitz [2010], a grande maioria dos limites superiores[1] para o *índice cromático*, são dados pelos próprios algoritmos de coloração de arestas. Em geral, esses algoritmos são projetados para classes específicas de grafos. Shannon

[1] O limite inferior para a coloração de arestas de um grafo G é dado por Δ. A prova deste limite inferior é trivial, pois são necessárias Δ diferentes cores para colorir as arestas incidentes nos vértices de maior grau do grafo.

[1949] provou que as arestas de qualquer grafo, incluindo multigrafos, podem ser coloridas com no máximo $\frac{3\cdot\Delta}{2}$, ou seja $\chi' \leq \frac{3\cdot\Delta}{2}$. Vizing [1964] provou que $\chi' \leq \Delta + \mu$ para qualquer grafo. Neste trabalho, os grafos estudados são todos simples, ou seja, um caso particular de multigrafos onde $\mu = 1$.

Um grafo simples é dito da Classe 1 se $\chi' = \Delta$ e da Classe 2 se $\chi' = \Delta + 1$. O Teorema de Vizing define que não existe outra possibilidade: todo grafo pertence ou à Classe 1, ou à Classe 2. A Tabela 2.1 apresenta os trabalhos mais importantes e suas contribuições quanto aos valores do número de cores utilizadas na coloração para várias classes de grafos e as complexidades assintóticas dos seus algoritmos de coloração de arestas.

Como pode ser visto pela Tabela 2.1, a coloração de arestas em grafos planares tem sido alvo de estudo por muitos pesquisadores. Vários investigações já foram desenvolvidas para esses grafos, com o objetivo de classificá-los quanto à sua classe de coloração. É sabido que existem grafos planares com $\Delta \in \{2, 3, 4, 5\}$, que pertencem à Classe 2. Sanders & Zhao [2001] provaram que todos os grafos planares com $\Delta = 7$ pertencem à Classe 1. Determinar a existência de um grafo planar com $\Delta = 6$ que pertença à Classe 2 é um problema que permanece em aberto. A classificação desses grafos permite guiar as pesquisas em direção à busca de algoritmos mais eficientes, considerando os novos valores de limites inferiores e superiores para χ'. Por outro lado, em certas aplicações, é necessário escolher entre eficiência ou qualidade da solução. Para grafos planares com $\Delta = 8$, Gabow et al. [1985] propuseram um algoritmo de coloração de arestas utilizando Δ cores, enquanto Cole & Kowalik [2008] propuseram, para o mesmo tipo de grafo, um algoritmo de coloração de arestas mais eficiente mas que utiliza no máximo $\Delta + 1$ cores.

Para um caso particular de grafos planares, grafos Teia, Sharebaf [2009] apresenta um algoritmo de coloração de arestas que utiliza Δ cores, sem verificar as cores das arestas incidentes em qualquer vértice do grafo. Este algoritmo tem como base a soma dos índices associados aos vértices conectados por uma aresta que será colorida. Esta soma é calculada em módulo k, onde k é o número de pernas do grafo Teia[2].

Para grafos bipartidos, Takabatake [2005] propôs um algoritmo de coloração de arestas com complexidade $\mathcal{O}(m \cdot \log \Delta + (\frac{m}{\Delta}) log(\frac{m}{\Delta}))$, a menor complexidade conhecida. Gabow et al. [1985] apresentaram um algoritmo para coloração de arestas de grafos

[2]Uma árvore $T = (V, E)$ é uma Teia Base quando existe no máximo um vértice de grau maior do que dois, denominado raiz. Uma perna da Teia Base é um caminho da raiz a um vértice de grau 1. Um Grafo Teia, T_k , com $k \geq 3$, é uma teia base T com k pernas p_1,..., p_k , sendo que cada perna p_i , $i = 1, \ldots, k$, tem tamanho pelo menos igual a dois. Além disso, dois vértices em pernas distintas, $v \in p_i$ e $u \in p_j$, com i = j, são adjacentes quando $|i - j| \in \{1, k - 1\}$ e a distância $d(v, c) = d(u, c)$, onde c é a raiz do grafo T.

simples com no máximo $\Delta + 1$ cores de complexidade $\mathcal{O}(m \cdot \sqrt{n \cdot \log n})$, no entanto até o momento nenhuma implementação desse algoritmo é relatada na literatura.

De acordo com Froncek [2010], o Método do Polígono, descoberto por Kirkman [1847], é o método mais conhecido e amplamente aplicado no planejamento de tabelas de jogos esportivos (ainda que não tenha sido desenvolvido para esse fim). Com esse método é possível obter uma coloração de arestas para grafos completos, K_n, onde n representa o número de vértices do grafo, utilizando Δ cores, quando n é par, bem como obter uma coloração de arestas para grafos K_n, utilizando $\Delta + 1$ cores, quando n é ímpar.

Nesta dissertação é apresentado um método capaz de colorir as arestas de qualquer grafo, que possua um vértice universal, em tempo $\mathcal{O}(m)$. Esse método utiliza uma estratégia semelhante àquela apresentada por Sharebaf [2009], fazendo uso das informações dos índices associados vértices adjacentes para determinar a cor da aresta que os conecta. Maiores detalhes podem ser encontrados na Seção 3.4.

Também é importante ressaltar a vasta literatura sobre as generalizações do problema de coloração de arestas. O problema de Coloração de Arestas Generalizado em multigrafos, ou f-coloração, proposto por Hakimi & Kariv [1986], é similar ao tradicional problema de Coloração de Arestas apresentado nesta dissertação, com a diferença de que uma cor pode incidir em um vértice até f vezes. No trabalho de Hsu et al. [2006], alguns novos limites inferiores para o número de cores utilizadas neste problema são definidos. Uma Coloração Generaliza das arestas de G necessita de no mínimo $\lceil \frac{\Delta}{f} \rceil$ cores. Assim também tem-se o número mínimo de cores necessárias para colorir as arestas incidentes em um vértice, dado por $\lceil \frac{\delta(v)}{f} \rceil$. A *discrepância global* é a diferença entre o número de cores utilizadas pela coloração de G e o limite inferior $\lceil \frac{\Delta}{f} \rceil$, ou seja $|C| - \lceil \frac{\Delta}{f} \rceil$. Da mesma forma é definido o conceito de *discrepância local* de um nó v como a diferença entre o número de cores incidentes no vértice e o limite inferior $\lceil \frac{\delta(v)}{f} \rceil$, ou seja $|C(v)| - \lceil \frac{\delta(v)}{f} \rceil$, onde $|C(v)|$ é o número de cores incidentes no vértice v. Um algoritmo, que utiliza no máximo $\lceil \frac{\Delta+\mu}{f} \rceil$ cores, de complexidade $\mathcal{O}(m\sqrt{m \cdot \log n})$ foi proposto por Shin-ichi et al. [1993]. Esse limite superior é naturalmente obtido do resultado do Teorema de Vizing para grafos de multiplicidade μ.

Na *Coloração de Arestas Acíclica*, onde não é permitida a existência de ciclos de duas cores em um grafo G, $\alpha'(G)$ representa o número mínimo de cores utilizadas para a coloração e para qualquer grafo, $\alpha'(G) \leq \Delta(G) + 2$. Dong & Xu [2010] provam que qualquer grafo planar que atenda a determinados valores do grau máximo Δ e do comprimento do menor ciclo $g(G) \in G$ possui $\alpha'(G) = \Delta(G)$

Em Liang et al. [1996] são estudados dois problemas de coloração de arestas: concluir a coloração das arestas de um grafo parcialmente colorido após a adição de

um novo vértice a este grafo; e colorir as arestas de um grafo utilizando no máximo $\Delta+1$ cores. Os autores propuseram algoritmos paralelos para a resolução deste problema, mas sem qualquer análise experimental.

Até o momento, poucos trabalhos deram ênfase a uma análise profunda das implementações dos algoritmos baseados na prova do Teorema de Vizing. Tem-se conhecimento de três implementações disponíveis. Uma delas Dong's [2010] não é competitiva com as outras duas. O código disponível em UTA [2010a] é baseado no algoritmo apresentado em Hu & Blake [1997] enquanto UTA [2010b] é inspirado em Misra & Gries [1992]. Estas implementações estão disponíveis *online* e foram desenvolvidos por alunos da *University of Texas at Arlington.*

2.3 O Teorema de Vizing

O principal resultado sobre o estudo de coloração de arestas surgiu com o Teorema de Vizing [1964]. Este teorema mostra que todo grafo G pode ser colorido com no máximo $\Delta + 1$ cores. A demonstração deste teorema, encontrado em Nakano et al. [1995], é baseada no aumento da coloração do grafo G, colorindo uma nova aresta a cada iteração. Esta prova mostra como colorir uma aresta de um grafo parcialmente colorido (o que pode exigir a recoloração de algumas arestas para garantir que todos os vértices sejam mantidos satisfeitos ao longo do processo de coloração) nunca utilizando mais que $\Delta + 1$ cores diferentes. Esse procedimento é repetido até que todas as arestas do grafo sejam coloridas. Dentre as diversas provas disponíveis na literatura, escolheu-se a demonstração a seguir devido à sua clareza e simplicidade[3].

Teorema 1 *Para todo grafo G simples, tem-se que:*

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1 \tag{2.1}$$

Prova. A inequação da esquerda é facilmente provada, uma vez que qualquer vértice de grau Δ precisa de uma cor diferente para cada uma de suas arestas incidentes. O restante da prova depende da estrutura de dados chamada fan. Seja $e_0(w, v_0)$ uma aresta não colorida de um grafo parcialmente colorido. Um fan F é uma sequência de arestas distintas $e_0(w, v_0), e_1(w, v_1), ..., e_k(w, v_k)$, onde o vértice w é considerado o

[3]Misra & Gries [1992] apresentam uma interessante discussão a respeito da evolução da prova do Teorema de Vizing. O estudo da atual versão do teorema iniciou com Edsger Wybe Dijkstra e ATAC (*Austin Tuesday Afternoon Club*) a pedido de Robert Tarjan, que considerava a prova deste teorema confusa e complexa. A demonstração do teorema tornou-se mais clara somente após a introdução dos elementos: fan, inversão e rotação.

centro do fan e os vértices v_i são suas folhas, tal que dada uma sequência de cores distintas $\alpha_0, \alpha_1, ..., \alpha_{k-1}$, as seguintes condições são satisfeitas:

1. A cor α_i encontra-se disponível no vértice $v_i, 0 \leq i \leq k-1$; e

2. Toda aresta e_i, $1 \leq i \leq k$, encontra-se colorida com a cor α_{i-1}.

A construção do fan inicia-se pela aresta descolorida $e_0(w, v_0)$, onde w é chamado de vértice central do fan. A seguir, adiciona-se a aresta $e_1(w, v_1)$, colorida com a cor α_0 disponível no vértice v_0. Este procedimento repete-se para as arestas $e_i(w, v_i)$, coloridas com as cores α_{i-1}, até a construção do fan maximal, ou seja, até que não se possa mais adicionar uma aresta no fan de modo que as condições apresentadas anteriormente mantenham-se satisfeitas.

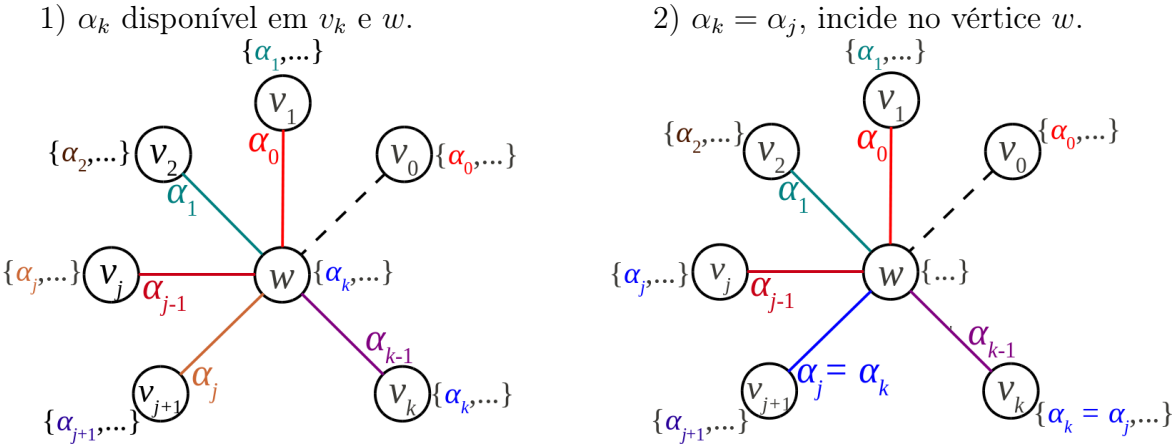


Figura 2.1. Critérios de parada na construção do fan maximal.

Um fan é dito maximal quando uma das seguintes condições for satisfeita (ver Figura 2.1): a cor α_k, disponível no vértice v_k, é uma cor disponível no vértice w; ou, a cor $\alpha_k = \alpha_j$, $0 \leq j < k$, onde α_j é uma cor disponível em um vértice v_j e colore a aresta $e_{j+1}(w, v_{j+1})$.

Rotacionar o fan em n passos é um procedimento onde cada aresta $e_i \in F$ é colorida com a cor α_i, para todo i, $0 \leq i \leq n \leq k$, e a cor α_{n-1} é removida da aresta e_n, fazendo com que esta aresta fique descolorida. Com a rotação de F em n passos, é possível obter uma outra coloração de arestas para G em que a aresta e_n, ao invés da aresta e_0, aparece descolorida (ver Figura 2.2).

Considere um grafo G parcialmente colorido com, no máximo, $\Delta + 1$ cores. Dada uma aresta descolorida $e(w, v_0) \in E(G)$, esta prova mostra como aumentar a coloração de G colorindo essa aresta, sem ultrapassar o limite de $\Delta+1$ cores diferentes, o que pode

exigir a troca das cores de algumas arestas já coloridas para garantir que a coloração permaneça válida.

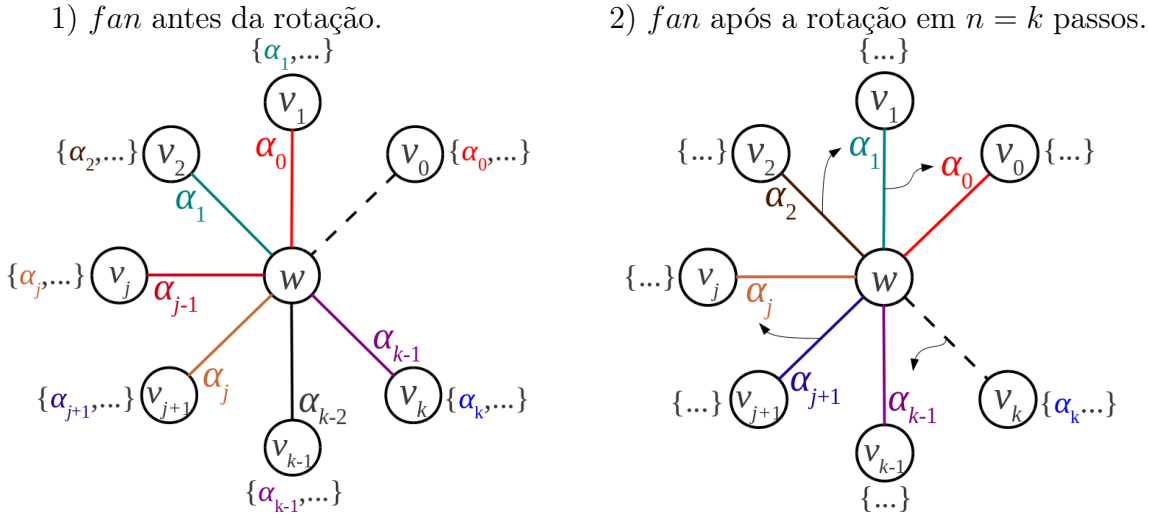


Figura 2.2. Exemplo de rotação do fan em $n = k$ passos.

Inicialmente, busca-se uma cor que esteja disponível tanto no vértice w quanto no vértice v_0, ou seja, busca-se uma cor $\beta \in Free(w)$, e uma cor $\alpha_0 \in Free(v_0)$, tal que $\alpha_0 = \beta$, onde $Free(v)$ representa o conjunto de cores disponíveis em um vértice v. Se essa cor existe, então ela será usada para colorir a aresta $e_0(w, v_0)$ e aumentar a coloração do grafo G. Porém, suponha $\alpha_0 \neq \beta$.

Seja o fan maximal F de tamanho k, tendo w como vértice central, começando pela aresta descolorida $e_0(w, v_0)$ até a aresta $e_k(w, v_k)$, e seja também a cor α_k disponível no vértice v_k.

Se a cor α_k também é uma cor disponível em w, basta rotacionar F em k passos, fazendo com que α_k seja uma cor disponível em ambas as extremidades da aresta descolorida $e_k(w, v_k)$. Consequentemente, colorir e_k com α_k aumentaria a coloração do grafo G, como ilustrado pela Figura 2.3.

Porém, suponha que a cor α_k não é uma cor disponível no vértice w. Neste caso, $\alpha_k = \alpha_j$, tal que $0 \leq j < k$, uma vez que sabemos que F é maximal. Seja β uma cor disponível no vértice w, e seja P o caminho maximal (α_k, β)-path que parte do vértice v_k.

Se P termina em qualquer vértice diferente de w ou v_j, inverte-se o caminho P, fazendo com que a cor β torne-se uma cor disponível nas extremidades da aresta $e_k(w, v_k)$. Então, rotaciona-se F em k passos, tornando a aresta $e_k(w, v_k)$ descolorida, bem como permitindo que e_k seja recolorida com a cor β, e assim aumentando a coloração de G, como ilustrado pela Figura 2.4.

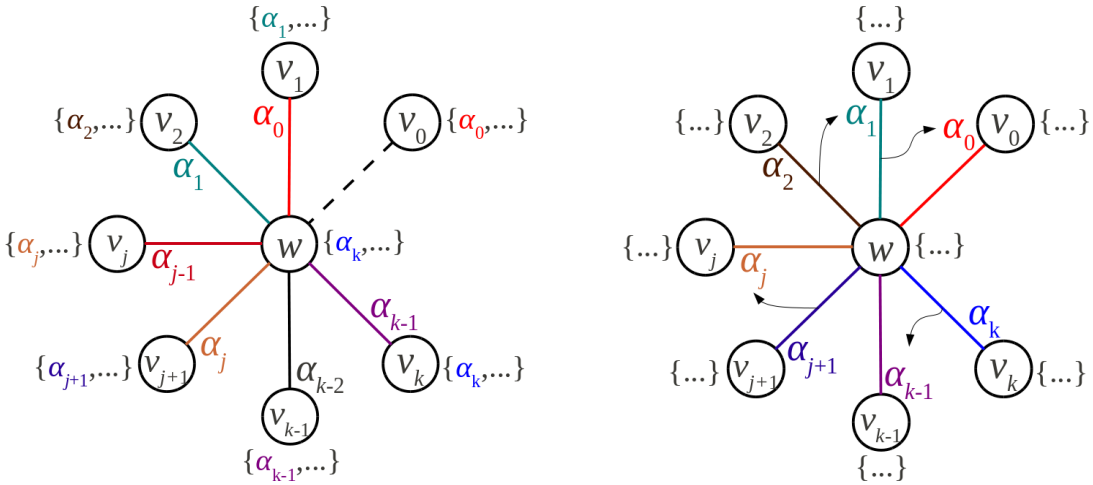


Figura 2.3. Figura à esquerda: o fan maximal é encontrado, visto que, uma cor α_k, disponível em v_k, também está disponível em w. Figura à direita: resultado após a rotação do fan em k passos.

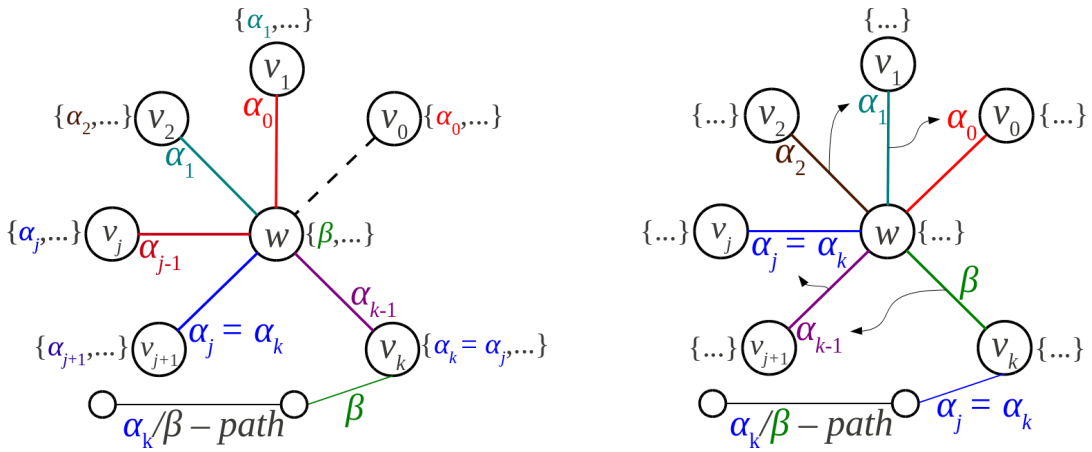


Figura 2.4. Figura à esquerda: P termina em um vértice diferente de vj ou w. Figura à direita: resultado após alternar o caminho P e rotacionar o fan em k passos.

Se P termina no vértice v_j, por uma aresta de cor β, inverte-se o caminho P, fazendo com que a cor β torne-se uma cor disponível nas extremidades da aresta $e_j(w, v_j)$. Então, rotaciona-se F em j passos, tornando a aresta $e_j(w, v_j)$ descolorida, bem como permitindo que e_j seja recolorida com a cor β, aumentando a coloração de G, como ilustrado pela Figura 2.5.

Se P termina no vértice w, por uma aresta de cor $\alpha_k = \alpha_j$, inverte-se P, fazendo com que a cor $\alpha_k = \alpha_j$ torne-se uma cor disponível nas extremidades da aresta $e_{j+1}(w, v_{j+1})$. Então, rotaciona-se F em $j + 1$ passos, tornando a aresta $e_{j+1}(w, v_{j+1})$ descolorida, bem como permitindo que e_{j+1} seja recolorida com a cor α_k, aumentando a coloração de G, como ilustrado pela Figura 2.6.

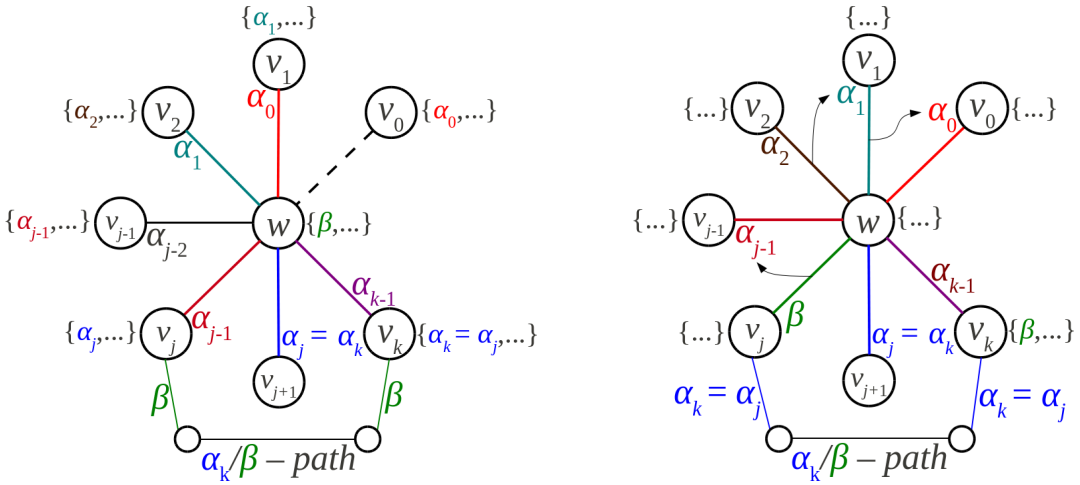


Figura 2.5. Figura à esquerda: caso no qual P termina no vértice vj. Figura à direita: resultado após alternar o caminho P e rotacionar o fan em j passos.

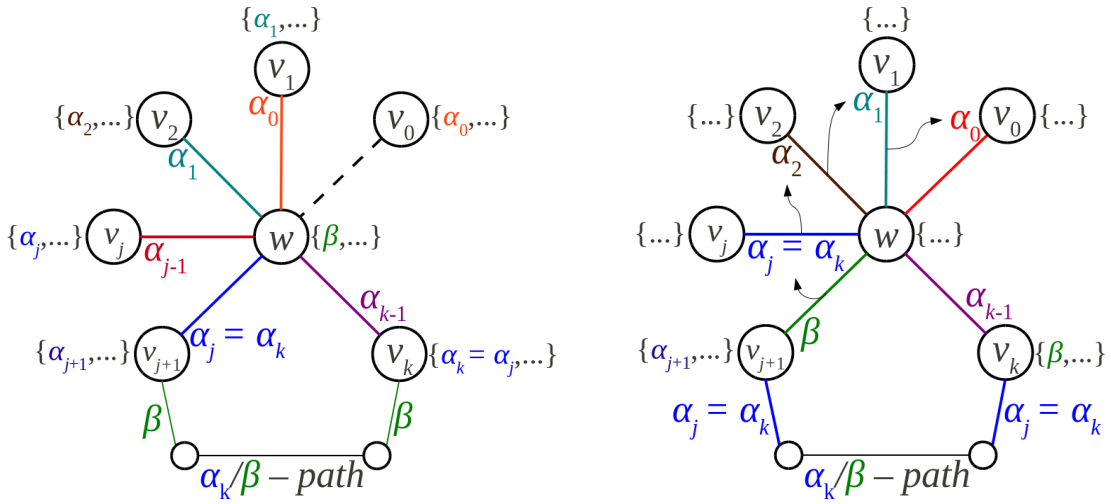


Figura 2.6. Figura à esquerda: caso no qual P termina no vértice w. Figura à direita: resultado após alternar o caminho P e rotacionar o fan em $j+1$ passos.

Haja visto que não existem outros casos a serem tratados, o aumento da coloração fica provado, sem que se faça necessário utilizar mais que $\Delta + 1$ cores.

Os passos dessa demonstração construtiva podem ser aplicados repetidas vezes a partir de um grafo completamente descolorido até que todas as arestas do grafo sejam coloridas. □

O Algoritmo 1 foi obtido a partir da demonstração apresentada, com base no texto de Nakano et al. [1995].

A partir do passos 1 e 2, nota-se que o algoritmo de coloração de arestas aumenta o número de arestas coloridas no grafo em uma unidade a cada iteração. Nos passos 3 e 4, procura-se uma cor α_0 disponível tanto no vértice w quanto no vér-

Algoritmo 1: Algoritmo de Coloração de Arestas Tradicional

```
1  enquanto ∃e ∈ Edge(G)|e.color = ∅ faça
2      e_0(v_0, w) ← {e_0 ∈ Edge(G)|e_0.color = ∅}
3      se ∃α_0 ∈ Free(v_0) ∩ Free(w) então
4          Colorir a aresta e_0 utilizando a cor α_0
5      senão
6          Construa o fan maximal F = e_0(w, v_0), e_1(w, v_1), ..., e_k(w, v_k)
7          Seja α_k ∈ Free(v_k)
8          se α_k ∈ Free(w) então
9              Rotacionar(F, k)
10             Colorir a aresta e_k utilizando a cor α_k
11         senão
12             Seja β ∈ Free(w) e j, tal que α_j = α_k
13             Construa o caminho maximal P = (α_k, β) − path partindo de v_k
14             se P termina em v_j então
15                 Inverter as cores no caminho P
16                 Rotacionar(F, j)
17                 Colorir a aresta e_j utilizando a cor β
18             senão se P termina em w então
19                 Inverter as cores no caminho P
20                 Rotacionar(F, j + 1)
21                 Colorir a aresta e_{j+1} utilizando a cor α_k
22             senão                    /* P não termina nem em w ou em v_j */
23                 Inverter as cores no caminho P
24                 Rotacionar(F, k)
25                 Colorir a aresta e_k utilizando a cor β
```

tice v_0. Se essa cor existe, a mesma é atribuída à aresta $e_0(w, v_0)$. Se não existe uma cor na interseção $Free(w) \cap Free(v_0)$ (passo 5), no passo 6 o fan maximal $F = e_0(w, v_0), e_1(w, v_1), ..., e_k(w, v_k)$ é construído. No passo 7, se a cor α_k encontra-se disponível no vértice v_k, e se α_k também é uma cor disponível no vértice w, rotaciona-se o fan F e então utiliza-se a cor α_k para colorir a aresta $e_k(w, v_k)$, de acordo com os passos de 8 a 10.

Caso a cor α_k, disponível no vértice v_k, seja igual a uma cor α_j atribuída a uma aresta pertencente ao fan, ou seja $\alpha_k = \alpha_j$, para $0 \leq j < k$ (passo 11), no passo 12 uma cor β é escolhida dentre as cores disponíveis no vértice w. No passo 13 é construído o caminho maximal $P = \alpha_k/\beta - path$ começando pelo vértice v_k. As próximas operações do algoritmo seguem as operações apresentadas nos três casos do Teorema de Vizing. Os passos 14 a 17 correspondem ao Caso 1, onde o caminho P termina no vértice v_j.

Os passos 18 a 21 correspondem ao Caso 2, onde o caminho P termina no vértice w e os passos 22 a 25 correspondem ao caso 3, onde o caminho P termina em qualquer vértice diferente de w ou v_j.

A corretude do algoritmo é omitida uma vez que segue a prova do Teorema de Vizing. A complexidade assintótica do algoritmo é facilmente demonstrada. Uma vez que o algoritmo aumenta a coloração do grafo, colorindo uma nova aresta a cada iteração, sua complexidade é dada por $\Theta(m)$, pois é necessário percorrer todas as arestas do grafo para obter uma coloração. O Teorema de Vizing garante que no máximo Δ arestas são selecionadas para a construção e rotação do fan, adicionando o custo para construir a alternar as cores no caminho $P = \alpha/\beta - path$, que é $\mathcal{O}(n)$. Em consequência, a complexidade assintótica do algoritmo é dada por $\mathcal{O}(m \cdot (\Delta + n))$ que é equivalente a $\mathcal{O}(m \cdot n)$.

No próximo capítulo, será apresentado um algoritmo de coloração de arestas baseado na demonstração do Teorema de Vizing disponível em Gould [1988]. Este algoritmo difere do algoritmo apresentado neste capítulo em algumas características que serão detalhadas no Capítulo 3.

Capítulo 3

Algoritmo para Coloração de Arestas

Este capítulo descreve o algoritmo de coloração de arestas proposto, que utiliza não mais que $\Delta + 1$ cores, considerando suas estruturas de dados e procedimentos de atualização, estratégias utilizadas na obtenção de ganhos de desempenho, estudo do pré-processamento das arestas e as suas diferentes formas de aplicação no algoritmo.

3.1 Algoritmo de Coloração de Arestas Proposto

O Algoritmo 2, que colore um grafo G com não mais que $\Delta + 1$ cores, é baseado na prova do Teorema de Vizing, disponível em Gould [1988].

Para garantir que o algoritmo obtenha uma coloração válida para G, em um número de passos finito, é definida uma variável auxiliar chamada de *taboo* para representar uma cor que tem o seu uso temporariamente proibido ao longo da coloração do grafo. Inicialmente inicializa-se a cor *taboo* com o elemento vazio. Esta cor possuirá um valor diferente de vazio em uma iteração i do algoritmo quando, na iteração $i-1$, para $i > 1$, o caminho alternado $\alpha/\beta - path$, partindo de v_0, terminar no vértice w.

Uma vez que a cor *taboo* nunca possui um valor diferente de vazio na primeira iteração do algoritmo, em cada iteração, como pode ser visto através do passo 6, uma aresta $e_0(w, v_0)$ é escolhida dentro do conjunto de arestas não coloridas de G.

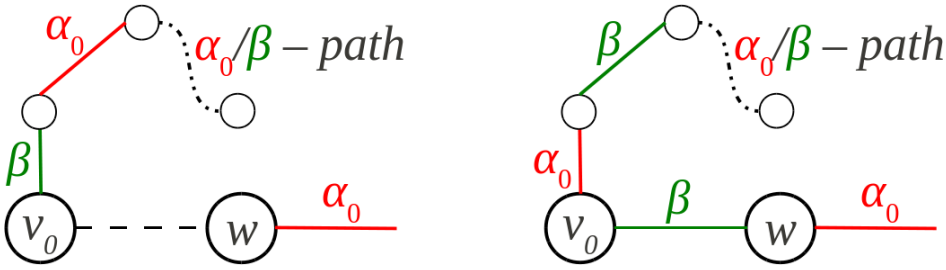
No passo 7, busca-se uma cor $\gamma \in Free(v_0) \cap Free(w)$. Se essa cor existe, utiliza-se γ para colorir a aresta $e_0(w, v_0)$ (passo 8) e atualiza-se a informação da cor *taboo*. Nos passos de 10 a 13, Se não existe uma cor disponível na interseção $Free(v_0) \cap Free(w)$, as cores $\alpha \in Free(v_0)$ e $\beta \in Free(w)$ são escolhidas para construir o caminho alternado maximal $\alpha/\beta - path$ começando por v_0. Pode-se afirmar que a cor $\alpha \in Free(v_0)$ incide

Algoritmo 2: Algoritmo de Coloração de Arestas Proposto

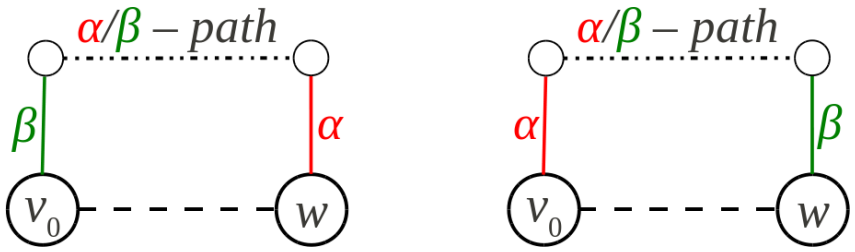
```
1  taboo ← nil
2  enquanto ∃e ∈ Edge(G)|e.color = ∅ faça
3      se taboo ≠ nil então
4          e_0(v_0, w) ← e_1(v_1, w)
5      senão
6          e_0(v_0, w) ← {e_0 ∈ Edge(G)|e_0.cor = ∅}
7      se ∃γ ∈ Free(v_0) ∩ Free(w) então
8          Colorir a aresta e_0 utilizando a cor γ
9          taboo ← nil
10     senão
11         Escolha α ∈ Free(v_0)|α ≠ taboo
12         se taboo ≠ nil então
13             Escolha β ∈ Free(w)|β ≠ taboo
14         se P = (α, β) − path começando em v_0 não termina em w então
15             Inverter as cores no caminho P
16             Colorir a aresta e_0 utilizando a cor β
17             taboo ← nill
18         senão
19             e_1 ← Table(w, α).e
20             Remover a cor α da a aresta e_1
21             Colorir a aresta e_0 utilizando a cor α
22             taboo ← α
```

no vértice w, caso contrário seria possível utilizar essa cor para colorir a aresta $e_0(v_0, w)$. Da mesma forma a cor $\beta \in Free(w)$ incide no vértice v_0, caso contrário seria possível utilizar essa cor para colorir a aresta $e_0(v_0, w)$.

Se, no passo 14, o caminho alternado $\alpha/\beta - path$ partindo do vértice v_0 não termina em w, as cores α e β são invertidas ao longo do caminho (passo 15), e a aresta $e_0(v_0, w)$ é colorida com β (passo 16), que agora também está livre no vértice v_0 graças à inversão das cores no caminho $\alpha/\beta - path$ (ver Figura 22a). Quando o caminho $\alpha/\beta - path$ termina em w, inverter as cores do caminho apenas irá trocar as posições das cores disponíveis nos vértices terminais de e_0, fazendo com que α fique disponível em w, mas não em v_0 e fazendo com que a cor β fique disponível em v_0, mas não em w (ver Figura 22b). Nesse caso, nos passos 19 e 20, a aresta $e(v_1, w)$, que é uma aresta colorida com a cor α e incide no vértice w, tem a sua cor α removida e, no passo 21, esta mesma cor será utilizada para colorir a aresta $e_0(v_0, w)$. Na próxima iteração, a aresta $e_1(v_1, w)$ (passos 3 e 4), será colorida com uma cor ψ diferente de $taboo = \alpha$. A cor ψ



a) Caminho alternado $\alpha/\beta - path$ não terminando no vértice w.



b) Situação onde o caminho alternado $\alpha/\beta - path$ encontra o vértice w.

Figura 3.1. Diferentes cenários para a construção do caminho alternado $\alpha/\beta - path$ partindo do vértice v_0.

existe, pois há pelo menos duas cores disponíveis nos vértices v_0 e w, já que em todo vértice incidem, no máximo, Δ arestas de cores diferentes e, além de sempre existirem $\Delta + 1$ cores disponíveis, a aresta $e_1(v_1, w)$ encontra-se descolorida. Essa última etapa é importante para manter a validade do algoritmo. Note que o algoritmo proposto não constrói a estrutura fan utilizada pelo algoritmo tradicional, no entanto a idéia do fan aparece de forma implícita, onde w é o vértice central.

Observe que quando a cor *taboo* assume um valor diferente de *nill* na iteração atual, a aresta a ser colorida na próxima iteração incide no vértice fixo w, que anteriormente estava colorida com a cor α. Neste caso, o vértice w e a cor $\beta \in Free(w)$ permanecem inalterados de uma iteração para a próxima até que a aresta incidente no vértice w seja finalmente colorida. A demonstração do Teorema de Vizing, apresentada em Gould [1988], garante que são necessárias, no máximo, Δ iterações de trocas de cores das arestas ao redor do vértice w para encontrar uma coloração válida para todas as suas arestas incidentes já coloridas.

Uma vez que o algoritmo apresentado aumenta a coloração de uma aresta a cada vez (em uma ou mais iterações), sua complexidade pode ser dada por $\Theta(m)$ vezes a complexidade para colorir uma única aresta para aumentar a coloração parcial do grafo. No algoritmo principal, a operação mais cara a ser executada em cada iteração é a construção e a inversão do caminho alternado maximal $\alpha/\beta - path$ onde

sua complexidade computacional é dada por $\mathcal{O}(n)$ uma vez que esse caminho é simples. Em consequência, o tempo de execução do algoritmo pode ser dado por $\mathcal{O}(m \cdot \Delta \cdot n)$.

Ainda que a complexidade do algoritmo proposto seja superior à complexidade do algoritmo de coloração de arestas tradicional, espera-se que na prática esses algoritmos apresentem desempenhos similares, ou ainda, que com o uso das estratégias de execução e de pré-processamento apresentadas, o algoritmo proposto apresente desempenho superior.

Esse tipo de resultado não é inédito na ciência da computação. De acordo com Matouek & Gärtner [2006], o método de resolução de problemas de programação linear Simplex, que possui complexidade assintótica exponencial, na prática possui um comportamento polinomial para a maioria dos problemas reais em que é aplicado. Além disso o método Simplex é, em geral, mais eficiente que o método dos Pontos Interiores e o método dos Elipsóides, que são algoritmos de complexidades polinomiais, mas que na prática tendem a apresentar comportamentos muito próximos aos resultados de pior caso das suas respectivas análises de complexidade.

3.2 Estruturas de Dados

A eficiência de um procedimento está fortemente associada à forma como os dados são organizados e manipulados. A escolha das estruturas de dados para a implementação do algoritmo de coloração de arestas proposto neste trabalho foi baseada nas análises das operações mais caras e realizadas com maior frequência, buscando obter menor complexidade em termos de espaço e tempo computacional.

Inicialmente, se faz necessário definir três estruturas de dados desenvolvidas para otimizar as operações realizadas com maior frequência durante a execução do algoritmo: $Free(v)$, $Edge(G)$ e $Table(v, \alpha)$.

A estrutura $Free(v)$ armazena o conjunto de até $\Delta + 1$ cores disponíveis para colorir as arestas incidentes a um dado vértice v.

A estrutura de dados $Edge(G) = \{e \in E(G)\}$ é um conjunto de tamanho m onde cada elemento é uma abstração de uma das arestas do grafo G e possui as seguintes variáveis: v, w e cor. As variáveis $e.v$ e $e.w$ representam os vértices conectados pela aresta e e a variável $e.cor$ armazena a informação referente à cor atribuída a esta aresta.

$Table(v, \alpha)$ é uma matriz em que cada uma das suas linhas está associada a um vértice $v \in V(G)$ e cada uma das suas colunas está associada a uma cor $\alpha \in C$, $|C| = \Delta + 1$. Cada elemento de $Table(v, \alpha)$ que associa o vértice v com a cor α possui as seguinte variáveis: $status$, adj, e, e $cIndex$. A variável booleana $Table(v, \alpha).status$

assume o valor 1 (verdade) sempre que uma aresta de cor α incide no vértice v, e assume o valor 0 (falso) caso esta cor encontre-se disponível no vértice v. A variável $Table(v, \alpha).adj$ representa o vértice adjacente a v que está conectado pela aresta de cor α, $Table(v, \alpha).e$ é um apontador que referencia a aresta e, que conecta os vértices v e $Table(v, \alpha).adj$, na estrutura $Edge(G)$. Finalmente, o campo $Table(v, \alpha).cIndex$ armazena a informação sobre localização da cor α no conjunto $Free(v)$.

3.2.1 Implementação e Atualização das Estruturas de Dados

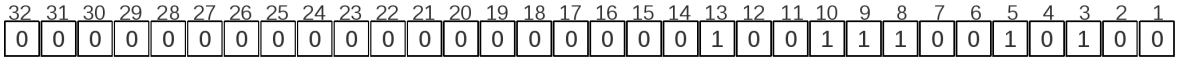
Neste trabalho, foram propostas duas implementações para a estrutura de dados que representa o conjunto $Free(v)$. A primeira delas utiliza efetivamente uma lista de números inteiros para a representação das cores. Caso a cor α seja selecionada para colorir uma aresta incidente no vértice v, o inteiro que a representa deve ser removido da lista $Free(v)$. Neste caso, a variável $Table(v, \alpha).cIndex$ indicará a posição desta cor na estrutura $Free(v)$. Por outro lado, se a cor α está sendo adicionada em $Free(v)$ (por que essa cor foi removida de alguma aresta incidente em v), basta apenas inseri-la no fim da lista $Free(v)$. As operações de inserção e remoção são elementares no algoritmo de coloração de arestas e são executadas em tempo $\mathcal{O}(1)$.

Para determinar uma cor não incidente em um vértice v, basta acessar uma posição válida na lista que representa $Free(v)$. No entanto, para determinar uma cor $\alpha \in Free(v) \cap Free(w)$ é necessário consultar cada elemento de $Free(v)$ e verificar se o mesmo está na lista de cores disponíveis de w em tempo $\mathcal{O}(1)$ utilizando a variável $Table(w, \alpha).status$. A operação completa é executada em tempo $\mathcal{O}(min\{|Free(v)|, |Free(w)|\})$, que, no pior caso, é $\mathcal{O}(\Delta)$. Esta metodologia utilizada para determinar uma cor disponível em um par de vértices foi primeiramente proposta em Cole & Kowalik [2008].

Na segunda implementação de $Free(v)$, utiliza-se um vetor de tamanho $\lceil \frac{\Delta + 1}{|palavra|} \rceil$, onde cada elemento é um número inteiro do tamanho da *palavra*[1] da arquitetura do computador onde o algoritmo está sendo executado. O j-ésimo *bit* do i-ésimo inteiro do vetor representa a disponibilidade da cor $|palavra| \cdot (i-1) + j$, onde $|palavra|$ representa o número de *bits* em uma *palavra*. Nesta estrutura, uma cor disponível em um vértice é definida por um *bit* 1, caso contrário o *bit* é definido como 0. Neste trabalho, foi utilizada uma arquitetura com *palavras* com 32 *bits*.

[1] O termo *palavra* (em inglês: *word*) é a unidade de informação natural usada por um tipo de computador particular. É um grupo de *bits* de tamanho fixo que é processado em conjunto numa máquina. O número de *bits* em uma palavra, ou o tamanho ou comprimento da palavra, é uma característica específica de cada arquitetura de computador.

a) Representação em binário de $p = 5012$ na posição i de $Free(v_0)$, onde seu *bit* mais significativo armazena a cor $32 \cdot (i-1) + 13$.



b) Após o deslocamento dos bits mais significativos de $p = 355090432$, tem-se $bitTable[p] = 13$, tal que o cálculo da cor armazenada é dado por $32 \cdot (i-1) + 29$.

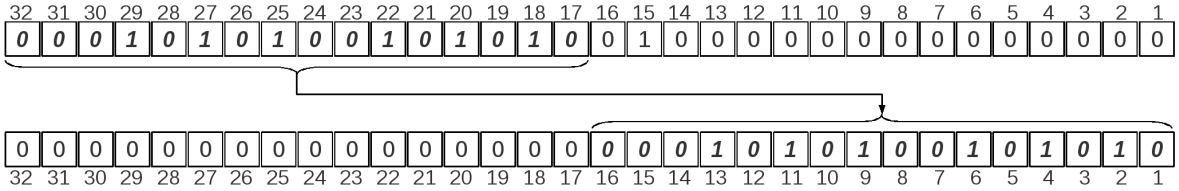


Figura 3.2. Obtendo uma cor disponível em um vértice utilizando estruturas de dados em *bits*.

A atualização do estado da cor α nessa estrutura também é feita em tempo $\mathcal{O}(1)$, visto que só é preciso calcular a localização da cor no vetor de palavras e definir o seu novo valor como 1 ou 0. Nota-se, que usando esta estratégia, não é mais necessário manter as variáveis $Table(v, \alpha).status$ e $Table(v, \alpha).cIndex$.

Nesta segunda implementação, determinar uma cor disponível em $Free(v)$ requer encontrar algum elemento inteiro diferente de zero nesse vetor. Uma vez encontrado, deve-se determinar um *bit* 1 (ligado) nesse inteiro. Utilizando *palavras* de uma arquitetura de 32 *bits*, essa operação pode ser feita de forma rápida se as posições dos *bits* mais significativos dos números decimais de 1 a 2^{16} forem calculadas *a priori* e armazenadas em uma tabela *bitTable*. Se o inteiro i diferente de zero é menor que $2^{16} = 65536$, a posição de seu correspondente *bit* mais significativo é $bitTable[i]$, caso contrário, a posição do seu *bit* mais significativo é dada por $bitTable[i >> 16] + 16$, onde $>>$ representa uma operação de deslocamento de *bits* à direita. Sabendo que a estrutura $Free(v)$, implementada com o uso de vetores de *palavras*, possui tamanho $\lceil \frac{\Delta + 1}{32} \rceil$, a operação para determinar uma cor disponível nessa estrutura tem um custo de $\mathcal{O}(\Delta)$, pois pode ser necessário avaliar todos os elementos da estrutura para que uma cor disponível seja encontrada.

Como exemplo do uso da operação de deslocamento de *bits*, em uma arquitetura de 32 *bits*, seja p a i-ésima *palavra* na estrutura $Free(v_0)$, para $0 < i \leq \lceil \frac{\Delta + 1}{32} \rceil$. De acordo com a Figura 3.2.1a, para uma palavra $p = 5012$ na posição i de $Free(v_0)$, as cores correspondentes aos seus *bits* iguais a 1 são dadas por $32 \cdot (i-1) + j$, para $j =$ 3,5,8,9,10 e 13. Uma vez que $p = 5012 < 65536$, a posição do *bit* mais significativo pré-

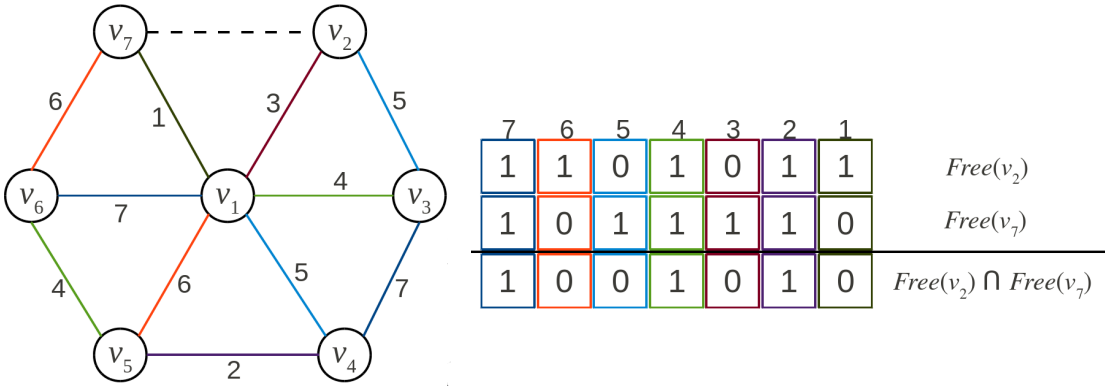


Figura 3.3. Exemplo da obtenção de uma cor disponível para a coloração da aresta $e(v_2, v_7)$ com a representação em *bits*. As cores disponíveis para a coloração, obtidas através da operação AND, são $Free(v_2) \cap Free(v_7) = \{2, 4, 7\}$.

calculado é dado por $bitTable[p] = 13$. Para um cenário onde $p = 355090432 > 65536$, tem-se a situação ilustrada pela Figura 3.2.1b. Neste caso, os 16 *bits* mais significativos serão deslocados, o seu *bit* mais significativo é dado por $bitTable[p] = 13$, de tal forma que a cor armazenada é dada por $32 \cdot (i-1) + (13+16) = 32 \cdot (i-1) + 29$.

Determinar uma cor $\alpha \in Free(v) \cap Free(w)$ implica em encontrar um *bit* 1 em comum nos dois vetores de *palavras*, ou seja, neste caso é necessário calcular o resultado da operação AND para cada elemento i, $0 < i \leq \lceil \frac{\Delta+1}{32} \rceil$, nos vetores de bits $Free(v)$ e $Free(w)$, até encontrar um resultado da operação AND que seja diferente de zero, como ilustrado na Figura 3.2.1. Mesmo que esta operação tenha tempo $\mathcal{O}(\Delta)$, na prática, é mais rápida que a mesma operação na implementação em lista, por permitir a comparação de 32 elementos em um único ciclo de processador.

Estas duas implementações da estrutura $Free(v)$ foram desenvolvidas devido à operação fundamental do algoritmo de coloração de arestas: determinar uma cor em comum disponível em dois vértices adjacentes, uma vez que é desejável atribuir uma cor válida para uma aresta o mais cedo possível.

Para o conjunto $Free(v)$ é importante discutir por que esta estrutura de dados não foi implementada sobre as arestas do grafo G, uma vez que esta seria uma estratégia natural a ser adotada sabendo que a informação referente às cores disponíveis está associada diretamente às arestas.

Se as cores disponíveis estivessem representadas nas arestas, buscar uma cor válida para colorir uma aresta $e = (v, w)$ tornar-se-ia muito mais simples, uma vez que bastaria acessar um elemento válido dentro da estrutura $Free(e)$. No entanto, as operações de inserção e remoção de uma dada cor implicaria em um alto custo para atualizações das informações presentes nas demais arestas incidentes nos vértices v e

w.

Enquanto a complexidade de tempo dessas operações é $\mathcal{O}(1)$ na estrutura $Free(v)$, utilizando a estrutura $Free(e)$, cada operação teria um custo $\mathcal{O}(\Delta)$, pois quando uma cor é inserida em uma aresta $e = (v, w)$, esta não poderia ser inserida em nenhuma outra aresta incidente em v ou w, e quando uma cor é removida de uma aresta $e = (v, w)$, esta tornar-se-ia disponível para a coloração de qualquer aresta incidente em v ou w, logo todas as informações presentes nas arestas incidentes em v e w deveriam ser atualizadas após cada operação de inserção ou remoção de uma cor no grafo.

Ao longo da execução do algoritmo de coloração de arestas, caso não exista uma cor disponível para colorir a aresta $e = (v, w)$, deverão ser escolhidas uma cor α disponível em v e uma cor β disponível em w. Novamente, enquanto a complexidade de tempo para a obtenção de uma cor disponível em um vértice é $\mathcal{O}(1)$ na estrutura $Free(v)$, obter tal cor utilizando a estrutura $Free(e)$ teria um custo $\mathcal{O}(\Delta)$, pois seria necessário verificar as cores disponíveis em todas as arestas incidentes em v e w.

Analisando a complexidade de espaço, a representação das cores disponíveis nas arestas possui complexidade $\mathcal{O}(m \cdot \Delta)$, ao passo que a representação das cores disponíveis nos vértices possui complexidade $\mathcal{O}(n \cdot \Delta)$. Logo, representar as cores disponíveis nos vértices é a opção mais adequada ao contexto deste trabalho, uma vez que os grafos analisados possuem $m > n$.

Outra importante operação no algoritmo de coloração de arestas é encontrar o vértice w adjacente a v que é conectado por uma aresta de cor α. Caso a cor α incida no vértice v, a estrutura $Table(v, \alpha).adj$ permite encontrar tal vértice em tempo $\mathcal{O}(1)$. Esta propriedade é importante na determinação do caminho $\alpha/\beta - path \in G$ a partir do vértice v com custo $\mathcal{O}(n)$.

3.3 Estratégias de Execução

O Algoritmo 2 deixa em aberto duas estratégias de execução. A primeira refere-se à seleção das cores a serem atribuídas a uma aresta quando o número de cores disponíveis é maior que um. A segunda estratégia leva em conta a ordem com que as arestas ainda não coloridas são processadas pelo algoritmo. Estas escolhas podem influenciar diretamente no tempo de execução do algoritmo implementado.

Dado um conjunto de cores disponíveis $Free(v) = \{\alpha_0, \alpha_1, ..., \alpha_\Delta\}$, o método de seleção *primeira disponível*[2] consiste em escolher a primeira cor livre α_i encontrada

[2]O método de seleção *primeira disponível* não exige que as cores estejam ordenadas no conjunto

na estrutura $Free$. Em um método de seleção aleatório, uma cor valida é selecionada aleatoriamente a partir do conjunto de cores disponíveis.

Para a seleção de arestas, é definido o conceito de seleção lexicográfica. Neste método de seleção, todas as arestas não coloridas incidentes no vértice v_i são processadas antes das arestas não coloridas incidentes no vértice v_{i+1}. Além disso, as arestas também podem ser processadas em ordem aleatória.

Combinando estas duas diferentes possibilidades de estratégias de execução, surgem quatro diferentes versões para o algoritmo proposto:

– PCAL: Seleção de cores pela primeira disponível e seleção de arestas em ordem lexicográfica.

– PCAA: Seleção de cores pela primeira disponível e seleção aleatória de arestas.

– CAAL: Escolha aleatória de cores e seleção de arestas em ordem lexicográfica.

– CAAA: Escolha aleatória de cores e arestas.

Diversas outras estratégias de escolhas de cores, arestas e vértices foram testadas, tais como: ordenar os vértices de forma a selecionar primeiramente as arestas daqueles de maior grau, ordenar os vértices de forma a selecionar primeiramente as arestas daqueles de menor grau, escolher a cor que aparece com a menor frequência no grafo para colorir a aresta, escolher a cor que aparece com a maior frequência no grafo e por fim escolher as arestas do grafo que incidem nos vértices com a maior quantidade de arestas incidentes descoloridas. No entanto, nenhuma dessas estratégias se mostraram tão eficientes quanto as que estão sendo apresentadas neste trabalho. A apresentação de tais resultados foi omitida com o objetivo de focar o texto nos resultados obtidos com maior relevância.

No Capítulo 4 são apresentados os resultados empíricos da análise das estratégias de execução propostas com o intuito de determinar aquela com o melhor desempenho.

3.4 Pré-processamento

Com o objetivo de encontrar rapidamente uma cor válida para as arestas do grafo, foram propostas estratégias de pré-processamento que verificam todas as arestas do grafo de entrada G na tentativa de determinar rapidamente uma cor válida para cada uma delas. O grafo de saída G é um grafo parcialmente colorido que será utilizado como entrada para os algoritmos de coloração de arestas apresentados nesta dissertação.

Este pré-processamento é uma função $hash$ que determina uma cor para uma dada aresta utilizando a soma dos índices dos vértices conectados por ela, calculada

$Free(v)$. Esta simplificação foi adotada para eliminar o custo de ordenação na estrutura $Free(v)$.

em módulo $\Delta + 1$. Por exemplo, para determinar uma cor de uma aresta $e(v_i, v_j)$ calcula-se o resultado de $(i + j) \mod \Delta + 1$ e então a cor $\alpha_{(i+j) \mod \Delta+1}$ será utilizada para colorir tal aresta.

Caso o grafo em questão possua um vértice universal, é possível garantir que as arestas do mesmo podem ser coloridas, utilizando no máximo $\Delta + 1$ cores, com complexidade $\Theta(m)$. Esta afirmação é consequência da proposição apresentada a seguir.

Proposição. Seja o conjunto de vértices $V(G) = \{v_o, v_1, ..., v_{n-1}\}$ e o conjunto de cores $C = \{\alpha_0, \alpha_1, ..., \alpha_\Delta\}$. Se G contém um vértice universal, é possível colorir qualquer aresta $e(v_i, v_j)$ com a cor $\alpha_{(i+j)mod(\Delta+1)}$ e assim obter uma coloração própria válida utilizando não mais que $\Delta + 1$ cores.

Prova. Sejam v_i, v_j e v_k três vértices distintos de um grafo que possui um vértice universal, tais que v_i é adjacente a v_j e a v_k. Assumindo, por absurdo, que as arestas (v_i, v_j) e (v_i, v_k) sejam coloridas com a mesma cor, gerando uma coloração inválida, temos que:

$$(i + j)mod(\Delta + 1) = (i + k)mod(\Delta + 1)$$

$$(i + j) \equiv (i + k)mod(\Delta + 1)$$

$$j \equiv k \ mod(\Delta + 1)$$

onde a última equação afirma que j e k devem ter o mesmo resto da divisão por $\Delta + 1$. Uma vez que $0 \leq j, k < n$, onde $n = \Delta + 1$ devido à presença de um vértice universal, implica que $j = k$, o que é impossível uma vez que os vértices em questão são distintos. Portanto, também é impossível que cores iguais sejam atribuídas às arestas (v_i, v_j) e (v_i, v_k). □

A proposição apresentada pode ser utilizada na tentativa de colorir a maior quantidade possível de arestas de um grafo. Se este grafo não possui um vértice universal, a proposição falha para algumas arestas e, assim, as arestas que não puderam ser coloridas pelo pré-processamento serão coloridas pelo algoritmo de coloração. Neste trabalho, o pré-processamento é aplicado de duas formas diferentes:

Pré-processamento *a priori*: Nesta abordagem, o algoritmo de pré-processamento tenta colorir cada aresta (v_i, v_j) com a cor $\alpha_{(i+j)mod(\Delta+1)}$. Se um conflito ocorre (porque outra aresta incidente em v_i ou em v_j já foi colorida com a mesma cor), a aresta não colorida (v_i, v_j) é armazenada na lista $EdgeList(E)$ para ser posteriormente processada pelo algoritmo principal. O Algoritmo 3 descreve os passos do pré-processamento *a priori*.

O passo 1 do Algoritmo 3, mostra que o procedimento deve ser aplicado em todas as arestas do grafo. No passo 2, o cálculo da cor α, a ser atribuída à aresta $e(v_i, v_j)$,

Algoritmo 3: Utilizando o pré-processamento *a priori*.

1 **para cada** $e(v_i, v_j) \in Edge(G)$ **faça**
2 $\quad \alpha \leftarrow (i + j) mod(\Delta + 1)$
3 $\quad$ **se** $\alpha \in Free(v_i) \cap Free(v_j)$ **então**
4 $\quad\quad$ Colorir a aresta $e(v_i, v_j)$ utilizando a cor α
5 $\quad$ **senão**
6 $\quad\quad$ $EdgeList(E) \leftarrow EdgeList(E) + e$
7 Aplique o algoritmo de coloração de arestas utilizando o grafo parcialmente colorido G e a lista $EdgeList(E)$ como entrada

é dado por $(i + j) mod(\Delta + 1)$. No passo 3, verifica-se a disponibilidade dessa cor nos vértices v_i e v_j. Caso α esteja disponível nos dois vértices, no passo 4, a aresta $e(v_i, v_j)$ será colorida com a cor α. Caso contrário, a aresta e será armazenada em uma lista de arestas não coloridas, $EdgeList(E)$, de acordo com os passos 5 e 6. No passo 7, o algoritmo de coloração de arestas é utilizado para colorir cada uma das arestas da lista $EdgeList(E)$, com o objetivo de obter uma coloração válida de G.

No Algoritmo 3, a ordem com que as arestas são fornecidas trata-se apenas de um detalhe de implementação. Qualquer uma das ordens de seleção de arestas apresentadas na Seção 3.3 podem ser adotadas. No entanto, a estratégia *a priori* não garante que as arestas serão coloridas na mesma ordem em que foram passadas para o algoritmo de pré-processamento, haja visto que algumas arestas podem ser armazenadas para serem coloridas posteriormente pelo algoritmo de coloração de arestas.

Pré-processamento *embutido*: Nesta abordagem, o algoritmo de pré-processamento tenta colorir cada aresta (v_i, v_j) com a cor $\alpha_{(i+j) mod(\Delta+1)}$. Se um conflito ocorre, o algoritmo de coloração de arestas é chamado imediatamente para resolvê-lo. Neste caso, o processamento *embutido* e o algoritmo principal de coloração de arestas trabalham juntos. O Algoritmo 4 descreve os passos do pré-processamento *embutido*.

Algoritmo 4: Utilizando o pré-processamento *embutido*.

1 **para cada** $e(v_i, v_j) \in Edge(G)$ **faça**
2 $\quad \alpha = (i + j) mod(\Delta + 1)$
3 $\quad$ **se** $\alpha \in Free(v_i) \cap Free(v_j)$ **então**
4 $\quad\quad$ Colorir a aresta $e(v_i, v_j)$ utilizando a cor α
5 $\quad$ **senão**
6 $\quad\quad$ Aplique o algoritmo de coloração de arestas em e

No Algoritmo 4, os passos de 1 a 5 são exatamente iguais aos descritos para o

Algoritmo 3. No entanto, no passo 5, ao invés de armazenar a aresta não colorida, o algoritmo de coloração de arestas é aplicado na aresta $e(v_i, v_j)$, sempre garantindo uma coloração parcial do grafo G, até que todas as arestas sejam coloridas.

Neste algoritmo, diferentemente do Algoritmo 3, o algoritmo de pré-processamento, juntamente com o algoritmo de coloração de arestas, colore as arestas de G na ordem em que são fornecidas. Assim, as estrategias de execução para a ordem de processamento das arestas, definidas na Seção 3.3, são respeitadas.

3.5 Outras Implementações para o Problema de Coloração de Arestas

Ao longo do desenvolvimento deste trabalho foram encontradas apenas três implementações do algoritmo de coloração de arestas baseados no Teorema de Vizing, no entanto nenhuma foi desenvolvida visando eficiência de implementação. A primeira implementação, disponível em Dong's [2010], apresenta um desempenho muito ruim e nenhuma análise sobre ela será apresentada neste texto.

A segunda implementação possui o código disponível em UTA [2010a] e é baseada na descrição do algoritmo apresentado em Hu & Blake [1997], que é similar ao algoritmo proposto. A última implementação, cujo código encontra-se disponível em UTA [2010b] é baseada na demonstração do Teorema de Vizing apresentada em Misra & Gries [1992], que é similar ao algoritmo tradicional, além de implementar a estrutura de dados fan de forma explícita. Essas duas implementações utilizam estruturas de dados muito simples, não implementando qualquer forma eficiente de buscar uma cor disponível em um vértice, e assim, realizam esta operação em tempo $\mathcal{O}(n)$.

Nas duas implementações, a representação das arestas dos grafos é feita com o uso de dois vetores de inteiros para armazenar os pares de vértices adjacentes. Uma matriz M de valores booleanos, com tamanho $n \cdot (\Delta + 1)$, é utilizada para indicar a presença de uma cor c em um vértice v quando o elemento $M[v][c] = 1$, e indicar que esta cor c encontra disponível no vértice v quando $M[v][c] = 0$.

Capítulo 4

Análises Experimentais

Neste capítulo serão apresentadas as análises experimentais realizadas a respeito dos algoritmos de coloração de arestas estudados ao longo do texto. Através de uma grande bateria de testes, busca-se avaliar o desempenho das estruturas de dados, ganhos obtidos com os métodos de seleção de cores e de arestas, pré-processamento e comparar o algoritmo desenvolvido neste trabalho com os algoritmos disponíveis na literatura.

4.1 Descrição do Ambiente Computacional e Instâncias de Teste

Os algoritmos descritos nos Capítulos 2 e 3 foram codificados na linguagem C++ e compilados com a versão 4.4.1 do compilador g++. As implementações UTA [2010a] e UTA [2010b], que também serão comparadas neste trabalho, foram compiladas com o mesmo compilador, utilizando os mesmos parâmetros de configuração. Todas as implementações foram executadas em um mesmo ambiente computacional (Pentium Core 2 Duo 2.0 GHz com 5 Gbyte de memória RAM) utilizando uma única *thread* do processador.

Dentre as possibilidades de instâncias que poderiam ser utilizadas para analisar o desempenho dos algoritmos, foram escolhidos os seguintes tipos de grafos, listados a seguir, considerando 10 replicações em cada um dos conjuntos, totalizando 930 diferentes instâncias de grafos.

– DENSOS: 30 grafos gerados aleatoriamente cujo número de vértices varia uniformemente no intervalo $n \in [100, 10000]$ com uma probabilidade de conexão entre vértices definida em 99%. Estes grafos fazem parte do conjunto de instâncias de testes por apresentarem um alto valor para Δ, e consequentemente são os que demandam maior

tempo computacional para serem coloridos. Nesse cenário, foram analisados 300 grafos.

– GRANDES: 33 grafos gerados aleatoriamente, cujo número de vértices é fixado em n=10000 e sua densidade varia uniformemente entre 1% e 99%. Com este conjunto de instâncias, pretende-se analisar a influência da variação de Δ sobre o tempo computacional das implementações estudadas. Nesse cenário, foram analisados 330 grafos.

– REGULARES: 30 grafos $k - regulares$ gerados aleatoriamente com o número fixo de vértices em n=10000 e o valor de k variando uniformemente entre 50 e 1500. Com este conjunto de instâncias, pretende-se determinar se a alta simetria presente nos grafos $k - regulares$ pode influenciar nos tempos computacionais. Nesse cenário, foram analisados 300 grafos.

Acredita-se que as instâncias escolhidas cobrem todas as características importantes a serem estudadas neste texto. Outros tipos de grafos, como grafos planares ou grafos bipartidos não foram considerados, uma vez que as referências para os melhores resultados de limites superiores de χ' e complexidades assintóticas podem ser encontrados no Capítulo 2, mostrando que já existem soluções mais eficientes para essas classes de grafos.

As instâncias de testes utilizadas neste trabalho foram geradas de forma a garantir a ausência de qualquer vértice universal. Como mostrado na Seção 3.4, uma vez que um grafo possua um vértice universal, é possível colorir tal grafo, utilizado $\Delta + 1$ cores, com custo computacional igual a $\mathcal{O}(m)$. Logo, essa medida foi tomada para evitar que o pré-processamento fosse capaz de realizar toda a coloração do grafo em uma única execução.

As análises de desempenho dos algoritmos não estão restritas somente aos seus tempos de execução. Também foram avaliadas as informações a respeito do número de vezes que uma cor é atribuída a uma aresta, pois sabe se que os algoritmos de colocação aqui descritos eventualmente necessitam recolorir algumas arestas para aumentar a coloração do grafo. É desejável encontrar a cor válida para a coloração de uma aresta o mais cedo possível, evitando calcular e inverter o caminho $\alpha/\beta - path$, uma vez que, isso implica em um alto custo de processamento na obtenção de ganhos pouco significativos com relação ao número de arestas coloridas acrescidas ao grafo, pois se faz necessário realizar várias manipulações nas estruturas de dados para aumentar a coloração do grafo em apenas uma cor.

4.2 Resultados

Em um cenário ideal, um algoritmo de coloração de arestas deveria atribuir uma cor a cada aresta uma única vez, assim como é feito pelos procedimentos de pré-processamento na presença de um vértice universal. Neste cenário, a razão de coloração, definida por $\frac{\kappa}{m}$, onde κ representa o número de vezes que uma cor é atribuída a uma aresta durante a execução do algoritmo de coloração, deve ser igual a 1. Nas análises realizadas, deseja-se determinar as melhores configurações de parâmetros e decisões de implementação que levarão a um algoritmo que forneça resultados próximos ao cenário ideal.

Nos experimentos iniciais, os procedimentos de pré-processamento não foram considerados para que não interferissem nas análises de desempenho das estratégias de execução e para que fosse possível perceber com maior clareza o comportamento das estruturas de dados utilizadas. Em todos os gráficos a serem apresentados, cada ponto do gráfico representa a média de 10 execuções das diferentes implementações estudadas.

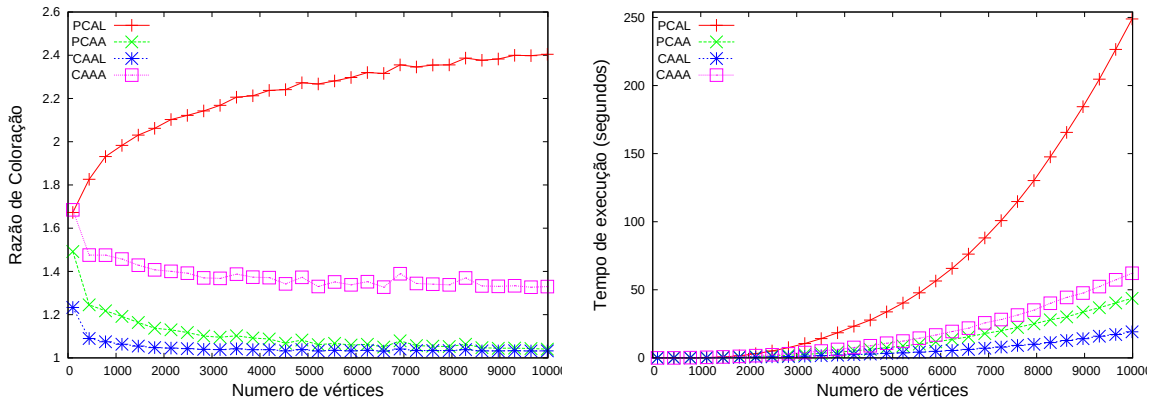


Figura 4.1. Grafos DENSOS: razão de coloração e tempos de execução.

O gráfico da Figura 4.1a, mostra a média de 10 execuções dos resultados dos valores de razão de coloração encontrados para os grafos DENSOS para cada uma das estratégias de execução apresentadas para o algoritmo proposto no Capítulo 3. Esse resultado foi obtido utilizando-se a estrutura de dados $Free(v)$ baseada na lista de inteiros para a representação de cores disponíveis. É possível observar, juntamente com a Figura 4.1b, que a razão de coloração está diretamente relacionada ao tempo de execução do algoritmo. Pode-se ver que as estratégias PCAL e CAAA são aquelas com os maiores valores de razão de coloração, ou seja, em que as arestas necessitam ser recoloridas uma maior quantidade de vezes, e as estratégias PCAA e CAAL são aquelas com os menores valores de razão de coloração e tempos de execução para grafos

densos. Nos gráficos da Figura 4.2 é possível observar que o mesmo comportamento também aparece para o conjunto GRANDES.

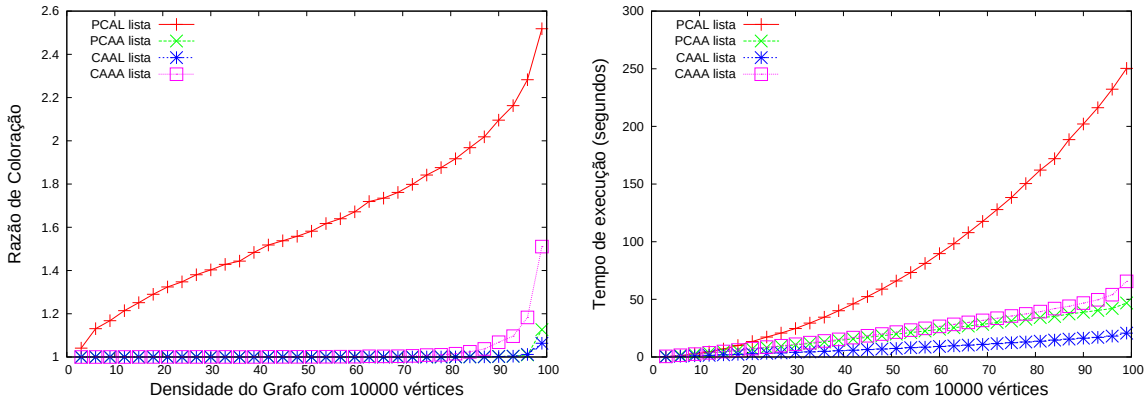


Figura 4.2. Grafos GRANDES: razão de coloração e tempos de execução.

Nos gráficos da Figura 4.3, observa-se que as estratégias CAAA e PCAA são aquelas que apresentaram os resultados menos satisfatórios em termos de razão de coloração e tempos de execução para as instâncias do conjunto REGULARES. Como pode ser visto pelos gráficos anteriormente apresentados, essas estratégias de execução apresentaram os segundo e terceiro piores resultados sobre os conjuntos de instâncias GRANDES e DENSOS.

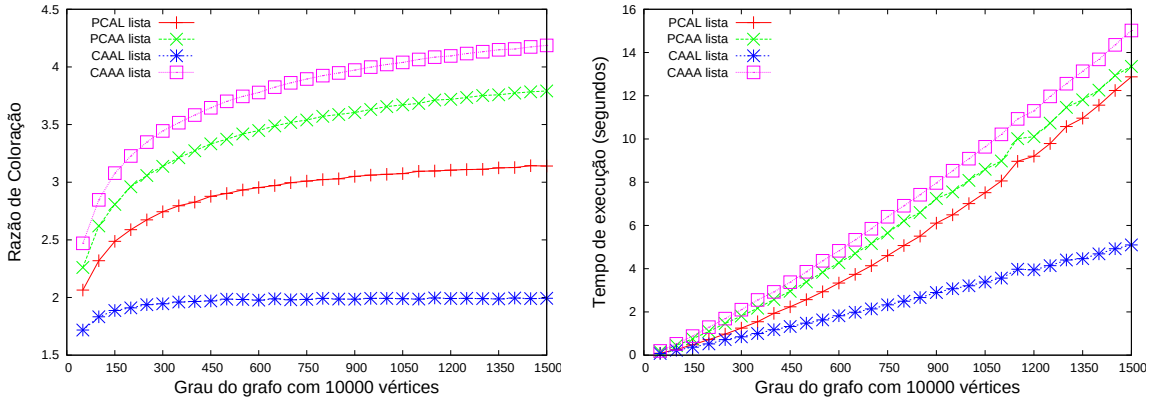


Figura 4.3. Grafos REGULARES: razão de coloração e tempos de execução.

Intuitivamente é possível concluir que, dentre as estratégias de execução adotadas, a escolha lexicográfica das arestas é aquela mais adequada a ser adotada, pois induz à coloração de todas as arestas de um vértice v_i antes de colorir as arestas incidentes no vértice v_{i+1} aumentando as chances de escolha das cores válidas mais adequadas para todas as arestas incidentes no vértice v_i com a expectativa de reduzir as possibilidades de recolorir estas arestas em um momento futuro.

Escolher prioritariamente uma cor α no momento de seleção das cores aumenta as chances de conflitos com as arestas já coloridas com esta cor. De fato, a ordem em que as cores e as arestas são selecionadas no momento da execução do algoritmo pode influenciar no seu desempenho. O uso da estratégia de busca aleatória por uma cor na interseção das cores disponíveis em dois vértices permitiu acelerar esta etapa do algoritmo.

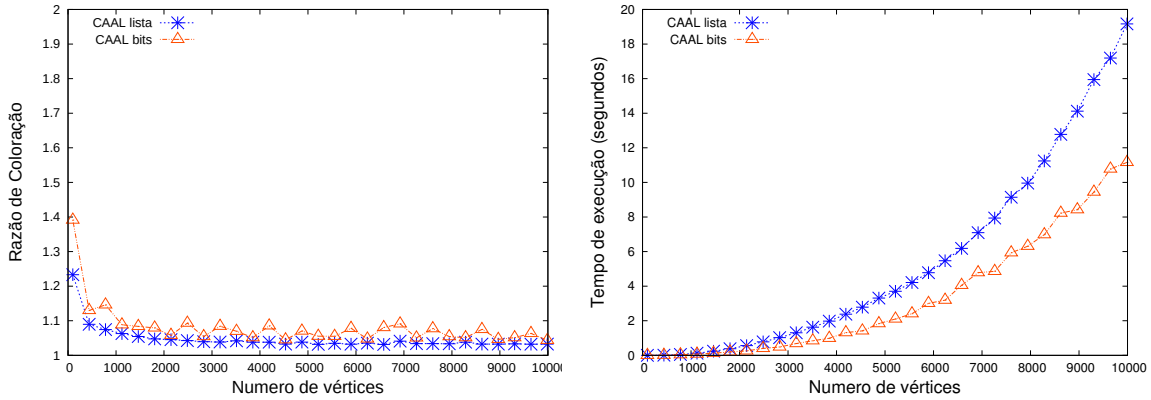


Figura 4.4. Grafos DENSOS: comparando as estruturas de dados que representam o conjunto $Free(v)$, para analisar a razão de coloração e tempos de execução.

O fato de a estratégia CAAL reduzir os valores de razão de coloração dos algoritmos é reflexo da grande taxa de acerto na escolha das cores atribuídas nas arestas, o que implica a redução do número de execuções da construção do caminho $\alpha/\beta - path$ (principalmente para o algoritmo proposto[1]) e da construção do fan (para o algoritmo tradicional).

É importante destacar que até o momento, a estratégia CAAL foi aquela que obteve os melhores resultados para todos os conjuntos de instâncias avaliados, em termos de razão de coloração e tempos de execução e, por esse motivo, será escolhida como a estratégia de execução base nos próximos resultados experimentais a serem apresentados.

Através dos gráficos das Figuras 4.4, 4.5 e 4.6 é possível perceber a clara redução no tempo de execução dos algoritmos com o uso da estrutura de dados em *bits*, embora os valores de razão de coloração obtidos com as duas estruturas de dados sejam muito similares. A implementação baseada em vetores de *palavras* é mais complexa em determinar uma cor disponível em um dado vértice que a implementação baseada em lista de inteiros, no entanto é mais eficiente em determinar uma cor na interseção dos conjuntos de cores disponíveis em dois vértices adjacentes. Uma vez que essa última

[1]Como visto na descrição do algoritmo proposto apresentada no Capítulo 3, nem toda a alternação do caminho $\alpha/\beta - path$ aumenta a coloração do grafo G.

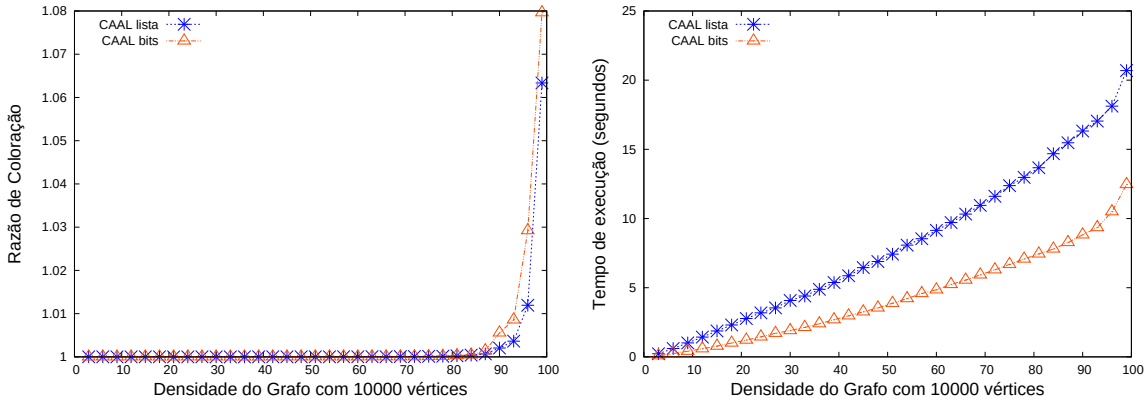


Figura 4.5. Grafos GRANDES: comparando as estruturas de dados que representam o conjunto $Free(v)$, para analisar a razão de coloração e tempos de execução.

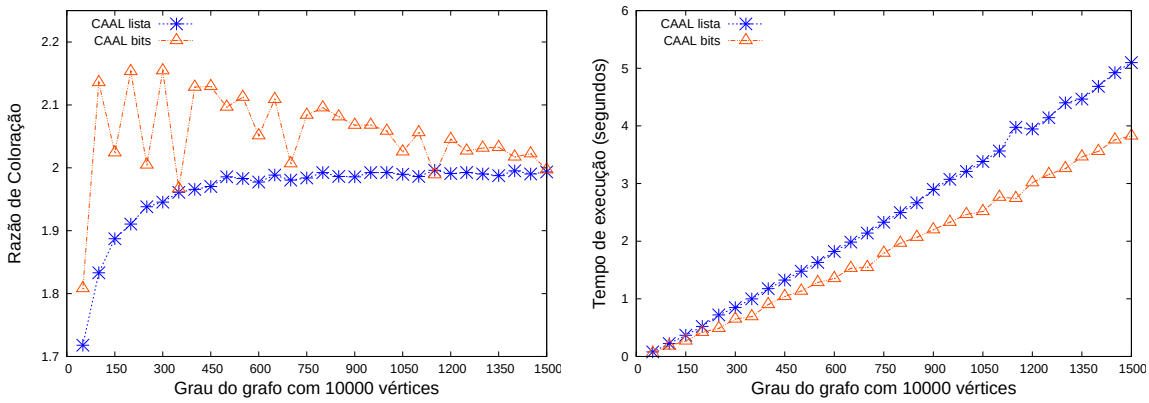


Figura 4.6. Grafos REGULARES: comparando as estruturas de dados que representam o conjunto $Free(v)$, para analisar a razão de coloração e tempos de execução.

operação é realizada com uma frequência muito maior que a primeira, o algoritmo é executado muito mais rápido na implementação que utiliza vetores de *palavras* para armazenar as cores disponíveis não incidentes em um vértice.

Considerando os resultados apresentados, a melhor versão do algoritmo proposto em termos de tempo de execução foi obtida considerando a escolha das arestas em ordem lexicográfica e cores aleatoriamente utilizando vetores de *palavras* para representar as cores disponíveis em um vértice (CAAL *bits*). Este resultado era esperado uma vez que esta versão do algoritmo utiliza a estratégia de execução que realiza o menor número de trocas de cores nas arestas do grafo, como visto através dos gráficos de razão de coloração, juntamente com a estrutura de dados que realiza a busca por uma cor disponível em dois conjuntos de cores disponíveis ($\alpha \in Free(v) \cap Free(w)$) da forma mais eficiente.

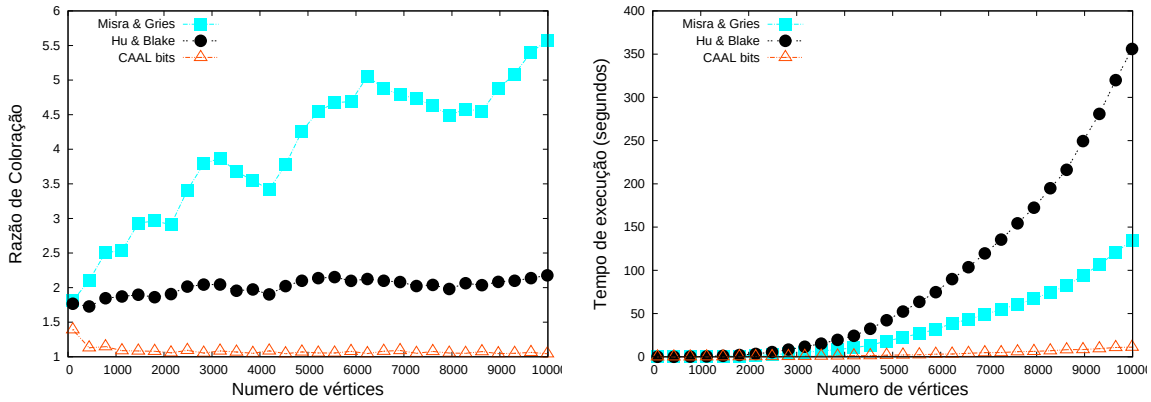


Figura 4.7. Grafos DENSOS: comparação com os algoritmos disponíveis na literatura.

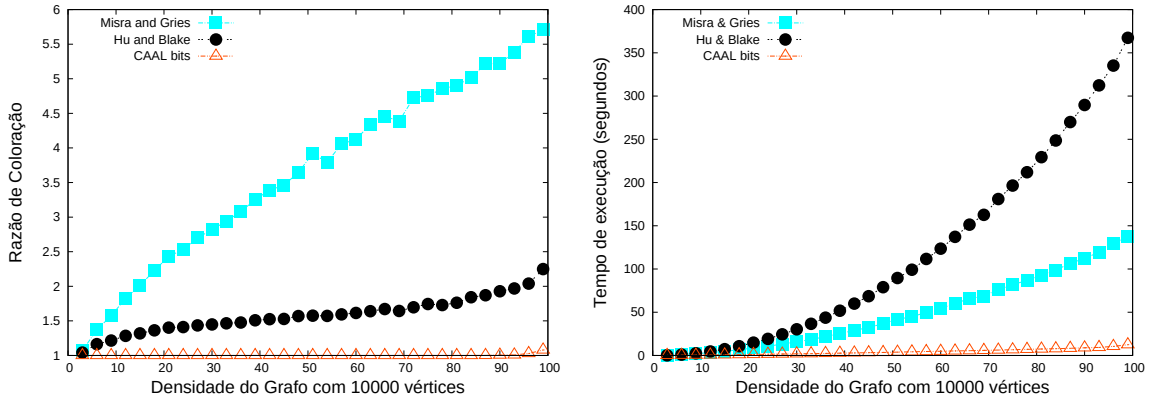


Figura 4.8. Grafos GRANDES: comparação com os algoritmos disponíveis na literatura.

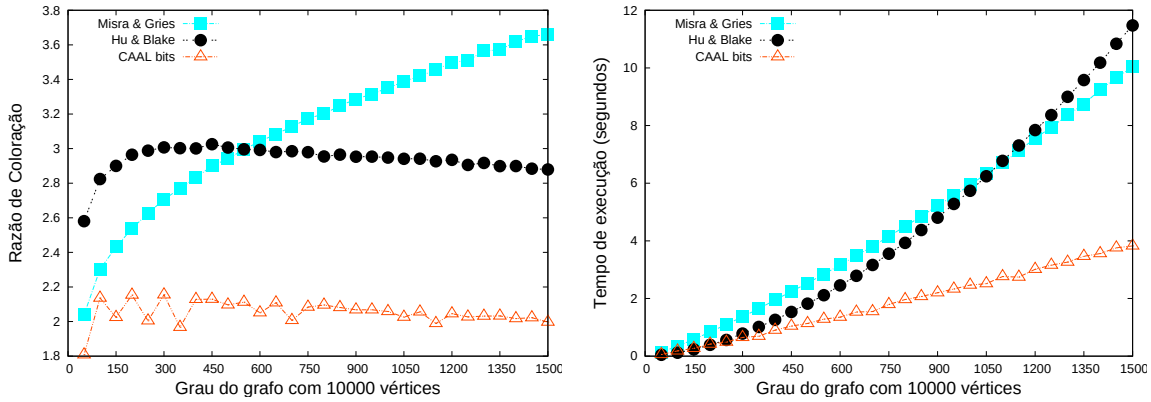


Figura 4.9. Grafos REGULARES: comparação com os algoritmos disponíveis na literatura.

Os gráficos das Figuras 4.7, 4.8 and 4.9 apresentam as comparações da implementação CAAL *bits* com os algoritmos UTA [2010b] inspirado em Misra & Gries [1992], e UTA [2010a] inspirado em Hu & Blake [1997], disponíveis na literatura, considerando a razão de coloração e tempos de execução. Para todas as instâncias consideradas é possível notar com clareza que a implementação CAAL *bits* supera as implementações disponíveis na literatura.

Nas legendas dos próximos gráficos, o termo "tradicional" será utilizado para referenciar a implementação do algoritmo de coloração de arestas baseada na prova do Teorema de Vizing disponível em Nakano et al. [1995], que utiliza a estrutura fan de forma explícita na demonstração introduzida no Capítulo 2. O termo "proposto" (ou "proposta") será utilizado para representar a implementação do algoritmo de coloração de arestas baseada na prova do Teorema de Vizing disponível em Gould [1988], como apresentada no Capítulo 3. Nas duas implementações, foi utilizada a estratégia de execução CAAL e a implementação da estrutura de dados $Free(v)$ utilizando a representação das cores disponíveis em vetores de *palavras*.

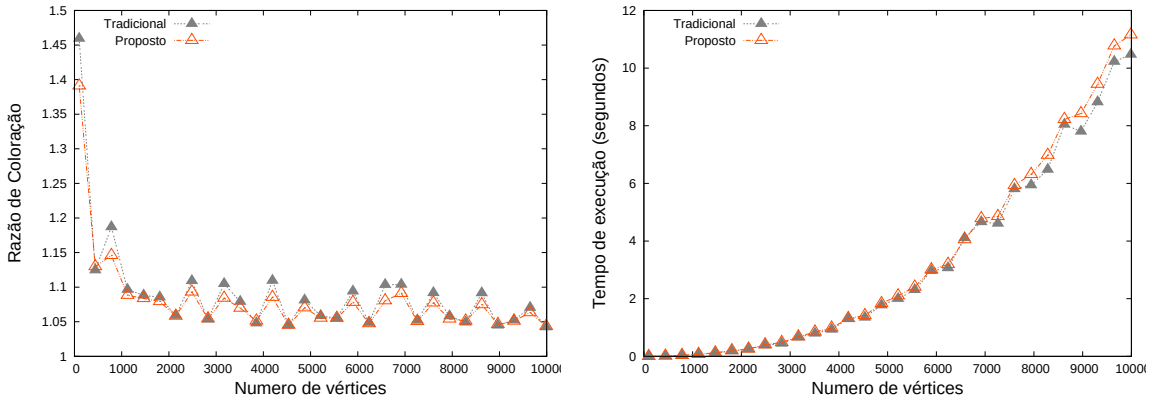


Figura 4.10. Grafos DENSOS: comparação entre as implementações básicas dos algoritmos estudados.

Os gráficos das Figuras 4.10, 4.11 and 4.12 apresentam as comparações para a razão de coloração e tempos de execução entre a implementação tradicional e a implementação proposta, para os grafos dos conjuntos DENSOS, GRANDES e REGULARES, respectivamente. É importante ressaltar que a implementação tradicional possui complexidade assintótica $\mathcal{O}(m \cdot n)$ enquanto a implementação proposta possui complexidade assintótica $\mathcal{O}(m \cdot n \cdot \Delta)$, no entanto os resultados experimentais mostram que ambas implementações apresentam comportamentos similares para as instâncias de grafos do conjunto DENSOS e que as diferenças de tempo de execução e razão de coloração que deveriam ser diferentes, com base nos resultados teóricos, não se confirmam na prática, já que o algoritmo tradicional é, na média, apenas um pouco mais

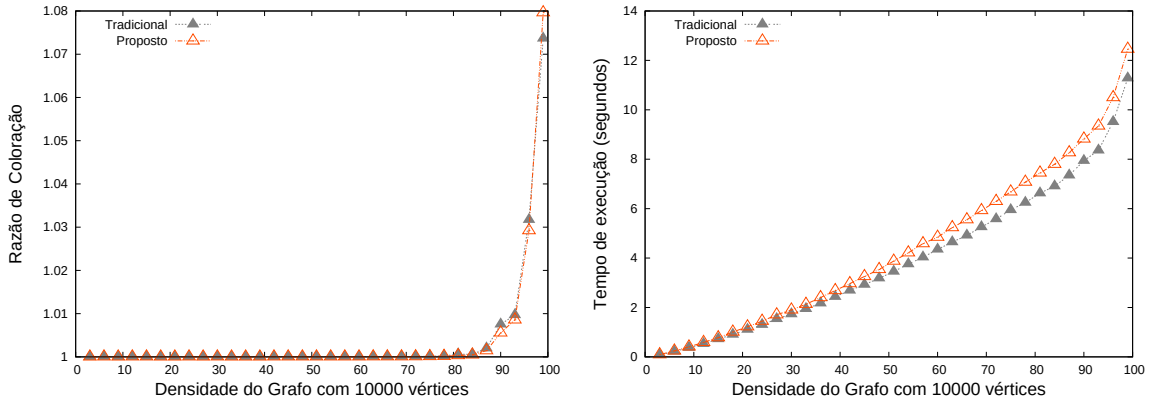


Figura 4.11. Grafos GRANDES: comparação entre as implementações básicas dos algoritmos estudados.

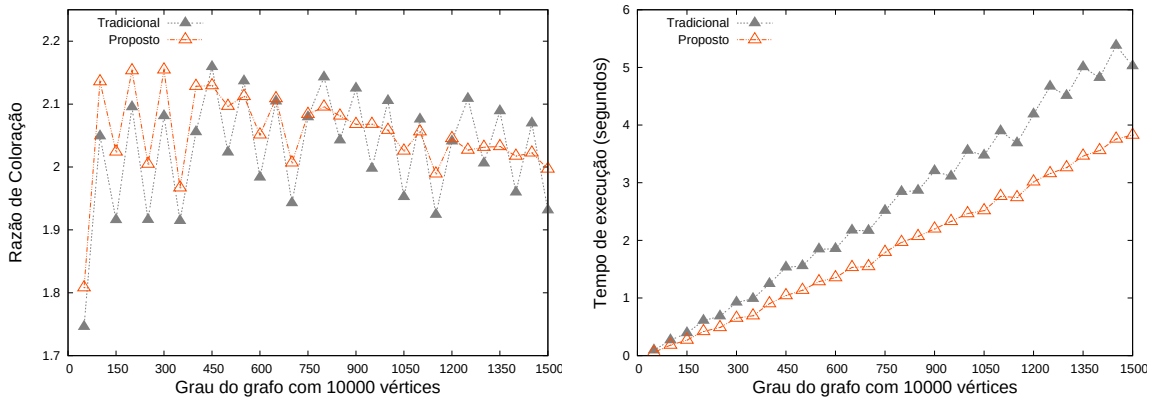


Figura 4.12. Grafos REGULARES: comparação entre as implementações básicas dos algoritmos estudados.

rápido. Para os grafos do conjunto GRANDES, observa-se que o tempo de execução da implementação baseada no algoritmo tradicional é ligeiramente menor que a implementação proposta. Os resultados obtidos para os grafos do conjunto REGULARES mostram que a implementação proposta supera, em termos de tempos de execução, a implementação baseada no algoritmo tradicional.

Dois métodos de pré-processamento foram aplicados aos algoritmos tradicional e proposto. Os gráficos das Figuras 4.13, 4.14 e 4.15 mostram, para cada conjunto de instâncias, o porcentual de redução do tempo de execução dos algoritmos e os tempos de execução acumulados que consideram a execução do pré-processamento *a priori* juntamente com a execução dos algoritmos.

Os gráficos das Figuras 4.16, 4.17 e 4.18 mostram, para cada conjunto de instâncias, o porcentual de redução do tempo de execução dos algoritmos, tradicional e proposto, e o tempo de execução acumulado do pré-processamento *embutido* e dos algoritmos.

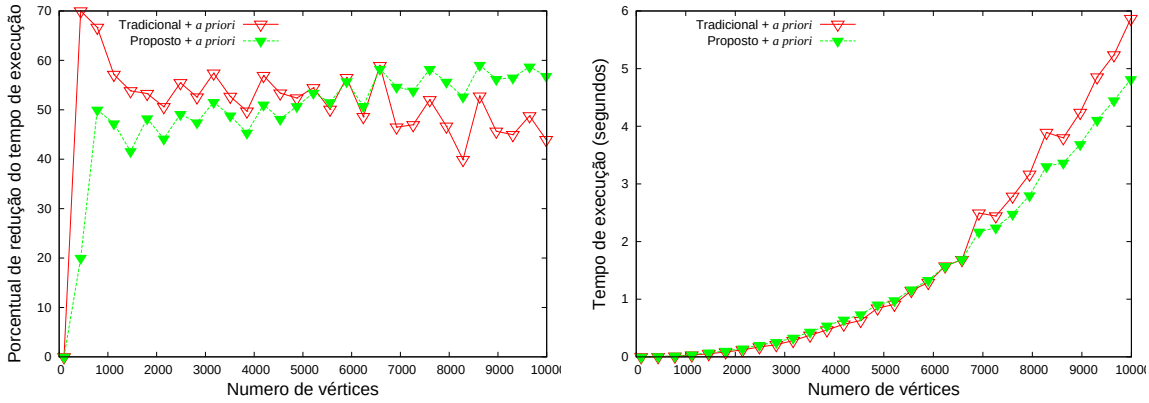


Figura 4.13. Grafos DENSOS: novos resultados obtidos com o auxílio da estratégia de pré-processamento *a priori*.

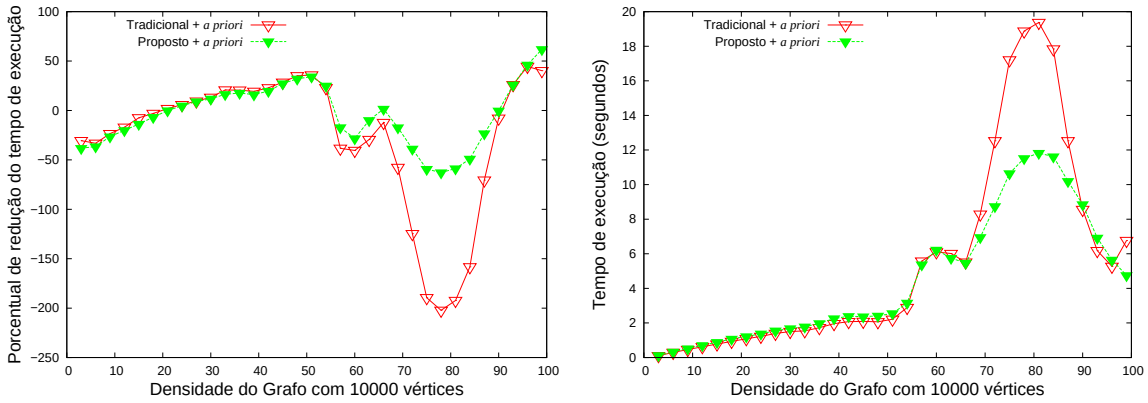


Figura 4.14. Grafos GRANDES: novos resultados obtidos com o auxílio da estratégia de pré-processamento *a priori*.

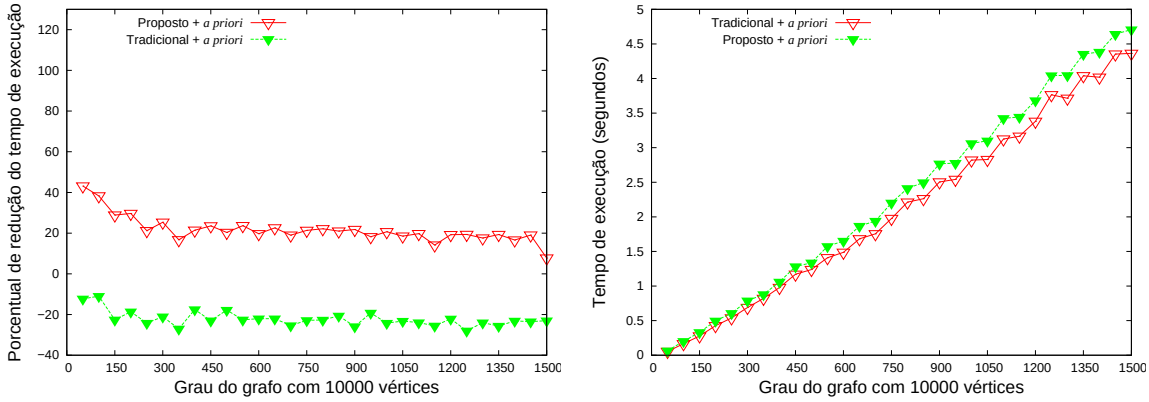


Figura 4.15. Grafos REGULARES: novos resultados obtidos com o auxílio da estratégia de pré-processamento *a priori*.

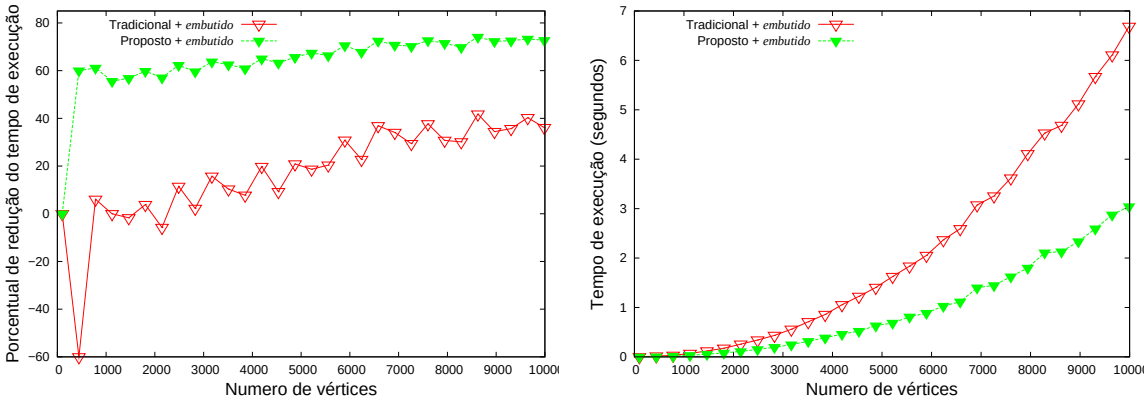


Figura 4.16. Grafos DENSOS: novos resultados obtidos com o auxílio do pré-processamento *embutido*.

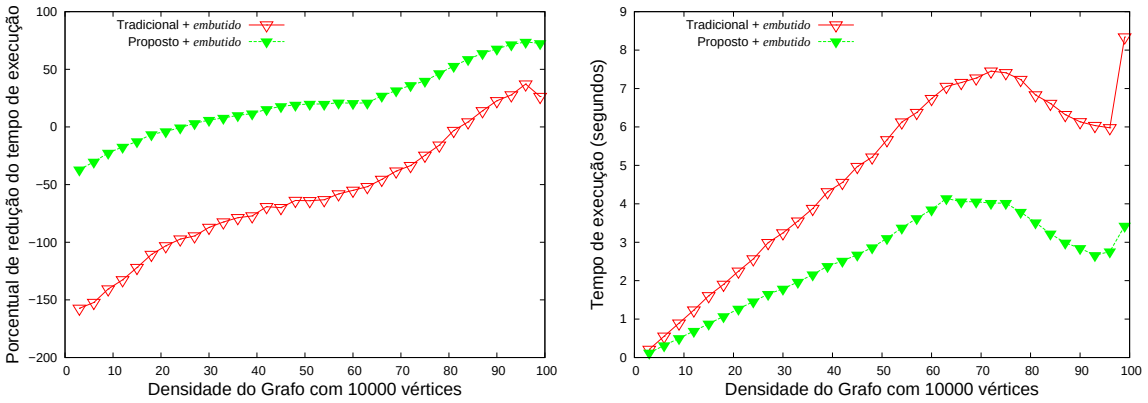


Figura 4.17. Grafos GRANDES: novos resultados obtidos com o auxílio do pré-processamento *embutido*.

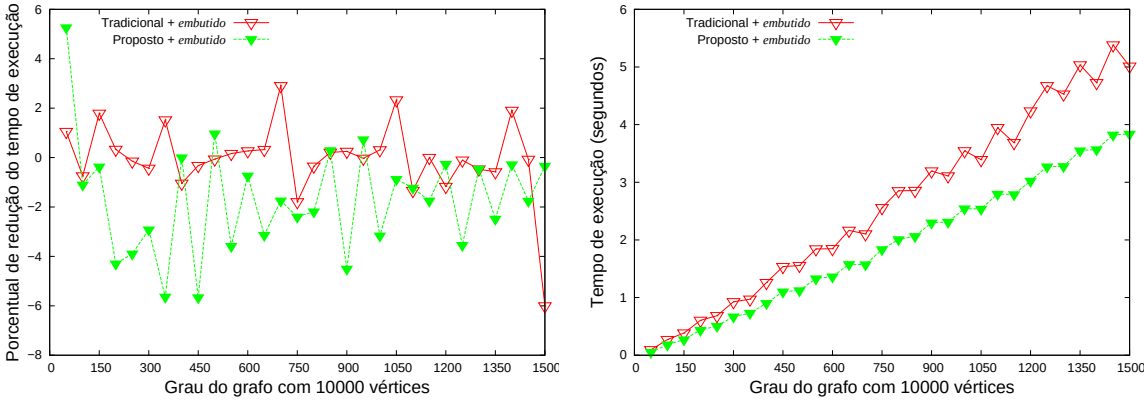


Figura 4.18. Grafos REGULARES: novos resultados obtidos com o auxílio do pré-processamento *embutido*.

Tabela 4.1. Valores das médias de redução no tempo de execução obtidos pelas estratégias de pré-processamento para os algoritmos estudados.

Instâncias de teste	Proposto		Tradicional	
	a priori	*embutido*	*a priori*	*embutido*
DENSOS	**49,18%**	**63,92%**	**50,63%**	**17,32%**
GRANDES	**-4,77%**	**21,34%**	**-26,97%**	**-59,37%**
REGULARES	**-22,16%**	**-1,70%**	**21,67%**	**-0,04%**
Médias	**7,41%**	**27,85%**	**15,11%**	**-14,03%**

As médias dos porcentuais de redução do tempo de execução obtidos pelas estratégias de pré-processamento em cada algoritmo podem ser visto pela Tabela 4.1. Com o auxílio dessa tabela é possível observar que, através dos resultados obtidos, todos os procedimentos de pré-processamento apresentaram os melhores desempenhos para os grafos do conjunto DENSOS, onde os valores de Δ estão próximos a n. Para a implementação do algoritmo proposto, os melhores resultados foram alcançados com o auxílio do pré-processamento *embutido*. Para a implementação tradicional, os melhores resultados foram alcançados com o auxílio do pré-processamento *a priori*. Considerando qualquer uma das estratégias de pré-processamento é possível ver, através dos gráficos das Figuras 4.13 e 4.16, que os melhores tempos de execução foram obtidos pela combinação dessas estratégias de pré-processamento com a implementação baseada no algoritmo proposto.

Considerando apenas os grafos do conjunto GRANDES, a Tabela 4.1 mostra que a estratégia de pré-processamento *embutido* foi aquela que efetivamente contribuiu na redução do tempo de execução da implementação baseada no algoritmo de coloração de arestas proposto. Considerando os demais cenários que envolvem os grafos do conjunto GRANDES, observa-se que os tempos de execução aumentaram com o uso das estratégias de pré-processamento. Esse fenômeno pode ser explicado pelo fato de que algumas das cores que foram atribuídas às arestas, com a utilização dos procedimentos de pré-processamento, podem ter sido temporariamente inadequadas, isso fez com que o algoritmo necessitasse trocar a coloração de certas arestas, previamente coloridas, para satisfazer todos os vértices do grafo. Considerando qualquer uma das estratégias de pré-processamento é possível ver, através dos gráficos das Figuras 4.14 e 4.17, que os melhores tempos de execução foram obtidos utilizando a combinação da estratégia de pré-processamento *embutido* com a implementação baseada no algoritmo proposto.

O pré-processamento pode não gerar bons resultados para a coloração inicial do grafo quando o valor de Δ é muito menor que n, onde as chances de conflitos no

momento de determinação de uma cor durante o pré-processamento são muito altas, o que justifica os cenários que apresentaram baixo desempenho dos procedimentos de pré-processamento nesse conjunto de grafos.

Através dos resultados ilustrados na Tabela 4.1, com relação aos grafos do conjunto REGULARES, é possível ver que a estratégia de pré-processamento *a priori* foi aquela que apresentou-se mais adequada para ser aplicada em conjunto com a implementação tradicional, enquanto que o pré-processamento *embutido*, foi o mais adequando para ser aplicado em conjunto com a implementação baseada no algoritmo proposto. Esses resultados são confirmados pelos gráficos da Figura 4.15, onde a implementação tradicional apresenta os melhores tempos de execução, e pelos gráficos da Figura 4.18, onde a implementação baseada no algoritmo proposto apresenta os melhores tempos de execução.

Com base nos resultados aqui apresentados, é possível realizar as seguintes afirmações:

1. as estratégias de execução propostas exerceram um papel fundamental no desempenho dos algoritmos de coloração de arestas, visto que, a escolha apropriada na ordem em que as arestas e as cores são processadas reflete nos tempos de execução dos algoritmos. Com destaque, tem-se a estratégia CAAL, que promove a escolha das cores aleatoriamente e a escolha das arestas em ordem lexicográfica;

2. o uso de vetores de *palavras*, na implementação da estrutura de dados $Free(v)$, ajudou a reduzir o tempo computacional para determinar uma cor na interseção das cores disponíveis em dois vértices durante a execução dos algoritmos estudados;

3. o uso dos procedimentos de pré-processamento produziram, na média, grandes reduções nos tempos computacionais dos algoritmos estudados, como visto através da Tabela 4.1. Em especial, enfatiza-se a relevância do pré-processamento *embutido*, por destacar-se como a estratégia mais apropriada para a obtenção de melhores ganhos em termos de redução no tempo de execução em conjunto com a implementação do algoritmo de coloração de arestas proposto, assim como o uso do pré-processamento *a priori* em conjunto com a implementação tradicional.

Os desempenhos dos algoritmos de coloração de arestas podem ser influenciados pelas características do grafo em que são aplicados. Para os grafos do conjunto DENSOS, a implementação do algoritmo proposto, com o auxílio do pré-processamento embutido, foi aquela que, de acordo com o gráfico da Figura 4.16b, obteve os menores tempos de execução.

De acordo com o gráfico da Figura 4.17b, a implementação do algoritmo proposto, com o auxílio do pré-processamento embutido, sempre apresenta os menores tempos de execução para os grafos do conjunto GRANDES, fazendo com que esse algoritmo seja escolhido como o mais adequado a ser aplicado nesse conjunto de grafos.

Com relação ao grafos do conjunto REGULARES, os resultados apresentaram diferenças expressivas. Mesmo que o uso do pré-processamento embutido seja o mais indicado para ser utilizado em conjunto com a implementação baseada no algoritmo proposto, esse não foi capaz de reduzir o tempo de execução do algoritmo de coloração de arestas para esse conjunto de grafos. O pré-processamento *a priori*, por sua vez, é o mais indicado para ser utilizado em conjunto com a implementação baseada no algoritmo tradicional nos grafos do conjunto REGULARES.

Capítulo 5

Conclusão

Neste trabalho, foram apresentadas diversas contribuições para o Problema de Coloração de Arestas utilizando não mais que $\Delta + 1$ cores em grafos simples, tais como definições das estratégias para a ordem de processamento das arestas e das cores, estratégias de pré-processamento e estruturas de dados para representação das cores disponíveis em um vértice. Duas implementações eficientes em termos de tempo de execução e espaço, baseadas no Teorema de Vizing, foram estudadas e analisadas empiricamente. Diferentes estruturas de dados foram propostas para representar o conjunto de cores disponíveis em um vértice do grafo. Estratégias de pré-processamento foram desenvolvidas com o objetivo de fornecer um grafo parcialmente colorido como entrada para o algoritmo de coloração. Tais técnicas de pré-processamento podem ser utilizadas por qualquer algoritmo de coloração de arestas.

Através dos resultados obtidos, foi possível perceber que a seleção das cores em ordem aleatória e das arestas em ordem lexicográfica (CAAL) mostrou-se mais adequada para os algoritmos estudados. Tal estratégia foi a que mais contribuiu para reduzir os valores de razão de coloração dos algoritmos de coloração de arestas, fato refletido na grande taxa de acerto na escolha das cores atribuídas às arestas.

O pré-processamento *embutido* foi aquele que mais contribuiu para o aumento do desempenho do algoritmo proposto, pois é o que menos interfere na estratégia CAAL, quando comparado com o pré-processamento *a priori*, para o mesmo algoritmo. Por outro lado, a estratégia de pré-processamento *a priori* contribuiu de forma mais efetiva na redução dos tempos computacionais obtidos pela implementação do algoritmo tradicional.

Na estratégia *embutido*, todas as arestas incidentes em um vértice v_i são consideradas antes das arestas incidentes no vértice v_{i+1}, de tal forma que as chances do algoritmo principal de recolorir as arestas incidentes no vértice v_i em alguma iteração

futura do algoritmo sejam minimizadas. Na estratégia *a priori*, as arestas descoloridas restantes no grafo parcialmente colorido são facilmente coloridas pelo algoritmo tradicional. Caso a cor de uma determinada aresta não seja obtida através da verificação das cores disponíveis nos vértices que são conectados por essa aresta, espera-se que seja possível construir e rotacionar um fan em um grafo parcialmente colorido com um menor número de passos nesse novo grafo.

Dado que algumas arestas incidentes em um vértice v_i podem ser deixadas descoloridas em caso de falha do pré-processamento, a construção da estrutura fan pode ser realizada de forma mais rápida, necessitando um número menor de arestas para construí-la.

A representação das cores disponíveis em um vértice com o uso de vetores de *palavras* permitiu obter uma significativa redução no tempo computacional com relação à implementação baseada em lista de números inteiros na etapa em que o algoritmo precisa encontrar uma cor na interseção de cores disponíveis em um par de vértices para colorir a aresta que os conecta. Como observado nos resultados computacionais, estas contribuições apresentaram grande impacto no desempenho dos algoritmos.

Como trabalho futuro, sugere-se a realização de análises empíricas da implementação do algoritmo de coloração de arestas proposto por Gabow et al. [1985]. Através desta análise, pretende-se verificar se o desempenho desse algoritmo na prática corresponde ao resultado teórico que o aponta como o melhor algoritmo de coloração de arestas já publicado[1], comparando tais resultados com os aqui apresentados.

[1] Até o momento da publicação deste texto, não se sabe a existência de uma implementação de tal algoritmo disponível na literatura.

Referências Bibliográficas

Chrobak, M. & Yung, M. (1989). Fast algorithms for edge-coloring planar graphs. *J. Algorithms*, 10:35--51.

Cole, R. & Kowalik, L. (2008). New linear-time algorithms for edge-coloring planar graphs. *Algorithmica*, 50:351--368.

Diestel, R. (2005). *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg.

Dong, W. & Xu, B. (2010). Some results on acyclic edge coloring of plane graphs. *Information Processing Letters*, 110:887--892.

Dong's, Y. (2010). Stony brook project implementations. `http://www8.cs.umu.se/kurser/TDBA77/VT06/algorithms/WEBSITE/IMPLEMEN/STONY/DISTRIB/VIZING/`. [Online; acessado em 10 de abril de 2010].

Froncek, D. (2010). Scheduling a tournament. Em Joseph A. Gallian, E., editor, *Mathematics and Sports*, capítulo 17. Mathematical Association of America, Duluth, Minnesota.

Gabow, H. N.; Nishizeki, T.; Kariv, O. Leven, D. & Terada, O. (1985). Algorithms for edge-coloring graphs. Relatório técnico, Tohoku University.

Gould, R. (1988). *Graph Theory*. Benjamin-Cummings.

Hakimi, S. L. & Kariv, O. (1986). A generalization of edge-coloring in graphs. *Journal of Graph Theory*, 10:139--154.

Holyer, I. (1981). The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10:718 -- 720.

Hsu, C.-C.; Liu, P.; wei Wang, D. & Wu, J.-J. (2006). Generalized edge coloring for channel assignment in wireless networks. *International Conference on Parallel Processing*, pp. 82--92.

Hu, Y. F. & Blake, R. J. (1997). Algorithms for scheduling with applications to parallel computing. *Adv. Eng. Softw.*, 28:563--572.

Januario, T. (2011). Implementação e análise de algoritmos para coloração de arestas. Dissertação de mestrado, Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais, Brasil.

Kirkman, T. P. (1847). On a problem in combinatorics. *Cambridge and Dublin Mathematical Journal*, 2:191--201.

Liang, W.; Shen, X. & Hu, Q. (1996). Parallel algorithms for the edge-coloring and edge-coloring update problems. *J. Parallel Distrib. Comput.*, 32:66--73.

Matouek, J. & Gärtner, B. (2006). *Understanding and Using Linear Programming (Universitext)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Misra, J. & Gries, D. (1992). A constructive proof of vizing's theorem. *Inf. Process. Lett.*, 41:131--133.

Nakano, S.; Zhou, X. & Nishizeki, T. (1995). Edge-coloring algorithms. Em *Computer Science Today*, volume 1000, pp. 172--183. Springer-Verlag.

Sanders, D. P. & Zhao, Y. (2001). Planar graphs of maximum degree seven are Class I. *J. Comb. Theory Ser. B*, 83:201--212.

Scheide, D. & Stiebitz, M. (2010). Vizing's coloring algorithm and the fan number. *J. Graph Theory*, 65:115--138.

Shannon, C. E. (1949). A theorem on coloring the lines of a network. *Journal of Mathematical Physics*, 28:148--151.

Sharebaf, S. R. (2009). Vertex, edge and total coloring in spider graphs. *Applied Mathematical Sciences*, 3:877 -- 881.

Shin-ichi, N.; Takao, N. & Nobuji, S. (1993). Approximation algorithms for the f-edge-coloring of multigraphs. *Transactions of the Japan Society for Industrial and Applied Mathematics*, 3:279--307. em Japonês.

Skulrattanakulchai, S. (2002). 4-edge-coloring graphs of maximum degree 3 in linear time. *Inf. Process. Lett.*, 81:191--195.

Takabatake, T. (2005). Another simple algorithm for edge-coloring bipartite graphs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E88-A:1303--1304.

UTA (2010a). Prototype vizing (degree+1) edge colorer. `http://reptar.uta.edu/NOTES4351/edgeColor4.c`. [Online; acessado em 10 de abril de 2010].

UTA (2010b). Prototype vizing (degree+1) edge colorer. `http://reptar.uta.edu/NOTES5311/misraGriesNew.c`. [Online; acessado em 10 de abril de 2010].

Vizing, V. G. (1964). On an estimated of the chromatic class of a p-graph. *Diskrete Analiz.*, 3:25 -- 30. em Russo.

West, D. B. (2001). *Introduction to Graph Theory (2nd Edition)*. Prentice Hall.